Why One Size Doesn't Fit All: Improving Music Discovery and Familiar Listening with Specialized Models

Mahesh Jindal
jindsmah@amazon.com
Audible
Newark, NJ, USA

Giuseppe Di Benedetto
bgiusep@amazon.com
Amazon Music
Berlin, Germany

Yannik Stein
syannik@amazon.de
Amazon Music
Berlin, Germany

Abstract

Collaborative filtering is a foundational component of music recommender systems, powering a variety of recommendation tasks from retrieving the most relevant tracks, albums, artists, and podcasts for a given user to more nuanced objectives such as content discovery, familiar listening, and new release recommendation. To enable scalable, low-latency inference, content-retrieval models compute latent user and item representations. These embeddings can be efficiently queried using approximate nearest-neighbor indices. A fundamental challenge in music recommendation systems is the catalog scale, with music streaming platforms commonly offering more than 100 million tracks. Training item embeddings at this scale presents significant operational challenges, leading to the prevalent industry practice of training a single, general-purpose content-retrieval model and adapting it to specific tasks through post-hoc filtering. For instance, discovery tasks apply filtering to exclude previously consumed content, while new release recommendations filter by recency.

We challenge this practice by quantifying substantial performance improvements of task-specific models over the standard approach of using a single general-purpose model with post-hoc filtering, with gains of up to 194.6% in NDCG@100 for discovery tasks and up to 122.4% for fine-tuned models. To mitigate the training costs that motivate the industry practice in the first place, we explore an intermediate approach: fine-tuning a foundational content-retrieval model on specific recommendation tasks. While fine-tuned models do not achieve the same performance as fully task-specific models, they consistently outperform the single embedding model approach at comparable training costs.

CCS Concepts

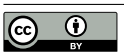
• **Information systems** → **Personalization**; **Recommender systems**; **Top-k retrieval in databases**; **Retrieval effectiveness**.

Keywords

Sequential recommendation, Music discovery, SASRec, Collaborative filtering

ACM Reference Format:

Mahesh Jindal, Giuseppe Di Benedetto, and Yannik Stein. 2026. Why One Size Doesn't Fit All: Improving Music Discovery and Familiar Listening with Specialized Models. In *The Nineteenth ACM International Conference on Web Search and Data Mining (WSDM Companion '26), February 22–26, 2026, Boise, ID, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3779211.3793165



WSDM Companion '26, Boise, ID, USA

ACM ISBN 979-8-4007-2358-2/2026/02
https://doi.org/10.1145/3779211.3793165

1 Introduction

Modern recommender systems typically employ a two-stage architecture: content retrieval and ranking [20, 21]. In the content retrieval stage, a highly scalable model identifies a subset of potentially relevant items from the catalog. In the ranking stage, the result set is then ordered using a more sophisticated model that can afford higher computational complexity due to the reduced candidate space [12].

Scalable collaborative filtering methods for content retrieval rely on learned embeddings for users and items, where proximity in the embedding space indicates relevance. The embeddings enable low-latency inference through approximate nearest-neighbor indices [7, 10, 17]. However, learning item embeddings at this scale requires sophisticated distributed training algorithms and incurs substantial computational costs [3]. Due to the high training costs, industry practitioners commonly train a single general-purpose collaborative-filtering model and reuse it across multiple recommendation tasks by applying post-hoc filters. For example, an artist discovery recommender could be implemented by performing a nearest-neighbor query with the user embedding and then filtering results to exclude artists the user has previously listened to. Many recommendation tasks can be similarly decomposed into content retrieval followed by task-specific filtering [6]. While this effectively amortizes the training costs across the recommendation tasks, we quantify the substantial opportunity cost of this practice, measuring performance gaps ranging from 2.7% to 194.6% in NDCG@100 across multiple music recommendation tasks when compared to models that are trained on-task. This performance gap is particularly critical for high-value recommendation tasks such as content discovery, which directly impact long-term user satisfaction [4, 16].

The key contributions of this paper are:

(1) We define and formalize four music recommendation tasks on academic datasets beyond standard user-item retrieval.
(2) We quantify substantial performance gaps between general-purpose collaborative-filtering models with task-dependent post-hoc filtering and dedicated task-specific models, measuring improvements of 43 – 195% in NDCG@100 for discovery tasks and 3 – 53% for familiar listening tasks.
(3) To mitigate training costs, we measure that fine-tuning a foundational content-retrieval model achieves 66 – 139% improvements over baseline models while offering comparable

training costs, though underperforming fully task-specific models by 20 − 50% on discovery tasks.

2 Related Work

Collaborative-filtering models, such as Alternating Least Squares (ALS), are a foundational building block of modern recommender systems; see [13] for a review. In sequential recommendation, we additionally take into account the order of user interactions and try to predict the next one. With the advent of transformer architecture in NLP [19], several transformer-based models for sequential recommendation have been proposed. Among the first were SASRec [7], which follows a decoder-only architecture, and BERT4Rec [17], which uses an encoder-only architecture.

However, these approaches focus primarily on general user-item relevance prediction. Modern streaming platforms need to solve a variety of recommendation tasks: for example, content discovery has been extensively studied [1, 14, 15, 18] because of its positive long-term impact [4, 16].

Most relevant to our approach are [18] and [9]. In [18], the authors consider repeated listening and discovery as separate recommendation tasks and propose a new model architecture to learn both through short and long-term user representations. Nevertheless, a single model is employed for both tasks, differing from our proposed approach. In [9], a generative model is learned to predict not only the next item that a user interacts with as in sequential recommendation, but also the type of interaction (called *behavior* in the paper). Here, behavior could for example be discovery or familiar listening. In both [18] and [9] multiple downstream recommendation tasks are considered, yet the proposed solutions train a single model for all tasks. In contrast, our work quantifies the performance cost of single-model approaches by measuring improvements of +43% to +195% in NDCG@100 when using dedicated task-specific models rather than training a single multi-task model.

3 Methodology

In order to show the hidden opportunity costs of using a general-purpose embedding model with recommendation task-specific filtering, we define several music recommendation tasks on a dataset of user listening histories. For a user u, let $\mathcal{H}_T(u) = (t_1, t_2, ..., t_n)$ denote the ordered *track* listening history of u. Similarly, let $\mathcal{H}_A(u) = (a_1, ..., a_n)$ denote the ordered *artist* listening history such that a_i is the main artist of track t_i. Then, we define the following tasks:

- **T1 Track Discovery:** Given $\mathcal{H}_T(u)$, predict the next track t^* that the user listens to with $t^* \notin \mathcal{H}_T(u)$.
- **T2 Familiar Track Listening:** Given $\mathcal{H}_T(u)$, predict the next track t^* that the user listens to with $t^* \in \mathcal{H}_T(u)$.
- **T3 Artist Discovery:** Analogous to **T1**, given $\mathcal{H}_A(u)$, predict the next artist a^* that the user listens to with $a^* \notin \mathcal{H}_A(u)$.
- **T4 Familiar Artist Listening:** Analogous to **T2**, given $\mathcal{H}_A(u)$, predict the next artist a^* that the user listens to with $a^* \in \mathcal{H}_A(u)$.

A general-purpose embedding model can be trained on the track and artist listening histories and used for all tasks by filtering its predictions to either tracks (artists) that the user previously did not listen to in the case of **T1** (**T3**), or to already known tracks (artists) in the case of **T2** (**T4**).

Next, we discuss how to train specialized models for the individual tasks. We define the user track discovery sequence $\mathcal{D}_T(u) = (t_i : t_i \notin \mathcal{H}_T^{<i}(u))$ as the ordered sequence of tracks that the user previously did not listen to, where $\mathcal{H}^{<i}$ denotes the first $i-1$ tracks in the user listening history. Similarly, the familiar track sequence is defined as $\mathcal{F}_T(u) = (t_i : t_i \in \mathcal{H}_T^{<i}(u))$, and $\mathcal{D}_A(u), \mathcal{F}_A(u)$ are defined analogously for artists. Now, we can train a specialized model for **T1** on the track discovery sequence $\mathcal{D}_T(u)$. Similarly, for **T2** we can train a model on $\mathcal{F}_T(u)$, for **T3** on $\mathcal{D}_A(u)$, and for **T4** on $\mathcal{F}_A(u)$.

Since training costs of content retrieval are substantial at the catalog scale of music streaming platforms, we also consider the fine-tuning paradigm in order to obtain specialized models for the individual tasks without having to train a model for each task from scratch. Here, we further fine-tune the general-purpose models on the discovery and familiar listening sequences of the respective tasks.

As an additional variant to the task-specific discovery and familiar listening sequences, we propose a new training data preprocessing transformation: *listening history condensation*. We say a sequence S is *k-condensed* if all elements in S appear at least k times in the listening history. The listening history condensation reduces noise and only keeps tracks or artists that the user repeatedly engages with. Specifically for the above tasks, a k-condensed track discovery sequence is defined as $(t_i \in \mathcal{D}_T(u) : n_i \geq k)$, where n_i is the number of occurrences in $\mathcal{H}_T(u)$. Similarly, k-condensed familiar track sequences as well as k-condensed familiar and discovery artist sequences can be defined.

So far we have not discussed the specifics of the collaborative filtering models. Our approach outlined above does not make any assumptions about the model architecture itself. We select Alternating Least Squares (ALS) [5] as a commonly used strong collaborative-filtering baseline, and gSASRec [10] as a transformer-based sequential model for content retrieval to assess the impact of the approach across different model types. Before we proceed with the experimentation, we briefly outline ALS and gSASRec.

3.1 Alternating Least Squares

Alternating Least Squares (ALS) [5] decomposes the user-item interaction matrix into lower-dimensional user and item latent factor matrices through an iterative optimization process in which the latent user representation is optimized while fixing latent item representations, and vice versa. The optimization objective minimizes the reconstruction error of the interaction matrix while applying regularization to prevent overfitting:

$$\min_{X,Y} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mathbf{x}_u^\top \mathbf{y}_i)^2 + \lambda(\|\mathbf{x}_u\|^2 + \|\mathbf{y}_i\|^2) \tag{1}$$

where r_{ui} is the implicit feedback from user u to item i, $\mathbf{x}_u, \mathbf{y}_i$ are latent factors for user u and item i respectively, and λ is a regularization parameter.

3.2 Generalized Self-Attention for Sequential Recommendation (gSASRec)

The gSASRec [10] model follows a decoder-only architecture identical to SASRec [7]. The order of the input sequence is encoded

through positional embedding, and the models are trained to predict the next item in the input sequence. We train the general-purpose model using gSASRec to predict the next track (artist) in the user listening history, and the task-specific models to predict the next track (artist) in the discovery (familiar) sequence depending on the task. The only difference from gSASRec to SASRec is the loss function: in SASRec, binary-cross entropy (BCE) with sampled negatives is used to allow for large catalog sizes. In the gSASRec paper, it is shown that this can lead to overconfidence and a generalized BCE loss function is introduced to calibrate:

$$\mathcal{L}^{\beta}_{\text{gBCE}} = -\frac{1}{|I_k^-|+1}\left(\log\left(\sigma^{\alpha\left(t\left(1-\frac{1}{\alpha}\right)+\frac{1}{\alpha}\right)}(s_{i^+})\right) + \sum_{i\in I_k^-}\log\left(1-\sigma(s_i)\right)\right) \tag{2}$$

where s_{i^+} is the score of the next item in the listening history, I_k^- is the set of k randomly sampled items from the catalog, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and $\alpha, t \in [0, 1]$ are hyperparameters.

4 Experiments

We aim to answer the following research questions:

- **R1.** By how much do task-specific models outperform post-hoc filtering on general-purpose collaborative-filtering models, and when is the performance gain most substantial?
- **R2.** What is the impact of listening history condensation on task-specific models?
- **R3.** How do fine-tuned general-purpose collaborative-filtering models compare to task-specific models?
- **R4.** Are tasks T1–T4 sequential recommendation tasks?

4.1 Datasets

We test the performance gap between general-purpose collaborative filtering models with post-hoc filtering and task-specific models on two music recommender datasets: Deezer [18] and Yambda-5B [11]. We applied the following additional post-processing steps:

(1) Popularity Threshold: Tracks were included in the final dataset only if they had been consumed by at least 50 unique customers within the most recent 175 customer interactions. This inclusion criterion ensures the exclusion of infrequently consumed tracks that would otherwise provide insufficient behavioral signals for robust model evaluation. **(2) Yambda-Specific Listening Consumption Filter:** For tracks from the Yambda dataset, we imposed an additional constraint requiring a minimum listening consumption rate of 20%. This ensures that the evaluation considers only tracks with meaningful engagement, excluding those where users exhibited minimal listening behavior. **(3) Sequence Length Filter:** Listening history sequences were filtered to include only those user requests with a minimum length of 3 entries, while the maximum sequence length was capped at 10,000 entries.

The postprocessed deezer dataset consists of listening histories from 3, 443, 895 users with 31, 550 unique tracks and an average listening history length of 204. Similarly, the postprocessed Yambda-5B dataset consists of listening histories from 915, 937 users, but its average listening history length of 2, 998 is substantially larger. It contains both track and artist information, with 160, 362 unique tracks and 25, 741 unique artists. We split both datasets into train, validation, and test at user-level using a 90-5-5 split ratio. Please see Table 1 for dataset statistics. Tasks T1 and T2 are defined on both datasets, while T3 and T4 are only defined on Yambda-5B since Deezer lacks artist information.

Statistic	Deezer	Yambda 5B
Number of users	3, 443, 895	915, 937
Number of tracks	31, 550	160, 362
Number of artists	N/A	25, 741
Avg sequence length	204	2, 998
Max sequence length	6, 710	10, 000

Table 1: Dataset Statistics

4.2 Models

We use ALS [5] as representative for non-sequential collaborative filtering methods and gSASRec [10] as representative for sequential recommendation approaches. We train both models on the track user listening histories to predict user-track relevance for Deezer, resulting in a general-purpose baseline models for tasks T1 and T2. On Yambda-5B, we train both models on the track listening history to obtain baseline models for T1 (track discovery) and T2 (track familiar listening), as well as on the artist listening history to obtain baseline models for T3 (artist discovery) and T4 (artist familiar listening), resulting in 4 baseline models for tasks T1–T4. In case of ALS, we learn to predict whether a track or artist appears in the listening history (at any position), whereas in gSASRec, we learn to predict whether a track or artist will be listened to next given the previous listening history by taking the order into account.

We performed hyperparameter optimization (HPO) on the baseline models using Optuna [2] search algorithm and reused these hyperparameters for the task-specific models. For ALS, we used rank (latent dimension) = 20, maximum number of iterations = 15, and regularization parameter $\lambda = 0.1$. For the hyper-parameters of gSASRec, please refer to Table 2.

Hyperparameter	Value
Embedding dimension	128
Number of transformer layers	2
Number of self attention heads	1
Intermediate FFN layer size	128
Dropout rate	0.09495572694179173
Weight decay	0.35520376809082366
Learning rate	0.0031804580753595846
Batch size	512
Maximum number of epochs	100
Number of negative samples	25
GBCE α	0.17112727113208503
GBCE t	0.5559143804253861

Table 2: Hyper-parameters for gSASRec

4.3 Evaluation

We split the historical sequence $\mathcal{H}(u)$ for each user into train and test partitions. For all tasks, we first filter items from the user

listening history according to the task domain and then take the last 10 filtered items to create the ground truth set for evaluation. The remaining partition of sequence i.e $\mathcal{H}^*(u)_{|\mathcal{H}^*(u)|-10}$ is used for training. For T1, the ground truth set consists of 10 most recent track discoveries (i.e., tracks that the user did not listen to before). For T2, the ground truth set consists comprises of 10 most recent listens to familiar track (i.e., tracks that the user previously listened to) within their listening history. Ground truth sets for T3 and T4 are defined similarly on the artist level. We use the listening history up until the last 10 items as model input and predict 100 items (tracks in case of T1 and T2, artists in case of T3 and T4). We evaluate the quality of the predicted items using NDCG@100 and Recall@100.

4.4 R1. By how much do task-specific models outperform post-hoc filtering on general-purpose collaborative-filtering models, and when is the performance gain most substantial?

Table 3 quantifies the performance differences between task-specific and general-purpose models. For gSASRec on Deezer, on-task training achieves NDCG@100 improvements of +194.61% in T1 (track discovery) and +53.04% in T2 (familiar track listening) over next-track prediction training with post-hoc filtering. These substantial gains nearly triple the performance on discovery tasks and demonstrate the high opportunity cost of the standard industry approach. The Yambda-5B dataset confirms these findings with NDCG@100 improvements of +43.26% (T1) and +2.68% (T2) for gSASRec when trained on-task. Results for T3 and T4 in Table 4 show even larger gains: NDCG@100 improvements of +111.49% (T3, artist discovery) and +6.91% (T4, familiar artist listening). Across all discovery tasks (T1, T3), the performance gains range from +43% to +195% in NDCG@100. While we observe improvements for discovery tasks (T1, T3) and familiar listening tasks (T2, T4), on-task training has a stronger effect on discovery tasks. Since music listening histories are dominated by familiar listening, this indicates that a model trained on the listening history overfits on the data-rich tasks (T2, T4) while underfitting on the data-sparse tasks T1 and T3.

Interestingly, for ALS, a non-sequential collaborative-filtering model, we do not observe the same effect: on-task learning does not consistently yield improvements. In tasks T1 (Deezer), T3, and T4 NDCG@100 improves ranging from +0.35% to +1.53%, whereas in tasks T1 (Yambda 5B) and T2 NDCG@100 decreases ranging from −0.11% to −18.77%. We attribute this performance difference to the sequential nature of the tasks and give further evidence on this when discussing **R4.**.

4.5 R2. What is the impact of listening history condensation on task-specific models?

From the results in tables 3 and 4, we observe for gSASRec that listening history condensation can have a strong positive impact on NDCG@100 with improvements up to +21.38% (T2, on task (5), Deezer) when compared to on-task training. It can however also decrease performance when the condensation parameter is too high: −4.54% for T1 on task (5) on Deezer. For ALS, listening history condensation has a negative impact across all tasks. This suggests that listening history condensation can be an impactful preprocessing technique for sequential models but its parameters k need to be optimized as a hyperparameter. The experimental results for gSASRec seem to suggest a unimodal relation, with NDCG@100 improving with k until an optimal value is reached and then decreasing again, but this warrants further investigation as future work.

4.6 R3. How do fine-tuned general-purpose collaborative-filtering models compare to task-specific models?

Tables 3 and 4 quantify the performance of fine-tuning as an intermediate approach between baseline models and fully task-specific models. For discovery tasks T1 and T3, fine-tuning achieves NDCG@100 improvements of +66% to +139% over baseline models. This substantially improves over post-hoc filtering but underperforms fully task-specific models by 20% to 50%. This quantifies the cost-performance tradeoff: fine-tuning captures 70-80% of the maximum possible gain at a fraction of the training cost. For familiar listening tasks, fine-tuning achieves even stronger results: NDCG@100 improvements of +26% to +122% over baseline models, yielding the best or second-best performing models on T2 and T4. We hypothesize that since music listening histories are dominated by familiar listening behavior (60-70% of interactions), the pre-training task aligns closely with T2 and T4, where discovery data acts as a regularizer. For discovery tasks T1 and T3, the baseline model's overfit on familiar listening quantifies to a $20-50\%$ performance penalty relative to models trained from scratch.

4.7 R4. Are tasks T1–T4 sequential recommendation tasks?

To determine whether T1–T4 are sequential recommendation problems, we apply the methodology from [8] to the on-task-trained gSASRec model: at test time, we measure the impact of shuffling the input sequences uniformly at random. Essentially, this is a form of permutation feature importance for the positional encoding in the gSASRec model. Please see Table 5 for the results. Across all tasks, we see a strong negative impact on both NDCG@100 and Recall@100 when shuffling the input sequence, ranging from −19.54% (T4) to −57.56% (T1 Deezer). This indicates that the trained model heavily depends on the order in which users discover content (tasks T1 and T3) or listen to familiar content.

5 Conclusion

Industrial recommender systems often use a single embedding space for multiple recommendation tasks by applying post-hoc filtering on the target task domain. We have quantified the opportunity cost of this practice through extensive experimentation across four music recommendation tasks on two datasets. Our key quantitative findings are: (1) task-specific sequential models outperform general-purpose models with post-hoc filtering by +43% to +195% in NDCG@100 for discovery tasks and +3% to +53% for familiar listening tasks; (2) listening history condensation provides additional

Training	T1: Track Discovery				T2: Track Familiar Listening			
	ALS		gSASRec		ALS		gSASRec	
	N@100	R@100	N@100	R@100	N@100	R@100	N@100	R@100
Dataset: Deezer								
LH-track	0.1295	0.2816	0.09938	0.16526	0.37582	0.54421	0.19752	0.30076
on task	0.12995 +0.35%	0.28092 -0.24%	<u>0.29277</u> +194.61%	**0.40121** +142.77%	0.30529 -18.77%	0.36773 -32.43%	0.30229 +53.04%	0.38972 +29.58%
on task (3)	0.0762 -41.16%	0.1996 -29.12%	**0.2962** +198.05%	<u>0.3976</u> +140.58%	0.24901 -33.74%	0.2356 -56.71%	0.33212 +68.14%	0.39872 +32.57%
on task (5)	0.0568 -56.14%	0.1606 -42.97%	0.27947 +181.22%	0.3681 +122.74%	0.22558 -39.98%	0.1811 -66.72%	0.36692 +85.76%	0.39413 +31.04%
FT on task	N/A	N/A	0.23689 +138.38%	0.38275 +131.60%	N/A	N/A	0.3652 +84.89%	<u>0.4944</u> +64.38%
FT on task (3)	N/A	N/A	0.24268 +144.20%	0.38162 +130.92%	N/A	N/A	<u>0.4049</u> +104.99%	**0.5037** +67.48%
FT on task (5)	N/A	N/A	0.23623 +137.71%	0.35956 +117.57%	N/A	N/A	**0.4392** +122.36%	0.4809 +59.89%
Dataset: Yambda 5B								
LH-track	0.03093	0.07419	0.09907	0.16744	0.09001	0.14663	0.15886	0.23883
on task	0.03090 -0.11%	0.07463 +0.59%	0.14193 +43.26%	0.22933 +36.96%	0.08847 -1.71%	0.1232 -15.98%	0.16311 +2.68%	0.23461 -1.77%
on task (3)	0.01592 -48.53%	0.04286 -42.22%	<u>0.14784</u> +49.23%	0.23341 +39.40%	0.08701 -3.33%	0.09918 -32.36%	0.17268 +8.70%	0.23514 -1.55%
on task (5)	0.01233 -60.14%	0.03450 -53.50%	**0.14919** +50.59%	0.23258 +38.90%	0.08601 -4.44%	0.08509 -41.97%	0.19034 +19.82%	0.23833 -0.21%
FT on task	N/A	N/A	0.13233 +33.58%	0.24084 +43.84%	N/A	N/A	0.2007 +26.34%	0.2876 +20.42%
FT on task (3)	N/A	N/A	0.13798 +39.28%	**0.24665** +47.31%	N/A	N/A	<u>0.2204</u> +38.74%	<u>0.2973</u> +24.48%
FT on task (5)	N/A	N/A	0.14037 +41.69%	<u>0.24502</u> +46.33%	N/A	N/A	**0.2544** +60.14%	**0.3175** +32.94%

Table 3: Experiment results for Tasks T1 and T2. Rows correspond to the different training approaches: LH-track is a baseline model that was trained on the track listening history and predictions are filtered to track discoveries (T1) or familiar track listens (T2) depending on the task. On task (k) refers to models that are trained on the task directly, where k is the preprocessing parameter for the condensing step. FT on task (k) refers to the LH model that was fine-tuned on task data, where again k refers to the condensing parameter. For all training variants except fine-tuning, we train both models ALS and gSASRec on both tasks and compute NDCG@100 and Recall@100, whereas the relative changes refer to the baseline LH-track models. The highest metric values for each task are marked bold, the second highest is underlined.

gains of up to +21% for sequential models, though the optimal condensation parameter requires task-specific tuning; (3) fine-tuning offers a practical middle ground, achieving +66% to +139% improvements over baseline models (capturing 70-80% of maximum possible gains on discovery tasks) while requiring only incremental training costs. As future research direction, experiments suggest a unimodal dependency between the listening history condensation parameter and model performance, yet this warrants more extensive experimentation.

References

[1] Himan Abdollahpouri and Steve Essinger. 2018. Towards effective exploration/exploitation in sequential music recommendation. *arXiv preprint arXiv:1812.03226* (2018).

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[3] Paul Baltescu, Haoyu Chen, Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. 2022. ItemSage: Learning Product Embeddings for Shopping Recommendations at Pinterest. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. 2703–2711. doi:10.1145/3534678.3539170

[4] Minmin Chen. 2021. Exploration in recommender systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 551–553.

[5] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*. 263–272.

[6] Junjie Huang, Jizheng Chen, Jianghao Lin, Jiarui Qin, Ziming Feng, Weinan Zhang, and Yong Yu. 2025. A Comprehensive Survey on Retrieval Methods in Recommender Systems. (2025). doi:10.1145/3771925

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[8] Anton Klenitskiy, Anna Volodkevich, Anton Pembek, and Alexey Vasilev. 2024. Does It Look Sequential? An Analysis of Datasets for Evaluation of Sequential Recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 1067–1072.

[9] Zihan Liu, Yupeng Hou, and Julian McAuley. 2024. Multi-behavior generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 1575–1585.

[10] Aleksandr Vladimirovich Petrov and Craig Macdonald. 2023. gsasrec: Reducing overconfidence in sequential recommendation trained with negative sampling. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 116–128.

Training	T3: Artist Discovery				T4: Artist Familiar Listening			
	ALS		gSASRec		ALS		gSASRec	
	N@100	R@100	N@100	R@100	N@100	R@100	N@100	R@100
LH-artist	0.07569	0.20125	0.10185	0.15812	0.25998	0.37829	0.34277	0.3863
on task	0.07685 +1.53%	0.20364 +1.19%	0.2154 +111.49%	0.35164 +122.39%	0.26374 +1.45%	0.35773 -5.43%	0.36646 +6.91%	0.39676 +2.71%
on task (3)	0.04707 -37.82%	0.13836 -31.25%	<u>0.24395</u> +139.51%	<u>0.38499</u> +143.48%	0.26213 +0.83%	0.32621 -13.77%	0.40846 +19.16%	0.42283 +9.46%
on task (5)	0.03878 -48.77%	0.11806 -41.33%	**0.2634** +158.62%	**0.40584** +156.67%	0.2609 +0.36%	0.30438 -19.54%	0.44375 +29.46%	0.43569 +12.79%
FT on task	N/A	N/A	0.16979 +66.71%	0.32156 +103.36%	N/A	N/A	0.4326 +26.21%	0.487 +26.07%
FT on task (3)	N/A	N/A	0.18614 +82.76%	0.34586 +118.73%	N/A	N/A	<u>0.4535</u> +32.30%	<u>0.49407</u> +27.90%
FT on task (5)	N/A	N/A	0.20365 +99.95%	0.36984 +133.90%	N/A	N/A	**0.5066** +47.80%	**0.5136** +32.95%

Table 4: Experiment results for Tasks T3 and T4 on the Yambda-5B dataset. Rows correspond to the different training approaches: LH-artist is a baseline model that was trained on the artist listening history and predictions are filtered to artist discoveries (T3) or familiar artist listens (T4) depending on the task. On task (k) refers to models that are trained on the task directly, where k is the preprocessing parameter for the condensing step. FT on task (k) refers to the LH-artist model that was fine-tuned on task data, where again k refers to the condensing parameter. For all training variants except fine-tuning, we train both models ALS and gSASRec on both tasks and compute NDCG@100 and Recall@100. The highest metric values for each task are marked bold, the second highest is underlined.

Input Order	T1		T2		T3		T4	
	N@100	R@100	N@100	R@100	N@100	R@100	N@100	R@100
Dataset: Deezer								
in order	0.29277	0.40121	0.30229	0.38972	-	-	-	-
shuffled	0.12424 -57.56%	0.19262 -51.99%	0.17416 -42.39%	0.25347 -34.96%	-	-	-	-
Dataset: Yambda 5B								
in order	0.14193	0.22933	0.16311	0.23461	0.2154	0.35164	0.36646	0.39676
shuffled	0.06607 -33.31%	0.11553 -31.00%	0.11958 -26.69%	0.18231 -22.29%	0.11545 -46.40%	0.21043 -40.16%	0.29485 -19.54%	0.3564 -10.17%

Table 5: Performance of the gSASRec models that are trained "on task" in two scenarios at test time: *in order*, where the user listening history is passed to the model in the correct order, and *shuffled*, where the user listening history is uniformly randomly permuted.

[11] A. Ploshkin, V. Tytskiy, A. Pismenny, V. Baikalov, E. Taychinov, A. Permiakov, D. Burlakov, E. Krofto, and N. Savushkin. 2025. Yambda-5B – A Large-Scale Multi-modal Dataset for Ranking And Retrieval. arXiv:2505.22238 [cs.IR] https://arxiv.org/abs/2505.22238

[12] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Keshavan, Trung Vu, Lukasz Heidt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender systems with generative retrieval. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23)*. Article 452.

[13] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2022. *Recommender Systems Handbook*. Springer, New York, NY.

[14] Bruno Sguerra, Viet-Anh Tran, and Romain Hennequin. 2022. Discovery dynamics: Leveraging repeated exposure for user and music characterization. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 556–561.

[15] Bruno Sguerra, Viet-Anh Tran, and Romain Hennequin. 2023. Ex2vec: Characterizing users and items from the mere exposure effect. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 971–977.

[16] Yi Su, Xiangyu Wang, Elaine Ya Le, Liang Liu, Yuening Li, Haokai Lu, Benjamin Lipshitz, Sriraj Badam, Lukasz Heldt, Shuchao Bi, et al. 2024. Long-term value of exploration: measurements, findings and algorithms. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 636–644.

[17] 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*. 1441–1450.

[18] Viet-Anh Tran, Guillaume Salha-Galvan, Bruno Sguerra, and Romain Hennequin. 2024. Transformers Meet ACT-R: Repeat-Aware and Sequential Listening Session Recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 486–496.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Adv. Neural Inf. Process. Syst.* 30 (2017).

[20] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, and Ed Chi (Eds.). 2019. *Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations*.

[21] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. Association for Computing Machinery, 43–51. doi:10.1145/3298689.3346997