This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3024022, IEEE Transactions on Knowledge and Data Engineering

Deep Pairwise Hashing for Cold-start Recommendation

Yan Zhang, Ivor W. Tsang, Hongzhi Yin, Guowu Yang, Defu Lian, and Jingjing Li

Abstract—Recommendation efficiency and data sparsity problems have been regarded as two challenges of improving performance for online recommendation. Most of the previous related work focus on improving recommendation accuracy instead of efficiency. In this paper, we propose a Deep Pairwise Hashing (DPH) to map users and items to binary vectors in Hamming space, where a user's preference for an item can be efficiently calculated by Hamming distance, which significantly improves the efficiency of online recommendation. To alleviate data sparsity and cold-start problems, the user-item interactive information and item content information are unified to learn effective representations of items and users. Specifically, we first pre-train robust item representation from item content data by a Denoising Auto-encoder instead of other deterministic deep learning frameworks; then we finetune the entire framework by adding a pairwise loss objective with discrete constraints; moreover, DPH aims to minimize a pairwise ranking loss that is consistent with the ultimate goal of recommendation. Finally, we adopt the alternating optimization method to optimize the proposed model with discrete constraints. Extensive experiments on three different datasets show that DPH can significantly advance the state-of-the-art frameworks regarding data sparsity and item cold-start recommendation.

Index Terms—Recommender system, denosing auto-encoder, hash code, cold-start

1 Introduction

PErsonlized recommender systems have been recognized as one of the most critical and effective approaches for alleviating information overload, and also it is a key factor for the success of various applications such as online E-commerce webs sites: Amazon, Netflix, Yelp, etc., online educational systems, and even online health systems. Typically, a recommender system recommends a particular user with a small set from the underlying pool of items that the user may be interested in. Most of the existing recommendation models were based on users' previous behavior data such as ratings, purchasing records, click actions, and watching records, like/dislike records, etc. They were known as collaborative filtering (CF).

CF-based recommender systems are proven to be very successful. They produce top-k items that users may be interested in by exploiting the historical interaction data. 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-based methods factorize an user-item interaction matrix into a low-dimensional real latent 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 real latent vectors, and the user's top-k preferred items can be produced by ranking the scores descendingly, and this procedure is named online recommendation.

However, datasets in the real world are generally sparse, which leads to poor performance with CF-based recommender systems. Besides, CF-based recommender systems are incapable of handling cold-start problems [1], since new users or new items are lack of interaction data in cold-start settings. Another type of recommender system, the content-based recommendation, can offer suggestions based on the content similarities of users and items. To tackle the data sparsity and cold-start problems, several state-of-the-art content-aware recommender systems [2–5] were proposed by combining the CF-based information and content-based information. Specifically, these content-aware recommender system extracted vector representations of new users or new items from their content data, and then these vector representations were unified into a CF-based framework, and thus it can be applied to cope with data sparsity and cold-start problems.

In the previous content-aware recommender systems, they first learned real-valued representations for users and items, respectively, and then conducted a recommendation by an online similarity search procedure, online recommendation. A successful recommender system should meet the users' requirement of fast response to recommend items from a large set by analyzing their browsing, purchasing, and searching history. However, the growing scale of users and items has made the online recommendation

- *Y. Zhang, is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and the Centre for Artificial Intelligence, University of Technology Sydney, Sydney 2007, Australia. E-mail: yixianqianzy@gmail.com.*
- *I. W. Tsang is with the Centre for Artificial Intelligence, University of Technology Sydney, Sydney 2007, Australia. E-mail: Ivor.Tsang@uts.edu.au.*
- *H. Yin is with School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane 4067, Australia. E-mail: h.yin1@uq.edu.au.*
- *G. Yang and J. Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. guowu@uestc.edu.cn, lijin117@yeah.net.*
- *D. Lian is with School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China. E-mail: dove.ustc@gmail.com.*

much more challenging since the time cost with real-valued representations is expensive. Hence a few recommendation frameworks [6, 7] were proposed to speed up the online recommendation. However, the time complexity has not yet been decreased markedly since these recommendations are still based on similarity search in the real space.

To speed up online recommendation fundamentally, several scholars put forward hashing-based recommendation frameworks [8–10] that can fundamentally improve the similarity search efficiency. To be specific, the hashing-based recommendation framework encodes users and items into binary codes in a Hamming space. Then the 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 [11]. So the time cost with hashing-based frameworks is significantly reduced compared with the real-based frameworks. 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 of real-valued vectors, which significantly reduces storage cost.

Most of the previous hashing-based recommender systems focus on rating prediction, but the rating-based objectives are not consistent with the ultimate goal of the recommender system – providing a ranking list of items. Thus, a ranking-based objective should be more appropriate to formulate the recommendation task. Previous ranking-based recommendation such as Bayesian personalized ranking (BPR) [12], Cofi rank-maximum margin matrix factorization (CofiRank) [13], and List-wise learning to rank with matrix factorization (ListCF) [14], have been proposed, showing superior performance of recommendation to rating-based methods. To speed up the online recommendation, Discrete Personalized Ranking (DPR) [10], were proposed based on a pair-wise ranking objective with discrete constraints. However, they cannot cope with data sparsity and cold-start issues.

To this end, this paper proposes a ranking-based hashing framework, Deep Pairwise Hashing (DPH), to alleviate data sparsity and cold-start problems, and also provide an efficient online recommendation. We have already presented our preliminary study of solving data sparsity and cold-start problems with an efficient way in Discrete Deep Learning (DDL) [15]. This paper extendes DDL with an in-depth study and performance analysis. Specifically, this paper makes the following new contributions: first, to make the proposed recommendation framework applicable in a wide range of platforms, we design a new recommendation framework based on implicit feedbacks, since they are much more common in real world applications than explicit ratings; second, we design a ranking-based objective in the proposed DPH, that is consistent with the ultimate goal of recommendation – providing a ranked items list for each user; third, to learn item robust representation, we choose Denoising Auto-encoder (DAE) instead of Deep Belief Network (DBN) embedded in DDL; and fourth, we conduct more extensive experiments to evaluate the performance of DDL and the enhanced DPH to overcome the cold-start and sparse problems on Amazon and Yelp datasets.

The main contributions of this paper are summarized as follows:

1) We propose a pairwise ranking based hashing recommendation framework, which is capable of handling cold-start item and data sparsity issues, and providing efficient online recommendation.
2) We choose the Denoising Auto-encoder instead of other deterministic deep learning frameworks to learn robust item representation by modeling the noise, which is beneficial to improve the recommendation performance in sparse and item cold-start setting.
3) We develop an alternating optimization algorithm by adding balance and irrelevant constraints on hash codes to solve the proposed discrete problem, which is helpful to extract compact and informative hash codes.

The rest of this paper is organized as follows: Section 2 introduces two types of related work, the content-aware recommendation, and the hashing-based recommendation. Followed by the introduction of main notations appeared in this paper and the problem formulation in Section 3. Next, we present the DPH framework and explain the detailed derivation of each step in Section 4, and show the initialization and the discrete optimization algorithm of solving the proposed model in Section 4.4. Explicitly, we first initialize DPH by unsupervised learning of DAE, and the robust item representations obtained are forwarded into the supervised DPH learning procedure. To optimize DPH, we adopt an alternating optimization algorithm composed of a series of mixed integer programming problems. Then, we introduce the experimental settings, description of datasets, and competing baselines in Section 5, followed by the experimental analysis in Section 5.3. Finally, we conclude this paper and disclose several future works could be done in Section 6.

2 Related Work

In this section, we review several major schemes closely relevant to this paper, that contains the content-aware recommendation, the ranking-based recommendation, and the hashing-based recommendation. But the three types have some overlaps, so we introduce the related works from the following two aspects: we first introduce several state-of-the-art content-aware recommender systems proposed to alleviate data sparsity and cold-start problems, we then introduce the latest two competing hashing-based recommendation frameworks.

2.1 Content-aware Recommender Systems

Collaborative topic regression [2] is a state-of-the-art content-aware recommender system, which was proposed by combining the topic model, collaborative filtering, and probabilistic matrix factorization (PMF) [16]. 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: the user's collection data and the article content data. It can be used to mitigate cold start and data sparsity settings.

Another type of content-aware recommender systems is deep learning based framework [17]. Collaborative deep learning (CDL) [3] was proposed as a probabilistic model by jointly learning a probabilistic stacked denoising autoencoder (SDAE) [18] and CF. Similar to CTR, CDL exploits the interaction and content data to alleviate cold start and data sparsity problems. However, unlike CTR, CDL took advantage of deep learning framework to learn effective real latent representations. Thus, it can also be applied in the cold-start and sparse settings. CDL is also a tightly coupled method for recommender systems by developing a hierarchical Bayesian model.

Visual Bayesian Personalized Ranking [19] is a factorization model by incorporating visual features into predictors of users' preferences. By utilizing visual features extracted from product images by (*pre-trained*) deep networks, VBPR is also helpful to alleviate cold start and sparse issues.

2.2 Hashing-based Recommender Systems

A pioneer work [20] was proposed to exploit Locality-Sensitive Hashing [21] to generate hash codes for Google News readers based on their click history. On this basis, A. Karatzoglou et al. [22] randomly projected real latent representations learned from regularized matrix factorization into hash codes. Similar to this, K. Zhou et al.[8] followed the idea of Iterative Quantization [23] 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 [24], hashing essentially only preserves similarity rather than preference based on inner product, since the magnitudes of 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 [25]. But the two-stage approach still suffered large information loss in the quantization procedure.

Hashing learning frameworks differs from two-stage hashing frameworks in solving the discrete optimization problem directly to learn binary codes. Thus more information is carried by hash codes than two-stage frameworks. Zhang et al. proposed discrete collaborative filtering (DCF) [9], 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 [12]. Then, a ranking-based hashing framework was proposed to improve the recommendation performance [10] by adding 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

3.1 Notations

Let user and item index sets are denoted by $U = \{1, \cdots, n\}$ and $I = \{1 \cdots, m\}$, respectively. $\mathbf{R} = (s_{ui})_{n \times m}$ is the observed implicit feedback matrix, thus entries in $\mathbf{R}$ contain only values '1' and '0', and $s_{ui} = 1$ denotes user u had an interaction with item i; otherwise there is no interactions between user u and item i. Examples of user-item interactions contain users' purchases, clicks, collections, etc. Note that although we evaluate DPH on implicit feedbacks, but it also works on explicit ratings by transferring explicit ratings into implicit feedbacks. The index set of observed implicit feedbacks in $\mathbf{R}$ is denoted as S, which is a subset of the Cartesian product of U and I: $S \subseteq U \times I$. Item content data is denoted as $\mathbf{C}$, which consists of bag-of-words vectors of all items. Other essential notations used in this paper are listed in Table 1.

TABLE 1
Notations

Symbol	Description
$\mathbf{B}$	binary matrix of all users in U
$\mathbf{D}$	binary matrix of all items in I
$\mathbf{b}_u$	hash code of the user u
$\mathbf{d}_i$	hash code of the item i
I_u^+	items rated by the user u
I_u^-	items not rated by the user u
U_i^+	users rated the item i
U_i^-	users not rated the item i
S_D	the 2–tuple index set of observed interactions
S_T	the 3–tuple index set of observed interactions
Θ	parameters of SDAE
$\mathbf{X}$	the delegated real matrix of $\mathbf{B}$
$\mathbf{Y}$	the delegated real matrix of $\mathbf{D}$

3.2 Denoising Auto-encoder

The principle behind denoising auto-encoders is to be able to reconstruct data from an input of corrupted data. After giving the auto-encoder the corrupted data, we force the hidden layer to learn only the more robust representations. The output will then be a more refined version of the input data [26]. Denoising Auto-encoders solve this problem by corrupting the data on purpose by randomly turning some of the input values to zero. Figure 1 is a example of DAE with $2L$ layers. In general, the percentage of input nodes which are being set to zero depends on the amount of data and input nodes we have. Commonly the hidden layer in the middle. $\mathbf{C}_L$, is the latent representation we need, and the input layer $\mathbf{C}_0$ is the corrupted version of the clean input data $\mathbf{C}$. DAE solves the following optimization problem:

$$\arg\min_{\mathbf{W},\mathbf{b}} \|\mathbf{C} - \mathbf{C}_L\|_F^2 + \delta\|\mathbf{W}\|_F^2, \tag{1}$$

where $\Theta = \{\mathbf{W}, \mathbf{b}\}$ is the parameters of $2L$-layer DAE that contains weight matrices $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_{2L-1}, \mathbf{W}_{2L}\}$ and bias vectors $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{2L-1}, \mathbf{b}_{2L}\}$, and δ is the hyper-parameter of the regularizer for generalization.

3.3 Problem Formulation

Deep Pairwise Hashing: Given a user-item interaction dataset $\mathbf{R}$, a content dataset $\mathbf{C}$, and an active user u, our goal is to recommend top-k items that user u would be interested in by a pairwise ranking function - Area under the ROC curve [27] (AUC). The problem becomes a cold-start item recommendation when the predicted top-k items

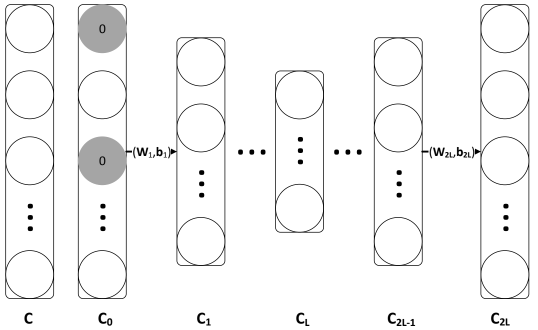


Fig. 1. $2L$-layer Denosing Auto-encoder.

are not in the interaction dataset $\mathbf{R}$, and meanwhile in the content data $\mathbf{C}$.

4 Deep Pairwise Hashing

Deep Pairwise Hashing is proposed to deal with the efficient online recommendation and data sparsity by hashing technique and a content-aware objective, and the entire framework is shown in Figure 2. It initializes item representations by a denoising auto-encoder, and then obtains hash codes from a joint training of the content-aware objective and pairwise hashing objective, and finally conduct a recommendation for a specific user based on hash codes obtained.

4.1 Pairwise Ranking Objective

In this section, we first formulate user's preference by hash codes in Hamming Space. Followed by the traditional Matrix Factorization [28], we also formulate the preference as the similarity between representations of users and items. In Real Space, similarity can be defined as different metrics, such as cosine similarity, inner product, Euclidean distance, etc. While in Hamming Space, the most common similarity is the Hamming distance. Suppose users and items are presented as r-dimension hash codes, $\mathbf{b}_u \in \{\pm 1\}^r$ and $\mathbf{d}_i \in \{\pm 1\}^r$, respectively, and the Hamming distance $\mathrm{H}(\mathbf{b}_u, \mathbf{d}_i) = \sum_{k=1}^{r} \mathbb{I}(b_{uk} \neq d_{ik})$, where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the input is true, otherwise it returns 0. Then the preference of user u for item i is denoted by a expression of Hamming distance

$$\widehat{p}_{ui} = 1 - \frac{1}{r}\mathrm{H}(\mathbf{b}_u, \mathbf{d}_i) = \frac{1}{2} + \frac{1}{2r}\mathbf{b}_u^T \mathbf{d}_i \tag{2}$$

where $\widehat{p}_{ui}$ is within the range of 0 to 1, that represents the predicted preference of the user u over the item i. From the above preference predicted equation, we observe that preference metric in the r-dimension Hamming space is consistent with the r-dimension Real space where preference is predicted by inner products of real latent vectors.

Discrete Pairwise Hashing (DPH) is devised as a pairwise framework for top-k recommendation. For a specific user u, let the hash code of user u be $\mathbf{b}_u$, and hash codes of item i and item j be $\mathbf{d}_i$ and $\mathbf{d}_j$, respectively, then the predicted pairwise preference on item i and item j are denoted as $\widehat{p}_{uij}$,

$$\widehat{p}_{uij} = \widehat{p}_{ui} - \widehat{p}_{uj}. \tag{3}$$

If $\widehat{p}_{uij} > 0$, we consider that user u prefers item i over j; otherwise, user u prefers item j over i. For implicit feedbacks, positive feedbacks refer to those '1' entries. Specifically, for the user u, the positive items are those rated by the user, and the negative ones are those not rated. We expect the predicted preference of the positive item i, $\widehat{p}_{ui}$, would be greater than the negative item j, $\widehat{p}_{uj}$, i.e., $\widehat{p}_{uij} > 0$. If we regard the pairwise ranking task as a two-class classification problem, then we choose a common metric, the Area under the ROC (AUC), as the objective. According to [12], AUC of each user u is defined as

$$\mathrm{AUC}(u) = \frac{1}{|I_u^+| \cdot |I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \mathbb{I}\left(\widehat{p}_{uij} > 0\right), \tag{4}$$

where I_u^+ and I_u^- are defined in Table 1. The value of $\mathrm{AUC}(u)$ presents the average 'correct' classification pairwise samples. The 'correct' classification means positive items have greater predicted preference value than negative items. $\mathrm{AUC}(u)$ ranges from zero to one. Two extreme cases are: $\mathrm{AUC}(u) = 1$ means all preferences are preserved, and the learned representations of users and items are perfect to predict preferences; $\mathrm{AUC}(u) = 0$ means the no preferences are preserved with the learned representations.

$$\mathrm{AUC} = \frac{1}{|U|} \sum_{u \in U} \mathrm{AUC}\left(u\right) \tag{5}$$

To simplify the following expressions, we define S as $S = \{(u,i,j) \,|u \in U, i \in I_u^+ \text{ and } j \in I_u^-\}$, and denote the coefficient $z_u = \frac{1}{|U||I_u^+||I_u^-|}$. Then we rewrite AUC as

$$\mathrm{AUC} = \sum_{(u,i,j) \in S} z_u \mathbb{I}\left(\widehat{p}_{ui} > \widehat{p}_{uj}\right) \tag{6}$$

However, the above objective is a discrete problem, thus optimizing AUC directly often leads to an NP-hard problem [29]. A feasible solution in practice is to minimize some pairwise surrogate losses

$$L = \sum_{(u,i,j) \in S} z_u \ell(\widehat{p}_{ui} - \widehat{p}_{uj}) \tag{7}$$

where $\ell : \mathbb{R} \to \mathbb{R}^+$ is a convex function such as exponential loss $\ell(t) = e^{-t}$, hinge loss $\ell(t) = \max(0, 1-t)$, logistic loss $\ell(t) = \log(1 + e^{-t})$, least square loss $\ell(t) = (1-t)^2$, etc.

As the least square loss is consistent with AUC [29] and could lead to efficient and closed forms for updating real latent vectors without sampling in an either continuous or discrete case. So we propose to use the least square loss $\ell(t) = (1-t)^2$, and to minimize the following pairwise least square loss:

$$\min \sum_{(u,i,j) \in S} z_u (1 - (\widehat{p}_{ui} - \widehat{p}_{uj}))^2, \tag{8}$$

In this paper, we are interested in mapping users and items into r-dimension binary codes for fast recommendation, where user-item similarity can be efficiently calculated via Hamming distance in the r-dimension Hamming space. If we stack users' hash codes $\mathbf{b}_u \in \{\pm 1\}^r$ and items' hash codes $\mathbf{d}_i \in \{\pm 1\}^r$ by column into matrix $\mathbf{B} \in \{\pm 1\}^{r \times n}$ and

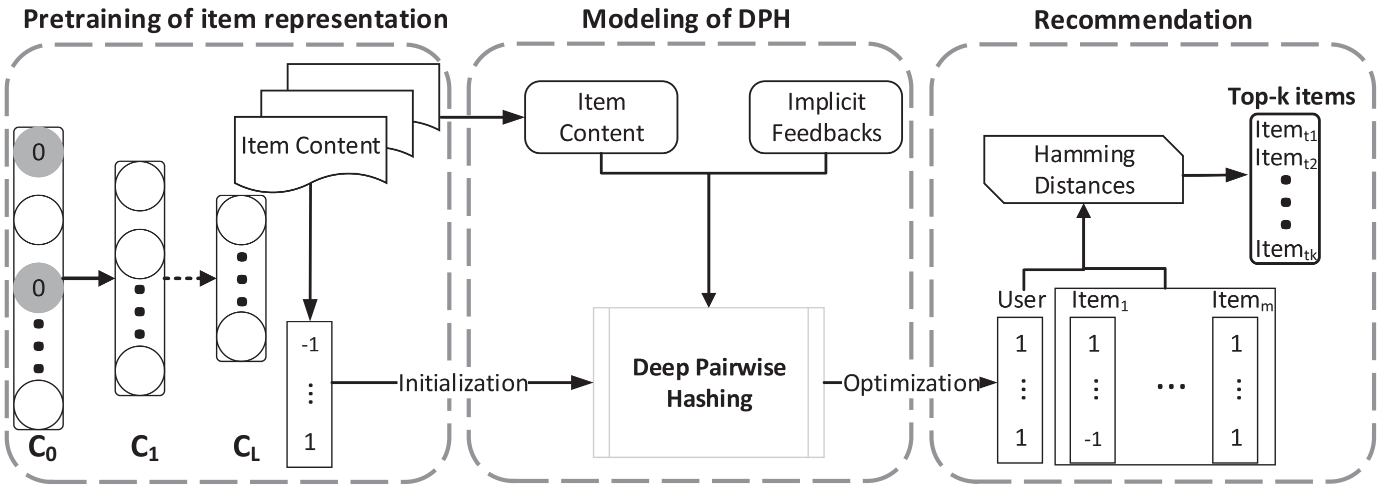


Fig. 2. The DPH framework: we first obtain items' continuous embeddings with Denoising Auto-encoder by the left pretraining step from items' content data; we then integrate the obtained items' embeddings with implicit feedbacks to formulate the proposed DPH model with a ranking-based objective; we finally develop an alternating optimization method to solve the proposed objective and conduct recommendation with the learned hash codes in Hamming space.

$\mathbf{D} \in \{\pm 1\}^{r\times m}$, respectively. Then, by substituting Equation (2) into Equation (8), we derive the Pairwise Hashing objective as follows:

$$\arg\min_{\mathbf{B},\mathbf{D}} \sum_{(u,i,j)\in S} z_u\Big(2r - \mathbf{b}_u^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 \tag{9}$$

4.2 Binary Content-aware Objective

To solve data sparsity and cold-start item problems, we incorporate content and interaction data into the proposed DPH framework. In this section, we introduce how to learn item robust representation from content data.

Deep learning models have been widely demonstrated as the most successful representation learning. Besides, data in our real world often contains some noise, and we know that DAE can be applied for extracting robust representation from Section 3.2, so we choose DAE for learning cold-start item representation in this paper.

For each item i, suppose the bag-of-words vector from item content data to be $\mathbf{c}_i$, we first extract real latent representation $\mathbf{f}_i$ from the middle layer (the L-th layer) by training a standard $2L$-layer DAE based on Equation (1) after randomly turning some of the input values to zero, and we denote the real latent representation $\mathbf{f}_i$ as

$$\mathbf{f}_i = \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta). \tag{10}$$

To derive hash codes, we directly minimize the difference between the real latent representation $\mathbf{f}_i$ and the expected hash code $\mathbf{d}_i$, and it is given by

$$\arg\min_{\mathbf{D},\,\Theta} \sum_{i=1}^{m} \|\mathbf{d}_i - \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta)\|_F^2, \tag{11}$$

where $\Theta = \{\mathbf{W}, \mathbf{b}\}$ is parameters of the $2L$-layer DAE that contains weight matrices $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_{2L}\}$ and bias vectors $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{2L}\}$. $\mathbf{D}$ is stacked by $\mathbf{d}_i$, where $i \in I$. By minimizing the objective in Equation (11), we obtain effective item hash code from content data.

4.3 Deep Pairwise Hashing

To improve recommendation accuracy and efficiency, we combine a pairwise ranking based hashing objective (9) and a content-aware objective (11) as the objective of Deep Pairwise Hashing

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\Theta} &\sum_{(u,i,j)\in S} z_u\Big(2r - \mathbf{b}_u^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 \\ &+ \lambda \sum_{i=1}^{m} \|\mathbf{d}_i - \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta)\|_F^2 \\ s.t.\;\; &\mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m},\end{aligned} \tag{12}$$

where $\lambda > 0$ is a hyper parameter that weights the importance of two objectives. To maximize the entropy of each binary bit, we add a balance constraint, so that each bit carries as much information as possible. In addition, to learn compact binary codes, we impose irrelevant constraints on hash codes, thus, we can obtain hash codes with no redundant information. Then the above problem in Equation (12) is reformulated as

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\Theta} &\sum_{(u,i,j)\in S} z_u\Big(2r - \mathbf{b}_u^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 \\ &+ \lambda \sum_{i=1}^{m} \|\mathbf{d}_i - \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta)\|_F^2 \\ s.t.\;\; &\mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m}, \\ &\mathbf{B}\mathbf{1}_n = \mathbf{0}, \mathbf{D}\mathbf{1}_m = \mathbf{0}, \mathbf{B}\mathbf{B}^T{=}n\mathbf{I}_r, \mathbf{D}\mathbf{D}^T{=}m\mathbf{I}_r,\end{aligned} \tag{13}$$

where $\mathbf{1}_n$ ($\mathbf{1}_m$) are n-dimension (m-dimension) all '1' entries vectors, and $\mathbf{I}_r$ is a $r \times r$-dimension identity matrix. As the problem (13) is a discrete optimization problem, which is proven to be an intractable NP-hard problem [30], we adopt a methodology like [9] to soften the balance and irrelevant constraints. Specifically, we add a delegated real valued $r \times n$-dimension matrix $\mathbf{X}$ and a $r \times m$-dimension $\mathbf{Y}$ to approximate hash codes $\mathbf{B}$ and $\mathbf{D}$, respectively. Thus the problem (13) can be rewritten as:

$$\begin{aligned} \underset{\mathbf{B},\mathbf{D},\Theta}{\arg\min} & \sum_{(u,i,j)\in S} z_u\Big(2d - \mathbf{b}_u^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 + \alpha \left\|\mathbf{B} - \mathbf{X}\right\|_F^2 \\ & + \lambda \sum_{i=1}^{m} \left\|\mathbf{d}_i - \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta)\right\|_F^2 + \beta \left\|\mathbf{D} - \mathbf{Y}\right\|_F^2, \\ & s.t. \quad \mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m} \\ & \mathbf{X}\mathbf{1}_n = \mathbf{0}, \mathbf{Y}\mathbf{1}_m = \mathbf{0}, \mathbf{X}\mathbf{X}^T{=}n\mathbf{I}_r, \mathbf{Y}\mathbf{Y}^T{=}m\mathbf{I}_r, \end{aligned} \tag{14}$$

where α and β are hyper parameters that allows 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 (14) can be equivalently transformed as the following mixed integer optimization problem:

$$\begin{aligned} \underset{\mathbf{B},\mathbf{D},\Theta,\mathbf{X},\mathbf{Y}}{\arg\min} & \sum_{(u,i,j)\in S} z_u\Big(2r - \mathbf{b}_u^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 - 2\alpha tr(\mathbf{B}^T\mathbf{X}) \\ & + \lambda \sum_{i=1}^{m} \left\|\mathbf{d}_i - \mathrm{DAE}(\mathbf{c}_{L,i}, \Theta)\right\|_F^2 - 2\beta tr(\mathbf{D}^T\mathbf{Y}) \\ s.t. \ & \mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m} \\ & \mathbf{X}\mathbf{1}_n = \mathbf{0}, \mathbf{Y}\mathbf{1}_m = \mathbf{0}, \mathbf{X}\mathbf{X}^T{=}n\mathbf{I}_r, \mathbf{Y}\mathbf{Y}^T{=}m\mathbf{I}_r. \end{aligned} \tag{15}$$

It's worth noting that we don't discard the binary constraints through the objective transformations, which avoids information loss caused by the quantization stage that conducted in two-stage hashing frameworks. By optimizing the above mixed integer problem, we obtain compact and informative hash codes from user-item interaction data and item content data with two constraints of balance and uncorrelation.

4.4 Model Optimization

In this section, an alternating optimization method is developed to solve the mixed integer optimization problem shown in Equation (15). We first initialize all parameters of DPH in Section 4.4.1. We then introduce how to update $\mathbf{B}$, $\mathbf{D}$, Θ, $\mathbf{X}$, and $\mathbf{Y}$, respectively in the following subsections.

4.4.1 Initialization

To learn effectiveness cold-start item representation, we first apply DAE on the corrupted bag-of-words vector to get robust real latent representation, and then we initialize item's hash code $\mathbf{d}_i = sgn(\mathrm{DAE}(\mathbf{c}_{L,i}, \Theta))$, where the function $sgn(x)$ returns 1 if $x > 0$, and -1 otherwise. We call this initialization procedure as 'Pretraining' in Figure 2. Similarly, we initialize Θ as the result of pretraining. For other parameters, we initialize them by the Gaussian randomly strategy. Specifically, we initialize $\mathbf{X}$ and $\mathbf{Y}$ by the standard Gaussian distribution with mean $\mathbf{0}$ and variance $\mathbf{I}$, independently, and initialize $\mathbf{B}$ by the sign of $\mathbf{X}$. Next, we will introduce our alternating optimization method to learn hash codes of users and items, respectively.

4.4.2 Update $\mathbf{B}$

Since the objective function in Equation (15) sums over users independently given $\mathbf{D}$, $\mathbf{X}$, $\mathbf{Y}$, and Θ, we update $\mathbf{B}$ by updating each $\mathbf{b}_u$ in parallel by minimizing the following objective function:

$$\begin{aligned} \underset{\mathbf{b}_u \in \{\pm 1\}^r}{\arg\min} \sum_{(i,j)\in S_u} z_u & \Big(((\mathbf{d}_i - \mathbf{d}_j)^T\mathbf{b}_u)^2 \\ & -4r(\mathbf{d}_i - \mathbf{d}_j)^T\mathbf{b}_u\Big) - 2\alpha \mathbf{x}_u^T\mathbf{b}_u, \end{aligned} \tag{16}$$

where $S_u = \{(i,j) | i \in I_u^+, j \in I_u^-\}$. This discrete optimization subproblem is NP-hard, and thus we adopt a bitwise learning strategy named Discrete Coordinate Descent (DCD) [31] to update $\mathbf{b}_u$. Particularly, let b_{uk} be the k-th bit of $\mathbf{b}_u$ and let $\mathbf{b}_{u\bar{k}}$ be the rest bits of $\mathbf{b}_u$, i.e, $\mathbf{b}_u = [\mathbf{b}_{u\bar{k}}^T, b_{uk}]^T$. Note that DCD can update b_{uk} given $\mathbf{b}_{u\bar{k}}$. Discarding terms independent of b_{uk}, the objective function in Equation (16) is rewritten as

$$\underset{b_{uk} \in \{\pm 1\}}{\arg\min} \ \hat{b}_{uk} b_{uk}, \tag{17}$$

where $\hat{b}_{uk} = \sum_{i,j\in S_u} z_u((\mathbf{d}_{i\bar{k}} - \mathbf{d}_{j\bar{k}})^T\mathbf{b}_{u\bar{k}}(d_{ik} - d_{jk}) - 2rd_{ik} + 2rd_{jk}) - \alpha x_{uk}$. Due to the space limit, we omit the derivation details. The above objective function reaches the minimal only if b_{uk} had the opposite sign of $\hat{b}_{uk}$. However, if $\hat{b}_{uk}$ was zero, b_{uk} would not be updated. Therefore, the update rule of b_{uk} is

$$b_{uk} = -sgn\left(K\left(\hat{b}_{uk}, b_{uk}\right)\right), \tag{18}$$

where

$$K(\hat{b}_{uk}, b_{uk}) = \begin{cases} \hat{b}_{uk}, \text{ if } \hat{b}_{uk} \neq 0; \\ -b_{uk}, \text{ otherwise.} \end{cases} \tag{19}$$

4.4.3 Update $\mathbf{D}$

Given $\mathbf{B}$, $\mathbf{X}$, $\mathbf{Y}$, and Θ, we derive the following subproblem regarding $\mathbf{d}_i$ by discarding terms irrelevant to $\mathbf{d}_i$ in Equation (15):

$$\begin{aligned} \underset{\mathbf{d}_i \in \{\pm 1\}^r}{\arg\min} & \sum_{(u,j)\in S_u^+} z_u\Big(2r - {\mathbf{b}_u}^T\left(\mathbf{d}_i - \mathbf{d}_j\right)\Big)^2 \\ & + \sum_{(u,j)\in S_u^-} z_u\Big(2r - {\mathbf{b}_u}^T\left(\mathbf{d}_j - \mathbf{d}_i\right)\Big)^2 \\ & + \lambda(\mathbf{d}_i - \mathbf{f}_i)^2 - 2\beta {\mathbf{y}_i}^T\mathbf{d}_i, \end{aligned} \tag{20}$$

where $S_i^+ = \{(u,j) | u \in U_i^+, j \in I_u^-\}$, $S_i^- = \{(u,j) | u \in U_i^-, j \in I_u^+\}$. Similarly, we optimize $\mathbf{d}_i$ by the bitwise learning, and obtain the following update rule:

$$d_{ik} = -sgn\left(K\left(\hat{d}_{ik}, d_{ik}\right)\right), \tag{21}$$

where

$$\begin{aligned} \hat{d}_{ik} = & \sum_{(u,j)\in S_u^+} z_u\left(-d_{jk} - 2rb_{uk} + \mathbf{b}_{u\bar{k}}^T\left(\mathbf{d}_{i\bar{k}} - \mathbf{d}_{j\bar{k}}\right)b_{uk}\right) \\ & - \sum_{(u,j)\in S_u^-} z_u\left(\mathbf{b}_{u\bar{k}}^T\left(\mathbf{d}_{j\bar{k}} - \mathbf{d}_{i\bar{k}}\right)b_{uk} + d_{jk} - 2rb_{uk}\right) \\ & - \lambda f_{ik} - \beta y_{ik}. \end{aligned} \tag{22}$$

4.4.4 Update Θ

Given $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$, and $\mathbf{Y}$, the optimization problem (15) is rewritten as

$$\arg\min_{\Theta} \sum_{i=1}^{m} \|\mathbf{d}_i - \text{DAE}(\mathbf{c}_{L,i}, \Theta)\|^2. \tag{23}$$

Problem (23) becomes a supervised DAE learning task. We often use stochastic gradient descent method to fine-tune parameters of DAE, where the gradient descent part is implemented by the Back-propagation algorithm. As $\mathbf{d}_i \in \{\pm 1\}^r$, we choose *tanh* function as the output function of DAE since its output is in the same range $[-1, 1]$ with hash codes. We choose sigmoid function as activation functions of hidden layers.

4.4.5 Update $\mathbf{X}$

Given $\mathbf{B}$, $\mathbf{D}$, $\mathbf{Y}$, and Θ, the Equation (15) is transformed as

$$\underset{\mathbf{X} \in \mathbb{R}^{r \times n}}{\text{argmax}}\, tr(\mathbf{B}^T\mathbf{X}),\ s.t.\ \mathbf{X}\mathbf{1}_n = \mathbf{0}, \mathbf{X}\mathbf{X}^T = n\mathbf{I}_r. \tag{24}$$

It can be solved with the help of SVD according to [9, 10]. Specifically, $\mathbf{X}$ is updated by

$$\mathbf{X} = \sqrt{n}[\mathbf{U}_s\ \widehat{\mathbf{U}}_s][\mathbf{V}_s\ \widehat{\mathbf{V}}_s]^T, \tag{25}$$

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

4.4.6 Update $\mathbf{Y}$

Given $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$, and Θ, the Equation (15) can be transformed as

$$\underset{\mathbf{Y} \in \mathbb{R}^{r \times m}}{\text{argmax}}\, tr(\mathbf{D}^T\mathbf{Y}),\ s.t.\ \mathbf{Y}\mathbf{1}_\mathbf{m} = \mathbf{0}, \mathbf{Y}\mathbf{Y}^T = m\mathbf{I}_r. \tag{26}$$

Similar to Section 4.4.5, $\mathbf{Y}$ can be updated by

$$\mathbf{Y} = \sqrt{m}[\mathbf{P}_s\ \widehat{\mathbf{P}}_s][\mathbf{Q}_s\ \widehat{\mathbf{Q}}_s]^T. \tag{27}$$

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 $\mathbf{D}$.

4.4.7 Algorithm

We integrate the above initialization and alternating optimization methods into Algorithm 1. We choose the optimal hyper-parameters α, β and λ by grid search from $[10^{-5}, 10^5]$ on validation datasets. Although, the algorithm composed of an outside loop of the alternating optimization on $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$, $\mathbf{Y}$ and Θ, and a inner loop of updating each $\mathbf{b}_u$ ($\mathbf{d}_i$). It costs about 50 times to convergence for the outside loop. For the inner loop, it costs about two or three times to convergence, so the training cost is acceptable.

Algorithm 1: Deep Pairwise Hashing (DPH)

```
Input: User-item implicit feedback $\mathbf{R}$, item conent
       data $\mathbf{C}$, DAE layer structure $[8000, 200, 30]$,
       $\alpha, \beta, \lambda$;
Output: User hash codes $\mathbf{B}$ and item hash codes $\mathbf{D}$;
Pretrain: $\Theta \leftarrow$ Equation (1);
Initialize: $\mathbf{X}, \mathbf{Y} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{B} \leftarrow sgn(\mathbf{X})$, $\mathbf{D} \leftarrow$
    $sgn(\text{DAE}(\mathbf{c}_{L,i}, \Theta))$ ;
repeat
    for $u \in \{1, \cdots, n\}$ do
        repeat
            for $k \in \{1, \cdots, r\}$ do
                update $b_{uk} \leftarrow$ Equation (18) ;
        until $\mathbf{b}_u$ convergence;
    for $i \in \{1, \cdots, m\}$ do
        repeat
            for $k \in \{1, \cdots, r\}$ do
                update $d_{ik} \leftarrow$ Equation (21) ;
        until $\mathbf{d}_i$ convergence;
    update $\mathbf{X} \leftarrow$ Equation (25) ;
    update $\mathbf{Y} \leftarrow$ Equation (27) ;
    finetune $\Theta$ by adding the objective Equation (23)
      ;
until Equation (15) convergence;
```

5 Experiments

We compare the proposed DPH with competing hashing baselines and content-aware real-valued baselines. Experiments show that DPH outperforms the competing baselines in various sparse and cold-start environments on Amazon and Yelp datasets. We also discuss the advantage of hashing based recommender systems over the real-valued frameworks, and show that hashing methods significantly improve the efficiency of online recommendation, and thus it is adaptable to the increasing items number with the E-commerce development.

5.1 Datasets

5.1.1 Data Description

To verify the effectiveness of the proposed DPH in cold-start and data sparsity settings, we choose two high sparse public datasets with the sparsity level of 99.9% (ratio of unobserved ratings to the users' size times the items' size), Amazon dataset[2] and Yelp Challenge Round 9 dataset[3]. Amazon dataset covers 142.8 million user-item interactions (ratings, review text, helpfulness votes), as well as item content (descriptions, category, etc.) on 24 product categories spanning May 1996 - July 2014. Yelp Round 9 dataset contains 4.1M reviews (ratings, reviews) and 947K tips by 1M users for 144K businesses in cites of UK and US. We separately choose two of the largest subsets: 'Clothing, Shoes & Jewelry' dataset from Amazon, and 'Phoenix' dataset from Yelp in our experiments. The detailed statistics of the two rating datasets are displayed in Table 2.

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

[3] https://www.yelp.com/dataset/challenge

TABLE 2
Statistics of datasets.

Dataset	#User	#Item	#Positive Feedbacks	Sparsity(%)
Amazon	39,387	23,033	278,653	99.97%
Yelp	122,097	11,854	353,772	99.98%

TABLE 3
Categories of baselines and DPH (the proposed model): 'Hash' means hash-based recommendation methods and 'UN-Hash' means continuous based methods.

Category	Methods	Binary	Ranking	Content
UN-Hash	CTR	×	×	✓
	CDL	×	×	✓
	VBPR	×	✓	✓
Hash	DCF	✓	×	×
	DPR	✓	✓	×
	DDL	✓	×	✓
	DPH	✓	✓	✓

5.1.2 Data Pre-processing

We first transfer explicit ratings into implicit feedbacks by setting all observed ratings to positive feedbacks '1' and all unobserved to negative feedbacks '0', and these implicit feedbacks are also called ratings 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 [32]. Finally, similar to [2], we choose top 8000 discriminative words from two datasets to form dictionaries separately by sorting the TF-IDF [33] values from high to low, then we get the bag-of-words $\mathbf{C}$ of all items.

5.1.3 Data Spliting

To verify the effectiveness of DPH in various extremely sparse and cold-start settings, we do experiments by adding another sparsity levels the above two rating datasets like [34]. For example, if we set the sparsity level 10% on the Amazon rating dataset, that means we randomly select 10% implicit ratings as training dataset, D_{train}, from the original Amazon dataset, and then the sparsity of D_{train} raised to 99.997%, and we finally test the recommendation performance of all methods on the remaining ratings, denoted as D_{test}. The random selection is carried out five times independently, and we report the experimental results as the average values.

5.2 Experimental Settings

5.2.1 Evaluation Methods

As introduced in Section 3.3, 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 performance of predicted ranking lists for users. Accuracy@k was widely adopted by many previous ranking based recommender systems [35], and MRR was also widely chosen as a metric for ranking tasks [36].

The basic idea of Accuracy@k is to test whether a user's favorite item appears in the predicted top-k items list, we regard the positive items with positive feedbacks as the user's favorite items. For each positive feedback $r_{ui} \in D_{test}$: (1) we randomly choose 1000 negative items and compute predicted rating scores for the ground-truth item i 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 i appears in the top-k ranked list, we have a 'hit'; otherwise, we have a 'miss'.

Accuracy@k has been widely used in evaluating recommendation accuracy by assessing the quality of the obtained top-k items list. Accuracy@k is formulated as:

$$\text{Accuracy@}k = \frac{\#hit@k}{|D_{test}|}, \tag{28}$$

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

The Mean Reciprocal Rank (MRR) [36] 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:

$$\text{MRR} = \frac{1}{|D_{test}|} \sum\nolimits_{r_{ui} \in D_{test}} \frac{1}{rank(u,i)}, \tag{29}$$

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

5.2.2 Competing Baselines and Experimental Settings

The proposed DPH has three key components: the ranking-based objective, the DAE framework for incorporating content information, and the application of hash technique. Intuitively, the combination of all three components will improve the recommendation performance. Thus, we do ablation studies in Section 5.3.5 to evaluate the effectiveness of each component in DPH. Besides, we choose two types of comparison methods as illustrated in Table 3, content-aware recommender systems including CTR, CDL, and VBPR, and hashing-based recommender systems including DCF, DPR, and DDL, to evaluate the effectiveness of the proposed DPH regarding cold-start and data sparse settings. In the Table 3, binary (✓) denotes binary representations and binary (×) represents continuous representations. Ranking (×) means the loss function of the method is rating-based objective. Content (×) denotes the method only use the rating information and content (✓) presents the method use both the rating and content data. In our experiments, we use cross-validation method to tune the optimal hyper-parameters for DPH and the competing baselines by grid search.

For the proposed DPH, we search the optimal hyper parameters α, β, and λ from $[10^{-5}, \cdots, 10^{5}]$, and we then set the $\alpha = 10^{-5}$, $\beta = 10^{-3}$, $\lambda = 20$. To be consistent with other baselines, we set the layer structure of DAE as $[8000, 200, 30]$.

For the content-aware baseline VBPR, we set $\lambda_{\Theta} = 10$; for CTR, we set $\lambda_u = 0.1$, $\lambda_v = 10$, $a = 1$, $b = 0.01$ and $K = 30$ since it can achieve better performance; for CDL, we set $\lambda_u = 1$, $\lambda_v = 10$, $\lambda_n = 1e^4$, $\lambda_w = 1e^{-4}$, $a = 1$, $b = 0.01$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3024022, IEEE Transactions on Knowledge and Data Engineering

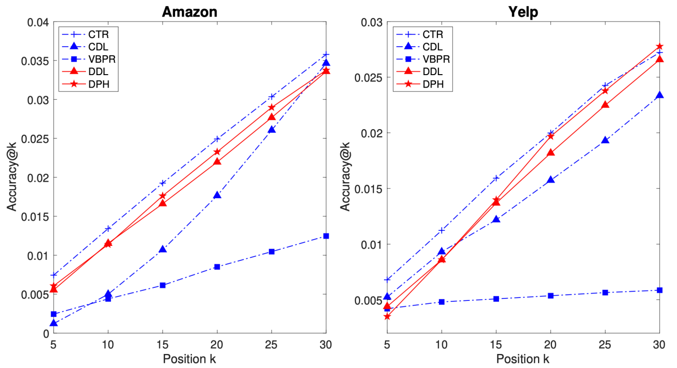


Fig. 3. Recommendation accuracy comparison in cold-start setting with sparsity level of 10%.

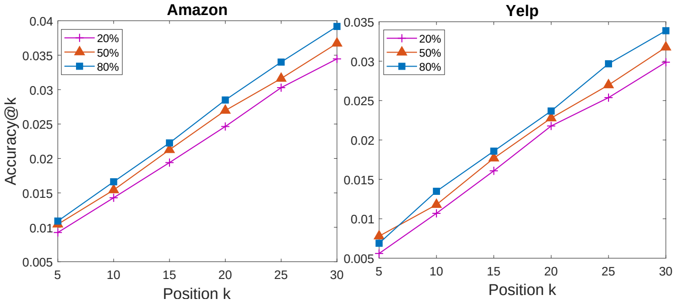


Fig. 4. Recommendation accuracy of DPH varies with sparsity levels in cold-start setting.

and set the layer structure of SDAE as $[8000, 200, 30]$ for aligning with the dimension of other methods.

For the hashing baselines DCF, we search the optimal hyper parameters α, β, and λ from $[10^{-5}, \cdots, 10^{5}]$, and set $\alpha = 10^{-3}$, $\beta = 10^{-3}$; For DPR, we set $\alpha = 10^{-4}$, $\beta = 10^{-3}$; for DDL, we set $\lambda = 5$, $\alpha = 10^{-3}$, $\beta = 10^{-3}$, and the layer structure of DBN as $[8000, 200, 30]$.

5.3 Experimental Results and Analysis

we evaluate the effectiveness of the proposed DPH in three aspects:

- the superiority over competing baselines in cold-start settings.
- the competing performance in various sparse settings (sparsity levels of 10%, 20%, 30%, and 40%),
- the high efficiency performance for online recommendation.

5.3.1 *Recommendation Accuracy in Cold-start Setting*

In this section, we compare the proposed DPH with three content-aware competing baselines and one hashing-based baseline, separately drawn as blue dashed lines and solid red lines in the following figures.

To evaluate the superior performance over other baselines in cold-start item setting, we first choose items with less than five ratings (positive feedbacks) as cold-start items, and test the performance on these cold-start items' rating data, denoted as D^{c}_{test}. To train our model, we randomly select 10% rating data as training data, D_{train}, from ratings of all users and items except for cold-start ones. We then test the performance on the simulated cold-start items' ratings D^{c}_{test}, and other remaining ratings D^{s}_{test}, and separately report the results as performances in cold-start and sparse settings.

TABLE 4
MRR in cold-start item setting (with sparsity level of 10%).

Methods	CTR	CDL	VBPR	DDL	DPH
Amazon	0.0082⋆	0.0071	0.0042	0.0077	0.0079^{o}
Yelp	0.0059	0.0022	0.0067^{o}	0.0060	0.0074⋆

We adopt Accuracy@k and MRR to respectively evaluate recommendation accuracy in the clod-start setting with the sparsity of 10%. The accuracy@k comparison of Amazon and Yelp datasets shown in Figure 3 indicates the superiority of DPH over the other hashing based baseline DDL on two datasets, as well as the competing performance compared with real-valued content-aware baselines on Amazon dataset, plus better performance than all baselines on Yelp dataset. The real-valued content-aware recommender systems can naturally achieve better accuracy than hashing based recommender systems, due to real-valued vectors intuitively carried much more information than hash codes. Because each dimension of a real-valued vector is stored with 32/64 bits, while each dimension of a hash code is saved by only one bit. That's the reason why hashing-based recommendation has inferior accuracy performance to the real-valued recommendation. But hashing-based frameworks have a significant advantage in online recommendation over real-valued methods, which will be evaluated in Section 5.3.6. Thus it is acceptable and reasonable to have small accuracy gaps between real-valued content-aware recommender systems and the proposed hashing-based DPH.

The MRR comparison displayed in Table 4 summarizes MRR results of the proposed DPH and four baselines, the best result is marked as '⋆' and the second best is marked as 'o' . We can conclude that the performance of DPH is very close to the best result, that is consistent with the result of Accuracy@k. We can also explore DDL and CDL can also perform well in cold-start setting, because we use a similar idea with the combination of deep representation learning and collaborative filtering. But we differ in designing objectives: CDL and DDL are rating based objectives, while DPH is pairwise ranking based objective, and the experiments in cold-start setting regarding the two above metrics show its superiority over other baselines.

Figure 4 reveals the recommendation accuracy in clod-start settings varies with respect to sparsity levels. As discussed at the beginning of Section 5.3.1, the recommendation performance on D^{c}_{test} is dependent on the sparsity level. Because we can learn better users' representation as well as better DAE structure (including parameters), if we apply more ratings for training. The result shown in Figure 4 also evaluate the intuitive observation on Amazon and Yelp datasets, respectively.

5.3.2 *Recommendation Accuracy in Sparse Settings*

Figure 4 reveals the recommendation accuracy in cold-start settings varies concerning sparsity levels. As discussed at

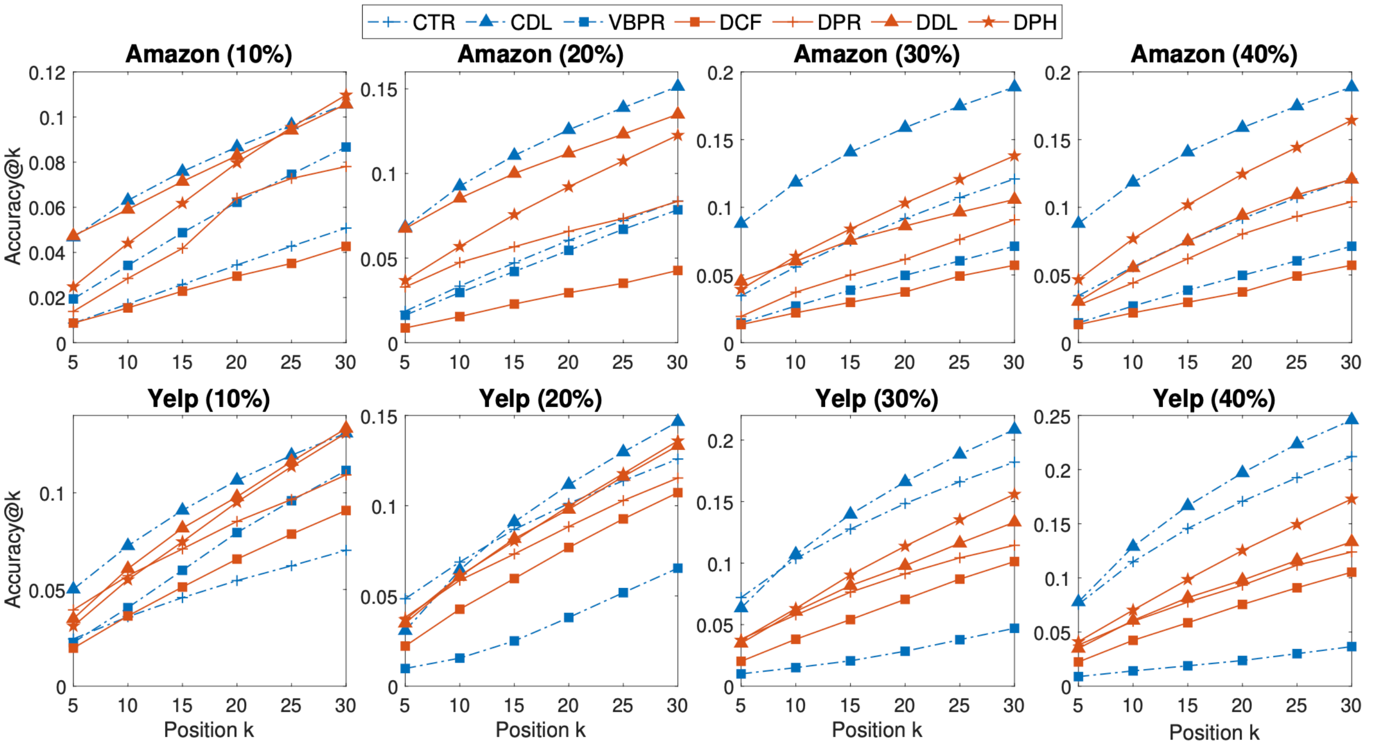


Fig. 5. Recommendation accuracy comparison on Amazon and Yelp datasets w.r.t different sparsity levels.

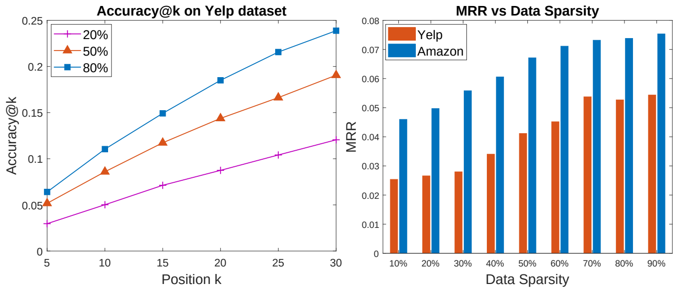


Fig. 6. Recommendation accuracy of DPH varies with sparsity levels in sparse settings.

TABLE 5
MRR w.r.t different sparsity levels (10%, 20%).

	Amazon		Yelp	
	10%	20%	10%	20%
CTR	0.0102	0.0168	0.0194	0.0355^o
CDL	0.0311	0.0481^o	$0.0360^\star$	$0.0558^\star$
VBPR	0.0176	0.0158	0.0206	0.0126
DCF	0.0102	0.0103	0.0195	0.0221
DPR	0.0200	0.0317	0.1581	0.1161
DDL	$0.0490^\star$	0.0263	0.0275	0.0151
DPH	0.0460^o	$0.050^\star$	0.0279^o	0.0296

the beginning of Section 5.3.1, the recommendation performance on D^c_{test} is dependent on the sparsity level. We can learn better users' representation as well as better DAE structure (including parameters) if we apply more ratings for training. The result shown in Figure 4 also evaluate the intuitive observation on Amazon and Yelp datasets, respectively.

5.3.3 Comparison with Hashing-based Frameworks

From Figure 5, we can see that DPH outperforms other hashing-based baselines: DCF, DPR, and DDL. By comparison of DDL, the Accuracy@k of DPH increases steadily with the increasing ratings (from 10% to 40%) for training, while the performance of DDL is not stable. From the observation for the performance of DDL, CDL, and together with DPH, we consider that the robust deep learning framework DAE applied in DPH works well on two datasets with different sparsity levels. As for DCF and DPR, they did not combine the content data into the collaborative filtering frameworks, which leads to poor performance in sparse settings.

5.3.4 Accuracy in Various Sparsity Settings

By the analysis of Figure 4 and Figure 6, we can conclude the recommendation performances in cold-start, and sparse settings are influenced by the sparsity of training data. Training with more ratings learns better users' representation and a better deep structure, which is helpful to conduct a cold-start item recommendation, and beneficial to the sparse recommendation.

5.3.5 Ablation Study

Compared with DDL, the proposed DPH adopts DAE other than DBN to integrate content embedding, and it is formulated with the ranking-based objective instead of the rating-based objective. Thus, we do some ablation studies to show the effectiveness of the above two components. Figure 7 demonstrates the performance comparison in terms of the metric Accuracy@k with DPH-R, DPH-B, DPH-B-R on Amazon and Yelp (with 30% sparsity). The only difference between DPH-R and DPH is that DPH-R is modeled as

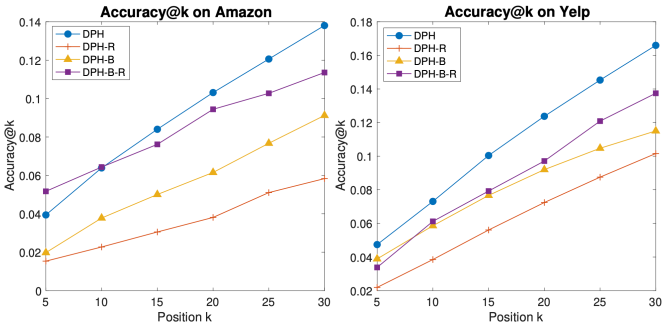


Fig. 7. Ablation studies with rating-based ('R') or ranking-based objectives, DBN ('B') or DAE for integrating content information (with 30% sparsity).

the rating-based objective; while DPH is formulated as the ranking-based objective. The only difference between DPH-B and DPH is that DPH-B integrates content information by DBN but DPH combines content data with DAE. There are two differences between DPH-B-R and DPH: DPH-B-R is formulated with the rating-based objective and it exploits content data with the DBN framework.

Figure 7 tells us that the performance of DPH is almost the best on both two datasets, and DPH-B-R has competitive performance with DPH. Actually, the DPH-B-R is DDL, and it performs better than other two baselines DPH-B and DPH-R, which tells us that the combination of the rating-based objective and the DBN framework is better than other two combinations (DPH-B and DPH-R). Because the rating-based objective model reconstructs ratings itself which is consistent with the goal of DBN framework that reconstructs each layer's entire input as accurately as impossible. DPH performs the best with the combination of the ranking-based objective and the DAE framework, because the ranking-based objective models the groundtruth of ranking which is in line with the ultimate goal of top-k recommendation. Besides, the DAE framework not only preserves the input information, but it also eliminates the effect of a corruption process stochastically applied to the input of the autoencoder, and it forces the hidden layer to discover more robust embeddings.

5.3.6 Efficiency for Online Recommendation

Compared with real-valued recommender systems, hashing-based frameworks has significant advantage of providing efficient online recommendation. We separately evaluate the efficiency regarding time and storage on synthetic data.

Time Complexity: In real-valued recommender systems, users and items are denoted by $d-$dimension real-valued vectors; while in hashing-based recommender systems, users and items are represented by $d-$dimension hash codes. For real-valued methods, the time cost of finding top$-k$ items from m items is $\mathcal{O}(md + m\log k)$ for each user u; while for hashing methods, 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.

TABLE 6
Time cost comparison on Amazon and Yelp ($\times 10^{-4}$seconds).

Methods	UN-Hash	Hash
Amazon	4.0491	1.0971
Yelp	12.5975	1.9878

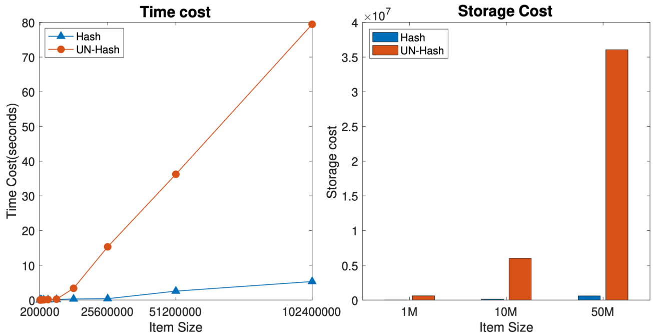


Fig. 8. Efficiency comparison between 'UN-Hash' (continuous) methods and 'Hash' methods for online recommendation on synthetic data. Left: time cost comparison. Right: storage cost comparison.

In this paper, we investigate the time cost comparison on synthetic datasets. We first use standard Gaussian distribution to generate users' and 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: 200,000, 400,000, ... , 102,400,000, to test the time cost variation of online recommendation. The variation is shown in the left of Figure 8. In addition, we evaluate the time efficiency on Amazon and Yelp datasets as well in the Table 6, where 'UN-Hash' denotes continuous based recommendations and 'Hash' represents hashing based methods. Experimental results on synthetic and real world datasets tells us that the time cost of real-valued vectors grows fast with the item number, in comparison, the time cost of hash codes increases much slower than continuous features. The experimental results show that hashing based recommendation has an evident advantage over the real-valued recommendation for online recommendation.

Storage Complexity: In the Real space, at least 64 bits are needed to store one dimension of a vector. As the dimension becomes large, it will cost much more space. While in Hamming space, only one bit is needed to store one dimension of a hash code. Thus the storage cost is reduced significantly.

We test the stdorage costs of hash codes and real-valued vectors on three different sizes of item sets: 1 million, 10 million, and 50 million. In the right of Figure 8, 'UN-Hash' represents continuous based methods and 'Hash' denotes hashing based models. It says that hash codes obtained from hashing models cost much less memory to store the same number of items.

6 Conclusion

In this paper, we propose a robust hashing-based recommendation framework deep pairwise hashing (DPH) to cope with the cold-start and sparse issues by integrating content and rating information. Specifically, to align with the ultimate goal of recommender system, we formulate the proposed framework DPH with a ranking-based objective to improve the performance. Besides, we choose Denoising Auto-encoder to extract robust items' representations from the content data. Then, we add uncorrelated and independent constraints on hash codes to learn short and informative hash codes. Furtherly, we develop an alternating optimization algorithm together with a discrete coordinate descent method to train the proposed model. Finally, we evaluate the effectiveness of the proposed DPH by metrics Accuracy@k and MRR on Amazon and Yelp datasets. Experimental results show its consistent superiority over the competing hashing-based baselines in cold-start item and sparse settings, and also demonstrate its competing performance with the continuous based content-aware recommender systems.

7 Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No. 61572109, 61801060). Prof.ni Ivor W. Tsang was supported by ARC FT130100746, DP180100106 and LP150100671. Dr. Hongzhi Yin was supported by ARC DE160100308, DP170103954, and New Staff Research Grant of the University of Queensland (Grant No.613134).

References

[1] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "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, 2002, pp. 253–260.

[2] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[3] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[4] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 353–362.

[5] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 153–162.

[6] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen, "Lcars: A spatial item recommender system," *ACM Transactions on Information Systems (TOIS)*, vol. 32, no. 3, p. 11, 2014.

[7] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2566–2581, 2016.

[8] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," in *Proc. of ACM SIGKDD*. ACM, 2012, pp. 498–506.

[9] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 325–334.

[10] Y. Zhang, D. Lian, and G. Yang, "Discrete personalized ranking for fast collaborative filtering from implicit feedback," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[11] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE TPAMI*, vol. 34, no. 12, pp. 2393–2406, 2012.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of UAI'09*. AUAI Press, 2009, pp. 452–461.

[13] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "Cofi rank-maximum margin matrix factorization for collaborative ranking," in *Advances in neural information processing systems*, 2008, pp. 1593–1600.

[14] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of RecSys'10*. ACM, 2010, pp. 269–272.

[15] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian, "Discrete deep learning for fast content-aware recommendation," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 717–726.

[16] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in *Nips*, vol. 1, no. 1, 2007, pp. 2–1.

[17] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[19] R. He and J. McAuley, "Vbpr: visual bayesian personalized ranking from implicit feedback," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[20] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proc. of WWW*. ACM, 2007, pp. 271–280.

[21] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

[22] A. Karatzoglou, M. Weimer, and A. J. Smola, "Collaborative filtering on a budget," in *International Conference*

on Artificial Intelligence and Statistics, 2010, pp. 389–396.

[23] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[24] Z. Zhang, Q. Wang, L. Ruan, and L. Si, "Preference preserving hashing for efficient recommendation," in *Proc. of SIGIR*. ACM, 2014, pp. 183–192.

[25] Y. Zhang, G. Yang, L. Hu, H. Wen, and J. Wu, "Dot-product based preference preserved hashing for fast collaborative filtering," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[26] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[27] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[28] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*. Springer, 2011, pp. 145–186.

[29] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou, "One-pass auc optimization," in *International Conference on Machine Learning*, 2013, pp. 906–914.

[30] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM*, vol. 48, no. 4, pp. 798–859, 2001.

[31] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *CVPR*, June 2015, pp. 37–45.

[32] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

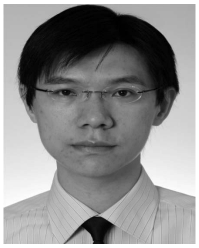
[33] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.

[34] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 931–940.

[35] Y. Koren, "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, 2008, pp. 426–434.

[36] E. M. Voorhees *et al.*, "The trec-8 question answering track report," in *Trec*, vol. 99, 1999, pp. 77–82.

Yan Zhang received her BSc degree in mathematics and applied mathematics from Sichuan normal university, China. She is currently working toward the PhD degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. Her research interests include recommender system, machine learning, and logic synthesis.



Ivor W. Tsang is an ARC Future Fellow and Professor of Artificial Intelligence, at University of Technology Sydney (UTS). He is also the Research Director of the UTS Flagship Research Centre for Artificial Intelligence (CAI). According to Google Scholar, he has more than 12,000 citations and his H-index is 53. In 2009, Prof Tsang was conferred the 2008 Natural Science Award (Class II) by Ministry of Education, China, which recognized his contributions to kernel methods. In 2013, Prof Tsang received his prestigious Australian Research Council Future Fellowship for his research regarding Machine Learning on Big Data. In addition, he had received the prestigious IEEE Transactions on Neural Networks Outstanding 2004 Paper Award in 2007, the 2014 IEEE Transactions on Multimedia Prize Paper Award, and the Best Student Paper Award at CVPR 2010. He serves as an Associate Editor for the IEEE Transactions on Big Data, the IEEE Transactions on Emerging Topics in Computational Intelligence and Neurocomputing, and area chair for NeurIPS.

Hongzhi Yin received the PhD degree in computer science from Peking University, in 2014. He is a senior lecturer with the University of Queensland. He won the Australia Research Council Discovery Early-Career Researcher Award in 2015. His research interests include recommendation system, user profiling, topic models, deep learning, social media mining, and location-based services.

Guowu Yang received his BS degree from University of Science and Technology of China in 1989, received his MSc degree from Wuhan University of Technology in 1994, and received his PhD degree in electrical and computer engineering from Portland State University in 2005. He has worked at Wuhan University of Technology from 1989 to 2001, at Portland State University from 2005 to 2006. He is a full professor at University of Electronic Science and Technology of China now. His research interests include verification, logic synthesis, quantum computing and machine learning. He has published over 100 journal and conference papers.

Defu Lian is a research professor in the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei. He received the B.E. and Ph.D. degrees in computer science from University of Science and Technology of China (USTC) in 2009 and 2014, respectively. His research interest includes spatial data mining, recommender systems, and learning to hash.

Jingjing Li received his MSc and PhD degree in Computer Science from University of Electronic Science and Technology of China in 2013 and 2017, respectively. Now he is a national Postdoctoral Program for Innovative Talents research fellow with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He has great interest in machine learning, especially transfer learning, subspace learning and recommender systems.