This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3185093

Explainable Discrete Collaborative Filtering

Lei Zhu, Yang Xu, Jingjing Li, Weili Guan, Zhiyong Cheng

Abstract—Using hashing to learn the binary codes of users and items significantly improves the efficiency and reduces the space consumption of the recommender system. However, existing hashing-based recommender systems remain black boxes without any explainable outputs that illustrate why the system recommends the items. In this paper, we present a new end-to-end discrete recommendation framework based on the multi-task learning to simultaneously perform explainable and efficient recommendation. Toward this goal, an Explainable Discrete Collaborative Filtering (EDCF) method is proposed to preserve the user-item interaction features and semantic text features into binary hash codes by adaptively exploiting the correlations between the preference prediction task and the explanation generation task. At the online recommendation stage, EDCF makes efficient top-K recommendation by calculating the Hamming distances between the feature hash codes, and simultaneously generates natural language explanations for recommendation results through the explanation generation module. To obtain the hash codes directly from the end-to-end neural network, we introduce an attentive TextCNN and an Adaptive Tanh layer in the preference prediction task. For explanation generation, Long Short-Term Memory is employed to generate the explanations for recommendation results from the binary hash codes of user and item. Experiments demonstrate the superiority of the proposed method.

Index Terms—Discrete collaborative filtering, Efficient recommendation, Explainable recommendation

1 Introduction

In the past few decades, with the rapid development of internet applications and the explosive growth of Internet contents, more and more users have been adopted recommender systems as the main entrance to obtain Internet information. For example, most users are used to browsing "hot and vital" or "recommended for you" topics to quickly get the news, short videos, social information, and shopping products they are interested in. However, it is more challenging than ever before to provide recommendations immediately with a dramatically growing number of users and items for the recommender systems.

Hashing is an effective technique to address this challenge due to its desirable advantages in terms of similarity computation and storage efficiency. It has been successfully applied in various research fields, such as computer vision [1, 2], information retrieval [3–7], and data mining [8, 9]. In recent years, researchers have paid more attention to applying hashing technology to the recommender systems, and we refer to such methods as hashing-based recommendations [10–13]. The core idea of hashing-based recommendation is converting the original features of users and items into low-dimensional binary hash codes, where the Hamming similarities between users and items can be computed very efficiently using an Exclusive-Or operation to provide efficient top-K recommendation.

As a critical class of hashing-based recommendation method, Discrete Collaborative Filtering (DCF) [14], as exemplified by Discrete Matrix Factorization (DMF) [10] is receiving increasing attention from both academia and industry. Given an $n \times m$ user-item rating matrix, DMF projects both users and items into an r-dimensional Hamming space, where each user and item are represented by r-bit binary hash codes. With the transformation, users' preference scores for items are predicted by the Hamming similarities between their hash codes. Compared with real-valued feature representations, binary hash codes can be tens of times faster on similarity computation [15], which applies to large-scale recommender systems. Inspired by the success of DCF, many hashing-based recommendation methods have been proposed. Representative works include Discrete Personalized Ranking (DPR) [16], Discrete Deep Learning (DDL) [17], Discrete Factorization Machines (DFM) [11], and Discrete Content-aware Matrix Factorization (DCMF) [18], etc.

Although existing hashing-based recommendation methods have shown promising performance in terms of recommendation accuracy and efficiency, they still suffer from an important limitation in practical applications. That is, existing hashing-based recommendation methods are still black boxes and cannot provide users with reasonable explanations for the recommendation results. Many studies have shown that providing explainability for recommendation results not only allows system designers to understand the working state of the system, but also effectively improves users' trust in the recommender systems and helps them make fast and accurate decisions [19, 20]. In addition, a growing number of countries are enacting laws and regulations[1] that explicitly require recommender systems to be sufficiently explainable.

The mainstream e-commerce and crowd-sourced review

- *L. Zhu and Y. Xu are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China. Code is available at https://github.com/zzmylq/EDCF. Y. Xu is the corresponding author.*
- *J. Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.*
- *W. Guan is with the Faculty of Information Technology, Monash University, Melbourne, Australia.*
- *Z. Cheng is with Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences).*

1. https://bit.ly/38GOqqr
https://bit.ly/378XO64
https://bit.ly/3KLyCAp

websites, such as Taobao[2] and Dianping[3], allow users to rate and comment on the purchased products or services. The textual reviews usually contain rich information about product features and user preferences. Existing content-aware hashing-based recommendation methods [10, 11, 17, 18] tend to exploit reviews to enhance the prediction accuracy, while ignoring to effectively use semantic information in the reviews to generate explanations for the recommendation results. In this paper, we aim at developing an efficient recommendation model that can preserve the semantic information in the reviews into hash codes for effective and efficient top-K recommendation, and more importantly, it can simultaneously generate natural language explanations for the top-K recommendation results by using the hash codes as input.

Motivated by the above analyses, an Explainable Discrete Collaborative Filtering (EDCF) method is proposed in this paper, which designs a multi-task learning framework to fully exploit the correlations between the preference prediction task and the explanation generation task, so that the two tasks reinforce each other in the learning process and achieve a more effective learning objective than any single task.

For the explanation generation task, generating natural language explanations only based on the hash codes of user and item is a difficult challenge. To address this challenge, in this paper, we employ Long Short-Term Memory (LSTM) [21] to translate the binary hash codes into sentences that can express the interests and feelings of users. Specifically, we design an end-to-end deep neural network for preference prediction and simultaneously utilize a TextCNN to extract sentiment features from user reviews. Moreover, we introduce a review-level attention mechanism to model the contributions of each review, as well as the usefulness to the users and items. Specifically, to achieve end-to-end binarization, a multiplayer perceptron network and an Adaptive Tanh (ATanh) layer are integrated to project the semantic features and user-item interaction to a joint low-dimensional Hamming space. Thus, all the neural parameters in both deep learning tasks as well as the hash codes for users and items are learned by a multi-task learning approach in an end-to-end training paradigm.

Remarkably, the performance of a multi-task learning system is strongly dependent on the weights between each task, and manually adjusting these weights is a difficult and expensive process [22]. In this paper, we consider the homomorphic uncertainty of each task to weigh the loss functions of the two tasks and design a principled loss function, which can automatically learn the relative weights from the data and has the robustness to the weight initialization. To the best of our knowledge, EDCF is the first hashing-based recommendation method that simultaneously considers the semantic features of hash codes for improving the prediction accuracy and generating natural language explanations for the recommendation results. The main contributions of this paper are summarized as follows:

- We propose an Explainable Discrete Collaborative Filtering (EDCF) method for the explainable efficient recommendation. EDCF can generate hash codes for users and items to enable efficient top-K recommendation, while using hash codes only as the input to generate natural language explanations for recommendation results. To the best of our knowledge, there is still no similar work.
- We propose a multi-task learning framework based end-to-end neural network architecture, which exploits the TextCNN and the attention mechanism to extract the semantic features from the textual reviews, and generates the hash codes of the users and items by fully exploiting the correlations between preference prediction task and explanation generation task.
- Instead of adopting the fixed task weights to train the neural network, we design a principled loss function to automatically adjust the task weights according to the difficulty of each task and the model state. The evaluation on experimental datasets shows that this strategy allows EDCF to prioritize learning easier task and gradually tackle the difficult task at a subsequent stage by adaptively adjusting the task weights.
- We conduct extensive experiments on four public datasets, and employ various widely tested evaluation metrics to measure the performance of preference prediction and explanation generation of EDCF. The evaluation results show the advantage of EDCF on improving recommendation accuracy and generating explanations compared with the baseline methods. We also verify and analyze the effectiveness of each module in EDCF experimentally.

2 Related Work

In this paper, we aim at generating natural language explanations for the recommendation results while achieving an efficient recommendation. Hence, in this section, we mainly review the recent advanced hashing-based recommendation methods and the explainable recommender systems based on natural language generation technology. For more details of hashing and explainable recommendation, please refer to [23] and [24], respectively.

2.1 Hashing-based Recommendation

Early studies of hashing-based recommendation frameworks focus on learning hash codes by employing a two-step optimization strategy, which consists of a relaxed optimization step and a binary quantization step. Under this strategy, the real-valued representations of users and items are first learned by the relaxed optimization, and then converted to binary codes by the quantization. A pioneering work in this research area is [25], which employs Locality-Sensitive Hashing (LSH) [26] and Jaccard measure method to learn binary codes for efficiently searching Google News. Based on this, Karatzoglou [27] generates binary codes from real-valued representations of users and items by employing random projections. [28] follows the idea of Iterative Quantization (ITQ) [29], which is widely used in the field of hashing-based image retrieval, to rotate and binarize real-valued latent representations of users and items. To improve recommendation performance, the decorrelated constraint [30] and constant feature norm constraint [31] are imposed

2. https://www.taobao.com/
3. https://www.dianping.com/

on real-valued latent factors of users and items before quantization. These two-step optimization strategies can learn binary hash codes for users and items. But they can easily bring significant quantization errors [14].

To alleviate quantization loss, Discrete Collaborative Filtering (DCF) [14] employs the Discrete Coordinate Descent (DCD) [32] for the first time to learn the hash codes of users and items directly, rather than through the two-step optimization strategy. DCF adds balanced and decorrelated constraints to matrix factorization formulation and learns hash codes using only rating information. Inspired by DCF, Discrete Content-aware Matrix Factorization (DCMF) [18] is the first hashing-based recommendation method that considers side information to improve recommendation performance and support cold-start recommendation scenarios. DCMF learns the hash codes directly by DCD method. The content features in the side information are obtained by optimizing a multi-objective loss function and are preserved in the hash codes. This strategy allows DCMF to learn the hash codes of users or items in a cold-start setting. Similar to DCMF, Discrete Deep Learning (DDL) [17] extracts content features of items from side information by employing Deep Belief Network (DBN) [33], and generates hash codes by solving a relaxed optimization problem with an alternating optimization strategy. Discrete Factorization Machines (DFM) [11] applies the factorization machines model to exploit the pair-wise correlations between content features and generate hash codes. Discrete Trust-aware Matrix Factorization (DTMF) [34] and Discrete Social Recommendation (DSR) [35] learn hash codes by reconstructing the rating and social relationship.

Recently, deep learning techniques demonstrate promising performance in hashing-based recommendation tasks [12, 13, 36]. For instance, DGCN-BinCF [12] utilizes Graph Convolutional Network (GCN) [37] to model high-order features of users and items, and distills the ranking information derived from GCN into binarized collaborative filtering to generate hash codes. Due to the success of Generative Adversarial Networks (GAN) [38], [13] applies GAN to the hashing-based recommendation task, and proposes an Adversarial Binary Collaborative Filtering framework (ABinCF), which is optimized by approximating sign function and Bernoulli distribution. [36] investigates the problem of unsupervised deep hashing with graph neural network for recommendation, and proposes a framework called HashGNN, which simultaneously learns deep hash functions and graph representations in an end-to-end manner. These methods based on deep learning and hashing technique can obtain more discriminative hash codes, but they mainly rely on the user-item interactions, without considering the rich content features of users and items. In addition, existing hashing-based recommendation algorithms are still black boxes, and they cannot generate explanations for the recommendation results.

2.2 Generation-based Explainable Recommendation

Generation-based explainable recommendation methods not only provide users with recommendation results, but also generate sentence explanations to clarify why such items are recommended. Based on natural language generation models, the recommender systems can automatically generate explanation sentences for the recommendation results. For example, [39] proposes a method for automatically generating natural language explanations based on LSTM. The model uses user ratings as auxiliary information to train the explanation generation module, so that the model can generate reviews with the corresponding sentiment based on the expected ratings. Inspired by how people explain word-of-mouth recommendations, [40] proposes a method to combine crowdsourcing and computation to generate personalized natural language explanations. The authors extract the topical aspects based on an unsupervised learning approach, and then generate natural language explanations for the topical aspects.

TABLE 1
Main notations used in this paper.

Notation	Description
$\boldsymbol{C}_u$	the user u's review document
$\boldsymbol{\mathcal{W}}_{ij}$	the explanation of recommending item v_j to user u_i
$\boldsymbol{R}$	the user-item rating matrix
$\boldsymbol{P}$	the interaction feature matrix of users
$\boldsymbol{Q}$	the interaction feature matrix of items
$\hat{\boldsymbol{S}}_{\boldsymbol{u}}$	the learned review feature of user u
$\boldsymbol{B}$	the binary hash code matrix of n users
$\boldsymbol{D}$	the binary hash code matrix of m items
$\boldsymbol{\mathcal{V}}$	the vocabulary of words in the reviews and explanations
n	the number of users
m	the number of items
α	the learnable scaling parameter in ATanh layer
φ_1, φ_2	the uncertainty in the two tasks

3 The Proposed Method

3.1 Overview

Due to the success of multi-task learning technique in academia and industry, it is widely applied to generation-based explainable recommendation tasks. [41] proposes a Hierarchical Sequence-to-Sequence (HSS) model based on multi-task learning for personalized recommendation and natural language explanation generation, which adopts an auto-denoising mechanism that selects sentences containing item features for model training. [42] proposes a deep framework named NRT based on multi-task learning which can simultaneously predict ratings and leverage gated recurrent neural networks to generate abstractive tips for the recommendation results. [43] proposes a multi-task recommendation model, which integrates a sequence-to-sequence learning model with matrix factorization and jointly performs rating prediction and recommendation explanation. [44] proposes an encoder-selector-decoder architecture for explainable recommendation inspired by human's information-processing model in cognitive psychology. The authors exploit the correlations between the recommendation task and the explanation task through co-attentive multi-task learning. These approaches utilize a multi-task learning strategy to generate natural language explanations of the recommendation results while predicting the ratings. However, in the large-scale recommendation scenario, these approaches still consume considerable computation time to generate top-K recommendations from the enormous number of candidate items.

Different from existing explainable recommendation algorithms, the proposed EDCF method has the following advantages. First, EDCF can generate binary representations

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3185093

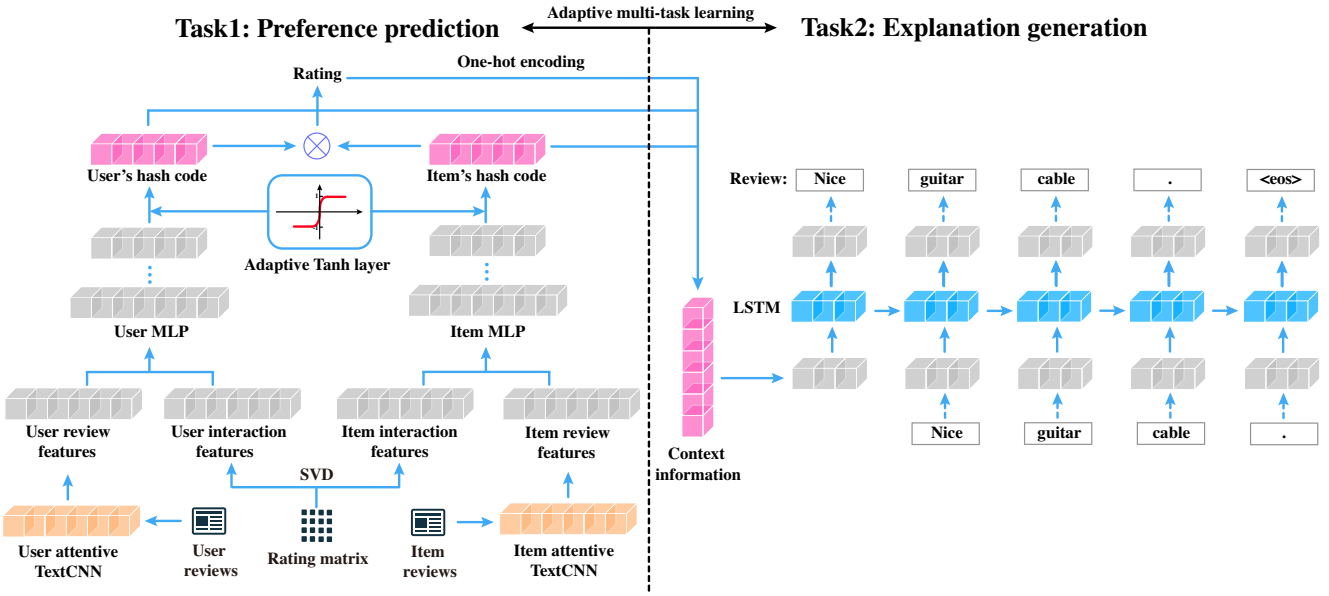


Fig. 1. The basic learning framework of the proposed EDCF.

of users and items to support efficient top-K recommendation, while translating the hash codes into personalized natural language explanations. Second, EDCF utilizes attentive TextCNN and ATanh layer to achieve end-to-end binary optimization for multiple tasks. Finally, EDCF can automatically adjust the multi-task weights during model training according to the task difficulty and model state.

The goal of our model is to learn the hash codes of users and items, as well as to train an explanation generation module that can translate the hash codes into natural language explanation. To this end, we introduce an Explainable Discrete Collaborative Filtering (EDCF) model for efficient explainable recommendation. The architecture of the proposed model is shown in Fig.1. The model is based on a multi-task learning strategy and contains two major components: neural discrete collaborative filtering on the left and natural language explanation generation on the right, which correspond to preference prediction task and explanation generation task respectively.

For neural discrete collaborative filtering, the input consists of the user set $\boldsymbol{U}$, the item set $\boldsymbol{V}$, the ratings and the reviews. Each user is represented as its ID $u \in \boldsymbol{U}$ and each item is denoted as the item ID $v \in \boldsymbol{V}$. Ratings are normalized through dividing by their maximum value. The reviews posted by the user u are represented as a document $\boldsymbol{C}_u = (\boldsymbol{C}_u^1, \boldsymbol{C}_u^2, ..., \boldsymbol{C}_u^{m_d})$, where m_d denotes the maximum number of reviews. If the number of reviews posted by the user is larger than m_d, a random selection strategy is applied to form $\boldsymbol{C}_u$. Each review $\boldsymbol{C}_u^i$ is denoted by a set of words in the review. Similarly, document $\boldsymbol{C}_v = (\boldsymbol{C}_v^1, \boldsymbol{C}_v^2, ..., \boldsymbol{C}_v^{m_d})$ denotes the set of reviews received by the item v.

Given user u's review document $\boldsymbol{C}_u$, the user attentive TextCNN module embeds the words, reviews and user ID, and generates review feature representation of user u. In addition, we obtain user u's interaction features from the rating matrix using the Singular Value Decomposition (SVD) method. And then, a Multi-Layer Perceptron (MLP) is employed to project the concatenation of interaction features and review features of user u to a real-valued representation via several layers of non-linear transformations. Specifically, we adopt an ATanh layer to transform the user's real-valued representation into a binary hash code. The same process is applied for item modeling network with similar layers. Then, we utilize Hamming similarity for preference prediction. The Hamming similarity between user u_i and item v_j is calculated as:

$$Hsim(u_i, v_j) = \frac{1}{2} + \frac{1}{2r}\boldsymbol{b}_i^T \boldsymbol{d}_j, \tag{1}$$

where r is the hash code length, $\boldsymbol{b}_i \in \{-1, 1\}^{r\times 1}$ and $\boldsymbol{d}_j \in \{-1, 1\}^{r\times 1}$ are the hash codes of user u_i and item v_j, respectively. For natural language explanation generation, a sequence decoding model based on LSTM is proposed to translate the hash codes of u_i and v_j into a sequence of words $\boldsymbol{\mathcal{W}}_{ij} = (w_1, w_2, ..., w_{m_e})$, which illustrates why user u_i likes or dislikes item v_j. w_k denotes the k-th word in the explanation, and m_e represents the maximum number of words in the generated natural language explanation. Moreover, $Hsim(u_i, v_j)$ is used as a part of the input for the decoding model to control the sentiment of the generated explanations. At the online recommendation stage, there is a significant efficiency difference between the neural network computing and the hash code matching. To address this issue, we first obtain the top-K recommendation results from the massive candidate items by matching the hash codes of the target user and candidate items rapidly, and then the hash codes of the target user and top-K items are fed into the explanation generation network to generate natural language explanation for the recommendation results. All the neural parameters and the hash codes of users and items are learnt by a multi-task learning approach. The model can be trained efficiently by an end-to-end learning paradigm using back-propagation algorithms.

Throughout this paper, we use bold lowercase letters to represent vectors and bold uppercase letters to represent matrices. All of the vectors in this paper denote column vectors. Non-bold letters represent scalars. Main notations used in this paper are listed in Table 1.

3.2 Neural Discrete Collaborative Filtering

As shown in Fig.1, the neural discrete collaborative filtering component consists of two parallel neural networks: user modeling network and item modeling network. In

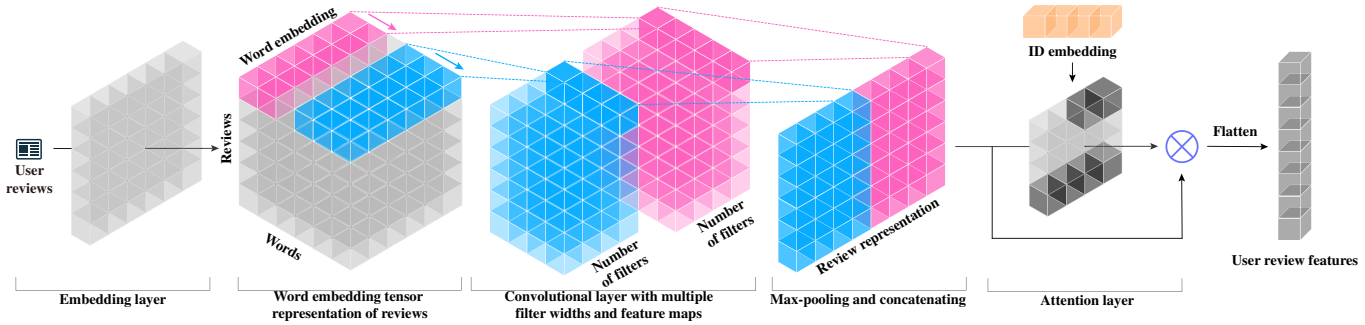


Fig. 2. The attentive TextCNN architecture.

the following, we focus on illustrating the process for user modeling network in detail. The same process is applicable to item modeling network with similar layers.

At the first stage of user modeling network, given a user review document $\boldsymbol{C}_u$, an attentive TextCNN network is applied to process $\boldsymbol{C}_u$. Fig.2 gives the architecture of the attentive TextCNN. In the first layer, a word embedding function is used to map each word in the user review document $\boldsymbol{C}_u$ to a d-dimensional word embedding vector, and then the given $\boldsymbol{C}_u$ is transformed to a fixed-size embedding tensor $\boldsymbol{V}_u \in \mathbb{R}^{m_d \times m_w \times d}$, where m_w denotes the maximum number of words in each review (padded with zero in $\boldsymbol{C}_u$ when the review length is less than m_w). Following the embedding layer is the convolution layer, where N_f filters $\boldsymbol{w}_i \in \mathbb{R}^{h \times d}, i \in \{1, 2, ..., N_f\}$ with a window of h words are applied to extract features by convolution operations on the word vectors. Let $\boldsymbol{V}_u^k$ denote the embedding matrix corresponding to the k-th review of the user u. Then, the features generated by the i-th filter $\boldsymbol{w}_i$ can be expressed as:

$$\boldsymbol{z}_u^{ik} = f(\boldsymbol{w}_i \otimes \boldsymbol{V}_u^k + \boldsymbol{\mu}_i), \tag{2}$$

where $\boldsymbol{\mu}_i$ is a bias term, $f(\cdot)$ denotes a non-linear activation function such as ReLU [45], and $\otimes$ represents the convolution operation. Then, we apply a max-pooling operation [46] over the features $\boldsymbol{z}_u^{ik}$ and take the maximum value $\hat{z_u^{ik}} = max\{\boldsymbol{z}_u^{ik}\}$ as the extracted feature of the k-th review corresponding to the filter $\boldsymbol{w}_i$. The idea of max-pooling is to capture the most important feature-one with the highest value. We concatenate the features extracted by these N_f filters, denoted as

$$\boldsymbol{s}_u^k = [\hat{z_u^{1}}^k, \hat{z_u^{2}}^k, ..., \hat{z_u^{N_f}}^k]^T. \tag{3}$$

In addition, the proposed model employs N_g sets of filters with different window sizes to extract multiple review features. The representation of the k-th review of the given user is the concatenation of multiple review features obtained by N_g sets of filters with varying window sizes, denoted by:

$$\boldsymbol{S}_u^k = [\boldsymbol{s}_u^{k,1}; \boldsymbol{s}_u^{k,2}; ...; \boldsymbol{s}_u^{k,N_g}]. \tag{4}$$

The final output of the convolutional layer is

$$\boldsymbol{S}_u = [\boldsymbol{S}_u^1, \boldsymbol{S}_u^2, ..., \boldsymbol{S}_u^{m_d}]. \tag{5}$$

Existing works tend to improve the model performance by exploiting latent features in the user's textual reviews [42, 47]. However, in practice, not all reviews are of high quality and have high availability. Currently, most studies do not consider the difference in the contribution of each review to user and item modeling, which is not robust in real life as each review has a different ability to represent user interests and item features. To address this issue, we introduce an attention mechanism in our model, which can help to learn the weight of each review in user and item modeling.

We employ a two-layer network to compute attention score a_u^k. The input to the attention layer in the user modeling network contains two components: a feature matrix composed of m_d review features of the given user and embedding representations of the item IDs $\boldsymbol{c}_u^k$. Intuitively, the ID embedding is added to identify the items that are not distinctive and have low reference value. For the item modeling network, user ID embedding is applied to the network to identify users who frequently post spam reviews. We formulate this part as

$$\hat{a}_u^k = \boldsymbol{W}_h^T f(\boldsymbol{W}_s \boldsymbol{S}_u^k + \boldsymbol{W}_u \boldsymbol{c}_u^k + \boldsymbol{\mu}_1) + \mu_2. \tag{6}$$

where $\boldsymbol{W}_h \in \mathbb{R}^h$, $\boldsymbol{W}_s \in \mathbb{R}^{h \times (N_f \times N_g)}$, $\boldsymbol{W}_u \in \mathbb{R}^{I_d \times h}$, $\boldsymbol{\mu}_1 \in \mathbb{R}^h, \mu_2 \in \mathbb{R}^1$ are model parameters, I_d is the dimension of ID embedding, h represents the hidden layer size of the attention network. The final attention score of the reviews can be obtained by using the softmax function to normalize the above scores, which can be interpreted as the contribution of the i-th review to model the preference of user u:

$$a_u^k = \frac{exp(\hat{a}_u^k)}{\sum_{k=1}^{m_d} exp(\hat{a}_u^k)}. \tag{7}$$

After that, the review feature of user u can be represented as a weighted concatenation as follows:

$$\hat{\boldsymbol{S}}_u = [a_u^1 \boldsymbol{S}_u^1; a_u^2 \boldsymbol{S}_u^2; ...; a_u^{m_d} \boldsymbol{S}_u^{m_d}]. \tag{8}$$

Specifically, we consider both user review features and interaction features in the model. Given a rating matrix $\boldsymbol{R}$ of size $n \times m$, where n and m are the numbers of users and items, respectively. The user interaction features are obtained by SVD method:

$$\boldsymbol{R} = \boldsymbol{P}\boldsymbol{Q}^T, \tag{9}$$

where the eigenvalue matrix is left blended into the feature vectors, $\boldsymbol{P} \in \mathbb{R}^{n \times l}$ denotes the interaction feature matrix of users, where the i-th row $\boldsymbol{p}_i$ is the interaction feature of the user u_i, $\boldsymbol{Q} \in \mathbb{R}^{m \times l}$ represents the interaction feature matrix of items, where the j-th row $\boldsymbol{q}_j$ denotes the interaction feature of the item v_j, l denotes the interaction feature length.

At the offline stage, we learn and store the hash codes of users and items. At the online recommendation stage, we make efficient top-K recommendations by matching the hash codes quickly. To this end, instead of using a neural latent factor model to predict ratings as in [48], we employ a multi-layer perceptron network to map the user's interaction feature and review feature into a shared latent space to obtain a real-valued representation of the user:

$$\begin{aligned} \boldsymbol{p}_i^1 &= [\boldsymbol{p}_i; \hat{\boldsymbol{S}_{u_i}}], \\ \boldsymbol{p}_i^2 &= \boldsymbol{W}_2^T \boldsymbol{p}_i^1 + \boldsymbol{b}_2, \\ &..... \\ \boldsymbol{p}_i^L &= \boldsymbol{W}_L^T \boldsymbol{p}_i^{(L-1)} + \boldsymbol{b}_L. \end{aligned} \tag{10}$$

In this paper, we aim to generate the binary representations $\boldsymbol{B}, \boldsymbol{D} \in \{-1, +1\}$ for the users and items, respectively. Since binary constraints are difficult to optimize in networks, we employ an Adaptive Tanh (ATanh) layer in our model, which is formulated as:

$$f(\boldsymbol{x}) = \tanh(\alpha \boldsymbol{x}), \alpha > 0, \tag{11}$$

where $\boldsymbol{x}$ is the activation of previous layers, α is a learnable scaling parameter. In practice, the learned α should be large enough to make the activations of ATanh fall into the binary value $\{-1, +1\}$. Since the ATanh is differentiable everywhere, it can be jointly trained with other layers via back-propagation, and directly generate the binary codes[4].

Then, the preference of user u_i for item v_j can be evaluated by Eq.(1). The loss function of the preference prediction module can be formulated as:

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{PP} = \sum_{(u_i, v_j) \in \boldsymbol{Y}^+ \cup \boldsymbol{Y}^-} & (\frac{r_{ij}}{\max(\boldsymbol{R})} \log Hsim(u_i, v_j) \\ & + (1 - \frac{r_{ij}}{\max(\boldsymbol{R})}) \log(1 - Hsim(u_i, v_j))), \end{aligned} \tag{12}$$

where r_{ij} indicates the rating of user u_i for item v_j, $\boldsymbol{Y}^+$ denotes the observed interactions, $\boldsymbol{Y}^-$ means the set of negative samples, which are randomly selected from the user-item pairs without observations in a certain proportion, $\boldsymbol{Y}^+ \cup \boldsymbol{Y}^-$ means all training interactions, $\max(\boldsymbol{R})$ represents the max score in all ratings, e.g., 5 in a 5-star rating system.

3.3 Natural Language Explanation Generation

At the online recommendation stage, the input is only the target user ID, and the binary hash code of the target user can be obtained from matrix $\boldsymbol{B}$. Then the top-K recommendation is fast generated by computing the Hamming distance between hash codes of the target user and the candidate items. In this paper, we aim to generate corresponding explanations while making top-K recommendations. Since the input does not contain any textual information, it is a challenging task to generate natural language explanations only based on the hash codes of the target user and the item.

Based on the above issues and considering the outstanding performance of Long Short-Term Memory (LSTM) model in text generation-related tasks, we adopt LSTM as the basic model for the explanation generation module. The right part of Fig.1 shows our explanation generation module, whose main idea is expressed as follows:

$$p(w_{ij}^t | w_{ij}^1, w_{ij}^2, ..., w_{ij}^{t-1}, \boldsymbol{\mathcal{C}}_{ij}) = f_h(\boldsymbol{h}_t), \tag{13}$$

where w_{ij}^t is the t-th word of the explanation $\boldsymbol{\mathcal{W}}_{ij}$, $\boldsymbol{\mathcal{C}}_{ij}$ denotes the context information about user u_i and item v_j, which will be described in the following section, f_h is a non-linear mapping function that decodes $\boldsymbol{h}_t$ into word w_{ij}^t, $\boldsymbol{h}_t$ is the hidden state at the time t and it depends on the input $\boldsymbol{x}_t$ at the time t and the previous hidden state $\boldsymbol{h}_{t-1}$:

$$\boldsymbol{h}_t = LSTM(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{14}$$

4. Although ATanh is very close to the sign function, there are still very few activations that are not -1 or +1. For these activations, we binarize them directly.

Sequence hidden state is updated based on the following operations:

$$\begin{aligned} \boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \otimes [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f), \\ \boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \otimes [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i), \\ \widetilde{\boldsymbol{C}}_t &= \tanh(\boldsymbol{W}_c \otimes [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c), \\ \boldsymbol{C}_t &= \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t, \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \otimes [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o), \\ \boldsymbol{h}_t &= \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t), \end{aligned} \tag{15}$$

where $\boldsymbol{x}_t$ is the embedding vector for the word e_t of the explanation and the vector is also obtained during the end-to-end learning process of our framework, $\boldsymbol{f}_t$ denotes the forget gate and $\boldsymbol{i}_t$ is the input gate, $\otimes$ represents element-wise multiplication and $\sigma(\cdot)$ means the softmax function.

As shown in Fig.1, when $t = 1$, the LSTM model has no word vector as an input. Therefore, we adopt the context information $\boldsymbol{\mathcal{C}}_{ij}$ as the input at $t = 1$. $\boldsymbol{\mathcal{C}}_{ij}$ will guide the explanation generation module to generate natural language explanation for the corresponding user-item pair, which will directly affect the performance of the recommender systems. In our framework, the context information $\boldsymbol{\mathcal{C}}_{ij}$ consists of predicted preference value $\hat{r}_{ij}$, and corresponding binary hash codes $\boldsymbol{b}_i$ and $\boldsymbol{d}_j$ of user u_i and item v_j, respectively.

For the hash codes $\boldsymbol{b}_i$ and $\boldsymbol{d}_j$, we can directly find them from the matrices $\boldsymbol{B}$ and $\boldsymbol{D}$, respectively. To control the sentiment tendency of the generated explanation, we add the predicted preference value $\hat{r}_{ij}$ to $\boldsymbol{\mathcal{C}}_{ij}$. Similar to NARRE [48], $\hat{r}_{ij}$ will be vectorized into its one-hot form. For example, $\hat{r}_{ij} = 0.34$, in the 5-star rating system, we scale it to the range $[0, 5]$ and round it, then we get the vector $\hat{\boldsymbol{r}}_{ij} = (0, 0, 0, 1, 0, 0)^T$.

After obtaining the context information $\boldsymbol{\mathcal{C}}_{ij} = [\boldsymbol{b}_i, \boldsymbol{d}_j, \hat{\boldsymbol{r}}_{ij}]$, we transform it into the initial input $\boldsymbol{x}_1^{ij}$ at $t = 1$ by a non-linear mapping:

$$\boldsymbol{x}_1^{ij} = \tanh(\boldsymbol{W}_{\mathcal{C}} \boldsymbol{\mathcal{C}}_{ij} + \boldsymbol{b}_{\mathcal{C}}). \tag{16}$$

where $\boldsymbol{W}_{\mathcal{C}}$ and $\boldsymbol{b}_{\mathcal{C}}$ are parameters to be learned.

Then, LSTM can perform sequence decoding and generate the hidden state sequence. The hidden state $\boldsymbol{h}_t^{ij}$ will be mapped into a $|\boldsymbol{\mathcal{V}}|$-size vector $\hat{\boldsymbol{e}}_t^{ij}$ through the final generation layer, where $\boldsymbol{\mathcal{V}}$ is the vocabulary of words in the reviews and explanations:

$$\hat{\boldsymbol{e}}_t^{ij} = \sigma(\boldsymbol{W}_e \boldsymbol{h}_t^{ij} + \boldsymbol{b}_e), \tag{17}$$

where $\boldsymbol{W}_e \in \mathbb{R}^{|\mathcal{V}| \times d_h}$ and $\boldsymbol{b}_e \in \mathbb{R}^{|\mathcal{V}|}$, d_h is dimensionality of the hidden state space, $\sigma(\cdot)$ denotes the softmax function. Then the word with the highest probability in step t is selected as the t-th word of the explanation:

$$\hat{w}_t^{ij} = \arg\max \hat{\boldsymbol{e}}_t^{ij}. \tag{18}$$

At the training stage, we employ Negative Log Loss (NLL) as the loss function for the explanation generation module:

$$\boldsymbol{\mathcal{L}}_{EG} = - \sum_{(u_i, v_j) \in \boldsymbol{Y}^+} \sum_{w \in \boldsymbol{\mathcal{W}}_{ij}} \log \hat{\boldsymbol{e}}_{ij}^{(I_w)}, \tag{19}$$

where I_w is the vocabulary index of the word w.

3.4 Adaptive Multi-task Learning

Traditional multi-task learning tends to simply conduct a weighted linear sum of the losses for each individual task, which usually requires manual adjustment of their weights. However, the learning performance of the model is very sensitive to the weights, and it is difficult to manually adjust

them to obtain a better model for multiple tasks simultaneously. In addition, manually adjusting these weights is a very time-consuming task in practice.

To address this problem, we introduce task-dependent uncertainty [22] for adaptive multi-task learning, and the overall objective function of the proposed model is as follows:

$$\mathcal{L} = \frac{1}{\varphi_1^2}\mathcal{L}_{PP} + \frac{1}{\varphi_2^2}\mathcal{L}_{EG} + \lambda_1 \log \varphi_1\varphi_2 + \lambda_2||\alpha^{-1}||_2^2 + \lambda_3||\Theta||_2^2, \tag{20}$$

where φ_1 and φ_2 denote the uncertainty in the two tasks, respectively, $||\alpha^{-1}||_2^2$ is a penalty term, which is a convenient way to make the α in Eq.(11) increase during the training process, Θ denotes the set of neural parameters, λ_1, λ_2 and λ_3 are regularization parameters. The whole framework can be effectively trained end-to-end using back-propagation.

Intuitively, the larger the $\varphi_1(\varphi_2)$, the greater the uncertainty of the task and the smaller the weight of the corresponding task. That is, during the training process, the model will preferentially learn tasks that are less noisy and easier to learn. However, it does not mean that the noisier and harder tasks are not emphasized. The effect of introducing uncertainty on model learning will be explored in detail in our experiments.

4 Experiments

4.1 Evaluation Datasets

We conduct the experiments on four well-known datasets. They have been widely tested in recent hashing-based recommendation methods [10, 11, 14, 17, 18, 49]. The ratings in these datasets are all based on the 5-star rating system. Specifically, three datasets are from Amazon 5-core[5]: **Kindle Store**, **Movies&TV** and **Electronics**. These datasets contain user ratings and reviews of various products, and all users and products correspond to at least 5 reviews. Another dataset is **Yelp**[6], which contains user ratings and reviews for locations such as restaurants, hotels, and shopping centers. Considering the sparse user reviews in the Yelp dataset, we remove the users with less than 5 reviews. After the filtering, there are 365,665 users, 159,108 items, as well as 5,766,970 ratings and reviews left in the Yelp dataset. For all these datasets, we filter out the stop words and low-frequency words from the reviews, and then build a vocabulary $\mathcal{V}$ for each dataset. The statistics of the datasets are summarized in Table 2.

TABLE 2
Statistics of experimental datasets.

Dataset	#User	#Item	#Rating	Sparsity	$\lvert\mathcal{V}\rvert$
Kindle Store	68,223	61,934	982,619	99.98%	44,656
Movies&TV	123,960	50,052	1,697,533	99.97%	69,434
Electronics	192,403	63,001	1,689,188	99.99%	50,023
Yelp	365,665	159,108	5,766,970	99.99%	83,553

4.2 Evaluation Baselines

To evaluate the performance of preference prediction, we compare our method with the following state-of-the-art baselines:

- **Discrete Collaborative Filtering (DCF)** [14] is the first binarized collaborative filtering method that can directly optimize the hash codes for users and items, which outperforms almost all two-stage hash code learning methods for collaborative filtering.
- **Discrete Content-aware Matrix Factorization (DCMF)** [18] is the state-of-the-art binarized method for CF with auxiliary information. It improves the discriminative ability of hash codes by encoding context information on the basis of DCF. The parameters λ_1 and λ_2 for modeling user and item auxiliary features are tuned within $\{1, 10, 50, 100, 500, 1000\}$.
- **Discrete Factorization Machines (DFM)** [11] is the first binarized factorization machines method to resolve the rating prediction problem. It binarizes the real-valued model parameters of each feature embedding into binary codes. In DFM, the parameter β for the softened de-correlation constraint is tuned within $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ according to the results of their sensitive analysis.
- **Discrete Deep Learning (DDL)** [17] is a binary deep recommendation approach. It uses a Deep Belief Network (DBN) to generate item hash codes from auxiliary information, and combines the DBN with DCF to solve the cold-start recommendation problem. The parameters α, β and λ are tuned within $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$. The layer structure of DBN is set as [8000, 800, 30].
- **Neural Hashing-based Collaborative Filtering (NeuHash-CF)** [49] is a deep hashing-based recommendation approach, which utilizes two joint autoencoder architectures to generate user and item hash codes from content information, respectively. This method uses only the user's ID to generate the user hash code and the auxiliary information of the item to generate the item hash code. This strategy makes it have better performance in both cold-start and general settings. We tune the parameters of this method as described in the corresponding paper.
- **Deep Matrix Factorization (DeepMF)** [50] is a matrix factorization model with a neural network structure. It uses a deep architecture to learn low-dimensional space representations of users and items from their explicit ratings and implicit feedback. We set the learning rate to 0.0001 and randomly initialize model parameters according to the original paper description.

To evaluate the performance of explanation generation, we compare EDCF with two explainable recommendation methods based on textual sentence explanation and a refined retrieval-based explanation generation method:

- **NARRE** [48] is a neural attentional regression model with review-level explanations for recommendation. It calculates the usefulness of user reviews while predicting ratings. NARRE selects useful reviews that provide detailed information about an item and valid purchase suggestions as an explanation of the recommendation results. The model parameters are tuned as described in the original text.
- **NRT** [42] is a generation-based explanation method. NRT uses a deep architecture to predict user ratings and utilizes a gated recurrent neural networks to

5. http://jmcauley.ucsd.edu/data/amazon
6. https://www.yelp.com/dataset/

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3185093

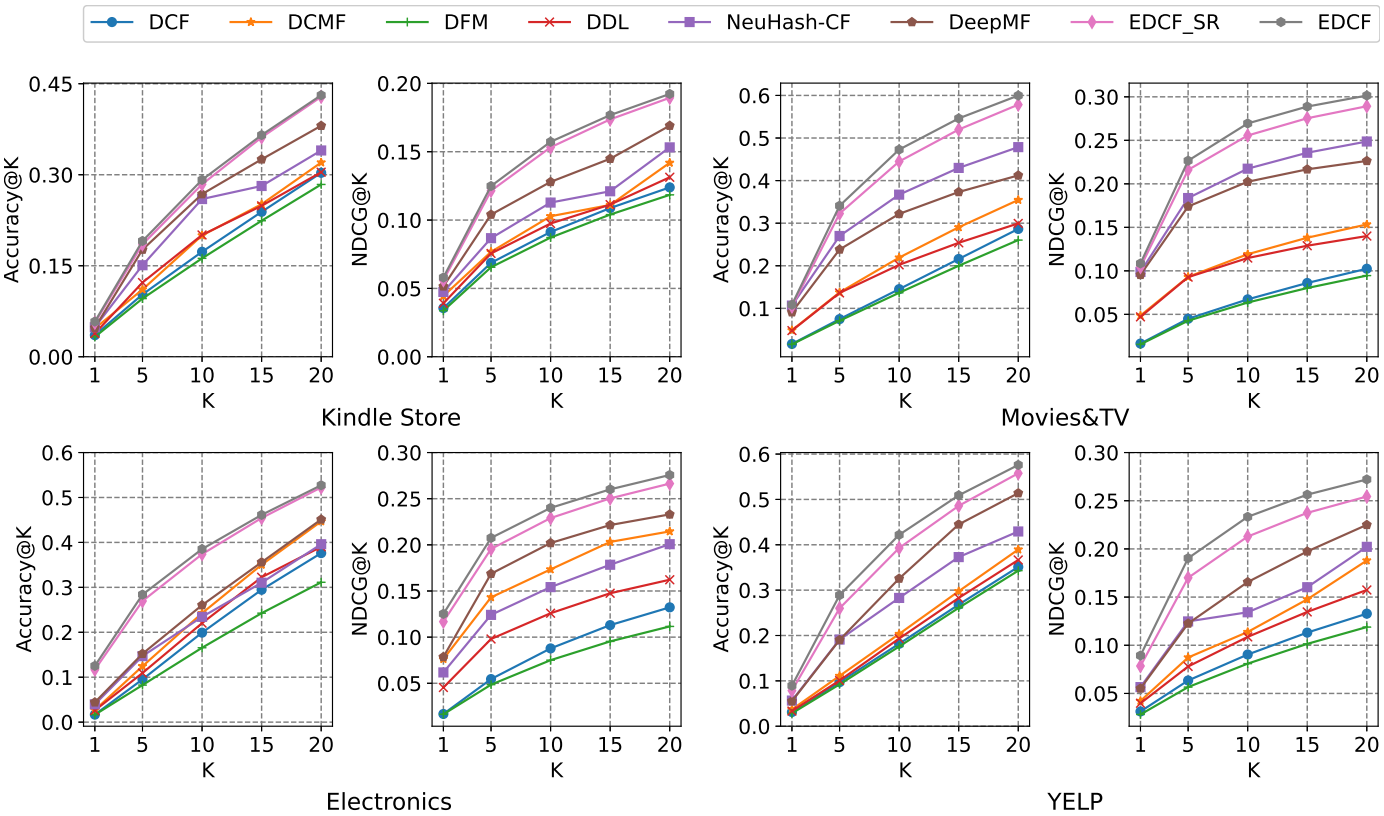


Fig. 3. Comparison of EDCF with baseline algorithms on Kindle Store, Movies&TV, Electronics and YELP.

generate a short text (tips) as an explanation of the recommendation. We set the dimension of latent factor as 300 and hidden size as 400. The parameters λ_r, λ_c, λ_s and λ_n are tuned within $\{10^{-4}, 10^{-2}, 1\}$.

- **LexRank** [51] is a classical method in the field of text summarization. It uses a graph-based approach to compute the correlation between sentences, and selects core sentences to generate summarization. Similar to [42], we refine LexRank to make it capable of extracting sentences for explanation generation. For a detailed description of the refinements, please refer to [42].

4.3 Evaluation Metrics

The goal of our proposed framework is to find out the top-K items that are most interesting to the target user, while generating natural language explanations of recommendation results.

To evaluate the performance of preference prediction, in our experiments, we adopt two common ranking evaluation methods: Accuracy@K and Normalized Discounted Cumulative Gain (NDCG), to evaluate the quality of the recommendation list. Accuracy@K is widely used as a metric for previous ranking based recommender systems [17, 52] to test whether the target user's favorite items appear in the top-K recommendation list. NDCG is a widely used measure for evaluating recommendation algorithms [53, 54], which incorporates both ranking precisions and the position of ratings.

For the evaluation of explanation generation, we take the review written by the user for the item as the ground truth and use ROUGE [55] and BLEU [56] as our evaluation metrics. They are widely used to test the performance of explainable recommendation algorithms [42, 44]. In our experiments, we use Recall, Precision and F-measure metrics of ROUGE-1, ROUGE-2 and ROUGE-L, as well as BLEU to evaluate the quality of the generated explanation.

4.4 Experimental Settings

To evaluate the performance of top-K recommendation, we adopt the widely used leave-one-out evaluation strategy [57, 58]. For each user, we select their latest interaction data to construct the test set, and the remaining interaction data are used for training. At the testing stage, we follow the common strategy [58, 59] that randomly selects 99 items that have not interacted with the target user as interference items for each ground-truth item, and rank these 100 items (99 interference items and the ground-truth item) to test the top-K recommendation performance of our method.

When training our model, inspired by [50], we randomly select two negative instances for each positive instance to form negative instance set $\boldsymbol{Y}^-$. Specifically, we set the ratings of negative user-item pairs to zero. However, since negative instances have no review information, they are only used to train the preference prediction module and are not involved in the training of the explanation generation module. The Word2Vec[7] model pre-trained on Google News is used to initialize the word embedding matrix, and the words without pre-trained are randomly initialized and updated during the training process.

For neural network, we randomly initialize model parameters with a Gaussian distribution and select Adam [60] as training optimizer. All the best hyper-parameters are found by grid search. We implemente EDCF based on TensorFlow, and the code is published on Github[8].

7. https://code.google.com/archive/p/word2vec/
8. https://github.com/zzmylq/EDCF

TABLE 3
Performance of explanation generation on Kindle Store.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	R	P	F1	R	P	F1	R	P	F1	
LexRank	0.17969	0.18116	0.17904	0.02636	0.02759	0.02637	0.13934	0.14072	0.13877	0.25085
NARRE	0.28953	0.15674	0.19324	0.02769	0.01246	0.01607	0.18544	0.09936	0.12269	0.22911
NRT	0.23923	0.15213	0.17317	0.03572	0.01716	0.02111	0.20891	0.12989	0.14879	0.15575
EDCF_SE	0.30162	0.41416	0.34645	0.05887	0.06047	0.05948	0.25454	0.34982	0.29243	0.31524
EDCF (ours)	**0.32911**	**0.43030**	**0.37061**	**0.07388**	**0.07526**	**0.07440**	**0.28307**	**0.37014**	**0.31873**	**0.33833**

TABLE 4
Performance of explanation generation on Movies&TV.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	R	P	F1	R	P	F1	R	P	F1	
LexRank	0.16431	0.16416	0.16298	0.02213	0.02294	0.02209	0.12820	0.12814	0.12703	0.24137
NARRE	0.22172	0.13928	0.16310	0.01090	0.00588	0.00720	0.14685	0.09072	0.10655	0.22944
NRT	0.21127	0.13716	0.15433	0.03247	0.01503	0.01915	0.18428	0.11646	0.13219	0.14708
EDCF_SE	0.27832	0.39083	0.32242	0.04590	0.04726	0.04642	0.23322	0.32798	0.27025	0.29155
EDCF (ours)	**0.29605**	**0.39821**	**0.33701**	**0.05367**	**0.05484**	**0.05407**	**0.25091**	**0.33780**	**0.28567**	**0.30834**

TABLE 5
Performance of explanation generation on Electronics.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	R	P	F1	R	P	F1	R	P	F1	
LexRank	0.17127	0.17313	0.17083	0.01713	0.01832	0.01718	0.13202	0.13372	0.13162	0.23948
NARRE	0.24037	0.15655	0.18331	0.01949	0.01059	0.01322	0.15395	0.09860	0.11604	0.24274
NRT	0.21521	0.14617	0.16119	0.03423	0.01617	0.01994	0.19198	0.12858	0.14246	0.13812
EDCF_SE	0.28375	0.37252	0.31982	0.04504	0.04573	0.04527	0.23839	0.31327	0.26875	0.28923
EDCF (ours)	**0.29773**	**0.38154**	**0.33232**	**0.05064**	**0.05127**	**0.05084**	**0.25135**	**0.32236**	**0.28060**	**0.30410**

TABLE 6
Performance of explanation generation on Yelp.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	R	P	F1	R	P	F1	R	P	F1	
LexRank	0.14798	0.14987	0.14801	0.01500	0.01585	0.01504	0.11515	0.11687	0.11518	0.22774
NARRE	0.31342	0.10629	0.15315	0.02617	0.00657	0.01006	0.19497	0.06550	0.09449	0.15703
NRT	0.19902	0.13559	0.15169	0.02473	0.01217	0.01515	0.17101	0.11436	0.12867	0.15719
EDCF_SE	0.27616	0.38478	0.31946	0.04682	0.04822	0.04744	0.23567	0.32898	0.27281	0.29208
EDCF (ours)	**0.32308**	**0.41699**	**0.35989**	**0.05742**	**0.05209**	**0.05398**	**0.25274**	**0.32509**	**0.28099**	**0.29846**

4.5 Results and Analysis

4.5.1 Preference Prediciton

In this subsection, we evaluate the preference prediction performance of our proposed EDCF method through a ranking task. Fig.3 demonstrates the ranking performance, including Accuracy@K and NDCG@K of EDCF and competing baselines on four real-world datasets when the hash code length is set as 128.

In our experiments, we select the prevailing hashing-based recommendation method as well as prevailing continuous value based recommendation method as baselines. Specifically, to evaluate the preference prediction performance of EDCF on the single-task case, we design a variant of the EDCF method called EDCF_SR. It removes the explanation generation module from EDCF and keeps the attentive TextCNN and neural discrete collaborative filtering modules. From Fig.3, we can observe that the proposed EDCF framework consistently and significantly outperforms all comparative methods with respect to Accuracy@K and NDCG@K on all datasets.

DCF is a classical hashing-based recommendation algorithm that adopts a bit-by-bit optimization strategy to learn hash codes. However, it only uses interaction information for rating prediction, which limits its recommendation performance. DCMF extends DCF and considers the content information of users and items. However, it fails to fully exploit the non-linear correlation between content features. DFM adopts the factorization machine to model the potential relevance between features, but ignores the collaborative interaction. DDL applies DBN to extract item features, which is not trained in an end-to-end deep network architecture. In other words, DDL extends DCF by adding DBN, which cannot fully exploit the feature learning capability of deep networks.

Moreover, these experimental results show that the proposed EDCF outperforms both the hashing-based deep recommendation method NeuHash-CF and the continuous value based deep recommendation method DeepMF. The

TABLE 7
Examples of the predicted ratings and the generated explanations.

Case	Ground truth		Prediction	
	Rating	Review	Rating	Review
1	5	I started reading Debora's books from the very beginning of the series, and they keep getting better and better. I am disappointed to see this line of the series end but cannot wait for the new one to start.	4.878	I love reading Geary's story the very beginning, see how author and book are better and better. I wait for the next story.
2	5	I love this series by Debora. I tripped into it then realized it was a series. I have read them out of order and still love them. I hope she continues to write them because the characters are really developing.	4.805	I love this series and the Geary. I have read all of the order, and the story are so good.
3	1	This was a good storyline but the grammar and phrasing were terrible. I would have enjoyed this read a lot more if I did not keep tripping over the horrific misspellings and completely wrong words.	0.821	It was a good story. But the spelling errors, I not like the book.
4	1	The story was ok, but the writer was terrible. There were so many grammatical errors and spelling errors. I wouldn't buy another book by this author.	1.289	The story line is good, but the author did many errors. I haven't recommend this book.

better performance of EDCF_SR than NeuHash-CF and DeepMF validates the effectiveness of nerual discrete collaborative filtering module, and demonstrates that the discriminative hash codes can be learned by our proposed deep hashing architecture. Additionally, from the experimental results, we notice that the proposed EDCF outperforms EDCF_SR, which demonstrates the effectiveness of the proposed multi-task learning strategy. A detailed analysis of EDCF_SR will be provided in section 4.5.4.

4.5.2 Explanation Generation

In the online recommendation stage, the top-k recommendation task is executed very fast because the Hamming similarity can be computed very efficiently using an Exclusive-Or operation. Differently, the explanation generation task relies on the LSTM model, and the generation process of natural language explanation requires computation operations in continuous numerical space, so there is a significant efficiency difference between the two tasks. To address this issue, we first generate top-K recommendations by efficient hash code matching, and then the LSTM-based explanation generation module is used to generate the corresponding K explanations. In addition, the base model of the explanation generation module can be replaced according to specific application scenarios.

To evaluate the explanation generation performance of EDCF in the single-task case, we designed a variant of the EDCF method called EDCF_SE. It removes the neural discrete collaborative filtering module from EDCF, and uses the features of target user and item candidates extracted by the attentive TextCNN module as the input to the LSTM to predict the target users' reviews of the item candidates.

The evaluation results of explanation generation of EDCF and the comparative methods are given in Table 3 - Table 6. In order to show more details, we also report Recall, Precision, and F1-measure of ROUGE-1, ROUGE-2 and ROUGE-L. From the experimental results, we can see that our proposed EDCF achieves significant improvement in all metrics compared to competitive methods on all four datasets.

From the results, we notice that explanation generation performance of EDCF is much better than EDCF_SE, which illustrates the effectiveness of the multi-task learning strategy. The better performance of EDCF_SE than LexRank, NARRE and NRT validates the effectiveness of the natural language explanation generation module. A detailed analysis of EDCF_SE will be provided in section 4.5.4. Moreover, the experimental results show that the proposed EDCF outperforms NRT, which is also a generation-based explainable recommendation algorithm. The reason is that the goal of NRT is to generate tips as natural language explanation of the recommendation results. In the original paper, the authors use the summary field in the Amazon dataset as tips for training and use reviews as auxiliary information to participate in model training. However, summaries and reviews are not always available. Since most recommender systems only have review information available, the performance of NRT will be degraded in this case.

4.5.3 Case Analysis

We provide some real cases in Table 7 to analyze the linguistic quality and the sentiment correlation between the predicted ratings and the generated explanations. Benefiting from the information sharing in multi-task learning, EDCF can take into account user interests and item features when generating natural language explanations. For example, from the analyses of case 1 and case 2, we can observe that the generated explanations accurately predict the users' preferences for the author Debora Geary, and predict that the book belongs to a series. Specifically, the generated explanations not only align with the ground truth ratings in terms of overall sentiment but also provide accurate explanations for the recommendation results at a fine-grained aspect level. For example, the generated explanations of case 3 and case 4 are shown that although users rate the book positively in terms of storyline, they end up with a one-star rating due to significant problems with the author's writing, which is consistent with the semantics expressed by the ground truth reviews.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3185093

TABLE 8
Ablation experimental results on Kindle Store and Movies&TV.

Dataset	Methods	Acc@20	NDCG@20	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
				R	P	F	R	P	F	R	P	F	
Kindle Store	EDCF	**0.43112**	**0.19226**	**0.32911**	**0.43030**	**0.37061**	**0.07388**	**0.07526**	**0.07440**	**0.28307**	**0.37014**	**0.31873**	**0.33833**
	EDCF_WA	0.41553	0.17510	0.32653	0.42762	0.36784	0.07245	0.07382	0.07296	0.28040	0.36723	0.31584	0.33602
	EDCF_EMW	0.40517	0.16912	0.32551	0.42758	0.36712	0.07191	0.07323	0.07239	0.27943	0.36707	0.31510	0.33577
	EDCF_FMW	0.42959	0.19075	0.32782	0.42969	0.36949	0.07340	0.07480	0.07393	0.28227	0.37004	0.31812	0.33737
	EDCF_SR	0.42846	0.18945	——	——	——	——	——	——	——	——	——	——
	EDCF_SE	——	——	0.30162	0.41416	0.34645	0.05887	0.06047	0.05948	0.25454	0.34982	0.29243	0.31524
Movies&TV	EDCF	**0.59973**	**0.30147**	**0.29605**	**0.39821**	**0.33701**	**0.05367**	**0.05484**	**0.05407**	**0.25091**	**0.33780**	**0.28567**	**0.30834**
	EDCF_WA	0.55520	0.27253	0.29251	0.39534	0.33363	0.05186	0.05303	0.05226	0.24732	0.33461	0.28214	0.30458
	EDCF_EMW	0.52787	0.25002	0.29097	0.39608	0.33288	0.05145	0.05268	0.05189	0.24619	0.33547	0.28170	0.30313
	EDCF_FMW	0.59019	0.29748	0.29444	0.39757	0.33576	0.05296	0.05420	0.05340	0.24963	0.33738	0.28471	0.30718
	EDCF_SR	0.57835	0.28919	——	——	——	——	——	——	——	——	——	——
	EDCF_SE	——	——	0.27894	0.40026	0.32623	0.04728	0.04879	0.04790	0.23304	0.33486	0.27265	0.29052

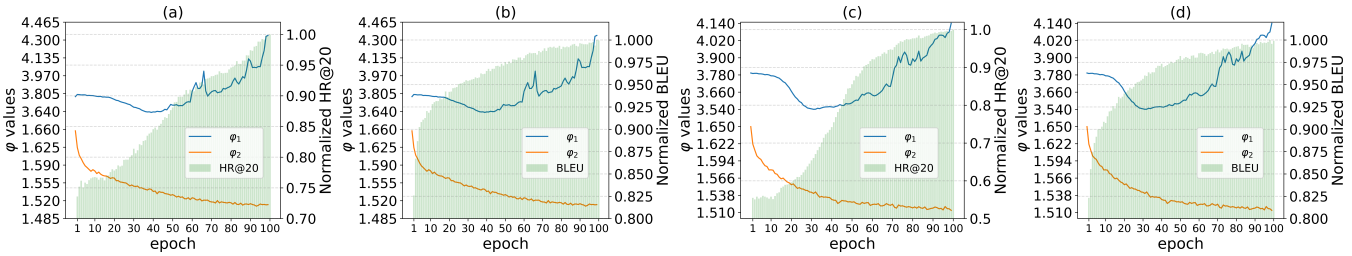


Fig. 4. Effects of the multi-task learning strategy on Kindle Store (a-b) and Electronics (c-d).

For linguistic quality, the syntax of some generated explanations is problematic and some redundant stop words may be generated. In addition, the words used in the generated explanations may be inaccurate, but these words are semantically similar to the ground-truth.

In general, although the language quality still needs to be improved, EDCF can generate fine-grained semantic-informative reviews to illustrate why the system recommends the items from multiple perspectives.

4.5.4 Ablation Analysis

In this paper, we propose a multi-task deep hashing framework for explainable recommendation that extracts user and item features from reviews via an attentive TextCNN, and adaptively adjusts the weights of multiple tasks during training by modeling their uncertainty. In this subsection, we design five variants of our method to evaluate the feature extraction performance of the attentive TextCNN and the effictiveness of adaptive weight learning strategy: 1) EDCF_WA: It replaces the review feature extraction module with a simple TextCNN network. 2) EDCF_EMW: It fixes the weight of each task to 1 during the training process. 3) EDCF_FMW: We perform a full training of EDCF in advance and obtain the weights w_1^{final} and w_2^{final} of the two tasks that converge at the end of the training. EDCF_FMW fixes the weights of the two tasks as w_1^{final} and w_2^{final} respectively during the training process. 4) EDCF_SR and EDCF_SE: Two variants of our method described in sections 4.5.1 and 4.5.2, respectively.

Table 8 shows the comparison of the top-K recommendation and explanation generation performance. It can be easily observed that the performance of our method is obviously higher than that of the variants in both top-K recommendation and explanation generation tasks.

The performance of EDCF is higher than EDCF_WA on both tasks, demonstrating the effectiveness of the attention module on extracting text features. In addition, the experimental results of EDCF_SR and EDCF_SE illustrate that multiple tasks can be mutually reinforced in the adaptive multi-task learning process. Additionally, the performance degradation of EDCF_EMW and EDCF_FMW also validates the effectiveness of the adaptive weight learning strategy. Specifically, the detailed experiments and analyses of the adaptive weight learning strategy are provided in the following subsection.

4.5.5 Effects of Adaptive Multi-task Learning

At the beginning of the training, we initialize both φ_1 and φ_2 to 1. The uncertainty-based adaptive weight learning strategy will tend to learn the easier tasks first. From Fig.4, we can see that the value of φ_1 is consistently larger than φ_2, which means that explanation generation is the easier task during training, while the preference prediction task is more difficult. Therefore, we can see that the BLEU metric, which measures the performance of explanation generation, achieves a high-level preferentially after a few epochs of training. However, the HR metric, which measures the performance of preference prediction, improves slowly during this period. Then, the φ_1 starts to decrease, i.e., the weight of the preference prediction task starts to increase, and the HR metric also improves rapidly. Finally, the performance evaluation metrics of both tasks tend to converge. That is, during the training process, the model gives priority to the easier explanation generation task, and then focuses on the more difficult preference prediction task, until both tasks are better solved.

TABLE 9
Performance w.r.t. different hash code length.

Dataset	HL	Acc@20	NDCG@20	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
				R	P	F	R	P	F	R	P	F	
Kindle Store	32	0.32137	0.11443	0.31101	0.42012	0.35482	0.06457	0.06599	0.06511	0.26462	0.35753	0.30188	0.32105
	64	0.38313	0.15416	0.32499	0.42681	0.36648	0.07168	0.07301	0.07216	0.27895	0.36634	0.31450	0.33478
	128	0.43112	0.19226	0.32839	0.42968	0.36988	0.07375	0.07513	0.07427	0.28260	0.36982	0.31828	0.33845
	256	**0.43667**	**0.19426**	**0.32911**	**0.43030**	**0.37061**	**0.07388**	**0.07526**	**0.07440**	**0.28307**	**0.37014**	**0.31873**	**0.33833**
Movies&TV	32	0.46313	0.20504	0.29058	0.39621	0.33264	0.05128	0.05253	0.05172	0.24542	0.33494	0.28098	0.30305
	64	0.53361	0.25455	0.29099	0.39557	0.33268	0.05165	0.05287	0.05208	0.24629	0.33514	0.28162	0.30341
	128	0.59973	0.30147	0.29605	0.39821	0.33701	0.05367	0.05484	0.05407	0.25091	0.33780	0.28567	0.30834
	256	**0.60544**	**0.30313**	**0.29707**	**0.39878**	**0.33796**	**0.05412**	**0.05527**	**0.05451**	**0.25190**	**0.33848**	**0.28663**	**0.30862**

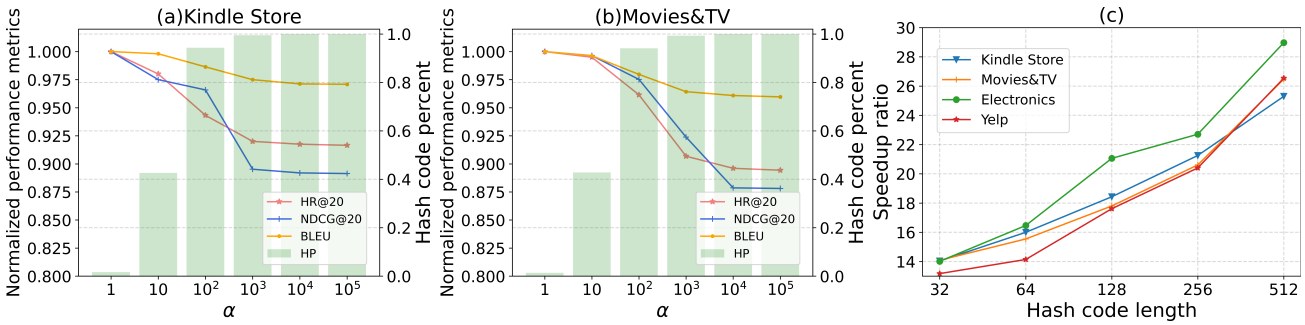


Fig. 5. Effects of ATanh with different α, and the speedup ratio w.r.t hash code length.

4.5.6 Hashing Performance Analysis

We conduct experiments to analyze the performance of hash learning. Table 9 shows the performance of EDCF on preference prediction and explanation generation tasks with different hash code lengths. From Table 9 we can observe that the performance metrics of EDCF are improving as the hash code length increases. However, when the hash code length is larger than 128 bits, the doubling of the hash code length has a limited effect on the performance improvement. Therefore, the storage space and performance requirements of the model need to be balanced in practical applications.

Next, we analyze the improvement of recommendation efficiency by using binary hash codes. Fig.5(c) shows the speedup of the binary hash codes in feature matching for different hash code lengths, which is compared with the continuous-valued feature representation method of equal length. We can see that the improvement of matching speed by the binary hash codes method becomes more and more significant as the length of the hash codes increases.

Additionally, we add ATanh layer to the neural network in order to enable end-to-end deep hash learning. The most important parameter in ATanh is α, which determines whether the learned feature representations are binary or not. Therefore, we design experiments to observe the effect of α on hash learning and model performance. From Fig.5(a) and Fig.5(b) we can see that when $\alpha \geq 10^3$, the learned feature representations are almost binary, and when $\alpha \geq 10^4$, the model can output stable binary feature representations. We also note that the performance degradation is about 2.5% to 12.5% using the binary feature representation ($\alpha \geq 10^4$) compared to the continuous-valued feature representation ($\alpha = 1$). However, considering the significant efficiency gain by using binary feature representation shown in Fig.5(c), this approach has important practical value,

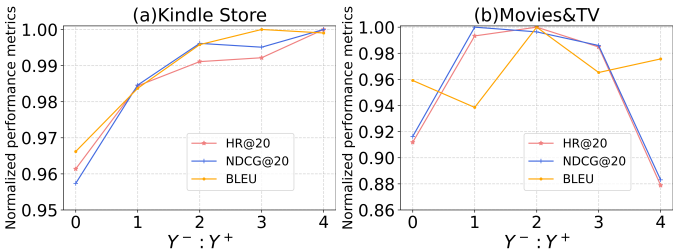


Fig. 6. Effects of selection ratio of negative samples on Kindle Store and Movies&TV.

especially in two-stage recommender systems, which consists of a hashing-based recalling stage and a fine-ranking stage. Finally, we investigate the effect of the number of negative instances in the training set on hash learning. Fig.6 shows the effect of the ratio of negative instances to positive instances on the performance of the model. We can observe that introducing a moderate amount of negative instances is advantageous, but a large number of negative instances damage the performance of the model due to the interference of huge noises. Therefore, in our experiments, we adopt the strategy of assigning two negative instances for each positive instance to introduce a moderate amount of negative instances.

5 Conclusion

In this paper, we propose an explainable discrete collaborative filtering method that projects user-item interaction and review information into hash codes to support efficient top-K recommendation while generating the natural language explanation of the recommendation results. The proposed method can generate binary hash codes directly during end-to-end deep network training, automatically adjust multi-task weights through an uncertainty-based adaptive weight strategy, and generate natural language explanations using

binary hash codes as input. Moreover, our method only uses users' publicly available ratings and reviews on the web as input, thus effectively protecting users' privacy. The evaluation on four widely tested datasets shows that the proposed method outperforms the state-of-the-art baselines on both preference prediction and explanation generation tasks.

REFERENCES

[1] S. Li, X. Li, J. Lu, and J. Zhou, "Self-supervised video hashing via bidirectional transformers," in *CVPR*, 2021, pp. 13 549–13 558.

[2] X. Wang, Z. Zhang, B. Wu, F. Shen, and G. Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in *CVPR*, 2021, pp. 16 357–16 366.

[3] L. Zhu, X. Lu, Z. Cheng, J. Li, and H. Zhang, "Deep collaborative multi-view hashing for large-scale image search," *TIP*, vol. 29, pp. 4643–4655, 2020.

[4] X. Lu, L. Zhu, Z. Cheng, L. Nie, and H. Zhang, "Online multi-modal hashing with dynamic query-adaption," in *SIGIR*, 2019, pp. 715–724.

[5] H. Cui, L. Zhu, J. Li, Y. Yang, and L. Nie, "Scalable deep hashing for large-scale social image retrieval," *TIP*, vol. 29, pp. 1271–1284, 2020.

[6] L. Xie, J. Shen, and L. Zhu, "Online cross-modal hashing for web image retrieval," in *AAAI*, 2016, pp. 294–300.

[7] L. Xie, J. Shen, J. Han, L. Zhu, and L. Shao, "Dynamic multi-view hashing for online image retrieval," in *IJCAI*, 2017, pp. 3133–3139.

[8] H. T. Shen, L. Liu, Y. Yang, X. Xu, Z. Huang, F. Shen, and R. Hong, "Exploiting subspace relation in semantic labels for cross-modal hashing," *TKDE*, vol. 33, no. 10, pp. 3351–3365, 2021.

[9] Y. Wang, X. Luo, L. Nie, J. Song, W. Zhang, and X. Xu, "BATCH: A scalable asymmetric discrete cross-modal hashing," *TKDE*, vol. 33, no. 11, pp. 3507–3519, 2021.

[10] D. Lian, X. Xie, and E. Chen, "Discrete matrix factorization and extension for fast item recommendation," *TKDE*, vol. 33, no. 5, pp. 1919–1933, 2021.

[11] H. Liu, X. He, F. Feng, L. Nie, R. Liu, and H. Zhang, "Discrete factorization machines for fast feature-based recommendation," in *IJCAI*, 2018, pp. 3449–3455.

[12] H. Wang, D. Lian, and Y. Ge, "Binarized collaborative filtering with distilling graph convolutional network," in *IJCAI*, 2019, pp. 4802–4808.

[13] H. Wang, N. Shao, and D. Lian, "Adversarial binary collaborative filtering for implicit feedback," in *AAAI*, 2019, pp. 5248–5255.

[14] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T. Chua, "Discrete collaborative filtering," in *SIGIR*, 2016, pp. 325–334.

[15] Y. Xu, L. Zhu, Z. Cheng, J. Li, Z. Zhang, and H. Zhang, "Multi-modal discrete collaborative filtering for efficient cold-start recommendation," *TKDE*, pp. 1–14, DOI: 10.1109/TKDE.2021.3 079 581, 2021.

[16] Y. Zhang, D. Lian, and G. Yang, "Discrete personalized ranking for fast collaborative filtering from implicit feedback," in *AAAI*, 2017, pp. 1669–1675.

[17] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian, "Discrete deep learning for fast content-aware recommendation," in *WSDM*, 2018, pp. 717–726.

[18] D. Lian, R. Liu, Y. Ge, K. Zheng, X. Xie, and L. Cao, "Discrete content-aware matrix factorization," in *KDD*, 2017, pp. 325–334.

[19] X. Chen, Y. Zhang, and Z. Qin, "Dynamic explainable recommendation based on neural attentive models," in *AAAI*, 2019, pp. 53–60.

[20] C. Li, C. Quan, L. Peng, Y. Qi, Y. Deng, and L. Wu, "A capsule network for recommendation and explaining what you like and dislike," in *SIGIR*, 2019, pp. 275–284.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[23] J. Wang, W. Liu, S. Kumar, and S. Chang, "Learning to hash for indexing big data - A survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[24] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *FTIR*, vol. 14, no. 1, pp. 1–101, 2020.

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

[26] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, 1999, pp. 518–529.

[27] A. Karatzoglou, A. J. Smola, and M. Weimer, "Collaborative filtering on a budget," in *AISTATS*, 2010, pp. 389–396.

[28] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," in *KDD*, 2012, pp. 498–506.

[29] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *TPAMI*, vol. 35, no. 12, pp. 2916–2929, 2013.

[30] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashing," in *CVPR*, 2014, pp. 2147–2154.

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

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

[33] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

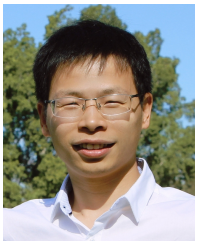
[34] G. Guo, E. Yang, L. Shen, X. Yang, and X. He, "Discrete trust-aware matrix factorization for fast recommendation," in *IJCAI*, 2019, pp. 1380–1386.

[35] C. Liu, X. Wang, T. Lu, W. Zhu, J. Sun, and S. C. H. Hoi, "Discrete social recommendation," in *AAAI*, 2019, pp. 208–215.

[36] Q. Tan, N. Liu, X. Zhao, H. Yang, J. Zhou, and X. Hu, "Learning to hash with graph neural networks for recommender systems," in *WWW*, 2020, pp. 1988–1998.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017, pp. 1–14.

[38] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.
[39] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations," in *IUI*, 2018, pp. 1–2.
[40] S. Chang, F. M. Harper, and L. G. Terveen, "Crowd-based personalized natural language explanations for recommendations," in *RecSys*, 2016, pp. 175–182.
[41] C. Hanxiong, C. Xu, S. Shaoyun, and Z. Yongfeng, "Generate natural language explanations for recommendation," in *SIGIRW*, 2019.
[42] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," in *SIGIR*, 2017, pp. 345–354.
[43] Y. Lu, R. Dong, and B. Smyth, "Why I like it: Multi-task learning for recommendation and explanation," in *RecSys*, 2018, pp. 4–12.
[44] Z. Chen, X. Wang, X. Xie, T. Wu, G. Bu, Y. Wang, and E. Chen, "Co-attentive multi-task learning for explainable recommendation," in *IJCAI*, 2019, pp. 2137–2143.
[45] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.
[46] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, pp. 2493–2537, 2011.
[47] W. Fu, Z. Peng, S. Wang, Y. Xu, and J. Li, "Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems," in *AAAI*, 2019, pp. 94–101.
[48] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *WWW*, 2018, pp. 1583–1592.
[49] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma, "Content-aware neural hashing for cold-start recommendation," in *SIGIR*, 2020, pp. 971–980.
[50] H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *IJCAI*, 2017, pp. 3203–3209.
[51] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *JAIR*, vol. 22, pp. 457–479, 2004.
[52] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen, "Streaming ranking based recommender systems," in *SIGIR*, 2018, pp. 525–534.
[53] Z. Cheng and J. Shen, "On effective location-aware music recommendation," *TOIS*, vol. 34, no. 2, pp. 1–32, 2016.
[54] C. Li, X. Niu, X. Luo, Z. Chen, and C. Quan, "A review-driven neural model for sequential recommendation," in *IJCAI*, 2019, pp. 2866–2872.
[55] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *ACL*, vol. 8, pp. 74–81, 2004.
[56] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *ACL*, 2002, pp. 311–318.
[57] X. He, H. Zhang, M. Kan, and T. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *SIGIR*, 2016, pp. 549–558.
[58] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
[59] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *WWW*, 2015, pp. 278–288.
[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015, pp. 1–15.



Lei Zhu is currently a professor with the School of Information Science and Engineering, Shandong Normal University. He received his B.Eng. and Ph.D. degrees from Wuhan University of Technology in 2009 and Huazhong University Science and Technology in 2015, respectively. He was a Research Fellow at the University of Queensland (2016-2017). His research interests are in the area of large-scale multimedia content analysis and retrieval. Zhu has co-/authored more than 100 peer-reviewed papers, such as SIGIR, ACM MM, IEEE TPAMI, IEEE TIP, IEEE TKDE, and ACM TOIS. His publications have attracted more than 3500 Google citations. At present, he serves as the editorial board member of 4 SCI-indexed journals, youth editorial board of IEEE/CAA Journal of Automatica Sinica. He has served as the Area Chair, Senior Program Committee or reviewer for more than 40 well-known international journals and conferences. He won ACM SIGIR 2019 Best Paper Honorable Mention Award, ADMA 2020 Best Paper Award, ACM China SIGMM Rising Star Award, Shandong Provincial Entrepreneurship Award for Returned Students, and Shandong Provincial AI Outstanding Youth Award.

Yang Xu received the Ph.D. degree at Shandong University, China, in 2018. He is currently a lecture with the School of Information Science and Engineering, Shandong Normal University, China. His research interests are recommender systems and multimedia computing.

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.

Weili Guan is currently a final year Ph.D. student with the Faculty of Information Technology, Monash University Clayton Campus, Australia. Her research interests are multimedia computing and information retrieval. She received the bachelor degree from Huaqiao University. She then obtained her master degree from National University of Singapore. After that, she joined Hewlett Packard enterprise Singapore as a software engineer and worked there for around five years. She has published dozens of papers at the top conferences and journals, like SIGIR, IEEE TIP, and IEEE PAMI.

Zhiyong Cheng is currently a Professor with Shandong Artificial Intelligence Institute, Qilu University of Technology (Shandong Academy of Sciences). He received the Ph.D degree in computer science from Singapore Management University in 2016, and then worked as a Research Fellow in National University of Singapore. His research interests mainly focus on large-scale multimedia content analysis and retrieval. His work has been published in a set of top forums, including ACM SIGIR, MM, WWW, TOIS, IJCAI, TKDE, and TCYB. He has served as the PC member for several top conferences such as MM, MMM etc., and the regular reviewer for journals including TKDE, TIP, TMM etc.