Personalized Outfit Recommendation with Learnable Anchors

Zhi Lu[†] Yang Hu[§] Yan Chen [§] [*] Bing Zeng[†]
[†]University of Electronic Science and Technology of China
[§]University of Science and Technology of China
zhilu@std.uestc.edu.cn, {eeyhu,eecyan}@ustc.edu.cn, eezeng@uestc.edu.cn

Abstract

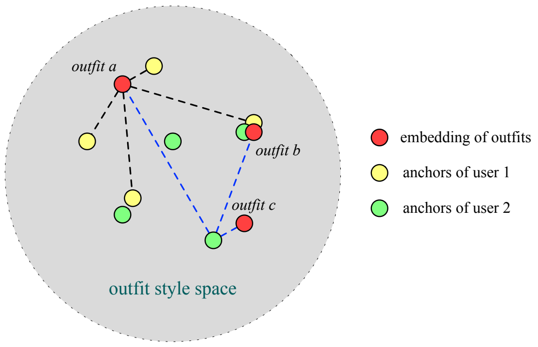
The multimedia community has recently seen a tremendous surge of interest in the fashion recommendation problem. A lot of efforts have been made to model the compatibility between fashion items. Some have also studied users' personal preferences for the outfits. There is, however, another difficulty in the task that hasn't been dealt with carefully by previous work. Users that are new to the system usually only have several (less than 5) outfits available for learning. With such a limited number of training examples, it is challenging to model the user's preferences reliably. In this work, we propose a new solution for personalized outfit recommendation that is capable of handling this case. We use a stacked self-attention mechanism to model the high-order interactions among the items. We then embed the items in an outfit into a single compact representation within the outfit space. To accommodate the variety of users' preferences, we characterize each user with a set of anchors, i.e. a group of learnable latent vectors in the outfit space that are the representatives of the outfits the user likes. We also learn a set of general anchors to model the general preference shared by all users. Based on this representation of the outfits and the users, we propose a simple but effective strategy for the new user profiling tasks. Extensive experiments on large scale real-world datasets demonstrate the performance of our proposed method.

1. Introduction

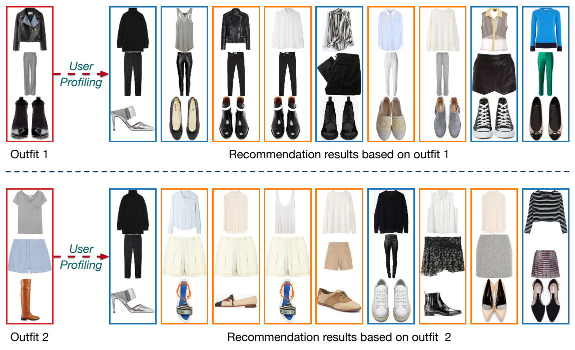
Personalized outfit recommendation is the task of predicting whether a set of fashion items that constitute an outfit are well matched and whether they fit the taste of a specific user. It has wide applications in fashion oriented social networks and online shopping. However, mining the compatibility relationships between items and capturing the diverse fashion preferences of different users can be very challenging. Compared to visual similarity, both the compatibility and the personal preference relations are more implicit and subtle.

Many previous works have studied the compatibility between a pair of fashion items [22, 36]. When it comes to outfits that consist of multiple items, some works decom-

[*]The corresponding author is Yang Hu. This work was supported by the National Natural Science Foundation of China (61602090).



(a)


User Profiling
Outfit 1
Recommendation results based on outfit 1
User Profiling
Outfit 2
Recommendation results based on outfit 2

(b)

Figure 1: (a) Outfit style space. Each user is represented by a set of anchors in the outfit style space. (b) Examples of recommended results for new users with only one historical outfit. The outfits in orange boxes are positive outfits and those in blue boxes are negative outfits. Our method recommends compatible outfits that fit the taste of a user even with very few learning examples.

pose the outfit into a set of pairwise interactions and resort to the pairwise approaches [34, 32]. This strategy, however, fails to capture the high-order relations among the items. To address this limitation, some holistic approaches [6, 18, 33] have been proposed. For example, Han el at. [6] take the items in an outfit as a sequence and use recurrent networks such as vanilla RNN and LSTM to model the compatibility of an outfit. These methods are usually not permutation invariant and the results may be sensitive to the order of the fashion items.

Furthermore, a user's personal taste is also critical for outfit recommendation. The outfits users like may have a variety of styles. Some previous works [14, 20, 31] have shown that by incorporating user's preference, the outfit recommending performance can be greatly improved. A widespread difficulty for personalized recommendation problems is that the number of historical samples for each user is usually very small [21]. To address this limitation, researchers exploit collaborative learning techniques that utilize rating data of other users to augment the data of current user. However, previous methods for personalized outfit recommendation usually only combine data of different users to model the content of fashion items. The parameters that characterize users' preferences are still learnt only with each user's own data. It would be difficult to learn these parameters well when the number of corresponding training outfits is small. The situation becomes even worse for new users that just join in and only have very few outfits rated. This cold-start problem has not been carefully studied by existing works on outfit recommendation.

In this work, we treat outfit compatibility prediction as a problem with set-input [17, 41] and use the self-attention mechanism [35] to model the interactions among fashion items. As illustrated in Fig. 1a, we represent each outfit as a point in the outfit style space. This outfit representation is compact, permutation invariant, and able to capture the high-order relation among the items. And to better capture the diverse tastes of users, we introduce a learnable personalized anchor embedding (LPAE) approach. We model the style preference of a user with a set of anchors, i.e. a group of learnable latent vectors in the outfit style space. We also learn a set of general anchors to model the general preference shared by all users. Based on this representation of the outfits and the users, the user's preference for an outfit is measured by the anchor-outfit similarity. Similarly, for new user modeling, the user's profile is learnt by assembling existing anchors based on the anchor-outfit similarity, which is effective when training data is scarce. We show two examples of the new user (cold start) problem [21] in Fig. 1b where each user only provides one outfit as reference. Since the anchor-outfit similarity is learnt in a metric-learning [13, 15] manner, it can serve as a bridge to measure the outfit-to-outfit and user-to-user relations. This similarity propagation process can introduce a certain degree of interpretability into the recommendation results.

2. Related Works

Fashion-related problems have been extensively studied in recent years [3, 19, 28]. We mainly focus on the fashion recommendation tasks. The fashion recommendation task, which is based on fashion compatibility learning, is to predict whether a set of fashion items are well matched. In personalized fashion recommendation, not only should the recommended items be compatible with each other, but the outfit they constitute should also fit the taste of a specific user. We discuss recent developments in fashion compatibility learning and introduce emerging applications built on compatibility learning.

2.1. Fashion compatibility learning

Existing studies on fashion compatibility learning can be divided into two categories: item-level and outfit-level approaches. The item-level approaches mainly model the compatibility between two fashion items. The pairwise interaction decomposition can be seen as an explicit prior structure [26] imposed on the outfit compatibility. In [37, 22], a single latent space was used for measuring the compatibility. However, the relationships between a pair of items in different aspects (e.g. color, pattern, category) can be quite different or even contradictory. To address the limitation of using a single latent space, Veit et al. [36] proposed the conditional similarity networks (CSN) that compared different items in different conditions to improve the performance. Tan et al. [32] improved the CSN method by learning the multiple conditional embeddings. They also used attention mechanism to discover the relative importance of different conditions. Another way to enhance the item-item compatibility prediction is to learn an image embedding that respects item type [34]. In this method, each item pair are compared in the conditional subspaces to improve the performance. For personalized outfit recommendation, Hu [14] et al. used additional user-item pairs to model the user's preference towards items. They used inner product instead of metric learning to measure the compatibility between pairs. Song et al. [31] also considered the personalized outfit recommendation problem. They characterized the item-item and user-item interactions using a matrix factorization method. When there are links between items, a graph-based approach can be used to improve the performance [4, 27]. Furthermore, Lu et al. [20] tackled the efficiency problem by learning hash codes for both users and items.

The outfit-level approaches treat the outfit as a whole and model the high-order interactions among the items. Tangseng et al. [33] learnt outfit embedding by concatenating all item features together. Han et al. [6] treated the outfit as an item sequence and learnt the compatibility with bidirectional LSTM. Although these methods learnt the compatibility in outfit-level, they did not handle the outfit in a permutation-invariant setting and the results may be sensitive to the order of items. Li et al. [18] proposed an instances pooling method to aggregate the item features to predict the outfit quality. They use RNN to do instance pooling to imporve the performance. The personalized outfit generation (POG) model in [2] was an approach for the outfit generation task. They treated the task as translation from an incomplete outfits to the remaining item and used the *Transformer* architecture [35] without positional encoding for outfit generation and recommendation. Different from the item-level approaches, the outfit-level approaches model outfit compatibility in an inexplicit way.

To extend the fashion recommendation system with side information, Yang et al. [39] proposed a translation-based network that learnt the compatibility with the category-specific relations. Song et al. studied the compatibility problem in the setting of the heterogeneous multi-modal data [30] and used rich fashion domain knowledge [29].

2.2. Emerging applications

The compatibility learning has been studied in different scenarios to solve practical problems. We conduct a brief review on a few of recent works. To interpret the compatibility score, Wang et al. [38] explored the problem of diagnosing the outfit compatibility on items by utilizing the gradient information. Several recent works studied the compatibility problems considering the body shapes [9, 11, 8]. Hsiao et al. [12] propose Fashion++ to make minimal adjustments to a full-body outfit that have a maximal improvement on its fashionability. The task of creating a set of outfits from personal wardrobes that are maximally compatible and versatile has also been proposed recently [10]. Dong et al. [5] extended this work with personalized information. Yu et al. [40] considered fashion compatibly in the fashion synthesizing problem.

3. Approach

Our learnable personalized anchor embedding (LPAE) model, as shown in Fig. 2, contains two components: the item aggregation network and the matching network. The item aggregation network encodes a set of items into a compact embedding in the outfit style space. The matching network obtains the user preference score by computing the similarities between an outfit embedding and a set of learnable personalized anchor embeddings.

3.1. Item aggregation model

An outfit, which consists of n items from different categories, is represented as a packet of features $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]^\intercal \in \mathbb{R}^{n\times d}$ where $\boldsymbol{x}_k \in \mathbb{R}^d$ is the feature representation of the k-th item and d is the dimension of item features. The item aggregation network is a transformation $f : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ that encodes n items $\mathbf{X}$ into an outfit feature $\boldsymbol{z} \in \mathbb{R}^d$ with a fixed length. The transformation f should take into account the compatibility relations between the fashion items. Also, since the order of the items in $\mathbf{X}$ should have no effect on the final prediction, the transformation function f should be permutation invariant. Inspired by the recent work [17] on problems with set-input, we build the outfit embedding model using the self-attention mechanism.

3.1.1 Multi-head attention

We first introduce the multi-head attention mechanism, a basic component for updating item features and feature aggregation. Given a set of n query vectors $\mathbf{Q} \in \mathbb{R}^{n\times d}$, an attention function [35] updates them via m key-value pairs $\mathbf{K} \in \mathbb{R}^{m\times d}, \mathbf{V} \in \mathbb{R}^{m\times d}$. For brevity, we set the dimension of all vectors to be the same as the query vectors. Each output vector is a weighted sum of the value vectors in $\mathbf{V}$ with the weights being computed by the inner product between the query and the key vectors in $\mathbf{K}$.

$$\mathbf{O} = \mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d}})\mathbf{V} \tag{1}$$

The multi-head attention extends the attention function by projecting the triplet $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ into h subspaces and concatenating the outputs computed by the attention function (Eq. (1)) on each subspace. h is the number of heads.

$$\begin{aligned} \mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \mathrm{Concat}\left(\mathbf{O}_1, \cdots, \mathbf{O}_h\right)\mathbf{W}^O \\ \text{where } \mathbf{O}_i &= \mathrm{Attn}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{2}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d\times d_k}, \mathbf{W}_i^K \in \mathbb{R}^{d\times d_k}, \mathbf{W}_i^V \in \mathbb{R}^{d\times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_v\times d_o}$ are projection parameters. The choice for the dimensions in this paper is $d_k = d_v = d/h$ and $d_o = d$.

3.1.2 Outfit self-attention

Given the features $\mathbf{X} \in \mathbb{R}^{n\times d}$ of the n items in an outfit, we compute the self-attentive output for them with multi-head attention as follows:

$$\begin{aligned} \mathrm{SelfAttn}(\mathbf{X}) &= \mathrm{LayerNorm}(\mathbf{H} + \sigma(\mathbf{H})) \\ \text{where } \mathbf{H} &= \mathrm{LayerNorm}\left(\mathbf{X} + \mathrm{MultiHead}(\mathbf{X}, \mathbf{X}, \mathbf{X})\right) \end{aligned} \tag{3}$$

$\sigma(\cdot)$ is a row-wise feedforward layer and $\mathrm{LayerNorm}(\cdot)$ is the layer normalization [1].

By applying the $\mathrm{SelfAttn}(\cdot)$ function to $\mathbf{X}$, we augment existing pairwise methods for fashion outfit by relating each item to all other items in the outfit concurrently. Following [17], we stack the self-attention function (see Fig. 2) to model the high-order relationships between items.

$$\mathbf{F} = \mathrm{SelfAttn}(\mathrm{SelfAttn}(\mathbf{X})) \in \mathbb{R}^{n\times d} \tag{4}$$

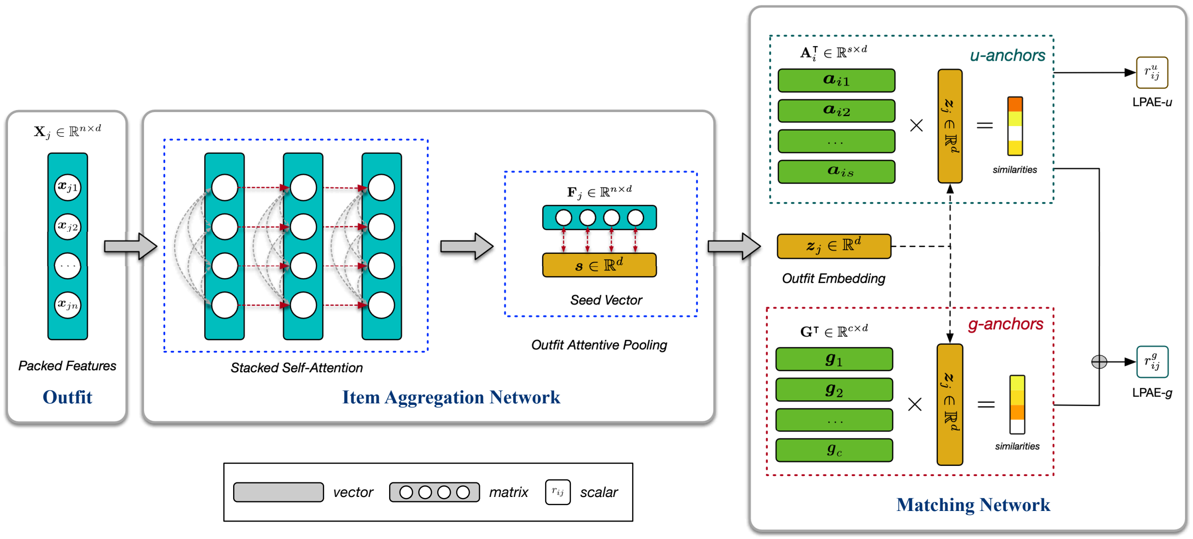


Figure 2: The architecture of our model. We first model the interactions between the items in an outfit with a stacked self-attention module. The items are then aggregated into a single outfit embedding using the outfit attentive pooling operation. The fashion preference of each user is characterized by a set of *u-anchors*. We also learn a group of *g-anchors* to describe the general preference. The LPAE-*u* model only uses the *u-anchors* for compatibility prediction and the LPAE-*g* model uses both *u-anchors* and *g-anchors* to compute the compatibility score.

3.1.3 Outfit attentive pooling

To get a fixed length outfit representation regardless of the number of items in it, we use a vector $\boldsymbol{s} \in \mathbb{R}^d$ to aggregate the output as follows:

$$\begin{aligned} \boldsymbol{z} &= \text{LayerNorm}(\boldsymbol{h} + \sigma(\boldsymbol{h})) \\ \text{where } \boldsymbol{h} &= \text{LayerNorm}\left(\boldsymbol{s} + \text{MultiHead}(\boldsymbol{s}, \mathbf{F}, \mathbf{F})\right) \end{aligned} \tag{5}$$

where $\boldsymbol{s}$ is learnt as a model parameter. The outfit attentive pooling produces a single compact vector $\boldsymbol{z} \in \mathbb{R}^d$ for an outfit. It can be easily seen that the transformation from n items $\mathbf{X}$ to a single outfit embedding $\boldsymbol{z}$ is invariant to the permutation of items.

3.2. Matching network

The outfits a user likes may be very variable in style. Although it is rare for two users to have exactly the same fashion taste, they may share a common view towards some specific styles. This makes it inadequate to use a single style vector to characterize a user. We therefore use multiple vectors to describe the fashion preferences of users. Each user is represented as a set of anchors scattered around the outfit space (see Fig. 1a). The number of anchors is set as a hyper-parameter.

Formally, we use $\mathcal{A} = \{\mathbf{A}_i\}_{i=1}^m$ to represent groups of anchors for m users, where $\mathbf{A}_i = [\boldsymbol{a}_{i1}, \dots, \boldsymbol{a}_{is}] \in \mathbb{R}^{d\times s}$ contains the anchors for the i-th user and $\boldsymbol{a}_{ik} \in \mathbb{R}^d$ is the k-th anchor in it. Each user is characterized by s learnable anchors in the outfit space. We call them the *u-anchors* and refer to the model that is built on them as LPAE-*u* as shown in Fig. 2. Given an outfit embedding $\boldsymbol{z}_j$, the preference score the i-th user has for it is obtained by computing the similarities between $\boldsymbol{z}_j$ the all *u-anchors* in $\mathbf{A}_i$ as follows:

$$\begin{aligned} r_{ij}^u &= \frac{1}{s}\sum_{k=1}^{s} \boldsymbol{a}_{ik}^\intercal \boldsymbol{z}_j, \\ s.t.\ \|\boldsymbol{a}_{ik}\|_2 &= \|\boldsymbol{z}_j\|_2 = 1, \end{aligned} \tag{6}$$

General anchors. The LPAE-*u* model only depends on the user specific parameters *u-anchors*. It requires sufficient user feedback to learn the *u-anchors* reliably. In the new user problem, due to the small number of outfits that are available for training, these user parameters may not be sufficiently learnt. To address this problem and to improve the performance in the cold start scenario, we add a general compatibility by defining the general anchors which are shared by all users. The general score indicates the overall tendency users have for an outfit. For user i, the anchor set is extended to:

$$\mathbf{A}_i^g = [\mathbf{A}_i, \mathbf{G}] \tag{7}$$

where $\mathbf{A}_i \in \mathbb{R}^{d\times s_1}$ contains the u-anchors for user i and $\mathbf{G} \in \mathbb{R}^{d\times s_2}$ is shared by all users. We call the elements in $\mathbf{G}$ the *g-anchors* and refer to the model that includes them as LPAE-g. The resulting preference score is computed as follows:

$$r_{ij}^{g} = \frac{1}{s_1}\sum_{k=1}^{s_1} \boldsymbol{a}_{ik}^{\intercal}\boldsymbol{z}_j + \frac{1}{s_2}\sum_{k=1}^{s_2} \boldsymbol{g}_{k}^{\intercal}\boldsymbol{z}_j, \quad s.t.\ \|\boldsymbol{a}_{ik}\|_2 = \|\boldsymbol{g}_k\|_2 = \|\boldsymbol{z}_j\|_2 = 1 \tag{8}$$

where $\boldsymbol{g}_k \in \mathbb{R}^d$ is the k-th general anchor in $\mathbf{G}$. We also show the LPAE-g model in Fig. 2.

We can expect that with enough feedback, the user preference can be captured well without *g-anchors*, so in this case LPAE-u and LPAE-g will have similar performance. However, when the training data is not enough for new users, the g-anchors can notably improve the results. And by further analyzing on the results of LPAE-u and LPAE-g, we can estimate the minimal number of outfits needed to capture a user's preference, i.e. on what level the recommendation results deviate from the common preference and mainly depend on user's past selections.

3.3. Objective function

3.3.1 Orthogonal regularization

To avoid the collapse of anchor embeddings and learn discriminative representatives for outfit contents, we use the log-determinant divergence (LDD) [16] to regularize each anchor matrix. Suppose $\mathbf{L} \in \mathbb{R}^{s\times d}$ is one of the anchor matrices, and the LDD is computed as follows:

$$D(\mathbf{L}\mathbf{L}^{\intercal}, \mathbf{I}) = \mathrm{tr}(\mathbf{L}\mathbf{L}^{\intercal}) - \log\det(\mathbf{L}\mathbf{L}^{\intercal}) - s \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{s\times s}$ is identity matrix.

3.3.2 Overall loss function

Suppose the outfits posted by each user is $\mathcal{Z}_i^+$. Since the dataset usually contains implicit feedback, we sample a set of outfits $\mathcal{Z}_i^-$ to be the negative samples for user i. The training set $\mathcal{P}$ contains a set of outfit pairs:

$$\mathcal{P} \equiv \{(i,j,k) | r_{ij} > r_{ik}, \forall \boldsymbol{z}_j \in \mathcal{Z}_i^+, \boldsymbol{z}_k l \in \mathcal{Z}_i^-\} \tag{10}$$

where (i,j,k) indicates that user i prefers outfit j over outfit k. We eliminate the superscript in r_{ij} for conciseness. The Bayesian personalized ranking (BPR) [25] criteria is used to train our models. And the overall loss function is defined as follows:

$$\ell = \sum_{(i,j,k)\in\mathcal{P}} \log\left(1 + \exp(-(r_{ij} - r_{ik})\right) + \lambda \cdot \ell_{reg} \tag{11}$$

where ℓ_{reg} is the LDD regularizer in Eq. (9).

4. New user profiling task

The preference scores defined in Eq. (6) and Eq. (8) are based on anchor-outfit similarities which pull the outfits a user likes close to his/her anchors. And users who have similar tastes get parts of their anchors embedded closely. Hence, the *u-anchors* set $\mathcal{A}$ can be used for new user profiling task by assembling anchors which are close to new user's outfits.

We define the anchor pool $\mathcal{Q} = \{\boldsymbol{q}_l\} = \bigcup_{i,k} \boldsymbol{a}_{ik}$ to be the union of all *u-anchors* and $\mathcal{Z} = \{\boldsymbol{z}_j\}$ to be the set of outfit embeddings of a given new user. A common approach to do new user profiling is to re-train the user-specific parameters $\mathbf{P} \in \mathbb{R}^{d\times s}$ [20]. Since the size of $\mathcal{Z}$ is usually small, $\mathbf{P}$ learnt using this way is easy to overfit it. In this section, we propose two learning by search strategies for new users that avoid direct parameter optimization. The strategies for learning $\mathbf{P}$ are summarized as follows:

- *Learning by optimization (SO)*. We directly learn the user parameters $\mathbf{P}$ by optimization with all other parameters fixed as in [20].
- *Learning by user-search (SU)*. We reuse $\mathbf{A}_i$ from existing users whose taste is the closest to the new user, i.e. the i-th user with which the sum of the preference scores r_{ij}^u for $\boldsymbol{z}_j$ in $\mathcal{Z}$ is the largest. Mathematically, we have

$$\arg\max_i \sum_{\boldsymbol{z}_j\in\mathcal{Z}} r_{ij}^u.$$

- *Learning by anchor-search (SA)*. We rank all anchors in $\mathcal{Q}$ according to the response score $\sum_i \boldsymbol{z}_i^{\intercal}\boldsymbol{q}_l$ and select the top s anchors to constitute $\mathbf{P}$.

For convenience, we refer to the three strategies as *SO*, *SU*, and *SA* respectively. Specially, for new users who only provide one outfit, we use all anchors with positive response scores to reduce the variance of recommendation results in strategy *SA*.

5. Experimental Design

Evaluation metrics. To evaluate the recommendation accuracy, we consider the Area Under the ROC curve (AUC) and the Normalized Discounted Cumulative Gain (NDCG) used in previous works [20, 14]. Those metrics indicate the quality of ranking a set of positive and negative outfits. The ratio between positive and negative outfits is 1:10 in the testing set, and the performance is averaged over all users.

Datasets. We consider two datasets: Polyvore-Us [20] and IQON-3000 [31]. In Polyvore-Us datasets, where U is the number of users, we use Polyvore-630/519 for evaluating the performance on personalized outfit recommendation tasks and Polyvore-53/32 for the analysis of new users. Each user has around 200 outfits for training and 40 outfits for testing, and each outfit includes 3 items from 3 different categories. For the IQON-3000 dateset, we filter out 608 users - each user has 85 outfits for training and 20 for testing. Each outfit contains 3∼8 items that belong to different categories. We further divide the users into two groups and end up in two subsets: IQON-550 and IQON-58.

Considered methods. We compare our models with several state-of-the-art methods. The Bi-LSTM model [6] uses a bidirectional LSTM to learn the compatibility of the outfits. The Type-Aware Embedding method [34] embeds pairs of items into type-specific spaces to learn the similarities. This method is an extension of Conditional Similarity Networks (CSN) [36] where each pair of items has two conditions. The SCE-Net [32] learns multiple conditional embeddings for items. The weights for each conditional embedding are computed using the attention mechanism. The FHN [20] learns the outfit compatibility and the user preference in a pairwise manner. We train the FHN without using the binarization operation for a more fair comparison.

In addition to our LPAE models, we also introduce a plain latent factor model (LFM) as a baseline. It also uses the outfit embedding as an outfit representation but uses a single vector $\boldsymbol{u}_i$ to characterize the i-th user. Given the i-th user representation $\boldsymbol{u}_i \in \mathbb{R}^d$ and j-th outfit $\boldsymbol{z}_j \in \mathbb{R}^d$, we compute the prediction score as follows:

$$r_{ij}^{\text{LFM}} = \boldsymbol{h}^\intercal a\big(\mathbf{W}(\boldsymbol{u}_i \odot \boldsymbol{z}_j) + \boldsymbol{b}\big) \tag{12}$$

where $\odot$ denotes the element-wise product of vectors, $a(\cdot)$ is a nonlinear activation function, $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}, \boldsymbol{h} \in \mathbb{R}^d$ are learnable parameters.

Implementation details. The item features are extracted from images with *ResNet*-34 [7] and used as the input for all methods for fair comparison, i.e. the backbone is not fine-tuned during training and testing. We set the latent dimension to 128 for all methods. We use $s = 64$ for our LPAE-u and $s_1 = s_2 = 32$ for LPAE-g. SGD with momentum [24] is used to train all methods. The learning rates are simply searched and reduced when the accuracy stops increasing on validation set. Our methods are implemented with PyTorch [23] and the code will be released for reproducible research.

6. Experimental Results

In this section, we compare our approaches with state-of-the-art methods and highlight our key experimental findings. More results can be found in the supplementary material.

6.1. Can the item aggregation model provide a better representation for outfits?

From the results in Table 1, we can see that our LPAE methods achieve better performances under all metrics which shows that the set modeling method in Eq. 5 is helpful for learning the outfit embeddings. And the results for both LFM and FHN are more convincing comparisons, because the main difference between the two methods lies in the way they model items.

We believe that due to the self-attention mechanism we use, the resulting outfit embedding can capture more subtle relationships between the items. And a compact outfit embedding is helpful for downstream tasks such as outfit retrieval. Since the effectiveness of the self-attention block has been discussed in the original work [17] and our key motivation is to address the cold start problem, we omit studies regarding the components of the item aggregation model here.

6.2. How important are anchors for cold start problems?

The primary motivation for our LPAE approach is to improve the cold start performance with anchors. This raises the question of how well we can learn from such limited data. Specifically, we test the performance when each new user has only 1 or 5 outfits for learning. We compare our LAPE models with non-personalized methods and report the performance in Table 2. We only use the SA strategy for LAPE since it is the most effective one in this extreme case. The results show that our LPAE methods outperform those non-personalized methods notably even with only one outfit available. Since our model reuses the existing anchors, it can perform user modeling very efficiently.

Furthermore, we perform the new user profiling task with different strategies and with different numbers of available outfits. And the results are shown in Table 3. Let N be the number of positive outfits available for each user during training, and when $N < 10$, we refer it as cold starter.For cold starters, the SA strategy outperforms the SO strategy, which shows the effectiveness of anchor embeddings. With the increase of N, the SA strategy still shows performance comparable to the optimization strategy. Compared to the FHN baseline, our LAPE model with anchor search is superior in almost all Ns.

The results of using the SU strategy are inferior to those of other strategies. This may be due to the difficulty of finding a single user who shares a very similar taste. However, it's much easier to characterize a user by reorganizing the taste representations learnt from other users, which conforms to idea of collaborative filtering.

6.3. How important are general anchors?

The general anchors learn non-personalized compatibility for outfits. By comparing the results of LPAE-u and LPAE-g in Table 2 and Table 3, we find that the general anchors play an important role for cold starters. Similar findings can be obtained when comparing LFM and FHN since FHN also uses the general scores. With the increase of N, the contribution of *g-anchors* starts to diminish. An inter-

	Polyvore-519		Polyvore-630		IQON-550	
Method	AUC	NDCG	AUC	NDCG	AUC	NDCG
Bi-LSTM [6]	0.7882	0.6306	0.7840	0.6328	0.8133	0.6678
Type-Aware [34]	0.7840	0.6162	0.7727	0.6023	0.7955	0.6427
SCE-Net [32]	0.7849	0.6167	0.7966	0.6524	0.8228	0.6823
FHN [20]	0.8992	0.8097	0.8701	0.7640	0.8765	0.7650
LFM	0.9104	0.8312	0.8872	0.7973	0.9161	0.8496
LPAE-u	**0.9119**	**0.8359**	**0.9006**	**0.8186**	**0.9275**	**0.8727**
LPAE-g	0.9119	0.8359	0.8995	0.8167	0.9253	0.8703

Table 1: Comparison of different methods on recommendation tasks.

	Polyvore-32		Polyvore-53		IQON-58	
# Outfits	1	5	1	5	1	5
Bi-LSTM [6]	0.8087	0.8087	0.7704	0.7704	0.7979	0.7979
Type-Aware [34]	0.7782	0.7782	0.7532	0.7532	0.7636	0.7636
SCE-Net [32]	0.7779	0.7779	0.7645	0.7645	0.7791	0.7791
LPAE-u (SA)	0.8383	0.8542	0.7894	0.8144	0.8514	0.8478
LPAE-g (SA)	**0.8410**	**0.8511**	**0.7938**	**0.8299**	**0.8775**	**0.8904**

Table 2: Comparison of different methods on new user tasks.

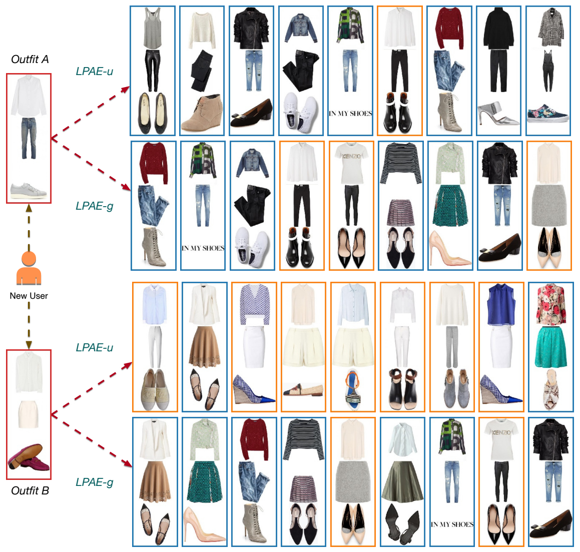


Figure 3: Examples for the new user problem. The orange box indicates the positive outfit and the blue box indicates the negative outfit.

esting finding is that, at around $N = 50$, the performance of LPAE-u and LPAE-g starts to become similar. Meanwhile, the LFM model begins to beat FHN. We guess that to properly capture a user's preference, around 50 samples are needed in this dataset.

Visualization. We visualize the recommendation results for the new user profiling task in Fig. 3, where the number of available outfits is 1. Outfits A and B are two example input outfits for the new user. We show the corresponding recommendation results for each outfit. For LPAE-u, since there is no general anchors, the accuracy largely depends on whether the outfit provided is the dominant style in testing data. For outfit A, it is the minority style of the user, and LPAE-g can play safer with the help of general anchors. For outfit B, LPAE-u works much better than LPAE-g since the user has many outfits similar to outfit B. Both methods learn the user preferences via anchor-search and show results relevant to the provided outfits.

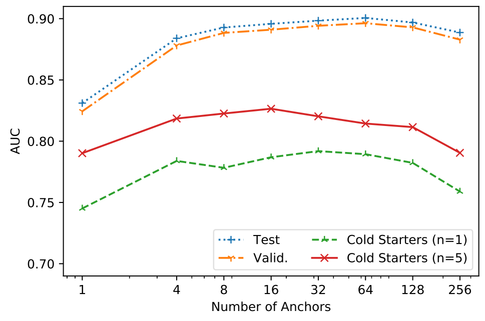


Figure 4: Performance for LPAE-u with different number of anchors. We also show the performance for cold starters under the SA strategy.

Methods	1	5	10	20	50	100
FHN (SO)	.7274 ± .0032	.7776 ± .0022	.7989 ± .0017	.8241 ± .0013	.8442 ± .0012	.8552 ± .0004
LFM (SO)	.6252 ± .0049	.7351 ± .0038	.7840 ± .0017	.8113 ± .0021	.8470 ± .0013	.8643 ± .0007
LPAE-*u* (SU)	.7467 ± .0024	.7917 ± .0025	.8073 ± .0018	.8159 ± .0009	.8258 ± .0013	.8252 ± .0011
LPAE-*u* (SA)	.7894 ± .0011	.8144 ± .0017	.8332 ± .0020	.8470 ± .0008	.8594 ± .0005	.8623 ± .0004
LPAE-*u* (SO)	.6886 ± .0039	.7897 ± .0016	.8281 ± .0020	.8501 ± .0013	.8701 ± .0006	.8783 ± .0005
LPAE-*g* (SU)	.7622 ± .0031	.8106 ± .0014	.8214 ± .0014	.8303 ± .0011	.8382 ± .0014	.8407 ± .0006
LPAE-*g* (SA)	**.7938** ± .0009	**.8299** ± .0017	**.8433** ± .0016	.8530 ± .0009	.8602 ± .0006	.8627 ± .0004
LPAE-*g* (SO)	.7840 ± .0023	.8195 ± .0015	.8431 ± .0011	**.8572** ± .0008	**.8725** ± .0006	**.8812** ± .0003

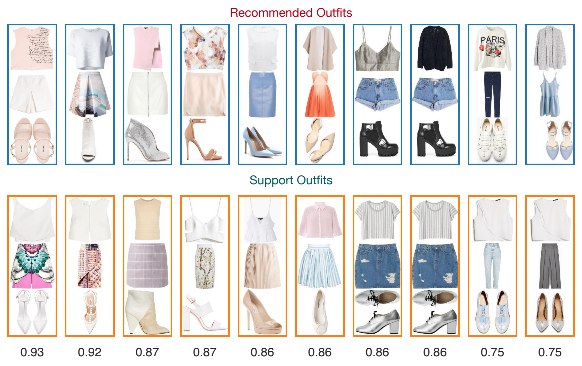
Table 3: The AUC with different number of positive training outfits for the new user profiling task on Polyvore-53. We run experiments 10 times and report the mean and standard deviation for different methods and different strategies. SO, SA and SU are three strategies to model the preferences of new users.

6.4. Are multiple anchors useful?

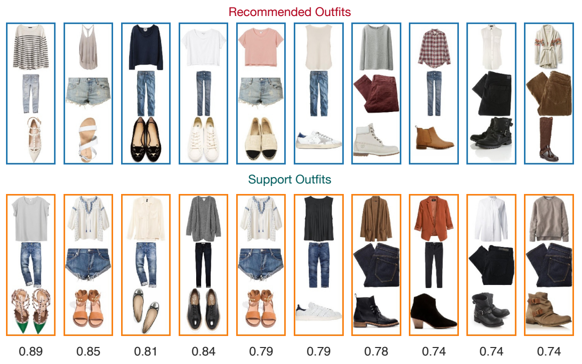
To show the impact of the number of anchors s, we trained LPAE-*u* with different numbers of anchors. The results are shown in Fig. 4. When the number of anchors s is 1, the LPAE model tries to cluster outfit embeddings into one class for each user. Since mapping all of the various outfits into one class can lose a lot of information, the performance is poor as expected. As s increases, the diverse tastes of users can be properly captured with multiple anchors, and the performance is significantly improved. As s continues to increase, the number of anchors becomes much larger than the optimal number of clusters required, and the performance begins to decline. To give an extreme example, when $s = 256$, the number of anchors exceeds the number of training outfits, and the performance drops dramatically.

6.5. Are the learnt outfit embeddings helpful to interpret the results?

For each recommended outfit, we can retrieve similar outfits from the user's past selections to support the recommendation results. Since other baseline methods do not establish the similarity measurements between outfits, we only show examples of ours in Fig. 5. Outfits at the top are the recommended ones, and outfits at the bottom are support outfits from the training set. We also show the cosine similarities between the recommended ones and support ones. As we can see, the similar visual style of the support outfits do help to interpret the recommendation results.



(a) User 1



(b) User 2

Figure 5: The outfit retrieval results.

7. Conclusion

In this paper, we propose a learnable personalized anchor embedding (LPAE) approach for personalized outfit recommendation and new user profiling. Our models encode the outfit into a compact embedding using the self-attention mechanism to capture the high-order relationships among fashion items. We model the fashion preference of each user with a set of anchors and compute the preference score based on the similarity between the outfit embedding to the user's anchors. The outfits are encoded in the outfit style space and the outfits with similar styles can be retrieved via distance in the space. With the proposed LPAE framework, we can characterize a new user's preference by simply reusing the existing anchors and achieve notably better performance when the data is scarce. Our methods outperform the state-of-the-art methods in both an ordinary setting and a cold start setting.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv*, 2016. 3

[2] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. In *KDD*, 2019. 3

[3] Wen-Huang Cheng, Sijie Song, Chieh-Yun Chen, Shintami Chusnul Hidayati, and Jiaying Liu. Fashion Meets Computer Vision: A Survey. *arXiv*, 2020. 2

[4] Guillem Cucurull, Perouz Taslakian, and David Vazquez. Context-Aware Visual Compatibility Prediction. In *CVPR*, 2019. 2

[5] Xue Dong, Xuemeng Song, Fuli Feng, Peiguang Jing, Xin-Shun Xu, and Liqiang Nie. Personalized Capsule Wardrobe Creation with Garment and User Modeling. In *ACM MM*, 2019. 3

[6] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning Fashion Compatibility with Bidirectional LSTMs. In *ACM MM*, 2017. 2, 6, 7

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 6

[8] Shintami Chusnul Hidayati, Ting Wei Goh, Ji-Sheng Gary Chan, Cheng-Chun Hsu, John See, Wong Lai Kuan, Kai-Lung Hua, Yu Tsao, and Wen-Huang Cheng. Dress with Style: Learning Style from Joint Deep Embedding of Clothing Styles and Body Shapes. *TMM*, 2020. 3

[9] Shintami Chusnul Hidayati, Cheng-Chun Hsu, Yu-Ting Chang, Kai-Lung Hua, Jianlong Fu, and Wen-Huang Cheng. What Dress Fits Me Best? Fashion Recommendation on the Clothing Style for Personal Body Shape. In *ACM MM*, 2018. 3

[10] Wei-Lin Hsiao and Kristen Grauman. Creating Capsule Wardrobes from Fashion Images. In *CVPR*, 2018. 3

[11] Wei-Lin Hsiao and Kristen Grauman. Dressing for Diverse Body Shapes. *arXiv*, 2019. 3

[12] Wei-Lin Hsiao, Isay Katsman, Chao-Yuan Wu, Devi Parikh, and Kristen Grauman. Fashion++: Minimal Edits for Outfit Improvement. In *ICCV*, 2019. 3

[13] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative Metric Learning. In *WWW*, 2017. 2

[14] Yang Hu, Xi Yi, and Larry S Davis. Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach. In *ACM MM*, 2015. 2, 5

[15] Brian Kulis. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning*, 5, 2013. 2

[16] Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-Rank Kernel Learning with Bregman Matrix Divergences. *JMLR*, 10(13), 2009. 5

[17] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*, 2019. 2, 3, 6

[18] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. Mining Fashion Outfit Composition Using an End-to-End Deep Learning Approach on Set Data. *TMM*, 19(8), 2017. 2, 3

[19] Si Liu, Luoqi Liu, and Shuicheng Yan. Fashion Analysis: Current Techniques and Future Directions. *IEEE MultiMedia*, 21(2), 2014. 2

[20] Zhi Lu, Yang Hu, Yunchao Jiang, Yan Chen, and Bing Zeng. Learning Binary Code for Personalized Fashion Recommendation. In *CVPR*, 2019. 2, 5, 6, 7

[21] Paolo Massa and Paolo Avesani. Trust-Aware Collaborative Filtering for Recommender Systems. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2004. 2

[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*, 2015. 1, 2

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca Antiga. Pytorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. 6

[24] Ning Qian. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks*, 12(1), 1999. 6

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, 2009. 5

[26] Steffen Rendle and Lars Schmidt-Thieme. Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In *WSDM*, 2010. 2

[27] Anirudh Singhal, Ayush Chopra, Kumar Ayush, Utkarsh Patel Govind, and Balaji Krishnamurthy. Towards a Unified Framework for Visual Compatibility Prediction. In *WACV*, 2020. 2

[28] Sijie Song and Tao Mei. When Multimedia Meets Fashion. *IEEE MultiMedia*, 25(3), 2018. 2

[29] Xuemeng Song, Fuli Feng, Xianjing Han, Xin Yang, Wei Liu, and Liqiang Nie. Neural Compatibility Modeling with Attentive Knowledge Distillation. In *SIGIR*, 2018. 3

[30] Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma. NeuroStylist: Neural Compatibility Modeling for Clothing Matching. In *ACM MM*, 2017. 3

[31] Xuemeng Song, Xianjing Han, Yunkai Li, Jingyuan Chen, Xin-Shun Xu, and Liqiang Nie. GP-BPR: Personalized Compatibility Modeling for Clothing Matching. In *ACM MM*, 2019. 2, 5

[32] Reuben Tan, Mariya I Vasileva, Kate Saenko, and Bryan A Plummer. Learning Similarity Conditions Without Explicit Supervision. In *ICCV*, 2019. 2, 6, 7

[33] Pongsate Tangseng, Kota Yamaguchi, and Takayuki Okatani. Recommending Outfits from Personal Closet. In *ICCV*, 2017. 2

[34] Mariya I Vasileva, Bryan A Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David A Forsyth. Learning Type-Aware Embeddings for Fashion Compatibility. In *ECCV*, 2018. 2, 6, 7

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017. 2, 3

[36] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional Similarity Networks. In *CVPR*, 2017. 1, 2, 6

[37] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning Visual Clothing Style with Heterogeneous Dyadic Co-Occurrences. In *ICCV*, 2015. 2

[38] Xin Wang, Bo Wu, and Yueqi Zhong. Outfit Compatibility Prediction and Diagnosis with Multi-Layered Comparison Network. In *ACM MM*, 2019. 3

[39] Xun Yang, Yunshan Ma, Lizi Liao, Meng Wang, and Tat-Seng Chua. TransNCFM: Translation-Based Neural Fashion Compatibility Modeling. In *AAAI*, 2019. 3

[40] Cong Yu, Yang Hu, Yan Chen, and Bing Zeng. Personalized Fashion Design. In *ICCV*, 2019. 3

[41] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *NeurIPS*, 2017. 2