

Collaborative Group-Aware Hashing for Fast Recommender Systems

YAN ZHANG, Faculty of Science and Technology, Charles Darwin University, Australia

LI DENG, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

LIXIN DUAN, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

IVOR W. TSANG, CFAR and IHPC, Agency for Science, Technology and Research, Singapore College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore

GUOWU YANG, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

The fast online recommendation is critical for applications with large-scale databases; meanwhile, it is challenging to provide accurate recommendations in sparse scenarios. Hash technique has shown its superiority for speeding up the online recommendation by bit operations on Hamming distance computations. However, existing hashing-based recommendations suffer from low accuracy, especially with sparse settings, due to the limited representation capability of each bit and neglected inherent relations among users and items. To this end, this paper lodges a Collaborative Group-Aware Hashing (CGAH) method for both collaborative filtering (namely CGAH-CF) and content-aware recommendations (namely CGAH) by integrating the inherent group information to alleviate the sparse issue. Firstly, we extract inherent group affinities of users and items by classifying their latent vectors into different groups. Then, the preference is formulated as the inner product of the group affinity and the similarity of hash codes. By learning hash codes with the inherent group information, CGAH obtains more effective hash codes than other discrete methods with sparse interactive data. Extensive experiments on three public datasets show the superior performance of our proposed CGAH and CGAH-CF over the state-of-the-art discrete collaborative filtering methods and discrete content-aware recommendations under different sparse settings.

CCS Concepts: • **Information systems** → **Recommender systems; Information extraction; Personalization; Top-k retrieval in databases; Similarity measures.**

Additional Key Words and Phrases: Recommender system, hash code, group affinity, content-aware recommendation

Authors' addresses: Yan Zhang, yan.zhang@cdtu.edu.au, Faculty of Science and Technology, Charles Darwin University, Darwin, NT, Australia, 0800; Li Deng, oliverdengli@gmail.com, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China, 611731; Lixin Duan, lxduan@gmail.com, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China, 611731; Ivor W. Tsang, Ivor_Tsang@cfar.a-star.edu.sg, CFAR and IHPC, Agency for Science, Technology and Research, Singapore College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore; Guowu Yang, guowu@uestc.edu.cn, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China, 611731.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

0004-5411/2022/12-ART1 \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

ACM Reference Format:

Yan Zhang, Li Deng, Lixin Duan, Ivor W. Tsang, and Guowu Yang. 2022. Collaborative Group-Aware Hashing for Fast Recommender Systems. *J. ACM* 1, 1, Article 1 (December 2022), 23 pages. <https://doi.org/10.1145/nmnnnn.nmnnnn>

1 INTRODUCTION

Recommender systems have been extensively applied in the growing number of e-commerce websites for helping their customers find desirable products. Recommender systems recommend a particular user with a small set items 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. One of the most successful Collaborative Filtering (CF) is Matrix Factorization (MF). MF maps users and items into r -dimensional real latent space, and then provides recommendations by ranking inner products between a specific user and all items. MF-based methods have achieved excellent recommendation accuracy in both academia and industrial. However, it's challenging to provide a fast response for online recommending with large number of items in the real latent space [49, 53].

Due to observed ratings in the real world are generally sparse, which leads to poor performance with CF based recommender systems. Existing work for solving sparse and cold start issues mainly includes three folds: (1) MF variants by incorporating latent vectors extracted from side information [12, 17, 24, 30, 32, 40], which improves the recommendation accuracy with the aid of side information, such as items' figures, users' profile, and the social network of users, et al. Specifically, content-aware recommender systems learn representations of users and items from their content data, and then these representations were unified into a CF-based framework, and thus it can be applied to cope with data sparsity problem. (2) Deep models of formulating projections from side information to interactive data [8, 9, 42, 45]. (3) Cross-modal models of transferring knowledge from a domain with rich information to another domain with very sparse historical data [4, 7, 25, 51].

However, the content-aware recommendations learn continuous representations which hinders to provide a fast response for online recommending with large number of items in the real latent space [49, 53], because similarity search in continuous space is time consuming when it comes across millions or billions of items. The emerge of hashing-based recommendations makes online recommendation efficient on both response time and storage costs. Several scholars put forward hashing-based recommendation frameworks [2, 10, 38, 44, 46, 53] that significantly accelerate the online recommendation. To be specific, the hashing-based recommendation framework firstly encodes users and items into binary codes in a common r -dimensional Hamming space. Then preference scores denoted by Hamming distances can be efficiently computed by bit operations. One can even use a fast and accurate indexing method to find approximate top- k preferred items with sublinear or logarithmic time complexity [33]. So the time cost with hashing-based frameworks is significantly reduced compared with continuous frameworks. In addition, each dimension of hash codes can be stored with only one bit instead of 32/64 bits that are used for storing one dimension of continuous vectors, which greatly reduces storage cost as well.

To improve both the accuracy and efficiency with sparse settings, many scholars proposed hashing-based recommendation schemes with content information [6, 15, 16, 20, 22]. Most existing hashing frameworks learned hash codes from users' observed historical behavior data and side information, which neglects the inherent relation among users or items, which leads to unsatisfactory recommendations. For example, NeuHash-CF [6] was modeled as an autoencoder architecture consisting of two joint hashing components for generating user and item hash codes. NeuHash-CF

works in sparse and cold-start settings, but hash codes learned from two joint hashing components with autoencoders cannot capture the inherent group relation amongst users and items, which leads to hash codes of similar items/users are different from each other as shown in Figure. 1. We hope to learn hash codes that are consistent with the group affinities in the original latent space of users and items. The idea of considering group affinity to formulate our model comes from the memory-based collaborative filtering in sparse settings, where a user's preference can be predicted by their neighbors.

Actually, in the real world applications, people's preferences over items have evident group features. The inherent group relationship is naturally exists in the real world. For example, users with similar characteristics like similar movies or books. Similarly, items with similar abstract information or category information are appealing to a group of users. Especially, the rise of social media has made group activities very common in social applications. For example, we often join a group to buy a product on Pinduoduo¹, and join a group to travel together on Tuniu². This motivates us to study behaviors of groups, which in turn benefit us to develop a recommendation for coping with the data sparsity and cold-start issues.

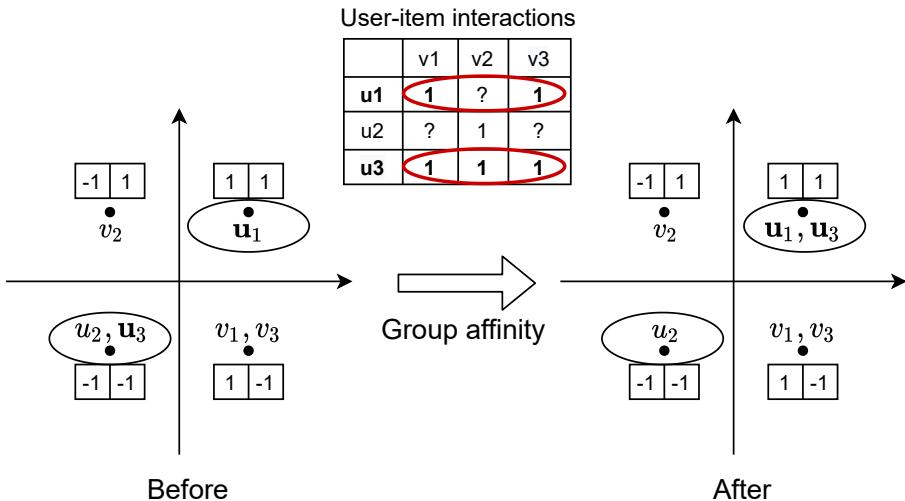


Fig. 1. The motivation of our method. From the top ‘User-item interactions’, we find user u_1 has similar behaviors with user u_3 , so we hope to learn similar hash codes of users u_1 and u_3 . Before considering inherent group information, we obtain different hash codes with DCF [44]. After considering group affinity, we can obtain the same or more similar hash codes for u_1 and u_3 by using the proposed CGAH

To this end, we propose a collaborative group-aware hashing method to improve both the accuracy in sparse settings and the efficiency of online recommendation for large-scale datasets. Motivated by the memory-based collaborative filtering methods in sparse settings, where a user's preference can be predicted by their neighbors. Besides, MF maps users and items into a shared r -dimensional latent space and predicts preference scores with their inner products, but it neglects inherent relations within users and items. The inherent relationship is naturally exists in the real world, so that users with similar characteristic like similar movies or books, similarly, items with similar abstract information is appealing to one group of users. In this paper, we seek an efficient

¹<https://en.pinduoduo.com/>

²<https://www.tuniu.com/>

way to improve the accuracy with inherent group information in sparse settings. It is expected to assist recommendations for users/items with just a few ratings. Our contributions are summarized as follows:

- We propose an efficient collaborative group-aware hashing (CGAH) method for recommendations, which encodes users and items with hash codes by incorporating inherent group affinities neglected by the state-of-the-arts hashing frameworks. To the best of our knowledge, this is the first hashing-based scheme which explicitly learns hash codes with the inherent group information.
- We build a novel hashing model for recommendations by formulating the preference model with the group affinity and the similarity of hash codes in Hamming space. Besides, the learned hash codes are in line with group affinities, which enhances the representation capability of hash codes and favors to providing more accurate recommendations under sparse settings.
- We develop an efficient discrete optimization method to solve the proposed objective.
- We design extensive experiments on three public datasets to show the superiority of CGAH and CGAH-CF on both accuracy and efficiency over the state-of-the-art discrete CF-based methods and the discrete content-aware recommendations under different sparse environments.

2 RELATED WORK

3 RELATED WORK

3.1 Content-aware Recommendations

Content-aware recommender systems have been extensively studied to cope with data sparsity and cold-start challenges by integrating side information, such as users' demographic data, items' descriptions and tags, users' reviews, users'/items' images, relations of users/items, etc. Such as the state-of-the-art collaborative deep learning (CDL) [32] was proposed as a probabilistic model by jointly learning a probabilistic stacked denoising auto-encoder (SDAE) [29] and CF. CDL exploits the interaction and content data to alleviate cold start and data sparsity problems. However, unlike CTR, CDL takes 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.

Another classic content-aware method is deep cooperative neural networks (CoNN)[52], which consists of two parallel neural networks: one learns user behaviors exploiting users' reviews, and the other one learns item properties from the items' reviews. A shared layer is introduced on the top to couple these two networks together. Then, dual attention mutual learning (DAML)[21] integrates ratings and reviews into a joint neural network with a local and mutual attention mechanism to strengthen the interpretability. In addition, higher-order nonlinear interaction of features are extracted by the neural factorization machines to predict ratings.

3.2 Group-aware Recommendations

As group activity is becoming popular in our life. Recommendations for a group of users becomes significant in current social medias. In fact, group-aware recommendations is also meaningful in traditional information system, because it can alleviate cold-start and sparse issues by group information. Many researchers focus on group recommendations, such as Agalya et al. [1] proposed a top-k recommendation scheme by the interrelationship of each group, which makes the recommendation efficient and accurate, that can be applied in Movies, Product, and Travel Recommendations.

Based on NCF, Cao et al. [3] proposed an attention mechanism to learn groups' representations. They combine interactions of individual users on items of groups and user-item interactions into their model to improve the performance. With the rise of sequential recommendations, Wang et al. [34] combined sequential recommendation and group recommendation to learn dynamic group representations by proposing a novel sequential group recommendation method, which is effective to obtain good performance. Specifically, they devised a Long-term Group-aware and Short-term Graph-aware Representation Learning approach and for sequential-based group recommendations, a long-term and a short-term group-aware graph to capture user-item interactive information. As current group recommendation frameworks fail to extract complex relations among users and items, Liang et al. [18] proposed a hierarchical fuzzy graph attention network to improve fuzzy profiling for both groups and items.

3.3 Hashing for Collaborative Filtering

hashing-based recommender systems contains two types: two-stage hashing and learning based hashing. One representative work of two-stage hashing methods is Binary Codes for Collaborative Filtering (BCCF) [53], which learn the real latent vectors of users and items, and then quantize them into binary codes. BCCF improves the efficiency of the online recommendation with binary codes, but it suffers poor accuracy performance due to the large information loss in the quantization stage.

Different from two-stage hashing frameworks, learning based hashing methods learn hash codes by directly optimizing discrete problems. Thus more information is carried by hash codes than two-stage frameworks. Zhang et al. proposed discrete collaborative filtering (DCF) [44]. 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 [26]. Then, a ranking-based hashing framework was proposed to improve the recommendation performance [46] by adding the same constraints with DCF, and it also obtained short and informative hash codes. To improve the accuracy of discrete recommendation, Wang et al. proposed the adversarial binary collaborative filtering (ABinCF) [31] formulated as a min-max optimization problem. Liu et al. proposed the compositional coding for collaborative Filtering (CCCF) [19] which innovatively represents each user/item with a set of binary vectors, which are associated with a sparse real-value weight vector. Each value of the weight vector encodes the importance of the corresponding binary vector to the user/item. The continuous weight vectors greatly enhances the representation capability of binary codes, and its sparsity guarantees the processing speed.

3.4 Hashing for Content-aware Recommendations

By combining with the content information, Zhang et al. presented the Discrete Deep Learning (DDL) [48] for solving cold start and spare problems. Besides, Liu et al. proposed a discrete social recommendation (DSR) [20] method by learning binary codes in a unified framework of users and items by considering social information. To take dual advantages of both social relations and discrete technique for fast recommendation, Discrete Trust-aware Matrix Factorization (DTMF) [5] was proposed by learning latent features of truster and trustees based on trust-aware matrix without sacrificing too much information. Then, Lian et al. [16] proposed LightRec, where each item was represented as additive composition of B codewords, which are optimally selected from each of the codebooks. LightRec and CCCF exploits similar compositional coding methods to present users and items, but LightRec utilizes a shared B codewords for improving the efficiency on both time and storage. Another type of hashing-based recommendations are developed with deep models, such as NeuHash-CF [6] was formulated as an autoencoder architecture consisting of two joint hashing components for generating user and item hash codes. To solve the cold-start and sparsity issue,

Table 1. Notations

Symbol	Description
I	users' index set
J	items' index set
R	binary matrix of all users in I
C_i	the content data of user i
C_j	the content data of item j
H	real latent vectors of all users in I
G	real latent vectors of all items in J
h_i	real latent vector of user i
g_j	real latent vectors of item j
B	hash codes of all users in I
D	hash codes of all items in J
b_i	hash code of user i
d_j	hash code of item j
\mathcal{K}	centroids (k -dimensional vectors) in the latent space of H and G
P	group indicators of H
Q	group indicators of G
p_{ik}	the distance of h_i and \mathcal{K}_k
p_{jk}	the distance of g_j and \mathcal{K}_k
s_{ij}^k	the group affinity of user i and item j
ξ_i	the embedding vector from the content C_i of user i
ζ_j	the embedding vector from the content C_j of item j
Θ_I	the parameters of encoding network of DAE for users
Θ_J	the parameters of encoding network of DAE for items
X	the delegated real matrix of B
Y	the delegated real matrix of D

Multi-Feature Discrete Collaborative Filtering (MFDCF) [41] was proposed by projecting multiple content features of users into binary hash codes via fully exploiting their complementarities.

4 COLLABORATIVE GROUP-AWARE HASHING (CGAH)

In this section, we first formulate the problem/task of this paper in Section 4.1. Then we present the preliminary study related to our proposed model in Section 4.2. Next, we introduce the proposed Collaborative Group-Aware Hashing (CGAH) for collaborative filtering and content-aware recommendations separately in Section 4.3 and Section 4.4. Finally, we develop an alternating optimization strategy to solve the mix-integer programming problems of CGAH in Section 4.5.

4.1 Problem Formulation

Let the index sets of users and items be denoted as $I = \{1, \dots, n\}$ and $J = \{1, \dots, m\}$, respectively. $r_{ij} \in R$ denotes the rating of user i rated to item j . C_i and C_j denotes content data of user i and item j , respectively. The detailed notations used in this paper are listed in Table 1

Our tasks includes:

- (1) Given ratings R , we firstly capture group affinities of users and items. Our objective is to learn hash codes of users and items by solving a discrete optimization problem, which is formulated with group affinities, ratings, and parameters; and then obtain top- k recommendations based on learned hash codes. We call the method as 'collaborative group-aware hashing for collaborative filtering' (CAGH-CF).
- (2) Given ratings $r_{ij} \in R$ and content C_i, C_j for user i and item j . Our objective is to capture group affinities of users and items from both ratings and content; and learn hash codes of users and items by solving a discrete optimization problem formated with group affinities and ratings; and finally recommend top- k items for users. We also name the method as 'CGAH for content-aware recommendations' (CGAH).

4.2 Preliminary

Collaborative filtering (CF) provides recommendations based on user-item interactive data, i.e., ratings used in this paper. Suppose each entry r_{ij} of the rating matrix $R \in \mathbb{R}^{n \times m}$ denotes the preference of user i to item j . The traditional real-valued CF methods encodes users and items with r -dimensional real latent vectors \mathbf{h}_i and \mathbf{g}_j by matrix factorization (MF), and then the unobserved preferences \hat{r}_{ij} are estimated by the inner product of the corresponding real latent vectors $\hat{r}_{ij} = \mathbf{h}_i^T \mathbf{g}_j$. Thus the MF model is generally formulated as

$$\arg \min_{H,G} \mathcal{L}_{MF}(R, H^T G) + \lambda \Gamma(H, G), \quad (1)$$

where $H \in \mathbb{R}^{r \times n}, G \in \mathbb{R}^{r \times m}$. The first term is the loss function of the true ratings and the predicted ratings. The loss functions of existing MF-based invariants consist of two types: (1) rating-based losses [13, 23, 23, 32, 49] and ranking-based losses [26, 28, 36, 37]. The second term is the regularizer of latent vectors H and G , which is applied to avoid overfitting. Recommendations based on MF have achieved good performance in both academia and industrial, however, the efficiency of online recommendation has becoming more and more crucial with the increasing numbers of items and users. Fortunately, hashing has been shown as one of the most promising technique for providing efficient online recommendations.

The state-of-the-art discrete CF-based methods encode users and items into r -dimensional binary codes \mathbf{b}_i and \mathbf{d}_j , and then formulate preferences \hat{r}_{ij} with similarities defined by the Hamming distances of hash codes $\hat{r}_{ij} = 1 - \frac{1}{r} \text{dis}_H(\mathbf{b}_i, \mathbf{d}_j)$. Taking DCF as an example, it is formulated as following objective,

$$\arg \min_{B,D} \mathcal{L}_{DCF}(R, B^T D) + \lambda \Gamma(B, D), \quad (2)$$

similarly, the loss functions contains rating-based and ranking-based loss functions. Due to the limit representation ability of each bit in hash codes, the discrete CF methods often suffer worse performance than continuous CF models, especially under sparse scenarios.

4.3 CGAH for Collaborative Filtering

Sparsity as one of challenges in recommender system, has becoming more and more serious with the increasing users and items. Thus it's significant to design accurate yet efficient recommendations under sparse settings. Previous studies alleviated the sparsity issue with hybrid recommendation techniques which incorporated content information and interactive data, but very few studies focus on providing both efficient and accurate recommendations only dependent on sparse interactive data. In this section, we attempt to improve the efficiency of CF without degrading the prediction performance. Since the fixed distance between each pair of binary codes impedes them from

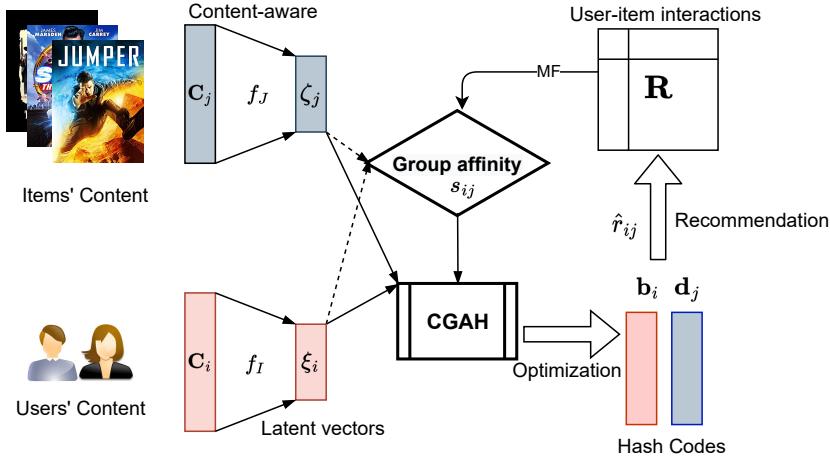


Fig. 2. The framework of CGAH: the input includes the user content C_i , the item content C_j , and user-item interactive data, i.e., ratings R ; CGAH first extracts latent vectors $\mathbf{h}_i, \mathbf{g}_j$ from ratings by matrix factorization model, and ξ_i, ζ_j respectively from users' content and items' content data by encoders; then it concatenates latent vectors from interactive data and content information to attain group infinites s_{ij} in the shared latent space; finally CGAH learns hash codes \mathbf{b}_i and \mathbf{d}_j by solving discrete objectives of CGAH-CF and CGAH for content-aware recommendations with group infinites.

modeling different magnitudes of relationships within users and items, we assign real-valued weights s_{ij} to each pair to explicitly denote the group affinities of users and items.

As shown in Figure 2, the framework of CGAH consists of two phases: the first phase learns latent vectors $\mathbf{h}_i, \mathbf{g}_j$ and ξ_i, ζ_j respectively from interactive data and users'/items' content data, and then concatenates them together to calculate group affinities s_{ij} in the shared latent space; the second phase learns hash codes \mathbf{b}_i and \mathbf{d}_j by solving discrete objectives of CGAH-CF and CGAH for content-aware recommendations with group infinites. We formulate CGAH with two objectives respectively for collaborative filtering and content-aware recommendations.

In this section, we will introduce the model of CGAH for collaborative filtering. Motivated by the memory based CF for the recommendation in sparse settings, where a user's preference can be predicted by their neighbors. Besides, the MF factorizes the rating matrix R into a user matrix H and an item matrix G , but it only considers users' and items' own affinities in the latent space to predict unobserved ratings, and does not consider the inherent group affinities existed within users and items, such as types of characteristics for users: generosity, integrity, loyalty, devotion, loving, kindness, et al., and abstract types of movies: romance, action, drama, et al. In this paper, we seek an efficient way to improve the recommendation accuracy under sparse settings with the inherent group information. In MF, we suppose users' latent preferences over items' abstract types are classified into κ groups, then we denote the real latent vectors of users and items H and G as multiplications of group indicator matrices $P \in \mathbb{R}^{\kappa \times n}, Q \in \mathbb{R}^{\kappa \times m}$ respectively:

$$H = \mathcal{K}^T P, G = \mathcal{K}^T Q, \quad (3)$$

where $\mathcal{K} \in \mathbb{R}^{\kappa \times r}$ is a shared codebook in the latent space, which is stacked by κ vectors in H and G . To enhance the representation ability of group indicators P and Q , we impose the uncorrelated

constraints on vectors in \mathcal{K} and derive the following objective,

$$\begin{aligned} \arg \min_{\mathcal{K}, P, Q} & ||H - \mathcal{K}^T P||_F^2 + ||G - \mathcal{K}^T Q||_F^2, \\ \text{s.t. } & \mathcal{K}\mathcal{K}^T = rI_{\kappa}. \end{aligned} \quad (4)$$

The problem (4) is intractable due to the uncorrelated constraint, thus we seek a way to find an approximate solution. Since group indicators P and Q can be considered as distances away from vectors in \mathcal{K} . By associating it with K-means clustering, these κ -dimensional vectors in the shared codebook \mathcal{K} can be regarded as centroids in the latent space of H and G , and each latent vector can be represented by these centroids. To simplify the calculation of group affinities, we obtain centroids in \mathcal{K} by K-means clustering of vectors in the latent space of H and G , and then we compute distances P and Q away from these centroids.

We suppose a general distance function f_{dis} , which is defined to measure the distance between a user to an item, which aims to compute the group affinity s_{ij} of user i and item j . The distance metric can be constructed with side information, like users' social relationships [35, 50] or using metric learning techniques [39]. However, most datasets do not include such data. So in this paper we explore a general method to estimate the group affinity. Followed by the idea [14] of estimating distances with cosine similarities of latent representations, we define the distance of each element $\mathbf{h}_i \in H$ and $\mathcal{K}_k \in \mathcal{K}$, and the distance of each element $\mathbf{g}_j \in G$ and $\mathcal{K}_k \in \mathcal{K}$, respectively as follows:

$$\begin{aligned} p_{ik} &= f_{dis}(i, k) = \text{cosine}(\mathbf{h}_i, \mathcal{K}_k), \\ p_{jk} &= f_{dis}(j, k) = \text{cosine}(\mathbf{g}_j, \mathcal{K}_k), \end{aligned} \quad (5)$$

where $k \in 1, \dots, \kappa$. p_{ik}, q_{jk} characterize distances of latent vectors away from the centroid \mathcal{K}_k . Intuitively, users/items in the same group has small discrepancy of distances, we define the group affinity (group similarity) with these distance discrepancies, which composes of κ components as follows:

$$s_{ij}^k = \sigma(1 - \Delta_{ij}^k), \quad (6)$$

where

$$\Delta_{ij}^k = |p_{ik} - q_{jk}| \quad (7)$$

for $k \in \{1, \dots, \kappa\}$ denotes the distance discrepancy of user i and item j away from the centroid \mathcal{K}_k , and σ is the sigmoid(logistic) function. We can conclude the group affinity is monotone decreasing with the distance discrepancy from Equation (6), that is reasonable because users/items from the same group having smaller distance discrepancies are more affinitive than that from different groups. Furtherly, we define the group affinity s_{ij} as

$$s_{ij} = \max_k s_{ij}^k. \quad (8)$$

From Equation (8), we regard the maximum group affinity from κ group affinity components as the group affinity of users and items, which is interpretable from the perspective of clustering, which vectors are classified into the nearest class. So we can classify users and items into the same group when the maximum affinity greater than a threshold, thus the maximum affinity with κ centroids can be looked as the group affinity.

Next, we focus on learning hash codes with the obtained group affinity. We assume users and items as binary codes $B \in \{\pm\}^{r \times n}$ and $D \in \{\pm\}^{r \times m}$, respectively, then the preference of user i over item j is formulated as follows,

$$\hat{r}_{ij} = s_{ij} \text{sim}_H(\mathbf{b}_i, \mathbf{d}_j), \quad (9)$$

where $\text{sim}_H(\mathbf{b}_i, \mathbf{d}_j)$ denotes the similarity function of hash codes \mathbf{b}_i and \mathbf{d}_j defined by their Hamming distances in r -dimensional Hamming space, and the similarity function is defined as follows,

$$\text{sim}_H(\mathbf{b}_i, \mathbf{d}_j) = \frac{1}{2} + \frac{1}{2r} \mathbf{b}_i^T \mathbf{d}_j. \quad (10)$$

By substituting Equation (10) into the preference prediction model (9), we can predict ratings with group infinities and similarities of hash codes. We formulate the preference as the product of the group affinity and the similarity of hash codes instead of the addition operation, because we tend to learn hash codes in consistent with the group affinity to improve the accuracy for sparse recommendation. Then we derive the following objective of the collaborative group-aware hashing (CGAH) for collaborative filtering,

$$\arg \min_{B, D} \mathcal{L}_{CGAH-CF}(R, S, \hat{R}) + \lambda \Gamma(B, D), \quad (11)$$

where \hat{R} is the matrix consisting of the predicted preference \hat{r}_{ij} , and the first term is the loss function between ground truth ratings and predicted preference scores, and the second term $\Gamma(B, D)$ denotes the regularizers (constraints) imposed on hash codes. In this paper, we use the square loss to learn the binary codes in line with the inherent group affinities from ratings. To obtain more compact yet informative hash codes, we impose balance constraints and uncorrelated limitations on hash codes like DCF [43]. By substituting the Equation (10) into Equation (9), the proposed CGAH for collaborative filtering is formulated as follows,

$$\begin{aligned} \arg \min_{B, D} \mathcal{L}_{CGAH-CF}(R, S, \hat{R}) &= \sum_{(i, j) \in \Omega} (r_{ij} - \frac{1}{2} s_{ij} - \frac{1}{2r} s_{ij} \mathbf{b}_i^T \mathbf{d}_j)^2 \\ \text{s.t. } B &\in \{\pm 1\}^{r \times n}, D \in \{\pm 1\}^{r \times m} \\ B\mathbf{1}_n &= 0, D\mathbf{1}_m = 0, BB^T = nI_r, DD^T = mI_r, \end{aligned} \quad (12)$$

where $\Omega = \{(i, j) | r_{ij} \in R\}$ presents the index of observed ratings in R .

The idea of CGAH is similar to the compositional coding method CCCF [19], that denotes hash codes as a weight aggregation of compositional codes. The difference lies in the construction of hash codes. CCCF formulates each hash code as a combination of κ hash codes, and the aggregation parameter can be regarded as the group infinity in this work; while CGAH denotes hash codes as binary vectors directly, and the group infinity is joined to learn these hash codes. So hash codes learned from CGAH captures the group information inherited amongst users and items. Besides, the CGAH attains more compact hash codes than CCCF. Because CCCF has to learn κ r -dimensional hash codes while CGAH only need to learn on r -dimensional hash code for each user or item.

4.4 CGAH for Content-aware Recommendations

To alleviate sparse issues in recommender systems, we design CGAH for recommendations by incorporating with sice information, such as users'/items' contexts, images, social links, etc. In this paper, we formulate CGAH with context information of users and items. From Figure 2, we first encode users'/items' content into embeddings ξ_i and ζ_j of user i and item j , and then we concatenate latent vectors from ratings(\mathbf{h}_i and \mathbf{g}_j), and embeddings (ξ_i and ζ_j) from contents to calculate their group affinities s_{ij} , and finally we learn hash codes by solving the discrete objective of CGAH.

In this work, we encode items'/users' content with denoising auto-encoders (DAE), which can extract robust representations by corrupting the input data by randomly setting some of the input values to zero. The percentage of input nodes which are being set to zero depends on the dimension and the number of input data. Commonly the hidden layer in the middle of DAE is the latent vector

we need, and the input layer is the corrupted version of the clean input data. Let the content of user i be denoted as C_i , and the content of item j be denoted as C_j , and latent vector is the output of the middle layer of DAE denoted as follows,

$$\xi_i = f_U(C_i, \Theta_I); \zeta_j = f_I(C_j, \Theta_I), \quad (13)$$

where $\Theta_I = \{W_U, b_U\}$ and $\Theta_J = \{W_I, b_I\}$ are the parameters of encoding networks of two DAE models for users and items, respectively. Actually, we first learn the whole DAE by minimizing the reconstruction error of users' and items' embedding, respectively, and then we fix Θ_I and Θ_J to extract latent vectors ξ_i and ζ_j by the forward steps.

To calculate the group infinity in latent space, we concatenate latent vectors from MF obtained by Equation (1) and DAE extracted by Equation (13) together. The concatenation latent vectors are as follows:

$$\mathbf{u}_i = [\mathbf{h}_i; \xi_i]; \mathbf{v}_j = [\mathbf{g}_j; \zeta_j]. \quad (14)$$

Then we take similar steps of Section 4.3 to calculate group affinities s_{ij} of users and items by replacing latent vectors \mathbf{h}_i and \mathbf{g}_j with \mathbf{u}_i and \mathbf{v}_j , respectively.

Specifically, we firstly calculate the centroids \mathcal{K} of latent vectors $\mathbf{u}_i \in U$ and $\mathbf{v}_j \in V$ by solving the problem (4); then we calculate the distance of each element in U and each element in \mathcal{K} , and the distance of each element in V each element in \mathcal{K} by the Equation (5), respectively; followed by estimating the group affinity s_{ij} by the Equation (6).

To solve sparse and cold-start problems, we need to learn hash codes carrying information from both ratings and contents. Motivated by DDL [48] which learns hash codes from content data by adding a supervised layer on the output layer of the encoder, and finetunes parameters of the encoder in the learning procedure of hash codes. Thus the objectives of supervised learning for users and items are denoted as follows:

$$\begin{aligned} & \arg \min_{\Theta_I} \sum_{i=1}^n \|\mathbf{b}_i - \xi_i\|_F^2; \\ & \arg \min_{\Theta_J} \sum_{i=1}^m \|\mathbf{d}_j - \zeta_j\|_F^2. \end{aligned} \quad (15)$$

By minimizing the objectives in Equation (15), we obtain effective item hash code from content data. Then we formulate the objective of CGAH with ratings and users' and items' content data as follows,

$$\arg \min_{B, D, \Theta_I, \Theta_J} \mathcal{L}_{CGAH}(R, S, \hat{R}, C_i, C_j) + \lambda \Gamma(B, D, \Theta_I, \Theta_J), \quad (16)$$

where the first term consists of the rating loss function and the two content-based loss functions in Equation (15), and it is denoted as follows,

$$\mathcal{L}_{CGAH}(R, S, \hat{R}, C_i, C_j) = \mathcal{L}_{CGAH-CF}(R, S, \hat{R}) + \lambda_1 \|\mathbf{b}_i - \xi_i\|_F^2 + \lambda_2 \|\mathbf{d}_j - \zeta_j\|_F^2, \quad (17)$$

where λ_1 and λ_2 are the hyperparameters that weight the importance of three objectives. Similarly, we impose balance constraints and uncorrelated limits on hash codes and obtain the following CGAH objective,

$$\begin{aligned} & \arg \min_{B, D, \Theta_I, \Theta_J} \mathcal{L}_{CGAH}(R, S, \hat{R}, C_i, C_j) \\ & s.t. B\mathbf{1}_n = \mathbf{0}, D\mathbf{1}_m = \mathbf{0}, BB^T = nI_r, DD^T = mI_r. \end{aligned} \quad (18)$$

4.5 Optimization

In this section, we introduce the way of solving the problems (12) and (18). As the problem (12) is similar to the problem (18), and the latter is more complex. We can easily derive the optimization steps of solving problem (12) by discarding the content objective (15). Hence, in this section, we introduce the detailed steps of solving the problem (18) for simplicity.

As the problem (18) is a discrete problem with strict conditions, it is intractable to directly solve objective. In previous work, they generally solve such discrete problems by using the Discrete Coordinate Decent (DCD) [27], which learns hash codes with a bitwise way. Another method to solve the discrete problems is proposed in DRMF [47]. In contrast to discrete coordinate descent, DRMF firstly finds an upper bound of the original discrete problem by adding variational parameters, and then transforms the upper bound into a series of Binary Quadratic Programming (BQP) subproblems, and finally updates the entire hash code \mathbf{u}_i in each iteration by directly optimizing these BQP problems with a BQP solver³. Due to the limit computation of current computers, the BQP solver is time consuming, which leads inefficiency to training the model.

Consequently, in this paper, we still take a bitwise way to solve the discrete problem. Firstly, we adopt a strategy to solve the problem by softening the balance and uncorrelated constraints with two delegate continuous matrices [43]. Let two sets $\mathcal{B} = \{X \in \mathbb{R}^{r \times n} | X\mathbf{1}_n = \mathbf{0}, XX^T = nI_r\}$, $\mathcal{D} = \{Y \in \mathbb{R}^{r \times m} | Y\mathbf{1}_m = \mathbf{0}, YY^T = mI_r\}$. Then we derive the following delegate objective of (18),

$$\begin{aligned} & \arg \min_{B, D, \Theta_I, \Theta_J} \mathcal{L}_{CGAH}(R, S, \hat{R}, C_i, C_j) - 2\alpha \text{tr}(B^T X) - 2\beta \text{tr}(D^T Y) \\ & \text{s.t. } B \in \{\pm 1\}^{r \times n}, D \in \{\pm 1\}^{r \times m}, \end{aligned} \quad (19)$$

where $\text{tr}(B^T X)$ and $\text{tr}(D^T Y)$ represent the discrepancies between the binary codes B and delegated values X , D and Y , and α and β are tuning parameters so that the second and last terms in Eq.(19) allow certain discrepancy hash codes and delegated vectors. In fact, the two discrepancies are equivalently transformed from $\|B - X\|_F^2$ and $\|D - Y\|_F^2$, respectively, because $\text{tr}(BB^T) = \text{tr}(XX^T) = nr$ and $\text{tr}(DD^T) = \text{tr}(YY^T) = mr$. It's worth noting that we do not discard the binary constraint on hash codes B and D in the objective (19). Then we adopt the following alternating optimization to solve the problem.

a. Update B by fixing D , X , Θ_I , Θ_J and Y

Since the objective function in the problem (19) sums over users independently, we update B by updating \mathbf{b}_u in parallel by solving the following problem,

$$\arg \min_{b_i \in \{\pm 1\}^r} \frac{1}{4r^2} \sum_{j \in \Omega_i} s_{ij}^2 (d_j^T b_i)^2 - \frac{1}{r} \sum_{j \in \Omega_i} g_{ij} (r_{ij} - \frac{1}{2} g_{ij}) d_j^T b_i - 2\alpha x_i^T b_i - 2\lambda_1 \xi_i^T b_i, \quad (20)$$

where Ω_i is the items set rated by the user i . This discrete optimization problem is NP-hard, and thus we adopt the Discrete Coordinate Descent (DCD) [27] to update \mathbf{b}_i with a bitwise way. Particularly, let b_{ik} be the k -th bit of \mathbf{b}_i , and $\mathbf{b}_{i\bar{k}}$ be the rest bits of \mathbf{b}_i , i.e., $\mathbf{b}_i = [\mathbf{b}_{i\bar{k}}^T, b_{ik}]^T$, then DCD can update b_{ik} given $\mathbf{b}_{i\bar{k}}$. Discarding the terms independent of b_{ik} , the objective in Eq.(20) is rewritten as

$$\arg \min_{b_{ik} \in \{\pm 1\}} \hat{b}_{ik} b_{ik}, \quad (21)$$

where $\hat{b}_{ik} = \frac{1}{4r^2} \sum_{j \in \Omega_i} (s_{ij}^2 d_{j\bar{k}}^T \mathbf{b}_{i\bar{k}} - 2r g_{ij} r_{ij} - r g_{ij}) d_{jk} + \alpha x_{ik} + \lambda_1 \xi_{ik}$. The above subproblem can achieve 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

³https://www_cvxpy.org

updated, so the update rule of b_{ik} is as follows,

$$b_{ik} = \text{sgn} \left(\chi \left(-\hat{b}_{ik}, b_{ik} \right) \right), \quad (22)$$

where $\chi(x, y) = x$ if $x \neq 0$; otherwise, $\chi(x, y) = y$.

b. Update D by fixing B , X , Θ_I , Θ_J and Y

Similarly, discarding terms irrelevant to d_j in Eq (19), we obtain the following subproblem:

$$\arg \min_{d_j \in \{\pm 1\}^r} \frac{1}{4r^2} \sum_{i \in \Omega_j} s_{ij} (\mathbf{b}_i^T \mathbf{d}_j)^2 - \frac{1}{2r} \sum_{i \in \Omega_j} g_{ij} (2r_{ij} - g_{ij}) \mathbf{b}_i^T \mathbf{d}_j - 2\beta \mathbf{y}_j^T \mathbf{d}_j - 2\lambda_2 \zeta_j^T \mathbf{d}_j, \quad (23)$$

where Ω_j is the users set that rated item j . Similarly, we obtain the following objective with the DCD method,

$$\arg \min_{b_{ik} \in \{\pm 1\}} \hat{d}_{jk} d_{jk}, \quad (24)$$

where $\hat{d}_{jk} = \frac{1}{4r^2} \sum_{i \in \Omega_j} (s_{ij}^2 \mathbf{b}_{ik}^T \mathbf{d}_{jk} - 2rg_{ij}r_{ij} - rg_{ij}) b_{ik} + \beta y_{jk} + \lambda_2 \zeta_{jk}$. Similarly, the update rule of d_{jk} is:

$$d_{jk} = \text{sgn} \left(\chi \left(-\hat{d}_{jk}, d_{jk} \right) \right). \quad (25)$$

c. Update X by fixing B , D , Θ_I , Θ_J and Y

Given B , D , Y , the Eq.(24) is furtherly transformed as

$$\arg \max_{X \in \mathbb{R}^{r \times n}} \text{tr}(B^T X), \text{ s.t. } X \mathbf{1}_n = \mathbf{0}, X X^T = n I_r. \quad (26)$$

It can be solved with the help of SVD according to [44]. Specifically, X is updated by

$$X = \sqrt{n} [\mathbf{U}_s \widehat{\mathbf{U}}_s] [\mathbf{V}_s \widehat{\mathbf{V}}_s]^T, \quad (27)$$

where \mathbf{U}_s and \mathbf{V}_s are respectively stacked by the left and right singular vectors of the row-centered matrix \bar{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 $[\mathbf{V}_s \mathbf{1}]^T \widehat{\mathbf{V}}_s = \mathbf{0}$.

d. Update Y by fixing B , D , Θ_I , Θ_J and X

Similarly, we derive the following subproblem of the Eq.(27)

$$\arg \max_{Y \in \mathbb{R}^{r \times m}} \text{tr}(D^T Y), \text{ s.t. } Y \mathbf{1}_m = \mathbf{0}, Y Y^T = m I_r, \quad (28)$$

then, Y is updated by

$$Y = \sqrt{m} [\mathbf{P}_s \widehat{\mathbf{P}}_s] [\mathbf{Q}_s \widehat{\mathbf{Q}}_s]^T. \quad (29)$$

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

Algorithm 1: CGAH

```

1 Input: Rating matrix  $R$ , Users' and Items' content  $T_U, T_I$ 
2 Parameters: The number of groups  $\kappa$ , the length of each hash code  $r$ , hyper-parameters  

 $\lambda_1, \lambda_2, \alpha$  and  $\beta$ 
3 Output: Hash codes  $B$  and  $D$ 

1: Get group indicators  $P$  and  $Q$  by Eq. (5)
   Get group affinities  $s_{ij}$  of user  $i$  and item  $j$  by Eq. (8)
2: while not converged do
3:   for  $i = 1 \dots n$  do
4:     while not converged do
5:       for  $k = 1 \dots r$  do
6:         Update  $b_{ik}$  according to Eq.(22)
7:       end for
8:     end while
9:   end for
10:  for  $j = 1 \dots m$  do
11:    while not converged do
12:      for  $k = 1 \dots r$  do
13:        Update  $d_{jk}$  according to Eq.(25)
14:      end for
15:    end while
16:  end for
17:  Update  $X$  according to Eq.(27)
18:  Update  $Y$  according to Eq.(29)
19:  Update  $\Theta_I$  and  $\Theta_J$  by learning problems in Eq. (30)
20: end while
21: return  $B, D$ 

```

e. Update Θ_I (Θ_J) by fixing B, D and X, Y

We finetunes parameters in encoders of users and items, respectively. The supervised objectives are as follows:

$$\begin{aligned} & \arg \min_{\Theta_I} \sum_{i=1}^n \|b_i - f_U(C_i, \Theta_I)\|^2; \\ & \arg \min_{\Theta_J} \sum_{j=1}^m \|d_j - f_I(C_j, \Theta_J)\|^2. \end{aligned} \quad (30)$$

Problems (30) are supervised learning tasks for users and items, respectively. We choose the stochastic gradient descent (SGD) method to fine-tune parameters of the two encoders, where the gradient descent part is implemented by the Back-propagation algorithm. As $b_i, d_j \in \{\pm 1\}^r$, we choose \tanh function as the output function of encoders since the output is in the same range $[-1, 1]$ with hash codes. Besides, we choose sigmoid functions as activation functions for hidden layers. We summarize the optimization method as Algorithm 1.

5 EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed CGAH by comparing with the state-of-the-art hashing based recommendations. As our method can be used for both collaborative filtering denoted as CGAH-CF (introduced in Section 4.3) and content-aware recommendations denoted as CGAH (introduced in Section 4.4), so we evaluate the performance of CGAH-CF and CGAH against discrete CF-based methods and discrete content-aware recommendations. Specifically, we present experimental results by answering the following questions:

- **RQ1:** How about the efficiency advantages of hashing-based methods over real-valued methods for the online recommendation?
- **RQ2:** Does the proposed CGAH provide more accurate recommendation with content information by comparing with the state-of-the-art discrete content-aware recommendations?
- **RQ3:** Does the proposed CGAH-CF outperforms the state-of-the-art discrete collaborative filtering methods ?
- **RQ4:** How does the group affinity affect the original matrix factorization?

5.1 Experimental Settings

5.1.1 Datasets. We evaluate the performance of our method and the competing baselines on three public datasets: MovieLens-1M⁴, CDs and Books subsets of Amazon datasets⁵. The MovieLens dataset contains 1,000,209 user-item interactive data, i.e., ratings within the range of 1 to 5 collected from 6040 users to 3706 movie, which reflects users' preferences over movies on MovieLens website⁶. Amazon datasets contain product reviews and metadata from Amazon⁷, which covers 142.8 million reviews (including ratings, text, helpfulness votes) spanning May 1996 - July 2014. In this paper, we use small '5-core' subsets of Amazon datasets for experiments, where each user has rated at least 5 items, and each item has been rated by at least 5 users.

As this paper propose two models for alleviating sparse issue under two settings: CGAH-CF with only ratings available and CGAH with both ratings and reviews available. So, in this paper, we use the rating datasets of all three datasets to evaluate the effectiveness of CGAH-CF, and use both ratings and reviews on Books and CDs to evaluate the effectiveness of CGAH for content-aware recommendations. For all datasets, we first remove users or items with less than 10 ratings. The statistics of datasets used in this paper are listed in Table 2.

To evaluate the performance of the proposed method on different sparse settings,we randomly choose different proportions' (10%, 50%, 90%) ratings of each user to construct the training datasets, and the remaining ratings to build testing sets. The random selection is carried out 5 times independently, and we report the experimental results as the average values. We evaluate the accuracy performance under NDCG@ k [11], which has been widely used for evaluating ranking tasks, especially for recommender systems.

5.1.2 Evaluation Metric. As introduced in Section 4.1, the task of this paper is to recommend top- k items that users may be interested in. So we choose one of the most common used evaluation metric Normalized Discounted Cumulative Gain(NDCG) [11], to evaluate the performance of predicted ranking lists of items for users. To evaluate the qualities of top- k recommendation results, we need to compute the value of NDCG@ k , which is widely adopted by many previous ranking-based recommender systems [12].

⁴<https://grouplens.org/datasets/MovieLens/>

⁵<http://jmcauley.ucsd.edu/data/amazon/>

⁶<https://MovieLens.org>

⁷<https://www.amazon.com>

Table 2. Statistics of datasets.

Dataset	#Users	#Items	#Ratings	Sparsity(%)
MovieLens	6,040	3,706	1,000,209	99.97%
CDs	25,400	24,904	43,193	99.98%
Books	603,374	348,957	8,575,903	99.99%

NDCG@ k is a good measure to evaluate the quality of a ranking task, because it takes position significance into account. It is defined as the ratio of Discounted Cumulative Gain(DCG) of a recommended ranked list to DCG of the ideal ranked list,

$$\text{NDCG}@k = \frac{\text{DCG}@k}{\text{iDCG}@k}, \quad (31)$$

and the ratio takes values in the range of [0, 1]. It takes the value ‘1’ when the recommended items’ top- k list is the same with the true top- k list. At this moment, the recommendation performance is considered as ideally catering the user’s interest. When it takes the value ‘0’, the recommended items’ top- k list is totally different from the true top- k list.

The DCG is defined as the relevance score by dividing it with the log of the corresponding position in the top- k list,

$$\text{DCG}@k = \sum_{i=1}^k \frac{2^{\hat{r}_i} - 1}{\log_2(i + 1)}, \quad (32)$$

where the relevance score \hat{r}_i is binary, i.e., it takes either ‘0’ or ‘1’. In predicted recommended top- k list, if a user rated to an item, the corresponding relevance score \hat{r}_i takes ‘1’, ‘0’ otherwise.

iDCG is defined as follows,

$$\text{iDCG}@k = \sum_{i=1}^k \frac{2^{r_i} - 1}{\log_2(i + 1)}, \quad (33)$$

where the relevance score r_i is also binary. Different from DCG@ k , iDCG@ k is calculated in the true dataset. In true recommendation top- k list, if a user rated to an item, the corresponding relevance score r_i takes ‘1’, ‘0’ otherwise.

5.1.3 Baselines. As introduced in Section 2, we choose two types of comparison methods including the state-of-the-art real-valued (continuous) method MF, three latest competitive hashing-based recommender systems BCCF, DCF, and CCCF for discrete collaborative filtering, and two kinds of discrete content-aware recommendations DDL and NeuHash-CF.

- **MF:** Matrix factorization [13] is the most favorite state-of-the-art technique in recommender system, and so there are a lot of variants methods based on MF were put forward in academia and industrial for improving existing recommendations. MF maps users and items in real latent space and then conducts recommendation with continuous representations, and so the online recommendation is not efficient .
- **BCCF:** Binary codes for collaborative filtering [53] is a two-stage hashing-based recommendation, which learn the real latent vectors of users and items, and then quantize them into binary codes. BCCF improves the efficiency of the online recommendation with binary codes, but it suffers poor accuracy performance due to the large information loss in the quantization stage.
- **DCF:** Discrete collaborative filtering [44] is the state-of-the-art hashing-based recommender systems. Different from BCCF, the DCF directly learn hash codes by optimizing a discrete optimization problem, and then maps the users and items into r -dimensional Hamming space.

DCF can provide efficient online recommendation, besides, the accuracy is also competitive with continuous methods. However, the DCF performs inferior under sparse scenarios.

- **CCCF:** Compositional coding for collaborative filtering [19] is also a hashing-based recommender system with directly optimizing a discrete problem. CCCF constructs a compositional hashing method with G components for each user and item, respectively.
- **DDL:** Discrete deep learning [48] is a content-aware hashing method for solving cold start and spare problem. DDL also collaborates interactive data and content data to learn hash codes of users and items. The differences between DDL and CGAH include the group infinity proposed in CGAH and the capability of solving the user cold start problem.
- **NeuHash-CF:** Content-aware Neural Hashing [6] was formulated as an autoencoder architecture consisting of two joint hashing components for generating user and item hash codes.

5.1.4 Hyper-parameter Settings. We tune the optimal hyper-parameters of our algorithm by the grid search method. Specifically, for the proposed CGAH, we search the optimal hyper-parameters λ , λ_1 and λ_2 over $\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^1, 1e^2\}$ on each dataset. The dimension of discrete/continuous representations is set to $r = 20$ by default. The number of groups for users' preference over items is set to $\kappa = 10$. Finally, we set $\lambda = 0.1$ for CGAH-CF, and set $\lambda_1 = 0.2$ and $\lambda_2 = 0.1$ for CGAH for content-aware recommendations. For other baselines, we use the default optimal hyper-parameters in their public codes to evaluate their performances on each dataset.

5.2 Experimental Results and Analysis

5.2.1 Efficiency comparison with baselines (RQ1**).** As discussed in other hashing-based methods [19, 44, 46], recommendations with hash codes can be significantly accelerated with the fast similarity search by XOR bit operations. Due to all continuous methods obtain real-valued vectors, and the online recommendation stage consists of calculating the predicted ratings based on the obtained continuous vectors and ranking these ratings to recommend top- k items for specific users. Hence, the online recommendation stage are the same for all continuous methods, and thus we choose the representative MF for comparison. Table 3 shows the efficiency comparisons of the online recommendation on Books dataset by comparing hashing-based methods with the continuous method MF.

From Table 3, we find that the proposed CGAH provide the fastest online recommendation among all hashing-based methods, and it takes least storage (the least compress ratio). Compared with CCCF, our method attains one r -dimension hash codes for each user/item; while CCCF obtain κ hash codes of r -dimension for each user/item. That's why CCCF costs most time and storage for online recommendation. For BCCF, it learns hash codes without balance constraint. If we save hash codes as sparse format, it will cost more than balance hash codes obtained from DCF and our method. Compared with DCF, our method gets similar hash codes for similar users/items (with greater group affinity), while DCF may get different hash codes for similar users/items. That's the reason that the proposed CGAH is efficient for online recommendations.

5.2.2 The Effectiveness of CGAH for Content-aware Recommendations(RQ2**).** As the objective is to solve the sparse issue in recommendations, thus, in this part we randomly choose 10% ratings of each user as training dataset, and the remaining 90% ratings of each user as testing set. Figure 3 shows the experimental results of CGAH for providing top- k recommendations with content information by comparing with other two content-aware hashing-based recommendations NeuHash-CF and DDL on CDs and Books, respectively.

Table 3. Speedup ratios(%) of hashing-based frameworks to the continuous MF on Books dataset, and the storage compress ratios(%) of items' representations

	Speedup ratios(k=10, 50, 100)			Compress ratios
	10	50	100	
CCCF	35.50	29.60	21.69	30.21
BCCF	17.65	14.38	10.07	15.53
DCF	14.17	10.31	9.25	10.29
CGAH	12.82	10.32	8.05	8.90

From the Figure 3, we find our method outperforms other two competing baselines in the sparse setting, and we have the following findings:

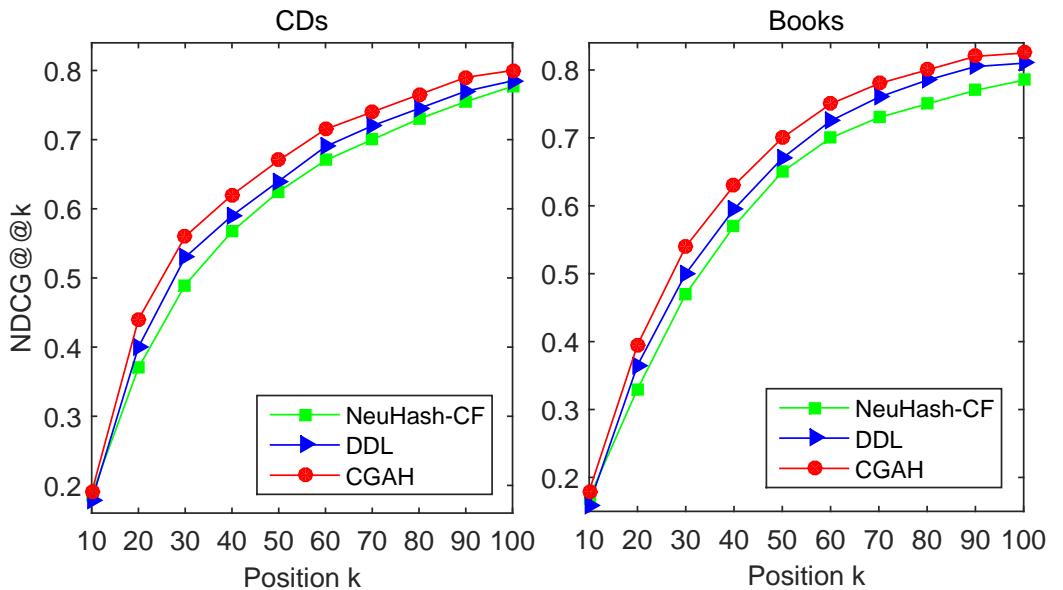


Fig. 3. The performance comparison of discrete content-aware recommendations on CDs and Books with the sparse setting 10%

- *The group information carried by hash codes can improve the accuracy.* We know that DDL and NeuHash-CF are hashing methods collaborated with content information. DDL and NeuHash-CF are designed to learn representations from ratings and content, which aims to learn models to fit ratings; while CGAH is formulated to adapt both the inherent group information (from content and ratings) and the observed ratings. So the CGAH attain binary codes with group information, and thus it can provide more accurate recommendations under the very sparse setting 10%.
- *The group information helps to enhance the representation ability of hash codes.* Despite that hashing-based recommender systems can speed up the online recommendation by the fast similarity search in Hamming space, but each bit naturally carries much less information than each continuous dimension, which leads to poor recommendation accuracy. By extracting

group information hided in ratings, CGAH obtains hash codes in consistent with the group affinity of users and items. That's why CGAH can achieve the best performance.

5.2.3 The Effectiveness of CGAH for Collaborative Filtering (CGAH-CF) (RQ3**)**. Figure 4 and Figure 5 show the accuracy performance by comparing with discrete CF methods on MovieLens and CDs datasets, respectively. The two figures indicate the proposed CGAH-CF outperforms other three discrete CF methods: BCCF, DCF and CCCF under different sparse settings 10%, 50% and 90%, and we have the following findings from the results.

- *The group information carried by hash codes can improve the accuracy.* We know that CGAH-CF, DCF and CCCF are hashing methods by directly optimizing discrete problems. DCF and CCCF are designed to fit the observed ratings; while CGAH-CF is formulated to adapt both the inherent group information and the observed ratings. So the CGAH can attain binary codes with group information, and thus it can provide more accurate recommendation under various sparse settings.
- *The group information helps to enhance the representation ability of hash codes.* As CGAH-CF can obtain hash codes that are consistent with the group affinities. Hence, hash codes should be more effective to represent both user-item interactions and user-item group affinities. However, other discrete CF methods are not capable of representing group information with hash codes. That's why the proposed CGAH-CF performs better than baselines.

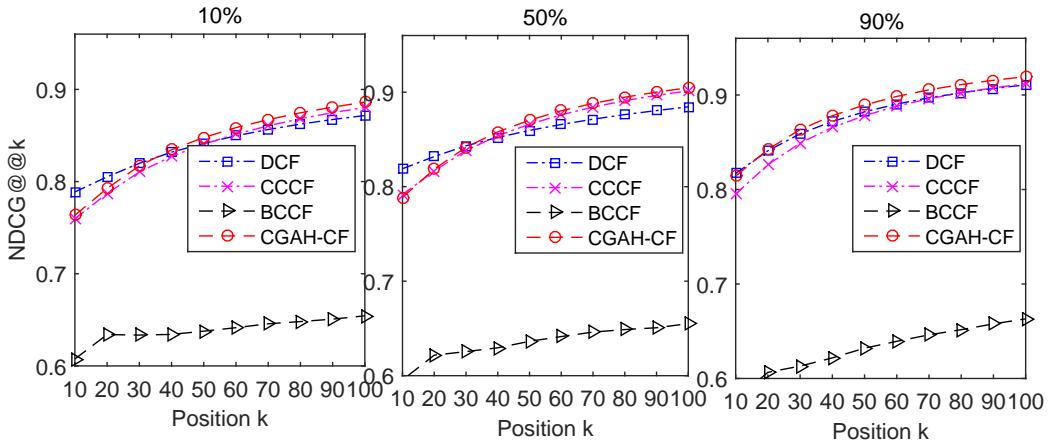


Fig. 4. The accuracy performance for collaborative filtering on MovieLens with different sparsity levels(10%, 50%, 90%).

5.2.4 The Effectiveness of the Group Affinity for Matrix Factorization(RQ4**)**. To evaluate the effectiveness of group affinities developed in the proposed CGAH, we do ablation studies on the state-of-the-art MF model. MF trained together with group affinities is denoted as MF-GA. The experimental results shown in Figure 6 and Table 4 tell us the performance of MF-GA is better than the original MF model. For MF-GA, we only add the group affinities calculated by Equation 8 on the original MF, and other components in MF and the learning algorithm keep the same with MF. The objective of MF is introduced in the problem (1), and the objective of MF-GA is as follows.

$$\arg \min_{H, G} \mathcal{L}_{MF-GA}(R, S, H^T G) + \lambda \Gamma(H, G), \quad (34)$$

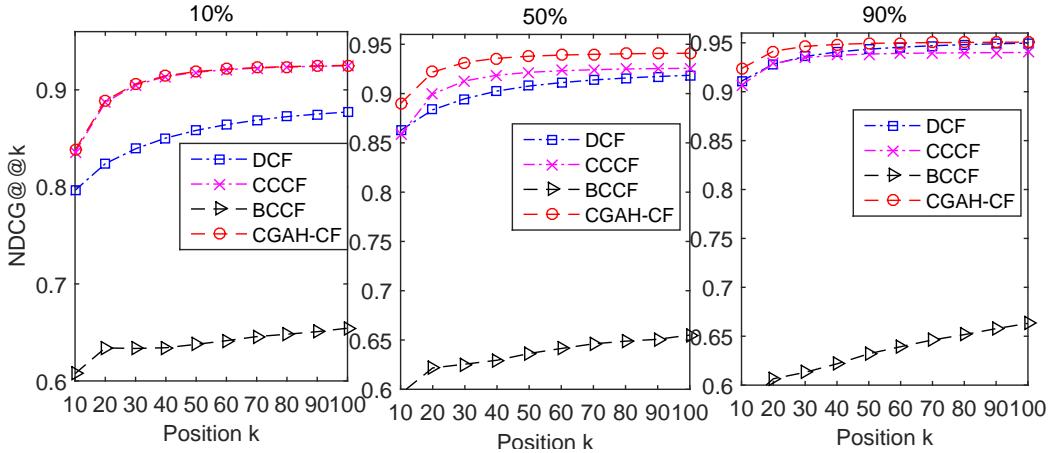


Fig. 5. The accuracy performance for collaborative filtering on CDs dataset with different sparse settings (10%, 50%, 90%).

where S is the group affinity computed by Equation (6) and (8).

Figure 6 shows the performance of MF-GA is better than the original MF. The superiority becomes more evident when two recommender systems recommend more items to users, as the difference of NDCG@ k between MF-GA and MF becomes obvious with the rise of k . Which indicates that the group affinity is effective to provide more accurate recommendations.

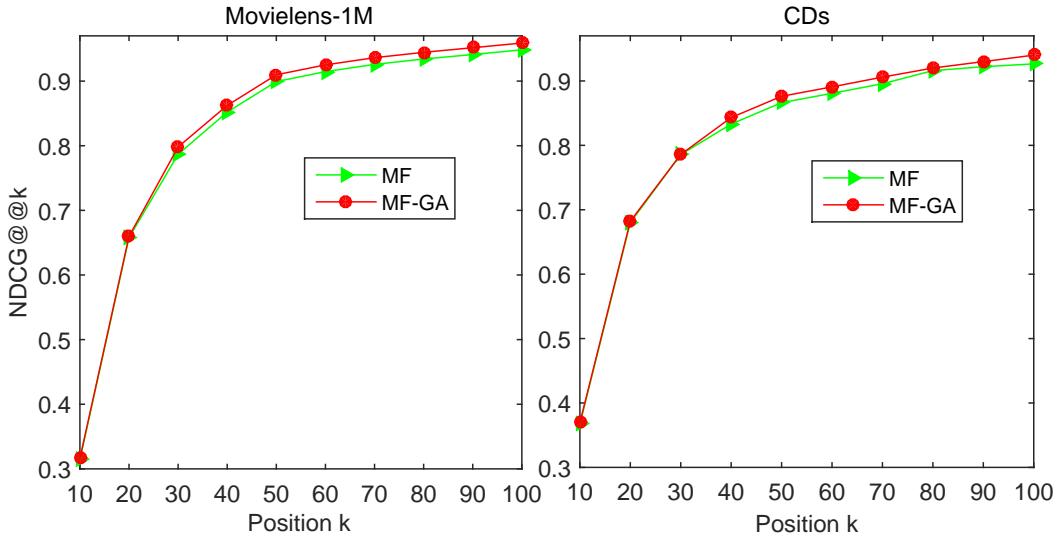


Fig. 6. Ablation studies of the group affinity on MF with the sparse setting 90%.

6 CONCLUSION

This paper proposes a framework named Collaborative Group-Aware Hashing (CGAH) for both collaborative filtering (CGAH-CF) and content-aware recommendations (CGAH), which not only

Table 4. The effectiveness of group affinity for MF. (MF-GA: MF with group affinity) on Books dataset

NDCG@k	10	20	30	40	50
MF	0.3286	0.6390	0.7705	0.8402	0.8754
MF-GA	0.3428	0.6492	0.7832	0.8528	0.8851

gains the group information from ratings, but also obtains a shared codebook of users and items, which is essential for enhancing the representation capability of hash codes. To the best of our knowledge, this is the first paper to explicitly capture the group affinities and integrate them with ratings into a model, which makes it possible to learn more effective representations of users and items for accurate recommendations.

Specifically, CGAH firstly extracts the group information of users and items in r -dimensional latent space, and then represents each user/item with these group centroids. The group centroids serve as a basis in the r -dimensional latent space. Then the group affinity is calculated by a function of distance discrepancy in the latent space. Finally, the preference is modeled as the product of the group affinity and the similarity of hash codes in r -dimensional Hamming space. By learning hash codes with the inherent group information, CGAH obtains more effective hash codes than other discrete methods under different sparse environments. Extensive experiments on three public datasets show the superiority of CGAH for discrete collaborative filtering and content-aware recommendations over competing baselines.

ACKNOWLEDGMENTS

This work is supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, the National Natural Science Foundation of China under Grant No. 62002052 and the Sichuan Science and Technology Program under Grant No. 2022YFG0189.

REFERENCES

- [1] D Agalya and V Subramaniyaswamy. 2016. Group-Aware recommendation using random forest classification for sparsity problem. *Indian J. Sci Technol* 9 (2016), 48.
- [2] Waqar Ali, Muhammad Ammad-Ud-Din, Xiangmin Zhou, Yan Zhang, and Jie Shao. 2025. Communication-efficient federated neural collaborative filtering with multi-armed bandits. *ACM Transactions on Recommender Systems* 4, 1 (2025), 1–28.
- [3] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 645–654.
- [4] Jingtao Ding, Guanghui Yu, Yong Li, Depeng Jin, and Hui Gao. 2019. Learning from hometown and current city: Cross-city POI recommendation via interest drift and transfer learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–28.
- [5] Guibing Guo, Enneng Yang, Li Shen, Xiaochun Yang, and Xiaodong He. 2019. Discrete Trust-aware Matrix Factorization for Fast Recommendation.. In *IJCAI*. 1380–1386.
- [6] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. 2020. Content-aware Neural Hashing for Cold-start Recommendation. *arXiv preprint arXiv:2006.00617* (2020).
- [7] Ming He, Jiuling Zhang, Peng Yang, and Kaisheng Yao. 2018. Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 225–233.
- [8] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.
- [9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.
- [10] Lin Hu, Junxiang Peng, Yan Zhang, Hong Wen, Shuai Tan, and Jiabing Fan. 2022. Artificial noise assisted interference alignment for physical layer security enhancement. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 4148–4153.

- [11] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.
- [12] 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.
- [13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
- [14] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *International conference on machine learning*. PMLR, 82–90.
- [15] Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. 2017. Discrete Content-aware Matrix Factorization. In *Proceedings of KDD’17*. ACM, 325–334.
- [16] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. LightRec: A Memory and Search-Efficient Recommender System. In *Proceedings of The Web Conference 2020*. 695–705.
- [17] Defu Lian, Zhenyu Zhang, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, and Xing Xie. 2016. Regularized content-aware tensor factorization meets temporal-aware location recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 1029–1034.
- [18] Ruxia Liang, Qian Zhang, and Jianqiang Wang. 2021. Hierarchical Fuzzy Graph Attention Network for Group Recommendation. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 1–6.
- [19] Chenghao Liu, Tao Lu, Xin Wang, Zhiyong Cheng, Jianling Sun, and Steven CH Hoi. 2019. Compositional coding for collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 145–154.
- [20] Chenghao Liu, Xin Wang, Tao Lu, Wenwu Zhu, Jianling Sun, and Steven Hoi. 2019. Discrete social recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 208–215.
- [21] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. Daml: Dual attention mutual learning between ratings and reviews for item recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 344–352.
- [22] Zhi Lu, Yang Hu, Yunchao Jiang, Yan Chen, and Bing Zeng. 2019. Learning binary code for personalized fashion recommendation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10562–10570.
- [23] 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.
- [24] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research* 11, 1 (2010).
- [25] Hanh TH Nguyen, Martin Wistuba, and Lars Schmidt-Thieme. 2017. Personalized tag recommendation for images using deep transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 705–720.
- [26] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI’09*. AUAI Press, 452–461.
- [27] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. 37–45.
- [28] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2289–2298.
- [29] 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.
- [30] 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.
- [31] Haoyu Wang, Nan Shao, and Defu Lian. 2019. Adversarial binary collaborative filtering for implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5248–5255.
- [32] 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.
- [33] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE TPAMI* 34, 12 (2012), 2393–2406.
- [34] Wen Wang, Wei Zhang, Jun Rao, Zhijie Qiu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Group-aware long-and short-term graph representation learning for sequential group recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1449–1458.
- [35] Xin Wang, Wei Lu, Martin Ester, Can Wang, and Chun Chen. 2016. Social recommendation with strong and weak ties. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 5–14.

- [36] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. 2007. Maximum margin matrix factorization for collaborative ranking. *Proceedings of NIPS'07* (2007), 1–8.
- [37] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. 2008. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*. 1593–1600.
- [38] Bin Wu, Tianren Shi, Lihong Zhong, Yan Zhang, and Yangdong Ye. 2023. Graph-coupled time interval network for sequential recommendation. *Information Sciences* 648 (2023), 119510.
- [39] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. 2002. Distance metric learning with application to clustering with side-information. In *NIPS*, Vol. 15. Citeseer, 12.
- [40] Qidi Xu, Fumin Shen, Li Liu, and Heng Tao Shen. 2018. Graphcar: Content-aware multimedia recommendation with graph autoencoder. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 981–984.
- [41] Yang Xu, Lei Zhu, Zhiyong Cheng, Jingjing Li, and Jiande Sun. 2020. Multi-Feature Discrete Collaborative Filtering for Fast Cold-Start Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 270–278.
- [42] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.
- [43] 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.
- [44] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huambo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of SIGIR'16*, Vol. 16.
- [45] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.
- [46] Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- [47] Yan Zhang, Haoyu Wang, Defu Lian, Ivor W Tsang, Hongzhi Yin, and Guowu Yang. 2018. Discrete ranking-based matrix factorization with self-paced learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2758–2767.
- [48] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 717–726.
- [49] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. 2014. Preference preserving hashing for efficient recommendation. In *Proceedings of SIGIR'14*. ACM, 183–192.
- [50] Huan Zhao, Quanming Yao, James T Kwok, and Dik Lun Lee. 2017. Collaborative filtering with social local models. In *2017 IEEE international conference on data mining (ICDM)*. IEEE, 645–654.
- [51] Lili Zhao, Sinno Jialin Pan, and Qiang Yang. 2017. A unified framework of active transfer learning for cross-system recommendation. *Artificial Intelligence* 245 (2017), 38–55.
- [52] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*. 425–434.
- [53] Ke Zhou and Hongyuan Zha. 2012. Learning binary codes for collaborative filtering. In *Proceedings of KDD'12*. ACM, 498–506.