Towards Explainable Collaborative Filtering with Taste Clusters Learning

Yuntao Du
Zhejiang University
Hangzhou, China
ytdu@zju.edu.cn

Jianxun Lian
Microsoft Research Asia
Beijing, China
jianxun.lian@outlook.com

Jing Yao
Microsoft Research Asia
Beijing, China
jingyao@microsoft.com

Xiting Wang
Microsoft Research Asia
Beijing, China
xitwan@microsoft.com

Mingqi Wu
Microsoft Gaming
Redmond, United States
mingqi.wu@microsoft.com

Lu Chen
Zhejiang University
Hangzhou, China
chenlu@zju.edu.cn

Yunjun Gao
Zhejiang University
Hangzhou, China
gaoyj@zju.edu.cn

Xing Xie
Microsoft Research Asia
Beijing, China
xingx@microsoft.com

ABSTRACT

Collaborative filtering (CF) is an effective and widely used technique for recommender systems. In the past decades, we have witnessed substantial progresses in the evolution of latent embedding-based CF for better accuracy performance (such as matrix factorization, neural collaborative filtering, and LightGCN), while the explainability of these model remains inadequately explored. Enabling explainability for recommendation models not only help us understand the decision-making logic of models for trustworthiness, but also benefit multiple applications, such as providing persuasive explanations for item recommendation, generating explicit profiles for users/items, or facilitating item producers in design improvement.

In this paper, we propose a neat, yet effective Explainable Collaborative Filtering (ECF) model based on learning interpretable clusters, with the goal of achieving the three merits: (1) accurate, the model should not sacrifice accuracy for achieving explainability; (2) coherent, the model's predictive logic should match the explanatory logic; (3) self-explainable, the model's explanation should truly reflect its decision making process. The core of ECF is mining taste clusters from user-item interactions and item profiles. We map each user and item to a sparse set of taste clusters, and taste clusters are distinguished by a few representative tags. The user-item preference, users/items' cluster affiliations, and the generation of taste clusters are jointly optimized under the same end-to-end model. We further propose a simple yet effective forest mechanism for ECF model, to guarantee the accuracy, explainability, and diversity of the recommender model simultaneously. Finally, we design several quantitative metrics, including in-cluster item coverage, tag utilization, silhouette, and informativeness, to evaluate the explainability quality of taste clusters in a comprehensive manner. We conduct extensive experiments on three real-world datasets to demonstrate the effectiveness of the model.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Clustering**; **Recommender systems**; *Web mining*; *Summarization*; • **Computing methodologies** → *Learning from implicit feedback.*

KEYWORDS

Collaborative Filtering; Explanability; Recommender Systems; Clustering

ACM Reference Format:
Yuntao Du, Jianxun Lian, Jing Yao, Xiting Wang, Mingqi Wu, Lu Chen, Yunjun Gao, and Xing Xie. 2018. Towards Explainable Collaborative Filtering with Taste Clusters Learning . In *Proceedings of the ACM Web Conference 2023 (WWW '23).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Collaborative filtering (CF) is an effective and widely used technique for recommender systems. The rationale behind CF is that a user's potential interests can be inferred from a group of like-minded users' behaviors (*e.g.,* user-based CF), or, a set of similar items to the user's behavior history (*e.g.,* item-based CF). Existing methods for CF can be roughly categorized into two types: memory-based CF and model-based CF. Memory-based CF methods [27] explicitly find the neighborhood of users/items for preference inference. These methods, although concise and effective, usually suffer from memory consumption, (inference) computational cost, and data sparsity issues. In contrast, model-based CF uses machine learning methods to help model the relationship between users and items. In the past decades, the research trends have been mainly focusing on model-based CF, with methods evolving from latent factorization-based methods (such as MF [17] and SVD++ [16]) to neural embedding-based methods (such as NCF [11] and CDAE [36]), and to recently neural graph-based methods (such as LightGCN [10] and Ultra-GCN [21]). The foundation model in this direction is to represent users and items with high-quality latent factor embeddings, so that users' preference towards items can be decoded from the latent embeddings.

A notable drawback of latent factor-based CF is the lack of transparency and explainability. Knowing the decision logic of recommender algorithms rather than using them as black boxes is important on multiple aspects, such as assisting developers in model debugging and abnormal case studies or generating persuasive explanations to promote the recommended items and increase conversion rates. Many prior attempts have been made to achieve this goal. To name a few representative works, [22] combines a latent matrix factorization model with a latent topic model, then each dimension of user/item embeddings can be associated with a textual topic mined from users' reviews. [40] proposes the Explicit Factor Model (EFM), which extracts aspect features from textual reviews and makes predictions by combining scores from a latent factor model and from aspect-level interest matching. [23] proposes a feature mapping paradigm. By mapping latent embeddings to interpretable aspect features, an uninterpretable model can now provide explainability. However, we argue that explainable CF is still an open question, because existing solutions fail to satisfy at least one of the properties: (P1,flexibility) the dimension of latent embeddings and the number of interpretable features/topics do not necessarily match (*e.g.,* [22] fails on this); (P2,coherence) a model's interpretable modules and predictive modules should be aligned during predictive decision making rather than being decoupled as independent modules (*e.g.,* [40] fails on this); (P3, self-explainable) a model can provide interpretable clues that truly reveal the model's running logic, instead of learning a post-hoc model for explanations (*e.g.,* [23] fails on this).

In this paper, we propose a neat yet effective Explainable Collaborative Filtering (ECF) framework with the goal of accurate and explainable recommendations, while satisfying all three properties. At the core of ECF is mining various taste clusters. We assume that items have some explicit features such as tags or aspects (hereinafter we will use tags for unification). A taste cluster is a group of items which are not only similar in users' latent interest space, but also explicitly share some common tags, so that the overlapping tags can be selected as descriptive interpretation for the taste cluster. Both users and items are mapped to multiple taste clusters. On the one hand, item recommendations can be made by measuring the coherence between users' and items' taste cluster affiliations; on the other hand, the descriptive tags interpret what determines users' preference towards items. We also design a new set of quantitative metrics to evaluate the quality of ECF beyond the accuracy, covering: (1) In-cluster item coverage; (2) Tag utilization; (3) Cluster cohesion and separation; (4) Tag informativeness, with the goal of discovering the discriminability of taste clusters as well as the exactitude, diversity, and uniqueness of clusters interpretability in a holistic manner (please refer to Section 2.4 for details). The ECF framework has many more application potentials beyond the user-item prediction and interpretation. For example, the affiliation of users to taste clusters can be used for user profiling, empowering web-scale services like ads audience targeting and audience extension. The affiliation of items to taste clusters can help missing tags discover for items, automatic item topic generation and theme-wise item recommendations (such as the playlist recommendation in online music service).

However, the implementation and optimization of ECF is a non-trivial task. Main challenges include: (1) how to construct an end-to-end model, so that taste cluster generation, tags selection, user- and item-cluster affiliations can be optimized jointly; (2) how to achieve good performance on both accuracy and explainability simultaneously, instead of sacrificing one for the other; (3) how to guarantee the diversity of generated taste clusters. To this end, we design an embedding-based model to generate taste clusters with discriminative tags and establish the sparse mapping between users/items and clusters. We further devise the forest mechanism for ECF, which can not only substantially lift the accuracy, but also provides a diverse set of taste clusters. For each targeting merit (such as exactitude, diversity, and uniqueness of clusters interpretability) we design a corresponding objective function, so that the whole model can be trained in an end-to-end manner.

In summary, our main contributions include:

- We present a neat yet effective explainable collaborative filtering framework, called ECF, which leverages interpretable taste clusters and sparse user- and item-cluster affiliations for recommendation in a flexible, coherent, and self-explainable way.
- We propose an optimization method for ECF to learn high quality taste clusters with informative tags and sparse affiliations simultaneously in an end-to-end manner.
- We design a new set of quantitative metrics to evaluate the explainability quality of taste clusters comprehensively.
- We conduct extensive experiments on three real-world datasets with various state-of-the-art methods to demonstrate the effectiveness of ECF. The code will be open-sourced, and a preview version is at https://anonymous.4open.science/r/ECF-WWW23.

2 METHODOLOGIES

In this section, we first introduce the task definition of explainable recommendations, then introduce the proposed ECF framework to solve the task. Next, we detail the end-to-end optimization techniques to train ECF. We further design a comprehensive set of metrics to quantitatively evaluate the quality of explainability. At last, we discuss the model complexity of ECF.

2.1 Problem Formulation

User-item Interactions and Item Tags. Here we focus on the implicit feedback [26] in recommendation, where the signal that a user provides about his/her preference is implicit (*e.g.,* review and click). Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ a set of items. Let $\mathbf{Y} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ as the user-item interaction matrix, where $y_{ui} = 1$ means that there is an interaction between user u and item i, otherwise $y_{ui} = 0$. Besides, we also assume there is a set of tags $\mathcal{T}$ for items, and each item i has a subset of tags $\mathcal{T}_i$ to describe its genre and features.

Task Description. Given the interactions data $\mathbf{Y}$ and items' tags $\mathcal{T}$, the task of explainable recommendation is to (1) predict how likely user u would adopt item i, and (2) explain what determines the recommendation of item i to user u with the descriptions from tags $\mathcal{T}$.

2.2 The Proposed ECF framework

We propose an ECF framework for explainable recommendations with collaborative filtering. The core of ECF is mining various taste clusters which represent specific interest patterns. We first give the definition of taste clusters and show how to leverage them for recommendation and explanation. Then we detail the proposed method to generate sparse affiliations between users/items and taste clusters.

The **Taste Cluster** is a collection of items $c = \{i_1, i_2, ..., i_c\}$, which can not only reveal the latent interest of users, but also be explicitly identified with common tags shared by items within. Besides, each taste cluster is associated with a set of tags $\mathcal{T}_c = \{t_1, t_2, ..., t_k\}$, which are derived from corresponding item sets and acts as the descriptive interpretation for the cluster.

Assuming there is a set of taste clusters $C = \{c\}$, we map users/items to these taste clusters with *sparse affiliation matrix*. To be specific, let $\mathbf{A} \in \mathbb{R}^{|\mathcal{U}|\times|C|}$ and $\mathbf{X} \in \mathbb{R}^{|\mathcal{I}|\times|C|}$ be the affiliation matrices between users/items and taste clusters, respectively. Each entry a_{uc} in $\mathbf{A}$ denotes the preference degree of user u to taste cluster c, and each entry x_{ic} denotes the relatedness between item i and taste cluster c. To improve the readability and persuasiveness of the explanation, the affiliation matrix is supposed to be *sparse* (*i.e.,* only the most relevant entries are non-zeros), so that the user preference and item belongings can be clearly identified with only a few taste clusters.

After that, both users and items are mapped to multiple taste clusters, and ECF can make user-item recommendation and explanation with taste clusters and affiliations as follows.

Item recommendation. We assume that a user's decision about whether to make a purchase/click is based on his/her preference with taste clusters, as well as the item's relatedness with them. Thus, the prediction score of user u and item i can be calculated by multiplying their affiliations:

$$\hat{y}_{ui} = \text{sparse\_dot}(\mathbf{a}_u, \mathbf{x}_i), \tag{1}$$

where sparse_dot$(\cdot)$ denotes sparse dot product for sparse vectors $\mathbf{a}_u$ and $\mathbf{x}_i$.

Personalized explanation. For each prediction $\hat{y}_{ui}$, ECF is able to generate explanation by measuring the coherence between users' and items' taste cluster affiliations. Specifically, the overlapped taste clusters are first derived from affiliation matrix:

$$C_{ui} = S(\mathbf{a}_u) \cap S(\mathbf{x}_i), \tag{2}$$

where $S(\cdot)$ is an index selector to collect corresponding clusters where the affiliation between users/items and taste clusters exists, and C_{ui} denotes the set of overlapped taste clusters for user u and item i. Thus, the descriptive tags of taste clusters in C_{ui} are used to interpret what determines user u preference toward item i. Moreover, importance score w_{ui}^c is introduced to quantify the contribution of each taste cluster in C_{ui}:

$$w_{ui}^c = a_{uc} \times x_{ic}. \tag{3}$$

Therefore, ECF is able to explain the prediction decision by using the descriptive tags of overlapped taste clusters C_{ui} and their corresponding importance scores w_{ui}^c. Due to the sparsity of affiliation matrix, the size of overlapped taste clusters is small, so that we could easily derive the readable decision logic of ECF by investigating only a few taste clusters. And the final explanation results can be constructed by existing methods (*e.g.,* template-based explanation [40]) or explanation paths from user to taste clusters and items (demonstrated in Section 3.5.2).

Despite the simplicity, it is still unclear how to obtain the desired taste clusters and sparse affiliation matrix. This is a non-trivial task since performing clustering algorithm (*e.g.,* K-means) for items or blindly collecting items with same tags would fail to encode both user interest and item profiles simultaneously (see Section 3.3 for comparison). Besides, directly learning the affiliation matrix from data is unable to train due to its sparsity nature. Instead, we represent items and taste clusters as embeddings, and measure their affiliations with cosine similarity:

$$\tilde{x}_{ic} = \cos(\mathbf{v}_i, \mathbf{h}_c), \tag{4}$$

where $\mathbf{v} \in \mathbb{R}^d$ and $\mathbf{h} \in \mathbb{R}^d$ are the embeddings of items and clusters, and d is the embedding size. To clearly identify the belongings between items and clusters, we only consider the Top-m clusters for each item:

$$m_{ic} = \begin{cases} 1 & \text{if } c \in \text{argTopm}(\tilde{\mathbf{x}}_i) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbf{x}_i = \sigma(\tilde{\mathbf{x}}_i) \odot \mathbf{m}_i, \tag{6}$$

where $\odot$ denotes Hadamard product, $\sigma(\cdot)$ is the sigmoid function. In the above equations, we first make Top-m selection for each item according to the similarity scores, and apply the sigmoid function to indicate the relatedness of items and clusters with none-zero probabilities, then use a binary mask to filter other less relevant clusters and obtain the final item-cluster affiliation matrix $\mathbf{X}$. However, due to the non-differentiability and discrete nature of argTop operation, the computation is intractable thus cannot be optimized with gradients. Therefore, we relax it by tempered softmax:

$$m_{ic} \approx \tilde{m}_{ic} = \frac{\exp(\cos(\mathbf{v}_i, \mathbf{h}_c)/T)}{\sum_c \exp(\cos(\mathbf{v}_i, \mathbf{h}_c)/T)}, \tag{7}$$

where T is the is a hyperparameter called *temperature*, which control the entropy of the distribution. Thus, for back-propagation, the continuous relaxation $\tilde{m}_{ic}$ is used to approximate m_{ic}. However, when calculating the probability of taste clusters, we should directly use m_{ic} instead of $\tilde{m}_{ic}$, to be consistent with the real affiliation relationship. To close the gap between forward pass and backward pass, we follow a similar idea to reparameterized trick [14, 38], and rewrite m_{ic} as follows:

$$\hat{m}_{ic} = \tilde{m}_{ic} + \text{detach\_gradient}(m_{ic} - \tilde{m}_{ic}), \tag{8}$$

where the detach_gradient will prevent the gradient from back-propagating through it. In the forward pass, detach_gradient has no effect, thus the affiliated clusters can be directly computed by argTop operation. In the backward pass, detach_gradient takes effect, so $\nabla_{\hat{m}_{ic}}\mathcal{L} = \nabla_{\tilde{m}_{ic}}\mathcal{L}$. So the whole computation is fully differentiable and can be smoothly optimized with the model.

We use the same mechanism to obtain the affiliations between users and clusters. Specifically, the user-cluster similarity matrix can be computed as:

$$\tilde{\mathbf{A}} = \mathbf{Y} \times \tilde{\mathbf{X}}, \tag{9}$$

where $\mathbf{Y}$ is the user-item interaction matrix. Then, we also force each user to connect with Top-n taste clusters to derive the user-cluster affiliation matrix $\mathbf{A}$.

Forest Mechanism. Due to the hidden nature of users' interest space, it is hard to determine how many taste clusters are needed to model user preference properly. Thus, to improve the diversity and accuracy of recommendation, we propose a simple yet effective ensemble method, called forest mechanism, to substantially enhance the expressiveness of taste clusters, while also maintaining good explainability. Specifically, for each single ECF model, we randomly select $|C|$ items as the initial taste clusters and use different random seeds for model training. Then we train F different instances to form the final ECF model, and the final prediction is based on the summation of all M models. We find this simple random mechanism would boost the performance of ECF and provide a comprehensive explanation for predictions (the impact of different number of models is discussed in Section 3.4.3).

2.3 Optimization of ECF

Next, we aim to constrain the taste cluster from different perspectives to meet its definition and optimize the model in an end-to-end manner. First, since the taste clusters can reveal users' interest and items' similarity, it is reasonable to directly predict user's preference towards item through their common taste clusters (as described in Equation (1)). Therefore, the cluster-based predictions can be optimized with BPR loss [26]:

$$\mathcal{L}_{\text{CS}} = \sum\nolimits_{(u,i,j)\in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}), \tag{10}$$

where $O = \{(u,i,j)|y_{ui} = 1, y_{uj} = 0\}$ denotes the training set of user-item interactions. Then we consider the informativeness of tags. We first calculate the tag distribution of taste clusters:

$$\tilde{\mathbf{D}} = \mathbf{X}^{\top}\mathbf{E}, \tag{11}$$

where $\mathbf{E}$ is the multi-hot matrix where each entry e_{it} denotes whether item i has the tag t, and $\tilde{\mathbf{D}}$ represents the tag frequency of each taste cluster. Intuitively, we can directly select the most frequent tags as the description of taste clusters. However, this naive approach would result in the indistinguishability between different clusters, since some tags are commonly shared by most items, and they would dominate the tag distribution across different taste clusters. Take the game genres at Xbox dataset as an example. Most of the games are online and allow for mutiplayer, thus selecting the "Multiplayer-Online" tag fails to provide informative description for taste clusters. To tackle this problem, we first reweight each tag according to its frequency of occurrence in items, and then compute the weighted distribution:

$$d_{ct} = \tilde{d}_{ct} \times \log(\frac{N}{f_t + \epsilon}), \tag{12}$$

where N is the number of items, f_t is the frequency of tag t across all items, and ϵ is a small fraction (we set it as 10^{-6}) to avoid numerical overflow. Thus, the above equation considers both tag frequency and items' affiliations for tag informativeness: the more frequently the distinctive tag appears, the more informative it is for interpreting the taste cluster.

Besides, to facilitate the understanding of taste clusters, we argue that the number of tags generated for taste clusters should not be too small or too large. This is because that disproportionate number of tags would be too abstract or too complex to identify users' true interests, thus makes it difficult to interpret. To achieve that, we first normalize their frequencies as:

$$\beta_{ct} = \frac{\exp(d_{ct}/\tau)}{\sum_{c_j\in\mathcal{T}} \exp(d_{ct}/\tau)}, \tag{13}$$

where τ is the is the temperature hyperparameter. For low temperatures, the distribution becomes sharper, thus only the tag with highest score would stand out. As the temperature increases, the distribution becomes more even and the distinctiveness between tags will decrease. Then, we consider maximizing the likelihood of the probabilities of Top-P tags so that the taste clusters can be easily interpreted by those tags:

$$\mathcal{L}_{\text{TS}} = \sum\nolimits_{c\in C} \sum\nolimits_{t\in \text{argTopP}(\beta_c)} -\log \beta_{ct}, \tag{14}$$

where C denotes the set of taste clusters. We fix the number of tags $P = 4$ for all experiments, since it achieves a good balance between informativeness and readability.

Moreover, different taste clusters should contain different items and reveal different user preferences, so that the latent interest space could be better preserved. Hence, for better model capacity and diversity, we encourage the embeddings of taste clusters to differ from each other's. There are many methods available for independence modelling, such as distance correlation [28], orthogonality [19], and mutual information [2, 34]. Here we opt for mutual information for all experiments due to its simplicity and effectiveness:

$$\mathcal{L}_{\text{IND}} = \sum\nolimits_{c\in C} -\log \frac{\exp(s(\mathbf{h}_c, \mathbf{h}_c))}{\sum_{c'\in C} \exp(s(\mathbf{h}_c, \mathbf{h}_{c'}))}, \tag{15}$$

where $s(\cdot)$ is the similarity function to measure the associations of any two taste clusters, which is set as cosine function here. Finally, we consider learning taste clusters by putting the above objective functions together:

$$\mathcal{L}_{\text{TC}} = \mathcal{L}_{\text{CS}} + \mathcal{L}_{\text{TS}} + \mathcal{L}_{\text{IND}}. \tag{16}$$

Note that we do not need to tune any weighting coefficients for each loss, since each of them defines an essential aspect that the taste clusters should accomplish. However, due to the argTop operations in affiliation selections, the supervised signals are sparse and hard to converge. Thus, we add auxiliary supervised signals from user-item predictions:

$$\mathcal{L}_{\text{CF}} = \sum\nolimits_{(u,i,j)\in O} -\ln \sigma(\mathbf{e}_u^{\top}\mathbf{v}_i - \mathbf{e}_u^{\top}\mathbf{v}_j), \tag{17}$$

where $\mathbf{e}_u$ denotes the embeddings of user u. We choose inner product of embeddings to measure the similarity between users and items for simplicity, but more sophisticated embedding-based models (*e.g.,* LightGCN [10]) can also be applied, which will be discussed in Section 3.6. By combing the taste clusters loss and collaborative loss, we minimize the following objective function to learn the model parameters:

$$\mathcal{L}_{\text{ECF}} = \mathcal{L}_{\text{TC}} + \lambda\mathcal{L}_{\text{CF}}, \tag{18}$$

where λ a hyperparameter to control the impact of auxiliary collaborative signals.

2.4 New Metrics for Explainability

Here we aim to design a holistic set of quantitative metrics to qualify the effectiveness of explanation *w.r.t.* taste clusters:

- **In-cluster item coverage** denotes the proportion of items in the taste cluster that the selected tags can cover:
$$\text{Cov.} = \frac{1}{Z}\sum\nolimits_{c\in C}\sum\nolimits_{i\in c}\frac{\mathbb{1}(\mathcal{T}_i \cap \mathcal{T}_c)}{|c|}, \tag{19}$$
where $\mathbb{1}(\cdot) = 1$ when the item and taste cluster share at least one tag, otherwise zero. When the item coverage ratio is high, we deem that these tags can be properly used as the descriptive interpretation for the taste cluster.
- **Tag utilization** represents how many unique tags are used for interpreting taste clusters:
$$\text{Util.} = \frac{1}{|\mathcal{T}|}\bigcup_{c\in C}\mathcal{T}_c, \tag{20}$$
where we union all the selected tags from each taste cluster, and a higher tag utilization indicates a more diverse interpretation of interest patterns.
- **Silhouette** is a clustering metric which measures the similarity difference between intra-cluster items and inter-cluster items:
$$\text{Sil.} = \frac{1}{|\mathcal{I}|}\sum\nolimits_{i\in\mathcal{I}}\frac{b(i)-a(i)}{\max\{a(i), b(i)\}}, \tag{21}$$
where $a(i)$ is the mean distance between item i and other items in the same taste cluster, and $b(i)$ is the smallest distance of item i to all items in other taste clusters. The high silhouette indicates the items are similar in the taste clusters.
- **Informativeness** measures the distinctiveness of selected tags to represent the items in the taste cluster:
$$\text{Info.} = \frac{1}{|C|}\sum\nolimits_{c_i\in C}\frac{|R(\mathcal{T}_c)\cap c|}{|c|}, \tag{22}$$
where $R(\mathcal{T}_c)$ is an post-hoc discriminator that predicts the most likely top $|c|$ items given the tags of taste cluster c . We detail the implementation of the function $R(\cdot)$ in Appendix A.7. The higher informativeness means that those tags are more accurate to describe the items in the taste clusters.

The above four metrics measure the quality of taste clusters from different aspects. Since ECF depends on taste clusters to make recommendation and explanation, these metrics can be also viewed as the evaluation of explanation. We also provide an overall metric by normalizing each metric with a random approach, which will be detailed in Section 3.3.

2.5 Complexity Analysis

2.5.1 ***Model Size.*** We analyze the size of ECF from both training and inference perspectives. During the training, the model parameters of single ECF consist of (1) ID embedding of users and items $\{\mathbf{U},\mathbf{V}|\mathbf{U}\in\mathbb{R}^{|\mathcal{U}|\times d},\mathbf{V}\in\mathbb{R}^{|\mathcal{I}|\times d}\}$, which are also used by all embedding-based methods; and (2) ID embedding of taste clusters $\{\mathbf{H}|\mathbf{H}\in\mathbb{R}^{|C|\times d}\}$. We utilize F different ECF models for training, so that the overall parameters are F times of the single model parameters. During the inference, different from other embedding-based methods, ECF only needs the sparse affiliations between users/items and clusters for prediction, and the parameters are $F(m|\mathcal{I}|+n|\mathcal{U}|)$, where m and n are the number of affiliations. The size of ECF is similar or even less than typical embedding-based methods (*i.e.,* MF need $d(|\mathcal{I}|+|\mathcal{U}|)$ parameters) when selected affiliations $m, n \ll d$.

2.5.2 ***Time Complexity.*** Time cost of ECF mainly comes from the taste clusters learning. For collaborative similarity learning of taste clusters, the computational complexity of calculating the affiliations is $O(|C|(d|\mathcal{I}|+|\mathbf{Y}|)$, where $|C|$, $|\mathcal{I}|$, $|\mathbf{Y}|$ and d denote the number of taste clusters, items, interactions, and the embedding size. For tag similarity learning, the computational complexity of tag distribution is $O(|C||\mathcal{I}||\bar{t}|)$, where $|\bar{t}|$ is the average number of tags that items have. As for independence modeling, the cost of mutual information is $O(d|C|^2)$. Besides, computing the auxiliary collaborative signals would cost $O(d|\mathcal{I}||\mathcal{U}|)$. Thus, the time complexity of the whole training epoch is $O(|C||\mathbf{Y}|+d|\mathcal{I}||\mathcal{U}|)$ when the number of taste clusters is far more less than the number of users/items. Under the same experimental settings (*i.e.,* same embeddings size and same number of tags), ECF has comparable complexity to EFM and AMCF, two representative explainable methods.

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

Dataset	#Users	#Items	#Interactions	#Tags
Xbox	465,258	330	6,240,251	115
MovieLens	6,033	3,378	836,434	18
Last-FM	53,486	2,062	2,228,949	54

3 EXPERIMENTS

We design experiments to answer the following research questions: (1) How does ECF perform, compared with related competitive methods? (2) How to measure the interpretation quality of ECF? (3) How do different components and hyper-parameters influence the performance of ECF? (4) What do the generated taste clusters look like and what are the potential applications of ECF?

3.1 Experimental Settings

We use three real-world datasets for experiments: Xbox, MovieLens and Last-FM, which vary in domain, size, tag numbers, and sparsity. The Xbox dataset is provided by Microsoft Gaming and collected from the GamePass scenario [1]. GamePass is a popular membership-based service offered by Xbox, with which users can freely play a few hundreds of high-quality games (that is why in Table 1 Xbox has only 330 items). MovieLens is a widely used dataset which contains user rating history and the types of movies, we follow [23] to get 18 neat tags for movie description. Last-FM is a public music listening dataset. We use the official APIs [2] of Last.fm to obtain the tags for each track and discard rare tags which are annotated by less than 50 people. Besides, to reduce noise, we adopt the 10-core setting, *i.e.,* retaining users and items with at least ten interactions. We use the same data partition with previous study [23] for comparison (i.e., the proportions of training, validation, and testing set are 80%, 10%, and 10% for all datasets). Table 1 summarizes the basic statistics of the three datasets. Evaluation metrics will be introduced in each subsection separately.

[1]https://www.xbox.com/en-US/xbox-game-pass
[2]https://www.last.fm/api

Table 2: Top-20 recommendation results. "†" indicates the improvement of the ECF over the baseline is significant at the level of 0.05.

	Xbox		MovieLens		Last-FM	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF	0.5048	0.3268	0.1603	0.2416	0.0658	0.0506
NCF	0.4746	0.2931	0.1606	0.2406	0.0618	0.0401
CDAE	0.5192	0.3286	0.1627	0.2499	0.0589	0.0534
LightGCN	0.4933	0.3261	0.1854	0.2698	0.0788	0.0675
EFM	0.5070	0.3312	0.1702	0.2525	0.0703	0.0549
AMCF	0.5036	0.3217	0.1604	0.2405	0.0675	0.0516
ECF$_{single}$	0.4231	0.2331	0.1068	0.1501	0.0467	0.0380
ECF	**0.5922**†	**0.3721**†	**0.2124**†	**0.2903**†	**0.0851**†	**0.0773**†

3.2 Evaluation of Accuracy (RQ1)

3.2.1 ***Baselines.*** We compare ECF with two groups of models:

- **MF** [26], **NCF** [11], **CDAE** [36] and **LightGCN** [10] are four strong and representative embedding-based CF methods which learn users' hidden preference from interactions. But they suffer from poor explainability.
- **EFM** [40] and **AMCF** [23] are two strong explainable recommendation models, but they fail to satisfy all three explanation merits. We adapt EFM model slightly to make it fit for our scenario (see Appendix A.1 for details).

We also add the single model version of ECF for variant comparison, denoted as "ECF$_{single}$".

3.2.2 ***Evaluation Metrics.*** We adopt two widely used evaluation protocols [26, 34, 40] for top-K recommendation: Recall@K and NDCG@K. We report the average metrics for all the users in the testing set. Hyper-parameters of all methods are reported in Appendix A.2.

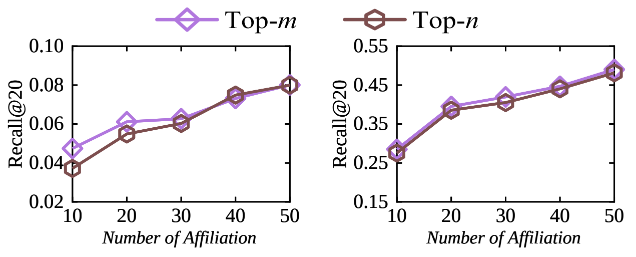
3.2.3 ***Performance Comparison.*** The performance *w.r.t.* Recall@20 and NDCG@20 is reported in Table 2. More experiments with different K can be found in Appendix A.3. In comparison, embedding-based methods achieve competitive performance on all datasets. LightGCN outperforms all other baselines on MovieLens and Last-FM datasets, we contribute this to the ability of capturing the high-order user-item relationship in bipartite graph. On the other hand, explainable methods - EFM and AMCF - only show a slight improvement over MF. As for the ECF$_{single}$, we can see the performance drops compared with MF, due to the sparsity design of taste clusters affiliations for interpretability. However, the forest mechanism in ECF makes up for its shortcomings in accuracy and outperforms all the baselines by a large margin (including the forest version of MF, see Appendix A.3 for details). This demonstrates that ECF is not taking a trade-off between accuracy and interpretability, instead, it can achieve excellent accuracy performance while providing interpretability.

3.3 Evaluation of Explainability (RQ2)

Then we evaluate the explainability of ECF. As elaborated in Section 2.4, we utilize four metrics to evaluate the explainability of our model: in-cluster item coverage, tag utilization, silhouette and informativeness. Since ECF is a cluster-based CF method, for comparison, we use three strong competitors to construct the clusters from different perspectives as baselines. Specifically, **TagCluster** is a tag-oriented method which collects items with the same tags; **K-means** is a similarity-oriented method which utilizes item embedding from FM to perform K-means algorithm; **Random** is a baseline method by randomly selecting items into clusters. The detailed implementation of these three competitors is reported in Appendix A.4.

Table 3: Explainability Evaluation of ECF. See Section 2.4 for details of four proposed metrics.

Method	Cov.	Util.	Sil.	Info.	Overall
Xbox					
ECF	0.8002	**0.7052**	0.2604	**0.3162**	**1.7463**
TagCluster	**0.9950**	0.2878	-0.1788	0.1579	0.9262
K-means	0.5710	0.3739	**0.4286**	0.0185	1.0563
Random	0.5396	0.1450	-0.3614	0.0125	0.0000
MovieLens					
ECF	0.7992	**0.7778**	0.1964	**0.3131**	**1.5651**
TagCluster	**0.991**	0.5259	-0.2573	0.1517	0.8898
K-means	0.6877	0.4478	**0.3265**	0.0168	0.9573
Random	0.5933	0.3672	-0.4452	0.0061	0.0000
Last-FM					
ECF	0.7648	**0.6259**	0.1584	**0.2996**	**1.5352**
TagCluster	**0.9880**	0.3703	-0.2511	0.1206	0.9143
K-means	0.5667	0.4841	**0.3197**	0.0182	1.0752
Random	0.5385	0.2275	-0.4673	0.0148	0.0000



(a) Last-FM dataset (b) Xbox dataset

Figure 1: Impact of Top-m and Top-n.

To have an overall understanding of the explainability, we introduce an aggregated metric denoted as *Overall* in Table 3. To overcome the scale inconsistency issue in different metrics, we first normalize each individual metric by calculating the relative gain over the *Random* baseline, then sum up the normalized four scores as the *Overall* metric. In terms of the overall metric, ECF outperforms all competitors by a large margin. Specifically, all the baseline methods can only preserve a certain aspect while ignoring the rest three aspects. For instance, since TagCluster only considers the explicit tags for clustering, it fails to preserve the hidden collaborative similarity in the cluster. Besides, traditional clustering methods like K-means suffer from low coverage ratio and informativeness, indicating the inability to construct meaningful taste clusters with representative tags. In contrast, ECF takes all aspects into consideration so that it can avoid obvious shortcomings on a certain metric.

3.4 Study of ECF (RQ3)

As the taste clusters learning is the core of ECF, we also conduct ablation studies to investigate the effectiveness of each component.

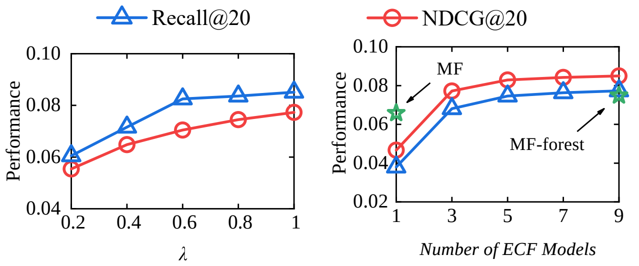


Figure 2: (a) Impact of λ on Last-FM dataset. (b) Impact of the forest mechanism.

3.4.1 ***Impact of Top-m and Top-n.*** As described in Section 2.2, we use Top-m and Top-n to select the most relevant taste clusters for items/users. The experimental results of different m and n on Xbox and Last-FM dataset are reported in Figure 1, while the results on MovieLens dataset is omitted due to similar trend and limited space. We observe that the recommendation performance boosts dramatically when m and n increase, especially from 10 to 20. This is because when the affiliations between taste clusters and users/items are sparse, it is unable to model the complex and latent users' preference, as well as the multi-faceted semantics of items. However, large m and n would make the explanation results hard to understand because of the excessive number of reachable paths between users, taste clusters and items. Therefore, to balance the readability and accuracy, we set m and n to 20 for all datasets.

3.4.2 ***Impact of λ.*** We then verify the effectiveness of the auxiliary collaborative signals. The results on Last-FM dataset is illustrated in the left of Figure 2, and the results on other two datasets are reported in Appendix A.6. We notice that when λ is small, the performance of ECF degrades dramatically due to lack of supervised signals for taste clusters learning. Besides, when λ is larger than 0.6, the performance improvement is marginal, which indicates ECF is able to be optimized with sufficient collaborative signals. Thus, we set λ to 0.6 for all experiments.

3.4.3 ***Impact of Forest Mechanism.*** To analyze the effect of the ensemble version of ECF, we gradually increase the size of forest, and the performance results are reported in the right of Figure 2. We find that there is a notable performance gain of ECF, compared with the single ECF model. We contribute the improvement to the holistic modeling of users' latent interest space with diverse taste clusters. Besides, we also find that the performance of ECF is even better than the forest version of MF (see Figure 2 and Appendix A1 for details), which indicates the effectiveness of taste cluster learning.

3.5 Case Study and Applications (RQ4)

In this section, we present the examples of Last-FM dataset to give an intuitive impression of taste clusters, as well as the explainability of ECF. We also provide more useful scenarios beyond item recommendations to illustrate the ability of ECF.

3.5.1 ***Learned Taste Clusters.*** We first illustrate the generated five taste clusters in Figure 3. Each taste cluster is interpreted with four tags, which are the best description of the tracks within. For instance, all the tracks in taste cluster c_3 are soundtrack, while some are ambient music (*e.g.,* "Hand Covers Bruise"), and some are tagged with "alternative" (*e.g.,* "Only Hope"). Besides, each taste cluster is able to explicitly represent certain users' preference. Taking the taste cluster c_2 as an example, users who have a high affiliation score with it tend to favor rap and hip-pop music, especially these tracks that are accompanied with dancing. Moreover, different taste clusters have different tags (*e.g.,* pop music in c_1 and folk music in c_5), indicating the diversity of users' preference.

It is also interesting to investigate the relationship between the tags of taste clusters and tags of items. We find taste clusters have the ability to serve as complementary information to the tagging of items, which we call *tag discovery.* For instance, folk song "Bubbly" in taste cluster c_5 is tagged with "female_vocalists|pop|folk|acoustic|love" by Last.fm. Thus, we can suspect whether "Bubbly" has the missing tag "singer_songwriter" since other tags in the taste cluster c_5 (*i.e.,* "folk|acoustic|pop") are perfectly to describe the track. In fact, the singer of the song, Colbie Caillat, is also a songwriter who wrote the song. Therefore, taste clusters could be a useful tool to check the missing tags of items within.

3.5.2 ***Recommendation Explanation.*** Then we demonstrate the explanation ability of ECF. Taking the recommendation of user u_{71} to track i_{77414} ("Bubbly") as an example, Figure 4 shows the listening history of u_{71} as well as the recommendation logic of ECF, and we have the following observations:

- ECF maps users as well as items with their corresponding taste clusters. For user u_{71}, since he/she has listened different kinds of songs from soundtrack to pop and soul music, three different taste clusters (*i.e.,* c_1, c_3 and c_4.) have been affiliated to reveal his interest. Similarly, track i_{77414} is also connected with three taste clusters (*i.e.,* c_1, c_4 and c_5), which is able to describe the song from different aspects. Besides, the weights of affiliation matrix indicate the relatedness between users/items with taste clusters.
- To recommend track i_{77414} to user u_{71}, ECF first finds their interactions of taste clusters according to the affiliations (*i.e.,* c_1, c_4), which are depicted as red in Figure 4. These taste clusters serve as a bridge to explicitly connect user's preference and items' semantics. Then two explanation paths ($i_{71} \rightarrow c_1 \rightarrow i_{77414}$ and $i_{71} \rightarrow c_4 \rightarrow i_{77414}$) are constructed for recommendation. For each path, we multiply their affiliation weight to generate the importance score of the taste cluster, according to Equation (1). At last, by summarizing all the scores over intersected taste clusters, we can calculate the final prediction score, as well as the explanation paths from user u_{71} to i_{77414}.

3.5.3 ***Taste Cluster Recommendation.*** Here we propose a new recommendation scenario, called *taste cluster recommendation*, which means users are recommended with a bundle of similar items, and these items can be described with few tags (or features). We find this task is ubiquitous in real-world applications, like playlist recommendation in Spotify [13], or game recommendation in Xbox. Current solutions to tackle this problem typically rely on manual selection by editors to guarantee the selected items are similar in tags (or other explicit features). However, this approach suffers from tedious selection process and poor personality. Our method could be a substitution for labor selections by learning taste clusters in an end-to-end manner. To be specific, since each learned taste cluster is a collection of items with descriptive tags, it can be naturally

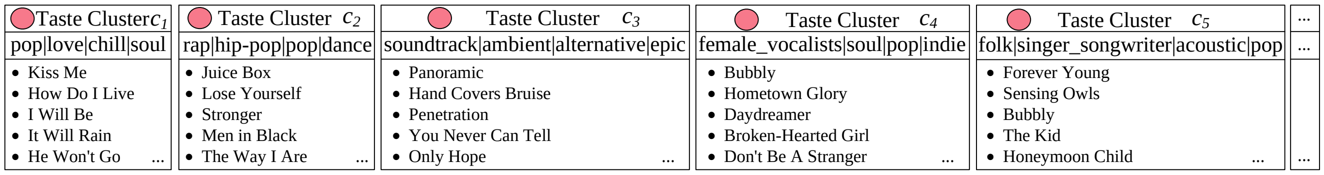


Figure 3: 5 real examples of learned taste clusters on Last-FM dataset.

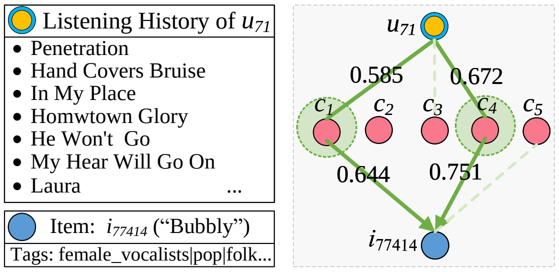


Figure 4: Explanations of the recommendation of user u_{71} to track i_{77414} ("Bubbly") on Last-FM dataset. c denotes the taste cluster which shows in Figure 3. The tags of the track i_{77414} are "female_vocalists|pop|folk|acoustic|love".

Table 4: The Recommendation Performance and Explainability of ECF and ECF$_{LGN}$ on Last-FM dataset.

	Performance		Explainability				
	R@20	N@20	Cov.	Util.	Sil.	Info.	Overall
ECF	0.0851	0.0773	0.7648	0.6259	0.1584	0.2996	1.5352
ECF$_{LGN}$	**0.0876**	**0.0792**	**0.7831**	**0.6430**	**0.1590**	**0.3042**	**1.5758**

viewed as a playlist or game set. Thus, we can directly recommend taste clusters to users according to their affiliation weights. Taking user u_{71} in Figure 4 as an example, ECF could recommend him/her with a pop song playlist which is sung by female artists, since he/she used to listen lots of pop music by female artists, like Colbie Caillat, Adele and Celine Dion.

3.5.4 ***User Profiling.*** The user-cluster affiliations discovered by ECF can also be used as user profiles directly. Take Figure 4 for example, ECF assigns user u_{71} to three clusters: c_1, c_3 and c_4. Then we can use the tag summary of these clusters as (part of) her profile. Such kind of clusters as user profiles can benefit various web applications, including (1) user-level predictive tasks, such as user demographic attributes prediction and user churn/uplift prediction [6, 12], where cluster profiles can be consumed as auxiliary features; (2) ad audience targeting [25, 30], where ad platforms or advertisers can leverage user clusters to reach a certain segment of users that have targeted interests and habits in an interpretable manner; (3) look-alike audience extension [18, 20], where the task is to reach users similar to a given set of seed users, especially with the goal of enhancing the performance of cold-start and long tail items and improving the recommendation diversity. User clusters are naturally suitable for this task because similar users have already been indexed in the same clusters.

3.6 Flexibility of ECF

Since the ECF framework only relies on embeddings to learn explainable taste clusters, it can be viewed as a model-independent explainable paradigm, which is easily applied with other popular embedding-based methods. Thus, we also incorporate the ECF framework with LightGCN [10], a competitive graph-based embedding method, to investigate its flexibility. Specifically, we use the item embeddings of the last layer in LightGCN for taste clusters learning, and the auxiliary collaborative signal is also replaced with the prediction loss from LightGCN. We denote this variant as ECF$_{LGN}$ and the results of both performance and explainability on Last-FM dataset are shown in Figure 4. The results indicate that the performance and explainability of ECF would also benefit from more complex embedding-based methods, which demonstrates the superiority and universality of ECF.

4 RELATED WORKS

Collaborative filtering techniques, especially model-based CF methods, have achieved great success in personalized recommender systems. Most (if not all) of those methods rely on the *latent* factor embeddings to generate rating predictions, which makes it difficult to give explanations for the predictions. However, explaining why a user likes an item can be as important as the accuracy of the prediction itself [39], since it can not only enhance the transparency and trustworthiness of the system, but also significantly improve user satisfaction. Therefore, varieties of strategies for rendering explainable recommendations have been proposed. We first review explainable collaborative filtering methods [1, 5, 9, 22, 23, 29, 31, 37, 40], which are closely related to our work. Then we discuss other recent explainable recommendation works.

Explainable collaborative filtering methods. EMF [1] adds an explainability regularizer into the objective function of MF to force user/item hidden vectors to be close if a lot of the user's neighbors also purchased the item. HFT [22] leverages the topic model to obtain interpretable textual labels for latent rating dimensions. EFM [40] factorizes a rating matrix in terms of both explicit phases from textual reviews as well as latent factors, and makes predictions by combing the scores of both explicit and implicit features. TriRank [9] models user-item-tag ternary relation as a heterogeneous tripartite graph and performs vertex ranking for recommendation. ATM [5] utilizes latent factors to estimate tag importance of a user towards an item, and makes predictions via a linear combination of tag ratings. More recently, AMCF [23] maps uninterpretable embeddings into the interpretable aspect features by minimizing both ranking loss and interpretation loss. Nonetheless, those methods are unable to satisfy all three important explanation properties: flexible, coherence and self-explainable, resulting in ineffectiveness to boost the transparency of recommender systems.

Other directions of explainable recommendation. There is also a large literature that considers other auxiliary information or other methods to explain recommendations [1, 3, 4, 7, 8, 24, 32–35]. For example, NARRE [3] designs an attention mechanism over the user and item reviews for rating prediction, and leverages the attention weight for explanation. TEM [33] devices a tree-enhanced embedding method to learn decision rules from cross-features. VECF [4] proposes visually explainable recommendation based on personalized region-of-interest highlights by a multimodal attention network. PLM-Rec [8] learns a language model over knowledge graph to generate explainable paths. Those methods leverage more expressive data for explanation, which can be the future direction of our ECF framework.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a neat yet effective explainable collaborative filtering framework, called ECF, which is able to learn informative taste clusters in an end-to-end manner, and perform both recommendation and explanation by measuring the coherence between user-/item- cluster affiliations. ECF is the first explainable framework which is flexible, coherent and self-explainable. We also design several quantitative metrics to evaluate the explainability quality of taste clusters in a comprehensive manner. Extensive experiments conducted on three real-world datasets demonstrate the superiority of ECF from both recommendation accuracy and explainability. We also devise two more real-world scenarios to show the applications of learned taste clusters. In the future, we aim to apply ECF to real-world recommendation scenarios with millions of users and items to improve its scalability. Besides, we also plan to incorporate ECF with more expressive data beyond tags, such as reviews [3], knowledge graphs [34] to fully exploit the power of this framework.

REFERENCES

[1] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable matrix factorization for collaborative filtering. In *WWW*. 5–6.

[2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *ICML*. 531–540.

[3] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *WWW*. 1583–1592.

[4] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized Fashion Recommendation with Visual Explanations Based on Multimodal Attention Network: Towards Visually Explainable Recommendation. In *SIGIR*. 765–774.

[5] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW*. 639–648.

[6] Eustache Diemert, Artem Betlei, Christophe Renaudin, and Massih-Reza Amini. 2018. A large scale benchmark for uplift modeling. In *KDD*.

[7] Francesco Fusco, Michalis Vlachos, Vasileios Vasileiadis, Kathrin Wardatzky, and Johannes Schneider. 2019. RecoNet: An Interpretable Neural Architecture for Recommender Systems.. In *IJCAI*. 2343–2349.

[8] Shijie Geng, Zuohui Fu, Juntao Tan, Yingqiang Ge, Gerard De Melo, and Yongfeng Zhang. 2022. Path Language Modeling over Knowledge Graphsfor Explainable Recommendation. In *WWW*. 946–955.

[9] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-Aware Explainable Recommendation by Modeling Aspects. In *CIKM*. 1661–1670.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[12] Bingquan Huang, Mohand Tahar Kechadi, and Brian Buckley. 2012. Customer churn prediction in telecommunications. *Expert Systems with Applications* 39 (2012), 1414–1425.

[13] Kurt Jacobson, Vidhya Murali, Edward Newett, Brian Whitman, and Romain Yon. 2016. Music personalization at Spotify. In *RecSys*. 373–373.

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

[15] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[18] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. 2019. Real-time attention based look-alike model for recommender system. In *KDD*. 2765–2773.

[19] Zheng Liu, Yu Xing, Fangzhao Wu, Mingxiao An, and Xing Xie. 2019. Hi-Fi Ark: Deep User Representation via High-Fidelity Archive Network.. In *IJCAI*. 3059–3065.

[20] Qiang Ma, Musen Wen, Zhen Xia, and Datong Chen. 2016. A sub-linear, massive-scale look-alike audience extension system a massive-scale look-alike audience extension. In *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 51–67.

[21] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *CIKM*. 1253–1262.

[22] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.

[23] Deng Pan, Xiangrui Li, Xin Li, and Dongxiao Zhu. 2020. Explainable Recommendation via Interpretable Feature Mapping and Evaluation of Explainability. In *IJCAI*. 2690–2696.

[24] Georgina Peake and Jun Wang. 2018. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *KDD*. 2060–2069.

[25] Foster Provost, Brian Dalessandro, Rod Hook, Xiaohan Zhang, and Alan Murray. 2009. Audience Selection for On-Line Brand Advertising: Privacy-Friendly Social Network Targeting. In *KDD*. 707–716.

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

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[28] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. 2007. Measuring and testing dependence by correlation of distances. *The annals of statistics* 35, 6 (2007), 2769–2794.

[29] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *IJCAI*. 2640–2646.

[30] Jian Tang, Ning Liu, Jun Yan, Yelong Shen, Shaodan Guo, Bin Gao, Shuicheng Yan, and Ming Zhang. 2011. Learning to Rank Audience for Behavioral Targeting in Display Ads. In *CIKM*. 605–610.

[31] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *IUI*. 47–56.

[32] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *SIGIR*. 165–174.

[33] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced embedding model for explainable recommendation. In *WWW*. 1543–1552.

[34] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *WWW*. 878–887.

[35] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*. 5329–5336.

[36] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.

[37] Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*. 199–208.

[38] Peitian Zhang and Zheng Li. 2022. GateFormer: Speeding Up News Feed Recommendation with Input Gated Transformers. *arXiv preprint arXiv:2201.04406* (2022).

[39] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* 14, 1 (2020), 1–101.

[40] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.

A APPENDIX

A.1 Adaptation of EFM

Since EFM relies on the phase-level reviews for prediction and explanation, we make a few modifications to fit our settings where only tags of items are available. First, we generate the user-feature matrix by counting all the features of items that users have interacted with, which means that all the mentioned features are considered as positive. Second, the negative opinions of item's features are discarded because no auxiliary information is available. Third, the item-feature quality matrix is simplified as a 0-1 matrix where 1 denotes the item has the tag in attributes, while 0 denotes otherwise. After reconstructing the user-feature attention matrix and item-feature quality matrix, we run the EFM model as it describes in the paper.

A.2 Hyper-Parameter Settings

We optimize all models with Adam [15], where the batch size is fixed at 1024. For CDAE, the hidden size is 100, and the dropout ratio is set to 0.5. For LightGCN, we set the number of layers to 3, and the dropout ratio is also set to 0.5. For EFM, we fix the percentage of explicit factors to 40%, and tune the coefficient of feature-based score for best performance. For AMCF, we use their official codes (https://github.com/pd90506/AMCF) for implementation and set the hyperparameter of feature mapping as 0.05. For ECF, we set $m = 20$ and $n = 20$, which means that we restrict that each item and item are associated with only 20 clusters. The temperature hyperparameters T and τ are all set as 2. We also set $|C| = 64$ for all datasets, which indicates each ECF model could generate 64 taste clusters. Besides, the number of single ECF model F is set as 9 for all experiments. Moreover, early stopping strategy is performed for all methods, *i.e.,* premature stopping if recall@20 on the test set does not increase for 10 successive epochs.

Besides, since ECF utilize sparse affiliations between users/items and taste clusters instead of embeddings to perform recommendation, we choose two options for fair comparison: (1) we fix the size of ID embeddings (users, items and taste clusters) as 64 for all methods; (2) since the affiliation size of ECF is 180 (top 20 affiliated taste clusters for each users/items, and we use 9 single ECF model), we align it with the embedding size of all other methods, and set the size of embeddings as 180. We conduct both experiments for all baselines and select the best performance for the report.

A.3 Complete Recommendation Performance

We report the comprehensive recommendation performance in Table A1, where $K = 20$ is omitted since the results are already reported in Table 2. From the results we see a similar trend as we discussed in Section 3.2: embedding-based methods (*e.g.,* MF and LightGCN) show strong performance across different datasets yet suffer from poor explanation; explainable methods can only slightly outperform MF, since they need to balance the trade-off between accuracy and transparency. However, ECF achieves the best performance in all ranking metrics (including the forest version of MF). Therefore, our model is able to not only provide accurate recommendation results, but also intuitive and clear explanations for each prediction with taste clusters.

A.4 Baselines for Explanation Evaluation

- **TagCluster** is a tag-oriented method which collects items according to their tags. To be specific, we first randomly sample $|C|$ items as the seed of clusters, and search for the items which have the same tags with them. When there is no item that has the exactly the same tags as the seed, we relax it by randomly discarding one tag and continue to search. We iteratively conduct the operations until the size of clusters meets threshold Z. And we choose the 4 most frequent tags in the clusters as the tags of taste clusters.
- **K-means** is a similarity-oriented method which leverages the item embeddings from FM to perform K-means clustering algorithm. The result clusters are as directly used as the taste clusters. And we also select the top-4 most common tags as the descriptive tags for taste clusters.
- **Random** is a baseline method which randomly select K items to construct the clusters. The procedure is repeated until all items are allocated in at least one cluster. We use the same mechanism as above methods to generate tags for each clusters.

A.5 Ablation Study *w.r.t.* λ

We also investigate the impact of different λ on other two datasets, and the results are shown in Figure A1. We find that as the increase of auxiliary collaborative signals, the performance of ECF improves gradually. However, when the weighting coefficient λ reaches beyond 0.6, the improvement becomes marginal. Thus, we set $\lambda = 0.6$ for all experiments.

A.6 Ablation Study *w.r.t.* Forest Mechanism

We briefly discuss how the number of single ECF model influences the performance. Since the single ECF model is unable to model the complex and hidden users' interest space at once, more learned taste clusters may help to alleviate this issue. We confirm this assumption by adding more single ECF model for ensemble prediction, and the results on Xbox and MovieLens dataset are illustrated in Figure A2. We can see the huge performance gains when we increase the number of ECF models, especially from a single model to three models. However, as the number of single models continuously increases, the improvements become negligible. We set the number of ECF models to 9 for all experiments.

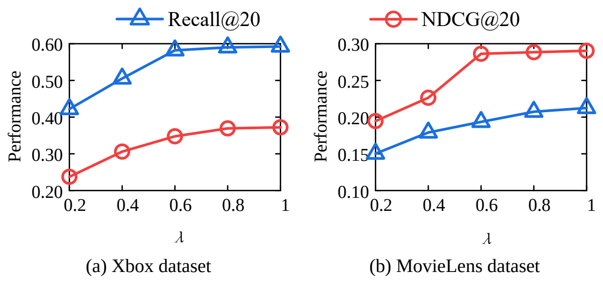
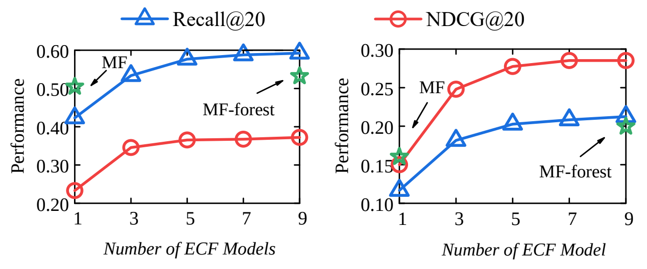


Figure A1: Impact of λ in Xbox and MovieLens datsets.

Table A1: Complete Top-K recommendation results. "†" indicates the improvement of the ECF over the baseline is significant at the level of 0.05. R and N refer to Recall and NDCG, respectively.

	Xbox				MovieLen				Last-FM			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
MF	0.2615	0.3686	0.2383	0.2824	0.0601	0.0975	0.2738	0.2511	0.0289	0.0446	0.0428	0.0443
NCF	0.2372	0.3433	0.2065	0.2503	0.0594	0.0985	0.2701	0.2517	0.0269	0.0456	0.0396	0.0383
CDAE	0.2604	0.3738	0.2346	0.2813	0.0609	0.0946	0.2671	0.2534	0.0286	0.0402	0.0431	0.0518
LightGCN	0.2684	0.3625	0.2382	0.2837	0.0699	0.1163	0.2979	0.2752	0.0398	0.0578	0.0605	0.0634
EFM	0.2647	0.3652	0.2368	0.2873	0.0657	0.1027	0.2866	0.2635	0.0319	0.0482	0.0471	0.0484
AMCF	0.2601	0.3613	0.2355	0.2806	0.0603	0.0986	0.2719	0.2498	0.0295	0.0488	0.0456	0.0457
MF$_{forest}$	<u>0.2907</u>	<u>0.3983</u>	<u>0.2615</u>	<u>0.3159</u>	<u>0.0787</u>	<u>0.1276</u>	<u>0.3122</u>	<u>0.2911</u>	<u>0.0374</u>	<u>0.0548</u>	<u>0.0562</u>	<u>0.0594</u>
ECF$_{single}$	0.1714	0.2763	0.1423	0.1854	0.0352	0.0608	0.1584	0.1505	0.0205	0.0315	0.0339	0.0345
ECF	**0.2970**†	**0.4299**†	**0.2644**†	**0.3193**†	**0.0788**	**0.1325**†	**0.3183**†	**0.2952**†	**0.0455**†	**0.0635**†	**0.0782**†	**0.0749**†



(a) Xbox dataset (b) MovieLens dataset

Figure A2: Impact of forest mechanism on Xbox and MovieLens datasets.

A.7 Implementation of Discriminator

We opt for a simple Multi-Layer Perceptron (MLPs) as the post-hoc discriminator. Specifically, it is a three-layer MLP (hidden size is 64) with ReLU as the activate function, cross-entropy as the loss function. We utilize the tags of items as the input data, and feed them into the MLPs to predict the item it belongs with. To improve the robustness of the model, we randomly mask 50% of the item's tags for training. After training, we use the tags of taste clusters as the input, and collect top $|c_i|$ items as $R(\mathcal{T}_{c_i})$ by ranking their prediction probabilities.