An Embarrassingly Simple Graph Heuristic Reveals Shortcut-Solvable Benchmarks for Sequential Recommendation

Haoyu Han[1], Li Ma[1], Hanbing Wang[1], Bingheng Li[1], Daochen Zha[2], Chun How Tan[2], Huiji Gao[2], Xin Liu[2], Stephanie Moyerman[2], Sanjeev Katariya[2], Hui Liu[1], Jiliang Tang[1]
[1]Michigan State University, [2]Airbnb, Inc.
{hanhaoy1, mali13, wangh137, libinghe, liuhui7, tangjili}@msu.edu
{daochen.zha, chunhow.tan, huiji.gao, xin.liu}@airbnb.com
{stephanie.moyerman, sanjeev.katariya}@airbnb.com

Abstract

Sequential recommendation is a central task in recommender systems, and recent research has increasingly shifted toward generative recommenders that leverage both sequential patterns and semantic item information. However, these methods are often evaluated on a small set of widely used benchmarks. This raises a natural question: *do these benchmarks actually require the advanced modeling capabilities of modern generative recommenders?* We conduct a benchmark audit using an intentionally simple graph heuristic: starting from only the last one or two interacted items, it retrieves candidates from a few-hop item-transition graph and ranks them with item-feature similarity. Surprisingly, despite its simplicity, this heuristic matches or outperforms a broad set of modern baselines on a variety of popular sequential recommendation benchmarks. For example, it achieves relative NDCG@10 improvements of 38.10% and 44.18% over the best competing baseline on the widely used Amazon Review Sports and CDs datasets, respectively.

We further show that this phenomenon is not merely an artifact of a particular heuristic, but reflects shortcut solvability in existing benchmarks. Specifically, we identify three shortcut structures that could make next-item prediction easier than expected: low-branching local transition structure, feature-smooth transitions, and limited dependence on long user histories. These shortcuts need not appear simultaneously. Depending on the dataset, even one or two strong shortcut signals can make simple local retrieval highly competitive, while weakening the relevant signals allows more sophisticated models to show clearer benefits. Our broader evaluation across 14 diverse datasets further shows that model rankings change substantially with dataset properties, while the simple graph heuristic remains competitive on 10 out of 14 datasets. These findings suggest that strong performance on several standard sequential recommendation benchmarks may not faithfully reflect whether recent methods achieve the advanced modeling capabilities they aim to demonstrate. Rather than treating datasets as interchangeable leaderboards, we argue for more careful dataset selection and dataset-level diagnostic analysis when using benchmarks to support claims about the benefits of new recommendation models.

1 Introduction

Sequential recommendation has long been a fundamental problem in recommender systems, where the goal is to infer a user's next interaction from past behaviors [1, 2, 3]. Early studies mainly relied on collaborative filtering [4, 5] and Markov-chain models [6, 7], while later neural methods introduced recurrent networks [8, 9], graph-based recommenders [10, 11], and attention-based sequence encoders [12, 13]. Recently, however, the field has seen a rapid rise of generative recommendation

Preprint.

methods [14, 15, 16]. These generative methods reformulate recommendation as generation or retrieval over discrete item identifiers, semantic IDs, or textual item representations [17, 18]. As a result, item-side information has become increasingly central: many recent models rely on item text, metadata, semantic embeddings, or LLM-generated representations to construct item codes, prompts, or prediction targets [19, 16, 20].

Along with this methodological shift, evaluation practice has also become strikingly concentrated. We analyze dataset usage across 94 recent generative-recommendation papers, detailed in Appendix A. As shown in Figure 1, Amazon Review datasets [21] dominate recent evaluations, while other datasets are used much less frequently. This dominance is understandable: Amazon Review datasets provide rich item-side metadata, such as titles, categories, brands, prices, reviews, and other textual fields, which can be naturally processed by text encoders or LLM-based generators [22, 23] for semantic and generative recommendation.

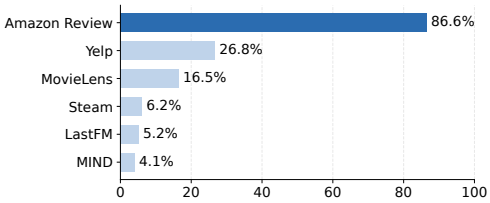


Figure 1: The proportion of surveyed sequential recommendation papers utilizing each dataset.

However, this concentration creates a critical question: *what do these dominant benchmarks actually measure?* High performance on these datasets is often taken as evidence that a model better captures user preferences, sequential dynamics, semantic structure, long-range dependencies, or generative reasoning. But this interpretation relies on an implicit assumption: the benchmark truly requires these advanced capabilities. If the same benchmark can be solved by a much simpler signal, then the comparison may be less informative than it appears. Strong performance may reflect dataset-specific shortcuts rather than genuinely advanced sequential modeling.

In this work, we uncover such shortcuts. We revisit widely used sequential recommendation benchmarks through an intentionally simple graph heuristic. Given only the training sequences, we construct an item-transition graph from observed item-to-item transitions. At test time, the heuristic looks only at the user's last one or two interacted items, retrieves candidates from their few-hop neighborhoods in the transition graph, and ranks them by item-feature similarity with a small direct-edge bonus. It has no deep sequence encoder, no generative objective, no learned user representations, and no training, making it simple and computationally efficient.

Despite its simplicity, this heuristic outperforms a broad set of strong baselines on the Amazon Review benchmarks, including traditional sequential models, graph-based recommenders, transformer-based recommenders, and recent generative recommenders. This result suggests that **strong performance on these datasets may be achievable through much simpler signals than expected**. Therefore, relying only on these datasets may not faithfully reflect the advanced modeling capabilities that recent methods aim to demonstrate. To understand why this happens, we identify three shortcut structures that potentially help explain this phenomenon: low-branching local transition structure, where recent items have small but useful transition neighborhoods; feature-smooth transitions, where transition-connected items are close in feature space; and limited dependence on long user histories, where the last one or two interactions already provide enough signal for next-item prediction.

We then ask whether this phenomenon is unique to Amazon Review benchmarks or reflects a broader issue in sequential recommendation evaluation. To answer this question, we evaluate the same baselines across 14 datasets with diverse transition structures, feature signals, and history dependencies. The simple heuristic remains competitive on 10 of them, suggesting that **shortcut-solvable structures are widespread in existing benchmarks**. At the same time, the heuristic is not universally superior: when the relevant shortcut signals are weakened, more sophisticated models can show clearer benefits. These results suggest two cautions for the community. For model designers, we encourage benchmark choices to be aligned with the capabilities being claimed. Methods that claim improved semantic understanding, generative modeling, or long-context reasoning should ideally be evaluated on datasets where simple local-transition, feature-similarity, or short-history shortcuts are insufficient. For dataset and benchmark creators, dataset releases would benefit from accompanying diagnostic analyses of transition structure, feature smoothness, history dependence, and other relevant properties. Such diagnostics can help clarify what a dataset actually measures and reduce the risk that future evaluations mistake shortcut-solvable performance for genuine modeling progress.

2 Related Work

Sequential recommendation. Sequential recommendation aims to predict the next item a user will interact with given a historical interaction sequence. Early approaches often combine collaborative filtering with sequential transition modeling [6]. For example, matrix factorization [24] learns user and item representations from historical interactions, while Markov-chain-based methods model short-term item-to-item transitions [25]. Later neural methods introduce more expressive sequence encoders to capture user dynamics. GRU4Rec [8] uses recurrent neural networks to model session-level behavior, Caser [26] applies convolutional filters over recent interactions, SASRec [27] uses self-attention to capture item dependencies in user histories, and BERT4Rec [28] adopts bidirectional Transformer-style representation learning with masked item prediction. Beyond sequence encoders, graph-based recommendation methods exploit the structure of user-item or item-item interactions. LightGCN [10] performs simplified graph propagation on the user-item interaction graph and serves as a strong collaborative filtering baseline. For sequential and session-based recommendation, SR-GNN [11] represents each session as an item-transition graph, while GCE-GNN [29] further combines local session graphs with global item-transition information. These methods show that collaborative structure and item-transition patterns are highly informative for next-item prediction.

More recently, sequential recommendation has moved toward increasingly sophisticated semantic, generative, and LLM-based models [30, 31, 32]. This trend is partly driven by datasets with rich item-side information [21, 33], where item titles, descriptions, categories, and reviews can be used to construct semantic representations or natural-language prompts. Generative recommenders [34, 35] reformulate recommendation as item generation or generative retrieval, often by assigning items discrete semantic identifiers or token sequences. HSTU [36] studies large-scale sequential transduction for recommendation, TIGER [14] formulates recommendation as generative retrieval over semantic item IDs, LETTER [15] integrates semantic information and collaborative signals to learn item tokenizations, and CoFiRec [19] learns a coarse-to-fine tokenizer for generative recommendation. LLM-based recommenders [17, 37] further leverage language models to encode item content, user histories, or recommendation prompts. Although these models are motivated by advanced modeling capabilities such as semantic understanding, generative retrieval, and long-context sequence modeling, many of them are evaluated on a relatively small set of widely used sequential recommendation benchmarks. In contrast to proposing another sophisticated recommender, our work asks whether these benchmarks actually require such advanced modeling capacity.

Evaluation and benchmark auditing. Offline evaluation is central to recommender system research, but prior work [38, 39, 40] has shown that empirical conclusions can be highly sensitive to preprocessing choices, data splitting, candidate sampling, hyperparameter tuning, and baseline selection. Several studies [41, 42, 43] further question whether complex neural recommenders always provide genuine progress over strong and well-tuned simpler baselines. Our work follows this benchmark-auditing perspective, but studies a different failure mode: *shortcut solvability* in sequential recommendation benchmarks. Prior critiques mainly examine whether evaluation protocols are fair and whether proposed models are compared against sufficiently strong baselines. In contrast, we ask what signals the benchmarks themselves reward. We show that several widely used sequential recommendation datasets can be largely solved by an embarrassingly simple graph heuristic.

3 Problem Setup and Diagnostic Heuristic

In this section, we introduce the sequential recommendation and our graph-based diagnostic heuristic.

3.1 Problem Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets, respectively. Each user $u \in \mathcal{U}$ has a chronological interaction sequence

$$S_u = (i_1^u, i_2^u, \ldots, i_{T_u}^u),$$

where $i_t^u \in \mathcal{I}$ is the item interacted with at time t. Given a prefix sequence

$$S_{1:t}^u = (i_1^u, \ldots, i_t^u),$$

the goal of sequential recommendation is to rank candidate items so that the next item i_{t+1}^u appears near the top.

3.2 A Simple Graph-Heuristic Probe

We use an intentionally simple graph heuristic as a diagnostic probe. The heuristic is motivated by two basic sources of recommendation signals: collaborative structure from user-item interaction sequences, and semantic similarity from item-side features. Rather than proposing a new recommendation architecture, we use this heuristic to assess how much predictive power these simple signals provide on widely used benchmarks.

Given the training sequences, we construct a directed item transition graph $G = (\mathcal{I}, \mathcal{E})$, where each node is an item $i \in \mathcal{I}$. A directed edge $(i, j) \in \mathcal{E}$ is added if item j appears immediately after item i in a training sequence. Let $N_{i,j}$ denote the number of observed transitions from item i to j. We assign each edge a normalized weight

$$w_{i,j} = \frac{\log(1 + N_{i,j})}{\max_{j':N_{i,j'}>0} \log(1 + N_{i,j'})},$$

so that outgoing normalized edge weights from each item are scaled to $[0, 1]$.

At inference time, given a prefix sequence $S^u_{1:t}$, the heuristic uses either the last item i^u_t or the last two items (i^u_{t-1}, i^u_t) as anchors. For each anchor item s, it retrieves candidates from a few-hop neighborhood of s in the transition graph. Within each hop, candidates are ranked mainly by feature similarity to the anchor, with a small edge-weight bonus for direct 1-hop neighbors. This bonus reflects the intuition that larger edge weights correspond to more frequent transitions and thus higher transition likelihood.

Specifically, each item i has an L2-normalized feature embedding $\hat{\mathbf{e}}_i$. For a candidate item c retrieved from the ℓ-hop neighborhood of anchor s, we compute

$$\text{score}_s(c) = \hat{\mathbf{e}}_s^\top \hat{\mathbf{e}}_c + \alpha \cdot \mathbf{1}[\ell = 1] \cdot w_{s,c}, \tag{1}$$

where α is a small edge-weight bonus weight. The first term is the cosine similarity between the anchor and the candidate. The second term is a direct-transition bonus based on the normalized edge weight, and is applied only to 1-hop candidates. For candidates from hops larger than one, the score reduces to feature similarity.

For each anchor and each hop, we keep only a small number of top-scoring candidates. If the same item is retrieved from multiple anchors or hops, we keep its highest score. The final recommendation list is obtained by sorting all retained candidates by their scores.

We refer to this heuristic as TGH, short for Transition-Graph Heuristic. It is deliberately simple: it uses only recent anchor items, local transition-graph neighborhoods, and item-feature similarity, without a sequence encoder, generative objective, representation learning, or training. As a result, it is substantially more efficient than existing learnable sequential recommenders, which typically require extensive training.

3.3 Experimental Setting

Evaluation protocol. We follow the standard next-item recommendation protocol [27, 14]. For each user, interactions are ordered chronologically, and models are evaluated by ranking the held-out next item given a prefix sequence. We report top-K ranking metrics, including Recall@K and NDCG@K, where $K \in [1, 5, 10]$.

Text representation. To ensure a fair comparison among methods that use item-side textual information, we use a shared text encoder for all of them. Specifically, we encode item text with `google/flan-t5-xl` [22], following [44]. The input text is constructed from the available metadata in each dataset, such as item title, category, description, or other textual fields.

Graph-heuristic variants. We evaluate two short-context variants of TGH. Although the retrieval budgets and hyperparameters could be tuned on the validation set, we fix them across all datasets to keep the heuristic deliberately simple and ensure that its performance is not driven by dataset-specific configuration choices.

In the **TGH-1** setting, TGH uses only the last interacted item i^u_t as the anchor. It considers the anchor's 1-, 2-, and 3-hop neighborhoods in the transition graph, and selects the top $(7, 2, 1)$ candidates from these hops according to the scoring function in Equation 1, respectively.

Table 1: Main results on Amazon Review benchmarks. We report Recall@10 and NDCG@10. All results are percentages. Best results are in **bold**, second-best results are underlined, and the best baseline is denoted by wavy underline.

Method	Beauty		Sports		Toys		CDs	
	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
LightGCN	7.21	4.36	3.69	2.10	8.63	5.32	2.63	1.52
SR-GNN	5.43	3.13	2.61	1.33	3.77	2.32	0.63	0.35
SASRec	6.21	3.31	3.32	1.83	7.41	4.23	4.44	2.33
HSTU	5.73	3.00	2.48	1.25	6.40	3.56	5.62	2.91
TIGER	6.41	3.60	3.70	1.96	6.01	3.27	1.44	0.75
LETTER	4.96	2.62	2.18	1.11	3.07	1.60	4.50	2.44
CoFiRec	6.24	3.36	3.82	2.02	5.63	2.89	5.64	3.01
TGH-1	7.66	5.01	4.27	2.76	9.13	6.12	6.21	4.13
TGH-2	**7.85**	**5.07**	**4.66**	**2.90**	**9.44**	**6.25**	**6.89**	**4.34**
Rel. Improv.	+8.88%	+16.28%	+21.99%	+38.10%	+9.39%	+17.48%	+22.16%	+44.18%

In the **TGH-2** setting, TGH uses both the last item i_t^u and the second-last item i_{t-1}^u as anchors. For the last item, it selects the top $(5, 1)$ candidates from its 1- and 2-hop neighborhoods. For the second-to-last item, it selects the top $(3, 1)$ candidates from its 1- and 2-hop neighborhoods.

We also set the edge-bonus weight to $\alpha = 0.5$ for all datasets.[1]

Compared methods. We compare the heuristic with representative methods from multiple recommendation families, including graph-based recommendation methods such as LightGCN [10] and SR-GNN [11], conventional sequential recommenders such as SASRec [27], and recent generative recommendation methods such as HSTU [36], TIGER [14], LETTER [15], and CoFiRec [19].

4 Experiments

4.1 The Graph Heuristic is Surprisingly Strong on Amazon Review Benchmarks

We first evaluate on the most widely used Amazon Review benchmarks [21], including Beauty, Sports, Toys and CDs. Table 1 reports Recall@10 (R@10) and NDCG@10 (N@10); full results with additional metrics are provided in Appendix E. Despite its simplicity, TGH is highly competitive across all four Amazon datasets. In particular, it achieves relative NDCG@10 improvements of 38.10% and 44.18% over the best competing baselines on Sports and CDs, respectively.

These results are surprising because TGH retrieves only a small number of candidates from local transition-graph neighborhoods and ranks them with item-feature similarity, yet still achieves strong performance. This suggests that the Amazon Review benchmarks contain simple predictive signals that can be exploited without advanced sequential or generative modeling. Considering that Amazon Review datasets are used in 86.6% of the surveyed generative recommendation papers, as shown in Figure 1, this finding raises concerns about whether relying heavily on performance on these benchmarks provides sufficient evidence for the claimed benefits of modern recommendation models.

4.2 What Signals Make the Heuristic Work?

Although TGH achieves surprisingly strong performance on Amazon Review benchmarks, it remains unclear why such a simple heuristic can outperform much more sophisticated models. To understand the underlying signals, we analyze dataset statistics and introduce a set of diagnostic baselines.

Dataset statistics. We consider both general dataset statistics and item-transition graph statistics. The general statistics include the number of users, the number of items, and the average sequence length (Avg. Seq. Len.). For the item-transition graph constructed from the training sequences, we report the average out-degree (Avg. Out-Deg.), the average edge weight (Avg. Edge W.), and the coverage of the ground-truth item within the 1-, 2-, and 3-hop neighborhoods of the last interacted item (Cov@1, Cov@2, and Cov@3).

[1]The code is available at https://github.com/haoyuhan1/GraphRec/.

Table 2: Statistics of the item-transition graphs on Amazon Review benchmarks. Coverage@k denotes the percentage of test targets reachable within k hops from the last interacted item.

Dataset	#Users	#Items	#Edges	Avg. Seq. Len.	Avg. Out-Deg.	Avg. Edge W.	Cov@1	Cov@2	Cov@3
Beauty	22,363	12,101	114,582	8.15	9.47	1.15	8.61%	24.85%	56.64%
Sports	35,598	18,357	180,610	7.96	9.84	1.05	5.13%	20.26%	58.16%
Toys	19,412	11,924	102,268	7.97	8.58	1.07	8.06%	20.16%	47.16%
CDs	75,258	64,443	810,347	14.58	12.57	1.08	9.07%	28.06%	61.44%

Table 3: Results of diagnostic baselines on Amazon review benchmarks.

Method	Beauty		Sports		Toys		CDs	
	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
ID-LAST	6.16	3.74	2.70	1.63	6.81	4.22	3.86	2.31
SEM-NN	5.32	3.30	2.68	1.55	8.04	5.01	1.20	0.75
ID+SEM	6.66	4.08	3.18	1.91	8.53	5.37	4.00	2.41
SASRec	6.21	3.31	3.23	1.83	7.41	4.23	4.44	2.33
-Last-1	5.98	3.33	3.05	1.66	6.59	3.89	3.77	1.94
HSTU	5.73	3.00	2.48	1.25	6.40	3.56	5.62	2.91
-Last-1	5.39	2.92	2.61	1.37	6.20	3.42	4.38	2.26
TGH-1	**7.66**	**5.01**	**4.27**	**2.76**	**9.13**	**6.12**	**6.21**	**4.13**

Diagnostic baselines. We introduce three controlled baselines to isolate simple signals in the data. First, ID-LAST learns a trainable embedding for each item ID and uses only the last interacted item to predict the next item. This baseline measures how much signal can be captured from the ID-based transitions. Second, SEM-NN is a training-free semantic nearest-neighbor baseline. Given the last interacted item, SEM-NN ranks all candidate items by cosine similarity between their text embeddings and the text embedding of the last item. This baseline tests whether item-feature similarity alone is predictive for next-item retrieval. Finally, ID+SEM combines the scores of ID-LAST and SEM-NN through late fusion, where the final score is the sum of the ID-based score and the semantic-similarity score. This baseline tests whether ID-based transition signals and item-feature similarity provide complementary information. We further conduct a history-window ablation on SASRec and HSTU. Instead of using the full interaction sequence, we restrict the maximum history window to 1, so that each prediction can only condition on the immediately previous item. We refer to these variants as LAST-1.

The item-transition graph statistics are shown in Table 2, and the performance of the diagnostic baselines is shown in Table 3. These results lead to the following key findings.

Finding 1: many targets are reachable through low-branching local transitions. As shown in Table 2, the Amazon Review transition graphs have relatively small average out-degrees compared with the total number of items. For example, the average out-degree is only 9.47 for Beauty and 12.57 for CDs, while the datasets contain thousands to tens of thousands of items. Meanwhile, the direct 1-hop neighbors already cover a non-trivial fraction of test targets, with Cov@1 of 8.61% and 9.07% on Beauty and CDs, respectively. This indicates that recent items provide a small but informative local candidate space. After expanding to a few hops, the target coverage increases substantially while the search remains restricted to local transition neighborhoods. We refer to this shortcut structure as *low-branching local transition structure*: the transition graph narrows next-item prediction from ranking over the full item universe to selecting among a small set of locally plausible candidates.

Finding 2: item features provide a useful ranking signal. As shown in Table 3, SEM-NN retrieves from the full item set using only feature similarity, yet already achieves strong performance on Beauty, Sports, and Toys datasets. This suggests that items adjacent in user sequences often have similar item features on these datasets, making feature similarity a useful signal for next-item prediction. We refer to this shortcut structure as *feature-smooth transitions*. However, this property does not hold equally across all Amazon Review datasets. On CDs, SEM-NN performs much worse, indicating that global feature similarity alone may be insufficient.

Finding 3: many Amazon Review benchmarks require little long-context information. The strong performance of TGH-1 suggests that the most recent interaction already contains much of the

useful signal. This observation is also supported by ID-LAST, which uses only the last item ID to predict the next item and still achieves strong performance on several Amazon Review datasets. To further examine whether long-range history is necessary, we conduct a history-window ablation on SASRec and HSTU. Specifically, we compare the full-history models with their LAST-1 variants, where each prediction only conditions on the immediately previous item. As shown in Table 3, using only a history window of size 1 achieves performance close to the full-history setting on Beauty, Sports, and Toys. This suggests that long-range sequential information may not be essential for strong performance on these benchmarks. We refer to this shortcut structure as *limited dependence on long user histories*: the benchmark can be largely solved using very recent interactions, without requiring models to capture long-range user preference or complex sequential dynamics.

Overall, these analyses suggest that the three shortcut structures could serve as useful diagnostic axes for interpreting the strong performance of TGH on Amazon Review datasets. While they are not meant to be an exhaustive explanation of benchmark behavior, they help characterize several simple signals that can make a dataset shortcut-solvable. The heuristic can benefit from low-branching local transition structure, feature-smooth transitions, and limited dependence on long user histories, but these shortcuts need not appear simultaneously. For example, CDs shows a weak global feature-similarity signal, as SEM-NN performs poorly when retrieving from the full item set. Yet TGH remains effective, suggesting that the transition graph first restricts retrieval to a local candidate set, where feature similarity becomes more useful for ranking. This example illustrates that shortcut-solvability may arise from the interaction of multiple simple signals, rather than requiring all identified shortcut structures to be present at once.

4.3 Beyond Amazon Review: Shortcut Solvability Across Diverse Benchmarks

The previous subsection shows that TGH performs surprisingly well on Amazon Review benchmarks and identifies three potential dataset shortcuts that could help explain this behavior. We next ask whether this phenomenon is specific to Amazon Review datasets, or whether similar shortcut-solvable patterns also appear in other sequential recommendation benchmarks with item-side information. To this end, we evaluate on a broader collection of datasets, including Delicious [45], LastFM [45], MovieLens-1M (ML-1M) [46], Yelp [47], MIND [48], Goodreads-Comics (GR-Comics) [49], Goodreads-Children (GR-Children) [49], STEAM [27], H&M [50], and Amazon-M2-UK (Amazon-UK) [33]. The dataset statistics are shown in Table 4. Compared with the Amazon Review datasets, these benchmarks cover more diverse data regimes, with substantially different sequence lengths, transition-graph densities, and average out-degrees. For example, some datasets have much longer user histories, while others have much larger local transition neighborhoods. This diversity allows us to examine whether the proposed shortcuts help interpret model behavior beyond the Amazon Review setting. We use the same baseline suite as in the Amazon Review experiments, except for CoFiRec, which is specifically designed for Amazon Review datasets.

Table 5 reports NDCG@10 on the broader set of datasets; additional metrics are provided in Appendix F. Overall, TGH remains competitive beyond Amazon Review benchmarks, achieving the best or second-best performance on 6 out of 10 datasets and comparable performance on Amazon-M2-UK. However, TGH is not universally superior. On MovieLens-1M, Yelp, and MIND, it underperforms traditional and generative sequential recommenders. We next use the shortcut structures identified from the Amazon Review analysis to interpret both the success and failure cases of TGH across these broader datasets.

Shortcut 1: low-branching local transition structure. As shown in Table 4, Delicious, LastFM, and Amazon-M2-UK have average out-degrees below 10, indicating low-branching local transition graphs. On these datasets, the local neighborhoods of recent items provide compact candidate spaces, and TGH achieves strong performance. In contrast, MovieLens-1M has a much larger average out-degree of 78.71. Although many targets may still be reachable within a few hops, the local candidate space is much larger, making fixed-budget local retrieval less effective. This helps explain why TGH underperforms stronger sequential recommenders on MovieLens-1M.

Shortcut 2: feature-smooth transitions. Some datasets remain favorable to TGH even when their transition graphs are not low-branching. For example, Steam and H&M have large average out-degrees of 186.54 and 116.81, respectively, yet TGH still performs well. We attribute this to feature-smooth transitions: adjacent items in user sequences tend to have similar item features. This is

Table 4: Statistics of the item-transition graphs on the ten additional benchmarks. Coverage@k denotes the percentage of test targets reachable within k hops from the last interacted item.

Dataset	#Users	#Items	#Edges	Avg. Seq. Len.	Avg. Out-Deg.	Avg. Edge W.	Cov@1	Cov@2	Cov@3
Delicious	718	1,200	4,016	9.13	3.35	1.1	9.33	11.98	18.38
LastFM	1,090	3,646	30,372	34.02	8.33	1.11	5.69	21.93	53.58
MovieLens-1M	6,040	3,416	268,867	74.06	78.71	1.6	48.11	96.59	99.93
Yelp	30,431	20,033	219,632	10.4	10.96	1.02	5.75	27.86	73.57
MIND	48,577	39,757	824,397	28.16	20.74	1.48	40.13	89.19	96.73
Goodreads-Comics	89,186	48,623	1,282,693	33.78	26.38	2.14	49.67	84.36	97.25
Goodreads-Children	163,143	55,221	1,622,817	24.26	29.39	2.14	57.04	88.57	98.13
STEAM	334,728	13,047	1,524,022	12.59	116.81	2.11	59.44	97.97	99.8
H&M	1,077,045	104,468	19,487,762	26.01	186.54	1.27	34.32	95.28	99.72
Amazon-M2-UK	1,182,181	494,409	1,500,196	5.12	3.03	1.67	30.35	43.69	52.94

Table 5: NDCG@10 (%) across datasets. **Bold**: best per dataset; underlined: second-best.

Method	**Delicious**	**LastFM**	**ML-1M**	**Yelp**	**MIND**	**GR-Comics**	**GR-Children**	**STEAM**	**H&M**	**Amazon-UK**
ID-LAST	6.64	2.11	6.86	0.94	5.72	18.70	9.39	13.40	0.79	25.38
SEM-NN	2.62	2.66	2.99	0.03	0.14	8.30	1.91	13.40	3.33	21.76
ID+SEM	6.68	2.60	7.09	0.65	5.46	18.43	7.00	13.59	3.77	26.16
LightGCN	4.37	2.22	4.32	0.40	2.78	19.62	8.03	1.98	2.93	**28.78**
SR-GNN	4.78	1.56	8.48	1.48	**13.87**	9.26	9.37	14.93	3.54	9.12
SASRec	**7.57**	**3.32**	**14.21**	1.33	12.67	16.08	12.10	4.49	3.68	14.10
HSTU	4.35	2.01	12.15	1.08	7.49	16.57	11.50	4.32	3.45	19.10
TIGER	5.10	1.45	9.98	1.05	2.19	16.02	10.48	14.24	4.10	6.37
LETTER	2.08	1.79	12.08	**1.76**	2.70	20.90	**14.39**	**15.70**	6.64	8.10
TGH-1	7.06	3.07	9.25	0.90	5.51	23.12	12.23	14.85	8.06	24.87
TGH-2	7.54	2.87	9.39	0.98	5.32	**23.66**	12.46	14.95	**8.70**	25.13

supported by the strong performance of SEM-NN, which retrieves items using only semantic similarity. In contrast, Yelp has a much smaller average out-degree of 10.96, but TGH still underperforms the baselines. This suggests that low branching alone is not sufficient. On Yelp, adjacent items are less feature-smooth, as indicated by the weak performance of SEM-NN. Therefore, even though the local candidate space is relatively small, feature similarity cannot reliably rank the correct next item.

Shortcut 3: limited dependence on long user histories. To test whether each dataset requires long-range user-history information, we compare SASRec and HSTU under two settings: the full-sequence setting and the LAST-1 setting, where each prediction only uses the immediately previous item. Figure 2 reports the relative performance gap between these two settings. On MovieLens-1M and MIND, the full-sequence models achieve much stronger performance than their LAST-1 variants, suggesting that these datasets rely more heavily on long-range user histories. Since TGH only uses the last one or two items as anchors, it cannot capture this long-context signal, which helps explain its weaker performance on these datasets.

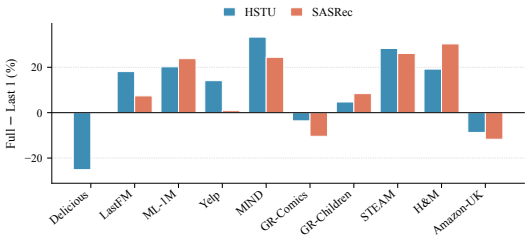
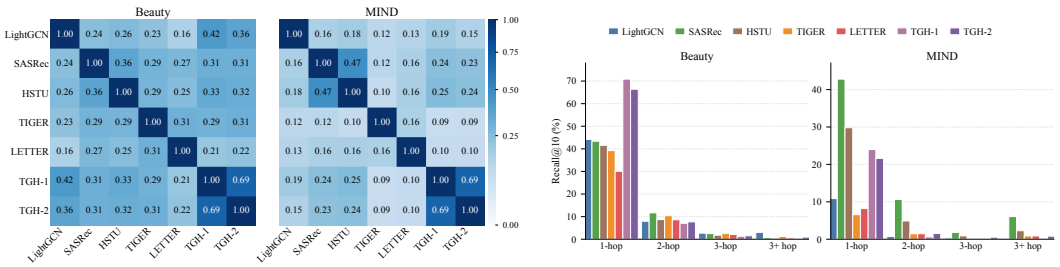


Figure 2: The relative performance gap between the Full sequence and Last-1 settings.

Overall, across the four Amazon Review benchmarks and ten additional datasets, TGH achieves competitive performance, ranking best or second-best on 10 out of 14 datasets. Its successes and failures can be potentially interpreted through the three shortcut structures identified above. Importantly, this does not imply that modern sequential or generative recommenders are ineffective. Rather, it shows that some benchmarks allow simple shortcuts to dominate evaluation, while other datasets expose prediction behaviors that TGH cannot capture. We therefore next examine prediction-level differences between TGH and learned recommenders to better understand what advanced models may capture beyond the heuristic.



(a) Prediction overlap analysis.

(b) Recall@10 by ground-truth hop distance.

Figure 3: Prediction-level comparison between TGH and learned recommenders.

4.4 Prediction-Level Differences Between the Heuristic and Learned Models

The previous results show that TGH can be highly competitive, but this does not imply that learned sequential or generative recommenders are ineffective. To better understand the difference between TGH and learned models, we further analyze their prediction patterns.

We first measure the Jaccard similarity between the correct-prediction sets of different methods. Figure 3(a) shows the overlap analysis on Beauty and MIND. The overlap between TGH and learned sequential recommenders is relatively low, indicating that they often succeed on different test instances. This suggests that, although TGH can achieve competitive overall performance by exploiting benchmark shortcuts, learned models capture additional signals that are not fully covered by the heuristic.

We further evaluate Recall@10 grouped by the hop distance between the last interacted item and the ground-truth item in the transition graph. As shown in Figure 3(b), TGH performs best when the ground-truth item is within a small number of hops, which is expected given its local retrieval mechanism. In contrast, learned sequential recommenders achieve stronger performance on harder cases, especially when the ground-truth item is beyond 3 hops. This indicates that learned models can capture prediction patterns beyond simple local retrieval, such as longer-range sequential dependencies or more complex user-behavior signals.

Overall, this analysis shows that the issue is not that current baselines are weak. Rather, many widely used benchmarks contain shortcuts that can be solved by simple heuristics, so aggregating the overall performance on these datasets alone may be unreliable evidence for the advantages of new methods. For model designers, this calls for evaluation on datasets that truly test the claimed capability, such as settings where local-transition retrieval, feature similarity, or short-history prediction is insufficient. For benchmark designers, this calls for reporting dataset-level diagnostics, such as transition branching, feature-smoothness, and history dependence, to make clear what signals the benchmark rewards.

5 Conclusion

In this work, we revisit the evaluation practice of sequential recommendation, especially in the context of recent generative recommenders. We show that an intentionally simple transition-graph heuristic can achieve competitive performance on many widely used benchmarks, including several Amazon Review datasets that dominate recent evaluations. This result suggests that strong benchmark performance may sometimes be driven by simple shortcut signals rather than advanced sequential or generative modeling ability.

Through dataset statistics, diagnostic baselines, and cross-dataset evaluation, we identify three shortcut structures that help explain this behavior: low-branching local transition structure, feature-smooth transitions, and limited dependence on long user histories. Across 14 datasets, the heuristic is competitive on 10, but it is not universally superior; when these shortcut signals are weakened, learned sequential and generative models can show clearer benefits.

Our findings highlight the need for more capability-aware evaluation. We encourage model designers to select datasets that directly test the capabilities their methods aim to improve, and benchmark designers to report dataset-level diagnostics that clarify what signals a benchmark rewards. Rather than treating benchmarks as interchangeable leaderboards, future work can use them as tools for understanding when and why different recommendation models succeed.

References

[1] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, 125:102427, 2024.

[2] Li-Wei Pan, Wei-Ke Pan, Mei-Yan Wei, Hong-Zhi Yin, and Zhong Ming. A survey on sequential recommendation. *Frontiers of Computer Science*, 20(3):2003606, 2026.

[3] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–42, 2020.

[4] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

[5] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

[7] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 161–169, 2017.

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[9] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 17–22, 2016.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[11] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 346–353, 2019.

[12] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068, 2018.

[13] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5941–5948, 2019.

[14] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[15] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2400–2409, 2024.

[16] Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian McAuley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. *arXiv preprint arXiv:2502.13581*, 2025.

[17] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pages 299–315, 2022.

[18] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item ids for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 195–204, 2023.

[19] Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. Cofirec: Coarse-to-fine tokenization for generative recommendation. *arXiv preprint arXiv:2511.22707*, 2025.

[20] Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. Genrec: Large language model for generative recommendation. In *European Conference on Information Retrieval*, pages 494–502. Springer, 2024.

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[22] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[25] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[26] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

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

[28] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[29] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 169–178, 2020.

[30] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.

[31] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 585–593, 2022.

[32] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1258–1267, 2023.

[33] Wei Jin, Haitao Mao, Zheng Li, Haoming Jiang, Chen Luo, Hongzhi Wen, Haoyu Han, Hanqing Lu, Zhengyang Wang, Ruirui Li, et al. Amazon-m2: A multilingual multi-locale shopping session dataset for recommendation and text generation. *Advances in Neural Information Processing Systems*, 36:8006–8026, 2023.

[34] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. A review of modern recommender systems using generative models (gen-recsys). In *Proceedings of the 30th ACM SIGKDD conference on Knowledge Discovery and Data Mining*, pages 6448–6458, 2024.

[35] Matthew O Ayemowa, Roliana Ibrahim, and Muhammad Murad Khan. Analysis of recommender system using generative artificial intelligence: A systematic literature review. *Ieee Access*, 12:87742–87766, 2024.

[36] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

[37] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large language models for generative recommendation: A survey and visionary discussions. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pages 10146–10159, 2024.

[38] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1748–1757, 2020.

[39] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models. In *Proceedings of the 14th acm conference on recommender systems*, pages 681–686, 2020.

[40] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. A revisiting study of appropriate offline evaluation for top-n recommendation algorithms. *ACM Transactions on Information Systems*, 41(2):1–41, 2022.

[41] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, pages 101–109, 2019.

[42] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, 39(2):1–49, 2021.

[43] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM conference on recommender systems*, pages 240–248, 2020.

[44] Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic ids: A practitioner's handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6420–6425, 2025.

[45] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In *Proceedings of the fifth ACM conference on Recommender systems*, pages 387–388, 2011.

[46] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[47] https://www.yelp.com/dataset. Yelp open dataset. 2023.

[48] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606, 2020.

[49] Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems*, pages 86–94, 2018.

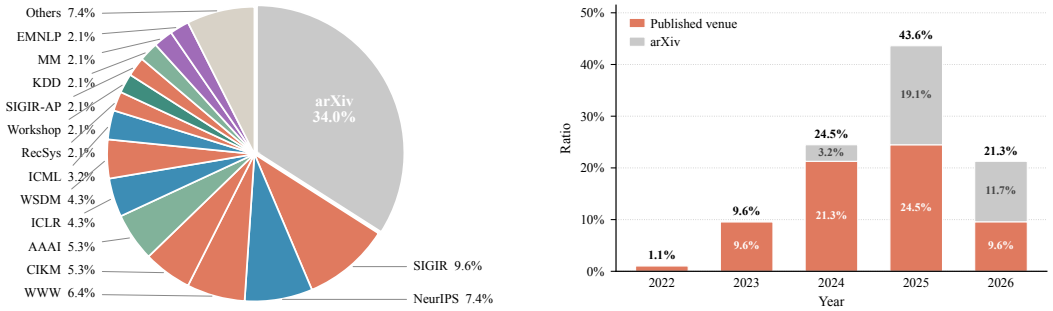
[50] https://www.kaggle.com/c/h-and-m-personalized-fashion recommendations/overview. H&m personalized fashion recommendations. 2022.

[51] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. Tokenrec: Learning to tokenize id for llm-based generative recommendations. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[52] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. Memory augmented graph neural networks for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5045–5052, 2020.

[53] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532, 2022.

A Details of Surveyed Papers



(a) Distribution of publication venues.

(b) Yearly share (venue vs. arXiv).

Figure 4: Statistics of the surveyed papers.

We collect 94 papers published between 2022 and 2026 that propose or evaluate *generative recommendation* methods; the full list of paper titles is provided in our code repository. The corpus statistics are summarized in Figure 4. These papers are gathered from major recommendation, information retrieval, machine learning, and data mining venues, as well as from arXiv. As shown in Figure 4a, arXiv preprints account for approximately 34.0% of the corpus, while the remaining papers are published in venues such as SIGIR, NeurIPS, WWW, CIKM, AAAI, ICLR, WSDM, ICML, and RecSys. Figure 4b further shows the rapid growth of generative recommendation research in recent years. The field expanded sharply after 2023: papers from 2024 account for 24.5% of the corpus, papers from 2025 account for 43.6%, and papers from 2026 already account for 21.3%, including preprints and accepted-but-not-yet-published works.

B Datasets

We conduct experiments on 14 widely used recommendation datasets covering e-commerce, review, news, music, movie, and game recommendation scenarios. To ensure fair comparisons with prior work, each dataset is preprocessed following the protocols used in previous studies. In general, we construct user interaction sequences according to interaction timestamps.

Amazon Review benchmarks [21]. The Amazon Review benchmarks are category-specific product recommendation datasets constructed from user reviews and interaction histories. We use four subsets: *Beauty*, *Sports*, *Toys*, and *CDs*. Following TIGER [14], GRID [44], and ActionPiece [16], we filter out users with fewer than 5 interactions. For item textual information, we retain the *title*, *price*, *brand*, and *categories* fields.

Delicious [45]. Delicious is a social bookmarking dataset that captures user interactions with web resources and their associated tags. We filter out users and items with fewer than 5 interactions and retain the *title* and *tags* as textual fields.

LastFM [45] and **MovieLens-1M (ML-1M)** [46]. LastFM is a music recommendation dataset based on user–artist interactions, while ML-1M is a movie recommendation dataset based on user ratings. We follow the preprocessing settings used in TokenRec [51]. Specifically, we filter out users with fewer than 5 interactions and truncate the maximum sequence length to 100. We keep *artist* and *tags* for LastFM, and *title* and *genres* for ML-1M.

Yelp [47]. Yelp is a business review dataset that reflects users' preferences over local businesses such as restaurants and services. Following LETTER [15], we filter out users with fewer than 5 interactions and retain the *title* and *description* fields as item textual information.

MIND [48]. MIND is a news recommendation dataset built from user click histories over online news articles. We use the validation split of the MIND-small dataset as our experimental dataset, which contains approximately 50K users and six weeks of click histories. Following [48], we retain the *news title* as the item textual feature and filter out users with fewer than 3 interactions.

GoodReads [49]. GoodReads is a book recommendation dataset constructed from users' reading and review histories. We use two subsets [52]: *GoodReads-Children* and *GoodReads-Comics*. We filter out users with fewer than 3 interactions and retain the *title*, *authors*, *publisher*, *year*, and *description* fields as textual information.

Steam [27]. Steam is a video game recommendation dataset based on user interactions with games on the Steam platform. We follow the commonly used preprocessing strategy in [27] and filter out users with fewer than 3 interactions. We retain the *title*, *developer*, *publisher*, and *tags* fields as textual information.

Amazon-M2-UK [33]. Amazon-M2 is a multi-market e-commerce dataset designed for recommendation across different regional Amazon marketplaces. We use the UK locale from the Amazon-M2 dataset and filter out users with fewer than 3 interactions. We retain the *title*, *brand*, *categories*, and *description* fields as textual information.

C Baselines

We compare our method with the following representative baselines:

- **ID-LAST** is a pure item-to-item collaborative filtering method trained with the BPR loss. It predicts the next item based on the last interacted item.
- **SEM-NN** ranks candidate items according to their pretrained language-model embedding similarity to the anchor item, serving as a content-based lower bound.
- **ID+SEM** combines ID-LAST and SEM-NN by integrating collaborative item-transition signals with semantic item similarity.
- **LightGCN** [10] simplifies GCN-based collaborative filtering by retaining only linear neighborhood aggregation and using layer-averaged embeddings as the final representation. For fair comparison with sequential baselines, we apply the same propagation rule on a global item-item transition graph constructed from training sequences. We also leverage the item text embeddings as the node features.
- **SR-GNN** [11] models each session as a directed graph and uses a gated graph neural network to capture complex item transitions. It then combines global session preference and current interest with attention for next-item recommendation.
- **SASRec** [27] is a Transformer-based sequential recommender that models user interaction histories with causal self-attention and predicts the next item by matching the sequence representation with item embeddings.
- **HSTU** [36] is a generative sequential recommender that reformulates ranking and retrieval as sequential transduction tasks. It encodes user actions into a unified sequence and uses efficient hierarchical sequential transduction units for long-sequence recommendation modeling.
- **TIGER** [14] is a generative retrieval framework for sequential recommendation. It represents each item with a short discrete semantic ID produced by an RQ-VAE [53] tokenizer trained on item content embeddings, and trains an encoder–decoder Transformer to autoregressively generate the semantic ID of the next item.
- **LETTER** [15] improves the RQ-VAE tokenizer by introducing collaborative-filtering alignment with SASRec [27] embeddings and a codebook-diversity loss to alleviate codebook collapse and semantic ID collisions.
- **CoFiRec** [19] extends the RQ-VAE tokenizer with a hierarchical design that separately quantizes textual views and collaborative signals into semantic IDs, which are then used by a TIGER-style encoder–decoder generator.

D Implementation Details

Following the standard leave-one-out evaluation protocol [27, 14], we use the last interacted item of each user for testing, the second-to-last item for validation, and the remaining interactions for training. For all baselines, we carefully follow the implementation details and hyperparameter settings suggested in their original papers. We truncate each user history to the most recent 50 items [16] for

ID-based methods and 20 items [15, 36] for generative methods. We set the latent dimension to 64 for all methods, except for LETTER, where we use a latent dimension of 32 following its original setting.

For generative methods, we reproduce TIGER [14] under the GRID [44] architecture, and implement LETTER [15] and CoFiRec [19] following their original papers. These methods generally follow a two-stage training recipe. In Stage 1, an RQ-VAE tokenizer compresses each item's pretrained text embedding into a 4-tuple of discrete codes, using $L{=}4$ residual codebooks with codebook size $K{=}256$. In Stage 2, a lightweight T5 encoder–decoder model is trained *from scratch* on user histories of up to $H{=}20$ items, where each item is represented as a flattened sequence of discrete codes. During inference, top-K recommendations are generated by beam search with beam size 10. Unless otherwise specified, all hyperparameters are kept consistent with those reported in the original papers. We use a single H200 GPU to train and inference all baselines.

E Detailed Results on Amazon Review Datasets

In Section 4, we report only Recall@10 (R@10) and NDCG@10 (N@10) on the four Amazon Review datasets. In this section, we provide the full results for Recall@1, Recall@5, Recall@10, NDCG@1, NDCG@5, and NDCG@10 on Beauty, Sports, Toys, and CDs. The detailed results are shown in Tables 6, 7, 8, and 9, respectively. From the results, we observe that the proposed TGH consistently outperforms all baselines across all datasets and evaluation metrics, further demonstrating its effectiveness on the Amazon Review benchmarks.

Table 6: Detailed results on **Beauty**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	1.89	4.41	6.16	1.89	3.17	3.74
SEM-NN	1.69	3.95	5.32	1.69	2.85	3.30
ID+SEM	2.08	4.87	6.66	2.08	3.51	4.08
LightGCN	2.18	5.21	7.21	2.18	3.71	4.36
SR-GNN	1.19	3.44	4.98	1.19	2.32	2.82
SASRec	1.21	3.99	6.21	1.21	2.59	3.31
HSTU	1.02	3.55	5.73	1.02	2.30	3.00
TIGER	1.48	4.40	6.41	1.48	2.95	3.60
LETTER	0.89	3.21	4.96	0.89	2.06	2.62
CoFiRec	1.23	4.09	6.24	1.23	2.67	3.36
TGH-1	**2.87**	**5.79**	7.66	**2.87**	**4.40**	5.01
TGH-2	**2.87**	5.78	**7.85**	**2.87**	**4.40**	**5.07**

Table 7: Detailed results on **Sports**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	0.83	1.92	2.70	0.83	1.38	1.63
SEM-NN	0.69	1.85	2.68	0.69	1.28	1.55
ID+SEM	0.88	2.31	3.18	0.88	1.63	1.91
LightGCN	0.92	2.46	3.69	0.92	1.70	2.10
SR-GNN	0.43	1.50	2.61	0.43	0.97	1.33
SASRec	0.81	2.14	3.23	0.81	1.48	1.83
HSTU	0.42	1.41	2.48	0.42	0.90	1.25
TIGER	0.70	2.30	3.70	0.70	1.51	1.96
LETTER	0.34	1.34	2.18	0.34	0.84	1.11
CoFiRec	0.69	2.45	3.82	0.69	1.58	2.02
TGH-1	1.58	3.17	4.27	1.58	2.41	2.76
TGH-2	**1.59**	**3.19**	**4.66**	**1.59**	**2.42**	**2.90**

Table 8: Detailed results on **Toys**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	2.18	5.03	6.81	2.18	3.65	4.22
SEM-NN	2.53	6.14	8.04	2.53	4.40	5.01
ID+SEM	2.86	6.28	8.53	2.86	4.64	5.37
LightGCN	2.68	6.50	8.63	2.68	4.63	5.32
SR-GNN	1.21	2.75	3.77	1.21	1.99	2.32
SASRec	1.74	5.26	7.41	1.74	3.54	4.23
HSTU	1.40	4.38	6.40	1.40	2.91	3.56
TIGER	1.12	4.07	6.01	1.12	2.62	3.27
LETTER	0.52	1.96	3.07	0.52	1.25	1.60
CoFiRec	0.84	3.59	5.63	0.84	2.22	2.89
TGH-1	3.71	**7.04**	9.13	3.71	5.45	6.12
TGH-2	**3.73**	7.01	**9.44**	**3.73**	**5.46**	**6.25**

Table 9: Detailed results on **CDs**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	1.51	3.56	4.91	1.51	2.56	3.00
SEM-NN	0.41	0.86	1.20	0.41	0.64	0.75
ID+SEM	1.53	3.63	5.02	1.53	2.60	3.05
LightGCN	0.70	1.76	2.63	0.70	1.24	1.52
SR-GNN	0.14	0.42	0.63	0.14	0.28	0.35
SASRec	0.82	2.73	4.44	0.82	1.79	2.33
HSTU	0.09	3.48	5.62	0.09	2.21	2.91
TIGER	0.24	0.92	1.44	0.24	0.58	0.75
LETTER	0.92	2.90	4.50	0.92	1.94	2.44
CoFiRec	1.08	3.62	5.64	1.08	2.36	3.01
TGH-1	2.32	**5.04**	6.21	2.32	**3.74**	4.13
TGH-2	**2.33**	**5.04**	**6.89**	**2.33**	**3.74**	**4.34**

F Detailed Results on More Datasets

For completeness, we further report the full per-metric results on the remaining ten benchmarks beyond the four Amazon Review datasets. These datasets cover diverse domains, including LastFM, Delicious, ML-1M, Yelp, MIND, Steam, GR-Children and GR-Comics, H&M, and Amazon-M2-UK. We report Recall@1, Recall@5, Recall@10, NDCG@1, NDCG@5, and NDCG@10 in Tables 10, 11, 12, 13, 14, 15, 16, 17, 18, and 19, respectively.

Across these ten datasets, TGH achieves the best or second-best performance on a large fraction of metrics. These results show that the proposed heuristic remains competitive beyond the Amazon Review benchmarks and can generalize to a wide range of domains and dataset scales. At the same time, the results also reveal that TGH is not universally superior, suggesting that different datasets exhibit different degrees of shortcut solvability.

Table 10: Detailed results on **LastFM**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	1.10	2.11	3.58	1.10	1.63	2.11
SEM-NN	0.64	3.39	5.14	0.64	2.09	2.66
ID+SEM	1.01	3.39	4.59	1.01	2.22	2.60
LightGCN	0.64	2.57	4.59	0.64	1.57	2.22
SR-GNN	0.55	1.74	3.03	0.55	1.15	1.56
SASRec	1.19	**4.13**	**6.24**	1.19	**2.65**	**3.32**
HSTU	0.91	2.11	3.57	0.91	1.52	2.01
TIGER	0.18	1.56	3.39	0.18	0.85	1.45
LETTER	0.64	2.11	3.67	0.64	1.30	1.79
TGH-1	**1.28**	3.39	5.50	**1.28**	2.38	3.07
TGH-2	**1.28**	0.39	4.86	**1.28**	2.38	2.87

Table 11: Detailed results on **Delicious**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	2.65	8.08	11.56	2.65	5.52	6.64
SEM-NN	0.97	2.92	4.74	0.97	2.04	2.62
ID+SEM	2.79	8.22	11.42	2.79	5.64	6.68
LightGCN	1.67	5.71	7.80	1.67	3.70	4.37
SR-GNN	2.79	5.43	7.52	2.79	4.10	4.78
SASRec	**4.18**	8.64	11.56	**4.18**	6.59	**7.57**
HSTU	1.53	5.71	8.21	1.53	3.53	4.35
TIGER	3.76	4.71	6.82	3.76	4.73	5.10
LETTER	0.50	2.93	3.90	0.50	1.75	2.08
TGH-1	3.62	9.19	10.72	3.62	6.54	7.06
TGH-2	3.62	**9.47**	**12.12**	3.62	**6.69**	7.54

Table 12: Detailed results on **ML-1M**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	2.76	8.34	12.27	2.76	5.60	6.86
SEM-NN	1.49	3.46	4.88	1.49	2.54	2.99
ID+SEM	2.90	8.69	12.62	2.90	5.83	7.09
LightGCN	1.52	4.97	8.39	1.52	3.22	4.32
SR-GNN	3.49	10.02	15.15	3.49	6.82	8.48
SASRec	**5.68**	**17.25**	**25.43**	**5.68**	**11.58**	**14.21**
HSTU	4.73	14.60	22.33	4.73	9.66	12.15
TIGER	3.56	12.30	18.54	3.56	7.97	9.98
LETTER	4.98	14.67	21.47	4.98	9.89	12.08
TGH-1	4.52	12.12	14.82	4.52	8.34	9.25
TGH-2	4.52	12.10	15.41	4.52	8.33	9.39

Table 13: Detailed results on **Yelp**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	0.32	1.06	1.86	0.32	0.69	0.94
SEM-NN	0.01	0.03	0.07	0.01	0.02	0.03
ID+SEM	0.24	0.75	1.23	0.24	0.50	0.65
LightGCN	0.12	0.47	0.78	0.12	0.30	0.40
SR-GNN	0.42	1.79	2.97	0.42	1.10	1.48
SASRec	0.39	1.51	2.70	0.39	0.95	1.33
HSTU	0.29	1.29	2.21	0.29	0.78	1.08
TIGER	0.36	1.28	1.97	0.36	0.83	1.05
LETTER	**0.61**	**2.10**	**3.34**	**0.61**	**1.35**	**1.76**
TGH-1	0.37	1.11	1.59	0.37	0.74	0.90
TGH-2	0.37	1.12	1.87	0.37	0.74	0.98

Table 14: Detailed results on **MIND**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	1.68	7.09	11.22	1.68	4.39	5.72
SEM-NN	0.07	0.14	0.25	0.07	0.10	0.14
ID+SEM	1.79	6.50	10.74	1.79	4.10	5.46
LightGCN	1.10	3.38	4.87	1.10	2.31	2.78
SR-GNN	**7.29**	**17.00**	22.01	**7.29**	**12.26**	**13.87**
SASRec	5.13	15.22	**22.73**	5.13	10.25	12.67
HSTU	2.45	9.01	14.50	2.45	5.72	7.49
TIGER	1.34	3.00	4.05	1.34	2.15	2.49
LETTER	1.54	3.24	4.28	1.54	2.37	2.70
TGH-1	1.77	7.56	9.93	1.77	4.70	5.51
TGH-2	1.77	7.57	9.50	1.77	4.70	5.32

Table 15: Detailed results on **STEAM**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	12.36	13.64	14.85	12.36	13.01	13.40
SEM-NN	12.36	13.88	14.60	12.36	13.16	13.40
ID+SEM	12.36	14.02	15.14	12.36	13.24	13.59
LightGCN	0.84	2.31	3.55	0.84	1.58	1.98
SR-GNN	12.15	15.80	18.71	12.15	14.00	14.93
SASRec	1.43	5.32	8.81	1.43	3.37	4.49
HSTU	1.41	5.17	8.41	1.41	3.28	4.32
TIGER	11.75	15.08	17.60	11.75	13.43	14.24
LETTER	12.72	**16.87**	**19.64**	12.72	**14.83**	**15.70**
TGH-1	**12.79**	16.00	17.23	**12.79**	14.43	14.85
TGH-2	**12.79**	16.00	17.65	**12.79**	14.43	14.95

Table 16: Detailed results on **GR-Children**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	4.87	11.01	15.31	4.87	8.01	9.39
SEM-NN	0.72	2.19	3.56	0.72	1.47	1.91
ID+SEM	4.04	8.30	10.68	4.04	6.23	7.00
LightGCN	4.17	9.74	12.73	4.17	7.07	8.03
SR-GNN	4.82	11.03	15.29	4.82	8.00	9.37
SASRec	5.30	14.74	20.83	5.30	10.14	12.10
HSTU	5.02	13.81	20.06	5.02	9.49	11.50
TIGER	5.42	12.38	17.11	5.42	8.96	10.48
LETTER	**7.74**	**17.13**	**22.87**	**7.74**	**12.54**	**14.39**
TGH-1	7.14	15.29	17.89	7.14	11.34	12.23
TGH-2	7.14	15.29	18.80	7.14	11.34	12.46

Table 17: Detailed results on **GR-Comics**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	14.24	20.76	23.62	14.24	17.77	18.70
SEM-NN	4.47	10.57	12.86	4.47	7.56	8.30
ID+SEM	13.72	20.74	23.74	13.72	17.46	18.43
LightGCN	14.49	22.30	25.36	14.49	18.63	19.62
SR-GNN	6.05	10.60	13.26	6.05	8.40	9.26
SASRec	8.62	19.31	25.33	8.62	14.14	16.08
HSTU	8.96	20.02	25.87	8.96	14.68	16.57
TIGER	11.68	17.96	21.18	11.68	14.97	16.02
LETTER	15.30	23.33	27.68	15.30	19.50	20.90
TGH-1	**17.79**	**26.41**	28.69	**17.79**	**22.35**	23.12
TGH-2	**17.79**	**26.41**	**30.48**	**17.79**	**22.35**	**23.66**

Table 18: Detailed results on **H&M**. **Bold**: best per metric; underlined: second-best.

Method	R@1	R@5	R@10	N@1	N@5	N@10
ID-LAST	0.29	0.91	1.49	0.29	0.60	0.79
SEM-NN	1.58	4.15	5.46	1.58	2.91	3.33
ID+SEM	1.99	4.63	5.88	1.99	3.37	3.77
LightGCN	1.61	3.59	4.45	1.61	2.66	2.93
SR-GNN	1.68	4.21	5.99	1.68	2.96	3.54
SASRec	1.36	4.42	6.87	1.36	2.90	3.68
HSTU	1.27	4.12	6.45	1.27	2.70	3.45
TIGER	1.60	5.12	7.30	3.40	1.60	4.10
LETTER	3.89	8.00	9.90	3.89	6.03	6.64
TGH-1	**4.92**	**9.97**	11.46	**4.92**	**7.56**	8.06
TGH-2	**4.92**	**9.97**	**13.51**	**4.92**	**7.56**	**8.70**

Table 19: Detailed results on **Amazon-UK**. **Bold**: best per metric; underlined: second-best.

Method	**R@1**	**R@5**	**R@10**	**N@1**	**N@5**	**N@10**
ID-LAST	17.79	29.18	33.97	17.79	23.83	25.38
SEM-NN	13.50	26.30	30.69	13.50	20.33	21.76
ID+SEM	18.16	30.27	35.07	18.16	24.61	26.16
LightGCN	**19.29**	**33.67**	**39.31**	**19.29**	**26.95**	**28.78**
SR-GNN	4.03	11.77	15.28	4.03	7.98	9.12
SASRec	6.84	17.25	23.10	6.84	12.20	14.10
HSTU	12.14	22.28	27.39	12.14	17.45	19.10
TIGER	3.76	10.34	13.39	2.30	7.11	8.10
LETTER	2.18	4.51	5.72	2.18	3.36	3.76
TGH-1	15.21	30.01	35.63	15.21	23.04	24.87
TGH-2	15.21	30.28	36.36	15.21	23.16	25.13

G Limitations

Our study has several limitations. First, the three shortcut structures we identify are not intended to be an exhaustive explanation of benchmark behavior. Sequential recommendation datasets may contain other shortcut signals, such as popularity effects, temporal regularities, repeated consumption patterns, or preprocessing artifacts. Our diagnostics provide a useful lens for interpreting the behavior of TGH, but they do not fully characterize all factors that influence model performance. Second, our analysis is based on a finite set of datasets and baselines. Although we evaluate on 14 datasets covering different domains, sequence lengths, graph structures, and feature signals, they still cannot represent all sequential recommendation scenarios. Similarly, while we include representative traditional, graph-based, sequential, and generative recommenders, new architectures or stronger implementations may behave differently. Third, our heuristic uses fixed hyperparameters and fixed retrieval budgets across datasets. This design keeps the heuristic simple and avoids dataset-specific configurations, but it may not be optimal for every dataset. Finally, our goal is not to argue that advanced sequential or generative recommenders are unnecessary. Instead, we show that some widely used benchmarks may not be sufficient to demonstrate their benefits. Future work can extend our diagnostics to more datasets, richer evaluation protocols, and more settings to better understand when advanced recommendation models provide genuine advantages.

H Broader Impacts

This work studies evaluation practice in sequential recommendation. Its main positive impact is to encourage more reliable and capability-aware benchmarking. By showing that some widely used datasets can be solved by simple shortcut signals, our analysis may help researchers avoid overclaiming model capabilities based on narrow benchmark results. It may also help benchmark creators release more transparent datasets by reporting diagnostic properties such as transition branching, feature-smoothness, and history dependence.

More careful benchmark selection can benefit both the research community and downstream users of recommender systems. If models are evaluated on datasets that better match their claimed capabilities, progress in semantic modeling, generative retrieval, and long-context recommendation can be measured more accurately. This may reduce wasted effort on optimizing for shortcut-solvable benchmarks and encourage the development of models that address harder and more realistic recommendation challenges.

Overall, we hope this work promotes more transparent evaluation practice in sequential recommendation, where both model designers and benchmark creators analyze dataset properties before drawing broad conclusions about model capability.