PatchRec: Multi-Grained Patching for Efficient LLM-based Sequential Recommendation

Jiayi Liao
ljy0ustc@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Ruobing Xie*
xrbsnowing@163.com
Machine Learning Platform Department, Tencent
Beijing, China

Sihang Li
University of Science and Technology of China
Hefei, China

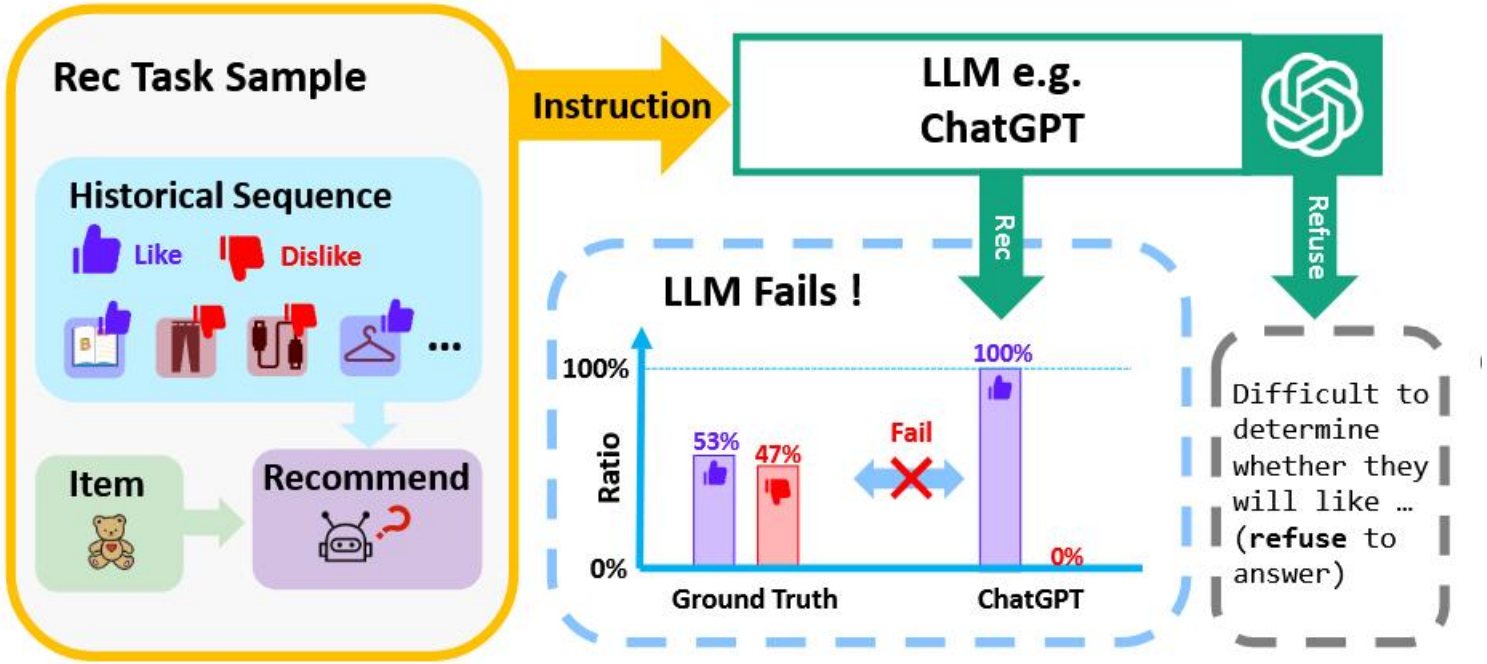
Xiang Wang
University of Science and Technology of China
Hefei, China

Xingwu Sun
Machine Learning Platform Department, Tencent
Beijing, China

Zhanhui Kang
Machine Learning Platform Department, Tencent
Shenzhen, China

Xiangnan He*
xiangnanhe@gmail.com
University of Science and Technology of China
Hefei, China

Large Language Models for sequential recommendation (LLM4SR)



The performance of LLMs in recommendation tasks remains suboptimal due to a **substantial disparity** between the **training tasks** for LLMs and **recommendation tasks**.

Large Language Models for sequential recommendation (LLM4SR)

A common paradigm for LLM4SR:

(1) formatting a user's interaction history (i.e., a sequence of previously interacted items) into an input prompt suitable for LLMs.

(2) fine-tuning the LLMs to generate the target response (i.e., predicting the next item of interest).

Large Language Models for sequential recommendation (LLM4SR)

A common paradigm for LLM4SR:

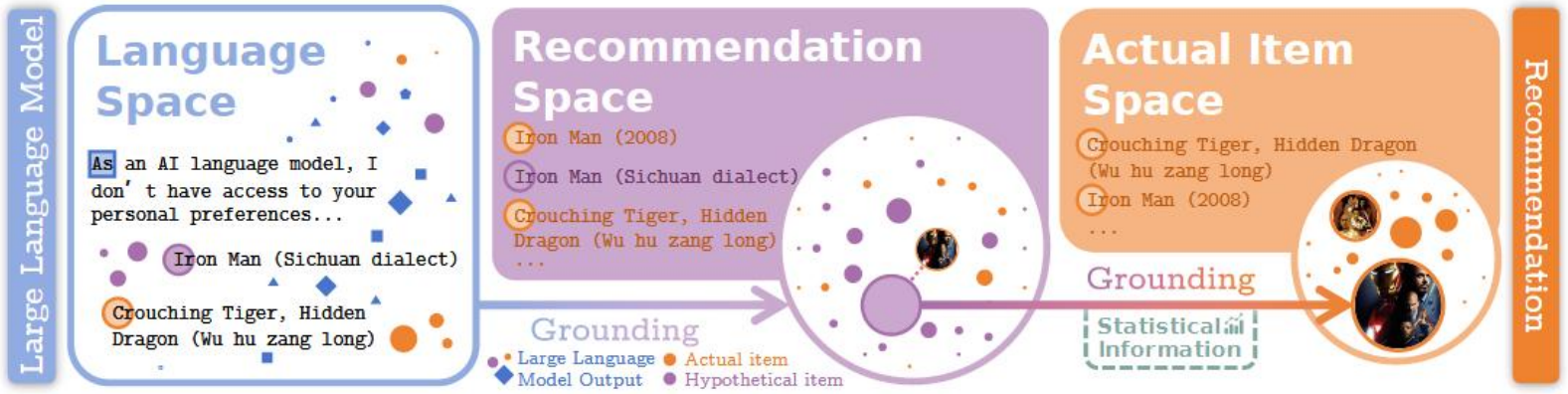


Fig. 1. Illustration of the BIGRec paradigm. During the first step, we ground the language space to recommendation space, which enables the model to generate token sequences of potential items including both actual and hypothetical items. During the second step, we ground the recommendation space to actual item space to provide users with suggestions for real-world items. In the second step, we can easily incorporate statistical information (*e.g.*, popularity and collaborative information) to obtain better recommendations.

[1] A Bi-Step Grounding Paradigm for Large Language Models in Recommendation Systems

Large Language Models for sequential recommendation (LLM4SR)

Challenges of long sequences modeling :

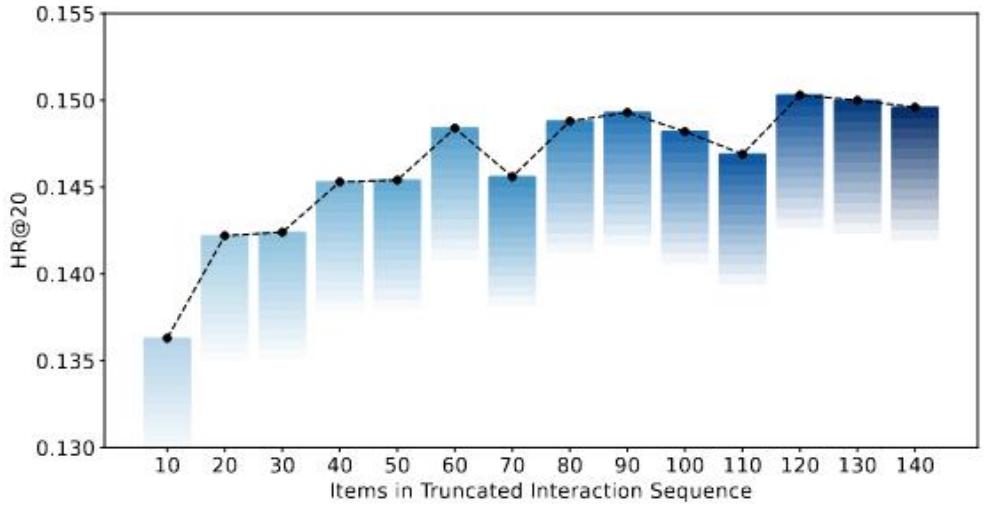
Most leading methods truncate each interaction history, **retaining only the most recent K items.**

This truncation integrates a much shorter interaction sequence into the input prompt, primarily due to the **context window size limitations** and **computational demand** of LLMs.

Large Language Models for sequential recommendation (LLM4SR)

Rich user behavior data has been proven to be of great value for recommender systems[1].

[1]Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction.



(b) Performance of LLM4SR *w.r.t.* truncated sequence length

Conduct preliminary experiments by supervised fine-tuning (SFT) Llama-3.2-1B-Instruct with different truncated sequence length on the Movielens-1M dataset.

PatchRec — a multi-grained (i.e., both item- and session-level) compression framework

1. Item Patch Compression.

$$\mathbf{z}_j^i = \frac{1}{|\mathcal{T}_j|} \sum_{t \in \mathcal{T}_j} \mathbf{x}_t,$$

where $\mathbf{z}_j^i \in \mathbb{R}^d$ denotes the item patch for the j-th item, $\mathcal{T}_j$ is the set of textual tokens associated with this item's title, and $\mathbf{x}_t$ is the textual embedding of token t.

2. Session Patch Compression.

$$\mathbf{z}_j^s = \frac{1}{|\mathcal{S}_j|} \sum_{t \in \mathcal{S}_j} \mathbf{z}_t^i,$$

where $\mathbf{z}_j^s \in \mathbb{R}^d$ denotes the session patch for the j-th session, $\mathcal{S}_j$ is the set of item patches associated with this session, and $\mathbf{z}_t^i$ represents the embedding of the t-th item patch.

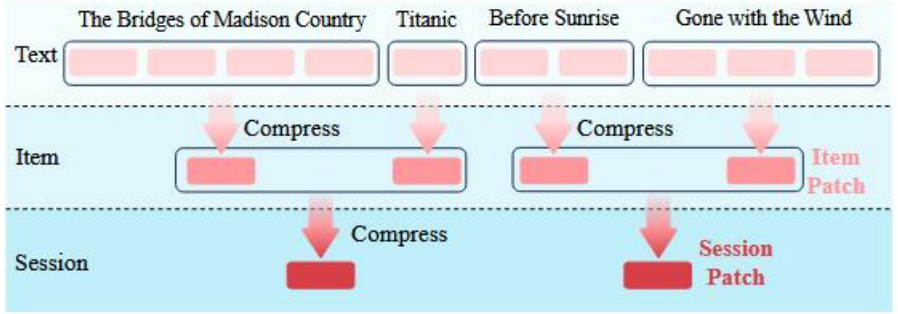


Figure 3: Hierarchical compression. The textual tokens of an item title are aggregated into a compact item patch. Then several adjacent item patches are further compressed into a denser session patch.

PatchRec — Two-Stage Training

1. Pre-training.

For each interaction sequence in a training batch, where items are represented by their textual titles, we augment the interaction sequence with a compressed version of it.

In this compressed interaction sequence, the textual tokens of each item are independently and randomly compressed into an item patch with probability **p**.

The probability **p** gradually increases from 0 to 1 during training.

PatchRec — Two-Stage Training

2. Patch Fine-tuning.

To incorporate the **temporal decay** of user interests in a sequence, we propose two fine-tuning strategies with different compression granularities: **Patch Fine-tuning on Items (PFT-I)** and **Patch Fine-tuning on Sessions (PFT-S).**

PFT-I: The training objective is formulated as:

$$P(Y|X;\Theta) = P(Y|\mathbf{z}_1^i, \mathbf{z}_2^i, \ldots, \mathbf{z}_{K-M}^i, \mathbf{x}_j, \ldots, \mathbf{x}_N; \Theta),$$

PFT-S: divides the truncated historical item sequence into groups of items based on interaction time, with each group containing at most L adjacent items. The training objective is formulated as:

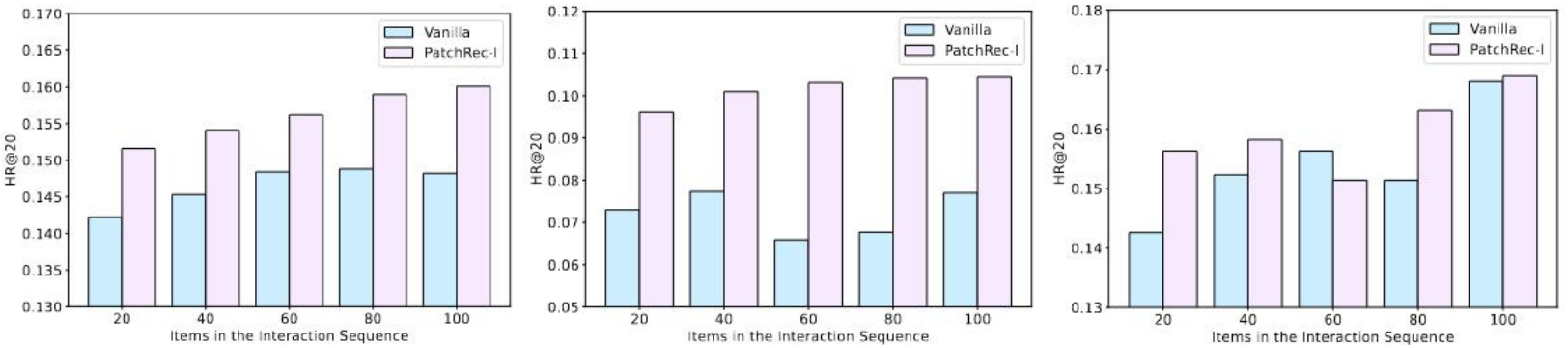
$$P(Y|X;\Theta) = P(Y|\mathbf{z}_1^s, \mathbf{z}_2^s, \ldots, \mathbf{z}_1^i, \mathbf{z}_2^i, \ldots, \mathbf{x}_1, \mathbf{x}_2, \ldots, ; \Theta).$$

Experiment

Table 2: Performance comparison with 100 items in the truncated user sequence. CR is short for compression ratio. Bold and underlined indicate the best and the second-best results, respectively. PatchRec improves recommendation accuracy with less computational demands. *(p-value $\ll$ 0.05)

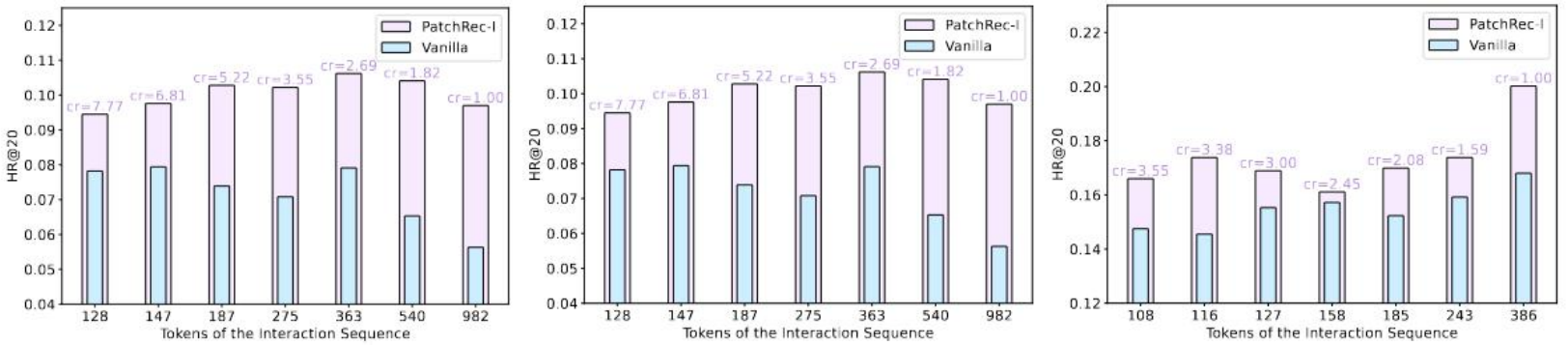
	MovieLens-1M*					Goodreads*					MovieLens-100K				
Model	HR@10	N@10	HR@20	N@20	CR	HR@10	N@10	HR@20	N@20	CR	HR@10	N@10	HR@20	N@20	CR
GRU4Rec	0.0623	0.0298	0.1106	0.0419	-	0.0401	0.0219	0.0592	0.0267	-	0.0661	0.0307	0.1211	0.0445	-
Caser	0.0384	0.0180	0.0712	0.0262	-	0.0116	0.0057	0.0220	0.0083	-	0.0501	0.0234	0.0910	0.0336	-
SASRec	0.0594	0.0286	0.1035	0.0397	-	0.0404	0.0229	0.0569	0.0271	-	0.0605	0.0290	0.1069	0.0406	-
LinRec	0.0622	0.0296	0.1063	0.0407	-	0.0251	0.0125	0.0398	0.0161	-	0.0619	0.0283	0.1048	0.0393	-
Mamba4Rec	0.0648	0.0303	0.1116	0.0421	-	0.0281	0.0146	0.0450	0.0188	-	0.0667	0.0309	0.1086	0.0415	-
MoRec	0.0341	0.0164	0.0616	0.0233	-	0.0122	0.0061	0.0236	0.0089	-	0.0383	0.0184	0.0762	0.0279	-
Vanilla	0.0933	0.0396	0.1482	0.0491	1.00	0.0563	0.0236	0.0770	0.0249	1.00	0.0986	0.0443	0.1680	**0.0611**	1.00
PatchRec-I	**0.1058**	**0.0455**	**0.1616**	**0.0525**	3.44	0.0733	0.0289	0.0976	0.0293	6.81	0.0967	0.0416	**0.1738**	0.0574	3.38
PatchRec-S	0.0967	0.0408	0.1526	0.0496	**9.23**	**0.0748**	**0.0295**	**0.1013**	**0.0301**	**13.62**	**0.1016**	**0.0450**	0.1680	0.0554	**3.85**

Experiment



(a) MovieLens-1M (b) Goodreads (c) MovieLens-100K

Figure 5: Performance comparison between PatchRec-I with vanilla with the same item numbers in interaction sequence. The compression ratios of PatchRec-I are 2.27, 3.06, and 2.25 for MovieLens-1M, Goodreads, and MoviLens-100K, respectively. Impressively, with improved efficiency, PatchRec-I does not compromise recommendation accuracy.



(a) MovieLens-1M (b) Goodreads (c) MovieLens-100K

Figure 6: Performance comparison between the PatchRec-I with vanilla with comparable numbers of tokens in the interaction sequence. The item number in the interaction sequence is 100 for PatchRec-I, with various compression ratios (cr). For PatchRec-I, the latest-M (M=3, 5, 10, 20, 30, 50, 100) items are represented with their textual titles, and earlier items are compressed into item patches. PatchRec-I delivers superior recommendation performance compared to vanilla at the same computational cost.

Thanks