LLM4RSR: Large Language Models as Data Correctors for Robust Sequential Recommendation

Yatong Sun[1,2], **Xiaochun Yang**[1*], **Zhu Sun**[3*], **Yan Wang**[2], **Bin Wang**[1,4,5*], **Xinghua Qu**[6]

[1]School of Computer Science and Engineering, Northeastern University, China
[2]School of Computing, Macquarie University, Australia
[3]Singapore University of Technology and Design, Singapore
[4]National Frontiers Science Center for Industrial Intelligence and Systems Optimization, China
[5]Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, China
[6]Bytedance(Seed), Singapore
yatong@stumail.neu.edu.cn, yangxc@mail.neu.edu.cn, zhu_sun@sutd.edu.sg, yan.wang@mq.edu.au, binwang@mail.neu.edu.cn, quxinghua17@gmail.com

Abstract

Sequential Recommenders (SRs) are trained to predict the next item as the *target* given its preceding items as the *input*, assuming every input-target pair is matched and is reliable for training. However, users can be induced by external distractions to click on items inconsistent with their true preferences, resulting in unreliable training instances with mismatched input-target pairs. To resist unreliable data, researchers attempt to develop Robust SRs (RSRs). However, our data analysis unveils that existing RSRs are data-driven. That is, for most instances formed by infrequently co-occurred items, existing RSRs are *uncertain* about their reliability. To fill this gap, we propose a generic framework – **LLM4RSR** (**L**arge **L**anguage **M**odels for **R**obust **S**equential **R**ecommendation) to semantically complement data-driven RSRs by correcting uncertain instances into reliable ones based on LLMs' semantic comprehension of items beyond co-occurrence. In this way, RSRs can be re-trained with the corrected data for better accuracy. This is a selective knowledge distillation procedure, where the LLM acts as a teacher guiding student RSRs via uncertain instances. To align LLMs with the data correction task and mitigate inherent hallucinations, we equip the LLM with profile, plan, and memory modules, which are automatically optimized via textual gradient descent, eliminating the need for human effort and expertise. Experiments on four real-world datasets spanning eight backbones verify the generality, effectiveness, and efficiency of LLM4RSR.

Code — https://github.com/AlchemistYT/LLM4RSR.

1 Introduction

Sequential Recommenders (SRs) (Wang et al. 2019a; Fang et al. 2020) have recently gained significant attention due to the dynamic and evolving nature of user preferences in real-world scenarios. SRs are trained to predict the next item a user will interact with based on her preceding interactions.

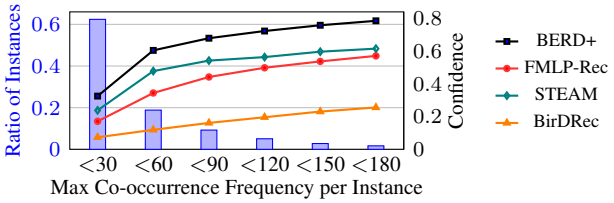

Figure 1: Bars show the ratio of instances with different max co-occurrence frequencies on ML-1M dataset. Lines show the relation between instances' co-occurrence frequency and RSRs' confidence in identifying instances' being reliable. References of the RSRs can be found in experiments.

As such, a training instance for SRs typically consists of an *input* item sequence and its next item as the *target*. However, distractions in daily life (e.g. friends' recommendations) can lead users to interact with items that are inconsistent with their true preferences (Sun et al. 2021). This results in unreliable training instances with either *completely* or *partially* mismatched input-target pairs (Sun et al. 2023a), which depends on whether the item caused by distractions acts as an unreliable target or an unreliable input, respectively.

Such an issue has prompted researchers to develop robust SRs (RSRs) that can resist unreliable instances (Lin et al. 2023; Zhang et al. 2024a). However, existing RSRs face a core limitation as shown in Fig. 1, that is, as data-driven methods, they tend to confidently classify instances involving frequently co-occurred items as reliable while being uncertain about the reliability of most instances formed by infrequently co-occurred items. That is, it is challenging for existing RSRs to differentiate whether the infrequent co-occurrence of two items is due to external influences (e.g. two irrelevant items on sale simultaneously), or data sparsity (e.g. two relevant but unpopular items seldom co-occur). Note that the co-occurrence frequency of an item pair (v_i, v_j) is the number of training instances that contain v_i as

*denotes the corresponding authors.

an input item and v_j as the target. The max co-occurrence frequency of an instance refers to the highest co-occurrence frequency of (v_i, v_j) within the instance. The confidence of an RSR is measured by the max probability of the RSR's prediction distribution over all items (Ma et al. 2023).

We therefore attempt to overcome data-driven RSRs' heavy reliance on co-occurrence by complementing them with LLMs' semantic comprehension and reasoning abilities (Dai et al. 2023; Sun et al. 2024). Ideally, LLMs can uncover the semantic relations between items, reason item matches, and thus complement RSRs by correcting unreliable instances. However, four key challenges are confronted.

C1. **How to efficiently exploit LLMs for data correction?** Due to the prohibitive computational cost, it is impractical to correct an entire dataset with LLMs. Hence, we let LLMs correct instances with higher uncertainty (i.e. lower confidence given by RSRs), as such instances pinpoint the weaknesses of RSRs and thus are more valuable for correction. By doing so, RSRs can be semantically complemented by re-training on the corrected data, and the merits of both data-driven RSRs and semantic-driven LLMs are kept. This is a selective knowledge distillation procedure where the LLM acts as a teacher transferring the crucial semantic knowledge to student RSRs via uncertain instances.

C2. **How to effectively align LLMs with data correction?** It is non-trivial to let LLMs correct uncertain instances without a specific definition of '*causally matched items*' and concrete steps to follow. Thus, we simplify the complex data correction task into an item selection task, i.e., selecting input items relevant to the target item. As such, LLMs can handle partial mismatch by eliminating irrelevant input items, and detect complete mismatch when all the input items are irrelevant to the target. Moreover, to help LLMs understand the item selection task, we equip the LLM with *profile* and *plan* modules. The profile module asks the LLM to act as a domain expert thereby eliciting specific comprehension of '*match*' (relevance), while the plan module breaks down the item selection task into simpler sub-tasks with clear objectives, making it easier for LLMs to process step by step.

C3. **How to obtain optimal profile and plan for various domains?** Designing profiles and plans through manual trial-and-error demands substantial human effort and domain expertise (Zamfirescu-Pereira et al. 2023), as steps and criteria for judging '*match*' may vary for different domains. Inspired by studies on automatic prompt optimization (Pryzant et al. 2023; Yang et al. 2023), we adopt textual gradient descent (TGD) to automatically optimize the profile and plan. Specifically, TGD uses LLMs to self-reflect flaws of the current prompt based on error cases, and then treats such flaws as textual-level gradients to refine the prompt.

C4. **How to mitigate LLMs' hallucinations for data correction?** Although LLMs have the semantic understanding of item content beyond co-occurrence to determine instance reliability, they may make mistakes due to hallucinations (Rawte, Sheth, and Das 2023). Thus, we devise an external *memory* to provide LLMs with factual item descriptions. To enhance the completeness, correctness, and succinctness of the memory, we optimize the memory with TGD. In particular, we use LLMs to self-reflect flaws of the current memory from three aspects: 1) missing key descriptions, 2) containing incorrect descriptions, and 3) containing redundant descriptions. Accordingly, we then ask the LLM to update the memory with three corresponding actions: 1) inserting, 2) modifying, and 3) removing. Thus, the optimized memory can provide better factual descriptions to alleviate LLMs' hallucinations for data correction.

Contributions. (1) We propose a generic LLM4RSR framework to complement existing RSRs. LLM4RSR corrects unreliable instances with higher uncertainty by leveraging LLMs' semantic comprehension of items. As such, data-driven RSRs can be semantically complemented by re-training on the corrected data. This process realizes selective knowledge distillation and avoids the prohibitive computational costs incurred by LLMs at the training and inference phases of RSRs. **(2)** To effectively align LLMs with the data correction task and mitigate inherent hallucinations, we equip LLMs with profile, plan, and memory modules, which are automatically optimized via TGD, circumventing the need for manual effort and domain expertise. **(3)** Experiments with eight backbones on four real-world datasets spanning varying domains and sizes demonstrate the generality, effectiveness, and efficiency of LLM4RSR.

2 Related Works

Robust SRs. Existing SRs adopt various techniques to encode users' interaction sequences, such as Markov Chains (Cheng et al. 2013), recurrent neural networks (Hidasi et al. 2016), convolution neural networks (Tang and Wang 2018), graph neural networks (Xu et al. 2019), and Transformers (Sun et al. 2019). They assume each training instance is a matched input-target pair and thus cannot handle unreliable instances. To combat unreliable data, existing RSRs can be categorized into three types. The *first type* handles complete mismatch by eliminating instances with unreliable targets (Sun et al. 2021, 2023b). The *second type* addresses partial mismatch by reducing the importance of unreliable input items (Chen et al. 2018; Zhou et al. 2022; Zhang et al. 2022; Fan et al. 2022, 2021). The *third type* tackles both complete and partial mismatch by correcting original training instances (Sun et al. 2023a; Lin et al. 2023; Zhang et al. 2024a). Yet, existing RSRs are data-driven and uncertain about the reliability of most instances formed by infrequently co-occurred items (a.k.a. uncertain instances).

LLM-based SRs. Recently, increasing efforts have been dedicated to using LLMs for SRs, which fall into two groups. The first group uses *LLMs as Recommenders*. Specifically, LLMs can be aligned with the recommendation task via training from scratch (Li et al. 2023), finetuning (Bao et al. 2023; Geng et al. 2022; Cui et al. 2022; Zheng et al. 2024), and in-context learning (Dai et al. 2023; Sun et al. 2024; Kim et al. 2024; Wu et al. 2024). The second group uses *LLMs as Feature Encoders* for the downstream SRs (Hou et al. 2022, 2023; Yuan et al. 2023; Xi et al. 2024; Zhang et al. 2024b; Cui et al. 2024). Yet, these methods all overlook the existence of unreliable data.

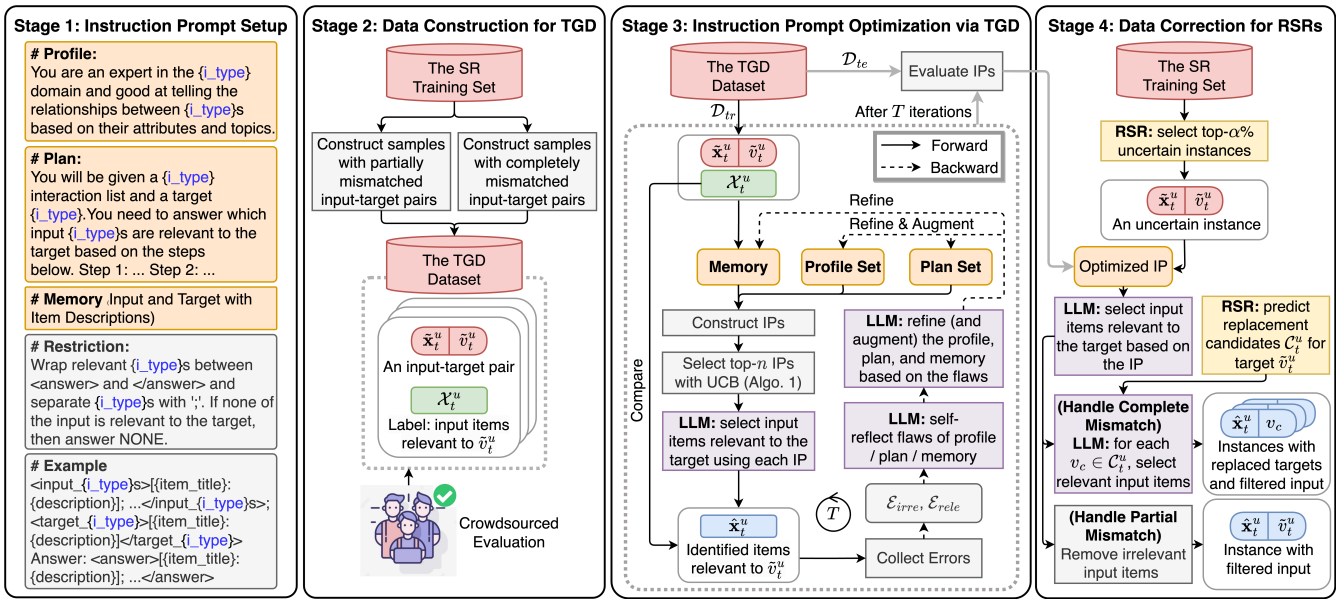

Figure 2: The overall architecture of the proposed LLM4RSR framework.

Prompt 1: Self-reflect Profile Flaws

I'm trying to write a profile for an LLM to help it select relevant input {item_type}s for a target {item_type}. An ideal profile should ask the LLM to act as a domain expert when solving the task. My current profile is {Profile}. But this profile gets the following errors:Some irrelevant item pairs are misclassified as relevant: {$\mathcal{E}_{irre}$}.Some relevant item pairs are misclassified as irrelevant: {$\mathcal{E}_{rele}$}.Summarize the reasons why the profile got these errors.

Prompt 2: Self-reflect Plan Flaws

I'm trying to write a plan for an LLM to help it select relevant input {item_type}s for a target {item_type}. An ideal plan should split the task into simpler subtasks with clear goals. My current plan is {Plan}. But this plan gets the following errors: Some irrelevant item pairs are misclassified as relevant: {$\mathcal{E}_{irre}$}. Some relevant item pairs are misclassified as irrelevant: {$\mathcal{E}_{rele}$}. Summarize the reasons why the plan got these errors.

Prompt 3: Refine the Profile

I'm trying to write a profile for an LLM to help it select relevant input {item_type}s for a target {item_type}. An ideal profile should ask the LLM to act as a domain expert. My current profile is {Profile}. But this profile gets the following errors: Some irrelevant item pairs are misclassified as relevant: {$\mathcal{E}_{irre}$}. Some relevant item pairs are misclassified as irrelevant: {$\mathcal{E}_{rele}$}. Based on these errors, the flaws with this profile is that {Flaws}. Based on the above information, please write a refined profile.

Prompt 4: Refine the Plan

I'm trying to write a plan for an LLM to help it select relevant input {item_type}s for a target {item_type}. An ideal plan should split the task into simpler subtasks with clear goals. My current plan is {Plan}. But this plan gets the following errors: Some irrelevant item pairs are misclassified as relevant: {$\mathcal{E}_{irre}$}. Some relevant item pairs are misclassified as irrelevant: {$\mathcal{E}_{rele}$}. Based on these errors, the flaws with this plan is that {Flaws}. Based on the above information, please write a refined plan.

Prompt 5: Self-reflect Memory Flaws

I'm trying to update the descriptions of two {item_type}s, {title of v_i} and {title of v_j}, to determine their relevance. The current descriptions are [{title of v_i}: {description of v_i}] and [{title of v_j}: {description of v_j}]. These two {item_type}s should be {relevant / irrelevant}, but are misclassified as {irrelevant / relevant}. Please infer the reason why the descriptions lead to the misclassification. The possible reasons can be: missing important descriptions, containing incorrect descriptions, or containing redundant descriptions.

Prompt 6: Refine the Memory

I'm trying to update the descriptions of two {item_type}s, {title of v_i} and {title of v_j}, to determine their relevance. The current summaries are [{title of v_i}: {description of v_i}] and [{title of v_j}: {description of v_j}]. These two {item_type}s should be {relevant / irrelevant}, but are misclassified as {irrelevant / relevant}. These descriptions lead to the misclassification because {Flaws}. Based on these flaws, please write improved descriptions by inserting, modifying, or removing the original descriptions.

Figure 3: The prompts for IP optimization with TGD.

3 The Proposed LLM4RSR Framework

Problem Definition. SRs aim to predict a target item given its preceding items. A training instance of SRs can be represented as an input-target pair $\langle \tilde{\mathbf{x}}_t^u, \tilde{v}_t^u \rangle$, where the target item $\tilde{v}_t^u$ is the t-th item user u interacts with, and the input sequence $\tilde{\mathbf{x}}_t^u = \{u, [\tilde{v}_{t-L}^u, ..., \tilde{v}_{t-2}^u, \tilde{v}_{t-1}^u]\}$ denotes the L preceding items. '$\sim$' denotes the original data that may be unreliable due to external distractions. The goal of LLM4RSR is to complement data-driven RSRs by correcting unreliable instances at the semantic level with LLMs.

Framework Overview. Fig. 2 depicts the overall framework of LLM4RSR, mainly composed of four stages. First, *Instruction Prompt Setup* generates the initial instruction prompt (IP), which directs LLMs to comprehend the data correction task. Then, *Data Construction for TGD* constructs the data to optimize the IP via textual gradient descent (TGD). Subsequently, *Instruction Prompt Optimization with TGD* optimizes the profile, plan, and memory modules of the IP. Lastly, *Data Correction for RSRs* utilizes the optimized IP to correct uncertain instances identified by RSRs. The corrected instances are finally used to re-train RSRs for more robust sequential recommendations.

3.1 Instruction Prompt Setup

This module aims to setup the initial IP, which simplifies the complex data correction task into an item selection task, i.e., selecting input items relevant to the target item. In this way, the LLM can address partial mismatch by removing irrelevant input items, and detect complete mismatch when all the input items are irrelevant to the target.

To align the LLM with the item selection task, each IP is formed by five modules. (i) The *profile* enables the LLM to act as a domain expert, so as to elicit high-quality output tailored to the specific domain. Note that the placeholder 'item_type' will be replaced depending on the domain. (ii) The *plan* decomposes the item selection task into simpler subtasks with specific goals, enabling the LLM to solve the task step by step. (iii) The *memory* stores factual item descriptions to reduce hallucinations by grounding the LLM's

output in relevant domain knowledge. Differs from model-level memory in existing studies on agents (Wang et al. 2023), we design item-level memory. To obtain high-quality memory, each item's description is initialized by the LLM based on item attributes (e.g. a movie's genres and actors) and updated by TGD. (iv) The *restriction* stipulates the format of the LLM output for result extraction. (v) The *example* helps LLM understand the task via in-context learning.

Among these modules, the restriction and example can be manually designed, as the format of the input and output are relatively simple and agnostic to domains. However, designing profile, plan, and memory modules requires much more human effort and domain expertise. This motivates us to automatically optimize these modules via textual gradient descent (TGD). To achieve this, we construct the data for TGD in Section 3.2, and then optimize profile, plan, and memory modules via TGD in Section 3.3.

3.2 Data Construction for TGD

TGD exploits LLMs to self-reflect flaws of current IPs based on errors made by LLMs for the input item selection task. Yet, such errors are inaccessible within existing datasets, as there is no ground truth indicating which input items are relevant to the target. This urges us to construct data to conduct TGD. In particular, each constructed data sample should consist of two parts: (i) an input-target pair $\langle \tilde{\mathbf{x}}_t^u, \tilde{v}_t^u \rangle$ same as SRs' training instances; and (ii) a label set $\mathcal{X}_t^u$ indicating which input items in $\tilde{\mathbf{x}}_t^u$ are relevant to the target item $\tilde{v}_t^u$.

Inspired by Fig. 1, our data construction assumes that the most frequently co-occurred items are more likely to be relevant, whereas the most infrequently co-occurred items tend to be irrelevant. Hence, to construct samples with partially mismatched input-target pairs, we first sample a training instance of SR. Then, each input item v_i that co-occurs with the target v_j more than K times (i.e., $co(v_i, v_j) > K$) is labeled as relevant. The remaining input items are first replaced by items that have never co-occur with the target, and be labeled as irrelevant. To construct samples with completely mismatched input-target pairs, an item pair (v_i, v_j) with zero co-occurrence frequency is sampled. Let v_j be the target item, and v_i be the last item of the input sequence. Next, we find v_i's preceding item $v_{i'}$ which should satisfy two conditions: $co(v_{i'}, v_j) = 0$ and $co(v_{i'}, v_i) > K$. Afterwards, we iteratively find the preceding item of $v_{i'}$ until the length of the input sequence reaches L. In this way, we do our utmost to ensure the input items are sequentially relevant while being irrelevant to the target item. Such data construction continues until it covers all the items.

The quality of the constructed data is examined by crowdsourced evaluation, demonstrating that the labels of the constructed data are highly consistent with human annotations.

3.3 Instruction Prompt Optimization with TGD

This module seeks to automatically optimize the profile, plan, and memory of the initial IP by performing TGD based on the data constructed in Section 3.2. In particular, we randomly select 100 samples for testing (denoted as $\mathcal{D}_{te}$) and the remaining samples (denoted as $\mathcal{D}_{tr}$) are used for the training of IP optimization via TGD.

Algorithm 1: Instruction Prompts Selection

Input: refined IP set $\mathcal{R}$, TGD training set $\mathcal{D}_{tr}$, batch size N, maximum epoch E, exploration parameter c, payoff function $f(\cdot)$.
Output: the expected payoffs $\overline{P}[\mathcal{R}]$

Randomly sample $\mathcal{B}_i \subset \mathcal{D}_{tr}$ where $|\mathcal{B}_i| = N$;
for *each* $r_i \in \mathcal{R}$ **do**
  $P[r_i, 1] = f(r_i, \mathcal{B}_i)$; /* initial payoff */
  $F[r_i] = 1$; /* initial frequency of r_i */
for $e = 1$; $e \leq E$; $e{+}{+}$ **do**
  for *each* $r_i \in \mathcal{R}$ **do**
    $\overline{P}[r_i] = \frac{1}{F[r_i]} \sum_{j=1}^{F[r_i]} (P[r_i, j])$; /* expectation */
    $V[r_i] = \frac{1}{F[r_i]} \sum_{j=1}^{F[r_i]} (P[r_i, j] - \overline{P}[r_i])^2$; /* variance */
  Sample $r_i \leftarrow \arg\max_r \left(\overline{P}[r] + c\sqrt{\frac{\log(e)}{F[r]} V[r]} \right)$;
  Randomly sample $\mathcal{B}_e \subset \mathcal{D}_{tr}$ where $|\mathcal{B}_e| = N$;
  $F[r_i] \leftarrow F[r_i] + 1$; /* update frequency */
  $P[r_i, F[r_i]] \leftarrow P[r_i] + f(r_i, \mathcal{B}_e)$; /* payoff */
return $\overline{P}[\mathcal{R}]$;

Forward Process of TGD. Mirroring numerical gradient descent, the forward process of TGD collects the loss, i.e., errors made by LLMs. Specifically, we first randomly select a batch of training samples from $\mathcal{D}_{tr}$. Then, we feed these samples to an LLM with the initial IP to conduct the item selection task. Afterwards, we evaluate the LLM's output based on the labels and collect the error cases. In particular, we define two sets of error cases, $\mathcal{E}_{irre}$ and $\mathcal{E}_{rele}$, for this batch of samples. If a pair of two irrelevant (relevant) items is misclassified as relevant (irrelevant), then this item pair will be added to $\mathcal{E}_{irre}$ ($\mathcal{E}_{rele}$).

Backward Process of TGD. The collected errors act as the textual loss to compute the gradients and then update IPs. Specifically, we ask the LLM to self-reflect flaws of the current profile, plan, and memory based on the error cases, and such flaws serve as the textual gradients to update these modules. As shown in Fig. 3, we use Prompts 1 and 2 to infer the flaws of the profile and plan, respectively. Prompts 3 and 4 are used to refine the profile and plan, respectively. The optimization for the memory is more complicated. An ideal memory should not only be complete and correct but also be succinct to enhance efficiency. Therefore, for each error case, we ask the LLM to infer flaws in descriptions of involved items from three aspects: missing important descriptions, containing incorrect descriptions, or containing redundant descriptions (Prompt 5). Accordingly, the inferred flaws are used to update item descriptions by inserting, modifying, and removing (Prompt 6). As such, the completeness, correctness, and succinctness of the memory can be enhanced.

Furthermore, we ask the LLM to augment the original and refined profiles and plans with the same semantic meanings, which widens the search space of TGD to help elicit better outputs This is crucial for profiles and plans as they strive to help LLMs better comprehend the task. By contrast, we do not augment the memory as it aims to provide item-level factual descriptions and mitigate hallucinations, which is not

	ML-1M	CD	Game	Kindle
# Users	6,040	15,248	10,606	13,105
# Items	3,417	14,585	16,869	12,966
# Interactions	999,611	392,428	217,663	504,240
# Instances	993,571	377,180	207,057	491,135
Avg. Seq. Length	165.5	25.7	20.5	38.5
Sparsity	95.16%	99.82%	99.88%	99.70%
# PM Samples for TGD	873	2,712	2,696	3,727
# CM Samples for TGD	187	545	439	153

Table 1: Data statistics. 'PM' and 'CM' are partial mismatch and complete mismatch, respectively.

sensitive to semantically similar prompts.

Instruction Prompts Selection. The refined/augmented profiles and plans then form a set of refined IPs, from which we aim to select the top-n IPs to proceed to the next iteration for further optimization. To enhance the efficiency of IP selection, we adopt the Upper Confidence Bound (UCB) algorithm (Auer, CesaBianchi, and Fischer 2002), which balances exploration and exploitation in searching for the best IP. Specifically, Algorithm 1 iteratively selects the best IP based on the expectation and variance of each IP's payoff (measured by F1-score). After E epochs, the top-n IPs with the highest expected payoff will proceed to the next iteration of TGD. Finally, after T iterations of TGD, the optimal IP is selected according to the payoff on the test set $\mathcal{D}_{te}$.

3.4 Data Correction for RSRs

This module exploits the optimized IP to correct the SR training set. Firstly, we select a ratio α of instances with the lowest confidence (i.e. highest uncertainty) based on existing RSRs. The confidence is measured by the max probability of RSRs' prediction distribution over all items (Ma et al. 2023). Such uncertain instances pinpoint the weaknesses of RSRs and thus are more valuable for correction. Next, the LLM selects the relevant input items for each uncertain instance using the optimized IP. The decision of the LLM will be used to handle complete mismatch or partial mismatch. In particular, if the LLM finds that none of the input items is relevant to the target, then this will be treated as a completely mismatched instance. Accordingly, we replace the target with candidate items $\mathcal{C}_t^u$ predicted by RSRs. By treating each candidate $v_c \in \mathcal{C}_t^u$ as the new target, we let the LLM select relevant input items again. By doing so, the updated input $\hat{\mathbf{x}}_t^u$ and the new target v_c form a corrected instance $\langle \hat{\mathbf{x}}_t^u, v_c \rangle$. Otherwise, if the LLM finds that some input items are relevant to the original target, then this will be treated as a partially mismatched instance. Thus, the instance can be corrected into $\langle \hat{\mathbf{x}}_t^u, \tilde{v}_t^u \rangle$ by removing irrelevant input items. Finally, the data-driven RSRs can be semantically complemented by re-training from scratch on the certain and corrected data, thereby alleviating the influence of unreliable instances from previous training. In future works, we will design, e.g., unlearning mechanisms (Xu et al. 2024), to update RSRs on the corrected data more efficiently.

4 Experiments and Results

We conduct extensive experiments to answer five research questions. (**RQ1**) Can LLM4RSR improve existing RSRs? (**RQ2**) Is LLM4RSR more efficient than existing LLM-based SRs? (**RQ3**) How do essential parameters affect LLM4RSR? (**RQ4**) How do different modules (e.g., memory and TGD) affect LLM4RSR? (**RQ5**) How does LLM4RSR optimize the IPs for various domains?

4.1 Experimental Setup

Datasets. `ML-1M` (Harper and Konstan 2015) is a movie recommendation dataset. `CD`, `Game`, and `Kindle` are product review datasets collected from Amazon.com for CD&Vinyl, Video Games, and eBooks, respectively (McAuley and Leskovec 2013). We preprocess all datasets by removing users and items whose interactions are less than five.

Baselines. To verify the efficacy of our LLM4RSR, we implement it with various backbones including both SRs and RSRs. For ***SRs***, `SASRec` (Kang and McAuley 2018) is built on Transformer (Vaswani et al. 2017). `LRURec` (Yue et al. 2024) is based on linear recurrent units. `Recformer` (Li et al. 2023) pre-trains and finetunes an LLM, LongFormer (Beltagy, Peters, and Cohan 2020), with embedding layers tailored for SRs. `PO4ISR` (Sun et al. 2024) uses LLMs via automatic prompt optimization for SRs. For ***RSRs***, `BERD+` (Sun et al. 2023b) handles complete mismatch by calibrating instance loss and uncertainty with item attributes. `FMLP-Rec` (Zhou et al. 2022) handles partial mismatch using Fourier Transform. `STEAM` (Lin et al. 2023), `SSDRec` (Zhang et al. 2024a), and `BirDRec` (Sun et al. 2023a) handle both complete and partial mismatch.

Evaluation Protocol. To evaluate the recommendation accuracy, three widely-used ranking metrics are adopted, namely, HR, NDCG, and MRR following (Sun et al. 2023c). For each user, we preserve the last two interactions for validation and testing, while the rest are used for training. We follow (Sun et al. 2023a; Ma et al. 2020) and evaluate the ranking results over the whole item set for fair comparison (Krichene and Rendle 2020). To evaluate LLM4RSR regarding the item selection task, we use Precision, Recall, and F1-score (Karypis 2001) to measure accuracy.

Implementation Details. LLM4RSR is implemented upon Llama2-7B (Touvron et al. 2023) with $\tau = 0, T = 10, E = 16, L = 5, n = 4, \alpha = 5\%, |\mathcal{C}_u^t| = 3$, and $K = 100$ for ML-1M and $K = 10$ for other datasets. All experiments are conducted on an NVIDIA A100 GPU.

4.2 Experimental Results and Analysis

Accuracy Comparison (*RQ1*). Table 2 presents the recommendation accuracy of backbones trained under two settings: the original Plain setting and our LLM4RSR setting (i.e., the training set is corrected by LLM4RSR). The results show that all baselines are boosted significantly with LLM4RSR compared to Plain, where the relative improvements are respectively 6.66%, 8.93%, 9.17%, and 8.08% on average, across the four datasets. This validates the ***generality*** and ***effectiveness*** of LLM4RSR. Besides, the improvement on SRs (9.59% on average) is generally more significant compared to RSRs (6.81% on average). The possible reason is that RSRs can inherently mitigate the issue of unreliable data, so less benefit would be brought by LLM4RSR on them. The result of PO4ISR is omitted in Table 2, as the

	Datasets	ML-1M					CD				
	Backbones	Setting	HR@5	HR@10	NDCG@5	NDCG@10	MRR	HR@5	HR@10	NDCG@5	NDCG@10	MRR
SRs	SASRec (ICDM-18)	Plain	0.1767	0.2663	0.1169	0.1453	0.1254	0.0743	0.0934	0.0585	0.0654	0.0618
		LLM4RSR	**0.1912**	**0.2773**	**0.1273**	**0.1541**	**0.1369**	**0.0827**	**0.1004**	**0.0653**	**0.0712**	**0.0691**
		Improv.	8.18%‡	4.14%†	8.86%‡	6.03%†	9.18%‡	11.28%‡	7.46%†	11.65%‡	8.86%‡	11.75%‡
	LRURec (WSDM-24)	Plain	0.1850	0.2673	0.1271	0.1536	0.1324	0.0757	0.0968	0.0596	0.0664	0.0620
		LLM4RSR	**0.2008**	**0.2794**	**0.1384**	**0.1620**	**0.1435**	**0.0843**	**0.1031**	**0.0661**	**0.0716**	**0.0688**
		Improv.	8.54%‡	4.51%†	8.92%‡	5.46%†	8.37%‡	11.31%‡	6.55%†	10.98%‡	7.80%†	10.98%‡
	RecFormer (KDD-23)	Plain	0.1908	0.2754	0.1311	0.1583	0.1358	0.0776	0.097	0.0651	0.0714	0.0684
		LLM4RSR	**0.2096**	**0.2901**	**0.1438**	**0.1690**	**0.1506**	**0.0868**	**0.1045**	**0.0728**	**0.0782**	**0.0771**
		Improv.	9.83%‡	5.33%†	9.72%‡	6.73%†	10.87%‡	11.89%‡	7.78%†	11.82%‡	9.54%‡	12.67%‡
RSRs	BERD+ (TOIS-23)	Plain	0.1924	0.2812	0.1266	0.1559	0.1331	0.0785	0.0989	0.0638	0.0702	0.0671
		LLM4RSR	**0.2042**	**0.2902**	**0.1355**	**0.1634**	**0.1430**	**0.0852**	**0.1039**	**0.0697**	**0.0751**	**0.0736**
		Improv.	6.11%†	3.19%†	7.01%†	4.78%†	7.47%†	8.53%‡	5.03%†	9.24%‡	6.98%†	9.65%‡
	FMLP-Rec (TheWebConf-22)	Plain	0.1785	0.2689	0.1204	0.1496	0.1297	0.0753	0.096	0.0608	0.0686	0.0652
		LLM4RSR	**0.1909**	**0.2775**	**0.1292**	**0.1565**	**0.1399**	**0.0822**	**0.1016**	**0.0667**	**0.0734**	**0.0717**
		Improv.	6.95%†	3.18%†	7.32%†	4.63%†	7.88%†	9.15%‡	5.88%†	9.63%‡	6.96%†	9.92%‡
	STEAM (TheWebConf-23)	Plain	0.1195	0.1951	0.0763	0.1005	0.0875	0.076	0.0977	0.0625	0.0692	0.0657
		LLM4RSR	**0.1270**	**0.2010**	**0.0820**	**0.1050**	**0.0941**	**0.0824**	**0.1025**	**0.0677**	**0.0733**	**0.0714**
		Improv.	6.26%†	3.01%†	7.43%†	4.52%†	7.55%†	8.47%‡	4.95%†	8.26%‡	5.88%†	8.73%‡
	BirDRec (NeurIPS-23)	Plain	0.2353	0.3254	0.1634	0.1913	0.1648	0.0906	0.1117	0.0754	0.0820	0.0783
		LLM4RSR	**0.2492**	**0.3354**	**0.1737**	**0.1995**	**0.1762**	**0.0980**	**0.1169**	**0.0820**	**0.0869**	**0.0848**
		Improv.	5.90%†	3.08%†	6.32%†	4.29%†	6.92%†	8.20%‡	4.66%†	8.78%‡	5.93%†	8.32%‡
	SSDRec (ICDE-24)	Plain	0.2190	0.3066	0.1503	0.1784	0.1526	0.0867	0.1074	0.0715	0.0782	0.0744
		LLM4RSR	**0.2314**	**0.3176**	**0.1599**	**0.1854**	**0.1622**	**0.0936**	**0.1120**	**0.0773**	**0.0828**	**0.0804**
		Improv.	5.64%†	3.58%†	6.42%†	3.95%†	6.29%†	7.97%‡	4.31%†	8.15%‡	5.85%†	8.11%‡
	Datasets	**Game**					**Kindle**					
SRs	SASRec (ICDM-18)	Plain	0.0426	0.0588	0.0332	0.0381	0.0366	0.0804	0.113	0.0558	0.0662	0.0591
		LLM4RSR	**0.0476**	**0.0636**	**0.0373**	**0.0419**	**0.0412**	**0.0885**	**0.1202**	**0.0616**	**0.0713**	**0.0653**
		Improv.	11.85%‡	8.15%‡	12.41%‡	9.93%‡	12.57%‡	10.05%‡	6.33%†	10.32%‡	7.75%‡	10.54%‡
	LRURec (WSDM-24)	Plain	0.0443	0.0596	0.0384	0.0441	0.0367	0.0812	0.1142	0.0563	0.0669	0.0597
		LLM4RSR	**0.0494**	**0.0637**	**0.0427**	**0.0478**	**0.0408**	**8.94%**	**12.16%**	**6.22%**	**7.20%**	**6.58%**
		Improv.	11.56%‡	6.93%†	11.24%‡	8.34%‡	11.25%‡	10.13%‡	6.51%†	10.48%‡	7.66%†	10.21%‡
	RecFormer (KDD-23)	Plain	0.0483	0.0642	0.0393	0.0425	0.0411	0.0861	0.1200	0.0605	0.0714	0.064
		LLM4RSR	**0.0542**	**0.0694**	**0.0440**	**0.0468**	**0.0465**	**0.0958**	**0.1290**	**0.0673**	**0.0778**	**0.0718**
		Improv.	12.17%‡	8.03%‡	12.05%‡	10.19%‡	13.15%‡	11.32%‡	7.54%†	11.25%‡	9.03%‡	12.13%‡
RSRs	BERD+ (TOIS-23)	Plain	0.0494	0.0647	0.0428	0.0481	0.0411	0.0927	0.128	0.0652	0.0763	0.0683
		LLM4RSR	**0.0538**	**0.0682**	**0.0469**	**0.0515**	**0.0452**	**0.1001**	**0.1335**	**0.0706**	**0.0809**	**0.0741**
		Improv.	8.86%‡	5.42%†	9.65%‡	7.13%†	9.88%‡	8.01%‡	4.32%†	8.23%‡	6.01%†	8.43%‡
	FMLP-Rec (TheWebConf-22)	Plain	0.0453	0.0606	0.0347	0.0395	0.0376	0.085	0.1191	0.0594	0.0704	0.063
		LLM4RSR	**0.0496**	**0.0643**	**0.0381**	**0.0423**	**0.0414**	**0.0921**	**0.1251**	**0.0644**	**0.0747**	**0.0685**
		Improv.	9.52%‡	6.11%†	9.89%‡	7.15%†	10.23%‡	8.31%‡	5.02%†	8.45%‡	6.11%†	8.72%‡
	STEAM (TheWebConf-23)	Plain	0.0475	0.0614	0.0412	0.0475	0.0398	0.0888	0.1232	0.0618	0.0729	0.065
		LLM4RSR	**0.0512**	**0.0646**	**0.0448**	**0.0499**	**0.0432**	**0.0952**	**0.1292**	**0.0670**	**0.0770**	**0.0701**
		Improv.	7.76%†	5.18%‡	8.75%‡	5.13%†	8.43%‡	7.21%†	4.88%†	8.35%‡	5.69%†	7.81%‡
	BirDRec (NeurIPS-23)	Plain	0.0555	0.0732	0.0437	0.0492	0.0476	0.1093	0.1445	0.0808	0.0918	0.0835
		LLM4RSR	**0.0596**	**0.0770**	**0.0473**	**0.0517**	**0.0517**	**0.1172**	**0.1498**	**0.0871**	**0.0972**	**0.0896**
		Improv.	7.46%†	5.15%†	8.23%‡	5.11%†	8.69%‡	7.20%†	3.66%†	7.78%‡	5.93%†	7.32%†
	SSDRec (ICDE-24)	Plain	0.0512	0.0694	0.0411	0.0470	0.0456	0.0979	0.1352	0.0708	0.0829	0.0744
		LLM4RSR	**0.0547**	**0.0729**	**0.0442**	**0.0494**	**0.0495**	**0.1046**	**0.1396**	**0.0758**	**0.0876**	**0.0799**
		Improv.	6.89%†	5.07%†	7.64%‡	5.18%†	8.48%‡	6.83%†	3.29%†	7.06%†	5.61%†	7.43%†

Table 2: Performance comparison of baseline SRs trained with the original 'Plain' setting and our LLM4RSR framework across the four datasets. *Improv.* means the relative improvement of 'LLM4RSR' setting over the 'Plain' setting. Statistical significance of the improvement is determined by a paired t-test († for p-value≤ 0.01 and ‡ for p-value≤ 0.001).

	Datasets	ML-1M		CD		Game		Kindle	
	Methods	Time	Space	Time	Space	Time	Space	Time	Space
SRs	SASRec	5.9	2.5	43.1	9.7	35.8	8.6	25.2	8.1
	LRURec	5.5	2.3	37.7	8.5	31.1	8.0	22.8	7.7
	RecFormer	59.32	10.6	471.2	55.1	381.4	48.9	191.6	43.7
	PO4ISR	**86976.0**	**56.2**	**911830.4**	**56.2**	**733935.2**	**56.2**	**699807.0**	**56.2**
RSRs	BERD+	6.5	2.9	46.9	10.8	42.7	9.5	30.1	8.6
	FMLP-Rec	5.7	2.4	40.3	8.9	33.4	8.3	23.9	7.9
	STEAM	35.7	2.7	221.3	40.2	204.0	33.1	148.9	25.7
	BirDRec	11.9	3.2	152.3	27.6	131.9	23.6	93.8	20.7
	SSDRec	31.7	2.6	195.5	38.2	183.1	31.1	135.7	23.7

Table 3: Time (s) and space (GiB) cost comparison. The highest value is boldfaced and the runner-up is underlined.

computation of ranking all items using LLMs for all users cannot be finished within a reasonable timeframe.

Complexity Comparison (*RQ2*). Table 3 compares the time and space costs of all methods during inference, which is essential as online services typically require real-time recommendation (Wang et al. 2019b). The results for PO4ISR are estimated based on the number of prompts required to rank all items for every user. We can note that the two LLM-based methods (Recformer and PO4ISR) are dramatically slower and more space-consuming than other approaches. In contrast, LLM4RSR does not introduce any additional computation during inference of existing SRs, which showcases its ***efficiency***. The main time cost of LLM4RSR is caused by IP optimization with TGD (Stage 3) and data correction (Stage 4), which are one-time operations and can be finished offline within several $(4 \sim 6)$ hours on all datasets.

Parameter Analysis (*RQ3*). We analyze the impact of key hyper-parameters on LLM4RSR. α controls the ratio of instances to be corrected. In particular, the training instances of each dataset are ranked in descending order based on their uncertainty and evenly divided into 100 sets, thus $\alpha = x\%$ means the top $x\%$ of instances with the highest uncertainty.

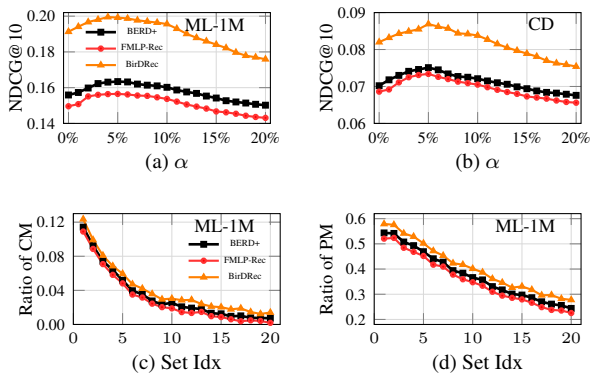

Figure 4: The impact of correction ratio α , and the ratio of completely mismatched (CM) and partially mismatched (PM) instances detected by LLM4RSR regarding different levels of uncertainty.

Datasets	ML-1M			CD		
Variants	Precision	Recall	F1-score	Precision	Recall	F1-score
w.o. profile	0.6359	0.4759	0.5036	0.5123	0.6998	0.5598
static profile	0.6469	0.5325	0.5426	0.5911	0.7363	0.6242
w.o. plan	0.5975	0.5018	0.5029	0.4569	0.7380	0.5314
static plan	0.6573	0.5823	0.5754	0.6824	0.7458	0.6896
w.o. memory	0.5663	0.4090	0.4403	0.3577	0.6668	0.4419
static memory	0.6991	0.5868	0.5912	0.7323	0.7418	0.7234
LLM4RSR	**0.7345**	**0.6091**	**0.6203**	**0.7545**	**0.7651**	**0.7496**
Variants	Game			Kindle		
w.o. profile	0.6661	0.8483	0.7229	0.4769	0.5883	0.5063
static profile	0.6992	0.8608	0.7547	0.4906	0.6003	0.5141
w.o. plan	0.6529	0.8574	0.7186	0.3951	0.5766	0.4621
static plan	0.7208	0.8700	0.7610	0.5206	0.6066	0.5282
w.o. memory	0.5052	0.8316	0.6078	0.3872	0.5646	0.4561
static memory	0.8648	0.8883	0.8626	0.5561	0.6183	0.5499
LLM4RSR	**0.8809**	**0.9050**	**0.8810**	**0.6931**	**0.6233**	**0.6427**

Table 4: Ablation analysis on LLM4RSR. '*w.o.*' means the module is removed, and '*static*' means the module is not optimized by TGD. The best accuracy is boldfaced.

Figs. 4(a)-(b) depict the results, where the best accuracy is gained with $\alpha \approx 5\%$, implying correcting only the top 5% of uncertain instances is optimal for keeping the merits of both LLMs and RSRs. It verifies our claim that highly uncertain instances are more valuable to correct, while correcting too many instances ($\alpha > 15\%$) with lower uncertainty may alter the data distribution and undermine RSRs' accuracy. This can be further confirmed by the results shown in Figs. 4(c)-(d), where the $x-$axis denotes the index of the divided sets, and a smaller index indicates the set contains instances with higher uncertainty. The figure unveils that high-uncertainty data contain a larger proportion of both CM and PM instances compared to low-uncertainty data, suggesting that uncertain instances are more likely to be unreliable and thus more valuable for correction.

Ablation Study (*RQ4*). To examine the efficacy of essential modules, we compare LLM4RSR with its variants by removing each module. Two key findings are noted. (i) The removal of profile, plan, and memory each leads to a decrease in accuracy, with omitting memory causing the most significant drop, indicating their effectiveness, particularly the importance of the factual descriptions provided by the memory for reducing hallucinations. (ii) Updating profile, plan, and memory modules via TGD consistently yields better performance compared to the static manual design, which verifies the efficacy of automatic optimization via TGD.

Initial Profile

You are an expert in the CD domain and good at telling the relationships between CDs based on their key attributes and topics.

Self-reflected Flaws

1. Overly broad scope: The current profile defines the LLM as an expert in the CD domain, which is quite broad. It's important to narrow down the scope to specific aspects of the CD domain that are relevant to the task at hand. For example, you could focus on a particular genre, era, or artist, or even specific themes or emotions conveyed through the music.
2. Lack of specificity: While the profile mentions that the LLM is good at telling relationships between CDs based on their key attributes and topics, it doesn't provide any specific examples or criteria for evaluating these relationships. It's important to define what constitutes a relevant relationship and how the LLM should go about identifying and quantifying it.
3. Overemphasis on subjective qualities: The profile primarily focuses on the LLM's ability to recognize relationships based on subjective qualities like attributes and topics. While these can be useful, it's also important to consider more objective factors like tempo, rhythm, and instrumentation. By incorporating both subjective and objective elements, the profile can help the LLM make more informed and well-rounded recommendations.

Refined Profile

You are an expert in the CD domain, and you are tasked with identifying the most relevant input CDs for a given target CD. To determine relevance, use the following criteria:
* Artist similarity: Weighted score of 2 if the target CD has an artist in common with the input CD, and the artists share a significant creative collaboration or have a notable musical connection.
* Genre similarity: Weighted score of 1 if the target CD belongs to a genre present in the input CD, and the genres share a significant creative connection or have a notable musical overlap.
* Theme similarity: Weighted score of 1 if the target CD has themes present in the input CD, and the themes share a significant creative connection or have a notable musical overlap.
* Release year similarity: Weighted score of 1 if the target CD was released in the same year as the input CD, or if the release years are within 2 years of each other.
Use these scores to calculate a total relevance score for each input CD. A higher total score indicates a more relevant input CD for the target CD.

Figure 5: The optimization of the profile with TGD on CD.

The Optimization of IPs (*RQ5*). Fig. 5 depicts how LLM4RSR optimizes the profile of an IP on CD dataset. In particular, the LLM first lists the key flaws of the initial profile by self-reflection. Accordingly, LLM4RSR then refines the profile.

5 Conclusion

We propose LLM4RSR, a generic framework that leverages LLMs to semantically complement data-driven RSRs for more robust sequential recommendation. It is efficiently achieved by exploiting the LLM to correct RSR's uncertain instances, thereby selectively distilling the semantic knowledge required by RSRs from the LLM. To effectively align the LLM with the data correction task, we equip the instruction prompt with profile, plan, and memory modules, which are automatically optimized through TGD, thereby eliminating the need for domain expertise. Experiments on four real-world datasets across eight backbones showcase the generality, effectiveness, and efficiency of LLM4RSR.

Acknowledgements

The work is partially supported by the National Key Research and Development Program of China (2024YFF0617702), the National Natural Science Foundation of China (Nos. U22A2025, 62072088, 62232007, U23A20309, 61991404), Liaoning Provincial Science and Technology Plan Project - Key R&D Department of Science and Technology (No.2023JH2/101300182), 111 Project (No. B16009), and ARC Discovery Projects DP200101441 and DP230100676. This research is also supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 1, SUTD Kickstarter Initiative (SKI 2021_06_12).

References

Auer, P.; CesaBianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3): 235–256.

Bao, K.; Zhang, J.; Zhang, Y.; Wang, W.; Feng, F.; and He, X. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, 1007–1014.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Chen, X.; Xu, H.; Zhang, Y.; Tang, J.; Cao, Y.; Qin, Z.; and Zha, H. 2018. Sequential Recommendation with User Memory Networks. In *WSDM*, 108–116.

Cheng, C.; Yang, H.; Lyu, M. R.; and King, I. 2013. Where you like to go next: successive point-of-interest recommendation. In *IJCAI*, 2605–2611.

Cui, Y.; Liu, F.; Wang, P.; Wang, B.; Tang, H.; Wan, Y.; Wang, J.; and Chen, J. 2024. Distillation Matters: Empowering Sequential Recommenders to Match the Performance of Large Language Model. In *RecSys*.

Cui, Z.; Ma, J.; Zhou, C.; Zhou, J.; and Yang, H. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*.

Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; and Xu, J. 2023. Uncovering chatgpt's capabilities in recommender systems. In *RecSys*, 1126–1132.

Fan, Z.; Liu, Z.; Wang, S.; Zheng, L.; and Yu, P. S. 2021. Modeling Sequences as Distributions with Uncertainty for Sequential Recommendation. In *CIKM*, 3019–3023.

Fan, Z.; Liu, Z.; Wang, Y.; Wang, A.; Nazari, Z.; Zheng, L.; Peng, H.; and Yu, P. S. 2022. Sequential Recommendation via Stochastic Self-Attention. In *TheWebConf*, 2036–2047.

Fang, H.; Zhang, D.; Shu, Y.; and Guo, G. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM TOIS*, 39(1): 1–42.

Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *RecSys*, 299–315.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *ACM TIIS*, 5(4): 1–19.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

Hou, Y.; He, Z.; McAuley, J.; and Zhao, W. X. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *TheWebConf*, 1162–1171.

Hou, Y.; Mu, S.; Zhao, W. X.; Li, Y.; Ding, B.; and Wen, J.-R. 2022. Towards universal sequence representation learning for recommender systems. In *KDD*, 585–593.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *ICDM*, 197–206.

Karypis, G. 2001. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, 247–254.

Kim, S.; Kang, H.; Choi, S.; Kim, D.; Yang, M.; and Park, C. 2024. Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System. In *KDD*.

Krichene, W.; and Rendle, S. 2020. On Sampled Metrics for Item Recommendation. In *KDD*, 1748–1757.

Li, J.; Wang, M.; Li, J.; Fu, J.; Shen, X.; Shang, J.; and McAuley, J. 2023. Text is all you need: Learning language representations for sequential recommendation. In *KDD*, 1258–1267.

Lin, Y.; Wang, C.; Chen, Z.; Ren, Z.; Xin, X.; Yan, Q.; de Rijke, M.; Cheng, X.; and Ren, P. 2023. A Self-Correcting Sequential Recommender. In *TheWebConf*, 1283–1293.

Ma, C.; Ma, L.; Zhang, Y.; Sun, J.; Liu, X.; and Coates, M. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In *AAAI*, 5045–5052.

Ma, Y.; Cao, Y.; Hong, Y.; and Sun, A. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! *arXiv preprint arXiv:2303.08559*.

McAuley, J.; and Leskovec, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 165–172.

Pryzant, R.; Iter, D.; Li, J.; Lee, Y. T.; Zhu, C.; and Zeng, M. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Rawte, V.; Sheth, A.; and Das, A. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*, 1441–1450.

Sun, Y.; Wang, B.; Sun, Z.; and Yang, X. 2021. Does Every Data Instance Matter? Enhancing Sequential Recommendation by Eliminating Unreliable Data. In *IJCAI*, 1579–1585.

Sun, Y.; Wang, B.; Sun, Z.; Yang, X.; and Wang, Y. 2023a. Theoretically Guaranteed Bidirectional Data Rectification for Robust Sequential Recommendation. In *NeurIPS*, 2850–2876.

Sun, Y.; Yang, X.; Sun, Z.; and Wang, B. 2023b. BERD+: A Generic Sequential Recommendation Framework by Eliminating Unreliable Data with Item-and Attribute-level Signals. *ACM TOIS*, 42(2): 1–33.

Sun, Z.; Fang, H.; Yang, J.; Qu, X.; Liu, H.; Yu, D.; Ong, Y.; and Zhang, J. 2023c. DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation. *IEEE TPAMI*, 45(7): 8206–8226.

Sun, Z.; Liu, H.; Qu, X.; Feng, K.; Wang, Y.; and Ong, Y.-S. 2024. Large Language Models for Intent-Driven Session Recommendations. In *SIGIR*, 324–334.

Tang, J.; and Wang, K. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*, 565–573.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*, 5998–6008.

Wang, L.; Zhang, J.; Chen, X.; Lin, Y.; Song, R.; Zhao, W. X.; and Wen, J.-R. 2023. Recagent: A novel simulation paradigm for recommender systems. *arXiv preprint arXiv:2306.02552*.

Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q. Z.; and Orgun, M. 2019a. Sequential recommender systems: challenges, progress and prospects. In *IJCAI*, 6332–6338.

Wang, X.; He, X.; Cao, Y.; Liu, M.; and Chua, T.-S. 2019b. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*, 950–958.

Wu, J.; Chang, C.; Yu, T.; He, Z.; Wang, J.; Hou, Y.; and McAuley, J. J. 2024. CoRAL: Collaborative Retrieval-Augmented Large Language Models Improve Long-tail Recommendation. In *KDD*.

Xi, Y.; Liu, W.; Lin, J.; Zhu, J.; Chen, B.; Tang, R.; Zhang, W.; Zhang, R.; and Yu, Y. 2024. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. In *RecSys*.

Xu, C.; Zhao, P.; Liu, Y.; Sheng, V. S.; Xu, J.; Zhuang, F.; Fang, J.; and Zhou, X. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*, 3940–3946.

Xu, H.; Zhu, T.; Zhang, L.; Zhou, W.; and Yu, P. S. 2024. Machine Unlearning: A Survey. *ACM Computing Surveys*, 56(1): 9:1–9:36.

Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; and Chen, X. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Yuan, Z.; Yuan, F.; Song, Y.; Li, Y.; Fu, J.; Yang, F.; Pan, Y.; and Ni, Y. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *SIGIR*, 2639–2649.

Yue, Z.; Wang, Y.; He, Z.; Zeng, H.; McAuley, J.; and Wang, D. 2024. Linear recurrent units for sequential recommendation. In *WSDM*, 930–938.

Zamfirescu-Pereira, J.; Wong, R. Y.; Hartmann, B.; and Yang, Q. 2023. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *SIGCHI*, 1–21.

Zhang, C.; Du, Y.; Zhao, X.; Han, Q.; Chen, R.; and Li, L. 2022. Hierarchical Item Inconsistency Signal Learning for Sequence Denoising in Sequential Recommendation. In *CIKM*, 2508–2518.

Zhang, C.; Han, Q.; Chen, R.; Zhao, X.; Tang, P.; and Song, H. 2024a. SSDRec: Self-Augmented Sequence Denoising for Sequential Recommendation. In *ICDE*, 803–815.

Zhang, C.; Sun, Y.; Wu, M.; Chen, J.; Lei, J.; Abdul-Mageed, M.; Jin, R.; Liu, A.; Zhu, J.; Park, S.; Yao, N.; and Long, B. 2024b. EmbSum: Leveraging the Summarization Capabilities of Large Language Models for Content-Based Recommendations. In *RecSys*.

Zheng, Z.; Chao, W.; Qiu, Z.; Zhu, H.; and Xiong, H. 2024. Harnessing Large Language Models for Text-Rich Sequential Recommendation. In *TheWebConf*, 3207–3216.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J.-R. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *TheWebConf*, 2388–2399.