Leveraging Negative Signals with Self-Attention for Sequential Music Recommendation

Pavan Seshadri

pseshadri9@gatech.edu
Georgia Institute of Technology, Music Informatics Group
Atlanta, Georgia, USA

Peter Knees

peter.knees@tuwien.ac.at
TU Wien, Faculty of Informatics
Vienna, Austria

ABSTRACT

Music streaming services heavily rely on their recommendation engines to continuously provide content to their consumers. Sequential recommendation consequently has seen considerable attention in current literature, where state of the art approaches focus on self-attentive models leveraging contextual information such as long and short-term user history and item features; however, most of these studies focus on long-form content domains (retail, movie, etc.) rather than short-form, such as music. Additionally, many do not explore incorporating negative session-level feedback during training. In this study, we investigate the use of transformer-based self-attentive architectures to learn implicit session-level information for sequential music recommendation. We additionally propose a contrastive-learning task to incorporate negative feedback (e.g skipped tracks) to promote positive hits and penalize negative hits. This task is formulated as a simple loss term that can be incorporated into a variety of deep-learning architectures for sequential recommendation. Our experiments show that this results in consistent performance gains over the baseline architectures ignoring negative user feedback.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Music retrieval**.

KEYWORDS

Sequential Recommendation, Music Recommendation, Self-attention, Contrastive Learning

1 INTRODUCTION

Recommendation systems have become integral to streaming services such as Spotify, Apple Music, Deezer, etc., and by proxy, the music industry as a whole. As the music streaming business model relies on continual user engagement and activity, consistent music discovery is an essential service. Sequential music recommendation is one such task in this domain, where given a current user session (i.e a current sequence of tracks listened to by a user), a system extends the session by recommending the user the next track. Within the music domain, sequential recommendation is generally split into two categories, *next song recommendation (NSR)*, and *automatic playlist continuation (APC)*. These two tasks can be learned in a similar manner from playlist and listening history information, but they differ in output length: APC aims to extend the session or playlist by an arbitrary length, while NSR only aims to provide the next relevant song in sequence [16]. For this study we focus specifically on NSR.

Music recommendation differs from other well-studied domains of recommendation (retail, movies, games, etc.) in a number of important ways. Singular music tracks generally are short and easily consumed, necessitating a thorough understanding of a user's preferences in order to provide both breadth and depth over a large quantity of relevant recommendations [17]. Robust music recommendation systems often leverage previous consumer history to learn user preferences through methods such as collaborative filtering [11]; however these approaches fall victim to the cold start problem [17]: for new users or new tracks, the recommendation model does not have any usable information and must guess preferences until the user and/or track has interacted with the system enough to learn a profile [6].

Sequential recommendation in general can alleviate this issue by learning session-level relationships instead of, or in tandem with, user-level relationships. By learning session item relationships from sequential interaction, item profiles can be rapidly built as they interact with the system as the recommendation engine can compare user sessions directly rather than using aggregate statistics via collaborative filtering, which takes much more data to build robust representations [6].

This study aims to leverage implicit and explicit signals present within listening sessions to learn robust profiles for sequential recommendations. Prior work has considered direct incorporation of user feedback for ad-hoc adjustments based on content and context similarity, e.g. [8, 13]. In this work, we investigate learning session-level information via transformer-based architectures, influenced by SoTA methods for sequential retail recommendation, as well as incorporating user feedback through a learned contrastive task. To our best knowledge, learning from negative signals/user feedback has not been explored thoroughly for sequential music recommendation due to a lack of public data containing thorough user feedback. Many public music recommendation datasets, such as Lastfm-1K [3], were collected before the streaming boom, where logged listening history would primarily be sourced from user creation, leading to a low source of negative signals. For this study, we employ the Music Streaming Sessions Dataset from Spotify [1]. Since many of the interactions present are from programmatic or expert curation, rather than user curation, they can be considered as exploration events where the user reacts positively (listens to track in entirety) or negatively (skips track). This provides a rich amount of negative samples to learn effective session-level representations from.

2 RELATED WORK

Sequential recommendation systems can generally be divided into two types: *session-aware* systems leverage session-level history

from identifiable users, while *session-based* systems ignore user-labels and aim to build user-agnostic representations using solely discrete sessions. [15]. In this study, we investigate a *session-based* system that implicitly learns a user profile through anonymous listening sessions.

Several session-based approaches have been proposed for retail recommendation tasks. CASER [20] and NextItNet [23] leverage convolutional filters to learn sequential representations. BERT4Rec [18] leverages the bidirectional attention mechanism from BERT [4] to learn a robust vocabulary of items for sequential recommendation.

Several sequential based approaches have been proposed for music recommendation tasks incorporating a variety of information to drive recommendation [16]. Most of such approaches leverage contextual and/or content features, largely by extensive user profiles and music tags. Relevant work for these respective approaches include CoSERNN [5], and Online Learning to Rank for Sequential Music Recommendation [14]. The former leverages contextual information such as device used, time of day during recommendation, etc. to drive contextual user-sequential embeddings for sequential recommendation, while the latter leverages content features via music tags for an online learning to rank scheme. For a study closest to our task, Wen et. al investigate leveraging implicit user feedback immediately after click for video and music recommendation, and find performance gains incorporating this information into a variety of recommendation approaches [22]. Most state-of-the-art sequential music recommendation approaches leverage several types of information that often are not present in public datasets (e.g lyrics, user contextual/demographic information, music tags, etc.). It would be increasingly difficult to re-implement and test these systems in a cold-start or academic setting due to the amount and variety of data required. Our approach aims to alleviate this data issue by taking advantage of implicit relationships from data present solely in listening sessions of songs, namely item labels and timestamps of user events. We additionally do not take into account long-term user history due to a lack of user labels; Thus, we focus on creating a *session-based* system.

3 METHOD

3.1 Problem Statement

In our scenario, we define a session S of length K and set of possible tracks $t \in T$ for user u. Track t_i, where $t_{1,2,...K} \in S$ represents the track at each time step i in session S, where $i \in [1...K]$. Generally, the task of a sequential recommendation system is to predict the desired next item h_i at time step $i + 1$ for each $t_i \in S$, given an interaction history S_i, where $S_i = \{t_a \in S \mid a \leq i\}$.

For negative feedback-agnostic sequential recommendation systems (i.e where the user has not explicitly responded negatively to any item), we define h_i for track t_i as the next track in the sequence, t_{i+1}.

For our feedback-aware system, we define the set of positive examples (no-skip) as P and negative examples (skipped tracks) as N per sequence S, such that:

$$p_j \in P, n_k \in N, \text{ and all } p_j, n_k \in S$$

where j, k correspond to the time step of each example in session S. Additionally for clarity, we define I_P and I_N as the set of time

steps for all positive and negative examples, respectively, where $j \in I_P$ and $k \in I_N$. For any track t_i, we define the desired next track h_i as the next positive example in the session, p_m, such that:

$$m = \min_j \{j \in I_P \mid j > i\}$$

Where the difference $m - i$ represents the number of skipped tracks between track t_i and its next positive sample.

To predict the desired next track at time step i, we model a probability distribution $p(h_i = t \mid S_i)$ over all possible tracks. Sorting this distribution provides a ranking of the most-relevant items. By learning from negative feedback, we aim to both raise the ranking of p_m, as well as lower the rankings of items in N in predicting each h_i.

3.2 Model Architecture

We investigate unidirectional and bidirectional transformer-based architectures in this study, inspired by the SASRec [9] and BERT4Rec [19] architectures, respectively. For both approaches we use the same base architecture described below, with the sole differences being the training procedure, learning objective, and the use of a causal attention mask in the case of the unidirectional model. We keep the implementation analagous to that of the aforementioned authors for better comparison.

3.2.1 Track Embeddings. We store learned track embeddings in a lookup-table $e_t \in E$ of size $T \times \mathbb{R}^{|d|}$, where T is the number of tracks and d is the embedding dimensionality. $E(\cdot)$ denotes the function retrieving the embeddings of a track or set of tracks from table E.

3.2.2 Positional Embeddings. To inject information about the position of each track in the sequence, we add a learnable positional embedding PE of size $K \times \mathbb{R}^{|d|}$ to each track embedding in the sequence, where K corresponds to the size of the sequence.

3.2.3 Encoder. We employ a standard transformer encoder to learn contextual session-level information. This is a fully attention based model employing multiple multi-head self-attention layers and position-wise feedforward layers to learn contextual information from sequential inputs.

3.2.4 Prediction Layer. After obtaining hidden vectors from the encoder with contextual information, we project them through a fully connected layer with GELU activation [7] to obtain predicted embeddings $\hat{y}_i$ for each $t_i \in S$. We then compute an inner product with the embedding table and apply a sampled softmax to get a probability distribution over each track.

3.2.5 Sampled Softmax. Additionally for training stability with such a large amount of classes ($\sim$ 1M tracks in this study), we employ a sampled softmax function during training. For each mini-batch for each session, we uniformly sample 1000 unseen tracks and rank the target tracks alongside these. These 1000 tracks are re-sampled each epoch, such that as training continues, the model continually learns to "rank" the target items with an increasing subset of the total tracks, as the number of unique tracks sampled for comparison increase.

3.3 Sequential Recommendation Task

For both approaches, we employ the same learning objective, the negative log likelihood (NLL), for training; however they differ in how this learning objective is used.

3.3.1 Unidirectional.
We employ the *next-item prediction* task for this approach. For each $t_i \in S$, we task the model with predicting the next item in the sequence, t_{i+1}. We then compute log-probabilities and pass this to the NLL Loss. Additionally, attention maps are computed using a causal mask, preserving the auto-regressive nature of unidirectional transformers.

3.3.2 Bidirectional.
We employ the *cloze*, or *masked language modelling (MLM)* task for this approach. We randomly mask a proportion p of each sequence with a special token [MSK] and task the model with predicting what the correct track is at these indices with a bidirectional attention map. For the sequential recommendation task, we also append the [MSK] token to the end of the sequence and set the target of this to the last track in the session targets, to ensure that this target does not appear in the bidirectional attention map.

3.4 Skip-informed Contrastive Task

To learn negative sequential track relationships, we employ a contrastive learning task using the skipped tracks in each listening session. We employ noise contrastive estimation with InfoNCE [21] shown below:

$$\mathcal{L}_{NCE} = -\mathbb{E}_X \left[\log \frac{f_k(\mathbf{p}, \mathbf{c})}{\sum_{x_j \in X} f_k(x_j, \mathbf{c})} \right]$$

Given a context vector c, positive anchor p and set of noise samples $x \in X$, this loss term uses a categorical cross entropy to classify the positive anchor from the set of noise samples, given scoring function $f_k(\mathbf{x}, \mathbf{c})$.

For each track $t_i \in S$, we adapt this to our task of promoting the next true positive sample p_m and penalizing all negative samples $n_j \in N$ by defining the following:

(1) $c = e_{t_i}$ **or** $\hat{y}_i$
(2) $X = E(N)$
(3) $p = E(p_m)$
(4) $f_k(\mathbf{x}, \mathbf{c}) = \dfrac{\mathbf{x} \cdot \mathbf{c}}{\|\mathbf{x}\|_2 \|\mathbf{c}\|_2}$

This maximizes the cosine similarity between the embedding e of track t_i and next-positive-sample p_m while minimizing the similarity between e_{t_i} and all $e_n \in E(N)$. Since during prediction, logits are computed by the inner product of $\hat{y}_i$ and E, this directly affects the rankings of p_m and all $n \in N$, by drawing t_i and p_m closer together in the learned embedding space, and consequently pushing t_i and all $n \in N$ farther away in the embedding space. We experiment with setting the context vector c as both $\hat{y}_i$ and e_{t_i}. Setting $c = \hat{y}_i$ includes the current session context, while setting $c = e_{t_i}$ ignores current session context and instead relies solely on the overall learned representation of the track. We explore both to examine the the extent to which immediate context and contextual history affect the learning of negative preference, respectively.

3.5 Dataset

For this study we use the Music Streaming Sessions Dataset (MSSD) [1] for training and evaluation, which contains 160 Million user sessions of 10 to 20 consecutively listened songs (<60 seconds between listens). These listening sessions are uniformly sampled from a variety of contexts, such as the user's personally curated collections, expertly curated playlists, contextual non-personalized recommendations, and personalized recommendations.

Notably, this dataset is pseudonymized, meaning all included sessions lack a user label. Consequently, we treat each session as a new user, ignoring long term history.

Skip labels are provided for each track in each session with strength 1-3, defined per the authors as the track "played very briefly", "played briefly", and "played mostly (but not completely)", respectively. For this study, we are primarily interested in strong negative interactions and therefore only consider tracks with skip strength 1 and 2 as negative examples in each session.

Due to time and computational restraints, we uniformly sample ~450K discrete sessions containing ~2 million item interactions with ~1 million total unique tracks to train and evaluate our models. We note that our subset of sessions contains roughly 15% skipped tracks.

3.6 Training Procedure

As with other contrastive recommendation systems [2, 24], we simply aggregate the sequential task loss and the contrastive shown below within one single training pass

$$\mathcal{L} = \alpha \mathcal{L}_{NCE} + \beta \mathcal{L}_{NLL}$$

where α and β are scalar terms. We empirically tune these parameters through the validation set.

3.7 Hyperparameters and Implementation

As our data contains variable length sessions between 10 and 20 interactions, we pad all sessions to length 20. We stack 2 encoder blocks with 8 attention heads. The embedding and hidden dimensions are both set to 128. Masking for the bidirectional model is applied per batch with proportion $p = 0.2$. We initialize all parameters via truncated normal sampling with $\mu = 0, \sigma = 1$ in range $[-0.02, 0.02]$. We tune the optimal $\alpha, \beta \in [.25, .5, .75, 1]$ using the validation set and select $\alpha = 0.5, \beta = 0.5$. We use the ADAM optimizer [10] with a learning rate of 0.005, selected after tuning through the validation set with $lr \in [0.0001, 0.0005, 0.001, 0.005]$. All models were implemented in python using pytorch-lightning and trained using an NVIDIA RTX 2070 GPU.

4 RESULTS AND DISCUSSION

4.1 Evaluation

We employ the next-item recommendation task used by [9, 19] for our evaluation. For each sequence, we leave out the final and penultimate items as the testing and validation targets, respectively and reserve the rest of the sequence for training. For each target, we uniformly sample 1000 unobserved tracks, where the task becomes to rank the target among these tracks. We employ the Hit Rate@K

	Unidir S-	Unidir S+ ($c = \hat{y}_i$)	Unidir S+ ($c = e_{t_i}$)	Bidir S-	Bidir S+ ($c = \hat{y}_i$)	Bidir S+ ($c = e_{t_i}$)
HR@ 1 (% increase)	.2821	.3049 (8.08%)	**.3073 (8.93%)**	.1945	.2101 (8.02%)	**.2339 (20.26%)**
HR@ 5 (% increase)	.4803	.5042 (4.98%)	**.5088 (5.93%)**	.4203	**.4370 (3.97%)**	.4358 (3.69%)
HR@ 10 (% increase)	.5593	.5836 (4.34%)	**.5867 (4.90%)**	.5135	**.5270 (2.63%)**	.5165 (0.58%)
HR@ 20 (% increase)	.6316	.6550 (3.70%)	**.6554 (3.77%)**	.5989	**.6101 (1.87%)**	.5911 (-1.30%)

Table 1: Hit Ratio @ [1, 5, 10, 20] for the unidirectional and bidirectional approaches. Baselines without skip-informed contrastive learning (S-) are compared to the proposed contrastive learning settings (S+, using either session context $\hat{y}_i$ or track representation vector e_{t_i} as context vector c). Numbers in parentheses show the relative increase in percentage of the approach over the respective baseline; bold entries mark the best performing approach within unidirectional and bidirectional architectures.

(equivalent to recall) as our evaluation metric, with $k \in [1, 5, 10, 20]$. The results are shown in Table 1.

4.2 Discussion

We note a number of observations from our experiments. Namely:

(1) The skip-informed contrastive task consistently outperforms the feedback-agnostic models, indicating that learning from negative feedback is beneficial for sequential music recommendation

(2) The unidirectional models consistently outperform the bidirectional models, with a waning performance gap as the top-K for the hit rate increases.

(3) Using the final hidden state $\hat{y}_i$ with immediate contextual information as the context vector for the contrastive task performs similarly but consistently slightly worse than using the item embeddings.

Overall, we observe that our contrastive task reliably increases the hit rate in a next-item recommendation scenario, with the exception of the HR@20 for the bidirectional model using only track embeddings. Interestingly, even though we create a mismatch between the targets for the sequential recommendation task and contrastive task, the hit rate for the sequential recommendation task increases, inferring that optimizing for the next positive example (p_m) and next track (t_{i+1}) in tandem raises the performance in selecting the next track during inference.

We also observe waning performance gains as the number of tracks in the ranking window increases, likely due to the fact that the contrastive task only relates observed tracks with each other. As the amount of unobserved tracks in the comparison increases (i.e HR@1 to HR@20), the weaker the effect of the contrastive task. Our experiments imply that the effect of learning from negative feedback in this fashion mostly affect the top ranked recommendations.

The relatively weak performance of the BERT-like architecture may be due to the relative high density of our dataset and our relatively short sequence lengths, so training in an autoregressive manner with each sample in the training sequence per each epoch may be better for learning latent sequential track relationships. More work is likely needed to find an optimal setup using bidirectional attention with the MLM task.

The slight performance improvement when using the track embeddings as the contextual vector for the contrastive task may imply that while immediate session-level contextual information is useful

in learning from negative feedback, reducing this emphasis may provide a slightly stronger signal for preference of a user's next desired track.

5 CONCLUSION AND FUTURE WORK

Overall, we have presented both a study on the use of transformer-based architectures for sequential music recommendation, as well as a contrastive-based task to learn from negative feedback. We show through our experiments that the contrastive task results in greater hit rate on both unidirectional and bidirectional architectures. Multiple avenues for future work arise, namely the inclusion of long-term user profiles for better modelling of long term and changing user-taste. Additionally, contextual and content information can be injected into the embeddings to learn more powerful contextual representations. An analysis of the performance on different session types and streaming behaviors (playlist, auto-generated, user-curated, etc.) [12] would also provide better insight into the performance in different listening contexts.

ACKNOWLEDGMENTS

REFERENCES

[1] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2594–2600. https://doi.org/10.1145/3308558.3313641

[2] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation.

[3] O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail.* Springer.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

[5] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/3383313.3412248

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. https://doi.org/10.48550/ARXIV.1708.05031

[7] Dan Hendrycks and Kevin Gimpel. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR* abs/1606.08415 (2016). arXiv:1606.08415 http://arxiv.org/abs/1606.08415

[8] Yajie Hu and Mitsunori Ogihara. 2011. NextOne Player: A Music Recommendation System Based on User Behavior.. In *Proceedings of the 12th International*

Society for Music Information Retrieval Conference. ISMIR, Miami, United States, 103–108. https://doi.org/10.5281/zenodo.1418301

[9] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018.* IEEE Computer Society, 197–206. https://doi.org/10.1109/ICDM.2018.00035

[10] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[11] Peter Knees and Markus Schedl. 2013. A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 10, 1 (2013), 1–21.

[12] Francesco Meggetto, Crawford Revie, John Levine, and Yashar Moshfeghi. 2021. On Skipping Behaviour Types in Music Streaming Sessions. In *Proceedings of the 30th ACM International Conference on Information Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21).* Association for Computing Machinery, New York, NY, USA, 3333–3337. https://doi.org/10.1145/3459637.3482123

[13] Elias Pampalk, Tim Pohle, and Gerhard Widmer. 2005. Dynamic Playlist Generation Based on Skipping Behavior.. In *Proceedings of the 6th International Conference on Music Information Retrieval.* ISMIR, London, United Kingdom, 634–637. https://doi.org/10.5281/zenodo.1414932

[14] Bruno L. Pereira, Alberto Ueda, Gustavo Penha, Rodrygo L. T. Santos, and Nivio Ziviani. 2019. Online Learning to Rank for Sequential Music Recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19).* Association for Computing Machinery, New York, NY, USA, 237–245. https://doi.org/10.1145/3298689.3347019

[15] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-Based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17).* Association for Computing Machinery, New York, NY, USA, 130–137. https://doi.org/10.1145/3109859.3109896

[16] Markus Schedl. 2019. Deep Learning in Music Recommendation Systems. *Frontiers in Applied Mathematics and Statistics* 5 (2019). https://doi.org/10.3389/fams.2019.00044

[17] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. 2015. Music recommender systems. *Recommender systems handbook* (2015), 453–492.

[18] 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. https://doi.org/10.48550/ARXIV.1904.06690

[19] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 1441–1450. https://doi.org/10.1145/3357384.3357895

[20] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. https://doi.org/10.48550/ARXIV.1809.07426

[21] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). arXiv:1807.03748 http://arxiv.org/abs/1807.03748

[22] Hongyi Wen, Longqi Yang, and Deborah Estrin. 2019. Leveraging Post-Click Feedback for Content Recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19).* Association for Computing Machinery, New York, NY, USA, 278–286. https://doi.org/10.1145/3298689.3347037

[23] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19).* Association for Computing Machinery, New York, NY, USA, 582–590. https://doi.org/10.1145/3289600.3290975

[24] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21).* Association for Computing Machinery, New York, NY, USA, 3985–3995. https://doi.org/10.1145/3447548.3467102