Coarse-to-Fine Sparse Sequential Recommendation

Jiacheng Li  
University of California, San Diego  
j9li@eng.ucsd.edu

Tong Zhao  
Amazon, United States  
zhaoton@amazon.com

Jin Li  
Amazon, United States  
jincli@amazon.com

Jim Chan  
Amazon, United States  
jamchan@amazon.com

Christos Faloutsos  
Carnegie Mellon University  
christos@cs.cmu.edu

George Karypis  
Amazon, United States  
gkarypis@amazon.com

Soo-Min Pantel  
Amazon, United States  
pantel@amazon.com

Julian McAuley  
Amazon, United States  
jumcaule@amazon.com

ABSTRACT
Sequential recommendation aims to model dynamic user behavior from historical interactions. Self-attentive methods have proven effective at capturing short-term dynamics and long-term preferences. Despite their success, these approaches still struggle to model sparse data, on which they struggle to learn high-quality item representations. We propose to model user dynamics from shopping intents and interacted items simultaneously. The learned intents are coarse-grained and work as prior knowledge for item recommendation. To this end, we present a coarse-to-fine self-attention framework, namely CaFe, which explicitly learns coarse-grained and fine-grained sequential dynamics. Specifically, CaFe first learns intents from coarse-grained sequences which are dense and hence provide high-quality user intent representations. Then, CaFe fuses intent representations into item encoder outputs to obtain improved item representations. Finally, we infer recommended items based on representations of items and corresponding intents. Experiments on sparse datasets show that CaFe outperforms state-of-the-art self-attentive recommenders by 44.03% NDCG@5 on average.

CCS CONCEPTS
• Information systems → Personalization.

KEYWORDS
Sequential Recommendation, Transformer, Sparse Data, Coarse-to-Fine Framework

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR ‘22, July 11–15, 2022, Madrid, Spain
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8732-3/22/07...
https://doi.org/10.1145/3477495.3531732

1 INTRODUCTION
The goal of sequential recommender systems is to predict next items for users by modeling historical interactions as temporally-ordered sequences. Sequential recommenders [9, 10, 16, 17] capture both long-term preferences and short-term dynamics of users simultaneously in order to improve recommendation accuracy.

Previous works employ Markov Chains (MC) [7, 16], RNN/CNNs [5, 12, 18, 24] and self-attentive models [10, 13, 17, 22] for sequential recommendation. Among these approaches, self-attentive recommenders arguably represent the current state-of-the-art, as the self-attention mechanism [21] is able to efficiently draw context from all past actions and obtain short-term dynamics. Some recent works [14, 23, 25] incorporate item context features into item representations due to the flexibility of self-attention. Despite the effectiveness of existing self-attentive models, in this paper we argue that sequential recommendation on highly-sparse sequences (i.e., containing long-tail items) is still a challenging problem for self-attentive recommenders.

To explain why self-attentive models fail on sparse sequences and validate our motivation, we first conduct two motivating experiments (Section 2.3) with a representative self-attentive recommender (BERT4Rec [17]). Results reveal that main reasons: (1) although self-attentive models directly attend on all interactions, they tend to focus on recent items when trained on (item-)sparse datasets. (2) embeddings of long-tail (infrequent) items are under-trained while models represent frequent items well.

To address the above problems, we can employ another dense sequence (called an intent sequence) to provide prior knowledge and well-trained representations for items. As shown in Figure 1, although a user interacts with many items (including infrequent items) in the item sequence, several fall under the same shopping
2 PRELIMINARIES

2.1 Problem Setup

We study sequential recommendation for a user set \( \mathcal{U} \), an item set \( \mathcal{V} \), an intent set \( \mathcal{C} \) (\(|\mathcal{C}| \ll |\mathcal{V}|\)) and a set of user interaction sequences \( \mathcal{S} = \{S_1,S_2,\ldots, S_{|\mathcal{U}|}\} \). Each item \( v \in \mathcal{V} \) has a unique corresponding intent \( c \in \mathcal{C} \). A user sequence consists of (temporally-ordered) interactions \( S_u = (s^{u}_{1}, s^{u}_{2}, \ldots, s^{u}_{|S_u|}) \), where \( s^{u}_{i} \in \mathcal{S} \), \( u \in \mathcal{U} \). We denote the interaction history \( S_u \), we predict the next item \( s^{u}_{|S_u|+1} \).

2.2 Self-Attentive Recommender

Self-attentive recommenders [10, 17, 22, 25] rely on Transformer structure [21] to encode sequential interactions \( S \). In this paper, our backbone model is a directional self-attentive model SASRec [10].

2.2.1 Embedding

For an item set \( \mathcal{V} \), an embedding table \( E \in \mathbb{R}^{d \times |\mathcal{V}|} \) is used for all items, whose element \( e_j \in \mathbb{R}^d \) denote the embedding for item \( v_j \) and \( d \) is the latent dimensionality. To be aware of item positions, SASRec maintains a learnable position embedding \( P \in \mathbb{R}^{d \times n} \), where \( n \) is the maximum sequence length. All interaction sequences are padded to \( n \) with a special ‘padding’ item. Hence, given a padded item sequence \( S^n = \{o_1,v_2,v_3,\ldots,v_n\} \), the input embedding is computed as:

\[
M^n = \text{Embedding}(S^n) = [e_1 + p_1, e_2 + p_2, \ldots, e_n + p_n]
\]

2.2.2 Transformer Encoder

The Transformer encoder adopts scaled dot-product attention [21] denoted as \( f^{att} \). Given \( H^l_0 \in \mathbb{R}^d \) is an embedding for \( v_l \) after the \( l \)-th self-attention layer and \( H^0 = e_j + p_i \), the output from multi-head (\( \#\text{head}=M \)) self-attention is calculated as:

\[
O_l = \text{Concat} \{O^{(1)}_l, \ldots, O^{(m)}_l, \ldots, O^{(M)}_l\} W_O.
\]

where \( W_Q^{(m)}, W_K^{(m)}, W_V^{(m)} \in \mathbb{R}^{d \times d/M} \) are the \( m \)-th learnable projection matrices; \( W_O \in \mathbb{R}^{d \times d} \) is a learnable matrix to get the output \( O_l \) from concatenated heads. Our backbone SASRec model is a directional self-attention model implemented by forbidding attention weights between \( v_i \) and \( v_j \) (\( j > i \)).

To prevent overfitting and achieve a stable training process, the next layer \( H_{l+1} \) is generated from \( O_l \) with Residual Connections [6], Layer Normalization [2] and Pointwise Feed-Forward Networks [21].

2.3 Self-Attentive Models on Sparse Data

To find reasons that self-attentive models fail on sparse data, we conduct two motivating experiments with BERT4Rec. In experiments, we set hidden size \( d = 128 \) and maximum sequence length of \( n = 50 \). We adopt the same training method as in [17].

\subsection{Attention Scope}

We investigate the difference between self-attention scope on dense versus sparse data: Amazon [15] (av. 2.81 interactions per item) is used as a sparse dataset; A dense version (av. 11.23 interactions per item) is constructed by setting the minimum item frequency to 5. We visualize the average attention map from the first self-attention layer in Figure 2a/2b which shows that
the model attends on more recent items on the sparse dataset, and less recent items for the dense dataset. This indicates that: (1) recent items are important sparse data; (2) self-attentive models combine long and short-term dynamics, but they still struggle to capture long-term preferences on item-sparse datasets.

2.3.2 Trained Embedding. In this experiment, we explore the difference of trained item embeddings between frequent items and infrequent items in the same dataset. Specifically, we first select three intents (i.e., categories in the Amazon dataset), then obtain 500 the most frequent items and 500 the most infrequent items in each intent and visualize their trained embeddings in Figure 2c and 2d respectively. We can see that frequent items with the same intent are usually close to each other (form three colored clusters in Figure 2c) while infrequent item embeddings scatter and are mostly around the origin. These observations indicate that (1) the model represents frequent items well, though infrequent items embeddings are under-trained; (2) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (2) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (2) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (3) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (3) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (4) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (5) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (6) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained; (7) intents can provide useful prior knowledge for items because the clusters from well-trained item embeddings are under-trained.

3 METHOD

In this section, we propose CaFe to advance sequential recommendation performance on sparse datasets.

3.1 Embedding Layer

In our method, an interaction sequence \( S_u \) includes an item sequence \( S^p_u \) and an intent sequence \( S^c_u \) for user \( u \). Hence, we maintain two embedding tables \( E^p \in \mathbb{R}^{d \times |V|} \) and \( E^c \in \mathbb{R}^{d \times |C|} \) for items and intents. Two separate position embeddings \( P^p \in \mathbb{R}^{d \times n} \) and \( P^c \in \mathbb{R}^{d \times n} \) are created. We encode \( S^p_u \) and \( S^c_u \) to get input embeddings \( M^p \) and \( M^c \) as follows:

\[
M^p = \text{Embedding}^p(S^p_u); \quad M^c = \text{Embedding}^c(S^c_u),
\]

where \( M^p, M^c \in \mathbb{R}^{d \times n} \) and \( \text{Embedding}(\cdot) \) is the positional embedding operation in Equation (1).

3.2 Coarse-to-Fine Encoder

Recalling the conclusions in Section 2.3, embeddings of infrequent items are under-trained and the self-attentive model tends to focus on short-term items when the dataset has sparse items. We can also see that intent types are highly aligned with item clusters trained by a self-attentive recommender. Motivated by these observations, we propose to explicitly learn intents in a sequential model and the outputs of the intent model are used as prior knowledge to improve item representations and understand long-term preferences. In this section, we introduce our coarse-to-fine encoder which includes two components, the intent encoder and the item encoder. The overall framework is illustrated in Figure 3.

3.2.1 Intent Encoder. For intent sequences, we aim to capture coarse-grained interest dynamics of users. Intent sequences are usually dense because \(|C| \) is much smaller than \(|V| \). Therefore, we apply a standard SASRec model (in Section 2.2) as the encoder for intent sequences. Given intent embeddings \( M^c \), outputs of the SASRec encoder are used as intent sequence representations \( R^c \in \mathbb{R}^{d \times n} \).

3.2.2 Item Encoder. From our motivating experiments, we see that more recent items are important for next item prediction on sparse datasets. Basically, our item encoder is also a directional Transformer but has enhanced ability to focus on recent items. Inspired by [8], we enhance short-term user dynamics modeling in the item encoder by applying a masking score \( \theta_{ij} \) on \( f_{\text{att}} \) in Equation (3). Formally, the re-weighted attention weights are calculated by:

\[
f_{\text{att}}(Q_i, K_j) = \exp(w_{ij} \cdot \theta_{ij}) / \sum_{k=1}^{|V|} \exp(w_{ik} \cdot \theta_{ik}) \quad w_{ij} = \frac{Q_i K_j^T}{\sqrt{d}} \tag{5}
\]

where \( \theta = 1 \) for a standard masked dot-product attention and \( \sqrt{d} \) is a scale factor. The masking operation \( \exp(w_{ij}) \cdot \theta_{ij} \) can be rewritten as \( \exp(w_{ij} + \ln\theta_{ij}) \). We learn \( \ln\theta_{ij} \) from \( H^i_d, H^j_d \) and the distance between items \( v_i \) and \( v_j \):

\[
\ln\theta_{ij} = (H^i_d W^m_Q + H^j_d W^m_m + d_{ij})(W^m_L + b_L) \tag{6}
\]

where \( W^m \in \mathbb{R}^{d/M \times |V|} \), \( b_L \in \mathbb{R}^{d} \), distance embedding \( d_{ij} \in \mathbb{R}^{d/M} \) is the \((n+i-R\cdot n+i)\)-th vector from distance embedding table \( D \in \mathbb{R}^{d \times n2} \) and \( W^m \) are from Equation (3). We encode \( M^p \) with the item encoder to get the item sequence representations \( R^p \in \mathbb{R}^{d \times n} \) and \( R^c \in \mathbb{R}^{d \times n} \) are used to get final representations \( R = R^p + R^c \).

3.2.3 Prediction Layer. In CaFe, we predict the next intent and item simultaneously from \( R^c \) and \( R^p \). Specifically, we adopt matrix factorization (MF) to compute the relevance at time step \( t \) between encoder outputs and embeddings:

\[
r^c_{j,t} = R^c_i E^c_j^T; \quad r^p_{k,t} = R^p_i E^p_k^T \tag{8}
\]

where \( E^c_j \in \mathbb{R}^{d}, E^p_k \in \mathbb{R}^{d} \) denotes embeddings of the \( j \)-th intent and the \( k \)-th item in \( E^c, E^p \) respectively.
we adopt the binary cross entropy loss:

$$\ell = -\sum_{s \in S} \sum_{t \in S} \left[ \log(\sigma(r_{c}^{i, j, t})) + \sum_{c \in S} \log(1 - \sigma(r_{c}^{i, j, t})) \right]$$

where \(y^c, y^d\) are an expected intent and item; \(c, v\) are a negative intent and item randomly generated for each time step in each sequence. Other training details are the same as in SASRec [10].

### 3.4 Inference

Previous sequential recommenders infer the next items conditioned on previous items. In contrast, CaFe computes the joint probability distribution of the item and corresponding intent conditioned on previous intents and items. Formally, at inference we find the item \(v_k\) and corresponding intent \(c_j\) that maximize the probability:

$$P(c_j, v_k | S_o^c, S_o^d, \Theta) = \sigma(r_{f, j, t}^c) \sigma(r_{o, k, t}^d)$$

where \(\sigma\) is the sigmoid function, \(\Theta\) denotes the parameter set of CaFe and \(S_o^c, S_o^d\) are intent and item sequences for user \(u\).

### 4 EXPERIMENTS

#### 4.1 Experimental Setting

**Data.** We consider two sparse datasets (see Table 2): Amazon [15] is collected from Amazon.com, and we use item categories as coarse-grained sequences; Tmall is released in the IJCAI-15 challenge [1]. Sellers of products are used as coarse-grained sequences. We follow [10, 17] to conduct a leave-last-2-out data split.

**Baselines.** We compare two groups of works as our baselines which include methods with only items and methods using both intents and items. Item-only Methods: PopRec, a baseline method that recommends items according to item occurrences in the dataset. SASRec [10], a directional self-attention method that is used as our backbone model. BERT4Rec [17], a bi-directional self-attention method that learns to recommend items via a cloze task similar to BERT [4]. SSE-PT [22], extends SASRec by introducing explicit user representations. LOCKER [8], enhances short-term user dynamics via local self-attention. Intent-aware Methods: NOVA [14], uses non-invasive self-attention to leverage side information. We use intents as side information. FDSA [25], applies separated item and feature sequences but does not explicitly learn the feature sequences. BERT-F, LOCKER-F, our extension of BERT4Rec and LOCKER, which incorporate intents in the same way as FDSA.

**Model comparison.** The best results are bold and the best baselines are underlined.

| Dataset | Metric | PopRec | SASRec | BERT4Rec | SSE-PT | LOCKER | NOVA | FDSA | BERT-F | LOCKER-F | CaFe |
|---------|--------|--------|--------|----------|--------|--------|------|------|--------|----------|------|
| Amazon  | NDCG@5 | 0.0286 | 0.1418 | 0.1830   | 0.2108 | 0.2170 | 0.0281| 0.0670| 0.2199 | 0.2436   | 0.3733| +5.23% |
| Amazon  | HR@5   | 0.0487 | 0.1844 | 0.2240   | 0.2501 | 0.2597 | 0.0475| 0.1089| 0.2676 | 0.2947   | 0.4813| +6.32% |
| Amazon  | MRR    | 0.0485 | 0.1522 | 0.1956   | 0.2239 | 0.2297 | 0.0477| 0.0857| 0.2329 | 0.2529   | 0.3656| +4.56% |
| Tmall   | NDCG@5 | 0.0360 | 0.0741 | 0.2753   | 0.2106 | 0.2961 | 0.0501| 0.1083| 0.2998 | 0.3182   | 0.4290| +34.82%|
| Tmall   | HR@5   | 0.0596 | 0.1205 | 0.3673   | 0.2977 | 0.3872 | 0.0812| 0.1685| 0.3917 | 0.4098   | 0.5152| +25.72%|
| Tmall   | MRR    | 0.0577 | 0.0948 | 0.2782   | 0.2173 | 0.2979 | 0.0716| 0.1265| 0.3014 | 0.3189   | 0.4268| +33.84%|

4.1.3 Evaluation and Implementation. We choose truncated Hit Ratio (HR@K), Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) to measure ranking quality (K=5). Following BERT4Rec [17], we randomly sample 100 negative items according to their popularity for each ground truth item. For all baselines on two datasets, the maximum length of sequences \(n\) is 50; hidden size \(d\) is 128; batch size is 64. We implement all models and tune other hyper-parameters following authors’ suggestions in papers.

### 4.2 Result Analysis

#### 4.2.1 General Performance.** Table 1 shows ranking performance on two datasets. We find that: (1) Previous intent-aware methods that fuse intent features into item representations achieve limited improvements. We think the reason is that baselines did not learn intent representations from intent sequences but from item sequences. However, on such sparse datasets, items hardly provide supervisory information. (2) Compared to global attention (BERT4Rec), local attention (LOCKER) can consistently improve recommendations.

#### 4.2.2 Ablation Study.** To validate the effectiveness of our proposed method, we conduct an ablation study on Tmall dataset; Table 3 shows results. Compared to (1) FDSA which fuses intent features
Table 3: Ablation study on Tmall dataset. (1) fusing intents into item embeddings; (2) modeling intents explicitly; (3) local self-attention of item encoder; (4) inference with joint probability distribution of items and corresponding intents.

| Backbone (SASRec) | + (1) (FDSA) | + (1)(2) | + (1)(2)(3) | + (1)(2)(3)(4) (CAFe) |
|-------------------|-------------|----------|-------------|----------------------|
| NDCG@5            | 0.0741      | 0.1083   | 0.3045      | 0.3159               |
| HR@5              | 0.1205      | 0.1685   | 0.3938      | 0.4066               |
| MRR               | 0.0948      | 0.1265   | 0.3069      | 0.3172               |

Figure 4: Improvement on Amazon compared to BERT4Rec.

into item representations, (2) modeling intents explicitly (i.e., learning intent representations from intent sequences) is critical to make intent representations effective for items. (4) joint probability inference largely improves recommendation performance by providing coarse-grained knowledge during inference. (3) local self-attention can further improve results by focusing on more recent items.

4.2.3 Improvement vs. Sparsity. We investigate CaFe performance compared to BERT4Rec with different dataset sparsity in Figure 4. Models (CaFe and BERT4Rec) are trained and tested on Amazon datasets with different minimum item frequency k. Smaller k means the dataset is sparser. We can see that the improvement is more than 100% on the dataset with k = 1 (original) and the improvement is less than 10% on the dataset with k = 20 (the most dense). The results show the effectiveness of CaFe on sparse datasets.

5 CONCLUSION
Self-attentive recommenders have shown promising results in sequential recommendation. However, from our motivating experiments, we find that existing methods still struggle to learn high-quality item representations from sparse data. In this paper, we introduce a coarse-to-fine framework (CaFe) that explicitly models intent sequences and enhances infrequent item representations by knowledge from intents. Furthermore, we propose to infer recommended items based on joint probability of intents and items. Experimental results show that CaFe significantly improves recommendation performance on sparse datasets.

REFERENCES
[1] 2015. https://ijcai-15.org/repeat-buyers-prediction-competition/. In IJCAI.
[2] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. ArXiv abs/1607.06450 (2016).
[3] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and M. de Rijke. 2020. Improving End-to-End Sequential Recommendations with Intent-aware Diversification. Proceedings of the 29th ACM International Conference on Information & Knowledge Management (2020).
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv abs/1810.04805 (2019).
[5] Robin Devougte and Hugues Bersini. 2017. Long and Short-Term Recommendations with Recurrent Neural Networks. Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (2017).
[6] Kaime He, X. Zhang, Shaoqiang Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016), 770–778.
[7] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. 2016 IEEE 16th International Conference on Data Mining (ICDM) (2016), 191–200.
[8] Zhankui He, Handong Zhao, Zhe Lin, Zhaowen Wang, Ajinkya Kale, and Julian McAuley. 2021. Locker: Locally Constrained Self-Attentive Sequential Recommendation. Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021).
[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Dominoskos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. CoRR abs/1511.06939 (2016).
[10] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. 2018 IEEE International Conference on Data Mining (ICDM) (2018), 197–206.
[11] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021. Intent-aware Sessional Sequential Recommendation with Structured Intent Transition. IEEE Transactions on Knowledge and Data Engineering (2021), 1–1.
[12] Jing Li, Pengjie Ren, Zhanmin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. Proceedings of the 30th ACM Conference on Information and Knowledge Management (2017).
[13] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. Proceedings of the 13th International Conference on Web Search and Data Mining (2020).
[14] Guoqiang Li, Xiaoguang Li, Guoshao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. In AAAI.
[15] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.
[16] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorization Machines. Proceedings of the 26th International Conference on Machine Learning (ICML) (2010), 995–1002.
[17] Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. In AAAI.
[18] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and M. de Rijke. 2020. Improving Sequential Models of Latent Intent for Next Item Recommendation. Proceedings of The Web Conference 2020 (2020).
[19] Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9 (2008), 2579–2605.
[20] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, A. Kaiser, M. Shlens, T. Petke. 2017. Attention is all you Need. ArXiv abs/1706.03762 (2017).
[21] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-FT: Sequential Recommendation Via Personalized Transformer. Fourteenth ACM Conference on Recommender Systems (2020).
[22] An Yan, Chaosheng Dong, Yan Gao, Jinmiao Fu, Tong Zhao, Yi Sun, and Julian McAuley. 2022. Personalized complementary product recommendation. In WWW.
[23] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2020. A Simple Convolutional Generative Network for Next Item Recommendation. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (2019).
[24] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation. In IJCAI.