Aspect Re-distribution for Learning Better Item Embeddings in Sequential Recommendation

Wei Cai
cai_wei@zju.edu.cn
Zhejiang University
Hangzhou, China

Weike Pan
panweike@szu.edu.cn
Shenzhen University
Shenzhen, China

Jingwen Mao
jingwenmao@zju.edu.cn
Zhejiang University
Hangzhou, China

Zhechao Yu
22121240@zju.edu.cn
Zhejiang University
Hangzhou, China

Congfu Xu*
xucongfu@zju.edu.cn
Zhejiang University
Hangzhou, China

ABSTRACT

Sequential recommendation has attracted a lot of attention from both academia and industry. Since item embeddings directly affect the recommendation results, their learning process is very important. However, most existing sequential models may introduce bias when updating the item embeddings. For example, in a sequence where all items are endorsed by a same celebrity, the co-occurrence of two items only indicates their similarity in terms of endorser, and is independent of the other aspects such as category and color. The existing models often update the entire item as a whole or update different aspects of the item without distinction, which fails to capture the contributions of different aspects to the co-occurrence pattern. To overcome the above limitations, we propose aspect re-distribution (ARD) to focus on updating the aspects that are important for co-occurrence. Specifically, we represent an item using several aspect embeddings with the same initial importance. We then re-calculate the importance of each aspect according to the other items in the sequence. Finally, we aggregate these aspect embeddings into a single aspect-aware embedding according to their importance. The aspect-aware embedding can be provided as input to a successor sequential model. Updates of the aspect-aware embedding are passed back to the aspect embeddings based on their importance. Therefore, different from the existing models, our method pays more attention to updating the important aspects. In our experiments, we choose self-attention networks as the successor model. The experimental results on four real-world datasets indicate that our method achieves very promising performance in comparison with seven state-of-the-art models.

CCS CONCEPTS

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

*Corresponding author.

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9278-5/22/09…$15.00
https://doi.org/10.1145/3523227.3546764

KEYWORDS

sequential recommendation, aspect re-distribution, aspect-aware item embedding

ACM Reference Format:
Wei Cai, Weike Pan, Jingwen Mao, Zhechao Yu, and Congfu Xu. 2022. Aspect Re-distribution for Learning Better Item Embeddings in Sequential Recommendation. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3523227.3546764

1 INTRODUCTION

Recommender systems have been widely applied on various online platforms [3, 8]. Some recent works focus on sequential recommendation [14, 31, 35]. Different from traditional recommendation, sequential recommendation takes into account the time of feedback. Sequential recommendation models often sort each user's feedback (e.g., "clicks" and "likes") in chronological order as a sequence and predict the next item of each given sequence. In sequential models, item embeddings directly affect the recommendation results [38]. Therefore, it is important to learn the item embeddings accurately and carefully.

However, the learning of item embeddings in existing models may introduce bias in some common situations. We show a training sample in Figure 1 (a), where i_1, i_2 and i_3 are all endorsed by Taylor Swift. For the item i_2 in the sequence and the positive item i_3, existing models [14, 31] draw their embeddings closer in a high-dimensional space as in Figure 1 (b). Some recent works [16, 21, 34, 41] represent an item with several embeddings. Similarly, they make the embeddings of the two items close to each other as in Figure 1 (c). Nevertheless, recalling the training sample, it can be inferred from the sequence that both items i_2 and i_3 appear in the sequence because the user likes Taylor Swift. In other words, their co-occurrence is only because of the endorser and not related to the other aspects. Therefore, it is more accurate to update the important aspects of an item, leaving the other aspects unchanged, as shown in the Figure 1 (d). In contrast, existing models either update the entire item as a whole, or update different aspects of the item without distinction. They do not capture the reason why the items are co-occurring.

In order to focus on updating the aspects that are important, we propose an aspect re-distribution (ARD) method. Our method consists of the following steps:

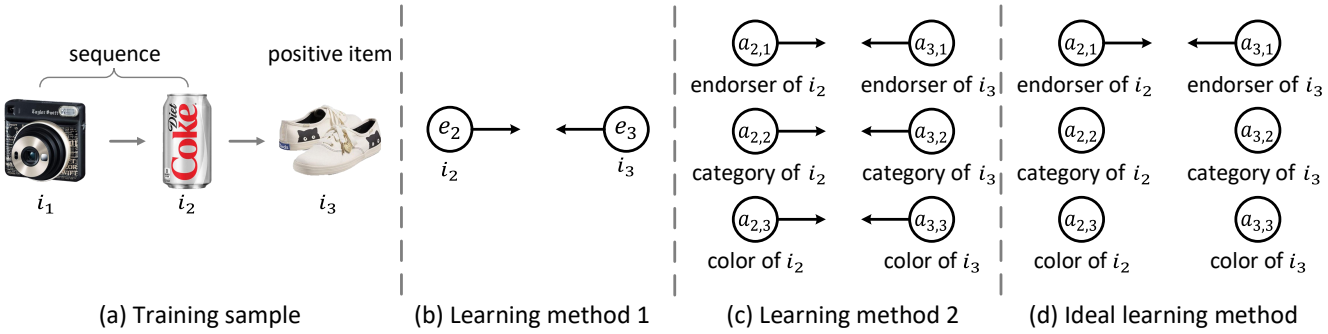

Figure 1: A training sample and three update methods. Subplots (b) and (c) illustrate the learning methods of the existing models, and subplot (d) illustrates a more accurate learning method. e_2 denotes the embedding of item i_2, $a_{2,1}$ denotes the embedding of aspect 1 of item i_2, and the others are similar. The arrows indicate the update direction of the embeddings.

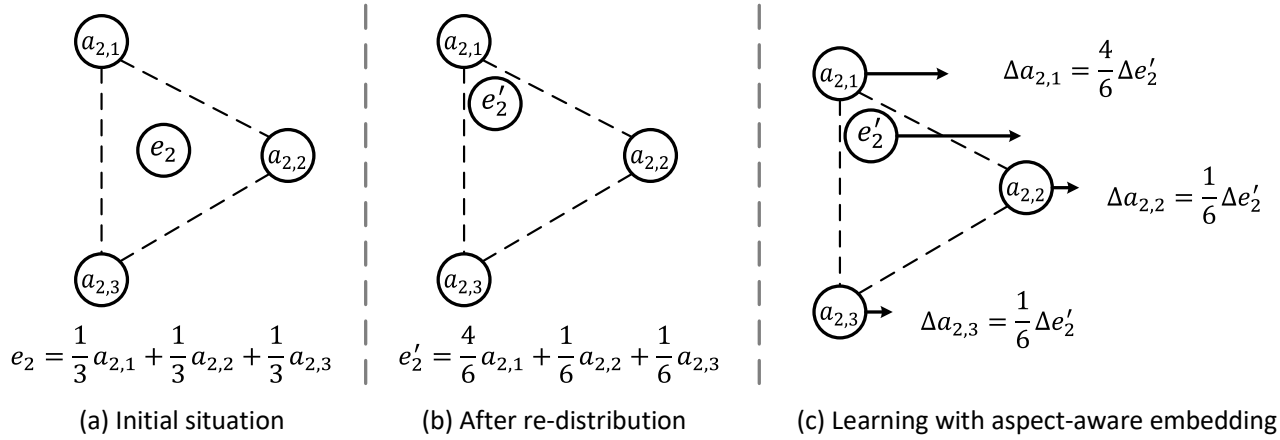

Figure 2: Illustration of the three steps of our proposed methods. $a_{2,1}$, $a_{2,2}$ and $a_{2,3}$ are the embeddings of the three aspects of the item i_2, respectively. e_2 and e_2' indicate the initial item embedding and the aspect-aware item embedding, respectively. Δ denotes the update of an embedding.

(1) For convenience, we focus on the update of item i_2. Before considering the aspect distribution (i.e., the importance of aspects), we first decompose the item embedding e_2 into several aspect embeddings $a_{2,1}$, $a_{2,2}$ and $a_{2,3}$. Although we can introduce additional information (such as knowledge graphs) to help with decomposition, the additional information is expensive and often not available in practice. Initially, as in Figure 2 (a), we consider each aspect to be equally important, i.e., the initial aspect distribution is uniform. Correspondingly, we make the center of the decomposed aspect embeddings coincide with the initial item embedding.
(2) Given a sequence, we re-calculate the importance of each aspect. Specifically, we consider an aspect important if it is similar to the other items in the same sequence. Then, we aggregate the aspect embeddings into a new item embedding according to the aspect distribution as in Figure 2 (b). The new item embedding is called aspect-aware item embedding.
(3) Finally, we directly provide the aspect-aware item embeddings to a successor model for training. As shown in Figure 2 (c), the advantage of this approach is that updates of an aspect-aware item embedding are assigned to each aspect embedding according to the aspect distribution, so it focuses on updating the important aspects.

Notice that since the aspect distribution is related to specific sequences, an item may have different aspect distributions and aspect-aware embeddings in different sequences. The flexibility of aspect-aware item embeddings also improves the capability of the successor sequential recommendation model.

We conduct extensive experiments on four public real-world datasets. We choose a state-of-the-art model, SASRec [14], as the successor model to provide the recommended results. The experimental results demonstrate that our ARD with SASRec outperforms all baselines including the original SASRec on four datasets. The results also show the effectiveness of our insight focusing on updating the important aspects and our ARD. We summarize the contributions of this work as follows:

- We propose to focus on updating the important aspects of an item, which can more accurately capture the reason for the co-occurrence of the item and other items.
- We propose a novel aspect re-distribution method, which provides a solution for our insight by re-calculating the aspect distribution and reconstructing the item embeddings.

- We conduct extensive experiments on real-world datasets and show the effectiveness of our method.

Table 1: Notations and descriptions.

Notation	Description
$\mathcal{I}$	item set
$\mathcal{S} = \{s\}$	sequence set
$s = (i_1^s, i_2^s, \ldots, i_{\lvert s\rvert}^s)$	a sequence
$s_t = (i_1^s, i_2^s, \ldots, i_t^s)$	a subsequence containing the first t items of s
$\mathcal{P} = \{1, 2, \ldots, \lvert\mathcal{P}\rvert\}$	aspect set
$p \in \mathcal{P}$	an aspect
$d \in \mathbb{N}$	embedding dimensionality
$\boldsymbol{e}_i \in \mathbb{R}^d$	embedding of item i
$\boldsymbol{a}_{i,p} \in \mathbb{R}^d$	embedding of aspect p of i
$w_{s,i,p} \in \mathbb{R}$	importance of aspect p of item i in sequence s
$\boldsymbol{e}'_{s,i} \in \mathbb{R}^d$	aspect-aware item embedding of item i in sequence s
$\hat{r}_{s_t,j} \in \mathbb{R}$	relevance of item j being the next item of sequence s_t

2 PROPOSED METHOD

In the problem setting of sequential recommendation, the dataset contains a sequence set $\mathcal{S}$ and an item set $\mathcal{I}$. Each $s \in \mathcal{S}$ is a chronological item sequence $(i_1^s, i_2^s, \ldots, i_{\lvert s\rvert}^s)$, where $i_t^s \in \mathcal{I}$. To a given item sequence s, our task is to predict the next item $i_{\lvert s\rvert+1}^s$ from $\mathcal{I} \setminus s$ and provide an item list sorted by the estimated preferences. We summarize the notations used in this paper in Table 1.

In this section, we propose our aspect re-distribution (ARD) method. We illustrate our ARD with SASRec [14] as the succesor sequential model in Figure 3. The left part illustrates the entire architecture. The right part shows the architecture of the generation of the aspect-aware item embeddings.

2.1 Aspect Embedding Generation

Most of the present sequential models do not explicitly consider the importance of different aspects when updating an item embedding [33]. However, as described in Section 1, different aspects of an item have different contributions to the co-occurrence pattern. Therefore, it is more accurate to focus on updating the important aspects of an item than to update the entire item embedding without emphasis.

Before considering the importance of different aspects, we need to represent an item with several aspect embeddings. Specifically, we decompose an initial item embedding into different aspect embeddings. Assuming $\mathcal{P}$ is the set of aspects that $\mathcal{P} = \{1, 2, \ldots, \lvert\mathcal{P}\rvert\}$ and $\lvert\mathcal{P}\rvert$ is a hyper-parameter, we use a projection matrix to project the item embedding $\boldsymbol{e}_i \in \mathbb{R}^d$ into the embedding of aspect $p \in \mathcal{P}$:

$$\boldsymbol{a}_{i,p} = \frac{\boldsymbol{W}_p \boldsymbol{e}_i}{\|\boldsymbol{W}_p \boldsymbol{e}_i\|_2}, \tag{1}$$

where $\boldsymbol{a}_{i,p} \in \mathbb{R}^d$ is the embedding of aspect p of item i, and $\boldsymbol{W}_p \in \mathbb{R}^{d \times d}$ is the projection matrix of aspect p that is shared by all items. We also normalize the aspect embeddings to facilitate subsequent cross-aspect calculations.

We decompose an item to better distinguish the reasons for its co-occurrence with the other items. The independence of the different aspect embeddings contributes to this purpose. If different aspects contain the same information, the multiple aspect embeddings degenerate to be equivalent to an item embedding. Therefore, we encourage different aspects to contain different information [19].

Inspired by previous work [34], we introduce an independence loss to constrain the aspect embeddings. Specifically, we use mutual information to encourage the separation of different aspect embeddings of an item. For item i, the mutual information of the aspects is:

$$L_{ind}^i = \frac{1}{\lvert\mathcal{P}\rvert} \sum_{p \in \mathcal{P}} -log \frac{\exp(\frac{s(\boldsymbol{a}_{i,p}, \boldsymbol{a}_{i,p})}{\tau})}{\sum_{p' \in \mathcal{P}} \exp(\frac{s(\boldsymbol{a}_{i,p}, \boldsymbol{a}_{i,p'})}{\tau})}, \tag{2}$$

where the $s(\cdot, \cdot)$ measures the similarity of two aspect embeddings and τ denotes the temperature in the softmax function which is a hyper-parameter. We use cosine similarity here:

$$s(\boldsymbol{a}_1, \boldsymbol{a}_2) = \frac{\boldsymbol{a}_1 \boldsymbol{a}_2^T}{\|\boldsymbol{a}_1\|_2 \|\boldsymbol{a}_2\|_2}. \tag{3}$$

For items that frequently appear in different sequences, the independence of different aspects helps capture the reasons for its co-occurrence with the other items. However, for items that appear only in a few sequences, the independence of different aspects is relatively not so important. Therefore, the proportion of an item in the loss function should be proportional to the number of times it appears in different sequences:

$$L_{ind} = \frac{1}{\lvert S\rvert} \sum_{s \in \mathcal{S}} \frac{1}{\lvert s\rvert} \sum_{i \in s} L_{ind}^i, \tag{4}$$

where s is a sequence in $\mathcal{S}$.

2.2 Initial Aspect Distribution

In the beginning, an intuitive assumption is that each aspect of an item is equally important. In other words, we expect each aspect embedding to have the same proportion of information in the initial item embedding. We define the initial aspect distribution, i.e., the importance of the aspects of item i as $\mathcal{D}_i = \left[w_{i,1}, w_{i,2}, \ldots, w_{i,\lvert\mathcal{P}\rvert}\right]$, where $w_{i,p} = \frac{1}{\lvert\mathcal{P}\rvert}$ denotes the importance of aspect p of item i. We then introduce a center loss to constrain the item embedding and the aspect embeddings:

$$L_{cent}^i = \left\| \boldsymbol{e}_i - \sum_{p \in \mathcal{P}} w_{i,p} \boldsymbol{a}_{i,p} \right\|_2, \tag{5}$$

where $\boldsymbol{e}_i$ is the embedding of item i, and $\boldsymbol{a}_{i,p}$ is the embedding of aspect p. In general, the center loss guarantees that each aspect embedding has the same initial proportion of the item embedding since $w_{i,p}$ is initialized with a same value.

Similar to Eq.(4), we use the times of an item appearing in different sequences to determine its proportion in the loss function:

$$L_{cent} = \frac{1}{\lvert S\rvert} \sum_{s \in \mathcal{S}} \frac{1}{\lvert s\rvert} \sum_{i \in s} L_{cent}^i, \tag{6}$$

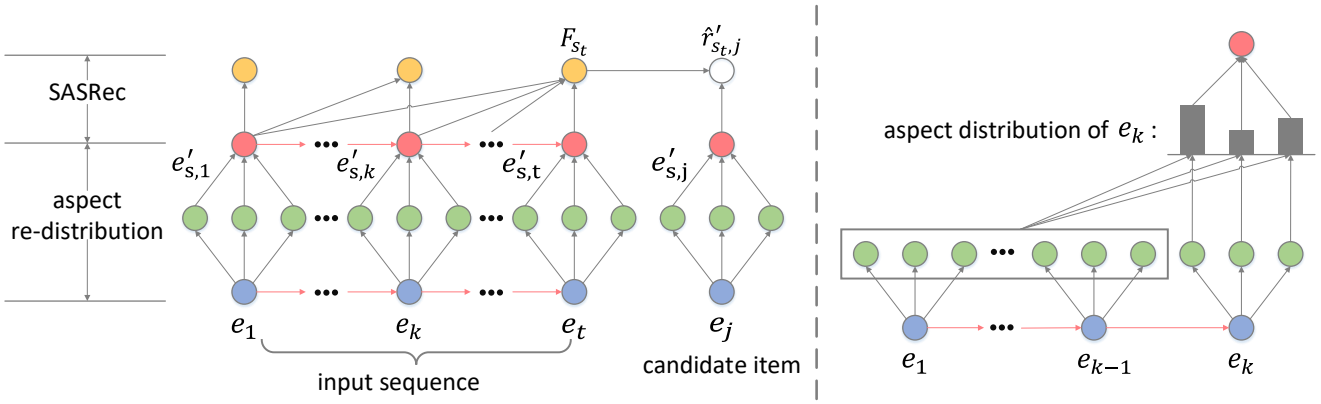

Figure 3: The architecture of our aspect re-distribution (ARD) with SASRec as the succesor sequential model. The left part: First, the embedding (blue node) of each item is decomposed under some constraints (see Eqs.(2~6)) into the aspect embeddings (green nodes) (see Eqs.(1~6)). Next, the aspect embeddings are aggregated into an aspect-aware item embedding (red node) (see Eqs.(7~10)). Finally, the sequence embedding (yellow node) is calculated from the aspect-aware item embeddings using SASRec [14], with which the relevance of the candidate item being the next item can be calculated (see Eqs.(11~12)). The right part: Take the generation of aspect-aware item embedding of item k as an example, all aspect embeddings of the previous items (i.e., items $1, 2, \ldots, k-1$) are accumulated (gray box) (see Eqs.(7~8)). Then, the aspect distribution (grey rectangles) of item k is calculated (see Eq.(9)). Finally, the aspect embeddings of item k are aggregated into an aspect-aware item embedding according to the aspect distribution (see Eq.(10)).

where s is a sequence in $\mathcal{S}$.

2.3 Aspect Distribution Re-calculation

As discussed previously, different aspects of an item have different contributions to the co-occurrence of this item with the other items. For example, given a sequence where all items are endorsed by a same celebrity. An item in the sequence appears together with the other items because of the endorser aspect. Therefore, the aspect distribution of an item is related to the sequence this item is in.

For item i in sequence s, we re-calculate the aspect distribution according to the items in the same sequence s. We define the range of items that influences the distribution as all items before item i in the sequence s to avoid the information leakage and also improve the parallelizability of our method:

$$C_{s,i} = \{j | j \in s, pos_{s,j} < pos_{s,i}\}, \tag{7}$$

where $pos_{s,i}$ represents the position of the item i in the sequence s. The information of the items in $C_{s,i}$ is accumulated into $\boldsymbol{q}_{s,i} \in \mathbb{R}^d$:

$$\boldsymbol{q}_{s,i} = \frac{1}{|C_{s,i}|} \sum_{j \in C_{s,i}} \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \boldsymbol{a}_{j,p}, \tag{8}$$

where $\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \boldsymbol{a}_{j,p}$ contains the information of all aspects of the item j. As for the aspect p of item i, if $\boldsymbol{q}_{s,i}$ is more similar to $\boldsymbol{a}_{i,p}$, we consider that the aspect is more important. Specifically, the fully connected layer is used to learn the importance $w'_{s,i,p} \in \mathbb{R}$ of aspect p of item i in sequence s from $\boldsymbol{q}_{s,i}$ and $\boldsymbol{a}_{i,p}$:

$$w'_{s,i,p} = softmax\left\{[\boldsymbol{q}_{s,i}, \boldsymbol{a}_{i,p}]\boldsymbol{W} + b\right\}, \tag{9}$$

where $[\cdot,\cdot]$ denotes the concatenation operation, and $\boldsymbol{W} \in \mathbb{R}^{2d \times 1}$ and $b \in \mathbb{R}$ denote the weights and bias of the fully connected layer, respectively. The softmax function can increase the proportion of the important aspects in the distribution. So far, we obtain a new aspect distribution $\mathcal{D}'_{s,i} = \left[w'_{s,i,1}, w'_{s,i,2}, \ldots, w'_{s,i,|\mathcal{P}|}\right]$.

2.4 Aspect-aware Item Embedding Generation

After re-calculating the aspect distribution, we need to make the sequential model focus on updating the important aspects. We adopt an effective and generic approach. Specifically, we use the re-calculated aspect distribution $\mathcal{D}'_{s,i}$ as weights and aggregate all aspect embeddings into a single embedding:

$$\boldsymbol{e}'_{s,i} = \sum_{p \in \mathcal{P}} w'_{s,i,p} \boldsymbol{a}_{i,p}, \tag{10}$$

where $\boldsymbol{e}'_{s,i} \in \mathbb{R}^d$ contains the information of all aspects. Thus, we call it aspect-aware item embedding. The aspect-aware item embedding $\boldsymbol{e}'_{s,i}$ can replace the initial item embedding $\boldsymbol{e}_i$ for a successor sequential recommendation model. This approach has the following advantages. First, update of $\boldsymbol{e}'_{s,i}$ is passed back to $a_{i,p}$ according to $w'_{s,i,p}$, which allows the model to focus on updating the important aspects. Second, $\boldsymbol{e}'_{s,i}$ can be used in most sequential models and thus has excellent generality. Third, in the previous sequential models, an item has the same embedding in different sequences. With the introduction of the aspect-aware item embeddings, the embedding of an item can be changed according to the sequence, which also improves the capability of the successor sequential model.

2.5 Next Item Prediction

We use one of the most well-known models, SASRec [14], as the successor recommendation model. Given a sequence of aspect-aware item embeddings $(\boldsymbol{e}'_{s,i^s_1}, \boldsymbol{e}'_{s,i^s_2}, \ldots, \boldsymbol{e}'_{s,i^s_t})$, SASRec uses the self-attention layers and the fully connected layers to aggregate

the item embeddings into a subsequence embedding $\boldsymbol{F}_{s_t} \in \mathbb{R}^d$:

$$\boldsymbol{F}_{s_t} = f_{SASRec}\left((\boldsymbol{e}'_{s,i_1^s}, \boldsymbol{e}'_{s,i_2^s}, \ldots, \boldsymbol{e}'_{s,i_t^s})\right), \tag{11}$$

where f_{SASRec} denotes the self-attention layers and fully connected layers used in SASRec. To predict the next item, SASRec adopts an MF layer on the aggregated sequence embedding $\boldsymbol{F}_{s_t}$ and the aspect-aware item embedding $\boldsymbol{e}'_{s,j}$ of the target item j:

$$\hat{r}_{s_t,j} = \boldsymbol{e}'_{s,j}\boldsymbol{F}_{s_t}^T, \tag{12}$$

where $\hat{r}_{s_t,j}$ denotes the relevance that the next item is item j. Since the architecture of SASRec is not the focus of this paper, please see [14] for more details about SASRec. Notice that the successor model can be replaced by almost all deep learning-based sequential models.

2.6 Network Training

2.6.1 Object Function. The goal of sequential recommendation is to correctly predict the next items w.r.t. a sequence. For a subsequence $s_t = (i_1^s, i_2^s, \ldots, i_t^s)$, the positive item is i_{t+1}^s. A binary cross-entropy loss is adopted as the loss function here:

$$L_{rec} = -\sum_{s\in\mathcal{S}}\sum_{t\in\{1,2,\ldots,|s|-1\}}\left[\log\left(\sigma(\hat{r}_{s_t,i_{t+1}^s})\right) + \sum_{j\notin s}\log\left(1-\sigma(\hat{r}_{s_t,j})\right)\right], \tag{13}$$

where item j is a negative item and $\sigma(\cdot)$ is the activation function. Combining Eq.(13) with Eq.(4) and Eq.(6), we have:

$$L = L_{rec} + \lambda_1 L_{ind} + \lambda_2 L_{cent}, \tag{14}$$

where λ_1 and λ_2 are hyper-parameters to control the weights of L_{ind} and L_{cent}, respectively.

We use the Adam optimizer to learn the parameters of our model. The Adam optimizer is an improved version of the stochastic gradient descent optimizer with better adaptability. We jointly train L_{rec}, L_{ind} and L_{cent}, i.e., optimizing them at the same time in each epoch. In each step, we randomly select an item j from $\mathcal{I}\setminus s$ as a negative item.

2.6.2 Dropout. Adding our ARD without changing the successor recommendation model may increase the depth of the model. As the model becomes more complex, the training process may be unstable, and the over-fitting problem is easy to occur. Therefore, we adopt the 'Dropout' regularization techniques [28] and residual connections [5] in model training.

During the training process, the dropout regularization techniques randomly invalidate the neurons according to a certain probability. As a result, only some of the neurons are used during training. In performance evaluation, the model uses all neurons to make predictions, so the dropout techniques can also be regarded as ensemble learning techniques that integrate exponential models. We adopt the dropout techniques in our method:

$$\boldsymbol{e}'_{s,i} = dropout\left(\boldsymbol{e}'_{s,i}\right), \tag{15}$$

where $\boldsymbol{e}'_{s,i}$ is the aspect-aware item embedding of item i in sequence s.

2.6.3 Residual Connections. The idea of residual connections [5] is to propagate the low-level embeddings directly to the high-level embeddings, bypassing the layers of the model. If the low-level embeddings are sufficient to provide accurate results, residual connections can reduce the negative effects of the model depth. Similarly, we allow the initial item embeddings to directly affect the recommendation results through residual connections. For the aspect embeddings, we adopt the residual connections by extending Eq.(1):

$$\boldsymbol{a}_{i,p} = \frac{\boldsymbol{W}_p\boldsymbol{e}_i}{\|\boldsymbol{W}_p\boldsymbol{e}_i\|_2} + \boldsymbol{e}_i. \tag{16}$$

For the aspect-aware item embeddings, we adopt the residual connections by extending Eq.(10):

$$\boldsymbol{e}'_{s,i} = \sum_{p\in\mathcal{P}} w'_{s,i,p}\boldsymbol{a}_{i,p} + \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\boldsymbol{a}_{i,p}. \tag{17}$$

With Eqs.(16~17), the initial item embedding $\boldsymbol{e}_i$ can be propagated directly to $\boldsymbol{e}'_{s,i}$.

2.7 Method Analysis

2.7.1 Space Complexity. The space complexity of SASRec is $O(|\mathcal{I}|d + nd + d^2)$ [14], where n is the maximum length of the sequences. The learnable parameters of our ARD are from the projection matrices and the fully connected layer. The space complexity of our ARD is $O(|\mathcal{P}|d^2)$, and the total space complexity of the entire model is $O(|\mathcal{I}|d + nd + |\mathcal{P}|d^2)$. Notice that $|\mathcal{P}|$ and d are usually small, such as 8 and 50, respectively. Therefore, $O(|\mathcal{P}|d^2)$ usually has little effect and the actual space occupied by the entire model is very close to that of SASRec.

2.7.2 Time Complexity. The time complexity of SASRec is $O(n^2d + nd^2)$ [14], where n is the maximum length of the sequences. In addition, the cost of our ARD is $O(n|\mathcal{P}|^2d + n|\mathcal{P}|d^2)$. Therefore, the total time complexity of the entire model is $O(n^2d + n|\mathcal{P}|^2d + n|\mathcal{P}|d^2)$. Similarly, since $|\mathcal{P}|$ and d are usually small, the dominant term is $O(n^2d)$. In addition, our ARD is fully parallelizable because the operations of different items are independent of each other. In practice, we can use GPUs to accelerate the training process because of the parallel property of our method.

3 EXPERIMENTS

To answer the following three research questions (RQs), we provide our experimental settings and results:

- RQ1: Compared with the existing sequential models, what performance does our ARD with SASRec achieve?
- RQ2: How do different components of our ARD affect the performance?
- RQ3: What is the impact of the key parameters of our ARD?

3.1 Experimental Settings

3.1.1 Datasets. We use four datasets from three real-world websites, i.e., Amazon[1], Tmall[2] and Twitter[3], following some existing

[1]https://www.amazon.com/
[2]https://www.tmall.com/
[3]https://twitter.com/

Table 2: Statistics of the datasets after preprocessing. Notice that #r/#u and #r/#i represent the average number of records for users and items, respectively.

Dataset	#users	#items	#records	#r/#u	#r/#i
Music	20,165	20,356	132,595	6.58	6.51
Tmall	232,909	97,677	2,048,857	8.80	20.98
Baby	27,626	18,750	216,360	7.83	11.54
Twitter	7,964	4,272	157,530	19.78	36.88

works [12, 17, 37]. The Amazon dataset[4] [8] contains reviews of various types of products on Amazon. We choose two categories of products, i.e., Music and Baby. The Tmall dataset[5] includes different user behaviors from the Tmall e-commerce website, such as purchases, clicks, and so on. The Twitter dataset[6] [3] contains ratings on movies extracted from Twitter. We choose the 200K version of the Twitter dataset.

Similar to [14], we process the datasets as follows: We take the rating records in Music and Baby, purchase records in Tmall and rating records in Twitter as positive feedback. Sequences are obtained by grouping feedback by users and sorting items by timestamps. Then we delete duplicate items in each sequence. Finally, we keep sequences and items with 5 or more records. The last item and the penultimate item in each sequence are for test and validation, respectively, and the other items are for training. The statistics of the processed datasets are shown in Table 2.

3.1.2 Baselines. We include several representative and state-of-the-art recommendation models in our experiments.

- BPR [23]: A classic general recommendation model that does not consider the chronological order of feedback. BPR combines a pairwise loss with a matrix factorization model [15].
- FPMC [24]: A first-order Markov chains model that predicts the next item based on the most recent item. FPMC factorizes user-item interactions and item transitions simultaneously.
- TransRec [6]: A translation-based model that adopts the item and user embeddings to learn the transition probability between items. TransRec predicts the next item according to the transition probability.
- GRU4Rec+ [10]: An RNN-based model for sequential recommendation. GRU4Rec+ is improved from GRU4Rec [11] in loss function and sampling strategy.
- Caser [31]: A CNN-based model that uses CNN to convolve the item embedding matrix to capture the dependencies of local subsequences.
- BERT4Rec [29]: A BERT-based model that applies bidirectional self-attention mechanism to each sequence in training. In this way, richer information is used to help train the model.
- SASRec [14]: A very competitive model that utilizes the self-attention mechanism [32] to aggregate item embeddings into sequence embeddings.

3.1.3 Evaluation Metrics. We evaluate the performance of the models using two metrics: Rec@10 and NDCG@10. In our problem setting, there is only one ground truth item for each sequence, so Rec@10 is proportional to Pre@10 and is equivalent to Hit@10. Rec@10 measures the accuracy of results, while NDCG@10 is also sensitive to the positions of the ground truth items in the lists. To reduce the computational cost, we follow [14, 17] to uniformly randomly sample 100 items as candidates for each user.

3.1.4 Implementation Details. We conduct the experiments with the codes provided by [7][7] for BPR and FPMC. For SASRec, we use the PyTorch version code[8] recommended by the authors of SASRec. For TransRec[9], GRU4Rec+[10], Caser[11] and BERT4Rec[12], we use the codes provided in the original papers. Our ARD is implemented based on the PyTorch version of SASRec mentioned above and the source codes are available[13].

We select the embedding dimensionality d from $\{10, 20, 30, 40, 50\}$ and then set d to 50 for all models to make a fair comparison. For all the baselines, we select the parameters from the ranges suggested in the original papers, such as the Markov order $\in \{1, 2, \ldots, 5\}$ for Caser, the dropout rate $\in \{0.1, 0.2, \ldots, 0.9\}$ for BERT4Rec and so on.

We set the batch size to 128 and the learning rate to 0.001 for our ARD with SASRec. For the SASRec part, we set the dropout rate to 0.5, the number of blocks to 2 and the sequence length to 100 for Twitter and 50 for the other three datasets following the suggested settings in SASRec. For our ARD part, we set the weight of the independence loss λ_1 to 0.5, the weight of the center loss λ_2 to 0.5, and select the dropout rate from $\{0.1, 0.2, \ldots, 0.9\}$ and the number of aspects $|\mathcal{P}|$ from $\{1, 2, 4, 8, 16\}$. Notice that the optimal values of the parameters are selected according to the performance of Rec@10 on the validation data.

3.2 Results

3.2.1 Performance Comparison (RQ1). We show the recommendation performance of our ARD with SASRec and seven baselines on four datasets in Table 3. We find that:

- Our ARD with SASRec achieves the best performance in all cases. In particular, the performance of our ARD with SASRec improves SASRec by 5.30% in terms of Rec@10 and 4.83% in terms of NDCG@10 on average of the four datasets. We attribute the improvement to learning item embeddings more accurately. The baseline models update the entire item embedding without discrimination, ignoring the fact that the importance of different aspects is different. Our ARD infers which aspects are important for the co-occurrence patterns and pays more attention to updating the important aspects. As a result, our ARD can learn more robust item embeddings. In addition, our ARD allows an item to have different aspect-aware item embeddings in different sequences, which provides flexibility and thus improves the capability of the sequential model. In contrast, an item in each baseline

[4] http://jmcauley.ucsd.edu/data/amazon/
[5] https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[6] https://github.com/sidooms/MovieTweetings
[7] https://drive.google.com/file/d/0B9Ck8jw-TZUEeEhSWXU2WWloc0k/view
[8] https://github.com/pmixer/SASRec.pytorch
[9] https://drive.google.com/file/d/0B9Ck8jw-TZUEVmdROWZKTy1fcEE/view
[10] https://github.com/hidasib/GRU4Rec
[11] https://github.com/graytowne/caser_pytorch
[12] https://github.com/FeiSun/BERT4Rec
[13] https://github.com/mdyx/ARD

Table 3: Recommendation performance of our ARD with SASRec and seven baselines on four datasets. Notice that the best results and the second best results are marked in bold and underlined, respectively.

Dataset	Metric	BPR	FPMC	TransRec	GRU4Rec+	Caser	BERT4Rec	SASRec	ARD
Music	Rec@10	0.3769	0.4087	0.5406	0.4446	0.5180	0.5467	<u>0.5526</u>	**0.5890**
	NDCG@10	0.2217	0.2505	0.3813	0.2980	0.3341	0.3630	<u>0.3888</u>	**0.4184**
Tmall	Rec@10	0.4272	0.4675	0.5666	0.5561	0.5525	<u>0.6155</u>	0.6121	**0.6459**
	NDCG@10	0.2514	0.2862	0.3791	0.3791	0.3496	<u>0.4052</u>	0.4046	**0.4373**
Baby	Rec@10	0.4145	0.4307	0.4834	0.4232	0.4496	<u>0.5137</u>	0.5092	**0.5265**
	NDCG@10	0.2299	0.2436	0.2874	0.2494	0.2702	<u>0.3131</u>	0.3054	**0.3159**
Twitter	Rec@10	0.7143	0.6977	0.7362	0.6279	0.7384	0.7363	<u>0.7403</u>	**0.7579**
	NDCG@10	0.4853	0.4690	0.4907	0.3824	0.5317	0.5384	<u>0.5399</u>	**0.5547**

can only have one same embedding in different sequences, which limits the model's capability.

- SASRec and BERT4Rec using self-attention networks achieve the second best performance, which indicates that the self-attention networks are the state-of-the-art architectures for modeling sequential dependencies. A possible reason is that, compared with the other deep learning-based models (e.g., GRU4Rec+ and Caser), SASRec and BERT4Rec can aggregate different item embeddings based on the weights, thus giving the model the capability to handle complex dependencies. On this basis, our ARD aggregates aspect embeddings at a finer granularity, which allows the model to capture more detailed co-occurrence patterns.
- The traditional models (BPR, FPMC, and TransRec) perform poorly on Tmall while achieving competitive performance on the other three datasets. The results show that the performance gap between the traditional models and the deep learning-based models is more pronounced on large datasets, which is consistent with the observations of previous work [14].

3.2.2 Ablation Study (RQ2). We remove different components from our ARD to study their effectiveness. The results on Rec@10 and NDCG@10 are shown in Table 4. We introduce the variants and analyze their results as follows:

- Without DC: We remove the decomposition operation in Eq.(1) in this variant. For fair comparison, each item has $|\mathcal{P}|$ original embeddings of length d (the same shape of several aspect embeddings of an item). The degradation of the performance indicates that it is more effective to decompose an item embedding into several aspect embeddings than to simply use different embeddings to represent an item.
- Without IL: We remove the independence loss L_{ind} in Eq.(4) and find that the performance becomes worse. The purpose of the independence loss is to allow different aspects of an item to contain different information. Without the independence loss, different aspects are more similar, which leads to the difference of aspect importance being unable to influence the prediction results.
- Without CL: We find that the performance degrades without the center loss L_{cent} in Eq.(6). As mentioned in Section 2.2, the center loss makes each aspect contain the same amount of information and forces each aspect to have the same initial importance. The experimental results also show that this intuitive preparation work is necessary and useful.
- Without DP: Without the dropout techniques, the performance is poor because of overfitting. In order to alleviate the overfitting caused by the model parameters introduced by our ARD, it is necessary to adopt the dropout techniques.
- Without RC: Without residual connections, the performance changes slightly. We find that in some situations, removing residual connections improves the performance. As mentioned in Section 2.6.3, with residual connections, the initial item embeddings can propagate to the aspect-aware item embeddings. Therefore, a possible reason for the results is that the initial embeddings are less important than the aspect-aware embeddings.

3.2.3 Quantitative Study (RQ3). We show the experimental results with different values of the embedding dimensionality d and the size of the aspect set $|\mathcal{P}|$ to study the impact of these two important parameters. We summarize the performance of our ARD with SASRec under different settings of d and $|\mathcal{P}|$ in Figure 4 and 5, respectively.

We change d in $\{10, 20, 30, 40, 50\}$ to study the impact of d and find that:

- For all datasets, our ARD with SASRec achieves better performance as d increases from 10 to 30. The reason is that the embeddings can model more information as d becomes larger. On Tmall and Amazon Baby, when $d \in \{10, 20, 30\}$, the performance improves rapidly with the increase of d, and when $d \in \{40, 50\}$, the performance improvement tends to be gentle.
- On Amazon Music and Twitter, the performance with $d = 40$ exceeds the performance with $d = 50$. The possible reason is that the scale of these datasets is relatively small and the increase of d brings overfitting.

In order to learn the impact of $|\mathcal{P}|$, we vary $|\mathcal{P}|$ in $\{1, 2, 4, 8, 16, 32\}$ and have the following observations:

- Changing $|\mathcal{P}|$ from 1 to 8 improves the performance of our ARD with SASRec. As the number of aspects increases, an item is divided into more fine-grained parts. In this case, our ARD can focus on updating more subtle parts of the item and thus learns the item embedding more accurately.

Table 4: Recommendation performance (Rec@10 and NDCG@10) of our ARD with SASRec and its seven variants on four datasets. Notice that the best results and the second best results are marked in bold and underlined, respectively.

Dataset	Metric	Default	W/o DC	W/o IL	W/o CL	W/o DP	W/o RC
Music	Rec@10	0.5890	0.5768	0.5667	0.5815	0.5855	**0.5951**
	NDCG@10	**0.4181**	0.4081	0.4015	0.4107	0.4080	0.4132
Tmall	Rec@10	**0.6459**	0.6353	0.6363	0.6334	0.6252	0.6438
	NDCG@10	0.4373	0.4276	0.4333	0.4229	0.4146	**0.4380**
Baby	Rec@10	**0.5265**	0.5166	0.5120	0.5105	0.5158	0.5237
	NDCG@10	**0.3159**	0.3119	0.3082	0.2979	0.2989	0.3143
Twitter	Rec@10	**0.7579**	0.7487	0.7535	0.7516	0.7504	0.7550
	NDCG@10	0.5547	0.5481	0.5511	0.5447	0.5450	**0.5564**

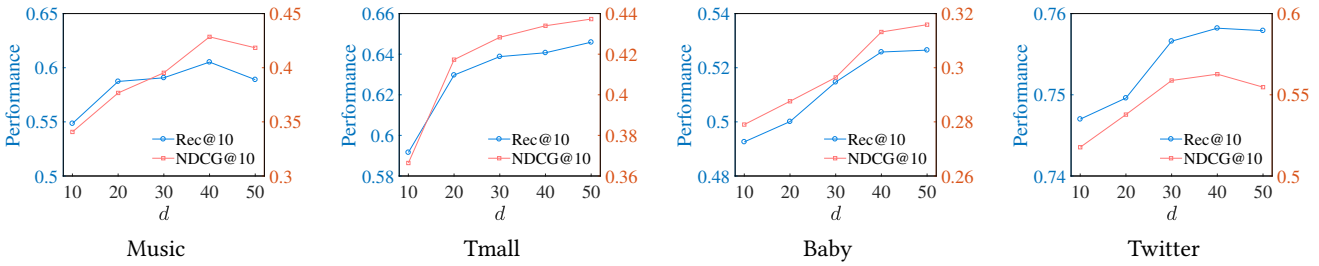

Figure 4: Recommendation performance (Rec@10 and NDCG@10) of our ARD with SASRec under different settings of embedding dimensionality $d \in \{10, 20, 30, 40, 50\}$ on four datasets.

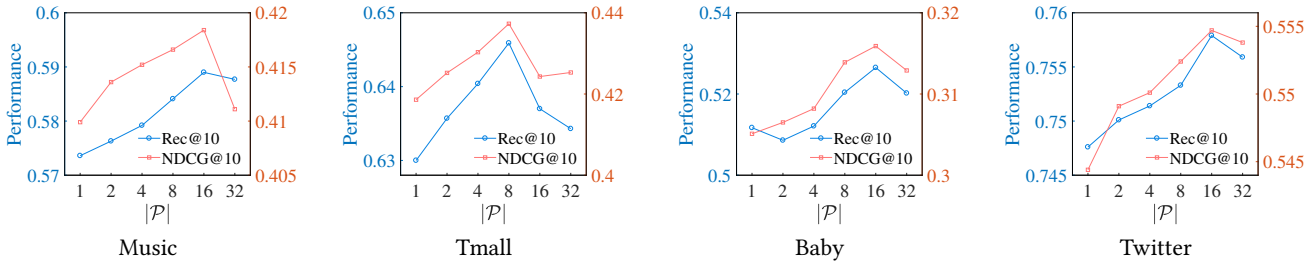

Figure 5: Recommendation performance (Rec@10 and NDCG@10) of our ARD with SASRec under different settings of the aspect numbers $|\mathcal{P}| \in \{1, 2, 4, 8, 16, 32\}$ on four datasets.

- In most cases (except NDCG@10 on Tmall), the performance decreases as $|\mathcal{P}|$ increases from 16 to 32. On the one hand, when the number of aspects $|\mathcal{P}|$ is sufficient to capture the reasons for item co-occurrence, increasing $|\mathcal{P}|$ does not improve the capability of our model. On the other hand, we fix the other parameters for fair comparison. However, increasing the number of aspects increases the complexity of our model, for which we may need to tune the parameters again.
- On Tmall, the optimal value of $|\mathcal{P}|$ is 8, while on the other three datasets, the optimal value is 16. Considering that different datasets are collected in different scenarios, the difference in optimal values of $|\mathcal{P}|$ is reasonable. The result prompts us to choose the appropriate number of aspects according to specific scenarios in practical applications.

4 RELATED WORK

In this section, we review the works that are related to ours. We first introduce the existing models representing item with one single embedding in general recommendation and sequential recommendation, and the relation between these works and our ARD. Then we discuss some models representing an item with several embeddings, and how our ARD improves on them.

4.1 Models Representing Item with Single Embedding

General Recommendation. General recommendation treats users' feedback as a set of user-item interaction records, regardless of the time of feedback. Early works [1, 25] focus on neighborhood-based models, which calculate the similarities between the users (user-based recommendation) or the items (item-based recommendation), which are then used to construct the neighborhood for the

prediction of users' preferences. Later, factorization-based models [13, 15, 23] are widely studied because of their promising performance. For example, some works [22, 23] model the user-item interaction records as an interaction matrix, and decompose the matrix into user and item representations. On this basis, FISM [13] treats each user representation as a set of item representations, which helps the model learn robust user representations on sparse data. Deep learning is introduced [9, 26] into general recommendation to infer users' complex preferences. NCF [9] is an early deep learning-based model, which applies a multi-layer perceptron (MLP) to learn users' preferences. As a common unsupervised model in deep learning, autoencoders (AE) [26] are also used to learn the representations or directly fill the interaction matrix. DIN [40] introduces the attention mechanism that assigns weights to historical items based on their similarity to the candidate item and proposes a new regularization technique.

Sequential Recommendation. We divide the works of sequential recommendation into two categories: traditional sequential models and deep learning-based sequential models. Traditional sequential models focus on Markov chains (MC) and matrix factorization (MF) techniques. Early works [4, 42] use first-order Markov chains to reduce model complexity. These models capture the short-term preferences of users, which consider only the last item in each sequence when predicting the next item. Some works propose to model the long-term preferences of users. For example, first-order and second-order Markov chains are combined [39], and Markov decision processes (MDP) is used to capture higher-order item transition patterns [27]. To capture the users' short-term and long-term preferences at the same time, FPMC [24] combines the first-order MC models and the MF models [15]. Inspired by FISM [13], Fossil [7] adopts the similarities between items and higher-order Markov chains to model user preferences. Instead of capturing short-term and long-term preferences in two components, TransRec [6] is proposed as a unified transition-based framework to study users' preferences. Deep learning-based sequential models adopt neural networks to learn sequential dependencies. GRU4Rec is an early deep learning-based sequential recommendation model that uses RNN to capture sequential dependencies. Some works [2, 10] improve the architectures of RNN based on GRU4Rec. CNN-based models [31, 36] are proposed to relax the rigid order assumption in sequential recommendation and learn local sequential dependencies. GNN-based models [18, 35] treat each sequence as a subgraph and aggregate the items with GNN on a global graph. Inspired by Transformer [32], a self-attention model called SASRec [14] uses the self-attention mechanism to learn the weights of the items and aggregates the item embeddings according to the weights. The self-attention models usually achieve state-of-the-art performance in practice.

However, almost all the existing models ignore the fact that different aspects of an item may have different importance for the co-occurrence patterns. For example, SASRec and MA-GNN [18] use an embedding to represent an item and train the model by drawing the entire embeddings of the two co-occurring items closer. These models simply update the entire embedding as a whole, which makes them limited in their capabilities. Our aspect re-distribution distinguishes different aspects and focuses on updating the important ones, overcoming the limitations of the existing models.

4.2 Models Representing Item with Several Embeddings

Before calculating the importance of each aspect of an item, we represent the item with different aspect embeddings. Some existing works disentangle each embedding into latent factors and also use several embeddings to represent a user or an item, which is called disentangled representation learning. We introduce these works and their differences with ours in the subsequent paragraphs.

MacridVAE [20], DGCF [34] and Disen-GNN [16] first disentangle each user into concepts. Then, they simply sum the similarity of the user and the candidate item within each concept for prediction. SINE [30] and KA-MemNN [41] aggregate several items within the same sequence into a single concept, thus disentangling a sequence into some concepts. Similarly, when predicting the next item, they accumulate the results of each concept without discrimination. The difference between these models and our aspect re-distribution is that they treat each aspect or concept equally in prediction and training. That is to say, they update different aspects of an item without distinction. In contrast, our aspect re-distribution pays more attention to training important aspects and thus allowing our model to update the item embeddings more accurately.

Recently, a new seq2seq training framework [21] for sequential recommendation is proposed. Specifically, it modifies the predicted target from the next item to the next sequence, and disentangles each sequence into several intentions. During training, it tries to make the input sequence and the target sequence similar within the same intention. Although it abandons some intentions with a too large loss, there is no difference in update strength for the different intentions involved in training. This leads to the model still updating different intentions with the same strength in essence.

5 CONCLUSIONS AND FUTURE WORK

We propose to focus on aspects that have a significant impact on co-occurrence patterns when updating item embeddings and propose a new aspect re-distribution method. Specifically, we decompose each item embedding into several aspect embeddings before considering the importance of aspects. We then design the center loss to ensure that the initial aspect distribution is uniform. On this basis, we re-calculate the importance of each aspect according to the other items in the current sequence. Finally, we aggregate aspect embeddings into an aspect-aware item embedding and select SASRec [14] as a successor sequential model. Compared with the existing sequential models such as SASRec, our ARD spends more attention on updating important aspects when learning the item embeddings, resulting in more robust item embeddings. Extensive experiments on four real-world datasets show that our ARD with SASRec achieves better performance than all the seven baselines. We also conduct ablation and quantitative experiments to study the effectiveness of different components and the impacts of the key parameters of our ARD.

In the future, we plan to adopt more sequential models as the successor model for our ARD in order to investigate the impact of the successor models on the performance of the ARD. In addition, a possible direction is to leverage additional information (e.g., item attributes, knowledge graphs, etc.) to help us generate more meaningful aspects to further improve the performance.

ACKNOWLEDGMENTS

This paper is supported by the National Key R&D Program of China under grant (2022ZD0208605), and partially supported by the National Natural Science Foundation of China (NSFC) under grant No.62172283 and No.61672449.

REFERENCES

[1] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.
[2] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Recsys'17*. 152–160.
[3] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. MovieTweetings: A movie rating dataset collected from Twitter. In *RecSys'13*.
[4] Magdalini Eirinaki, Michalis Vazirgiannis, and Dimitris Kapogiannis. 2005. Web path recommendations based on page ranking and Markov models. In *WIDM'05*. 2–9.
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR'16*. 770–778.
[6] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Recsys'17*. 161–169.
[7] Ruining He and Julian McAuley. 2016. Fusing similarity models with Markov chains for sparse sequential recommendation. In *ICDM'16*. 191–200.
[8] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW'16*. 507–517.
[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW'17*. 173–182.
[10] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM'18*. 843–852.
[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[12] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *WSDM'19*. 573–581.
[13] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-N recommender systems. In *KDD'13*. 659–667.
[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM'18*. 197–206.
[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[16] Ansong Li, Zhiyong Cheng, Fan Liu, Zan Gao, Weili Guan, and Yuxin Peng. 2022. Disentangled graph neural networks for session-based recommendation. *arXiv preprint arXiv:2201.03482* (2022).
[17] Jing Lin, Weike Pan, and Zhong Ming. 2020. FISSA: Fusing item similarity models with self-attention networks for sequential recommendation. In *RecSys'20*. 130–139.
[18] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation. In *AAAI'20*, Vol. 34. 5045–5052.
[19] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *ICML'19*. 4212–4221.
[20] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. *arXiv preprint arXiv:1910.14238* (2019).
[21] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *KDD'20*. 483–491.
[22] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup'07*, Vol. 2007. 5–8.
[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW'10*. 811–820.
[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW'01*. 285–295.
[26] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW'15*. 111–112.
[27] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).
[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[29] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM'19*. 1441–1450.
[30] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-interest network for sequential recommendation. In *WSDM'21*. 598–606.
[31] Jiaxi Tang and Ke Wang. 2018. Personalized top-N sequential recommendation via convolutional sequence embedding. In *WSDM'18*. 565–573.
[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[33] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet Orgun, and Defu Lian. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).
[34] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *SIGIR'20*. 1001–1010.
[35] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *SIGIR'20*. 169–178.
[36] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *WSDM'19*. 582–590.
[37] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders. In *SIGIR'17*. 957–960.
[38] Wen Zhang, Yuhang Du, Taketoshi Yoshida, and Ye Yang. 2019. DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function. *Information Sciences* 470 (2019), 121–140.
[39] Zhiyong Zhang and Olfa Nasraoui. 2007. Efficient hybrid web recommendations based on Markov clickstream models and implicit search. In *WI-IAT'07*. 621–627.
[40] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI'19*, Vol. 33. 5941–5948.
[41] Nengjun Zhu, Jian Cao, Yanchi Liu, Yang Yang, Haochao Ying, and Hui Xiong. 2020. Sequential modeling of hierarchical user intention and preference for next-item recommendation. In *WSDM'20*. 807–815.
[42] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2013. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320* (2013).