$\mathtt{EX}^3$: Explainable Attribute-aware Item-set Recommendations

YIKUN XIAN, Rutgers University, United States
TONG ZHAO, JIN LI, and JIM CHAN, Amazon, United States
ANDREY KAN, JUN MA, and XIN LUNA DONG, Amazon, United States
CHRISTOS FALOUTSOS, Amazon/Carnegie Mellon University, United States
GEORGE KARYPIS, Amazon/University of Minnesota, United States
S. MUTHUKRISHNAN, Amazon/Rutgers University, United States
YONGFENG ZHANG, Rutgers University, United States

Existing recommender systems in the e-commerce domain primarily focus on generating a set of relevant items as recommendations; however, few existing systems utilize underlying item attributes as a key organizing principle in presenting recommendations to users. Mining important attributes of items from customer perspectives and presenting them along with item sets as recommendations can provide users more explainability and help them make better purchase decision. In this work, we generalize the attribute-aware item-set recommendation problem, and develop a new approach to generate sets of items (recommendations) with corresponding important attributes (explanations) that can best justify why the items are recommended to users. In particular, we propose a system that learns important attributes from historical user behavior to derive item set recommendations, so that an organized view of recommendations and their attribute-driven explanations can help users more easily understand how the recommendations relate to their preferences. Our approach is geared towards real world scenarios: we expect a solution to be scalable to billions of items, and be able to learn item and attribute relevance automatically from user behavior without human annotations. To this end, we propose a multi-step learning-based framework called Extract-Expect-Explain ($\mathtt{EX}^3$), which is able to adaptively select recommended items and important attributes for users. We experiment on a large-scale real-world benchmark and the results show that our model outperforms state-of-the-art baselines by an *11.35%* increase on NDCG with adaptive explainability for item set recommendation.

Additional Key Words and Phrases: Recommender system, Explainable recommendation, Item set recommendation

ACM Reference Format:
Yikun Xian, Tong Zhao, Jin Li, Jim Chan, Andrey Kan, Jun Ma, Xin Luna Dong, Christos Faloutsos, George Karypis, S. Muthukrishnan, and Yongfeng Zhang. 2021. $\mathtt{EX}^3$: Explainable Attribute-aware Item-set Recommendations. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460231.3474240

1 INTRODUCTION

Recommender systems have been widely deployed in modern e-commerce websites, helping users overcome overwhelming selection issues in large catalogs and contributing large business impact [9, 20, 32]. Many existing recommender systems in industry focus on generating a set of relevant items based on a set of pivot/query items along with metadata such as item attributes. However, few of them utilize the underlying item attributes as a way to explain why the items are recommended to users. Without distinguishing attributes, recommendations can often be overlooked by users who are unfamiliar with the items [23], especially when they have to click into corresponding detail pages to find more in-depth information. In this work, we generalize the attribute-aware item-set recommendation problem [3, 4, 10, 28], which aims to generate exact-K sets of recommended items along with attribute-driven explanations to help users quickly locate the items of interest according to objective item properties (brand, color, size, etc) and subjective user feedback (ratings). In particular, we propose a method to learn behavior-oriented attribute importance from historical user actions, a technique which can be applied to other use cases beyond explainable recommendations including query rewriting [25] and review summarization [40].

Throughout the paper, we study the explainable attribute-aware item-set recommendation problem by learning an item-to-item-set mapping guided by attribute differences. Formally, *given a "pivot" item, our goal is to generate K sets of items (recommendations), each of which is associated with an important attribute (explanation) to justify why the items are recommended to users*. We aim to not only generate relevant item recommendations, but also provide corresponding explanations based on those important item attributes whose value changes will affect user purchase decision. Unlike existing work [3] that focuses primarily on making understandable substitute recommendations, we attempt to help users broaden their consideration set by presenting them with differentiated options by attribute type. Additionally, different from generating explanations based on user–item and item–attribute interactions [3], we propose to infer important attributes directly from users' historical behaviors, providing a framework to understand how users reason about recommendations when making decisions. To the best of our knowledge, we are the first to approach the explainable item-set recommendations via behavior-oriented important attribute identification in e-commerce domain.

The main idea in solving this problem is to first learn important attributes based on users' historical behaviors, and then generate corresponding item recommendations. Note that learning important attributes can benefit many other applications beyond item-set recommendations alone. Modeling behavior-oriented attribute importance from users' historical actions rather than manual identification is a critical component to conduct explainable recommendations. It saves time-consuming effort in manual labeling and provides a more robust way to model user preference. Once important attributes are derived, we can utilize them to build user profiles, e.g., identifying users' preferred size, color, flavor, etc, which can be used in generating personalized recommendations. We can also perform brief item summarization based on important attributes, and the proposed method can also be easily extended to involve more contextual information (e.g., users' sequential actions) to provide customized item summarization [40]. We can further leverage the behavior-driven important attributes to advance query rewriting techniques in the item search domain, by attending to those terms that are closely related to items' important attributes.

To this end, we propose a multi-step framework called *Extract-Expect-Explain* ($\texttt{EX}^3$) to approach the explainable item-set recommendation problem. Our $\texttt{EX}^3$ framework takes as input a pivot/query item and a list of candidate items as well as their catalog features (e.g., title, item type), and adaptively outputs sets of recommended items associated with important attributes as explanations. Specifically, in the first *Extract-step*, we introduce an attention-based item embedding learning framework, which is scalable to generating embeddings for billions of items, and can be leveraged to refine coarse-grained candidate items for a given pivot item. Then, in the *Expect-step*, we propose an Attribute-Differentiating Network to learn the expected utility score on the tuples of {query item, candidate item, attribute} to indicate how the difference on attribute values between query item and candidate item affects users' purchase decision. The goal of this step is to learn attribute importance based on the impact of value changes towards user purchase behavior. For instance, if we observe that value changes of "shoe size" affected more user purchase decisions than color changes, the *Expect-step* is more likely to predict higher utility score on {query shoe, candidate shoe, size} than {query shoe, candidate shoe, color}. Given the refined candidate items and the estimated utility scores, we propose a bipartite b-Matching-based algorithm in the *Explain-step* to balance the relevance and behavior-driven attribute importance to deliver the final results for item-set recommendation. Such a multi-step framework design provides the flexibility to serve the explainable attribute-aware item-set recommendation and other relevant applications.

To guarantee the robustness and scalability in real world environment, $\texttt{EX}^3$ is carefully designed to overcome several inherent challenges. (1) *The foremost challenge is how to dynamically recommend items and attributes that provide comprehensive information contributed to users' purchase decision.* In this work, we propose to train $\texttt{EX}^3$ with user behavior signals in the distant supervision manner, and leverage attribute value difference and historical purchase signals to capture user-behavior driven important attributes. We believe that the important attributes are those whose value changes will critically affect users' purchase decision, e.g., size for shoes, roast type for coffee. (2) *In real-world environment, we are always facing data challenges, especially on the attribute missing/sparsity issues.* To have a robust performance even when attribute coverage is poor, we develop a robust attention mechanism called Random-masking Attention Block in *Expect-step* to bound the softmax output based on prior attribute coverage information. (3) *Scaling $\texttt{EX}^3$ to millions of different items is also challenging.* To ensure $\texttt{EX}^3$ to be generalized to multiple item types and large-scale items, we introduce a highly-scalable item embedding framework in *Extract-step*, design an attribute-driven attention mechanism in *Expect-step* to directly learn attribute importance from user behavior without human labeling, and propose a constrained bipartite b-Matching algorithm in *Explain-step* that can be easily parallelized to generate top items and important attributes for explainable item-set recommendation. The contributions of this paper are three-fold.

- We highlight the importance of jointly considering important attributes and relevant items in achieving the optimal user experience in explainable recommendations.
- We propose a novel three-step framework, $\texttt{EX}^3$, to approach the explainable attribute-aware item-set recommendation problem along with couples of novel components. The whole framework is carefully designed towards large-scale real-world scenario.
- We extensively conduct experiments on the real-world benchmark for item-set recommendations. The results show that $\texttt{EX}^3$ achieves 11.35% better NDCG than state-of-the-art baselines, as well as better explainability in terms of important attribute ranking.

2 PRELIMINARY

In this section, we start with the introduction of relevant concepts and formulation of the explainable attribute-aware item-set recommendation problem. Then, we introduce how to approach this problem via distant supervision.

Problem Formulation. Let $\mathcal{P}$ be the universal set of items and $\mathcal{A}$ be the set of all available attributes. We define the attribute value to be a function $v : \mathcal{P} \times \mathcal{A} \mapsto C^{d_v}$ that maps an item and an attribute to a sequence of characters, where C denotes a set of predefined characters and d_v is the maximum length of the sequence.[1] An item $p \in \mathcal{P}$ is said to have value $v(p, a)$ on attribute $a \in \mathcal{A}$ if $v(p, a) \neq \emptyset$. Accordingly, the attribute-value pairs of an item p on multiple attributes are defined as $A_p = \{(a_1, v_1), \ldots, (a_{|A_p|}, v_{|A_p|}) \mid a_i \in \mathcal{A}, v_i = v(p, a_i), v_i \neq \emptyset, i = 1, \ldots, |A_p|\}$. In addition, we define an *explainable group* $G_a = (a, P_a)$ to be a tuple of an attribute $a \in \mathcal{A}$ and a subset of items $P_a \subset \mathcal{P}$ and each item in P_a has non-empty value on attribute a. The item set P_a is assumed for recommendation and the attribute a is used to generate the explanation. The problem of explainable attribute-aware item-set recommendation can be formalized as follows.

[1]Note that in practice an attribute value can be of arbitrary data types such as string, numeric, timestamp. In this work, we regard it to be string (character sequence) for simplicity since any other types can be converted to a string.

Definition 1 (Problem Definition). *Given the pivot item $q \in \mathcal{P}$ with attribute-value pairs A_q, and the number of groups, K, the goal is to output K ordered explainable groups $G_{a_{(1)}}, \ldots, G_{a_{(K)}}$ such that the user utility (e.g., purchase) of displaying K such groups is maximized.*

Intuitively, the goal of the problem is to recommend K groups of items with attributes such that the likelihood of these recommended items being clicked or purchased is maximized after users compare them with the pivot item and view the displayed attribute-based justifications. In other words, it is required to generate important attributes given different pivot and candidate items so that they are useful to users, e.g., "screen resolution" is relatively more important than "height" for a TV item. Note that the explainable item set recommendation can be considered to be a item-to-item-set recommendation problem in e-commerce shopping scenario, and we assume user context information is not available in this work. The challenges of this problem are threefold.

- How to automatically identify important attributes without supervision and aggregate relevant items into the corresponding groups for recommendation?
- How to make the model robust to the data issues including missing attributes and noisy and arbitrary values?
- How to effectively reduce the search space of seeking similar items for item set recommendation and make the model scalable to large real-world dataset?

Distant Supervision. In order to capture the comparable relationship among various items, we consider three common user behavior signals to construct datasets to provide distant supervision [11, 20, 30]: co-purchase ($\mathcal{B}_{\text{cp}}$), co-view ($\mathcal{B}_{\text{cv}}$) and purchase-after-view ($\mathcal{B}_{\text{pv}}$) between items, where $\mathcal{B}_{\text{cp}}, \mathcal{B}_{\text{cv}}, \mathcal{B}_{\text{pv}} \subseteq \mathcal{P} \times \mathcal{P}$ denote how items are co-purchased, co-viewed and view-then-purchased together. From the above definition, one can notice that $\mathcal{B}_{\text{pv}}$ offers an opportunity to simulate users' shopping behaviors. When users view an item and then purchase another one in a short period of time (e.g., within the same session), it is reasonable to assume that users are making comparison between relevant items. Through empirical analysis on Amazon Mechanical Turk (MTurk), we observe that item pairs within $\mathcal{B}_{\text{pv}}$ have more than 80% similarities, which verifies our assumption that users are comparing similar items before purchase. In order to further improve the relevance from raw behavior signals to build up distant supervision with high quality, by further combing $\mathcal{B}_{\text{cp}}$ and $\mathcal{B}_{\text{cv}}$, we conduct several annotation experiments via MTurk and observe that $\mathcal{B} = \mathcal{B}_{\text{cv}} \cap \mathcal{B}_{\text{pv}} - \mathcal{B}_{\text{cp}}$, which contains items pairs in both $\mathcal{B}_{\text{cv}}$ and $\mathcal{B}_{\text{pv}}$ but not in $\mathcal{B}_{\text{cp}}$, gives us the best relevance signals and mimics users' shopping actions on $\mathcal{B}_{\text{pv}}$. Throughout the paper, we will use this way to construct datasets for model learning and offline evaluation on multiple item categories.

3 PROPOSED METHOD

In this section, we first formulate an optimization-based method for the explainable attribute-aware item-set recommendation problem and pose several potential issues of this solution in industrial scenario. Then, we propose a novel learning-based framework called Extract-Expect-Explain (EX^3) as a feasible and scalable alternative.

An Optimization-based Method. Suppose we have a utility function $u(q, p, a)$ that estimates how likely users will click (or purchase) a recommended item $p \in \mathcal{P}$ after comparing it with the pivot item $q \in \mathcal{P}$ on attribute $a \in \mathcal{A}$, i.e., $u : \mathcal{P} \times \mathcal{P} \times \mathcal{A} \mapsto [0, 1]$. We can formulate an optimization problem for explainable item set recommendation as follows. Given a pivot item q, m candidate items $\{p_1, \ldots, p_m\} \subseteq \mathcal{P}$ and n attributes $\{a_1, \ldots, a_n\} \subseteq \mathcal{A}$, we aim to find an

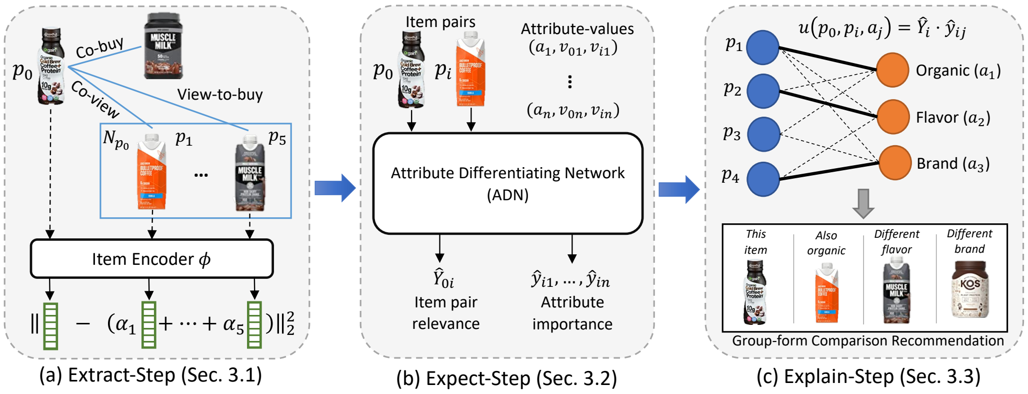


Fig. 1. Illustration of the proposed framework Extract-Expect-Explain ($\texttt{EX}^3$).

assignment $X \in \{0,1\}^{m \times n}$ that maximizes the overall utilities subject to some constraints:

$$\begin{aligned} \max_{X} \quad & \sum_{i \in [m], j \in [n]} u(q, p_i, a_j) X_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{m} X_{ij} \le D_{\text{grp}}, \ \forall j \in [n] \qquad \text{(Group capacity constraint)} \\ & \sum_{j=1}^{n} X_{ij} \le D_{\text{div}}, \ \forall i \in [m] \qquad \text{(Item diversity constraint)} \end{aligned} \tag{1}$$

where $X_{ij} = 1$ means the item p_i is assigned to the explainable group G_{a_j} with attribute a_j, and otherwise $X_{ij} = 0$. The *group capacity constraint* restricts the max number of items assigned in each group with an upperbound $D_{\text{grp}} \in \mathbb{N}$, while the *item diversity constraint* limits the occurrence of each item in overall recommendations with upperbound $D_{\text{div}} \in \mathbb{N}$. The problem defined in Eq. 1 can be deemed as the *weighted bipartite b-matching problem* [22], which can be solved by modern LP solvers. Once the n sets of item assignments are derived from X, we can easily select top-K groups with any heuristic method based on group-level utility, e.g., the average of all item-attribute utilities in the group.

However, there are two major issues with this method. First, the optimization in Eq. 1 cannot be efficiently solved when m is very large and let alone take all items in $\mathcal{P}$ as input. Second, the utility $u(q, p, a)$ is not directly available from distant user behavior signal (e.g. view-then-purchase) because users will not explicitly express which attributes are important to them to compare the items. Meanwhile, attribute frequency is also not a good indicator for $u(q, p, a)$ due to the common data issue of large amount of missing attribute values.

To this end, we propose a learning based multi-step framework called `Extract-Expect-Explain` ($\texttt{EX}^3$). As illustrated in Fig. 1, the first `Extract` step aims to reduce the search space of candidate items by learning item embeddings with distant supervision and approximating coarse-grained item similarity. Next, the `Expect` step aims to estimate the utility function $u(q, p, a)$ by decomposing it into two parts: fine-grained item relevance and attribute importance. The last `Explain` step leverages the outputs from two previous steps to solve the optimization problem and derive the K explainable groups for item set recommendations.

3.1 Extract-Step

In this step, we aim to learn an item encoder $\phi : \mathcal{P} \mapsto \mathbb{R}^{d_p}$ that maps each item in $\mathcal{P}$ to d_p-dimensional space such that the items with relationships in $\mathcal{B}$ are closer in the latent space. The latent item vectors generated by ϕ can be subsequently used as pretrained item embeddings in downstream steps and extracting coarse-grained similar candidates with respect to pivot items.

Specifically, each item $p \in \mathcal{P}$ is initialized with either a one-hot vector or a raw feature vector extracted from metadata such as item title and category. Then, it is fed to the item encoder ϕ, which is modeled as a multilayer perceptron (MLP) with non-linear activation function. In order to capture relatedness among items, we assume that each item is similar to its related items in $\mathcal{B}$ and is distinguishable from other unrelated items. As illustrated in Fig. 1(a), let $N_p = \{p_i | (p, p_i) \in \mathcal{B}\}$ be the related items for an item $p \in \mathcal{P}$. We define a metric function $f(p, N_p)$ to measure the distance between the item and its related items:

$$f(p, N_p) = \lambda - \|\phi(p) - h(N_p)\|_2^2, \tag{2}$$

where λ is the base distance to distinguish p and N_p, and $h(\cdot)$ denotes an aggregation function over item set N_p, which encodes N_p into the same d_p-dimensional space as $\phi(p)$. In this work, we define $h(\cdot)$ to be a weighted sum over item embeddings via dot-product attention:

$$h(N_p) = \sum_{p_i \in N_p} \alpha_i \phi(p_i), \quad \alpha_i = \frac{\exp\left(\phi(p)^\intercal \phi(p_i)\right)}{\sum_{p_j \in N_p} \exp\left(\phi(p)^\intercal \phi(p_j)\right)} \tag{3}$$

We assign a positive label $y^+ = 1$ for each pair of (p, N_p). For non-trivial learning to distinguish item relatedness, for each item p, we also randomly sample $|N_p|$ items from $\mathcal{B}_{\text{pv}}$ as negative samples denoted by N_p^- with assigned label $y^- = -1$. Therefore, the encoder ϕ can be trained by minimizing a hinge loss with the following objective function:

$$\ell_{\text{extract}} = \sum_{p \in \mathcal{P}} \max(0, \epsilon - y^+ f(p, N_p)) + \max(0, \epsilon - y^- f(p, N_p^-)), \tag{4}$$

where ϵ is the margin distance.

Once the item encoder ϕ is trained, for each pivot item $q \in \mathcal{P}$, we can retrieve a set of m $(|N_p| \ll m \ll |\mathcal{P}|)$ coarse-grained related items as its candidate set C_q, i.e., $C_q = \{p_i | \text{rank}\,(f(q, \{q\})) = i, p_i \in \mathcal{P} \setminus \{q\}, i \in [m]\}$.

3.2 Expect-Step

The goal of this step is to learn the utility function $u(q, p, a)$ to estimate how likely a candidate item p will be clicked or purchased by users after being compared with pivot item q on attribute a. For simplicity of modeling, we assume that the utility function can be decomposed into two parts:

$$u(q, p, a) = g(\underbrace{u_{\text{rel}}(q, p)}_{\text{Item relevance}}, \underbrace{u_{\text{att}}(a|q, p)}_{\text{Attribute importance}}), \tag{5}$$

where $g : [0, 1] \times [0, 1] \mapsto [0, 1]$ is a binary operation. The first term $u_{\text{rel}}(q, p)$ reveals the fine-grained item relevance, or equivalently, the likelihood of item p being clicked by users after compared with pivot q (no matter which attributes are considered). The second term $u_{\text{att}}(a|q, p)$ indicates the importance of displaying attribute a to users when they compare items q and p. It is natural to learn these two functions if well-curated datasets are available. However, practically, even though the item relevance can be simulated from distant user behavior signals, e.g., $\mathcal{B}_{\text{pv}}$ view-then-purchased, the groundtruth of important attributes still remain unknown. This is because users will not explicitly express the usefulness of item attributes when they do online shopping, which leads to the challenge of how to infer the attribute importance without supervision. In addition, the data issue of missing attributes and noisy values is quite common since it costs much time and effort to manually align all the attributes of items. That is to say each item may contain arbitrary number of attributes and their values may contain arbitrary content and data types.

To overcome the issues, we propose a novel neural model named *Attribute Differentiating Network* (ADN) to jointly approximate u_{rel} and u_{att}. Formally, it takes as input a pivot item q and a candidate item p along with the corresponding n

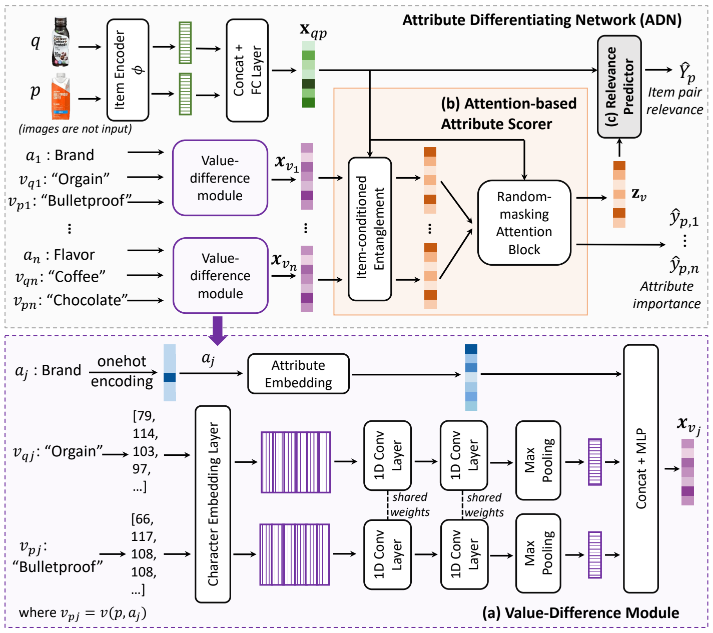


Fig. 2. Network architecture of the proposed Attribute Differentiating Network (ADN) including (a) a value-difference module, (b) an attention-based attribute scorer, and (c) a relevance predictor.

attribute-value pairs A_q, A_p (e.g., $A_q = \{(a_1, v(q, a_1)), \ldots, (a_n, v(q, a_n))\}$), and simultaneously outputs an item relevance score $\hat{Y}_p \in [0, 1]$ and attribute importance scores $\hat{y}_{p,j} \in [0, 1]$ for attribute a_j $(j = 1, \ldots, n)$.

Network Overview. As illustrated in Fig. 2, ADN consists of three components: a *value-difference module* to capture the difference levels of attribute values of two items, an *attention-based attribute scorer* to implicitly predict the attribute contribution, and a *relevance predictor* that estimates the fine-grained relevance of two items. Specifically, two input items are first respectively vectorized by the encoder ϕ from the `Extract` step. The derived item embeddings are then mapped into low-dimensional space via linear transformation, i.e. $\mathbf{x}_{qp} = W_p[\phi(q); \phi(p)]$, where $[;]$ denotes the concatenation and W_p is the learnable parameters. Then, each attribute-value tuple $(a_j, v(q, a_j), v(p, a_j))$ is encoded by the value-difference module into a vector denoted by $\mathbf{x}_{v_j}$. All these vectors $\mathbf{x}_{v_1}, \ldots, \mathbf{x}_{v_n}$ together with $\mathbf{x}_{qp}$ will be further fed to the attention-based attribute scorer to produce attribute importance scores $\hat{y}_{p,1}, \ldots, \hat{y}_{p,n}$ as well as an aggregated vector $\mathbf{z}_v$ about value-difference information on all attributes. The relevance predictor finally yields $\hat{Y}_p$ based on the joint of $\mathbf{x}_{qp}$ and $\mathbf{z}_v$.

Value-Difference Module. As shown in Fig. 2(b), we represent each attribute a_j as a one-hot vector and then embed it into d_a-dimensional space via linear transformation, i.e., $\mathbf{a}_j = W_a a_j$, with learnable parameters W_a. Since the value $v(p, a_j)$ of item p and attribute a_j can be of arbitrary type, inspired by character-level CNN, we treat it as a sequence of characters and each character is embedded into a d_c-dimensional vector via linear transformation with parameters W_c. Suppose the length of character sequence is at most n_c. We can represent the value $v(p, a_j)$ as a matrix $\mathbf{v}_{pj} \in \mathbb{R}^{n_c \times d_c}$. Then, we adopt convolutional layers to encode the character sequence as follows:

$$\mathbf{x}_{v_j} = \text{maxpool}(\text{ReLU}(\text{conv}(\text{ReLU}(\text{conv}(\mathbf{v}_{pj}))))) \tag{6}$$

where conv$(\cdot)$ denotes the 1D convolution layer and maxpool$(\cdot)$ is the 1D max pooling layer. The output $\mathbf{x}_{ij} \in \mathbb{R}^{d_c}$ is the latent representation of arbitrary value v_{ij}. To capture value difference on attribute a_j between items q, p, we further encode the attribute vector $\mathbf{a}_j$ and the value vectors $\mathbf{x}_{qj}$ and $\mathbf{x}_{pj}$ via an MLP:

$$\mathbf{x}_{v_j} = \mathrm{MLP}_v([\mathbf{a}_j; \mathbf{x}_{qj}; \mathbf{x}_{ij}]), \tag{7}$$

where $\mathbf{x}_{v_j}$ is supposed to encode the value-difference information between values $v(q, a_j)$ and $v(p, a_j)$ on attribute a_j.

Attention-based Attribute Scorer. Since our goal is to detect important attributes with respect to the pair of items, we further entangle each value-difference vector $\mathbf{x}_{v_j}$ of attribute a_j conditioned on item vector $\mathbf{x}_{qp}$ as follows:

$$\mathbf{w}_{pj} = \mathrm{MLP}_p([\mathbf{x}_{qp}; \mathbf{x}_{v_j}; \mathbf{x}_{qp} \odot x_{v_j}; \|\mathbf{x}_{qp} - \mathbf{x}_{v_j}\|]), \tag{8}$$

where another MLP_p is employed to generate the item-conditioned value-difference vector $\mathbf{w}_{pj}$.

Natually, we can use attention mechanism to aggregate n item-conditioned attribute vectors $\mathbf{w}_{p1}, \ldots, \mathbf{w}_{pn}$ for better representation and automatic detection of important attributes. However, directly applying existing attention mechanism here will encounter several issues. First, the learned attention weights may have bias on frequent attributes. That is higher weights may not necessarily indicate attribute importance, but only because they are easily to acquire and hence occur frequently in datasets. Second, attribute cardinality varies from items to items due to the issue of missing attribute values, so model performance is not supposed to only rely on a single attribute, i.e. distributing large weight on one attribute. To this end, we propose the Random-masking Attention Block (RAB) to alleviate the issues. Specifically, given item vector $\mathbf{x}_{qp}$ and n item-conditioned value-difference vectors $\mathbf{w}_{p1}, \ldots, \mathbf{w}_{pn}$, the RAB block is defined as follows.

$$Q = W_Q \mathbf{x}_{qp}, K_j = W_K \mathbf{w}_{pj}, V_j = W_V \mathbf{w}_{pj}, j \in [n] \tag{9}$$

$$\hat{y}_{p,j} = \frac{\exp\left(\frac{Q^\intercal K_j}{\sqrt{d}\tau_j}\right) \cdot \eta_j}{\sum_{i \in [n]} \exp\left(\frac{Q^\intercal K_i}{\sqrt{d}\tau_i}\right) \cdot \eta_i} \tag{10}$$

$$\mathbf{z}_v = \ln(\mathrm{MLP}_o(o) + o), \; o = \ln(Q + \sum_j \hat{y}_{p,j} V_j), \tag{11}$$

where η_j is a random mask that has value γ with probability $freq_j$ (frequency of attribute a_j) in training and value 1 otherwise. It is used to alleviate the influence by imbalanced attribute frequencies. τ_j is known as the temperature in softmax and is set as $(1 + freq_j)$ by default, which is used to shrink the attention on the attribute assigned with large weight. The RAB block can be regarded as a variant of the scaled dot-product attention by incorporating randomness of attribute frequencies and item-conditioned information. The attention weights $\{\hat{y}_{p,j}\}_{j \in [n]}$ are used to approximate attribute importance $u_{\text{att}}(a_j|q, p)$. The output $\mathbf{z}_v$ encodes the aggregated information contributed by all attributes.

Relevance Predictor. We adopt a linear classifier model to predict the relevance of two items based on the item vector as well as encoded attribute-value vector:

$$\hat{Y}_p = \sigma(W_y[\mathbf{x}_{qp}; \mathbf{z}_v]) \tag{12}$$

We treat the problem as a binary classification with the objective function defined as follows:

$$\ell_{\text{expect}} = -\sum_{(q,p,Y)} Y \log \hat{Y}_p - (1 - Y) \log(1 - \hat{Y}_p). \tag{13}$$

Note that pairwise ranking loss can also easily be extended here and the choice of a better ranking loss function is beyond the scope of this paper.

Algorithm 1 Explain-step Inference Algorithm

```
1:  Input: pivot item $q$, all items $\mathcal{P}$, all attributes $\mathcal{A}$, upperbounds $D_{\text{grp}}, D_{\text{div}}$
2:  Output: $K$ groups $G_{a_{(1)}}, \ldots, G_{a_{(K)}}$
3:  procedure MAIN()
4:      Get candidate set $C_q$ in the Extract-step.
5:      for $p_i \in C_q$ do
6:          Make forward pass of ADN to obtain $\hat{Y}_{p_i}, \{\hat{y}_{p_i,j}\}$.
7:          Compute $u_{ij} = \hat{Y}_{p_i} \cdot \hat{y}_{p_i,j}$, for $j = 1, \ldots, |\mathcal{A}|$.
8:      Solve optimization in Eq. 1 and obtain $X$.
9:      Initialize $|\mathcal{A}|$ priority queues, $G_1, \ldots, G_{|\mathcal{A}|}$.
10:     for $a_j \in \mathcal{A}$ do
11:         for $p_i \in C_p$ do
12:             if $X_{ij} = 1$ then Insert $(p_i, u_{ij})$ into queue $G_j$ ordered by $u_{ij}$ in descending order.
13:         Compute $s_j = \sum_{(p_i,u_{ij}) \in G_j} u_{ij} / |G_j|$.
14:     Get top $K$ groups $G_{(1)}, \ldots, G_{(K)}$ s.t. $s_{(1)} \geq \cdots \geq s_{(K)}$.
15:     return $G_{(1)}, \ldots, G_{(K)}$
```

Once the model is trained, we can obtain the relevance score $u_{\text{rel}}(q,p) \approx \hat{Y}_p$ that implies whether candidate item p is relevant to query item q, and the attribute importance score $u_{\text{att}}(a_j|q,p) \approx \hat{y}_{p,j}$ $(j = 1, \ldots, n)$ indicating how important each attribute a_j is to users when they compare items q and p. We adopt a simple binary operation $g(u_{\text{rel}}(q,p), u_{\text{att}}(a_j|q,p)) \approx \hat{Y}_p \cdot \hat{y}_{p,j}$ to estimate the utility value $u(q,p,a_j)$.

3.3 Explain-Step

In this step, the goal is to present K explainable groups $G_{a_{(1)}}, \ldots, G_{a_{(K)}}$ such that the whole utility is maximized. The complete inference algorithm is described in Alg. 1. Specifically, it first extracts a small subset of similar candidate items C_q with respect to the pivot item q. For each pair of q and $p_i \in C_q$, it computes the relevance score of two items as well as the importance scores of attributes. Then, the LP problem defined in Eq. 1 is solved to obtain the assignments of candidate items on attribute-based groups. For each group, the algorithm takes the score from the most significant item as the heuristic score for group-level ranking. Finally, the top K groups are generated as the recommendation with attribute-based explanations. Note that we adopt template-based generation approach to generate the natural language explanation based on attributes, which is not the focus in this paper.

3.4 Implementation Detail

In the Extract-step, the raw features of each item consist in n-gram features extracted from items' titles, key words and categories. The feature extractor ϕ consists of 3 fully-connected layers of sizes 1024, 1024, 128 with ReLU [24] as nonlinear activation function. Margin parameters $\lambda = 1.0$ and $\epsilon = 1.0$. The model is trained with Adam optimizer with learning rate 0.001 and batch size 128.

In the Expect-step, the network parameters are as follows. $W_p \in \mathbb{R}^{256\times 64}$, $W_a \in \mathbb{R}^{|\mathcal{A}|\times 64}$. We restrict maximum character sequence length to $n_c = 200$ and the value of characters ranges from $0 - 255$. The character embedding size $d_c = 64$ with $W_c = \mathbb{R}^{255\times 64}$. Each convolution layer contains 64 filters is 64 and kernels of size 3. The multilayer perceptron MLP_v consists of two fully-connected layers of sizes 172, 64 with ReLU as activation. The MLP_p has two fully-connected layers of sizes 256, 64. In attention, $W_q, W_k, W_v \in \mathbb{R}^{64\times 64}$ and MLP_o contains two fully-connected layers of sizes 64, 64. Masking value η is set to 0.3 and the attribute frequency freq_j is estimated from the training set. The

Dataset	**Overall**	**Battery**	**Coffee**	**I. Protector**	**Laundry**	**Shampoo**	**T. Paper**	**Vitamin**
#Items	286K	114K	61K	29K	11K	35K	6K	30K
#Attributes	19	19	19	19	19	19	19	19
$\mathcal{B}_{\text{cv}}$ pairs	16.9M	2.9M	3.3M	1.7M	4.1M	1.4M	2.3M	1.2M
$\mathcal{B}_{\text{cp}}$ pairs	3.1M	490K	1M	557K	130K	412K	45K	489K
$\mathcal{B}_{\text{pv}}$ pairs	709K	203K	205K	88K	31K	98K	12K	80K

Table 1. Dataset statistics across 7 subdomains.

linear predictor layer $W_y \in \mathbb{R}^{128\times 1}$. The Expect model is trained with Adam optimizer with learning rate of 5×10^{-4}, weight decay 10^{-5} in total 20 epochs. All deep neural model are implemented in PyTorch and deployed based on Spark.

In the Explain-step, we set both D_{grp} and D_{att} to be 5 by default. Candidate set size $|C_q| = 30$, $|\mathcal{A}| = 19$ and $K = 5$. The LP problem is solved by PuLP library[2].

4 EXPERIMENTS

In this section, we comprehensively evaluate the performance of the proposed method EX^3 in terms of both recommendation and attribute ranking on a real-world benchmark.

4.1 Experimental Setup

Dataset. We take experiments on a real-world industrial dataset collected from Amazon.com including 7 subcategories: Battery, Coffee, Incontinence Protector, Laundry Detergent, Shampoo, Toilet Paper and Vitamin. Following distant supervision manner mentioned in Section 2, each subset can be regarded as an individual benchmark. To enable fast experiments, we randomly sample products from each product category and select their corresponding attributes to construct the datasets. The statistics of these datasets are summarized in Table 1. Similar metadata can also be found in [20, 21]. Due to the large-scale product pool, we generate candidate products for each query product via the proposed Extract-Step, which leads to around 30 similar candidate products per query. Our model and all the baselines are trained and evaluated based on the extracted candidates. We randomly split the dataset into training set (80%), validation set (10%) and test set (10%).

Baselines & Metrics. We compare our method with following baselines.

- Relevance is the method that computes item similarity based on item embeddings learned in Extract step.
- BPR [26] is the Bayesian personalized ranking method for making recommendations, which is modified to item-to-item prediction in this work.
- ACCM [27] is a CF-based and CB-based recommendation approach that leverages attribute to enrich the representation of items. We adapt this method to our item-to-item recommendation.
- A2CF [3] is the state-of-the-art attribute-based recommendation model that outputs substitutes for pivot items.
- EX^3 is our approach proposed in Expect step.

For fair comparison, we generate the a candidate set of 30 items for each pivot from the Extract step. All the baselines are evaluated based on the candidate set and also leverage the pretrained item embeddings as input if necessary.

We adopt NDCG@10, Recall@10, Precision@10 as the metrics to evaluate the top-N recommendation performance.

4.2 Top-N Recommendation Performance (Expect-Step)

In this experiment, we first evaluate the recommendation performance output by the Expect step, which produces the same results as traditional recommendations. Specifically, given a pivot item, both our method and all other baselines

[2]https://pypi.org/project/PuLP/

	Overall			**Battery**			**Coffee**			**Incontinence Protector**		
Measures (%)	NDCG	Recall	Prec.	NDCG	Recall	Prec.	NDCG	Recall	Prec.	NDCG	Recall	Prec.
Relevance	0.4489	0.6875	0.1310	0.3252	0.5916	0.1062	0.4656	0.6724	0.1475	0.3887	0.6613	0.1052
BPR	0.6373	0.8409	0.1695	0.5536	0.7609	0.1410	0.7246	0.8816	0.2041	0.5794	0.8353	0.1372
ACCM	0.6969	0.9029	0.1817	0.5849	0.8162	0.1507	0.7532	0.9305	0.2151	0.7333	0.9425	0.1568
A2CF	0.7207	0.9184	0.1854	0.6209	0.8589	0.1580	0.7898	0.9451	0.2194	0.7482	0.9483	0.1577
EX^3	**0.8177**	**0.9667**	**0.1953**	**0.7304**	**0.9245**	**0.1700**	**0.8716**	**0.9786**	**0.2278**	**0.8660**	**0.9783**	**0.1635**
Improve	11.35%	5.26%	5.34%	17.64%	7.64%	7.59%	10.36%	3.54%	3.83%	15.74%	3.16%	3.68%
	Laundry Detergent			**Shampoo**			**Toilet Paper**			**Vitamin**		
Measures (%)	NDCG	Recall	Prec.	NDCG	Recall	Prec.	NDCG	Recall	Prec.	NDCG	Recall	Prec.
Relevance	0.3650	0.5958	0.0987	0.4771	0.7689	0.1176	0.2591	0.3893	0.0695	0.4379	0.7116	0.1247
BPR	0.4882	0.7066	0.1246	0.5440	0.8230	0.1276	0.5016	0.6793	0.1246	0.5608	0.8001	0.1450
ACCM	0.5253	0.7399	0.1329	0.6753	0.9276	0.1447	0.4619	0.6337	0.1158	0.6061	0.8375	0.1535
A2CF	0.5368	0.7595	0.1371	0.6737	0.9329	0.1456	0.4966	0.6813	0.1251	0.6307	0.8715	0.1606
EX^3	**0.7351**	**0.9158**	**0.1658**	**0.7609**	**0.9703**	**0.1517**	**0.7750**	**0.9135**	**0.1715**	**0.7392**	**0.9405**	**0.1734**
Improve (%)	36.94%	20.58%	20.93%	12.94%	4.01%	4.19%	54.50%	34.08%	37.09%	17.20%	7.92%	7.80%

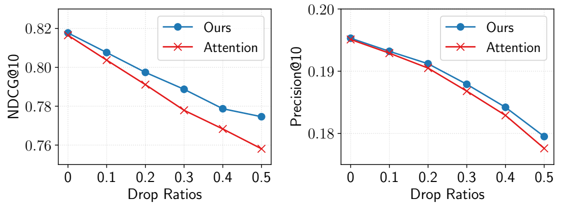
Table 2. EX^3 wins. Top-10 recommendation performance of our model and baselines on the dataset across 7 subdomains.

outputs top 10 recommendations from 30 candidates generated by Extract step. The goal of this experiment is to verify if our model can output more relevant items than others.

The results are reported in Table 2. We observe that our model EX^3 consistently outperforms all baselines across all datasets on all metrics. For instance, our model achieves NDCG of 0.8177, Recall of 0.9667 and Precision of 0.1953, which are higher than the results produced by the best baseline A2CF by a large margin. It is interesting to see that our model shows significant improvements on the item ranking performance, resulting at least 11.35% improvement in NDCG in Overall dataset and 10.36%–56.06% improvements across 7 subdomains. In addition, we notice that for datasets Coffee and Incontinence Protector, the recommendation performance of all models are better than the overall (average) performance. For example, our model achieves NDCG of 0.8716 and 0.8660 respectively, which are higher than Overall NDCG of 0.8177. Other models share similar trends. This indicates that the cases in these two datasets are easier to learn to capture user behavior.

4.3 Model Robustness to Missing Attributes

We further show our model is robust to missing attributes in inference data with the proposed masking attention. Specifically, we randomly drop 10%, 20%, 30%, 40% and 50% attributes in the test set and evaluate the top-N recommendation performance of our model with and without the proposed attention mechanism. All other settings remain the same. As shown in Fig. 3, our model w/ the technique (red curve) is consistently better than the baseline (blue curve) under different attribute dropping ratios in both NDCG and precision. In addition, we notice that the performance decrease of our model is slower than that of baseline, as the slope of the curve is smaller. This results imply that the proposed model is robust to the missing attributes during inference, which is essential in real-world scenarios.



(a) NDCG@10 (b) Precision@10

Fig. 3. Results of Top-N recommendation under different degrees of missing attributes on Overall dataset.

	Random		Greedy		EX^3	
Measures	Avg.	Norm.	Avg.	Norm.	Avg.	Norm.
$D_{\text{div}} = 1, D_{\text{grp}} = 2$	1.158	2.879	2.050	4.929	**2.272**	**5.380**
$D_{\text{div}} = 1, D_{\text{grp}} = 3$	1.180	2.836	2.051	4.890	**2.277**	**5.400**
$D_{\text{div}} = 1, D_{\text{grp}} = 5$	1.173	2.852	2.048	4.884	**2.271**	**5.347**
$D_{\text{div}} = 2, D_{\text{grp}} = 2$	1.164	2.792	2.068	4.797	**2.273**	**5.408**
$D_{\text{div}} = 2, D_{\text{grp}} = 3$	1.158	2.951	2.035	4.883	**2.279**	**5.439**
$D_{\text{div}} = 2, D_{\text{grp}} = 5$	1.156	2.934	2.046	4.895	**2.277**	**5.400**
$D_{\text{div}} = 3, D_{\text{grp}} = 2$	1.146	2.867	2.031	4.850	**2.279**	**5.424**
$D_{\text{div}} = 3, D_{\text{grp}} = 3$	1.162	2.800	2.070	4.810	**2.273**	**5.435**
$D_{\text{div}} = 3, D_{\text{grp}} = 5$	1.157	2.794	2.065	4.808	**2.273**	**5.427**

Table 3. Results of attribute ranking performance on Overall dataset. Avg. is average score and Norm. is normalized score.

Attr./Items	B000YG1INI	B082FPF9HZ	B0153VTN9E	B00FU5BY2S
scent	peppermint	peppermint	Tea Tree	eucalyptus
brand	Desert Essence	Natural V.I.P	HONEYDEW	Trader Joe's
special ingredient	tea-tree-oil	–	tea-tree-oil	tea-tree-oil
hair type	all types	Dry	Dry	All types
target gender	unisex	–	unisex	unisex
Attr./Items	**B000YG1INI**	**B01KPUTIM0**	**B00N648M66**	**B001B3RFK8**
scent	peppermint	lemon	coconut	Lemon Tea
hair type	All types	Color treated	dry, frizzy	Oily
special ingredient	tea-tree-oil	–	jojoba-oil	–
brand	Desert Essence	Desert Essence	Desert Essence	Desert Essence
target gender	unisex	–	unisex	unisex

Table 4. Example of adaptive attribute ranking under different candidate items given same pivot item "B000YG1INI".

4.4 Attribute Ranking Performance (Explain-Step)

Effectiveness of Attribute Ranking. In this experiment, we evaluate the performance of the proposed Explain-Step in identifying important attributes. We specifically consider following three baselines.

- `Random` is a simple grouping algorithm by randomly assigning items into attribute-based groups as long as the corresponding value exists. Then the groups are ordered in the way same as Alg. 1 (line 13–14).
- `Greedy` is an iterative algorithm by always picking the pair of item and attribute with larger utility value
- `EX`^{3} is our proposed method of the Explain-Step.

Note that all compared methods differ in the grouping ways but take the same utility function as input, which is generated by the Expect-step for fair comparison. To quantify the attribute ranking performance, we randomly sample around 1000 cases and ask human evaluators to manually score each attribute in a 5-point scale given a pivot item and a set of candidate items. Then we can calculate the average and the normalized score of the predicted important attributes by each model. The results are reported in Table 3.

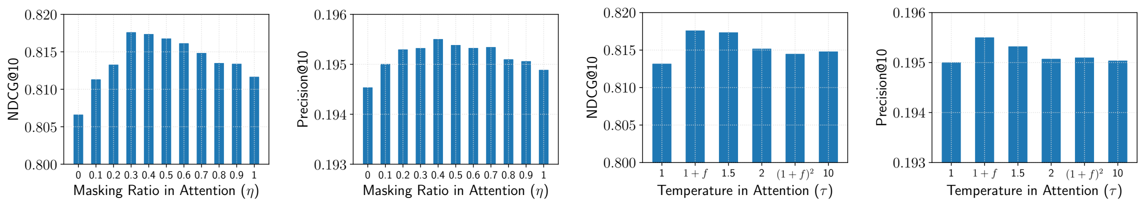
We observe that our method `EX`^{3} gives the better performance in important attribute ranking compared with two baselines. One interesting fact is that the Greedy algorithm is actually an approximation algorithm for the optimization problem Eq. 1, which interprets that its performance is slightly worse than ours.

Adaptive attribute ranking. In addition, we show that for the same pivot item, our model will rank attributes differently if the candidates are different. We showcase an example in Table 4 to demonstrate this characteristics of our model. Given a shampoo product with ID "B000YG1INI" as pivot item [3], whose attributes are listed in the second column, we feed two sets of candidate items to our model that is able to generate two different attribute rankings as shown in the upper and lower parts of the table. It is interesting to see that the model is able to rank attributes based on value differences and diversity. Take "brand" attribute as example. In the first case (upper table), "brand" is ranked in the second place and considered as a relatively important attributes when users compare different shampoo products. In contrast, in the second case (lower table), "brand" is ranked lower because all the candidates have the same brand "Desert Essence" and it is less informative for users to enhance their shopping experience.

4.5 Ablation Study

We first show the recommendation performance under different masking ratios (η) in the proposed attention mechanism. Specifically, we adopt different values of η to train the model of Expect step, e.g. $\eta = 0, 0.1, \ldots, 0.9, 1.0$. Note that $\eta = 0$

[3]Product detail can be found in https://www.amazon.com/dp/B000YG1INI



(a) NDCG@10 (b) Precision@10 (c) NDCG@10 (d) Precision@10

Fig. 4. Top-N Recommendation performance on the overall dataset under different masking ratios (η) in attention (Fig. (a), (b)), and under different temperatures (τ) in attention (Fig. (c), (d)).

means the attribute is completely dropped while $\eta = 1$ means there is no attribute dropping. We report the top-N recommendation performance under various η's in Fig. 4 (a, b). We observe that the masking ratios influence on ranking performance (NDCG) and the model achieves the best performance when $\eta = 0.3$. For precision, we find that the performance does not vary a lot, but still show similar trends as the NDCG, i.e. $\eta = 0.4$ leads to the relatively better performance.

Next, we evaluate the influence of different values of temperatures (τ) in the attention mechanism. Specifically, we experiment with two ways of imposing temperatures over softmax function. The first one relies on the predefined attribute frequencies, i.e. $\tau = (1 + freq_i)^n$ with $n = 1, 2$. The other one uses the fixed value of $\tau = 1, 1.5, 2, 10$. All other training settings remain the same. The results of top N recommendation are reported in Fig. 4 (c, d). We can see that the default choice of $\tau = 1 + freq_i$ leads to the best performance in both NDCG and precision. Besides, note that when $\tau = 1$, it is equivalent to the original softmax function. Our model with the default τ shows superior performance over such setup, which indicates the effectiveness of the proposed attention mechanism.

4.6 Online Simulation and Experiments

In this experiment, we evaluate the overall performance of the group-form explainable item set recommendation. Before serving the proposed method to real users, we generate a batch of explainable item set recommendations in an offline mode and leverage human annotators to help us evaluate the recommendation quality. For each of 7 product categories, we sample top 50 most popular pivot products from our recommendation dataset and ask the annotators to evaluate whether the attribute-based explainable recommendations can help users make better purchase decision. Note that the evaluation metric contains two-fold interactive measurement on both product relevance and attribute importance, as the ranked important attribute list should depend on what products are recommended to users. Through human evaluation, we obtain over 80% acceptance rate on high-quality item set recommendations with over 86% accuracy on comparable product recommendation performance, which is 2x higher than using raw $\mathcal{B}_{\text{pv}}$ data for recommendation. We also conduct online A/B testing through real user traffic on a large-scale e-commerce website, and the results show significant increase of conversion (+0.080%) and revenue (+0.105%) in online A/B experiments.

5 RELATED WORK

In this section, we discuss the related work regarding explainable recommendation and item relationship mining.

Explainable Recommendation. In the era of e-commerce, recommender systems have been widely used to provide users with relevant item suggestions. Most of existing methods are based on collaborative filtering [16], matrix factorization [17] and neural recommendation model [34]. Recently, to further improve user experience of recommender systems [18], great research efforts have been promoted to explainable recommendation problems [6, 7, 35]. One common way to generate explanation for recommendation is to leverage knowledge graphs [5, 15, 31, 33, 39]. For example, Xian et al. [32] propose to leverage reinforcement learning on knowledge graph to provide behavior-based explanation for product recommendations, while Zhao et al. [37] also employs reinforcement learning but propose a different demonstration-based knowledge graph reasoning framework for explainable recommendation. Besides knowledge graph, sentiment/opinion based explainable recommendation is also a popular research topic [2, 8]. Zhang et al. [36] integrate sentiment analysis into factorization model to improve explainable recommendation performance. Wang et al. [29] develop a multi-task learning solution for explainable recommendation, where two learning tasks on user preference modeling for recommendation and opinionated content modeling for explanation are joint learning via a shared tensor factorization framework. There are also research work around attribute-based explainable recommendation. Hou et al. [14] extract visual attributes from product images to conduct explainable fashion recommendation. Chen et al. [3] propose to leverage both user and item attributes to generate interpretable recommendations. Most of existing work focuses on explainable user-item recommendation problems but lack of the discussion on explainable item-to-item-set recommendation tasks, which is also important for e-commerce platforms. Moreover, explainable item-to-item-set recommendation problem is a harder case in explainable recommendation. Unlike explainable user-item recommendation problem where users and items do not always share same properties and thus allow more tolerance on generating explanations, in explainable item-to-item-set scenario, (1) we need to explicitly and rigorously provide reasonable attribute-based explanations between items since they always share same properties, e.g., display size for all TVs, and (2) the item set recommendations should balance both relevance and diversity on multiple item attributes.

Item Relationship Mining. As our work is around item-to-item-set recommendation, we will also discuss existing work on item relationship mining. Identifying relationships among items is a fundamental component of many real-world recommender systems [20, 30]. Linden et al. [19] designs an item-to-item collaborative filtering to generate similar item recommendation for Amazon.com. Zhang et al. [38] discuss the impact of substitute and complement relationship between items on recommendations. Similar efforts [1, 12] have been put to target at explicitly modeling relationship between items for recommendations. Representative examples include Sceptre [20], which proposes a topic modeling method to infer networks of products, and PMSC [30], which incorporates path constraints in item pairwise relational modeling. He et al. [13] design a framework to use visual features to identify compatibility relationship between clothes and jewelry. These methods seek to distinguish substitutes, complements and compatibilities, but fail to provide any clear explanation on why these items are substitutable and comparable.

6 CONCLUSION

In this work, we study the important problem of explainable attribute-aware item-set recommendation. We propose a multi-step learning-based framework called Extract-Expect-Explain ($\mathtt{EX}^3$) to approach the problem by first extracting coarse-grained candidate sets of items with respect to the pivot to reduce the search space of similar items (Extract-step), followed by a joint prediction of pairwise item relevance and attribute importance (Expect-step), which are subsequently fed to a constrained optimization solver to generate the group-form recommendations with explanations (Explain-step). The experiments are conducted on a real-world large-scale dataset and the results demonstrate that our proposed model achieves over 10% higher NDCG than state-of-the-art baselines in the explainable recommendation domain. Moreover, our proposed method can adaptively generate attribute-based explanations for various products, and the resulting explainable item-set recommendations are also shown to be effective in large-scale online experiments. There are several promising areas that we consider for future work, such as leveraging the learnt important attributes for query rewriting and product categorization.

REFERENCES

[1] Arijit Biswas, Mukul Bhutani, and Subhajit Sanyal. 2017. Mrnet-product2vec: A multi-task recurrent neural network for product embeddings. In *ECML PKDD*.

[2] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate natural language explanations for recommendation. *EARS*.

[3] Tong Chen, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang, and Meng Wang. 2020. Try This Instead: Personalized and Interpretable Substitute Recommendation. *SIGIR*.

[4] Wen-Hao Chen, Chin-Chi Hsu, Yi-An Lai, Vincent Liu, Mi-Yen Yeh, and Shou-De Lin. 2018. Attribute-aware collaborative filtering: survey and classification. *arXiv preprint arXiv:1810.08765* (2018).

[5] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, and Gerard de Melo. 2020. Fairness-Aware Explainable Recommendation over Knowledge Graphs. In *SIGIR*.

[6] Zuohui Fu, Yikun Xian, Shijie Geng, Yingqiang Ge, Yuting Wang, Xin Dong, Guang Wang, and Gerard de Melo. 2020. ABSent: Cross-Lingual Sentence Representation Mapping with Bidirectional GANs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7756–7763.

[7] Zuohui Fu, Yikun Xian, Yaxin Zhu, Shuyuan Xu, Zelong Li, Gerard de Melo, and Yongfeng Zhang. 2021. HOOPS: Human-in-the-Loop Graph Reasoning for Conversational Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2415–2421.

[8] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable Recommendation Through Attentive Multi-View Learning. AAAI.

[9] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards Long-term Fairness in Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 445–453.

[10] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q Zhu. 2019. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 617–626.

[11] Junheng Hao Hao, Tong Zhao, Jin Li, Xin Luna Dong Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2020. P-Companion: A principled framework for diversified complementary product recommendation. *CIKM* (2020).

[12] Ruining He, Charles Packer, and Julian McAuley. 2016. Learning compatibility across categories for heterogeneous item recommendation. In *ICDM*.

[13] R. He, C. Packer, and J. McAuley. 2016. Learning Compatibility Across Categories for Heterogeneous Item Recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 937–942.

[14] Min Hou, Le Wu, Enhong Chen, Zhi Li, Vincent W. Zheng, and Qi Liu. 2019. Explainable Fashion Recommendation: A Semantic Attribute Region Guided Approach. *IJCAI* (2019).

[15] Xiaowen Huang, Quan Fang, Shengsheng Qian, Jitao Sang, Yiyang Li, and Changsheng Xu. 2019. Explainable Interaction-driven User Modeling over Knowledge Graph for Sequential Recommendation. *ACM MM* (2019).

[16] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*.

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

[18] Zhenyu Liao, Yikun Xian, Xiao Yang, Qinpei Zhao, Chenxi Zhang, and Jiangfeng Li. 2018. TSCSet: A crowdsourced time-sync comment dataset for exploration of user experience improvement. In *23rd International Conference on Intelligent User Interfaces*. 641–652.

[19] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003).

[20] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *ACM SIGKDD*.

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM.

[22] Aranyak Mehta. 2013. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science* (2013).

[23] Felipe Moraes, Jie Yang, Rongting Zhang, and Vanessa Murdock. 2020. The Role of Attributes in Product Quality Comparisons. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*. 253–262.

[24] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*. 807–814.

[25] Yannis Papakonstantinou and Vasilis Vassalos. 1999. Query rewriting for semistructured data. *ACM SIGMOD Record* 28, 2 (1999), 455–466.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[27] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. *CIKM* (2018).

[28] Karen Tso and Lars Schmidt-Thieme. 2006. Attribute-aware collaborative filtering. In *From data and information analysis to knowledge engineering*.

[29] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. *SIGIR*.

[30] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *WSDM*.

[31] Yikun Xian, Zuohui Fu, Qiaoying Huang, Shan Muthukrishnan, and Yongfeng Zhang. 2020. Neural-Symbolic Reasoning over Knowledge Graph for Multi-Stage Explainable Recommendation. *AAAI DLGMA Workshop* (2020).

[32] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*.

[33] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1645–1654.

[34] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*.

[35] Yongfeng Zhang and Xu Chen. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends in Information Retrieval* (2020).

[36] 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. *SIGIR* (2014), 83–92.

[37] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *SIGIR*.

[38] Jiaqian Zheng, Xiaoyuan Wu, Junyu Niu, and Alvaro Bolivar. 2009. Substitutes or Complements: Another Step Forward in Recommendations. In *EC*.

[39] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully Explainable Recommendation via Neural Logic Reasoning. *arXiv preprint arXiv:2104.07869* (2021).

[40] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *CIKM*. 43–50.