Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models

Yunjia Xi*
xiyunjia@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Weiwen Liu*
liuweiwen8@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Jianghao Lin
chiangel@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Xiaoling Cai
caixiaoling2@huawei.com
Consumer Business Group, Huawei
Shenzhen, China

Zhu Hong
zhuhong8@huawei.com
Consumer Business Group, Huawei
Shenzhen, China

Jieming Zhu
jiemingzhu@ieee.org
Huawei Noah's Ark Lab
Shenzhen, China

Bo Chen
chenbo116@huawei.com
Huawei Noah's Ark Lab
Shanghai, China

Ruiming Tang
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Weinan Zhang†
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yong Yu†
yyu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Recommender system plays a vital role in various online services. However, its insulated nature of training and deploying separately within a specific closed domain limits its access to open-world knowledge. Recently, the emergence of large language models (LLMs) has shown promise in bridging this gap by encoding extensive world knowledge and demonstrating reasoning capabilities. Nevertheless, previous attempts to directly use LLMs as recommenders cannot meet the inference latency demand of industrial recommender systems. In this work, we propose an Open-World Knowledge Augmented Recommendation Framework with Large Language Models, dubbed *KAR*, to acquire two types of external knowledge from LLMs — the *reasoning knowledge* on user preferences and the *factual knowledge* on items. We introduce *factorization prompting* to elicit accurate reasoning on user preferences. The generated reasoning and factual knowledge are effectively transformed and condensed into augmented vectors by a *hybrid-expert adaptor* in order to be compatible with the recommendation task. The obtained vectors can then be directly used to enhance the performance of any recommendation model. We also ensure efficient inference by preprocessing and prestoring the knowledge from the LLM. Extensive experiments show that KAR significantly outperforms the state-of-the-art baselines and is compatible with a wide range of recommendation algorithms. We deploy KAR to Huawei's news and music recommendation platforms and gain a 7% and 1.7% improvement in the online A/B test, respectively.

*Both authors contributed equally to this research.
†The corresponding author.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, Large Language Model, Knowledge Augmentation

ACM Reference Format:
Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Zhu Hong, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3640457.3688104

1 INTRODUCTION

Recommender systems (RSs) are ubiquitous in today's online services, enhancing user experiences in various domains such as movie discovery [36], online shopping [24], and music streaming [65]. However, a common characteristic of existing recommender systems is their *insulated nature* — the models are trained and deployed within closed systems. As depicted in Figure 1(a), the data utilized in a classical recommender system is confined to one or a few specific application domains [36, 84], isolated from the knowledge of the external world, thereby restricting the information that could be

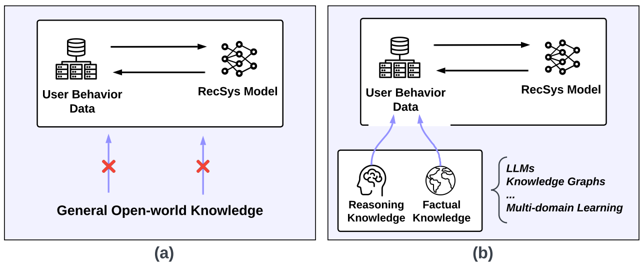


Figure 1: Comparison between (a) closed recommender systems and (b) open-world recommender systems.

learned for a recommendation model. In fact, knowledge beyond the given domains can significantly enhance the predictive accuracy and the generalization ability of recommender systems [16, 42]. Hence, we posit that, instead of solely learning from narrowly defined data in the **closed systems**, recommender systems should be the **open-world systems** that can proactively acquire knowledge from the external world, as shown in Figure 1(b).

In particular, two types of information from the external world are especially useful for recommendation, which we refer to as *open-world knowledge* for recommendation — (1) the *reasoning knowledge* on in-depth user preferences which is inferred from user behaviors and profiles, and (2) the *factual knowledge* on items that can be directly obtained from the web. On the one hand, the reasoning knowledge inferred from the user behavior history enables a more comprehensive understanding of the users, and is critical for better recommendation performance. The user's personality, occupation, intentions, preferences, and tastes could be reflected in their behavior history. On the other hand, the factual knowledge on items provides valuable common sense information about the candidate items, thereby improving the recommendation quality. Take movie recommendation as an example, the external world contains additional movie features such as *plots, related reports, awards, critic reviews* that have not been included in the recommendation dataset, which is beneficial to the recommendation task.

There are many existing studies that aim to enhance closed recommender systems with knowledge graphs [21, 67] or multi-domain learning [22, 62]. However, constructing comprehensive knowledge graphs or multi-domain datasets requires substantial human effort and still has limitations in accessible knowledge. Moreover, they only focus on extracting factual knowledge from the external world and overlook the reasoning knowledge on user preferences [21]. Recent advancements in large language models (LLMs) like GPT-4 [53] and LLaMA [64] show new promise in bridging the gap between traditional recommenders and open-world knowledge, enabling logical reasoning that aligns with our known facts and relationships and problem-solving based on extensive encoded world knowledge [5, 7, 81, 85].

Recently, a lot of studies have attempted to apply LLMs as recommenders [3, 10, 17, 19, 37, 45, 46, 74, 78, 82], but the results of directly using LLMs as recommenders are far from optimal for real-world recommendation scenarios due to the following shortcomings. 1) **Inference latency.** Due to the excessive number of model parameters, it is impractical to directly use LLMs as recommender systems in industrial settings. With billions of users and thousands of user behaviors, LLMs fail to meet the low latency requirement in recommender systems (usually within 100 milliseconds). The large model size also hinders the possibility of employing real-time user feedback to update and refine the model as in classical recommenders. 2) **Compositional gap.** LLMs often suffer from the issue of compositional gap, where LLMs have difficulty in generating correct answers to the compositional problem like recommending items to users, whereas they can correctly answer all its sub-problems [57]. Requiring direct recommendation results from LLMs is currently beyond their capability, it cannot fully exploit the open-world knowledge in LLMs [10, 34].

The aim of this study is to effectively incorporate open-world knowledge from LLMs, while retaining the benefits of traditional recommenders (*e.g.*, low inference latency). However, despite the impressive capabilities of LLMs, extracting and leveraging knowledge from them presents challenges. LLMs encode vast amounts of world knowledge, making it difficult to identify relevant information for recommendations and accurately understand user preferences. Furthermore, the textual nature of LLM-generated knowledge and the large dense vectors of decoded outputs pose compatibility issues with recommender systems. The knowledge from LLMs can sometimes be unreliable or misleading due to the hallucination problem [33]. Hence, it's crucial to ensure that the transformation of knowledge suits the recommendation and mitigates noise.

Therefore, we propose an Open-World Knowledge Augmented Recommendation Framework with Large Language Models, (dubbed *KAR*). Specifically, KAR is a model-agnostic framework with three stages: (1) knowledge reasoning and generation, (2) knowledge adaptation, and (3) knowledge utilization. For knowledge reasoning and generation, to avoid the compositional gap, we propose *factorization prompting* to break down the complex preference reasoning problem into several vital factors to generate the reasoning knowledge on users and the factual knowledge on items. Then, the knowledge adaptation stage transforms the generated knowledge into augmented vectors in the recommendation space. In this stage, we propose *hybrid-expert adaptor* module to reduce dimensionality and ensemble multiple experts for robust knowledge learning, thus increasing the reliability and expressivity of the generated knowledge. Finally, in the knowledge utilization stage, the recommendation model incorporates the augmented vectors for prediction. We also propose to prestore the obtained knowledge to avoid the inference latency issue when incorporating LLMs in RSs. Our main contributions can be summarized as follows:

- We present an open-world recommender system, KAR, which bridges the gap between the recommendation domain knowledge and the open-world knowledge from LLMs. To the best of our knowledge, this is the first practical solution that solves compositional gap with LLMs in recommendations.
- KAR transforms open-world knowledge to dense vectors in recommendation space, compatible with any recommendation models. We also release the code and generated textual knowledge from LLMs[1] to facilitate future research.

[1]Code and knowledge are available at https://github.com/YunjiaXi/Open-World-Knowledge-Augmented-Recommendation and https://github.com/mindspore-lab/models/tree/master/research/huawei-noah/KAR

- The knowledge augmentation can be preprocessed and prestored for fast training and inference, avoiding the large inference latency when adopting LLMs to RSs. Now, KAR has been deployed to Huawei's news and music recommendation platforms and gained a 7% and 1.7% improvement in online A/B test. This is one of the first successful attempts at deploying LLMs to real-world recommender systems.

Extensive experiments on public datasets also show KAR significantly outperforms SOTA models, and is compatible with various recommendation algorithms. We believe that KAR not only sheds light on a way to inject the knowledge from LLMs into the recommendation models, but also provides a practical framework for open-world recommender systems in large-scale applications.

2 RELATED WORK

2.1 LLM as Recommender Itself

The emergence of language models has brought tremendous success in Natural Language Processing (NLP), and it also shows great potential in other domains like recommendations [38, 39, 72, 72, 75]. In this earlier stage, the sizes of the language models for recommendation are relatively small (*e.g.*, under billions of parameters), and finetuning is usually involved for better performance, such as P5 [18] and M6-Rec [9]. With the scaling of the model size and corpus volume, especially with the emergence of ChatGPT [53], LLMs have shown uncanny capability in a wide variety of tasks [15]. Several studies apply LLMs as recommenders and achieve some preliminary results [10, 17, 23, 27, 34, 46, 68, 79]. For instance, Liu *et al.* [46] study whether ChatGPT can serve as a recommender with task-specific prompts and report the zero-shot performance.

However, directly using LLMs as recommenders generally falls behind state-of-the-art recommendation algorithms, implying the importance of domain knowledge and collaborative signals for recommendation tasks [10, 34, 44]. Therefore, there are also methods exploring the incorporation of recommendation collaborative signals into LLMs via parameter-efficient finetuning approaches [3, 23, 43, 45, 47, 66]. For example, TALLRec [3] finetunes LLaMA-7B model [64] with LoRA [28] on recommendation data.

2.2 LLM as Component of Traditional Recommender

There is another category of work leverages LLMs as an auxiliary component in traditional RSs. Here, language models are usually adopted to encode the textual features (*e.g.*, item descriptions, user reviews) or provide extra knowledge for classical recommendations for better user or item representations [12, 25, 26, 59]. For example, UniSRec [26] utilizes BERT [35] to encode user behaviors, and therefore learns universal sequence representations for downstream recommendation. The above methods usually use small language models (*e.g.*, BERT-base with 110M parameters), while another line of work adopts large language models with billion-level parameters, focusing on encoding or prompting external open-world knowledge from LLMs. Some work focuses on extracting knowledge from LLMs [4, 13, 37, 48, 50, 51, 56, 60, 71, 74], either by performing auxiliary tasks such as tag generation [37], heterogeneous knowledge fusion [74], and query rewriting [56], by extracting content from long text in domains like job and news [4, 13], or user profiling [8, 48]. Other work focuses on improving encoding from LLMs, such as S&R Multi-Domain Foundation model [19], NoteLLM [78], and LSVCR [82], but they require integration with additional tasks from notes, videos, and comments, with a relatively narrow scope of applicability. These methods either only use LLMs as encoders to convert original texts into dense vectors without generating additional textual knowledge, or do not make specific designs to incorporate the generated textual information into traditional recommender systems. This may lead to the instability of RS due to the noise or high dimension of LLM embeddings.

3 PRELIMINARIES

In this section, we formulate the recommendation task and introduce the notations. The recommendation task is generally formulated as a binary classification problem over multi-field categorical data. The dataset is denoted as $\mathcal{D} = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where x_i represents the categorical features for the i-th instance and y_i denotes the corresponding binary label (0 for no-click and 1 for click). Usually, x_i contains sparse one-hot vectors from multiple fields, such as item ID and genre. We can denote the feature of the i-th instance as $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,F}]$ with F being the number of field and $x_{i,k}$, $k = 1, \ldots, F$ being the feature of corresponding field.

Recommendation models usually aims to learn a function $f(\cdot)$ with parameters θ that can accurately predict the click probability $P(y_i = 1|x_i)$ for each sample x_i, that is $\hat{y_i} = f(x_i; \theta)$. In practice, industrial recommender systems are confronted with massive users and items. Thus, their recommendation is usually divided into multiple stages, *i.e.*, candidate generation, ranking, and reranking [49], where different models are used to narrow down the relevant items.

However, these classical recommendation models are typically trained on a specific recommendation dataset (*i.e.*, a closed system), overlooking the benefits of accessing open-world knowledge.

4 METHODOLOGY

4.1 Overview

This framework of KAR, as shown in in Figure 2, is model-agnostic and consists of the following three stages:

Knowledge Reasoning and Generation Stage leverages our designed factorization prompting to extract recommendation-relevant knowledge from LLMs. We first decompose the complex reasoning tasks by identifying major factors that determine user preferences and item characteristics. Then, according to each factor, LLMs are required to generate (i) reasoning knowledge on user preferences and (ii) factual knowledge about items. Thus, we can obtain open-world knowledge beyond the original recommendation dataset.

Knowledge Adaptation Stage converts textual open-world knowledge into compact and relevant representations suitable for recommendation. First, the knowledge obtained from LLMs are encoded into dense representations by a knowledge encoder. Next, a hybrid-expert adaptor is designed to transform the representations from the semantic space[2] to the recommendation space. In this way, we obtain the reasoning augmented vector for user preferences and the fact augmented vector for each candidate item.

[2]the embedding space from language models

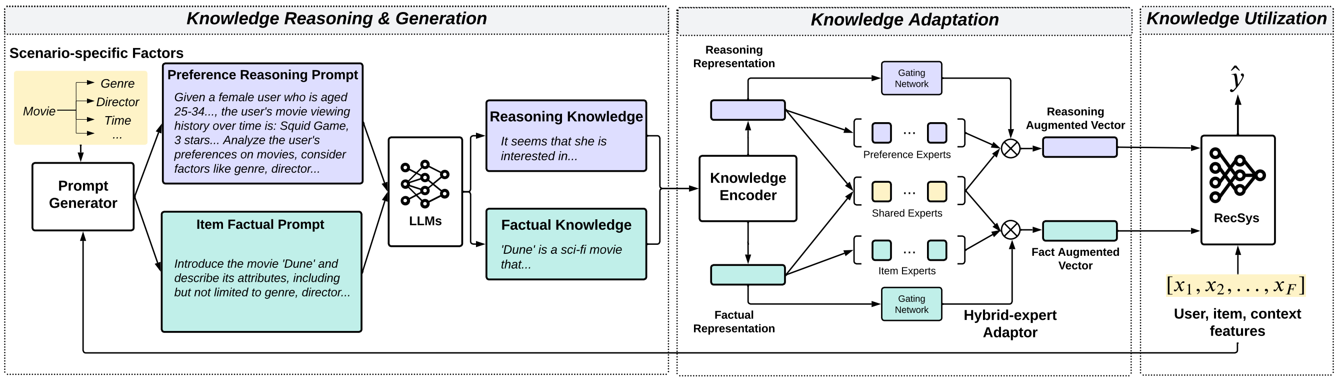

Knowledge Reasoning & Generation
Scenario-specific Factors
Movie
Genre
Director
Time
...
Prompt Generator
Preference Reasoning Prompt
Given a female user who is aged 25-34..., the user's movie viewing history over time is: Squid Game, 3 stars... Analyze the user's preferences on movies, consider factors like genre, director...
Item Factual Prompt
Introduce the movie 'Dune' and describe its attributes, including but not limited to genre, director...
LLMs
Reasoning Knowledge
It seems that she is interested in...
Factual Knowledge
'Dune' is a sci-fi movie that...
Knowledge Adaptation
Reasoning Representation
Knowledge Encoder
Factual Representation
Gating Network
Preference Experts
Shared Experts
Item Experts
Gating Network
Reasoning Augmented Vector
Fact Augmented Vector
Hybrid-expert Adaptor
Knowledge Utilization
$\hat{y}$
RecSys
$[x_1, x_2, \ldots, x_F]$
User, item, context features

Figure 2: The overall framework of KAR.

Knowledge Utilization Stage integrates the reasoning and fact augmented vectors into existing recommendation models, leveraging both domain knowledge and open-world knowledge.

The knowledge generation and encoding are conducted through preprocessing. The hybrid-expert adaptor and recommendation model are jointly trained in an end-to-end manner.

4.2 Knowledge Reasoning and Generation

As the model size scales up, LLMs can encode a vast array of world knowledge and have shown emergent behaviors such as the reasoning ability [29, 58], offering new opportunities to integrate reasoning and factual knowledge into recommendation systems. However, extracting such knowledge from LLMs faces two challenges. Regarding reasoning knowledge, LLMs often struggle with the *compositional gap* issue [57], where they may fail at answering complex questions despite correctly answering their sub-questions. Given that user preferences involve multiple reasoning steps, expecting LLMs to provide precise recommendations in one step might be too ambitious. Regarding factual knowledge, although LLMs possess vast world knowledge, not all of it is relevant for recommendations. General requests to LLMs may yield correct but irrelevant information that does not align with user preferences, limiting recommendation system performance.

Therefore, inspired by the success of Factorization Machines [61] in RSs, we design *factorization prompting* to explicitly "factorize" user preferences into several major factors for effectively extracting the open-world knowledge from LLMs. With the factors incorporated into *preference reasoning prompt*, the complex preference reasoning problem can be broken down into simpler subproblems for each factor, thus alleviating the compositional gap of LLMs. Besides, we also design *item factual prompt*, which utilizes those factors to extract factual knowledge relevant to user preferences. This ensures the generated reasoning knowledge and factual knowledge are aligned for the effective utilization in RSs.

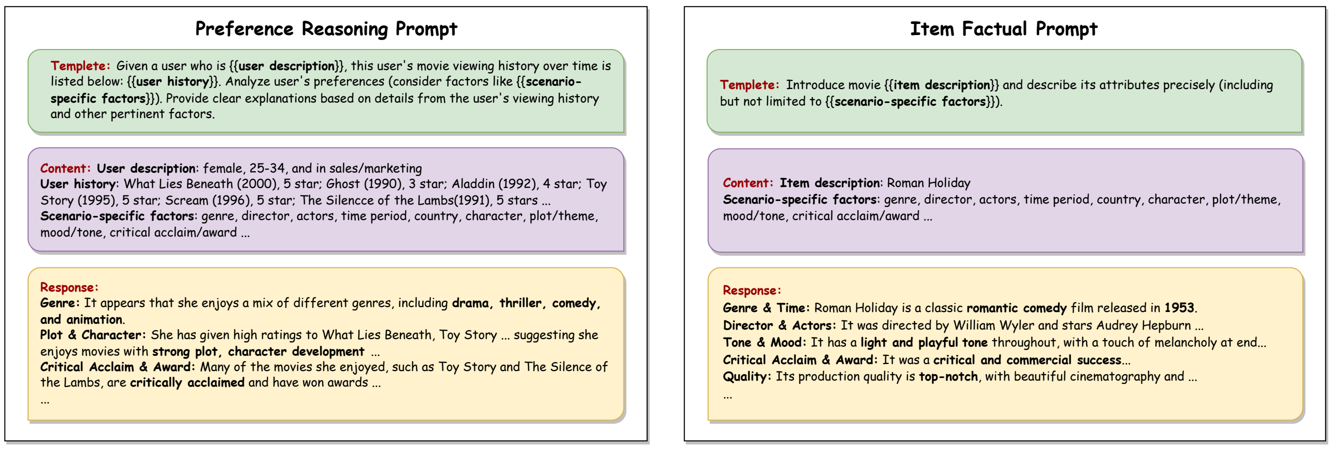
4.2.1 Scenario-specific Factors. The factors determining user preferences may vary for different recommendation scenarios. To determine the factors for different scenarios, we rely on a combination of interactive collaboration with LLMs and expert opinions. For example, in movie recommendation, given a prompt "*List the important factors or features that determine whether a user will be interested in a movie*", LLMs can provide some potential factors. Then, we involve human experts to confirm and refine the outputs to acquire the final scenario-specific factors for movie recommendation — including *genre, actors, directors, theme, mood, production quality, and critical acclaim*. Similarly, in news recommendation, we may obtain factors like *topic, source, region, style, freshness, clarity, and impact*. The collaborative process between LLMs and experts ensures that our chosen factors encompass the critical dimensions of user preference and item characteristics for each scenario.

Note that the specification of these factors is required only once for each scenario and it does not demand much domain expertise with the aid of LLMs. This shows that the proposed method can be easily generalized to different scenarios with little human intervention and manual effort.

4.2.2 LLM as Preference Reasoner & Knowledge Provider. After obtaining the scenario-specific factors, we introduce them into our prompt engineering: preference reasoning prompt and item factual prompt, as illustrated in Figure 3.

Preference reasoning prompt is constructed with the user's profile description, behavior history, and scenario-specific factors. Figure 3 shows an example of the prompt and real response from the LLM, where the user profile description and behavior history provide the LLM with the necessary context and user-specific information to understand the user's preferences. Scenario-specific factors can instruct the LLM to analyze user preference from different facets and allow the LLM to recall the relevant knowledge more effectively and comprehensively. For example, in the factor of genre, the LLM infers user preference for genres such as thriller, comedy, and animation based on user's positive ratings on thriller movies like *What Lies Beneath*, *Scream*, as well as comedy animations like *Toy Story* and *Aladdin*. With the designed prompt, LLM can successfully analyze the user's preferences toward corresponding factors, which is beneficial for recommendations.

Item factual prompt is designed to fill the knowledge gap between the candidate items and the generated reasoning knowledge. Since the dataset in RS may lack relevant knowledge about scenario-specific factors from items, we need to extract corresponding knowledge from LLM to align the generated user and item knowledge. As illustrated in Figure 3, an item prompt consists of two parts – the target item description and the scenario-specific factors. This prompt can guide LLM in compensating for the missing knowledge within the dataset. For instance, in Figure 3, LLM supplements *Roman Holiday* with *"a light and playful tone"*, which is rarely recorded in the datasets. In this way, LLM provides external



Preference Reasoning Prompt

Templete: Given a user who is {{**user description**}}, this user's movie viewing history over time is listed below: {{**user history**}}. Analyze user's preferences (consider factors like {{**scenario-specific factors**}}). Provide clear explanations based on details from the user's viewing history and other pertinent factors.

Content: **User description**: female, 25-34, and in sales/marketing
User history: What Lies Beneath (2000), 5 star; Ghost (1990), 3 star; Aladdin (1992), 4 star; Toy Story (1995), 5 star; Scream (1996), 5 star; The Silencce of the Lambs(1991), 5 stars ...
Scenario-specific factors: genre, director, actors, time period, country, character, plot/theme, mood/tone, critical acclaim/award ...

Response:
Genre: It appears that she enjoys a mix of different genres, including **drama, thriller, comedy, and animation**.
Plot & Character: She has given high ratings to What Lies Beneath, Toy Story ... suggesting she enjoys movies with **strong plot, character development** ...
Critical Acclaim & Award: Many of the movies she enjoyed, such as Toy Story and The Silence of the Lambs, are **critically acclaimed** and have won awards ...
...

Item Factual Prompt

Templete: Introduce movie {{**item description**}} and describe its attributes precisely (including but not limited to {{**scenario-specific factors**}}).

Content: **Item description**: Roman Holiday
Scenario-specific factors: genre, director, actors, time period, country, character, plot/theme, mood/tone, critical acclaim/award ...

Response:
Genre & Time: Roman Holiday is a classic **romantic comedy** film released in **1953**.
Director & Actors: It was directed by William Wyler and stars Audrey Hepburn ...
Tone & Mood: It has a **light and playful tone** throughout, with a touch of melancholy at end...
Critical Acclaim & Award: It was a **critical and commercial success**...
Quality: Its production quality is **top-notch**, with beautiful cinematography and ...
...

Figure 3: Example prompts for KAR. The green, purple, and yellow text bubbles represent the prompt template, the content to be filled in the template, and the response generated by LLMs, respectively (some text has been omitted due to the page limits).

knowledge that aligns with user preferences, allowing for more accurate and personalized recommendations.

4.3 Knowledge Adaptation

The knowledge generated by LLMs presents new challenges in harnessing its potential to assist recommendation models: 1) The knowledge generated by LLMs is usually in the form of text, which cannot be directly leveraged by traditional RSs that typically process categorical features. 2) Even if some LLMs are open-sourced, the decoded outputs are usually large dense vectors (*e.g.*, 4096 for each token) and lie in a semantic space that differs significantly from the recommendation space. 3) The generated knowledge may contain noise or unreliable information [33]. To address these challenges, we have devised two modules: a *knowledge encoder* and a *hybrid-expert adaptor*. The knowledge encoder module encodes the generated textual knowledge into dense vectors. The hybrid-expert adaptor converts dense vectors from the semantic space to the recommendation space. It tackles dimensionality mismatching and allows for noise mitigation. Thus, the knowledge adaptation stage increases the reliability and availability of generated knowledge.

4.3.1 Knowledge Encoder. To harness the potential of textual knowledge generated by LLMs, we employ a knowledge encoder, *e.g.*, BERT [35], to obtain the encodings for each token within the text. Then, we require an aggregation process that combines each token to generate the *preference reasoning representation* $r_i^p \in \mathbb{R}^m$ and the *item factual representation* $r_i^\iota \in \mathbb{R}^m$ of size m as follows

$$\begin{aligned} r_i^p &= \text{Aggr}(\text{Encoder}(klg_i^p))\,, \\ r_i^\iota &= \text{Aggr}(\text{Encoder}(klg_i^\iota))\,, \end{aligned} \tag{1}$$

where klg_i^p and klg_i^ι denote the textual reasoning knowledge and factual knowledge generated by LLMs of the i-th instance in the dataset. Here, various aggregation functions can be employed, such as the representation of the [CLS] token and average pooling. In practice, we primarily adopt average pooling. Note that the knowledge encoder is devised for situations where we only have access to the textual outputs of LLMs. The separate knowledge encoder can be eliminated if the dense vector outputs from LLMs are available.

4.3.2 Hybrid-expert Adaptor. To effectively transform and compact the attained aggregated representations from the semantic space to the recommendation space, we propose a hybrid-expert adaptor module. The aggregated representations capture diverse knowledge from multiple aspects, so we employ a structure that mixes shared and dedicated experts, inspired by the Mixture of Experts (MoE) [31] approach. This allows us to fuse knowledge from different facets and benefit from the inherent robustness offered by multiple experts.

In particular, to fully exploit the shared information of the preference reasoning representation and the item factual representation, we have designed both shared experts and dedicated experts for each kind of representation. The shared experts capture the common aspects, such as shared features, patterns, or concepts, that are relevant to both preference reasoning and item factual knowledge. Reasoning and factual representations also have their dedicated sets of experts to capture the unique characteristics specific to the reasoning or factual knowledge. Mathematically, denote $\mathcal{S}_s, \mathcal{S}_p$, and $\mathcal{S}_\iota$ as the sets of shared experts and dedicated experts for preference reasoning and item factual knowledge with the expert number of n_s, n_p and n_ι. The output is the *reasoning augmented vector* $\hat{r}_i^p \in \mathbb{R}^q$ and the *fact augmented vector* $\hat{r}_i^\iota \in \mathbb{R}^q$ of size q (q is much less than the original dimension m), which are calculated as follows

$$\begin{aligned} \alpha_i^p &= \text{Softmax}(g^p(r_i^p))\,, \quad \alpha_i^\iota = \text{Softmax}(g^\iota(r_i^\iota))\,, \\ \hat{r}_i^p &= \sum\nolimits_{e\in\mathcal{S}_s} \alpha_{i,e}^p \times e(r_i^p) + \sum\nolimits_{e\in\mathcal{S}_p} \alpha_{i,e}^p \times e(r_i^p)\,, \\ \hat{r}_i^\iota &= \sum\nolimits_{e\in\mathcal{S}_s} \alpha_{i,e}^\iota \times e(r_i^\iota) + \sum\nolimits_{e\in\mathcal{S}_\iota} \alpha_{i,e}^\iota \times e(r_i^\iota)\,, \end{aligned} \tag{2}$$

where $g^p(\cdot)$ and $g^\iota(\cdot)$ are the gating networks for preference reasoning and item factual representations, and their outputs α_i^p and α_i^ι are of size $n_s + n_p$ and $n_s + n_\iota$. Here $e(\cdot)$ denotes the expert network, and $\alpha_{i,e}^p$ and $\alpha_{i,e}^\iota$ are the weights of expert $e(\cdot)$ generated by the gating network for preference and item, respectively. Here, each expert network $e(\cdot)$ is designed as a Multi-Layer Perceptron (MLP), facilitating dimensionality reduction and space transformation. Empirical experiments show that about 5 experts are sufficient to generate a satisfying performance[3].

[3]There is no need to set a large number of experts here, and thus, the model does not suffer from convergence issues.

4.4 Knowledge Utilization

Once we have obtained the reasoning augmented vector and the fact augmented vector, we can then incorporate them into backbone recommendation models. This section explores a straightforward approach where these augmented vectors are directly treated as **additional input features**. Specifically, we use them as additional feature fields in recommendation models, allowing them to interact with other features explicitly. During training, the hybrid-expert adaptor module is jointly optimized with the backbone model to ensure the transformation process adapts to the current data distribution. Generally, KAR can be formulated as

$$\hat{y}_i = f(x_i, h_i, \hat{r}_i^p, \hat{r}_i^l; \theta), \tag{3}$$

which is enhanced by the the reasoning augmented vector $\hat{r}_i^p$ and the fact augmented vector $\hat{r}_i^l$. Importantly, KAR only modifies the input of the backbone model and is independent of the design and loss function of the backbone model, so it is flexible and compatible with various backbone model designs. Furthermore, it can be extended to various recommendation tasks, such as sequential recommendation and direct recommendation, by simply adding two augmented vectors in the input.

4.5 Speed-up Approach

Due to the immense scale of the model parameters, the inference of LLMs takes extensive computation time and resources, and the inference time may not meet the latency requirement in real-world recommender systems with large user and item sets.

To address this, we employ an acceleration strategy to **prestore knowledge representations** r_i^p and r_i^l generated by the knowledge encoder or the LLM into a database. As such, we only use the LLM and knowledge encoder once before training backbone models. During the training and inference of the backbone model, relevant representations are retrieved from the database. Besides, the efficiency of offline knowledge generation with LLM can further be enhanced via quantization or hardware acceleration techniques.

If we have stricter requirements for inference time or storage efficiency, we can detach the adaptor from the model after training and further **prestore augmented vectors** *i.e.*, $\hat{r}_i^p$ and $\hat{r}_i^l$, for inference. The dimension of the augmented vectors (*e.g.*, 32) is usually much smaller than that of the knowledge representations (*e.g.*, 4096), which improves the storage efficiency. Additionally, prestoring the augmented vectors reduces the inference time to nearly the same as the original backbone model, and we have provided experimental verification in Section 5.5. In particular, assume the inference time complexity of the backbone model is $O(f(n,m))$, where n is the number of fields and m is the embedding size. The polynomial function $f(n,m)$ varies depending on different backbone models. With KAR, the inference time complexity is $O(f(n+2,m)) = O(f(n,m))$, which is equivalent to the complexity of the original model.

Item features remain relatively stable, making it feasible to prestore item factual knowledge. Conversely, LLMs face challenges in providing real-time reasoning knowledge about evolving user behaviors. However, given the stability of long-term user preferences and the emphasis on recent user behaviors in backbone models, real-time access to behaviors is unnecessary for LLMs. They can infer long-term preferences based on past behaviors, reducing inference overhead and allowing for convenient prestoring of generated knowledge without frequent updates. Backbone models handle short-term preferences with timely updates, maximizing the benefits of both LLMs and recommendation models. Similar to common practice for cold start users or items [80], we use default vectors when encountering new users or items at inference time. Subsequently, we will generate the knowledge for those new users and items offline and add them to the database.

5 EXPERIMENT

To gain more insights into KAR, we tend to address the following research questions (RQs) in this section.

- **RQ1:** What improvements can KAR bring to backbone models on different tasks, such as CTR prediction and reranking?
- **RQ2:** How does KAR perform compared with other PLM-based baseline methods?
- **RQ3:** Does the knowledge from LLM outperform other methods of knowledge, such as knowledge graph?
- **RQ4:**Does KAR gain improvement when deployed online?
- **RQ5:** How do the reasoning knowledge and factual knowledge generated by the LLM contribute to performance improvement?
- **RQ6:** Does the acceleration strategy, preprocessing and prestorage, enhance the inference speed?

By answering these questions, we aim to comprehensively evaluate the performance and versatility of our proposed framework.

5.1 Setup

5.1.1 Dataset. Our experiments are conducted on public datasets, MovieLens-1M[4] and Amazon[5]. **MovieLens-1M** contains 1 million ratings provided by 6000 users for 4000 movies. Following the data processing similar to DIN [84], we convert the ratings into binary labels by labeling ratings of 4 and 5 as positive and the rest as negative. The data is split into training and testing sets based on user IDs, with 90% assigned to the training set and 10% to the testing set. The dataset contains user features like age, gender, occupation, and item features like item ID and category. The inputs to the models are user features, user behavior history (the sequence of viewed movies with their ID, category, and corresponding ratings), and target item features. **Amazon-Book** [52] is the "Books" category of the Amazon Review Dataset. After filtering out the less-interacted users and items, we remain 11, 906 users and 17, 332 items with 1, 406, 582 interactions. The preprocessing is similar to MovieLens-1M, with the difference being the absence of user features. Additionally, ratings of 5 are regarded as positive and the rest as negative.

5.1.2 Backbone Models and Pretrain Baselines. We select two crucial recommendation tasks: **CTR prediction** and **reranking**, to validate the effectiveness of KAR across various tasks. CTR prediction aims to anticipate how likely a user is to click on an item, usually used in the ranking stage of recommendation. We choose 9 representative CTR models as our backbone models, which can be categorized into feature interaction models and user behavior models. The former focuses on modeling feature interactions between different

[4]https://grouplens.org/datasets/movielens/1m/

[5]https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

Table 1: Comparison between KAR and backbone CTR prediction models.

Backbone model	MovieLens-1M						Amazon-Books					
	AUC			LogLoss			AUC			LogLoss		
	base	KAR	Improv.	base	KAR	Improv.	base	KAR	Improv.	base	KAR	Improv.
DCNv2	0.7830	**0.7935***	1.34%	0.5516	**0.5410***	1.92%	0.8269	**0.8350***	0.98%	0.4973	**0.4865***	2.17%
DCNv1	0.7828	**0.7927***	1.28%	0.5528	**0.5411***	2.12%	0.8268	**0.8348***	0.97%	0.4973	**0.4869***	2.11%
DeepFM	0.7824	**0.7919***	1.22%	0.5518	**0.5432***	1.56%	0.8269	**0.8347***	0.94%	0.4969	**0.4873***	1.93%
FiBiNet	0.7820	**0.7936***	1.49%	0.5531	**0.5405***	2.27%	0.8269	**0.8351***	0.99%	0.4973	**0.4870***	2.07%
AutoInt	0.7821	**0.7931***	1.40%	0.5520	**0.5430***	1.62%	0.8262	**0.8357***	1.16%	0.4981	**0.4863***	2.37%
FiGNN	0.7832	**0.7935***	1.32%	0.5510	**0.5437***	1.33%	0.8270	**0.8352***	0.99%	0.4977	**0.4870***	2.14%
xDeepFM	0.7823	**0.7926***	1.32%	0.5520	**0.5420***	1.81%	0.8271	**0.8351***	0.97%	0.4971	**0.4866***	2.10%
DIEN	0.7853	**0.7953***	1.27%	0.5494	**0.5394***	1.83%	0.8307	**0.8391***	1.01%	0.4926	**0.4812***	2.32%
DIN	0.7863	**0.7961***	1.24%	0.5486	**0.5370***	2.10%	0.8304	**0.8418***	1.38%	0.4937	**0.4801***	2.77%

$*$ denotes statistically significant improvement over backbone CTR prediction models (t-test with p-value < 0.05).

feature fields and we adopt widely-used **DeepFM** [20], **xDeepFM** [41], **DCN** [69], **DCNv2** [70], **FiBiNet** [30], **FiGNN** [40], and **AutoInt** [63]. The latter emphasizes on modeling sequential dependencies of user behaviors and we employ **DIN** [84] and **DIEN** [83]. Reranking is to reorders the items from previous ranking stage and derives a list that yields more utility and user satisfaction [49]. On this task, we implement the state-of-the-art models, e.g., **DLCM** [2], **PRM** [55], **SetRank** [54], and **MIR** [73], as backbone models.

As for baselines, we compare KAR with methods that leverage language models to enhance recommendation, such as P5 [18], UniSRec [26], VQRec [25], **TALLRec** [3], and. According to [23], we design **LLM2DIN** which initializes DIN with item embeddings obtained from chatGLM[14][6]. We utilize the publicly available code of these three models and adapt the model to the CTR task with necessary minor modifications. We also align the data and features for all the methods to ensure fair comparisons.

5.1.3 Evaluation Metrics & Implementation Details. On CTR prediction task, we employ widely-used *AUC* (Area under the ROC curve) and *LogLoss* (binary cross-entropy loss) as evaluation metrics following [20, 63, 70, 84]. As for reranking task, several widely used metrics, *NDCG@K* [32] and *MAP@K* [76], are adopted, following previous work [2, 55, 73]. Due to the way metrics are calculated, results for MAP@1 and NDCG@1 are identical, so we omit @1. Since @3 and @5 are too close, we choose $K = \{3, 7\}$. We utilize ChatGPT (gpt-3.5-turbo) as the LLM to generate knowledge. During our experiment, we utilized approximately 30,000 knowledge messages generated by ChatGPT, with an average token length of around 550. And the total cost is about \$150, including some preliminary exploratory experiments. Then, ChatGLM [14] is employed to encode the knowledge, followed by average pooling as the aggregation function in Eq. (1). Each expert in the hybrid-expert adaptor is implemented as an MLP with a hidden layer size of [128, 32]. The number of experts varies slightly across different backbone models, typically with 2-5 shared experts and 2-6 dedicated experts. We keep the embedding size of the backbone model as 32, and the output layer MLP size as [200, 80]. Other parameters, such as batch size and learning rate, are determined through grid search to achieve the best results.

[6]TALLREC is a representative of LLMs as recommender, and knowledge enhancement, like user profiling [8, 48], is not compared as they are similar to ablation in section 5.4.

5.2 Effectiveness Comparison

5.2.1 Improvement over Backbone Models (RQ1). On **CTR prediction** task, we implement our proposed KAR upon 9 representative CTR models. From the table 1, we can have the following observations: (i) Applying KAR significantly improves the performance of CTR models. For example, when using FiBiNet as the backbone model on MovieLens-1M, KAR achieves a 1.49% increase in AUC and a 2.27% decrease in LogLoss, demonstrating the effectiveness of using open-world knowledge from LLMs into RSs. (ii) As a model-agnostic framework, KAR can be applied to various types of baseline models, whether focusing on feature interaction or behavior modeling. With KAR, the selected 9 CTR models on two datasets all achieve an AUC improvement of about 1-1.5%, indicating the universality of the KAR. **(iii)** KAR shows more remarkable improvement in feature interaction models compared to user behavior models. This may be because the knowledge augmented vectors by KAR are utilized more effectively by the feature interaction layer than the user behavior modeling layer. The dedicated feature interaction design may better exploit the information in knowledge vectors.

To investigate KAR's compatibility on other tasks, *e.g.*, **reranking**, we incorporate KAR into the state-of-the-art reranking models. The results on the Amazon-Books dataset are presented in Table 2, from which the following observations can be made: (i) KAR significantly enhances the performance of backbone reranking models. For example, when PRM is employed as the backbone, KAR achieves a remarkable increase of 5.71% and 4.71% in MAP@7 and NDCG@7. (ii) KAR demonstrates more pronounced improvements in methods that do not involve history modeling, such as DLCM, PRM, and SetRank. The user preference knowledge provided by KAR is particularly advantageous for these methods. However, in MIR, which adequately explores the relationship between history and candidates, the enhancement is slightly smaller.

5.2.2 Improvement over Baselines (RQ2). Next, we compare KAR with recent baselines using language models or sequence representation pretraining on CTR task. The results are presented in Table 3, from which we make the following observations: (i) KAR

Table 2: The comparison of KAR and backbone reranking models on Amazon-Books dataset.

Backbone Model	MAP@3			MAP@7			NDCG@3			NDCG@7		
	base	KAR	Improv.	base	KAR	Improv.	base	KAR	Improv.	base	KAR	Improv.
DLCM	0.6365	**0.6654***	4.54%	0.6247	**0.6512***	4.24%	0.5755	**0.6109***	6.15%	0.6891	**0.7142***	3.64%
PRM	0.6488	**0.6877***	6.00%	0.6359	**0.6722***	5.71%	0.5909	**0.6379***	7.95%	0.6983	**0.7312***	4.71%
SetRank	0.6509	**0.6711***	3.10%	0.6384	**0.6538***	2.41%	0.5947	**0.6137***	3.19%	0.7006	**0.7164***	2.26%
MIR	0.7178	**0.7241***	0.88%	0.7011	**0.7078***	0.96%	0.6747	**0.6837***	1.33%	0.7549	**0.7597***	0.64%

$*$ denotes statistically significant improvement over backbone reranking models (t-test with p-value < 0.05).

significantly outperforms models based on pretrained language models. For instance, with DIN as the backbone on Amazon-Books, KAR achieves a 1.27% improvement in LogLoss over the strongest baseline TALLRec. (ii) When integrating PLMs into RSs, the improvements brought by smaller PLMs are relatively modest. For instance, UniSRec, VQ-Rec, and P5, based on PLMs with parameters less than one billion, tend to exhibit poorer performance, even worse than the baseline DIN. (iii) Conversely, leveraging LLMs can lead to more substantial improvements, as seen in TALLRec and KAR. However, achieving effective enhancements with LLM embedding as initialization proves challenging, *e.g.*, there is little difference between the results of LLM2DIN and DIN.

Table 3: Comparison between KAR and baselines.

Model	Backbone PLM	MovieLens-1M		Amazon-Books	
		AUC	LogLoss	AUC	LogLoss
UnisRec	BERT-110M	0.7702	0.5641	0.8196	0.5063
VQ-Rec	BERT-110M	0.7707	0.5641	0.8226	0.5025
P5	T5-223M	0.7790	0.5543	0.8333	0.4908
LLM2DIN	ChatGLM-6B	0.7874	0.5473	0.8307	0.4930
TALLRec	LLaMa2-7B	<u>0.7892</u>	<u>0.5451</u>	<u>0.8342</u>	<u>0.4862</u>
base(DIN)	N/A	0.7863	0.5486	0.8304	0.4937
KAR(DIN)	gpt-3.5-turbo	**0.7961***	**0.5370***	**0.8418***	**0.4801***

$*$ denotes statistically significant improvement over the second best baselines which is underlined (t-test with p-value < 0.05).

5.2.3 Improvement over Other Knowledge (RQ3). Finally, in Figure 4, we compare knowledge from LLMs and other sources, such as knowledge graph (KG), with DCNv1, DeepFM, and DIN as the backbone on MovieLens-1M dataset. We utilize a knowledge graph from LODrecsys [11], which maps items of MovieLens-1M to DBPedia entities. Then, the entity embedding of each item is extracted following KTUP [6] and used as an additional feature for the backbone model. The legend "**None**" denotes the backbone model without knowledge enhancement, while "**KG**", "**LLM**", and "**Both**" represent the backbone model enhanced by knowledge from KG, LLM, and both sources, respectively.

From Figure 4, we notice that knowledge from both KG and LLM can bring performance improvements; however, the enhancement of KG is much smaller. This might be attributed to the fact that KG only possesses manually annotated item-side knowledge while it lacks reasoning knowledge for user. According to our analysis in the next subsection 5.4, reasoning knowledge for user preference usually contributes more gains compared to item factual knowledge. Furthermore, jointly using the two kinds of knowledge does

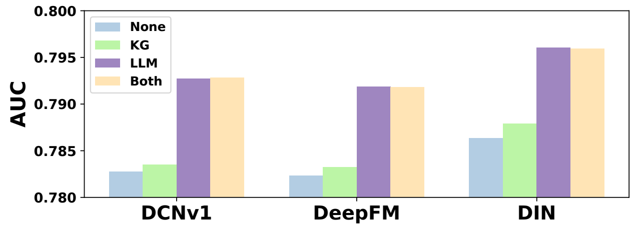


Figure 4: Comparison between knowledge from knowledge graph and LLM on MovieLens-1M dataset.

not exhibit significant improvement over using LLM alone. This suggests that LLM may already encompass the typical knowledge within KG, making knowledge derived solely from LLM sufficient.

5.3 Deployment & Online A/B Test (RQ4)

To validate the effectiveness of KAR, we conducted two online experiments on Huawei's news and music platforms, respectively. On the news platform, the experimental group, where we deployed KAR, utilized Huawei's large language model PanGu[77] to generate user preference, followed by retrieving relevant news through vector similarity. The control group utilized the original recall model as our baseline. During the online A/B test, KAR exhibited a 7% improvement on Recall metric compared with the baseline, resulting in significant business benefits.

In the music scenario, 10% of users were randomly selected into the experimental group and another 10% were in the control group. Both groups used the same base model for generating recommendations. The difference lies in their inputs, where the input of the experimental group included the knowledge representation augmented by KAR. To enhance the storage efficiency, we also employed PCA to reduce the dimension of representation to 64. In a 7-day online A/B test, KAR demonstrated a 1.7% increase in song play count, a 1.64% increase in the number of devices for song playback, and a 1.57% increase in total duration. Now, KAR has already been deployed online and serves the main traffic. This indicates that KAR can be successfully implemented in industrial settings and improve recommendation experience for real-world users.

5.4 Ablation Study (RQ5)

To study the impact of knowledge generated by LLMs, we conduct an ablation study on reasoning and factual knowledge on the Amazon-Books dataset. We select DCNv2, AutoInt, and DIN as backbone models and compare their performance with different knowledge enhancements, as shown in Figure 5. The legend **"None"** represents the backbone model without any knowledge enhancement. **"Fact"** and **"Reas"** indicate the backbone models enhanced with factual knowledge on item and reasoning knowledge about user preference, respectively, while **"Both"** represents the joint use of both knowledge types.

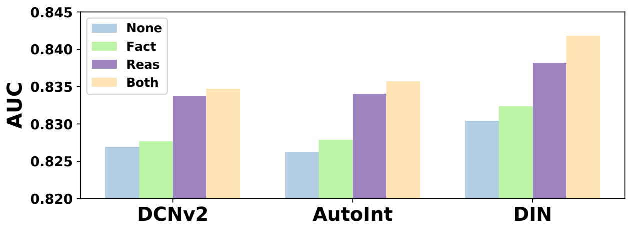


Figure 5: Ablation study about reasoning and factual knowledge on Amazon-Books dataset.

From Figure 5, we observe that both reasoning knowledge and factual knowledge can improve the performance of backbone models, with reasoning knowledge exhibiting a larger improvement. This could be attributed to the fact that reasoning knowledge inferred by the LLMs captures in-depth user preferences, thus compensating for the backbone model's limitations in reasoning underlying motives and intentions. Additionally, the joint use of both reasoning and factual enhancements outperforms using either one alone, even achieving a synergistic effect where $1 + 1 > 2$. One possible explanation is that reasoning knowledge contains external information that is not explicitly present in the raw data. When used independently, this knowledge could not be matched with candidate items. However, combining factual knowledge on items from the LLMs aligned with reasoning knowledge allows RSs to gain a better understanding of items according to user preferences.

5.5 Efficiency Study (RQ6)

To quantify the actual time complexity of KAR, we compare the inference time of KAR based on DIN with ChatGPT API, strongest baseline TALLRec, and base DIN model in Table 4. For **ChatGPT API**, we follow the zero-shot user rating prediction in [34] that provides the user viewing history and ratings as prompt and invokes the API to predict user ratings on candidate items. Since this approach does not allow setting a batch size, the table presents the average response time per sample. For KAR, we evaluate the two acceleration strategies as introduced in Section 4.5: **KAR**$_{w/\ apt}$, where the adaptor participates in the inference stage, and **KAR**$_{w/o\ apt}$, where the adaptor is detached from inference. The experiments of KAR and base model are all conducted on a Tesla V100 with 32G memory, with a batch size of 256. With the same Tesla V100, we also test **TALLRec** and only showcase the average inference time per sample, since 32G memory cannot handle LLM with batch size of 256. Table 4 presents the average inference time, from which we draw following conclusions.

Firstly, adopting LLM for direct inference is not feasible for RSs due to its large computational latency. The response latency of ChatGPT API is 4-6 seconds, which does not meet the real-time requirement of RSs typically demanding a response latency of within 100ms. Even if we finetune a relatively small LLM, such as TALLRec based on LLaMa2-7B, its inference latency is still close to 1 second, which is unbearable for industrial scenarios. Secondly, both acceleration methods of KAR achieve an inference time within 100ms, satisfying the low latency requirement. Importantly, if we employ the approach of prestoring reasoning and factual augmented vectors, *i.e.*, KAR$_{w/o\ apt}$, the actual inference time is nearly the same as that of the backbone model. This demonstrates the effectiveness of our proposed KAR and acceleration strategies.

Table 4: The comparison of inference time (s).

Model	MovieLens-1M	Amazon-Books
ChatGPT API	5.54	4.11
TALLRec	7.63×10^{-1}	7.97×10^{-1}
KAR$_{w/\ apt}$	8.08×10^{-2}	9.39×10^{-2}
KAR$_{w/o\ apt}$	6.64×10^{-3}	1.11×10^{-2}
base DIN	6.42×10^{-3}	1.09×10^{-2}

6 LIMITATIONS AND FUTURE WORK

Our work mainly optimizes online inference by using offline LLMs to generate knowledge so it can meet inference latency requirements for online services. Although offline LLM inference is already a resource-efficient solution, it still requires significant time and resources faced with large-scale users and items, e.g., billions. We can further speed up offline knowledge generation or reduce the quantity of knowledge we need to generate. Additionally, LLMs might introduce potential biases, such as popularity bias, inherent biases from training data, and hallucinated knowledge. Measuring and mitigating these biases is necessary and worth investigating.

7 CONCLUSION

Our work presents KAR, a framework for effectively incorporating open-world knowledge into recommender systems by exploiting large language models. KAR identifies two types of critical knowledge from LLMs, the reasoning knowledge on user preferences and factual knowledge on items, which can be proactively acquired by our designed factorization prompting. A hybrid-expert adaptor is devised to transform the obtained knowledge for compatibility with recommendation tasks. The obtained augmented vectors can then be used to enhance the performance of any recommendation model. Additionally, efficient inference is achieved through preprocessing and prestoring the LLM knowledge. KAR shows superior performance compared to the state-of-the-art methods and is compatible with various recommendation algorithms.

ACKNOWLEDGMENTS

The Shanghai Jiao Tong University team is partially supported by National Natural Science Foundation of China (62177033, 62076161) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). The work is also sponsored by Huawei Innovation Research Program. We thank MindSpore [1] for its partial support. The author Yunjia Xi is also supported by Wu Wen Jun Honorary Doctoral Scholarship.

REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn/

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 135–144.

[3] 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).

[4] Alexander Brinkmann, Roee Shraga, Reng Chiz Der, and Christian Bizer. 2023. Product Information Extraction using ChatGPT. *arXiv preprint arXiv:2306.14921* (2023).

[5] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023).

[6] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*. 151–161.

[7] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. 2023. When large language models meet personalization: Perspectives of challenges and opportunities. *arXiv preprint arXiv:2307.16376* (2023).

[8] Konstantina Christakopoulou, Alberto Lalama, Cj Adams, Iris Qu, Yifat Amir, Samer Chucri, Pierce Vollucci, Fabio Soldo, Dina Bseiso, Sarah Scodel, et al. 2023. Large language models for user interest journeys. *arXiv preprint arXiv:2305.15498* (2023).

[9] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *arXiv preprint arXiv:2205.08084* (2022).

[10] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering ChatGPT's Capabilities in Recommender Systems. *arXiv preprint arXiv:2305.02182* (2023).

[11] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. SPRank: Semantic Path-based Ranking for Top-N Recommendations using Linked Open Data. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2016).

[12] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318* (2021).

[13] Yingpeng Du, Di Luo, Rui Yan, Xiaopei Wang, Hongzhi Liu, Hengshu Zhu, Yang Song, and Jie Zhang. 2024. Enhancing job recommendation through llm-based generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8363–8371.

[14] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 320–335.

[15] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).

[16] Luke Friedman, Sameer Ahuja, David Allen, Terry Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. 2023. Leveraging Large Language Models in Conversational Recommender Systems. *arXiv preprint arXiv:2305.07961* (2023).

[17] 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).

[18] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt and Predict Paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*. 299–315.

[19] Yuqi Gong, Xichen Ding, Yehui Su, Kaiming Shen, Zhongyi Liu, and Guannan Zhang. 2023. An Unified Search and Recommendation Foundation Model for Cold-Start Scenario. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. 4595–4601.

[20] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia). 1725–1731.

[21] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3549–3568.

[22] Xiaobo Guo, Shaoshuai Li, Naicheng Guo, Jiangxia Cao, Xiaolei Liu, Qiongxu Ma, Runsheng Gan, and Yunan Zhao. 2023. Disentangled Representations Learning for Multi-target Cross-domain Recommendation. *ACM Transactions on Information Systems* 41, 4 (2023), 1–27.

[23] Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging Large Language Models for Sequential Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1096–1102.

[24] 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.

[25] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *Proceedings of the ACM Web Conference 2023*. 1162–1171.

[26] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 585–593.

[27] 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).

[28] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[29] Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards Reasoning in Large Language Models: A Survey. arXiv:2212.10403 [cs.CL]

[30] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: Combining Feature Importance and Bilinear Feature Interaction for Click-through Rate Prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

[31] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3, 1 (03 1991), 79–87.

[32] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* (2002), 422–446.

[33] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[34] 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).

[35] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

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

[37] Chen Li, Yixiao Ge, Jiayong Mao, Dian Li, and Ying Shan. 2023. TagGPT: Large Language Models are Zero-shot Multimodal Taggers. *arXiv preprint arXiv:2304.03022* (2023).

[38] 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).

[39] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2023. Large Language Models for Generative Recommendation: A Survey and Visionary Discussions. *arXiv preprint arXiv:2309.01157* (2023).

[40] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 539–548.

[41] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. XDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1754–1763.

[42] Guo Lin and Yongfeng Zhang. 2023. Sparks of Artificial General Recommender (AGR): Early Experiments with ChatGPT. *arXiv preprint arXiv:2305.04518* (2023).

[43] Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023. ClickPrompt: CTR Models are Strong Prompt Generators for Adapting Language Models to CTR Prediction. *arXiv preprint arXiv:2310.09234* (2023).

[44] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023. How Can Recommender Systems Benefit from Large Language Models: A Survey. *arXiv preprint arXiv:2306.05817* (2023).

[45] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. *arXiv preprint arXiv:2308.11131* (2023).

[46] 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).

[47] Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2023. Pre-train, prompt and recommendation: A comprehensive survey of language modelling paradigm adaptations in recommender systems. *arXiv preprint arXiv:2302.03735* (2023).

[48] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 452–461.

[49] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural Re-ranking in Multi-stage Recommender Systems: A Review. *arXiv preprint arXiv:2202.06602* (2022).

[50] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, and Jiebo Luo. 2023. Llmrec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780* (2023).

[51] Sheshera Mysore, Andrew McCallum, and Hamed Zamani. 2023. Large Language Model Augmented Narrative Driven Recommendations. *arXiv preprint arXiv:2306.02250* (2023).

[52] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[53] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023). https://doi.org/10.48550/arXiv.2303.08774

[54] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 499–508.

[55] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[56] Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Enhong Chen, et al. 2023. Large language model based long-tail query rewriting in taobao search. *arXiv preprint arXiv:2311.03758* (2023).

[57] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and Narrowing the Compositionality Gap in Language Models. arXiv:2210.03350 [cs.CL]

[58] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with Language Model Prompting: A Survey. arXiv:2212.09597 [cs.CL]

[59] Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-BERT: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4320–4327.

[60] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2023. Representation learning with large language models for recommendation. *arXiv preprint arXiv:2310.15950* (2023).

[61] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[62] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[63] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[64] 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).

[65] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. *Advances in neural information processing systems* 26 (2013).

[66] Hangyu Wang, Jianghao Lin, Xiangyang Li, Bo Chen, Chenxu Zhu, Ruiming Tang, Weinan Zhang, and Yong Yu. 2023. FLIP: Towards Fine-grained Alignment between ID-based Models and Pretrained Language Models for CTR Prediction. *arXiv e-prints* (2023), arXiv–2310.

[67] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*. 3307–3313.

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

[69] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*. Article 12, 7 pages.

[70] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-Scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*. 1785–1797.

[71] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.

[72] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A Survey on Large Language Models for Recommendation. *arXiv preprint arXiv:2305.19860* (2023).

[73] Yunjia Xi, Weiwen Liu, Jieming Zhu, Xilong Zhao, Xinyi Dai, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2022. Multi-Level Interaction Reranking with User Behavior History. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[74] Bin Yin, Junjie Xie, Yu Qin, Zixiang Ding, Zhichao Feng, Xiang Li, and Wei Lin. 2023. Heterogeneous knowledge fusion: A novel approach for personalized recommendation via llm. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 599–601.

[75] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[76] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 271–278.

[77] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. 2021. Pangu-α: Large-scale autoregressive pretrained Chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369* (2021).

[78] Chao Zhang, Shiwei Wu, Haoxin Zhang, Tong Xu, Yan Gao, Yao Hu, and Enhong Chen. 2024. NoteLLM: A Retrievable Large Language Model for Note Recommendation. *arXiv preprint arXiv:2403.01744* (2024).

[79] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Agentcf: Collaborative learning with autonomous language agents for recommender systems. *arXiv preprint arXiv:2310.09233* (2023).

[80] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing Cold Start in Recommender Systems: A Semi-Supervised Co-Training Algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. 73–82.

[81] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[82] Bowen Zheng, Zihan Lin, Enze Liu, Chen Yang, Enyang Bai, Cheng Ling, Wayne Xin Zhao, and Ji-Rong Wen. 2024. A Large Language Model Enhanced Sequential Recommender for Joint Video and Comment Recommendation. *arXiv preprint arXiv:2403.13574* (2024).

[83] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-through Rate Prediction *(AAAI'19)*. Article 729, 8 pages.

[84] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1059–1068.

[85] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).