GLoSS: Generative Language Models with Semantic Search for Sequential Recommendation

Krishna Acharya
Georgia Institute of Technology
Atlanta, Georgia, USA
kacharya33@gatech.edu

Aleksandr V. Petrov
Viator, Tripadvisor
Glasgow, UK
spetrov@tripadvisor.com

Juba Ziani
Georgia Institute of Technology
Atlanta, Georgia, USA
jziani3@gatech.edu

Abstract

We propose **Generative Low-rank language model with Semantic Search (GLoSS)**, a generative recommendation framework that combines large language models with dense retrieval for sequential recommendation. Unlike prior methods such as GPT4Rec, which rely on lexical matching via BM25, GLoSS uses semantic search to retrieve relevant items beyond lexical matching. For query generation, we employ 4-bit quantized LlaMA-3 models fine-tuned with low-rank adaptation (LoRA), enabling efficient training and inference on modest hardware. We evaluate GLoSS on three real-world Amazon review datasets: Beauty, Toys, and Sports, and find that it achieves state-of-the-art performance. Compared to traditional ID-based baselines, GLoSS improves Recall@5 by 33.3%, 52.8%, and 15.2%, and NDCG@5 by 30.0%, 42.6%, and 16.1%, respectively. It also outperforms LLM-based recommenders such as P5, GPT4Rec, LlamaRec and E4SRec with Recall@5 gains of 4.3%, 22.8%, and 29.5%. Additionally, user segment evaluations show that GLoSS performs particularly well for cold-start users in the Amazon Toys and Sports datasets, and benefits from longer user histories in Amazon Beauty dataset, demonstrating robustness across different levels of interaction lengths. Our code and model checkpoints are publicly available at: https://github.com/krishnacharya/GLoSS

1 Introduction

Sequential Recommender Systems predict the next item in the sequence of user-to-item interaction. Until recently, *id-based* approaches dominated the field of sequential recommendation: these methods typically learn an embedding representation for every item in the catalog, as well as a *sequence encoder*—a model that represents the whole sequence of user-item interactions in the same space of embeddings. Id-based methods then compute item scores using a similarity function (e.g., dot product) between the sequence embedding and the item embedding. Some of the most notable examples of id-based methods include SASRec [16] and BERT4Rec [35].

Despite achieving high recommendation accuracy on many sequential datasets, id-based methods have a number of limitations. In particular, these id-based methods require item embeddings to be *learned* for every item in the catalog, meaning that new items can be only recommended after the recommendation model has been retrained. In addition, models like SASRec and BERT4Rec are trained only using data directly presented in the training set, which constrains their ability to generalize beyond seen patterns or infer relevance based on rich item metadata or natural language descriptions.

In contrast, textual document retrieval systems do not suffer from these limitations: documents (equivalent of "items" in recommender systems) are typically indexed independently of the scoring model, and, hence can be retrieved immediately after being added to index. Moreover, the widespread adaptation of pre-trained language models (such as BERT [7]), has enabled fine-tuning for retrieval, allowing the models to generalize beyond the training data and retrieve relevant documents even in the absence of direct lexical overlap, leading to substantial improvements in retrieval effectiveness [17, 32].

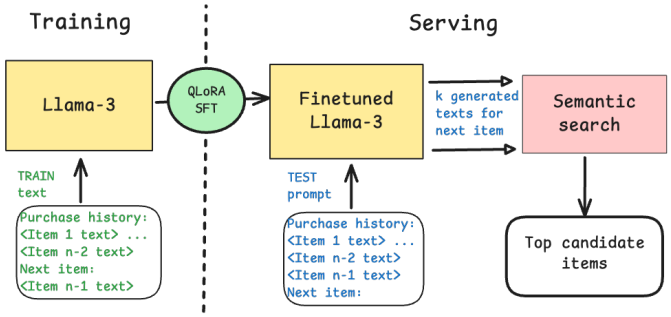

Figure 1: GLoSS: a finetuned LLM generates queries, which are then used to perform semantic search over the item catalog.

Recent advances in generative language models have enabled their integration into recommendation pipelines. For instance, P5Rec [9] adopts T5 [30] as a unified backbone to support multiple tasks including sequential recommendation, rating prediction, and review summarization. GPT4Rec [21] uses GPT-2 to generate multiple queries per user, which are then matched to items via a BM25 lexical retriever. More recently, LlamaRec [41] employs LLaMA-2 as a reranker over candidates selected by an LRU-based recommender [42]. Other methods, such as CALRec and GenRec [14, 23], follow a similar query generation and retrieval framework, combining LLMs like PaLM [2] or LLaMA-2 [36] with BM25-based lexical search.

However, many of these methods use BM25, a lexical matching algorithm that does not capture the semantic meaning of generated queries. Furthermore, approaches like P5, GPT4Rec, and CALRec rely on older language models (e.g., T5, GPT-2, PaLM) and use full fine-tuning, which is computationally intensive.

Over the past five years, generative language models have improved significantly due to advancements in architecture, better training data, and more effective training methods. This progress has resulted in models like the LLaMA-3 series [10], which are substantially more capable for tasks such as query generation. Additionally, parameter-efficient fine-tuning techniques such as QLoRA [6] have dramatically reduced the resource requirements for training, enabling efficient fine-tuning on consumer GPUs.

Inspired by the GPT4Rec approach and aiming to address its limitations, we propose **Generative Low-rank language model with Semantic Search (GLoSS)**, a generative recommendation framework for sequential recommendation. Unlike GPT4Rec, GLoSS uses

semantic search (aka. dense retrieval) to enable item recommendation beyond direct lexical matching. Our query generation backbone uses 4-bit quantized LLaMA-3 models fine-tuned via low-rank adapters.

We evaluate GLoSS on three real-world Amazon review datasets: Beauty, Toys, and Sports [11]—and demonstrate that it achieves state-of-the-art performance. Compared to ID-based baselines, GLoSS yields substantial gains in Recall@5 (33.3% on Beauty, 52.8% on Toys, and 15.2% on Sports) and NDCG@5 (30.0%, 42.6%, and 16.1%, respectively). GLoSS also outperforms LLM-based recommenders such as P5, GPT4Rec, LlamaRec and E4SRec, with Recall@5 improvements of 4.3% on Beauty, 22.8% on Toys, and 29.5% on Sports—while maintaining competitive NDCG@5 without relying on a reranking stage.

In summary, the contributions of this paper are as follows

(1) We apply low-rank adaptation (LoRA) to fine-tune 4-bit quantized LLaMA-3 models for query generation, showing that high-quality generative performance can be achieved with minimal resources.
(2) We integrate dense retrieval into the generative recommendation pipeline to enable retrieval beyond lexical matching. Our analysis in Section 4.2.2 shows that dense retrieval is a critical component, yielding substantial gains in NDCG@5 and Recall@5 over BM25.
(3) Through extensive experiments on Amazon datasets, we show that GLoSS outperforms both ID- and LLM-based baselines with consistent double-digit improvements in Recall@5 and NDCG@5.
(4) We analyze GLoSS across user segments based on interaction history length: cold-start, regular, and power users. To our knowledge, this is the first such study for LLM-based recommenders. GLoSS performs best on cold-start users in Toys and Sports, and benefits from longer histories in Beauty, demonstrating robustness across user interaction lengths.

The rest of the paper is organised as follows: Section 2 contains the description of similar methods in RecSys and IR; Section 3 describes our proposed GLoSS; Section 4 describes experimental evaluation; Section 5 contains concluding remarks. We now describe prior related to ours.

2 Recommendation through Retrieval

This section describes methods from the fields of Recommender Systems and Information Retrieval. Section 2.1 describes classic content-based methods; Section 2.2 describes recent generative methods; Section 2.3 discusses similar methods from the IR field.

2.1 Content-based filtering

Our work retrieves items from the catalog using textual description, generated by a language model. Hence, our method has similarities with *content-based* recommendation methods that were developed in the early days of the recommender systems research. Indeed, in the mid-1990s, Lang [20] developed a news recommendation algorithm that used textual representation of past user netnews articles to recommend future articles to read, using lexical matching. Pazzani et al., developed the "Syskill & webert" software agent [27] that generated a search query to find interesting websites based on past visited websites. Balabanović et al. developed a famous "FAB" recommender system that combined textual matching with id-based collaborative filtering. While these early content-based methods resemble certain similarities with our proposed method, they have relied on simplistic natural language methods, such as TF-IDF [34] vector space model. Due to the simplicity of these models, since the early 2000s, the id-based methods (e.g. Matrix Factorization [18]) showed better performance on many recommendation tasks and since then they remain de-facto standard. However, given the rapid progress of Large Language Models within the the last few years, we argue that a new generation of content-based models can make an unexpected comeback. Indeed, our proposed GLoSS can be seen as a content-based method, as it does not learn item representations and only uses their textual representations for matching. Nevertheless, in our experiments it achieves superior performance compared to state-of-the-art ID-based methods, such as SASRec, TIGER and ActionPiece.

2.2 Generative Sequential Recommendation

Recently, a number of works used generative models to directly generate item id. Some of the most notable examples include HSTU [43] that used atomic item ids, P5 [9] that learns to generate numeric item IDs represented as strings, TIGER [31] that uses *semantic IDs* obtained from item embeddings, GPTRec[1] [28, 29] that obtains item ids from collaborative item embeddings, ActionPiece [13] that constructs item ids from unordered set of its features. A central challenge addressed by these works is designing item id representations that generative models can effectively learn and reproduce. However, regardless of how item ids are represented, large language models have not encountered these ids during pre-training (as they are specific for the method). As a result, the models must learn to generate item ids from scratch, without leveraging the vast internet-scale training data that typically benefits other text generation tasks.

In contrast, in our GLoSS—similarly to GPT4Rec [21], which served as our inspiration—a large language model is used to generate textual queries for a retrieval system. That is, we employ the LLM in its native textual domain, without expanding its vocabulary or requiring it to learn atypical token sequences. This design allows GLoSS to fully leverage the internet-scale corpus used during the model's pretraining.

2.3 Automatic Query Generation in Information Retrieval

Our approach uses an LLM to formulate queries which are then sent to a retrieval system in order to retrieve relevant items. This is similar to automatic query generation approaches in Information Retrieval. For example, Cui et al. [4] used search terms mined from user's log data to expand search queries. Jones et al. [15] developed a statistical model that returned new queries that could be used as the substitutions to original queries. Recently, LLMs have also been employed for query generation. For example, Ye et al. [40]. used LLMs to rewrite the original query in order to retrieve better search results. While these methods have some similarities with our method, they are focused on the classic information retrieval task, where user provides an explicit query. In contrast, our approach focuses on recommendation task, that does not require user to provide any explicit inputs.

With this, we conclude the overview of related work. In summary, no prior work uses a large language model to generate natural language queries for semantic item retrieval in a recommendation

[1] not to be confused with GPT4Rec

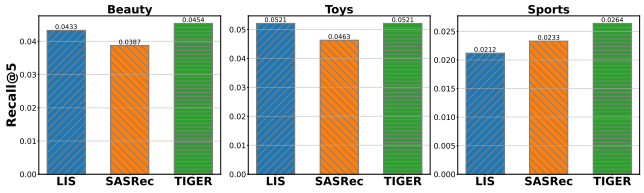

Figure 2: Last-item text-based search (LIS) is a simple, training-free baseline that performs competitively on Recall@5 with modern sequential models like SASRec [16] and TIGER [31]. Corresponding tables in Appendix E.1

setting. The most similar approach to ours is GPT4Rec [21], which relies on lexical matching rather than dense retrieval; we provide a detailed comparison between GLoSS and GPT4Rec in Section 3.4. We now describe details of GLoSS methodology.

3 GLoSS Methodology

In this section, we describe our methodology. Section 3.1 formalizes the sequential recommendation task. Section 3.2 describes LIS, a simple training-free content-based model we use as a baseline. Section 3.3 describes our GLoSS pipeline. Section 3.4 compares GloSS with the most related methods, GPT4Rec and LlamaRec.

3.1 Sequential recommendation

Assume that we have a set of users and items, denoted by $\mathcal{U}$ and $\mathcal{I}$, respectively, where $u \in \mathcal{U}$ denotes a user and $i \in \mathcal{I}$ denotes an item. The numbers of users and items are denoted as $|\mathcal{U}|$ and $|\mathcal{I}|$, respectively. A user u is represented by a chronologically-ordered interaction sequence with items $\{i_1, \ldots, i_{n-1}\}$, where $n-1$ is the number of interactions and i_t is the t-th item that the user u has interacted with. For convenience, we use $i_{j:k}$ to denote the subsequence, i.e., $i_{j:k} = \{i_j, \ldots, i_k\}$ where $1 \le j \le k \le n-1$. Besides, each item i is associated with several attributes $\mathcal{A}_i = \{a_1, \ldots, a_m\}$. For example, a product on Amazon is associated with its title, category, etc. The set of all attributes is denoted the attribute set $\mathcal{A}$, and the number of attributes is denoted as $|\mathcal{A}|$.

Based on the above notations, we now define the task of sequential recommendation. Formally, given the historical behaviors of a user $\{i_1, \ldots, i_{n-1}\}$ and the attributes $\mathcal{A}_i$ of each item i, the task of sequential recommendation is to predict the next item that the user is likely to interact with at the n-th step.

3.2 Last Item Search (LIS) baseline

Drawing on the content-based filtering methods discussed in Section 2.1, we began with a preliminary investigation using **Last Item text Search (LIS)**—a simple, training-free[2] baseline that retrieves recommendations based on the text content of the last item a user interacted with. Despite its simplicity, this method performs surprisingly well, achieving Recall@5 scores comparable to those of modern sequential recommenders like SASRec and TIGER across the Beauty, Toys, and Sports datasets. Notably, the results shown in Figure 2 use BM25 [33], a classic lexical retriever developed over three decades ago.

[2]We do not consider computing IDF weights in BM25 as training

3.3 GLoSS pipeline

The GLoSS pipeline is illustrated in Figure 1. Given a user's item sequence, we: i) *Serialize* the user's interaction history into natural language text using item metadata (e.g., titles), and *Finetune* a large language model on this data; ii) *Generate* candidate texts representing the next likely item; iii) *Retrieve* items from the catalog that closely match these candidate texts. Here, we present a high-level overview—further implementation details can be found in Section 4.1.4.

3.3.1 Serialize and Finetune. We illustrate our serialization process with a simple example. Suppose a user has interacted with $n = 4$ items. The corresponding training text is constructed using the first $n-1 = 3$ items and structured as shown below. We then fine-tune a large language model using QLoRA [6] on this serialized training data.

Training Text

Below is a customer's purchase history on Amazon, listed in chronological order (earliest to latest). Each item is represented by the following format: `Title: <item title>`. Based on this history, predict only one item the customer is most likely to purchase next in the same format.
Purchase history:
Title: 63cm Long Zipper Beige+pink Wavy Cosplay Hair Wig Rw157
Title: MapofBeauty Long Wave Curly Hair Wig Full Wig for Women Long (Black)
Next item:
Title: 32" 80cm Long Hair Heat Resistant Spiral Curly Cosplay Wig (Red Dark)

Test Prompt

Below is a customer's purchase history on Amazon, listed in chronological order (earliest to latest). Each item is represented by the following format: `Title: <item title>`. Based on this history, predict only one item the customer is most likely to purchase next in the same format.
Purchase history:
Title: 63cm Long Zipper Beige+pink Wavy Cosplay Hair Wig Rw157
Title: MapofBeauty Long Wave Curly Hair Wig Full Wig for Women Long (Black)
Title: 32" 80cm Long Hair Heat Resistant Spiral Curly Cosplay Wig (Red Dark)
Next item:

3.3.2 Generate. To obtain candidate texts for the test target item, we feed the above test prompt, which includes all previous items in the purchase history, into the LLM. The model then continues generation after the "Next item:\n" token.

3.3.3 Retrieve. We construct a dense retrieval index over the item metadata corpus and store it for inference. For each generated text, we perform semantic search by computing the dot product between its embedding and the embeddings of all catalog items, retrieving the closest match.

3.4 Comparison with GPT4Rec and LlamaRec

In this section, we describe the differences between GLoSS and two related methods: GPT4Rec [21], which also employs a "generate-then-retrieve" pipeline, and LlamaRec [41], which similarly utilizes LLaMA-series models for recommendation.

First, GPT4Rec [21] is the most similar approach to ours, and we explicitly acknowledge it as the source of our inspiration. The main differences from GPT4Rec include: (i) the use of semantic search instead of BM25-based lexical matching; (ii) the adoption of the modern LLaMA-3 backbone instead of the older GPT-2; and (iii) the use of QLoRA for parameter-efficient fine-tuning. Collectively, these improvements enable us to achieve higher effectiveness compared to GPT4Rec and reach state-of-the-art results (see Section 4.2).

Second, we are not the first to apply LLaMA to recommendation. Yue et al. proposed LlamaRec [41], which also incorporates LLaMA within the recommendation pipeline. However, its usage differs significantly from ours: LlamaRec employs the LLM for ranking candidates retrieved by a simpler model. In contrast, GLoSS uses LLaMA to retrieve high-quality candidates directly, without additional reranking. Since GLoSS leverages a stronger model at the retrieval stage, it is expected to retrieve a superior set of candidates compared to LlamaRec (i.e., we expect higher recall). However, as we do not perform reranking, ranking-specific metrics such as NDCG may be weaker than those of LlamaRec. The retrieval enhancements of GLoSS and the reranking strengths of LlamaRec are likely complementary. Nevertheless, due to the unavailability of LlamaRec's trained checkpoints, we leave the investigation of this potential addition for future work.

This summarizes the differences of GLoSS with GPT4Rec and LlamaRec. We now turn to experimental evaluation of GLoSS.

4 Experiments

This section contains experimental evaluation of GLoSS. Section 4.1 describes the setup we use for our experiments; Section 4.2 contains the results of our experiments.

4.1 Setup

4.1.1 Datasets. We perform experiments on three real-world benchmarks from the Amazon Product Reviews dataset [11]: "Beauty" (**Beauty**), "Toys and Games" (**Toys**), "Sport and Outdoors" (**Sports**). These datasets contain user reviews and item metadata collected between May 1996 and July 2014. Details on preprocessing and dataset statistics are provided in Appendix B.

4.1.2 Evaluation. We adopt the widely used leave-last-out split [16, 26, 35], where the last item in the interaction sequence is used for testing and the previous items used for training.

For evaluation, we utilize two widely adopted top-k metrics: Recall@5 (hit rate) and Normalized Discounted Cumulative Gain at 5 (NDCG@5). Recall@5 assesses whether a relevant item appears within the top-5 predictions, while NDCG@5 also considers the ranked position of the target. To ensure fair and unbiased evaluation, we compute these metrics by scoring all items in the catalog. This avoids the bias of sampled metrics, which can favor popular items [3, 19]. While computationally intensive, full-catalog evaluation provides more accurate results, as sampled metrics are inconsistent with exact versions and can misrepresent recommender performance.

4.1.3 Baselines. We compare the performance of GLoSS against 10 existing models, grouped into ID-based and LLM-based. The ID-based are further classified as Classic, Feature-enhanced and Semantic ID-based.

- **Classic ID-based**: SASRec [16] and BERT4Rec [35].
- **Feature-enhanced ID-based**: FDSA [44] and S3Rec [45].
- **Semantic ID-based**: TIGER [31] and ActionPiece [13].
- **LLM-based**: P5 [9], GPT4Rec [21], LlamaRec [41] and E4SRec [22]

A detailed description of these models is provided in Appendix D. Results for the ID-based baselines are taken from publicly available metrics reported in the ActionPiece paper (Appendix G of [13]). For LLM-based recommenders, we report results as published in P5 [9], GPT4Rec [21], LlamaRec [41] and E4SRec [22]. Notably, only the Beauty dataset is consistently used across these works, while standard benchmarks like Toys and Sports are not reported for GPT4Rec and LlamaRec. We do not reproduce their models on these datasets due to (i) the lack of open-source code models (e.g., GPT4Rec) and (ii) compute constraints[3]. This aligns with common practice in the LLM-based recommender literature, where both training and inference are more resource-intensive than ID-based methods. For instance, LlamaRec [41] reports results only on the Beauty dataset for this reason.

4.1.4 Implementation Details for GLoSS.

Finetuning. The LLMs used in our GLoSS pipeline are from the LLaMA-3 model family [10]. We use models with increasing parameter sizes: LLaMA 3.2 (1B, 3B) and LLaMA 3.1 (8B). We limit the maximum context length to 1024 tokens, truncating from the left if necessary. The models are finetuned using QLoRA [6] via the Unsloth library [5], with 4-bit quantization and LoRA rank and α set to 16. This results in 11M, 24M, and 42M trainable adapter parameters for the 1B, 3B, and 8B models, respectively, enabling training on a single RTX A5000 GPU with 24GB memory.

We train for 10 epochs. Our batch size and gradient accumulation are 4 each, yielding an effective batch size of 16. Optimization is performed using AdamW with a learning rate of 0.0001, a linear schedule with 300 warm-up steps, and a weight decay of 0.01. Early stopping is based on validation loss, with a patience of 3 epochs.

Generation. For text generation, we use beam search decoding with 5 beams, producing 5 candidate texts per user. The `max_new_tokens` i.e., the tokens generated after "Next item:") is set to 50.

Retrieval. For retrieval, we use the `e5-small-v2` encoder[4] to index the item catalog [38]. This model has 33M parameters and produces embeddings of dimension $d = 384$. The dense index is constructed using the `retriv` package[5].

We denote each GLoSS variant with the suffix **GLoSS-1B**, **GLoSS-3B**, or **GLoSS-8B**, indicating the LLaMA-3 model size used for query generation. Unless otherwise specified, experiments in Sections 4.2.1 and 4.2.3 use `e5-small-v2` as the retriever. In Section 4.2.2, we compare against a larger dense retriever (`e5-base-v2`) and a sparse BM25 retriever to assess the importance of dense retrieval and whether encoder size impacts performance. The code,

[3] Our experiments are limited to a single RTX A5000 GPU.
[4] https://huggingface.co/intfloat/e5-small-v2
[5] https://github.com/AmenRa/retriv

datasets and checkpoints are open-sourced at https://github.com/krishnacharya/GLoSS.

4.2 Results

In this section, we evaluate the performance of our GLoSS pipeline on the three Amazon datasets—Beauty, Toys, and Sports. Our evaluation is guided by the following key research questions, which aim to measure how GLoSS compares to existing sequential recommenders (both ID-based and LLM-based), and to examine the impact of dense retrieval, and user interaction length:

RQ1 Does GLoSS outperform existing sequential recommenders, including both ID-based and LLM-based baselines?
RQ2 What is the effect of using dense retrieval, and does query encoder size matter?
RQ3 How does GLoSS perform across users with varying interaction lengths, and can it effectively handle cold-start users?
We now answer these research questions in turn.

4.2.1 RQ1: Does GLoSS outperform existing baselines? We train three GLoSS variants—GLoSS-1B, GLoSS-3B, and GLoSS-8B, corresponding to different sizes of the LLaMA-3 backbone. In Table 1, we compare these against six ID-based baselines: SASRec, BERT4Rec, FDSA, S3Rec, TIGER, and ActionPiece. GLoSS-8B consistently outperforms all ID-based methods, achieving state-of-the-art performance. The improvements in Recall@5 are substantial, with gains of 33.27%, 52.78%, and 15.19% on the Beauty, Toys, and Sports datasets respectively. Similarly, we observe NDCG@5 improvements of 30%, 42.59%, and 16.1%. We also find that performance improves with larger LLaMA model sizes, which is expected given the enhanced generation quality of larger language models.

In Table 2, we compare GLoSS against four LLM-based recommenders: P5, GPT4Rec, LlamaRec, and E4SRec. GLoSS achieves higher Recall@5 across all datasets—showing gains of 4.29%, 22.84%, and 29.54% on Beauty, Toys, and Sports, respectively. While GLoSS is competitive in NDCG@5, it does not surpass LlamaRec, which benefits from an additional reranking stage. As discussed in Section 3.4, LlamaRec retrieves candidates using a simpler model and applies an LLM for reranking, optimizing ranking metrics. In contrast, GLoSS uses LLaMA directly for retrieval, yielding high-quality candidates and strong recall. The two approaches are likely complementary, but we leave exploration of a combined system to future work due to the unavailability of LlamaRec's checkpoints.

We also note that P5's reported results on the Toys dataset are likely optimistic; a recent reproduction study [24] reports significantly lower scores. However, to ensure a conservative and robust comparison, we retain the numbers from the original P5 paper [9].

These results support that GLoSS achieves state-of-the-art performance among both ID-based and LLM-based recommenders, demonstrating the efficacy of integrating dense retrieval with strong LLM generation backbones.

4.2.2 RQ2 How important is dense retrieval and does encoder size matter? We investigate the impact of dense retrieval by directly comparing it to traditional lexical retrieval using BM25. Specifically, we evaluate two dense retrievers of increasing model capacity: `e5-small-v2` (33M parameters, embedding dimension $d = 384$) and `e5-base-v2` (109M parameters, $d = 768$). As shown in Table 3, dense retrieval consistently outperforms BM25, with substantial gains in NDCG@5, reaching double digits. Improvements in Recall@5, while generally smaller in magnitude, still reflect the value of semantic representations in identifying relevant items beyond lexical matches. Interestingly, a larger encoder does not always yield better performance; for example, on the Beauty dataset, `e5-small-v2` outperforms `e5-base-v2` when used with the 1B and 8B GLoSS variants.

4.2.3 RQ3: How does GLoSS perform across users with varying interaction histories, and can it effectively handle cold-start users? We categorize users into three groups based on the length of their interaction sequences. Given a user history $U_i = \{i_1, i_2, ..., i_n\}$, $|U_i| = n$, users are classified as:

- **Short-sequence (cold-start) users:** $n \leq l$
- **Medium-sequence (regular) users:** $l < n \leq h$
- **Long-sequence (power) users:** $n > h$

The choices for the l, h thresholds and the corresponding group sizes are available in Appendix C.2. This segmentation allows us to evaluate how model performance varies with the size of user history. Having a model that recommends well with limited interactions is critical for new customer retention [37, 46], and we note that similar user segment-wise evaluation has been proposed in prior work [1, 39]

In the interest of space, Tables 6 and 7 reporting user segment performance are presented in Appendix C.1. These show that GLoSS achieves high Recall@5 and NDCG@5 for cold-start users. In fact, in the Toys and Sports datasets, users with short interaction histories achieve the highest performance. In contrast, the Beauty dataset shows the opposite trend: performance improves with longer histories. This suggests that in Toys and Sports, recent interactions may provide more distinctive signals, whereas in Beauty, longer histories offer richer context. Similar trends are observed with the Last-item text search baseline in Appendix E.1, indicating that this is a dataset-specific rather than model-specific effect.

In any case, GLoSS shows robust performance across datasets, remaining effective in both sparse and interaction-rich scenarios.

5 Conclusion

In this paper, we introduced GLoSS (Generative Low-rank language model with Semantic Search), a novel generative recommendation framework. GLoSS distinguishes itself from prior approaches like GPT4Rec by replacing lexical BM25 retrieval with semantic search, leading to more accurate and context-aware item retrieval. For efficient query generation, GLoSS leverages 4-bit quantized LLaMA-3 models fine-tuned via LoRA, enabling efficient training and inference without sacrificing quality.

Our comprehensive experiments on three real-world Amazon review datasets—Beauty, Toys, and Sports—demonstrate that GLoSS consistently achieves state-of-the-art performance. It significantly outperforms ID-based baselines in both Recall@5 and NDCG@5, and surpasses existing LLM-based recommenders on Recall@5. While GLoSS shows competitive but sometimes slightly lower NDCG scores compared to LLamaRec on Beauty and P5Rec on Toys, a promising future direction involves incorporating LLMs as

Table 1: Performance of GLoSS compared to ID-based sequential recommenders. The best overall metric is shown in bold, and the second best is underlined. The † symbol indicates the best-performing baseline (excluding our models). GLoSS-8B achieves the best performance across all datasets, with percentage improvements over † reported in brackets.

Model	Beauty		Toys		Sports	
	Recall@5	NDCG@5	Recall@5	NDCG@5	Recall@5	NDCG@5
SASRec	0.0387	0.0249	0.0463	0.0306	0.0233	0.0154
BERT4Rec	0.0203	0.0124	0.0116	0.0071	0.0115	0.0075
FDSA	0.0267	0.0163	0.0228	0.0140	0.0182	0.0122
S3Rec	0.0387	0.0244	0.0443	0.0294	0.0251	0.0161
TIGER	0.0454	0.0321	0.0521†	0.0371†	0.0264	0.0181
ActionPiece	0.0511†	0.0340†	N/A	N/A	0.0316†	0.0205†
GLoSS-1B	0.0456	0.0297	0.0677	0.0441	0.0226	0.0145
GLoSS-3B	0.0653	0.0423	0.0728	0.0478	0.0294	0.0188
GLoSS-8B	**0.0681** (+33.27%)	**0.0442** (+30.00%)	**0.0796** (+52.78%)	**0.0529** (+42.59%)	**0.0364** (+15.19%)	**0.0238** (+16.10%)

Table 2: Performance of GLoSS compared to LLM based recommenders. The best metric overall is in bold and second best is underlined. The † symbol indicates the best-performing baseline (excluding our models). GLoSS-8B achieves the highest Recall@5 across all datasets.

Model	Beauty		Toys		Sports	
	Recall@5	NDCG@5	Recall@5	NDCG@5	Recall@5	NDCG@5
P5	0.0503	0.0370	0.0648†	**0.0567**†	0.0272	0.0169
GPT4Rec	0.0653†	N/A	N/A	N/A	N/A	N/A
LlamaRec	0.0648	**0.0450**†	N/A	N/A	N/A	N/A
E4SRec	0.0525	0.0360	0.0566	0.0405	0.0281†	0.0196†
GLoSS-1B	0.0456	0.0297	0.0677	0.0441	0.0226	0.0145
GLoSS-3B	0.0653	0.0423	0.0728	0.0478	0.0294	0.0188
GLoSS-8B	**0.0681** (+4.29%)	0.0442 (-1.77%)	**0.0796** (+22.84%)	0.0529 (-6.70%)	**0.0364** (+29.54%)	**0.0238** (+21.43%)

Table 3: Comparison of different retrievers on generated texts. Dense retrieval outperforms BM25, showing substantial gains in NDCG@5 and moderate improvements in Recall@5. Brackets indicate percentage improvements over BM25.

Model Configuration		**Beauty**		**Toys**		**Sports**	
		Recall@5	**NDCG@5**	**Recall@5**	**NDCG@5**	**Recall@5**	**NDCG@5**
1B	**BM25**	0.0453	0.0269	0.0666	0.0401	0.0221	0.0133
	e5-small-v2	0.0456 (+0.66%)	0.0297 (+10.41%)	0.0677 (+1.65%)	0.0441 (+9.98%)	0.0226 (+2.26%)	0.0145 (+9.02%)
	e5-base-v2	0.0458 (+1.10%)	0.0295 (+9.67%)	0.0675 (+1.35%)	0.0443 (+10.47%)	0.0228 (+3.17%)	0.0139 (+4.51%)
3B	**BM25**	0.0652	0.0400	0.0720	0.0436	0.0288	0.0178
	e5-small-v2	0.0653 (+0.15%)	0.0423 (+5.75%)	0.0728 (+1.11%)	0.0478 (+9.63%)	0.0294 (+2.08%)	0.0188 (+5.62%)
	e5-base-v2	0.0652 (+0.00%)	0.0425 (+6.25%)	0.0728 (+1.11%)	0.0474 (+8.72%)	0.0295 (+2.43%)	0.0182 (+2.25%)
8B	**BM25**	0.0681	0.0423	0.0784	0.0472	0.0359	0.0218
	e5-small-v2	0.0681 (+0.00%)	0.0442 (+4.49%)	0.0796 (+1.53%)	0.0529 (+12.08%)	0.0364 (+1.39%)	0.0238 (+9.17%)
	e5-base-v2	0.0683 (+0.29%)	0.0435 (+2.84%)	0.0790 (+0.77%)	0.0526 (+11.44%)	0.0365 (+1.67%)	0.0228 (+4.59%)

judges for reranking; this could further improve NDCG and overall recommendation quality. Beyond overall performance, GLoSS demonstrates high performance robustness across user segments, with notably high performance for cold-start users on the Sports and Toys datasets.

Acknowledgments

The authors gratefully acknowledge the compute resources used in these experiments, provided by Simian—a GPU workstation set up by Prof. Jacob Abernethy and maintained by PhD student Tyler LaBonte. Prof. Ziani's research was supported by NSF CAREER Award IIS-2336236.

References

[1] Krishna Acharya, David Wardrope, Timos Korres, Aleksandr V Petrov, and Anders Uhrenholt. 2025. Improving Minimax Group Fairness in Sequential Recommendation. In *European Conference on Information Retrieval*. Springer, 355–370.

[2] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403* (2023).

[3] Rocío Cañamares and Pablo Castells. 2020. On target item sampling in offline recommender system evaluation. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 259–268.

[4] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2003. Query expansion by mining user logs. *IEEE transactions on knowledge and data engineering* 15, 4 (2003), 829–839.

[5] Michael Han Daniel Han and Unsloth team. 2023. *Unsloth*. http://github.com/unslothai/unsloth

[6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLORA: efficient finetuning of quantized LLMs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 441, 28 pages.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 4171–4186.

[8] Dario Di Palma, Felice Antonio Merra, Maurizio Sfilio, Vito Walter Anelli, Fedelucio Narducci, and Tommaso Di Noia. 2025. Do LLMs Memorize Recommendation Datasets? A Preliminary Study on MovieLens-1M. *arXiv preprint arXiv:2505.10212* (2025).

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

[10] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[12] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web* (Montréal, Québec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 507–517. doi:10.1145/2872427.2883037

[13] Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian McAuley, and Derek Zhiyuan Cheng. 2025. ActionPiece: Contextually Tokenizing Action Sequences for Generative Recommendation. *arXiv preprint arXiv:2502.13581* (2025).

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

[15] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web* (Edinburgh, Scotland) *(WWW '06)*. Association for Computing Machinery, New York, NY, USA, 387–396. doi:10.1145/1135777.1135835

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

[17] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proc. EMNLP* (2020-09-30). arXiv:2004.04906 [cs] http://arxiv.org/abs/2004.04906

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

[19] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (June 2022), 75–83. doi:10.1145/3535335

[20] Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995*, Armand Prieditis and Stuart Russell (Eds.). Morgan Kaufmann, San Francisco (CA), 331–339. doi:10.1016/B978-1-55860-377-6.50048-7

[21] Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. GPT4Rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879* (2023).

[22] Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443* (2023).

[23] Yaoyiran Li, Xiang Zhai, Moustafa Alzantot, Keyi Yu, Ivan Vulić, Anna Korhonen, and Mohamed Hammad. 2024. Calrec: Contrastive alignment of generative llms for sequential recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 422–432.

[24] Pasquale Lops, Antonio Silletti, Marco Polignano, Cataldo Musto, and Giovanni Semeraro. 2024. Reproducibility of LLM-based Recommender Systems: the Case Study of P5 Paradigm. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 116–125. doi:10.1145/3640457.3688072

[25] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 43–52. doi:10.1145/2766462.2767755

[26] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 681–686. doi:10.1145/3383313.3418479

[27] Michael Pazzani, Jack Muramatsu, and Daniel Billsus. 1996. Syskill & webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1* (Portland, Oregon) *(AAAI'96)*. AAAI Press, 54–61.

[28] Aleksandr V. Petrov and Craig Macdonald. 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. In *Proc. GenRec@TheWebConf* (2024-03-07). arXiv:2403.04875

[29] Aleksandr V. Petrov and Craig Macdonald. 2023. Generative Sequential Recommendation with GPTRec. In *Proc. Gen-IR@SIGIR*.

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (Jan. 2020), 67 pages.

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

[32] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In *Proc. EMNLP* (2019-08-27). arXiv:1908.10084 [cs] http://arxiv.org/abs/1908.10084

[33] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[34] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[35] 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.

[36] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[37] Jianling Wang, Kaize Ding, and James Caverlee. 2021. Sequential recommendation for cold-start users with meta transitional learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1783–1787.

[38] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533 [cs.CL] https://arxiv.org/abs/2212.03533

[39] Hongyi Wen, Xinyang Yi, Tiansheng Yao, Jiaxi Tang, Lichan Hong, and Ed H. Chi. 2022. Distributionally-robust Recommendations for Improving Worst-case User Experience. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3606–3610. doi:10.1145/3485447.3512255

[40] Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. 2023. Enhancing conversational search: Large language model-aided informative query rewriting. *arXiv preprint arXiv:2310.09716* (2023).

[41] Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089* (2023).

[42] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian Mcauley, and Dong Wang. 2024. Linear Recurrent Units for Sequential Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (Merida, Mexico) *(WSDM '24)*. Association for Computing Machinery, New York, NY, USA, 930–938. doi:10.1145/3616855.3635760

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

[44] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 4320–4326.

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

[46] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1121–1130. doi:10.1145/3397271.3401178

A Discussion and Limitations

Here, we discuss some limitations of LLM-based recommenders. Compared to ID-based models, larger LLMs such as the 8B LLaMA model used in our experiments take longer to train and require more VRAM during both fine-tuning and generation. We hope that our open-source codebase, which uses quantized LoRA fine-tuning (via `Unsloth`) and integrates paged attention (with `vLLM`), can help the community address these common challenges.

Additionally, large language models are typically trained on vast internet-scale corpora that may include portions of open datasets used for benchmarking. Recent work studying memorization on Movielens-1M [8] shows that LLaMA-3 models can memorize datasets like Movielens, with larger models exhibiting higher memorization rates. For example, the memorization rate reported for LLaMA-3.1 405B is 12.9%, while for LLaMA-3.1 8B it is lower at 5.82% percent. Since we evaluate GLoSS on Amazon datasets and use LLaMA-3 models (1B, 3B, and 8B) that exhibit relatively low memorization rates—at least in the case of Movielens—we consider the memorization risk in our setup to be low.

B Dataset statistics

We conduct experiments on three real-world datasets from the Amazon Product Reviews corpus [11], specifically the "Beauty" (**Beauty**), "Toys and Games" (**Toys**), and "Sports and Outdoors" (**Sports**) categories. These datasets contain user reviews and item metadata collected between May 1996 and July 2014. We use the official 5-core subset [12, 25], which retains users with at least five reviews.[6] User–item interactions are grouped by user and sorted chronologically based on interaction timestamps. For evaluation, we adopt the standard leave-last-item-out split [16, 26, 35]. A summary of dataset statistics is provided in Table 4.

Table 4: Statistics for the three Amazon datasets

Dataset	#Users	#Items	#Interactions	Mean Seq Length
Beauty	22,363	12,094	198,371	8.87
Toys	19,406	11,865	166,757	8.59
Sports	35,597	18,267	295,091	8.29

[6]Available at https://jmcauley.ucsd.edu/data/amazon/index_2014.html. All compared methods also adopt 5-core filtering. For models applying their own filtering (e.g., TIGER [31]), the number of interactions typically differs by few 100s from Table 4, which we believe does not affect metrics.

C Evaluation Across User Segments

C.1 Thresholding by Sequence Length

The threshold choices for segmenting users into cold-start, regular, and power user groups are summarized in Table 5. We set the lower threshold $l = 5$ across all three datasets, while the upper threshold h is set to 14, 13, and 13 for the Beauty, Toys, and Sports datasets, respectively. These thresholds result in approximately 10% power users, 60% regular users, and 30% cold-start users.

Table 5: User Distribution Across Segments Based on Sequence Length Thresholds

Dataset	Sequence Length Segment	User Split Percentage (%)
Beauty	≤ 5 (Cold-Start)	32.11
	(5, 14] (Regular)	58.11
	> 14 (Power)	9.78
Toys	≤ 5 (Cold-Start)	34.52
	(5, 13] (Regular)	56.34
	> 13 (Power)	9.14
Sports	≤ 5 (Cold-Start)	32.60
	(5, 13] (Regular)	59.00
	> 13 (Power)	8.40

C.2 Performance Across Segments

Tables 6 and 7 report performance across user segments for Recall@5 and NDCG@5, respectively. GLoSS achieves consistently high values for both metrics across all 3 segments.

Table 6: Recall@5 for the three user segments. The * symbol indicates the user segement with the highest performance for a given dataset and model.

Dataset	Model	Overall	Short	Med	Long
Beauty	GLoSS-1B	0.0456	0.0439	0.0449	0.0549*
	GLoSS-3B	0.0653	0.0575	0.0626	0.1065*
	GLoSS-8B	0.0681	0.0618	0.0640	0.1129*
Toys	GLoSS-1B	0.0677	0.0755*	0.0640	0.0617
	GLoSS-3B	0.0728	0.0809*	0.0689	0.0660
	GLoSS-8B	0.0796	0.0861*	0.0774	0.0694
Sports	GLoSS-1B	0.0226	0.0275*	0.0211	0.0156
	GLoSS-3B	0.0294	0.0333*	0.0280	0.0250
	GLoSS-8B	0.0364	0.0401*	0.0349	0.0327

D Detailed description of prior ID-based and LLM based recommenders

We compare GLoSS with the following ID-based and LLM based models

Table 7: NDCG@5 for the three user segments. The * symbol indicates the user segment with the highest performance for a given dataset and model.

Dataset	Model	Overall	Short	Med	Long
Beauty	GLoSS-1B	0.0297	0.0272	0.0298	0.0370*
	GLoSS-3B	0.0423	0.0362	0.0406	0.0724*
	GLoSS-8B	0.0442	0.0398	0.0414	0.0750*
Toys	GLoSS-1B	0.0441	0.0515*	0.0407	0.0375
	GLoSS-3B	0.0478	0.0534*	0.0451	0.0430
	GLoSS-8B	0.0529	0.0583*	0.0510	0.0455
Sports	GLoSS-1B	0.0145	0.0176*	0.0135	0.0102
	GLoSS-3B	0.0188	0.0213*	0.0179	0.0163
	GLoSS-8B	0.0238	0.0262*	0.0226	0.0226

D.1 Classic ID-based

- SASRec [16] Represent each item by a unique item ID, it encodes a user's item ID sequence with a self-attentive Transformer decoder, the model is trained using BCE loss.
- BERT4Rec [35] also represents each item by unique item ID, but it encodes user sequences with a bidirectional transform encode (like BERT [35]). The model is trained using a masked prediction objective and the cross entropy loss

D.2 Feature+ID-based

- FDSA [44] Integrates item feature embeddings with vanilla attention layers to obtain feature representations. It then processes item ID and feature sequences seperately through self-attention blocks
- S3Rec [45] uses self supervised pre-training to capture correlation between item feature and item ID, these checkpoints are then loaded and finetuned for next-item prediction using only item IDs.

D.3 Semantic ID-based

- TIGER [31] encodes text features and quantizes them into semantic IDs using RQ-VAE. The model is then trained to predict the next semantic ID and uses beam search during inference.
- ActionPiece [13] incorporates context by building a tokenizer that maps sets of item features to actions, generating tokens based on the co-occurrence patterns of these features.

D.4 LLM based

- P5Rec [9] T5 [30] as a unified backbone to support multiple tasks, including sequential recommendation, rating prediction, and review summarization.
- GPT4Rec [21] leverages GPT-2 to generate multiple queries per user, which are then matched to items using a BM25 lexical retriever
- LlamaRec [41] employs Llama-2 7B as a reranker over candidates retrieved via an LRU-based recommender [42]
- E4SRec [22] operates in two phases. First, it trains a SASRec using only item ID information. Then, it instruction-tunes a LLaMA-2 13B model, where the serialized user history is used as input, and at each position i, the input token consists of a concatenation of the item's text embedding and its corresponding SASRec-derived item embedding.

E Additional Results

E.1 Results with Last item text search (LIS)

Building on the content-based filtering methods discussed in Section 2.1, we conducted a preliminary investigation using **Last Item text Search (LIS)**, a training-free baseline that retrieves recommendations based on the text of the last item a user interacted with.

Figure 2 in Section 3.2 showed LIS with the BM25 [33] retriever; we also evaluated dense encoders (e5-small-v2 and e5-base-v2) to enable semantic similarity-based last item search.

Despite its simplicity, LIS with all three retrievers performs competitively, achieving Recall@5 and NDCG@5 scores comparable to modern sequential recommenders like SASRec and TIGER across the Beauty, Toys, and Sports datasets. Full results are provided in Table 8.

Table 8: Metrics for Last Item Search (LIS), SASRec, and TIGER. Best in bold, second best underlined

Dataset	Model	Recall@5	NDCG@5
Beauty	LIS-BM25	0.0433	0.0232
	LIS-e5-small-v2	0.0418	0.0224
	LIS-e5-base-v2	<u>0.0421</u>	0.0226
	SASRec	0.0387	<u>0.0249</u>
	TIGER	**0.0454**	**0.0321**
Toys	LIS-BM25	0.0521	0.0267
	LIS-e5-small-v2	<u>0.0553</u>	0.0288
	LIS-e5-base-v2	**0.0572**	0.0295
	SASRec	0.0463	<u>0.0306</u>
	TIGER	0.0521	**0.0371**
Sports	LIS-BM25	0.0212	0.0110
	LIS-e5-small-v2	0.0212	0.0112
	LIS-e5-base-v2	0.0216	0.0113
	SASRec	<u>0.0233</u>	<u>0.0154</u>
	TIGER	**0.0264**	**0.0181**

F Generated text samples

Here, we present an example of the 5 generated texts from the finetuned models. The outputs demonstrate that the LLMs produce well-structured candidate texts, effectively capturing the context of the seen items' texts. The target item's text is highlighted in blue.

Sample generated texts with the finetuned Llama-3 models

Target ASIN Text:
Title: Sigma F80 - Flat Kabuki TM
Seen ASINs Texts:
Title: SKINFOOD Salmon Darkcircle Concealer Cream #1 Blooming Light Beige (Whitening Care) 10g
Title: SKIN79 Hot Pink Super Plus Beblesh Balm 40g
Title: Laneige Water Sleeping Pack EX 80ml
Title: Sigma E65 - Small Angle

Generated Texts with Finetuned Llama1b:

Title: SKINFOOD Salmon Darkcircle Concealer Cream #1 Blooming Light Beige (Whitening Care) 10g
Title: SKIN79 Super Plus Beblesh Balm Triple Functions - VIP Gold 40g
Title: Sigma F80 - Flat Kabuki TM
Title: SKINFOOD Salmon Darkcircle Concealer Cream #1 Blooming Light-beige (Whitening Care) 10g
Title: SKINFOOD Salmon Darkcircle Concealer Cream #1 blooming light beige (Whitening Care) 10g

Generated Texts with Finetuned Llama3b:

Title: Sigma E75 - Angled Brow
Title: Etude House Precious Mineral BB Cream Bright Fit SPF30/PA++ #N02 Light Beige
Title: Etude House Precious Mineral BB Cream Bright Fit SPF30/PA++ #W13 Natural Beige
Title: SKIN79 BB Cleanser with Skin Detox Effect
Title: Etude House Precious Mineral BB Cream Bright Fit SPF30/PA++ #W24 Honey Beige

Generated Texts with Finetuned Llama-8b:

Title: SKINFOOD Peach Sake Pore BB Cream #2(SPF20/PA+) 30ml
Title: SKIN79 Super Plus Triple Functions BB Vital Cream 40g
Title: Etude House Wonder Pore Freshner 500ml
Title: Sigma E75 - Angled Brow
Title: Sigma F80 - Flat Kabuki TM