LlamaRec: Two-Stage Recommendation using Large Language Models for Ranking

Zhenrui Yue
zhenrui3@illinois.edu
University of Illinois Urbana-Champaign
Champaign, USA

Sara Rabhi
srabhi@nvidia.com
NVIDIA
Ontario, Canada

Gabriel de Souza Pereira Moreira
gmoreira@nvidia.com
NVIDIA
São Paulo, Brazil

Dong Wang
dwang24@illinois.edu
University of Illinois Urbana-Champaign
Champaign, USA

Even Oldridge
eoldridge@nvidia.com
NVIDIA
British Columbia, Canada

ABSTRACT

Recently, large language models (LLMs) have exhibited significant progress in language understanding and generation. By leveraging textual features, customized LLMs are also applied for recommendation and demonstrate improvements across diverse recommendation scenarios. Yet the majority of existing methods perform training-free recommendation that heavily relies on pretrained knowledge (e.g., movie recommendation). In addition, inference on LLMs is slow due to autoregressive generation, rendering existing methods less effective for real-time recommendation. As such, we propose a two-stage framework using large language models for ranking-based recommendation (LlamaRec). In particular, we use small-scale sequential recommenders to retrieve candidates based on the user interaction history. Then, both history and retrieved items are fed to the LLM in text via a carefully designed prompt template. Instead of generating next-item titles, we adopt a verbalizer-based approach that transforms output logits into probability distributions over the candidate items. Therefore, the proposed LlamaRec can efficiently rank items without generating long text. To validate the effectiveness of the proposed framework, we compare against state-of-the-art baseline methods on benchmark datasets. Our experimental results demonstrate the performance of LlamaRec, which consistently achieves superior performance in both recommendation performance and efficiency.

CCS CONCEPTS

• **Information systems** → **Language models**; **Recommender systems**; **Personalization**.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Large Language Models, Recommender Systems

ACM Reference Format:
Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2018. LlamaRec: Two-Stage Recommendation using Large Language Models for Ranking. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recent advances in large language models (LLMs) have shown significant improvements in various language understanding and generation tasks [4, 21, 29, 30]. By pretraining language models on the next-token prediction task using billions of tokens, LLMs incorporate both extensive knowledge and a wide spectrum of abilities ranging from storytelling to numerical reasoning. For example, the recent Llama 2 model outperforms open-source language models on both human evaluation and benchmark datasets [31]. Motivated by such advances, LLMs are also employed in recommendation tasks like retrieval and ranking, showcasing enhanced performance across multiple scenarios [1, 3, 12, 14, 18, 19, 33, 38].

Nevertheless, the majority of existing works focus on applying LLMs in recommendation and largely ignore the need for efficient inference [3, 17, 34]. In other words, the autoregressive generation of LLMs is often too slow for real-time recommendation, rendering such approaches ineffective for real-world scenarios. To improve recommendation efficiency, one possible solution is to leverage classification or regression heads on top of LLMs to avoid token generation [14]. Yet the heads introduce additional parameters and are trained upon specific settings (e.g., 5-way classification), thereby limiting their applicability for further recommendation tasks. Additionally, many existing studies concentrate on subtasks within recommendation or utilize LLMs for multiple stages, which causes a further reduction in inference speed and thus poses excessive difficulties in achieving efficient recommendation.

In this work, we propose a novel framework for LLM-based two-stage recommendation (LlamaRec), which not only provides a complete solution that includes both retrieval and ranking (also

known as recall and (re)rank), but also outperforms existing methods with enhanced recommendation performance and inference efficiency. In particular, we utilize state-of-the-art sequential recommenders to perform ID-based item retrieval, which can efficiently generate candidates regardless of the user history length. Then, we construct the ranking input using a carefully designed template that transforms user history and candidates to text. The constructed text prompt is used to perform parameter-efficient fine-tuning (PEFT) on a pretrained LLM, where the model ranks candidate items by generating scores over candidate indices. Here, we adopt a verbalizer to transform the LLM head output to a probability distribution without additional parameters. The advantages of ranking with our verbalizer approach include: (1) our LLM achieves significantly reduced inference time by circumventing the autoregressive generation; and (2) we can generate scores for all candidates within one forward pass and avoid the memory-intensive decoding process (e.g., beam search). As such, LlamaRec can train and infer efficiently in our two-stage framework and provide improved ranking performance compared to state-of-the-art baseline methods.

We summarize our contributions below[1]:

(1) We propose a novel framework for LLM-based two-stage recommendation (LlamaRec), which provides a complete solution with both retrieval and ranking.
(2) We use an ID-based sequential recommender as retriever and design a verbalizer approach for LLM-based ranking, which significantly improves the time and memory efficiency for LLM-based recommendation.
(3) We demonstrate the effectiveness of our LlamaRec on benchmark datasets, where the proposed LlamaRec consistently outperforms baseline methods with considerable improvements in sequential recommendation.

2 RELATED WORK

2.1 Large Language Models

Recently, substantial progress has been made in the development of large language models (LLMs), with an illustrious example being the launch of ChatGPT[2], a powerful LLM-based chatbot. Such advancements in LLMs can be roughly attributed to two main factors: (1) scaling up the size of language models; and (2) expanding text corpora in the pretraining stage [2, 4, 24, 25, 29, 35, 39]. Pretrained LLMs leverage self-attention to process input text globally and are optimized via next-token prediction [23, 32]. By incorporating knowledge from the pretraining corpora, LLMs demonstrate advantages in language understanding and generation. Following such paradigm, recent GPT and Llama models [2, 21, 30, 31] show significant improvements on benchmark datasets and human evaluation, and therefore stand as state-of-the-art language models. In this paper, we leverage Llama 2 as our base model and design a efficient tuning and inference framework for two-stage recommendation.

2.2 LLM-based Recommendation

LLMs are applied as recommender systems to understand item text features and improve recommendation performance [8, 16, 18]. The majority of existing LLM-based recommenders are tuning-free and leverage pretrained knowledge to generate next-item recommendation [12, 19, 26, 28, 33, 34]. For example, Chat-REC [7] utilizes ChatGPT to understand user preferences and improve interactive and explainable recommendation. Another stream of LLM-based recommendation focuses on designing tuning strategies upon subtasks (e.g., rating prediction) to further improve performance [3, 14, 17, 22, 38]. For instance, TallRec [1] performs instruction-tuning to decide if an item should be recommended. However, most existing works focus on specific recommendation tasks and adopt autoregressive generation to perform inference, leading to substantially increased waiting time. As such, we aim to provide a LLM-based two-stage recommendation framework in LlamaRec, which outperforms existing methods in both performance and efficiency.

3 METHODOLOGY

3.1 Setup

Based on sequential recommendation, our two-stage recommendation framework takes user interaction history $\boldsymbol{x}$ from dataset $\mathcal{X}$ as input. In particular, $\boldsymbol{x}$ is a sequence of interacted items $[x_1, x_2, \ldots, x_T]$ in chronological order, in which each item is defined in the item space $\mathcal{I}$ ($x_i \in \mathcal{I}, i = 1, 2, \ldots, T$). In the retrieval stage, the items are represented with unique IDs due to large volumes of product data. As for the ranking stage, we leverage item titles to understand user behavior and transition patterns. The output of our framework is the ranking scores $\hat{y} \in \mathbb{R}^{|\mathcal{I}|}$, with the ground truth denoted by $y \in \mathcal{I}$. For optimization, we denote our model with f parameterized by $\boldsymbol{\theta}$ (i.e., $\hat{y} = f(\boldsymbol{\theta}; \boldsymbol{x})$), which comprises an efficient retrieval model $f_{\text{retriever}}$ and a LLM-based ranking model f_{ranker} ($f = f_{\text{ranker}} \circ f_{\text{retriever}}$). Ideally, the highest ranked item in $\hat{y}$ should be the ground truth item y (i.e., $y = \arg\max \hat{y}$). Therefore, the objective of our framework is to maximize ground truth item score. That is, we seek to minimize the expectation of negative log likelihood loss $\mathcal{L}$ w.r.t. parameters $\boldsymbol{\theta}$ over $\mathcal{X}$:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{X}}[\mathcal{L}(f(\boldsymbol{\theta}; \boldsymbol{x}), y)]. \tag{1}$$

3.2 The Proposed LlamaRec

For large-scale recommendation datasets, the size of the item scope $|\mathcal{I}|$ can often be of millions. As such, a common solution is to leverage the two-stage framework, where a fast retriever efficiently generates potential candidates and a more powerful model performs ranking upon the retrieved candidates [5, 11]. Inspired by the two-stage framework, we propose LlamaRec with efficient retrieval and improved ranking via LLM. We illustrate our LlamaRec in Figure 1 and describe the details of our method in the following.

3.2.1 Retrieval. Since LlamaRec is designed with a two-stage framework, it is possible to select arbitrary model for the retrieval stage. In this work, we adopt the linear recurrence-based LRURec as our retrieval model $f_{\text{retriever}}$ [36]. LRURec is a small-scale sequential recommender that utilizes linear recurrent units to efficiently process input sequences. LRURec is optimized via autoregressive training to capture user transition patterns and generate predicted item features. For inference, LRURec computes dot product between the item embeddings and the predicted features as item scores. In

[1]Our implementation is available at https://github.com/Yueeeeeeee/LlamaRec.
[2]https://chat.openai.com/

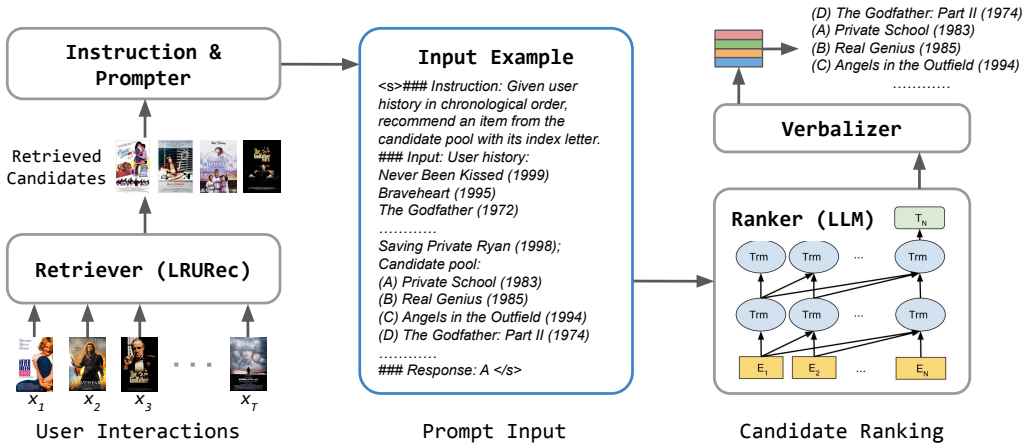

Figure 1: The proposed LlamaRec. The left subfigure illustrates the retrieval stage that generates candidate items with LRURec. Using an instruction template, we transform user history and candidates into text for ranking via Llama 2 (right subfigure).

our LlamaRec, we collect the top-k ($k = 20$ in our experiments) recommendation from LRURec for each input sequence, and the candidate items are saved for the next ranking stage.

3.2.2 LLM Ranker. As mentioned, we select Llama 2 as the base model for f_{ranker} [31]. Specifically, we use the 7B version of Llama 2 to perform ranking among the candidate items from the previous retrieval stage. To construct the text input, we prepend an instruction to describe the task, followed by both history and candidate items represented by their titles. Our prompt template is:

> ### Instruction: Given user history in chronological order, recommend an item from the candidate pool with its index letter.
> ### Input: User history: { `history` }; Candidate pool: { `candidates` }
> ### Response: { `label` }

where `history`, `candidates` and `label` are replaced by history item titles, candidate item titles and the label item of each data example. For inference, the `label` position is left empty for prediction. We provide an example of our input prompt for movie recommendation in the mid subfigure in Figure 1.

Despite using instructions to prompt LLM, the generated output does not directly provide ranking scores for candidates. To solve this problem, existing works prompt LLMs to generate a ranked list of candidate items [3, 12, 17, 19, 34]. However, generating a long list is computationally expensive and often requires further processing, as the titles may not exactly match. Unlike such methods, we propose to leverage a simple verbalizer that efficiently transforms the output from the LLM head (i.e., output scores over all tokens) to ranking scores over candidate items (see Figure 2). Specifically, we adopt index letters to identify candidate items (e.g., (A) Private School (1983) (B) Real Genius (1985) (C) ... etc.) and map the ground truth item to the corresponding index letter. Then, the candidate scores can be computed by retrieving the logits of index letters from the LLM head. In other words, the retrieved scores correspond to the next-token probability distribution within the index letters.

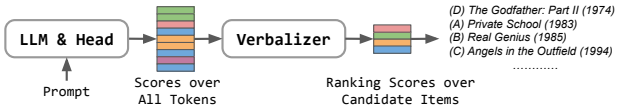

Figure 2: Verbalizer in our LlamaRec ranker.

In training, the ground truth item score is maximized by using index letter of the ground truth item as label. As such, our training paradigm is identical to the next-token-prediction task employed by LLMs. Consequently, LlamaRec can be seamlessly combined with arbitrary causal language modeling task to enable multi-tasking capabilities. As for inference, LlamaRec only requires one single forward pass to obtain the LLM head output, followed by retrieving the logits with the proposed verbalizer and ranking the associated items. By using index letters and a simple verbalizer, the LLM model learns to rank items based on user preference while preserving its generative capabilities. Moreover, the ranking inference only needs one forward pass to obtain all scores over candidate items, and thereby significantly improving the ranking efficiency.

In our LlamaRec implementation, we apply instruction tuning and optimize the model on the response section of the prompt. That is, we only compute loss for `label` tokens (i.e., index letters and EOS token) in the prompt for each data example. This is because optimizing on the entire input does not yield further improvements, while reducing the loss computation to `label` section is slightly more efficient in training. To reduce input length of the LLM, we set the maximum value at 20 for user history items and rank the top-20 candidate items from the retriever model. In our experiments, we adopt QLoRA [6] to perform quantization on model parameters for efficient training with reduced computation and carbon footprint. As such, the trainable parameters in our model is less than 1% of the original 7B size and can be performed on consumer GPUs.

4 EXPERIMENTS

4.0.1 Datasets. Our model is evaluated on the following datasets:

	ML-100k					Beauty					Games				
	NARM	BERT	SAS	LRU	LlamaRec	NARM	BERT	SAS	LRU	LlamaRec	NARM	BERT	SAS	LRU	LlamaRec
M@5	0.0298	0.0188	0.0333	0.0390	**0.0440**	0.0289	0.0246	0.0336	0.0376	**0.0385**	0.0479	0.0422	0.0515	0.0533	**0.0600**
N@5	0.0359	0.0232	0.0409	0.0468	**0.0543**	0.0342	0.0298	0.0397	0.0435	**0.0450**	0.0576	0.0512	0.0617	0.0640	**0.0714**
R@5	0.0544	0.0368	0.0641	0.0705	**0.0852**	0.0503	0.0457	0.0582	0.0614	**0.0648**	0.0874	0.0788	0.0930	0.0966	**0.1061**
M@10	0.0332	0.0244	0.0378	0.0491	**0.0529**	0.0321	0.0276	0.0371	0.0417	**0.0428**	0.0541	0.0478	0.0583	0.0598	**0.0671**
N@10	0.0441	0.0366	0.0522	0.0705	**0.0759**	0.0420	0.0372	0.0481	0.0533	**0.0554**	0.0729	0.0649	0.0783	0.0800	**0.0887**
R@10	0.0801	0.0785	0.0993	0.1426	**0.1524**	0.0746	0.0686	0.0844	0.0916	**0.0971**	0.1351	0.1214	0.1446	0.1463	**0.1599**

Table 1: Main recommendation performance, with the best results marked in bold and second best results underlined.

Datasets	Users	Items	Interact.	Lenth	Density
ML-100k	610	3,650	100k	147.99	4e-2
Beauty	22,332	12,086	198K	8.87	7e-4
Games	15,264	7,676	148K	9.69	1e-3

Table 2: Dataset statistics after preprocessing.

- **ML-100k**: A benchmark dataset for movie recommendation with around 100k user-item interactions [9].
- **Beauty**: A product review dataset from Amazon website consisting of user feedback on Beauty products [10, 20].
- **Games**: A video game dataset from Amazon with user reviews and ratings on video game products [10, 20].

For preprocessing, we follow [3, 37] to construct input sequences in chronological order and iteratively filter users and items that are fewer than 5 interactions (i.e., 5-core). Items without meta data (i.e., title) are also filtered. We report the statistics (i.e., users, items, interactions, sequence length and dataset density) in Table 2.

4.0.2 Baseline Methods. We adopt multiple state-of-the-art sequential recommenders, which include RNN models (i.e., NARM), transformer-based recommenders (i.e., SASRec, BERT4Rec) and linear recurrence-based LRURec. In addition, we adopt LLM-based sequential recommender for comparison in Section 4.0.6:

- **NARM**: NARM is a RNN-based model that leverages local and global encoder for sequential recommendation [15].
- **SASRec**: SASRec adopts unidirectional attention to process input at a sequence-level to generate next items [13].
- **BERT4Rec**: A bidirectional attention-based recommender model, BERT4Rec is trained via predicting masked items [27].
- **LRURec**: An efficient sequential recommender based on linear recurrence, also used as retriever model in LlamaRec [36].

4.0.3 Evaluation. In our evaluation, we follow the leave-one-out strategy and in each data example, we use the last item for testing, the second last item for validation, and the rest items for training. The evaluation metrics are mean reciprocal rank (MRR@k), normalized discounted cumulative gain (NDCG@k) and recall (Recall@k) with $k \in [5, 10]$. We save the model with best validation scores for evaluation (Recall@10 for retrieval and NDCG@10 for ranking), where predictions are ranked against all items in the dataset.

4.0.4 Implementation. For baseline methods and LRURec retriever in LlamaRec, the models are trained with AdamW optimizer using the learning rate of 0.001 and the maximum epoch of 500. Validation is performed every 500 iterations and early stopping is triggered if validation performance does not improve in 20 consecutive rounds. To determine hyperparameters, we perform grid search with weight decay from [0, 1e-2] and dropout rate from [0.1, 0.2, 0.3, 0.4, 0.5]. We used 200 as maximum length for ML-100k and 50 for the other datasets. For our ranker, we use maximum 20 history items and rank the top-20 candidates from the retriever model. The title length is truncated if exceeds 32 tokens. We adopt QLoRA to quantize the Llama 2-based ranker and adopt 8 as LoRA dimension, 32 as α as well as 0.05 dropout. The LoRA learning rate is 1e-4 with target modules being the Q and V projection matrices. The model is tuned for 1 epoch and validated every 100 iterations. Similarly, the model with the best validation performance is saved for test set evaluation.

4.0.5 Main Results. We evaluate recommendation performance of LlamaRec and baseline methods, with the results reported in Table 1. Furthermore, we present the performance within the valid retrieval subset. This valid subset only comprises of predictions for which the ground truth item is within the top-20 retrieved items (by $f_{\text{retriever}}$), as detailed in Table 3. In both tables, each row represents an evaluation metric and each column stands for one recommender method (and dataset). We use BERT, SAS and LRU to abbreviate BERT4Rec, SASRec and LRURec, while M, N and R stand for MRR, NDCG and Recall respectively. For clarity, we mark the best results in bold and underline the second best results. Notice that the ranking model f_{ranker} only improves recommendation performance within the valid retrieval subset. Based on experiment results, LlamaRec achieves superior performance compared to baseline methods. In particular, we observe: (1) LlamaRec perform the best across all metrics on all datasets. Compared the the best performing baseline (i.e., LRURec), LlamaRec achieves 11.99%, 3.99% and 11.06% average improvements on ML-1M, Beauty and Games respectively. (2) LlamaRec achieves the largest performance gains on ML-100k, with 12.82%, 16.02% and 20.85% improvement on MRR@5, NDCG@5 and Recall@5 respectively. The reason may be attributed to the pretrained movie knowledge and extensive user interactions on ML-100k, where long user history (~150) allows for a more comprehensive understanding of user preferences. (3) Within the valid retrieval subset (Table 3), LlamaRec demonstrates larger (absolute) performance gains compared to Table 1. For instance, LlamaRec achieves a further Recall@10 improvement of 0.0643 on Games, compared to only 0.0136 over all user predictions. Overall, the results suggest that LlamaRec can effectively rank candidate items

	ML-100k					Beauty					Games				
	NARM	BERT	SAS	LRU	LlamaRec	NARM	BERT	SAS	LRU	LlamaRec	NARM	BERT	SAS	LRU	LlamaRec
M@5	0.1369	0.0887	0.1449	0.1965	**0.2184**	0.1961	0.1587	0.2296	0.2944	**0.3016**	0.2039	0.1765	0.2177	0.2504	**0.2825**
N@5	0.1607	0.1032	0.1793	0.2356	**0.2693**	0.2284	0.1901	0.2679	0.3403	**0.3524**	0.2424	0.2109	0.2571	0.3009	**0.3360**
R@5	0.2329	0.1477	0.2833	0.3544	**0.4227**	0.3263	0.2861	0.3843	0.4801	**0.5071**	0.3600	0.3160	0.3776	0.4544	**0.4995**
M@10	0.1485	0.1054	0.1596	0.2384	**0.2623**	0.2128	0.1743	0.2491	0.3259	**0.3350**	0.2248	0.1947	0.2408	0.2811	**0.3158**
N@10	0.1886	0.1435	0.2147	0.3367	**0.3766**	0.2689	0.2281	0.3152	0.4168	**0.4337**	0.2931	0.2551	0.3134	0.3760	**0.4173**
R@10	0.3188	0.2720	0.3927	0.6654	**0.7560**	0.4517	0.4043	0.5312	0.7170	**0.7600**	0.5168	0.4530	0.5521	0.6879	**0.7522**

Table 3: Recommendation performance on the valid retrieval subset, in which the ground truth item is among the top 20 retrieved items. The best results are marked in bold and second best results are underlined.

	Beauty					
	P5	PALR	GPT4Rec	RecMind	POD	LlamaRec
N@5	0.0367	N/A	N/A	0.0289	0.0395	**0.0450**
R@5	0.0493	N/A	**0.0653**	0.0415	0.0537	0.0648
N@10	0.0416	0.0446	N/A	0.0375	0.0443	**0.0554**
R@10	0.0645	0.0721	0.0810	0.0574	0.0688	**0.0971**

Table 4: Recommendation performance compared to LLM-based baseline methods.

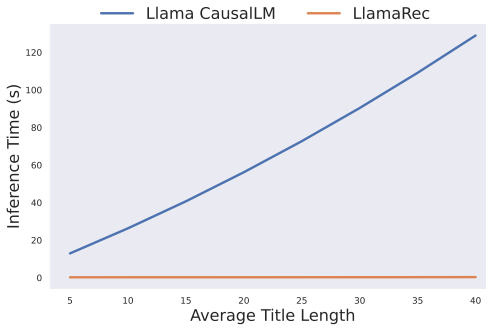

Figure 3: Inference Efficiency of Llama and LlamaRec.

that are of user interest, and thereby improving recommendation performance via language-based preference understanding.

4.0.6 Comparison to LLM-based Baselines. We further compare our LlamaRec framework with LLM-based recommendation methods. Among existing works, we find similar studies on LLM-based sequential recommendation, with the majority of them employing Beauty as a standard benchmark. As the implementation is often not available, we compare our results against the reported results from the original works. The adopted LLM-based methods include: P5, PALR, GPT4Rec, RecMind and POD [3, 8, 17, 18, 34]. The comparison results are reported in Table 4 in a similar fashion to Table 1. Surprisingly, we observe significant improvements using LlamaRec on Beauty, with an average gain of 14.31% compared to the second best results across all metrics. For instance, LlamaRec can achieve 24.22% performance improvement on NDCG@10 against the best-performing baseline PALR. In the case of Recall@5, GPT4Rec exhibits a slight performance advantage of 0.0005 over LlamaRec, yet it's important to note that the metrics of GPT4Rec are computed based on five queries against one forward pass in LlamaRec. In summary, the evaluation results against existing LLM-based methods indicate the effectiveness of our two-stage design and verify the effectiveness of LlamaRec in both retrieval and ranking.

4.0.7 Efficiency of LlamaRec. We now evaluate the ranking efficiency of our verbalizer approach in comparison to the generation approach. For our baseline, we adopt the same Llama 2 model (Llama CausalLM) and vary candidate title lengths in the prompt to generate a ranked list of the candidate titles (as in [3, 12, 17, 19, 34]). In response generation, we perform greedy search for decoding and terminate the generation if response length exceeds the length of all titles combined. In contrast to Llama CausalLM, LlamaRec only need one forward pass to obtain the ranking scores over all candidate items. We present the visualized results of inference time in Figure 3, with x-axis and y-axis representing the average title length and time (in s). As expected, we observe significantly improved efficiency using LlamaRec. For example, the inference time of LlamaRec is consistently under 1s regardless of title length, whereas for an average title length of 20, the generation approach takes 56.16s inference time. In sum, the efficiency of LlamaRec outperforms the baseline generation method by a large margin, showing its potential in real-world recommendation scenarios.

5 CONCLUSION

In this paper, we propose a novel LLM-based two-stage framework LlamaRec for sequential recommendation. The proposed method comprises of two stages: (1) retrieval stage that adopts sequential recommender to efficiently retrieve candidate items; and (2) ranking stage, where a LLM-based ranking model is adopted to understand user preference for fine-grained recommendation. Our LLM ranker leverages textual features for preference understanding and is specifically designed to accelerate inference via a simple verbalizer. We demonstrate the effectiveness and efficiency of LlamaRec by performing experiments on benchmark datasets, where LlamaRec consistently achieves superior recommendation results over state-of-the-art baselines. Moreover, LlamaRec exhibits significantly improved inference speed compared to existing generation-based recommenders, showing its potential to further enhance user experience in LLM-based recommendation.

REFERENCES

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *arXiv preprint arXiv:2305.00447* (2023).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Zheng Chen. 2023. PALR: Personalization Aware LLMs for Recommendation. *arXiv preprint arXiv:2305.07622* (2023).

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314* (2023).

[7] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524* (2023).

[8] 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.

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

[10] 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.

[11] Karl Higley, Even Oldridge, Ronay Ak, Sara Rabhi, and Gabriel de Souza Pereira Moreira. 2022. Building and Deploying a Multi-Stage Recommender System with Merlin. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 632–635.

[12] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *arXiv preprint arXiv:2305.08845* (2023).

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

[14] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction. *arXiv preprint arXiv:2305.06474* (2023).

[15] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[16] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. *arXiv preprint arXiv:2305.13731* (2023).

[17] 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).

[18] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. *Training* 1 (2023), P1.

[19] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149* (2023).

[20] 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*. 43–52.

[21] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023).

[22] Aleksandr V Petrov and Craig Macdonald. 2023. Generative Sequential Recommendation with GPTRec. *arXiv preprint arXiv:2306.11114* (2023).

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[24] Adam Roberts, Colin Raffel, Katherine Lee, Michael Matena, Noam Shazeer, Peter J Liu, Sharan Narang, Wei Li, and Yanqi Zhou. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. (2019).

[25] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100* (2022).

[26] Damien Sileo, Wout Vossen, and Robbe Raymaekers. 2022. Zero-shot recommendation as language modeling. In *European Conference on Information Retrieval*. Springer, 223–230.

[27] 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.

[28] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv preprint arXiv:2304.09542* (2023).

[29] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).

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

[31] 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).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[33] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. *arXiv preprint arXiv:2304.03153* (2023).

[34] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. 2023. Recmind: Large language model powered agent for recommendation. *arXiv preprint arXiv:2308.14296* (2023).

[35] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652* (2021).

[36] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. 2023. Linear Recurrent Units for Sequential Recommendation. (2023).

[37] Zhenrui Yue, Huimin Zeng, Ziyi Kou, Lanyu Shang, and Dong Wang. 2022. Defending substitution-based profile pollution attacks on sequential recommenders. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 59–70.

[38] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).

[39] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009