Try This Instead: Personalized and Interpretable Substitute Recommendation

Tong Chen
The University of Queensland
tong.chen@uq.edu.au

Hongzhi Yin*
The University of Queensland
h.yin1@uq.edu.au

Guanhua Ye
The University of Queensland
g.ye@uq.net.au

Zi Huang
The University of Queensland
huang@itee.uq.edu.au

Yang Wang
Hefei University of Technology
yangwang@hfut.edu.cn

Meng Wang
Hefei University of Technology
eric.mengwang@gmail.com

ABSTRACT

As a fundamental yet significant process in personalized recommendation, candidate generation and suggestion effectively help users spot the most suitable items for them. Consequently, identifying substitutable items that are interchangeable opens up new opportunities to refine the quality of generated candidates. When a user is browsing a specific type of product (e.g., a laptop) to buy, the accurate recommendation of substitutes (e.g., better equipped laptops) can offer the user more suitable options to choose from, thus substantially increasing the chance of a successful purchase. However, existing methods merely treat this problem as mining pairwise item relationships without the consideration of users' personal preferences. Moreover, the substitutable relationships are implicitly identified through the learned latent representations of items, leading to uninterpretable recommendation results.

In this paper, we propose attribute-aware collaborative filtering (A2CF) to perform substitute recommendation by addressing issues from both personalization and interpretability perspectives. In A2CF, instead of directly modelling user-item interactions, we extract explicit and polarized item attributes from user reviews with sentiment analysis, whereafter the representations of attributes, users, and items are simultaneously learned. Then, by treating attributes as the bridge between users and items, we can thoroughly model the user-item preferences (i.e., personalization) and item-item relationships (i.e., substitution) for recommendation. In addition, A2CF is capable of generating intuitive interpretations by analyzing which attributes a user currently cares the most and comparing the recommended substitutes with her/his currently browsed items at an attribute level. The recommendation effectiveness and interpretation quality of A2CF are further demonstrated via extensive experiments on three real-life datasets.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

*Corresponding author; contributing equally with the first author.

KEYWORDS

Substitute Recommendation; Product Relationship; Interpretability; Collaborative Filtering

ACM Reference Format:
Tong Chen, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang, and Meng Wang. 2020. Try This Instead: Personalized and Interpretable Substitute Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401042

1 INTRODUCTION

On modern e-commerce platforms, it is a common practice to deploy recommender systems for retrieving items that match users' personal interests [17]. With the increasingly high heterogeneity of user-item interaction data, recommender systems are expected to understand more complex contexts other than user preferences, such as purchase sequences [11, 32], fine-grained user intents [5, 33] and social connections [2].

More recently, mining product relationships for different online shopping stages have shown its strength in further improving customer satisfaction and sales revenue. One typical line of research is to recommend complementary items that are mutually compatible (e.g., iPhone 11 and phone cases), which is also known as bundle recommendation [1, 27]. In a variety of successful attempts, the identified complements can either stimulate further purchases after a user has bought a compatible item [14, 15], or attract potential customers via bundle advertisements beforehand [4, 54]. In contrast to complementary relationships, substitutable relationships exist among items that are interchangeable and functionally similar (e.g., iPhone 11 and Samsung Galaxy S10). As a typical decision process in e-commerce [26], when a user is looking for a particular type of product to buy (e.g., a laptop), she/he tends to first acquire a set of candidate items for comparison, and then pick the most suitable one (if there is any). Correspondingly, in a user's decision-making process, recommending items that are substitutable and even superior to the one currently being browsed can expand the user's view to make a better decision and eventually increase the chance of a successful purchase [50].

However, compared with complement recommendation that has been widely studied and applied to multiple domains like fashion recommendation [14, 30], targeted advertising [18, 54] and online retail [1, 35], the problem of substitute recommendation remains

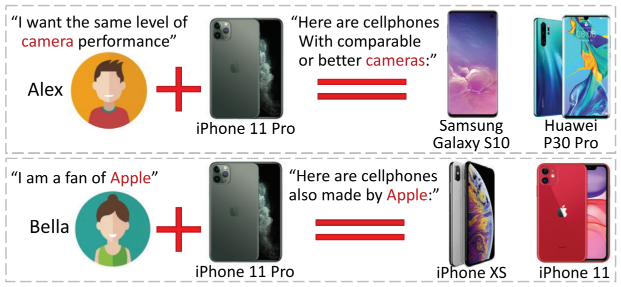

"I want the same level of camera performance"
Alex
iPhone 11 Pro
"Here are cellphones With comparable or better cameras:"
Samsung Galaxy S10
Huawei P30 Pro
"I am a fan of Apple"
Bella
iPhone 11 Pro
"Here are cellphones also made by Apple:"
iPhone XS
iPhone 11

Figure 1: The effect of personalization and interpretability in substitute recommendation.

largely unexplored. [24] is the first work to systematically investigate product relationships using reviews. In [24], items are modelled via latent Dirichlet allocation (LDA) that captures textual information from reviews, so that the substitutable relationships between any two items can be predicted by comparing their textual contents. With textual features, RRN [49] and LVA [28] are two neural network-based extensions that identify substitutable products using feed-forward networks and variational autoencoders respectively. Recent models like PMSC [43] and SPEM [50] further leverage structural constraints in observed product co-occurrence (e.g., "also viewed") graphs for discriminating substitutes. Despite the importance of generating appropriate substitutable candidates, most existing solutions treat substitute recommendation as a straightforward item-item retrieval task. That is, given an arbitrary item as the query, the recommendation model is expected to output the most relevant items as substitutes based on a pairwise scoring function [24, 49, 50]. As a result, the retrieved substitutes are purely conditioned on the query item while different users' personal preferences are neglected. Taking Figure 1 as an example, user Alex and Bella both come across with iPhone 11 Pro (i.e., the query item) when searching for a cellphone to buy. Since Alex cares much about camera quality on cellphones, the generated substitutes should be other top-tier cellphones with comparable or higher camera performance. Meanwhile, Bella simply prefers the Apple brand, so other similarly equipped iPhones are expected to be recommended. Unfortunately, without personalization, all the aforementioned methods will yield the same recommendation results for both users, because they are only determined by the similarity between the query item and substitutes. In [28], though the authors extend their proposed LVA to personalized CLVA, CLVA simply introduces user-item matrix factorization as an add-on module to LVA, which is incapable of learning users' explicit demands on fine-grained item attributes (e.g., "camera" and "brand").

Furthermore, most existing substitute recommendation models are purely built upon latent factor models like word2vec [25] and deep neural networks [19]. Though promising recommendation results are reported, a widely recognized drawback of latent factor models is that, the recommendation process is not transparent and the recommendation results are hardly interpretable to users [12, 51, 52]. Consequently, it inevitably creates a bottleneck for convincing a user why the recommended substitutes are more suitable than the query item, and how they meet this user's specific needs.

To this end, in this paper, we aim to advance substitute recommendation by addressing issues from both personalization and interpretability perspectives. On top of user-item interaction records, the growing availability of user reviews offers abundant information about their most cared attributes of a product and their sentiment towards product attributes [12, 36, 52]. As such, we propose attribute-aware collaborative filtering (A2CF), which makes full use of explicit item attributes to express user-item preferences and item-item substitutions. Instead of directly using extracted attributes as features for item representation learning [24, 28, 49], we decouple user-item interactions into user-attribute and item-attribute interactions, enabling A2CF to explicitly learn a user's attention and an item's performance across various attributes. In this way, we can not only recognize two substitutable items by comparing their attributes, but also integrate users' personal preferences to suggest more suitable substitutes. In addition, as illustrated in Figure 1, our proposed A2CF also enables personalized explanations. Specifically, we design a novel attribute-aware comparison scheme in A2CF, which infers each user's current demand on specific product attributes, and then highlights these attributes with more advantageous performance provided by the recommended substitutes.

In summary, the contributions of this paper are three-fold:

- We identify the shortcomings of existing substitute recommendation methods, and propose a new problem – personalized and interpretable substitute recommendation that aims to suggest substitutable items customized for users' preferences, along with intuitive explanations.
- We propose A2CF, a novel attribute-aware collaborative filtering model. By incorporating explicit user-attribute and item-attribute associations, A2CF simultaneously optimizes both substitution and personalization constraints for recommendation, and further interprets the recommendation results via an attribute comparison scheme.
- We conduct extensive experiments on three million-scale datasets. Comparisons with state-of-the-art methods demonstrate the superior recommendation performance of A2CF and its capability to generate high-quality interpretations.

2 PRELIMINARIES

2.1 Notations

Throughout this paper, all vectors and matrices are respectively denoted by bold lowercase and bold uppercase letters. e.g., user embedding $\mathbf{u}$, item embedding $\mathbf{v}$, attribute embedding $\mathbf{a}$, user-attribute matrix $\mathbf{X}$, and item-attribute matrix $\mathbf{Y}$. All sets are calligraphic uppercase letters, e.g., user, item, attribute sets are respectively denoted by $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{A}$. All vectors are ***column vectors*** unless specified, e.g., $\mathbf{v} \in \mathbb{R}^{d\times 1}$.

2.2 Extracting Attributes and Sentiments

Attributes are the components, features or properties of an item [12]. Based on user reviews on items, attribute-sentiment pairs can be extracted and have shown prominent contributions to review-based recommendation [36, 48, 53]. As we mainly focus on leveraging the attribute information for recommendation, we apply a well-established tool named Sentires [53] to extract attribute-sentiment pairs due to its excellent performance in a wide range of recommendation tasks [10, 12, 52]. In short, the attribute extraction will result in a sentiment lexicon $\mathcal{Z}$ for the entire review dataset and each entry is $(user, item, attribute, sentiment)$, denoted by (u, v, a, s). The sentiment polarity s is either $+1$ or -1, e.g., $(Alex, price, +1)$. To enable subsequent computations, we define the user-attribute matrix $\mathbf{X}$ and item-attribute matrix $\mathbf{Y}$ as follows.

Definition 1 (User-Attribute Matrix): The user-attribute matrix $\mathbf{X}$ is a $|\mathcal{U}| \times |\mathcal{A}|$ matrix carrying information on how each user $u_i \in \mathcal{U}$ cares about a specific attribute $a_n \in \mathcal{A}$. Intuitively, a user tends to comment on several attributes (e.g., "battery" for electronics and "taste" for food) that she/he is most interested in. So, we define each entry $x_{in} \in \mathbf{X}$ as:

$$x_{in} = \begin{cases} 0, \text{ if } u_i \text{ has never mentioned } a_n \\ 1 + (N-1)\frac{1-\exp(-t_{in})}{1+\exp(-t_{in})}, \text{ otherwise} \end{cases}, \tag{1}$$

where t_{in} is the total count of user u_i mentioning attribute a_n. N is the scaling factor such that $1 \le x_{in} \le N$. Following [36, 52], we set N as the highest score of the rating system (e.g., 5 for Amazon).

Definition 2 (Item-Attribute Matrix): Similarly, the $|\mathcal{V}|\times|\mathcal{A}|$ item-attribute matrix $\mathbf{Y}$ reflects the aggregated feedback of all users on each attribute $a_n \in \mathcal{A}$ of item $v_j \in \mathcal{V}$, where each element y_{jn} is:

$$y_{jn} = \begin{cases} 0, \text{ if } v_j \text{ has no reviews mentioning } a_n \\ 1 + \frac{(N-1)}{1+\exp(-t_{jn}\bar{s}_{jn})}, \text{ otherwise} \end{cases}, \tag{2}$$

where we also let $1 \le y_{jn} \le N$, and t_{jn} means that the attribute a_n of item v_j is mentioned n times by all users, while $\bar{s}_{jn}$ is the mean of all the sentiment scores in those t_{jn} mentions.

3 ATTRIBUTE-AWARE COLLABORATIVE FILTERING

In matrix $\mathbf{X}$, each non-zero entry x_{in} reflects user u_i's amount of attention paid to a specific attribute a_n during shopping. Similarly, for each $y_{jn} \in \mathbf{Y}$ where $y_{jn} \neq 0$, it indicates the explicit user feedback on item v_j's attribute a_n, which is crowdsourced and summarized from all user reviews. We hereby present the design of our proposed attribute-aware collaborative filtering (A2CF) model.

3.1 User-Attribute Collaborative Filtering

In traditional collaborative filtering (CF) like matrix factorization [17], given a user-item matrix with partially observed feedback, a fundamental mechanism is to learn the vectorized representations (a.k.a. embedding vectors) of both users and items using available user-item interactions, and then the unobserved values of user-item interactions can be easily estimated via the inner product of two vectors. In A2CF, with the user-attribute matrix $\mathbf{X}$, we aim to learn the representations of each user u_i and attribute a_n so that every scalar $x_{in} \in \mathbf{X}$ (i.e., the pairwise user-attribute preference) can be inferred. Specifically, we achieve this goal by minimizing the following squared loss function:

$$L_{UA} = \sum_{x_{in} \in \mathcal{D}_{UA}} (x_{in} - \widehat{x}_{in})^2, \tag{3}$$

where $\mathcal{D}_{UA} = \{x_{in} \mid x_{in} \in \mathbf{X} \wedge x_{in} \neq 0\}$ is the set of all observed user-attribute relationships, and $\widehat{x}_{in}$ is the estimated value between user u_i and attribute a_n. In what follows, we describe the computation of the interaction score $\widehat{x}_{in}$ in detail.

With the given u_i and a_n, we firstly model the nonlinear pairwise interaction between them. As pointed out by [13, 42], traditional matrix factorization-based CF methods are subject to limited expressiveness because pairwise interactions are linearly modelled via two vectors' inner product. Thus, in A2CF, we leverage a residual feed-forward network with l hidden layers for modelling pairwise user-attribute interactions:

$$\mathbf{h}_{l'} = \mathbf{h}_{l'-1} + \text{ReLU}(\mathbf{W}_{l'}\mathbf{h}_{l'-1} + \mathbf{b}_{l'}), \tag{4}$$

where $l' \le l$, ReLU is the rectified linear unit for nonlinear activation, while $\mathbf{h} \in \mathbb{R}^{d\times 1}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{b} \in \mathbb{R}^{d\times 1}$ are respectively the latent representation, weight and bias in each layer. $\mathbf{h}_0$ is initialized by concatenating both user and attribute embeddings:

$$\mathbf{h}_0 = [\mathbf{u}_i; \mathbf{a}_n], \tag{5}$$

where $\mathbf{u}_i \in \mathbb{R}^{d\times 1}$ and $\mathbf{a}_n \in \mathbb{R}^{d\times 1}$ are embeddings of user u_i and attributes a_n to be learned.

On top of the l-layer residual network, we utilize the final latent representation $\mathbf{h}_l$ to compute a scalar output $\widehat{x}_{in}$ as the estimated score for the given user-attribute pair:

$$\widehat{x}_{in} = \tanh_{[1,N]}(\mathbf{w}_x^\top \mathbf{h}_l), \tag{6}$$

where $\tanh_{[1,N]}(\cdot)$ is our modified version of $\tanh(\cdot)$ function to emit prediction results in a customized range. Specifically, for an arbitrary scalar input r, the computation process of $\tanh_{[1,N]}(r)$ is:

$$\begin{aligned} \tanh_{[1,N]}(r) &= \frac{N-1}{2} \cdot \frac{\exp(2r)-1}{\exp(2r)+1} + \frac{N+1}{2} \\ &= \frac{N\exp(2r)+1}{\exp(2r)+1}. \end{aligned} \tag{7}$$

As $\tanh_{[1,N]}(r) \approx 1$ when $r = -\infty$ and $\tanh_{[1,N]}(r) \approx N$ when $r = +\infty$, function $\tanh_{[1,N]}(\cdot)$ forces $\widehat{x}_{in}$ to fall in the rage of $[1, N]$.

So far, we can summarize the computation of $\widehat{x}_{in}$ as:

$$\widehat{x}_{in} = \tanh_{[1,N]}\left(\mathbf{w}_x^\top H_l^{UA}([\mathbf{u}_i; \mathbf{a}_n])\right), \tag{8}$$

where we use $H_l^{UA}(\cdot)$ to denote the l-layer residual feed-forward network in Eq.(4), and $\mathbf{w}_x \in \mathbb{R}^{d\times 1}$ is the projection weight. With $\widehat{x}_{in}$ computed, we are now able to optimize the learned user and attribute representations by minimizing the loss in Eq.(3).

3.2 Item-Attribute Collaborative Filtering

Similar to $\mathbf{X}$, given the item-attribute matrix $\mathbf{Y}$, we can also learn the representations of item v_j and attribute a_n. With an estimation of the item-attribute relationship $\widehat{y}_{jn}$ (i.e., the quality of attribute a_n on item v_j), we define another squared loss as the following:

$$L_{IA} = \sum_{y_{jn} \in \mathcal{D}_{IA}} (y_{jn} - \widehat{y}_{jn})^2, \tag{9}$$

with $\mathcal{D}_{IA} = \{y_{in} \mid y_{in} \in \mathbf{Y} \wedge y_{in} \neq 0\}$ containing all non-zero entries in $\mathbf{Y}$. We hereby adopt a similar approach as described for user-attribute collaborative filtering to predict $\widehat{y}_{jn}$:

$$\widehat{y}_{in} = \tanh_{[1,N]}\left(\mathbf{w}_y^\top H_l^{IA}([\mathbf{v}_j; \mathbf{a}_n])\right), \tag{10}$$

where the rescaled activation function $\tanh_{[1,N]}(\cdot)$ is also adopted, and $\mathbf{w}_y \in \mathbb{R}^{d\times 1}$ is the weight vector. $H_l^{IA}(\cdot)$ is another l-layer residual feed-forward network whose input is the concatenation of item embedding $\mathbf{v}_j \in \mathbb{R}^{d\times 1}$ and attribute embedding $\mathbf{a}_n \in \mathbb{R}^{d\times 1}$. Note that $H_l^{IA}(\cdot)$ shares the same architecture with $H_l^{UA}(\cdot)$ but uses a different set of network weights and biases.

At the same time, since the learned attributes will serve the purpose of quantifying both user preferences and item characteristics, we make all attribute embeddings shared between user-attribute

CF and item-attribute CF. Hence, this enables the joint optimization of both objectives, leading to stronger expressiveness of learned attribute representations.

3.3 Estimating Missing Entries in X and Y

Suppose that we have already learned the representations for users, items and attributes that yield sufficiently small values for both L_{UA} and L_{IA}. Before entering the next stage where all the learned representations are further fine-tuned for recommendation, we need to infer the unobserved user-attribute and item-attribute values. In other words, we estimate the missing entries (i.e., 0-valued elements) in $\mathbf{X}$ and $\mathbf{Y}$ with Eq.(8) and Eq.(10) respectively, which can be viewed as two well-trained nonlinear regressors. Formally, this process is written as:

$$\widetilde{x}_{in} = \begin{cases} \widehat{x}_{in}, & \text{if } x_{in} = 0 \\ x_{in}, & \text{if } x_{in} \neq 0 \end{cases}, \quad \widetilde{y}_{jn} = \begin{cases} \widehat{y}_{jn}, & \text{if } y_{jn} = 0 \\ y_{jn}, & \text{if } y_{jn} \neq 0 \end{cases}. \tag{11}$$

We use $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ to denote the fully estimated user-attribute and item-attribute matrices. In addition, we use $\widetilde{\mathbf{x}}_i = [\widetilde{x}_{i1}, \widetilde{x}_{i2}, \cdots, \widetilde{x}_{i|\mathcal{A}|}]$ to denote the i-th row of $\widetilde{\mathbf{X}}$. Specifically, each element in $\widetilde{\mathbf{x}}_i$ corresponds to user u_i's preference on an item attribute, parameterized in the range of $[1, N]$. In a similar way, $\widetilde{\mathbf{y}}_j$ refers to the j-th row of $\widetilde{\mathbf{Y}}$, which is actually a collection of item v_j's performance scores (within $[1, N]$ range) regarding all attributes.

3.4 Uniting Substitution and Personalization Constraints for Recommendation

Given user u_i and query item v_q that she/he is currently browsing, we would like to recommend a list of items that are: (1) strong alternatives to v_q; and (2) of great interest to u_i. To achieve this, we extend Bayesian personalized ranking (BPR) [29] to the substitute recommendation setting. Despite being widely adopted for personalized recommendation, BPR is originally designed for optimizing pairwise user-item scoring functions, and lacks the consideration of affinity between two substitutable items. In this section, we propose a variant of BPR to unify both substitution and personalization. For user u_i and query item v_q, we firstly pair them with the ground truth substitute v_{j^+}, denoted by (u_i, v_q, v_{j^+}), implying that user u_i has eventually chosen v_{j^+} after browsing v_q. Correspondingly, a negative case (u_i, v_q, v_{j^-}) can be constructed, where v_{j^-} holds either or both of the properties: (1) v_{j^-} is never visited by u_i; (2) v_{j^-} is not substitutable to v_q. Thus, we can form a set of training samples consisting of $(u_i, v_q, v_{j^+}, v_{j^-})$, denoted by $\mathcal{D}_{Rec}$. With the training samples, we aim to learn a triplet ranking function $f(\cdot,\cdot,\cdot)$ to generate scalar ranking scores such that $f(u_i, v_q, v_{j^+}) > f(u_i, v_q, v_{j^-})$. In this regard, we define the following loss function that integrates BPR with substitutable item relationships, which we term L_{BPR-S}:

$$\begin{aligned} L_{BPR-S} &= -\log \prod_{(u_i,v_q,v_{j^+},v_{j^-})\in\mathcal{D}_{Rec}} \sigma\Big(f(u_i, v_q, v_{j^+}) - f(u_i, v_q, v_{j^-})\Big) \\ &= -\sum_{(u_i,v_q,v_{j^+},v_{j^-})\in\mathcal{D}_{Rec}} \log\bigg(\sigma\Big(f(u_i, v_q, v_{j^+}) - f(u_i, v_q, v_{j^-})\Big)\bigg), \end{aligned} \tag{12}$$

where $\sigma(\cdot)$ is the *Sigmoid* function. In general, we expect $f(\cdot,\cdot,\cdot)$ to fulfill two special constraints, namely the ***substitution constraint*** and ***personalization constraint***, to produce accurate recommendations. Before explaining the details, we let $f(u_i, v_q, v_j) = \gamma f_S(v_q, v_j) + (1-\gamma) f_P(u_i, v_j)$, and rewrite Eq.(12) as follows:

$$\begin{aligned} L_{BPR-S} = -\sum_{(u_i,v_q,v_{j^+},v_{j^-})\in\mathcal{D}_{Rec}} \sigma'\Big(&\gamma f_S(v_q, v_{j^+}) + (1-\gamma) f_P(u_i, v_{j^+}) \\ &- \gamma f_S(v_q, v_{j^-}) - (1-\gamma) f_P(u_i, v_{j^-})\Big), \end{aligned} \tag{13}$$

where we let $\sigma'(\cdot) = \log(\sigma(\cdot))$ to be succinct, $f_S(\cdot,\cdot)$ and $f_P(\cdot,\cdot)$ are respectively the pairwise item-item and user-item scoring functions complying with the substitution and personalization constraints. $\gamma \in (0, 1)$ is a hyperparameter for adjusting the trade-off between two terms. Next, we introduce the designs of $f_S(\cdot,\cdot)$ and $f_P(\cdot,\cdot)$.

Implementing Substitution Constraint. For an arbitrary item v_j ($j \in \{j^+, j^-\}$) paired with the query item v_q, $f_S(\cdot,\cdot)$ is expected to generate a high affinity score $f_S(v_q, v_j)$ if they are mutually substitutable, and a relatively low score if not. Different from substitute recommenders that calculate item-item similarities in a pure latent factor-based manner [28, 43, 50], $f_S(\cdot,\cdot)$ additionally incorporates information of explicit attributes on top of the learned item representations. Intuitively, a substitutable item must be functionally similar to the query item v_q, which is reflected by their similar distributions over attributes. That is to say, we can compare two items via their attribute information. So, item v_j can be represented as a weighted combination of all attribute embeddings:

$$\widetilde{\mathbf{v}}_j = \sum_{n=1}^{|\mathcal{A}|} \varphi_n^{qj} \mathbf{a}_n, \tag{14}$$

where $\widetilde{\mathbf{v}}_j$ is the representation of item v_j after the attribute aggregation, $\{\varphi_n^{qj}\}_{n=1}^{|\mathcal{A}|}$ is a probability distribution (i.e., $\sum_{n=1}^{|\mathcal{A}|} \varphi_n^{qj} = 1$) over all attributes, which is dependent on v_q and v_j:

$$\{\varphi_n^{qj}\}_{n=1}^{|\mathcal{A}|} = \text{softmax}\Big(\frac{\widetilde{\mathbf{y}}_q \odot \widetilde{\mathbf{y}}_j}{\beta}\Big), \tag{15}$$

where $\widetilde{\mathbf{y}}_q$ and $\widetilde{\mathbf{y}}_j$ are respectively the q-th and j-th row of the estimated item-attribute matrix $\widetilde{\mathbf{Y}}$, $\odot$ is the element-wise product of two vectors, and β is the scaling factor that controls the strength of dominating attributes. Instead of picking only a small fraction of most relevant attributes to represent each item [24, 52], we use *softmax* to avoid potential information loss by discriminatively considering all attributes. As discussed in Section 3.3, in $\widetilde{\mathbf{y}}_q$ and $\widetilde{\mathbf{y}}_j$, the n-th elements $\widetilde{y}_{qn}$ and $\widetilde{y}_{jn}$ reflect both items' quality on attribute a_n. Apparently, if item v_j is similar to the query item v_q on attribute a_n, Eq.(15) will generate a large weight φ_n^{qj}. The output $\{\varphi_n^{qj}\}_{n=1}^{|\mathcal{A}|}$ essentially carries all attribute-wise similarity scores between v_q and v_j. By setting β to different values, we can amplify or weaken the influence of larger values in $\{\varphi_n^{qj}\}_{n=1}^{|\mathcal{A}|}$.

Then, we compute the substitution affinity score between v_q and v_j:

$$f_S(v_q, v_j) = \mathbf{w}_s^\top [\mathbf{v}_q \odot \mathbf{v}_j; \widetilde{\mathbf{v}}_j], \tag{16}$$

where $\mathbf{w}_s \in \mathbb{R}^{2d\times 1}$ is the weight for linear projection, and $\mathbf{v}_q \odot \mathbf{v}_j$ models the element-wise similarity between the latent vectors of two items. In Eq.(16), we concatenate information from two aspects: (1) the comparison between the latent representations of v_q and v_j; (2) the attribute-aware representation of v_j generated by comparing explicit item attributes. As this provides substantial information for generating a discriminative ranking score $f_S(v_q, v_j)$, we adopt a simple linear projection rather than sophisticated deep neural network-based regressors [49] for efficient computation.

Implementing Personalization Constraint. For user u_i and item v_j, The pairwise scoring function $f_P(u_i, v_j)$ should yield a large score if v_j has been visited by u_i, and vice versa. Similar to $f_S(\cdot,\cdot)$, we firstly generate another attribute-aware representation of v_j, denoted by $\widetilde{\mathbf{v}}'_j$:

$$\widetilde{\mathbf{v}}'_j = \sum_{n=1}^{|\mathcal{A}|} \lambda_n^{ij} \mathbf{a}_n, \tag{17}$$

with the attentive weights $\{\lambda_n^{ij}\}_{n=1}^{|\mathcal{A}|}$ computed by comparing user u_i's interests on all attributes (i.e., $\widetilde{\mathbf{x}}_i$) and item v_j's performance on all attributes (i.e., $\widetilde{\mathbf{y}}_j$):

$$\{\lambda_n^{ij}\}_{n=1}^{|\mathcal{A}|} = \text{softmax}(\frac{\widetilde{\mathbf{x}}_i \odot \widetilde{\mathbf{y}}_j}{\epsilon}), \tag{18}$$

where we also adopt a scaling factor ϵ to either diverge or smoothen the probability distribution, and $\widetilde{\mathbf{x}}_i \in \widetilde{\mathbf{X}}$ quantifies user u_i's concerns about each attribute. Intuitively, Eq.(18) justifies how well item v_j's attribute quality (i.e., $\widetilde{\mathbf{y}}_j$) aligns with u_i's demand, and then Eq.(17) generates a combinatorial representation for item v_j as a weighted sum of all different attributes. Afterwards, a lightweight linear projection scheme is applied to generate the personalization ranking score:

$$f_P(u_i, v_j) = \mathbf{w}_p^\top [\mathbf{u}_i \odot \mathbf{v}_j; \widetilde{\mathbf{v}}'_j], \tag{19}$$

where $\mathbf{w}_p \in \mathbb{R}^{2d\times 1}$ is the projection weight, and $\mathbf{u}_i \odot \mathbf{v}_j$ further infuses the comparison between the user's and item's latent representations to allow for precise user-item preference prediction.

3.5 Optimization Strategy

As A2CF is built upon the deep neural network structure, we learn the model parameters on multiple objectives with a mini-batch stochastic gradient decent algorithm, namely Adam [16] optimizer. To be specific, we firstly optimize $L' = L_{UA} + L_{IA}$, i.e., the combined loss for user-attribute CF and item-attribute CF for T_1 iterations. Then, we estimate $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ matrices, and optimize the recommendation loss L_{BPR-S} for T_2 iterations. Notably, $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ are precomputed before training the recommendation part, thus avoiding redundant and inefficient computations. We update all model parameters in each iteration and repeat the entire training process described above until L_{BPR-S} converges or is sufficiently small.

We tune the hyperparameters using grid search. Specifically, the latent dimension d is searched in $\{16, 32, 64, 128, 256\}$; the depth of the residual feed-forward network l is searched in $\{1, 2, 3, 4, 5\}$; the trade-off coefficient γ is searched in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$; and both scaling factors β and ϵ are searched in $\{1, d^{0.25}, d^{0.5}, d^{0.75}, d\}$. We will further discuss the impact of these key hyperparameters to the recommendation performance in Section 6.4. For optimizing the final recommendation loss L_{BPR-S}, we draw 5 negative samples (u_i, v_q, v_{j^-}) for each positive label (u_i, v_q, v_{j^+}) in training set $\mathcal{D}_{Rec}$. We set the the learning rate to 1×10^{-3} and batch size to 256 according to device capacity. To prevent overfitting, we adopt a dropout [31] ratio of 0.4 on all deep layers of A2CF.

4 UTILIZING A2CF FOR RECOMMENDATION AND INTERPRETATION

After obtaining a fully-trained A2CF model, here we showcase the roadmap towards personalized and interpretable substitute recommendation using A2CF.

4.1 Personalized Substitute Recommendation

As a natural extension to personalized top-K recommendation [13, 40], the essence of personalized substitute recommendation is to assign a ranking score to each possible item v_j given the context (u_i, v_q) (i.e., the user and query item), and then recommend K top-ranked items to the user. With an input triplet denoted by (u_i, v_q, v_j), recall that we define the triplet ranking function as $f(u_i, v_q, v_j) = \gamma f_S(v_q, v_j) + (1-\gamma) f_P(u_i, v_j)$. Hence, for each $v_j \in \mathcal{V}$, we can generate a ranking score for triplet (u_i, v_q, v_j) by measuring how well the item v_j fits both the substitution and personalization constraints. Ideally, only the items that satisfy both parts will be ranked at the top of the recommendation list.

4.2 Personalized Interpretation

Latent factor-based substitute recommenders [28, 43, 50] are uninterpretable as the ranking scores are directly computed via the pure latent representations of items and/or users. Consequently, when performing personalized substitute recommendation, a user may find it difficult to understand the "try this instead" suggestions from the recommender systems. In contrast, our proposed A2CF is advantageous owing to its capability of generating intuitive interpretations. Specifically, A2CF composes interpretations by considering two aspects: (1) the key attributes u_i will consider when searching for items similar to v_q; (2) how the recommended item v_j performs compared with the query item v_j on these key attributes. With the estimated user-attribute matrix $\widetilde{\mathbf{X}}$ and item-attribute matrix $\widetilde{\mathbf{Y}}$, we address these two points by comparing v_q with v_j in a quantitative way:

$$\begin{aligned}\{\Delta_n^{iqj}\}_{n=1}^{|\mathcal{A}|} &= \widetilde{\mathbf{x}}_i \odot \widetilde{\mathbf{y}}_j - \widetilde{\mathbf{x}}_i \odot \widetilde{\mathbf{y}}_q \\ &= \widetilde{\mathbf{x}}_i \odot (\widetilde{\mathbf{y}}_j - \widetilde{\mathbf{y}}_q).\end{aligned} \tag{20}$$

In short, Eq.(20) aligns the user's demand with each item's quality attribute-wise, and uses subtraction to compute the advantages (or disadvantages) of v_j against v_q regarding each attribute a_n, denoted by Δ_n^{iqj}. Next, Z attributes $\{a_{n_1}, a_{n_2}, ..., a_{n_Z}\}$ with highest values in $\{\Delta_n^{iqj}\}_{n=1}^{|\mathcal{A}|}$ are selected to generate the personalized interpretation. In this paper, we leverage a template-based interpretation scheme to state the key reasons for recommending v_j as the substitute of v_q for user u_i. The template is as follows:

$$\left[\begin{array}{l}\text{“Based on the item } v_q \text{ you are currently browsing, we} \\ \text{recommend you to try } v_j \text{ instead because it comes with:} \\ \text{[adjective] } a_{n_1}\text{, [adjective] } a_{n_2}, \cdots, \text{ and [adjective] } a_{n_Z}\text{.”}\end{array}\right] \tag{21}$$

where for $1 \le z \le Z$, each [adjective] is determined via:

$$[\text{adjective}] = \begin{cases} \text{“better”}, & \text{if } \Delta_{n_z}^{iqj} > 0 \\ \text{“comparable”}, & \text{otherwise} \end{cases}. \tag{22}$$

It is worth mentioning that we prefer assigning a small value to Z (usually below 5) to keep the interpretations concise, because users have very limited attention and time to read the generated explanations, and it rarely happens for $\Delta_{n_z}^{iqj} \le 0$ when only the top attributes are selected.

5 EXPERIMENTAL SETTINGS

5.1 Datasets

We consider three real-world review datasets commonly used for recommendation tasks, which are originally crawled from Amazon and made public by [24], and different product categories are

Table 1: Statistics of datasets in use.

Dataset	#Review	#User	#Item	#Attribute
Cellphone	1,639,166	665,900	48,763	2,178
Automotive	2,212,111	420,394	132,467	3,501
Office	1,641,901	515,813	58,534	2,801

treated as separate datasets. We use three of the largest categories, namely Cellphone and Accessories (Cellphone for short), Automotive, and Office. Based on well-established rules adopted in the substitute recommendation literature [24, 28, 49, 50], item substitution relationships are extracted from all datasets. Specifically, two items v and v' are substitutable if: (1) users viewed v also viewed v', or (2) users viewed v eventually bought v'.

Following [21, 29], we filter out inactive users with less than 5 interacted items and unpopular items visited by less than 5 users. The primary statistics are shown in Table 1, where each review corresponds to a user-item interaction record. Note that we use the workflow described in Section 2.2 to extract attributes from each individual dataset, and attributes with less than 2 mentions are discarded to reduce excessive noise from the data. We split all datasets with the ratio of 80%, 10% and 10% for training, validation and test, respectively.

5.2 Evaluation Protocols

Evaluating Recommendation Effectiveness. We leverage two ranking metrics, namely hit ratio at rank K (HR@K) and normalized discounted cumulative gain at rank K (NDCG@K) that are widely adopted in recommendation research [37, 39, 47]. For each positive test instance (u_i, v_q, v_{j^+}), we mix the ground truth v_{j^+} with J random negative items, then rank all these $J+1$ items and compute the HR@K and NDCG@K scores. We set $J = 1,000$ following [6] and adopt $K = 5, 10, 20, 50$ for evaluation.

Sampling Query Items. However, a sticking point is the selection criteria of query item v_q. Though v_{j^+} and v_{j^-} can be easily obtained via each user's interaction record, the majority of popular e-commerce datasets like Amazon [24] and Yelp [3] only record actual purchases as user-item interactions. Such limitation prevents us from knowing which other options (i.e., query items) that u_i has browsed before deciding to purchase v_{j^+}. Hence, for each observed user-item interaction (u_i, v_{j^+}), we propose to leverage ***popularity-biased sampling*** to pick v_q to simulate the query item:

I. For (u_i, v_{j^+}), find all substitutes of v_{j^+} that u_i has never interacted with, denoted as $\mathcal{S}_{ij^+}$;
II. For each $v_q \in \mathcal{S}_{ij^+}$, calculate the amount of users it has interacted with, denoted as pop_q;
III. Draw one v_q with probability $P(v_q) \sim pop_q^{0.75}$.

Note that there will be only one query item v_q sampled for every (u_i, v_{j^+}) instance. As such, we can obtain triplets (u_i, v_q, v_{j^+}) as the ground truth for training and test. The rationale of popularity-biased sampling is that, when searching for a specific type of product to buy, customers usually tend to start with the most popular ones, and then expand their candidates as their demands become clearer. Motivated by successful graph theory practices [3, 34], we smoothen the item popularity with 0.75 power. This moderately increases the chance of drawing unpopular items and helps validate the robustness of the model with diverse query compositions. It is worth noting that, in the test set, we restrict that the ground truth item v_{j^+} in each instance (u_i, v_q, v_{j^+}) is neither paired with user u_i nor v_q for model training. This allows for generating the hardest possible test instances because the test item is always new for both the user and the query item.

Evaluating Interpretation Quality. In the emerging area of explainable recommendation, the quality of text-based interpretations is mainly tested via either human evaluation [36, 52] or visualization [12, 41]. Despite being effective to some extent, they lack quantitative measurements for the generated interpretations. Because the process of generating attribute ranking lists for interpretation can be viewed as a special type of document retrieval task, metrics like mean average precision (MAP) from relevant domains [20, 44] can be used to evaluate the ranks of attributes mentioned in the actual user reviews. However, it is pointless to evaluate interpretations generated for non-ground truth items, as the user has never visited nor reviewed them. Therefore, we only consider the MAP score for the interpretation for each ground truth item v_{j^+}, and trade it off using the NDCG score for v_{j^+}. In this way, we define ***attribute trade-off coverage*** (ATC) as a generic evaluation metric for attribute-aware interpretable recommendation models:

$$\text{ATC} = \underset{\forall(u_i, v_q, v_{j^+})}{\text{MEAN}} \frac{2 \times \text{MAP}_{iqj^+} \times \text{NDCG}_{iqj^+}}{\text{MAP}_{iqj^+} + \text{NDCG}_{iqj^+}}, \tag{23}$$

where MAP_{iqj^+} is the mean average precision of $\{\Delta_n^{iqj}\}_{n=1}^{|\mathcal{A}|}$ (i.e., the attribute ranking list used for recommendation explanation) w.r.t. actually mentioned attributes in user u_i's review on item v_{j^+}, and NDCG_{iqj^+} is the NDCG score of ground truth item v_{j^+} in the ranking list. By taking the harmonic mean of MAP_{iqj^+} and NDCG_{iqj^+}, we expect a large ATC score only when the interpretation and recommendation are both accurate for test case (u_i, v_q, v_{j^+}).

5.3 Baseline Methods

We briefly introduce the baseline methods for comparison below.

- **Sceptre** [24]**:** The key idea of Sceptre is combining topic modeling with supervised link prediction to predict possible substitutable relationships between two products.
- **PMSC** [43]**:** It firstly embeds products into relation-specific spaces, then incorporates multiple path constraints to enhance the expressiveness of learned product embeddings.
- **SPEM** [50]**:** It models substitutable product relationships by penalizing first-order proximity, rewarding second-order proximity and semantic similarity in the constructed product co-purchasing graph, thus making the representations of two substitutable products align closely in the latent space
- **CLVA** [28]**:** As another state-of-the-art substitute recommendation model, it links two variational autoencoders conditioned on the observed links among items. On top of that, the integration of probabilistic matrix factorization further allows for personalized substitute recommendation.

5.4 Hyperparameter Settings

To be consistent, we report the overall recommendation performance of A2CF with a unified set of parameters where $d = 64$, $l = 1$, $\gamma = 0.7$ and $\beta = \epsilon = d^{0.5} = 8$. The effect of different hyperparameter settings will be discussed in Section 6.4. For all baselines, we use grid search to obtain their optimal hyperparameters.

Table 2: Recommendation results. Numbers in bold face are the best results for corresponding metrics.

Dataset	Method	HR@K				NDCG@K			
		K=5	K=10	K=20	K=50	K=5	K=10	K=20	K=50
Cellphone	Sceptre [24]	0.2009	0.2689	0.3486	0.4605	0.1570	0.1695	0.1998	0.2421
	PMSC [43]	0.1085	0.1276	0.1731	0.2023	0.0827	0.0995	0.1086	0.1125
	SPEM [50]	0.1187	0.1449	0.1858	0.2251	0.0951	0.1034	0.1135	0.1300
	CLVA [28]	0.1795	0.2621	0.3434	0.5167	0.1224	0.1421	0.1729	0.1976
	A2CF	**0.2466**	**0.3449**	**0.4548**	**0.6229**	**0.1663**	**0.1980**	**0.2263**	**0.2845**
Automotive	Sceptre [24]	0.1618	0.1709	0.2441	0.3346	0.1021	0.1087	0.1225	0.1401
	PMSC [43]	0.1249	0.1653	0.1904	0.2955	0.1078	0.1183	0.1292	0.1485
	SPEM [50]	0.1014	0.1113	0.1305	0.1787	0.0934	0.0965	0.1014	0.1103
	CLVA [28]	0.1565	0.2201	0.3179	0.3868	0.1024	0.1397	0.1531	0.1754
	A2CF	**0.2187**	**0.2978**	**0.3899**	**0.5518**	**0.1532**	**0.1788**	**0.2021**	**0.2395**
Office	Sceptre [24]	0.2489	0.3623	0.4762	0.6100	0.1570	0.1948	0.2242	0.2512
	PMSC [43]	0.1345	0.1924	0.2596	0.3701	0.0954	0.1182	0.1425	0.1759
	SPEM [50]	0.0741	0.0906	0.1149	0.1528	0.0579	0.0632	0.0693	0.0735
	CLVA [28]	0.2322	0.3750	0.4869	0.6343	0.1480	0.1946	0.2378	0.2733
	A2CF	**0.3143**	**0.4220**	**0.5310**	**0.6644**	**0.2193**	**0.2540**	**0.2815**	**0.3101**

6 EXPERIMENTAL RESULTS AND ANALYSIS

Following the settings in Section 5, we conduct experiments[1] to evaluate the performance of A2CF regarding both recommendation effectiveness and interpretation quality. In particular, we aim to answer the following research questions (RQs) via experiments:

RQ1: How effectively can A2CF perform personalized substitute recommendation compared with state-of-the-art baselines?

RQ2: How is the quality of attribute-based interpretations generated by A2CF?

RQ3: What is the contribution of each key component of the proposed model structure?

RQ4: How the hyperparameters affect the performance of A2CF in terms of recommendation effectiveness?

6.1 Recommendation Effectiveness (RQ1)

We summarize all models' performance on personalized substitute recommendation with Table 2. We discuss our findings as follows.

The first observation is that our proposed A2CF outperforms all baselines consistently and significantly (p-value < 0.01) on these three datasets. Compared with the second best results on all datasets, A2CF achieved an average improvement of 28.1% and 29.2% respectively on HR@5 and NDCG@5. As PMSC and SPEM do not take personalization into consideration, the recommended substitutes can hardly match users' personal preferences, leading to inferior performance. At the mean time, as a personalized substitute recommender, CLVA additionally utilizes latent user representations learned via probabilistic matrix factorization, and thus produces better recommendation results than PMSC and SPEM. However, by implementing substitution and personalization constraints via an attribute-aware scheme, A2CF represents the state-of-the-art effectiveness on this recommendation task.

Second, compared with deep learning models that are entirely based on product graphs (i.e., PMSC and SPEM), models that make use of the item attributes (i.e., A2CF, Sceptre and CLVA) show obvious advantages in terms of recommendation performance. This further validates the benefit of incorporating attribute information for personalized substitute recommendation. Particularly, though the pure LDA-based Sceptre is neither a personalized recommender nor enhanced by deep neural networks, it can still yield highly competitive results, and is even able to produce higher recommendation accuracy than CLVA in some cases (e.g., HR@5 on all three datasets). This is probably because when modelling items with attributes, Sceptre additionally introduces a category tree for selecting different attributes, which helps it learn more specific and high-quality attribute-based item representations.

Lastly, we find that the sparsity of user-item interactions and the amount of available attributes have opposite influences to the recommendation effectiveness of A2CF. On one hand, Cellphone and Office have similar sparsity (both are around 99.995%), but the larger amount of attributes allows A2CF to thoroughly learn attribute-aware representations for both users and items, leading to better results on Office. One the other hand, despite offering the most attributes, the Automotive dataset witnesses relatively lower performance of A2CF due to its substantially higher sparsity.

6.2 Interpretation Quality (RQ2)

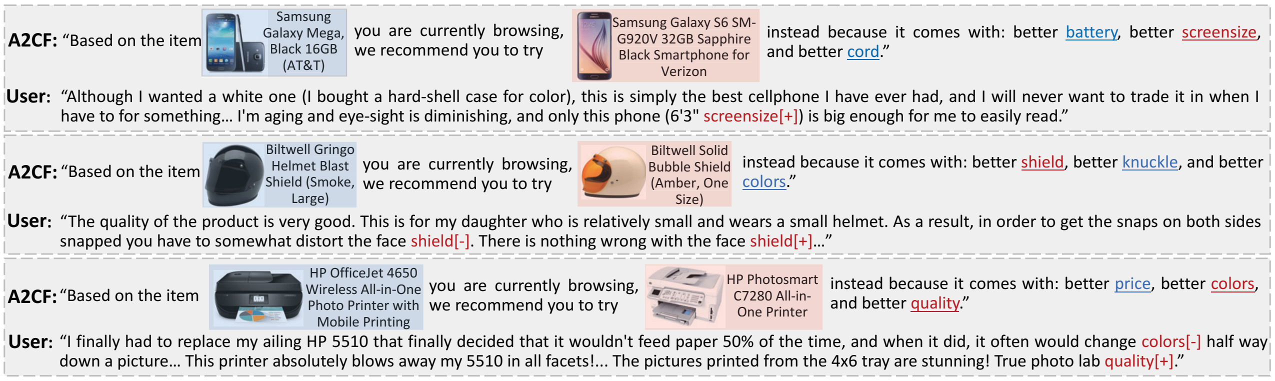
Quantitative Analysis. To quantitatively evaluate the interpretation quality of A2CF, we adopt the metric ATC proposed in Section 5.2, and compare A2CF with the other two attribute-based substitute recommenders Sceptre and CLVA. For both baselines, an item-attribute relevance score can be inferred, which enables us to obtain an attribute ranking list for each ground truth item to compute the ATC score. For fairness, we truncate the attribute ranking list of Sceptre, CLVA and A2CF with the length of 500 for evaluation because different numbers of attributes are used by them. Then, we present the quantitative results in Table 3. Apparently, A2CF leads Sceptre and CLVA regarding ATC scores on all three datasets due to its ability to generate personalized interpretations.

Case Study. We further conduct a case study in Figure 2 to qualitatively examine the interpretability of A2CF. Specifically, we visualize and compare the interpretations generated by A2CF and users' real review texts. Note that we use the top three attributes

Table 3: Quantitative results on interpretation quality.

Method	ATC		
	Cellphone	Automotive	Office
Sceptre	0.0108	0.0094	0.0124
CLVA	0.0096	0.0077	0.0112
A2CF	**0.0132**	**0.0113**	**0.0154**

[1]Public access to codes: https://bit.ly/bitbucket-A2CF



A2CF: "Based on the item Samsung Galaxy Mega, Black 16GB (AT&T) you are currently browsing, we recommend you to try Samsung Galaxy S6 SM-G920V 32GB Sapphire Black Smartphone for Verizon instead because it comes with: better battery, better screensize, and better cord."

User: "Although I wanted a white one (I bought a hard-shell case for color), this is simply the best cellphone I have ever had, and I will never want to trade it in when I have to for something... I'm aging and eye-sight is diminishing, and only this phone (6'3" screensize[+]) is big enough for me to easily read."

A2CF: "Based on the item Biltwell Gringo Helmet Blast Shield (Smoke, Large) you are currently browsing, we recommend you to try Biltwell Solid Bubble Shield (Amber, One Size) instead because it comes with: better shield, better knuckle, and better colors."

User: "The quality of the product is very good. This is for my daughter who is relatively small and wears a small helmet. As a result, in order to get the snaps on both sides snapped you have to somewhat distort the face shield[-]. There is nothing wrong with the face shield[+]..."

A2CF: "Based on the item HP OfficeJet 4650 Wireless All-in-One Photo Printer with Mobile Printing you are currently browsing, we recommend you to try HP Photosmart C7280 All-in-One Printer instead because it comes with: better price, better colors, and better quality."

User: "I finally had to replace my ailing HP 5510 that finally decided that it wouldn't feed paper 50% of the time, and when it did, it often would change colors[-] half way down a picture... This printer absolutely blows away my 5510 in all facets!... The pictures printed from the 4x6 tray are stunning! True photo lab quality[+]."

Figure 2: Case study on interpretations generated by A2CF on Cellphone (top), Automotive (middle), and Office (bottom). Comparisons are performed between interpretations and users' real reviews on the ground truth items.

to explain each recommendation result, i.e., $Z = 3$. In each case, the query item is marked blue, while the interpretation and review are for the same ground truth item in the red box. Each ground truth item is positioned higher than 10 in the ranking list. We use red to mark attributes that are mentioned in both the interpretation and the corresponding review, and label the sentiment of attributes (positive/negative) in the user review. For the first two users respectively browsing a cellphone and a helmet, A2CF successfully recognizes their demand on "screensize" and "shield", and recommends the correct items that perform better than the query items on corresponding attributes. We can also see there is positive user feedback on both the "screensize" and "shield" of the recommended cellphone and helmet. Furthermore, in the third case, two better attributes "colors" and "quality" of the recommended printer are used for interpretation, which are both mentioned by the user. Though attribute "colors" expresses negative sentiment in this review, in fact the user is complaining about a previously purchased printer. This further validates our model's capability of aligning item attributes with user preferences to generate interpretations.

6.3 Importance of Key Components (RQ3)

To better understand the performance gain from the major components proposed in A2CF, we perform ablation analysis on different degraded versions of A2CF. Table 4 summarizes the recommendation outcomes in terms of HR@10.

Removing Substitution Constraint. The substitution constraint in Eq.(13) enforces that the recommended items should hold substitutable relationship with the query item. We remove the substitution constraint by setting $\gamma = 0$. After that, a severe performance decrease can be observed from all datasets, where the performance on Automotive is the most vulnerable. Because Automotive contains substantially more items than Cellphone and Office, the substitution constraint is of great significance to helping A2CF filter out irrelevant items and ensure successful recommendations.

Removing Personalization Constraint. Similar to the previous variant, we remove the personalization constraint by setting $\gamma = 1$ in Eq.(13). As personal preferences are not properly captured in this variant, A2CF suffers from a significant performance drop due to the absence of personalization constraint. Compared with removing substitution constraint, the deletion of personalization constraint has more negative effect on datasets with relatively denser user-item interactions (i.e., Cellphone and Office).

Table 4: Ablation test with different model architectures.

Variant	HR@10		
	Cell-phone	Auto-motive	Office
Default	**0.3449**	**0.2978**	**0.4220**
Removing Substitution Constraint Eq.(13) $\rightarrow -\sum_{(u_i,v_{j^+},v_{j^-})\in\mathcal{D}_{Rec}} \sigma'\Big(f_P(u_i,v_{j^+}) - f_P(u_i,v_{j^-})\Big)$	0.2772	0.1997	0.3745
Removing Personalization Constraint Eq.(13) $\rightarrow -\sum_{(v_q,v_{j^+},v_{j^-})\in\mathcal{D}_{Rec}} \sigma'\Big(f_S(v_q,v_{j^+}) - f_S(v_q,v_{j^-})\Big)$	0.2523	0.2312	0.3510
Removing Attribute Aggregation for Items Eq.(16) $\rightarrow \mathbf{w}_s^\top[\mathbf{v}_q \odot \mathbf{v}_j]$	0.3116	0.2733	0.3994
Removing Attribute Aggregation for Users Eq.(19) $\rightarrow \mathbf{w}_p^\top[\mathbf{u}_i \odot \mathbf{v}_j]$	0.3075	0.2708	0.3763

Removing Attribute Aggregation for Items. With the removal of the attribute aggregation for items in Eq.(16), there is a noticeable performance decrease for A2CF. The aggregated attribute-aware item representation (i.e., $\widetilde{\mathbf{v}}_j$) aims to fully leverage the attributes to offer more information for item-item substitutable relationship predictions. Hence, this verifies the efficacy of merging attribute information with latent item representations for identifying substitutable relationships among items.

Removing Attribute Aggregation for Users. We build the forth variant of A2CF by deleting the aggregated attribute information for users in Eq.(19). Generally, the inferior performance on three datasets indicates that considering users' preferred attributes practically improves user-item preference modelling.

6.4 Impact of Hyperparameters (RQ4)

We answer RQ4 by investigating the performance fluctuations of A2CF with varied hyperparameters. Particularly, we study our model's sensitivity to network depth l, coefficient γ, as well as two scaling factors β and ϵ. Based on the standard hyperparameter setup in Section 5.4, we vary the value of one hyperparameter while keeping the others unchanged, and record the new recommendation result achieved. Similar to Section 6.3, HR@10 is used for demonstration. Figure 3 lays out the results with different parameter settings.

Impact of l. The value of the network depth l is examined in $\{1, 2, 3, 4, 5\}$. In general, A2CF benefits from a relatively larger l on all three datsets, but the performance improvement tends to stop when l reaches a certain size (3 and 4 in our case) due to overfitting.

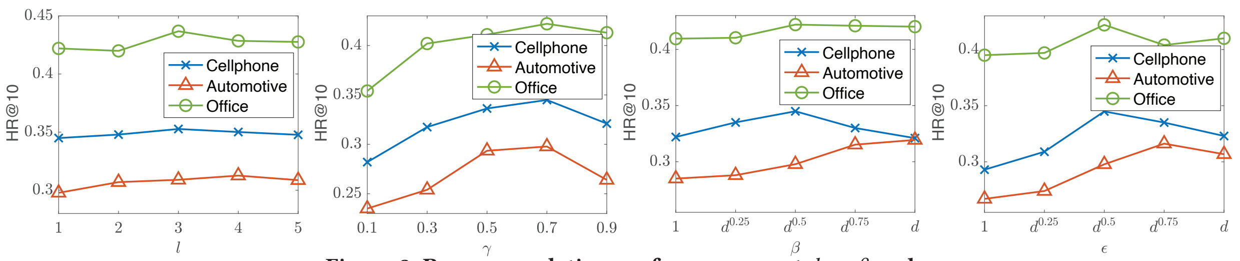


Figure 3: Recommendation performance w.r.t. l, γ, β and ϵ.

Impact of γ. We also study the impact of $\gamma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ which balances the substitution and personalization constraints. The HR@10 scores on all datasets imply that, in A2CF, by laying enough emphasis on the substitution constraint (0.7 in our case), the joint effect of personalization and substitution can be maximized for recommendation.

Impact of β and ϵ. The scaling factors β and ϵ are respectively introduced in Eq.(15) and Eq.(18) to control the influence of dominating attributes when aggregating attribute information for items and users. As can be concluded from Figure 3, A2CF behaves differently on varied datasets when the scaling factors are adjusted in $\{1, d^{0.25}, d^{0.5}, d^{0.75}, d\}$ ($d = 64$ in our setting). This is possibly caused by the differences in the quantity of attributes extracted from three datasets. Since Automotive has more attributes due to higher diversity of items, larger β and ϵ will encourage a more scattered probability distribution over all attributes, so more attributes are taken into consideration when modelling users and items, making the model adaptive to high data sparsity.

7 RELATED WORK

7.1 Product Relationship Mining

Understanding how products relate to each other is important to the fulfilment of customer satisfaction in different online shopping stages. In economics literature, two common product relationships are substitution and complement [23]. Recently, product relationship mining has become a prospective research direction to enhance existing recommender systems that are unable to differentiate the relationships among products. The problem of mining product relationships is first introduced to recommendation research in [24], where a topic model-based approach Sceptre is proposed. With latent Dirichlet allocation (LDA), Sceptre infers product relationships by comparing the topic distributions of two products. More recently, the trend of utilizing reviews has carried over to neural network approaches, such as RRN [49] that learns product embeddings from both the textual reviews and manually crafted features. In contrast to those hybrid models combining reviews with product graphs, pure graph-based methods like PMSC [43] and SPEM [50] are also introduced to fully utilize various properties in a product graph, e.g., path constraints and node proximities.

Unfortunately, as recommender systems, all the aforementioned methods treat substitute recommendation as a pure item-based search task, thus neglecting the key aspect of personalization. As a recent solution, CLVA [28] fuses probabilistic matrix factorization with variational autoencoders to support personalized recommendation by jointly modelling user ratings leverages and item relationships. However, like most existing product relationship mining methods, the latent factor-based representation learning scheme in CLVA hinders it from offering explainable recommendations, rendering it difficult to fully convince and attract potential customers.

7.2 Attribute Modelling for Recommendation

In conventional top-K recommendation, a large body of works adopt collaborative filtering methods e.g., matrix factorization to utilize implicit interactions to infer the links between users and items [17, 22, 29]. The predominant problem is usually learning latent factors for users and items, and predicting user-item interactions by ranking the inner product of the user and item embeddings [17].

Despite the promising recommendation quality of latent factor models, their latent characteristics make it frustratingly difficult to explain the generated recommendations [36], and such methods fail to adaptively generate recommendations when we are aware of a user's most cared features [52]. To overcome the limitations of latent factor models, review-based attribute modelling has attracted growing attention in recommendation research [7, 8]. Attributes are commonly extracted from reviews [8, 10, 12] or tags [38, 45, 46] associated with each item. Such methods assume that a user's decision of purchase is largely determined by how each product meets her/his expectations on several key attributes [52]. Ganu et al. [9] utilize aspects and sentiments manually crafted from restaurant reviews to enhance the performance of rating prediction. With the increasingly available review data, Zhang et al. [52] successfully combine techniques for automated attribute-level sentiment analysis with matrix factorization. Other subsequent variants are proposed to better capture the complex yet subtle relationships among users, products and attributes, such as modelling such heterogeneous relationships as a tripartite graph with TriRank [12], utilizing joint factorization in MTER [36], and incorporating attention networks into AARM [10]. Attribute modelling opens up an opportunity for generating interpretations for the recommended items. Existing attribute-aware methods, however, can only produce generic explanations [36] as they do not account for the attention shifts among different attributes when users are browsing different items. In contrast, our proposed A2CF is able to firstly infer a user's desired attributes, and retrieve advantageous attributes of the recommended items to construct personalized interpretations.

8 CONCLUSION

In this paper, we study a new research problem, namely personalized and interpretable substitute recommendation, and propose a novel A2CF model as a solution. A2CF fully utilizes attribute information extracted from reviews, which effectively bridges user preferences and item properties for generating personalized substitute recommendations and endows the model with high explainability. The experimental results evidence that A2CF can yield superior performance on both recommendation and interpretation.

ACKNOWLEDGEMENT

This work has been supported by Australian Research Council (Grant No. DP190101985, DP170103954 and FT200100825) and National Natural Science Foundation of China (Grant No. NSFC 61806035, U1936217, 61732008 and 61725203).

REFERENCES

[1] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. 2019. Personalized Bundle List Recommendation. In *WWW*. 60–71.

[2] Hongxu Chen, Hongzhi Yin, Tong Chen, Weiqing Wang, Xue Li, and Xia Hu. 2020. Social Boosted Recommendation with Folded Bipartite Network Embedding. *TKDE* (2020).

[3] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*.

[4] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching user with item set: collaborative bundle recommendation with deep attention network. In *IJCAI*. 2095–2101.

[5] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. AIR: Attentional intention-aware recommender systems. In *ICDE*. 304–315.

[6] Tong Chen, Hongzhi Yin, Quoc Viet Hung Nguyen, Wen-Chih Peng, Xue Li, and Xiaofang Zhou. 2020. Sequence-Aware Factorization Machines for Temporal Predictive Analytics. *ICDE* (2020).

[7] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan S Kankanhalli. 2018. A^ 3NCF: An Adaptive Aspect Attention Model for Rating Prediction.. In *IJCAI*. 3748–3754.

[8] 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.

[9] Gayatree Ganu, Yogesh Kakodkar, and AméLie Marian. 2013. Improving the quality of predictions using textual information in online user reviews. *Information Systems* 38, 1 (2013), 1–15.

[10] Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng, and Tat-Seng Chua. 2019. Attentive aspect modeling for review-aware recommendation. *TOIS* 37, 3 (2019), 28.

[11] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *SIGKDD*. 1569–1577.

[12] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*. 1661–1670.

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

[14] Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, and Julian McAuley. 2019. Complete the Look: Scene-based Complementary Product Recommendation. In *CVPR*. 10532–10541.

[15] Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2018. Recommendation through mixtures of heterogeneous item relationships. In *CIKM*. 1143–1152.

[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2015).

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

[18] Pigi Kouki, Ilias Fountalis, Nikolaos Vasiloglou, Nian Yan, Unaiza Ahsan, Khalifeh Al Jadda, and Huiming Qu. 2019. Product collection recommendation in online retail. In *RecSys*. 486–490.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[20] Binyang Li, Lanjun Zhou, Shi Feng, and Kam-Fai Wong. 2010. A unified graph model for sentence-based opinion retrieval. In *ACL*. 1367–1375.

[21] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. 2016. Point-of-interest recommendations: Learning potential check-ins from friends. In *SIGKDD*. 975–984.

[22] Qinghua Liu, Andrew Henry Reiner, Arnoldo Frigessi, and Ida Scheel. 2019. Diverse personalized recommendations with uncertainty from implicit preference data with the Bayesian Mallows model. *Knowledge-Based Systems* 186 (2019), 104960.

[23] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. 1995. *Microeconomic theory*. Vol. 1. Oxford university press New York.

[24] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *SIGKDD*. 785–794.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[26] Wendy W Moe. 2006. An empirical two-stage choice model with varying decision rules applied to internet clickstream data. *Journal of Marketing Research* 43, 4 (2006), 680–692.

[27] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. In *SIGIR*. 1073–1076.

[28] Vineeth Rakesh, Suhang Wang, Kai Shu, and Huan Liu. 2019. Linked variational autoencoders for inferring substitutable and supplementary items. In *WSDM*. 438–446.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[30] Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma. 2017. Neurostylist: Neural compatibility modeling for clothing matching. In *MM*. 753–761.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* (2014), 1929–1958.

[32] Ke Sun, Tong Chen, Tieyun Qian, Yiqi Chen, Hongzhi Yin, and Ling Chen. 2019. What can history tell us? Identifying relevant sessions for next-item recommendation. In *CIKM*. 1593–1602.

[33] Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2017. Collaborative intent prediction with real-time contextual data. *TOIS* (2017).

[34] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.

[35] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *CIKM*. 1133–1142.

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

[37] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next Point-of-Interest Recommendation on Resource-Constrained Mobile Devices. In *The Web Conference*. 906–916.

[38] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2015. Geo-SAGE: A geographical sparse additive generative model for spatial item recommendation. In *SIGKDD*. 1255–1264.

[39] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming ranking based recommender systems. In *SIGIR*. 525–534.

[40] Weiqing Wang, Hongzhi Yin, Shazia Sadiq, Ling Chen, Min Xie, and Xiaofang Zhou. 2016. Spore: A sequential personalized spatial item recommender system. In *ICDE*.

[41] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. *SIGKDD* (2019).

[42] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[43] 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*. 619–627.

[44] Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013. Automatic coupling of answer extraction and information retrieval. In *ACL*. 159–165.

[45] Hongzhi Yin, Zhiting Hu, Xiaofang Zhou, Hao Wang, Kai Zheng, Quoc Viet Hung Nguyen, and Shazia Sadiq. 2016. Discovering interpretable geo-social communities for user behavior prediction. In *ICDE*. 942–953.

[46] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. 2016. Adapting to user interest drift for poi recommendation. *TKDE* (2016), 2566–2581.

[47] Hongzhi Yin, Xiaofang Zhou, Yingxia Shao, Hao Wang, and Shazia Sadiq. 2015. Joint modeling of user check-in behaviors for point-of-interest recommendation. In *CIKM*. 1631–1640.

[48] Lei Zhang and Bing Liu. 2014. Aspect and entity extraction for opinion mining. In *Data mining and knowledge discovery for big data*. 1–40.

[49] Mingyue Zhang, Xuan Wei, Xunhua Guo, Guoqing Chen, and Qiang Wei. 2019. Identifying Complements and Substitutes of Products: A Neural Network Framework Based on Product Embedding. *TKDD* 13, 3 (2019), 34.

[50] Shijie Zhang, Hongzhi Yin, Qinyong Wang, Tong Chen, Hongxu Chen, and Quoc Viet Hung Nguyen. 2019. Inferring substitutable products with deep network embedding. *IJCAI* (2019), 4306–4312.

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

[52] 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.

[53] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*. 1027–1030.

[54] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. 2014. Bundle recommendation in ecommerce. In *SIGIR*. 657–666.