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

Jieming Zhu
jamie.zhu@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

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

Rui Zhang
rayteam@yeah.net
ruizhang.info
Shenzhen, China

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

ABSTRACT

Recommender systems play a vital role in various online services. However, the insulated nature of training and deploying separately within a specific domain limits their 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 capability. Nevertheless, previous attempts to directly use LLMs as recommenders have not achieved satisfactory results. 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.

1 INTRODUCTION

Recommender systems (RSs) are ubiquitous in today's online services, shaping and enhancing user experiences in various domains such as movie discovery [24], online shopping [13], and music streaming [41]. 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 [24, 54], isolated from the knowledge of the external world, thereby restricting the information that could be 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 [7, 27]. Hence, in this work, 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).

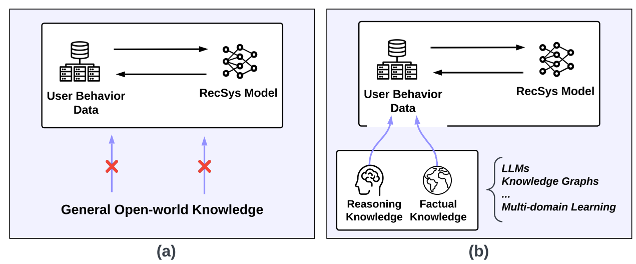

Figure 1: Comparison between (a) closed recommender systems and (b) open-world recommender systems.

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 — the *reasoning knowledge* on in-depth user preferences which is inferred from user behaviors and profiles, and 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. Deducing the underlying preferences and motives that drive user behaviors can help us gain deeper insights and clues about the users. A person's personality, occupation, intentions, preferences, and tastes could be reflected in their behavior history. This preference reasoning can even integrate seasonal factors (*e.g.*, holiday-themed movie preferences during Christmas) or external events (*e.g.*, an increased interest in health products during a pandemic) and provide human-like recommendations with clear evidence, which goes beyond identifying

*Both authors contributed equally to this research.

basic behavior patterns as in classical recommenders. On the other hand, the factual knowledge on items provides valuable common sense information about the candidate items and thereby improves 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 expands the original data and is beneficial to the recommendation task.

Several existing studies attempt to complement the closed recommender systems with additional information by knowledge graphs [11, 43] or multi-domain learning [12, 38]. However, constructing comprehensive and accurate knowledge graphs or multi-domain datasets requires considerable extra human effort, and the accessible knowledge remains limited. Moreover, they only focus on extracting the factual knowledge from the external world, and overlook the reasoning knowledge on user preferences [11].

Recent rapid developments in large language models (LLMs) have revolutionized the learning paradigm of various research fields and show great potential in bridging the gap between classical recommenders and open-world knowledge [51]. With the immense scale of the model and the corpus size, these large pretrained language models like GPT-4 [31], LLaMA [40] have shown remarkable capabilities, such as problem solving, logical reasoning, creative writing [1]. Learning from an extensive corpus of internet texts, LLMs have encoded a vast array of world knowledge — from basic factual information to complex societal norms and logical structures [1]. As a result, LLMs can perform basic logical reasoning that aligns with known facts and relationships [47, 52].

Recently, a few studies have attempted to apply LLMs as recommenders by converting recommendation tasks and user profiles into prompts [4, 8, 30]. Though some preliminary findings have been obtained, the results of using LLMs as recommenders are far from optimal for real-world recommendation scenarios due to the following shortcomings. 1) **Predictive accuracy.** The accuracy of LLMs is generally outperformed by classical recommenders in most cases, since LLMs have not been trained on specific recommendation data [28, 30]. The lack of recommendation domain knowledge and collaborative signals prevents LLMs from adapting to individual user preferences. 2) **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. 3) **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 [32]. Requiring direct recommendation results from LLMs is currently beyond their capability and cannot fully exploit the open-world knowledge encoded in LLMs [4, 22].

Therefore, the goal of this work is to effectively incorporate open-world knowledge while preserving the advantages of classical recommender systems. However, despite the appealing capabilities of LLMs, extracting and utilizing knowledge from them is a non-trivial task. For one thing, LLMs encode vast corpora of world knowledge across various scenarios. Identifying useful external knowledge for recommendation and eliciting the accurate reasoning process on user preferences are quite challenging. Moreover, the world knowledge generated by LLMs is in the form of human-like texts and cannot be interpreted by recommendation algorithms. Even if some LLMs are open-sourced, the decoded outputs are usually large dense vectors (*e.g.*, 4096 for each token), which are highly abstract and not directly compatible with recommender systems. Effectively transforming the output knowledge to be compatible with the recommendation space, without information loss or misinterpretation, is pivotal for the quality of the recommendation. Moreover, the knowledge generated by LLMs can sometimes be unreliable or misleading due to the hallucination problem [21]. Hence, it is critical to increase the reliability and availability of the generated knowledge to fully unleash the potential of open-world recommender systems.

To address the above problems, we propose an Open-World Knowledge Augmented Recommendation Framework with Large Language Models, (dubbed *KAR*). KAR is a model-agnostic framework that bridges classical recommender systems and open-world knowledge, leveraging both reasoning and factual knowledge from the LLMs. By first leveraging LLMs to generate open-world knowledge, and then applying classical recommender systems to model the collaborative signals, we combine the advantages of both LLMs and RSs and significantly improve the model's predictive accuracy. We also propose to prestore the obtained knowledge to avoid the inference latency issue when incorporating LLMs in RSs.

Specifically, KAR consists of 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 key factors to generate the reasoning knowledge on users and the factual knowledge on items. In knowledge adaptation, we encode and transform the generated knowledge to augmented vectors in recommendation space by our proposed *hybrid-expert adaptor* module. Finally, in the knowledge utilization stage, the recommendation model incorporates the augmented vectors with original domain features for prediction, combining both the recommendation domain knowledge and the open-world knowledge. Our main contributions can be summarized as follows:

- We present an open-world recommender system exploiting large language models, KAR, which bridges the gap between the recommendation domain knowledge and the open-world knowledge. To the best of our knowledge, this is the first practical solution that introduces logical reasoning for user preferences to the recommendation domain.
- KAR transforms the open-world knowledge to dense vectors located in recommendation space, which are compatible with any recommendation models and are flexible to use.
- The knowledge generation and encoding process of KAR can be preprocessed and prestored for fast training and inference, avoiding the large inference latency when using LLMs in RSs.

Extensive experiments conducted on public datasets show that KAR significantly outperforms the state-of-the-art models, and is compatible with various recommendation algorithms. We will release the code of KAR as well as the generated textual knowledge from LLMs[1] to facilitate future research. We believe that KAR not only sheds light on the possibilities of injecting 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

This section reviews studies on traditional recommender systems and recent advances in recommendation with language models.

2.1 Traditional Recommender Systems

Traditional recommender systems emphasize feature interaction learning over multi-field categorical data (*i.e.,* user, item, and context features) [10, 19, 25, 39, 45, 46, 53, 54]. For example, DeepFM [10] and xDeepFM [26] combine explicit feature interactions and deep neural networks. AutoInt [39] utilizes a multi-head self-attentive neural network to capture feature interactions with different orders. Besides, attention mechanisms are introduced to model user behavior sequences in DIN [54] and DIEN [53]. Yet the data used to train these recommender systems are restricted to a specific domain.

To incorporate external knowledge or related background, several studies tend to augment recommender systems with knowledge graphs [11, 42, 43] or multi-domain learning techniques [2, 12, 38]. However, constructing comprehensive and accurate knowledge graphs or multi-domain datasets requires considerable extra human effort. Instead, LLMs offer a more straightforward approach, in which a significant amount of open-world knowledge has already been encoded within the pretrained model parameters.

2.2 Recommendation with Language Models

2.2.1 LM as Textual Encoder. Pretrained language models (LMs) are used to encode the textual features in recommendations (*e.g.*, item descriptions, user reviews) for better user or item representations [5, 15, 16, 34]. U-BERT [34] uses the users' review texts encoded by BERT to complement user representations. Similarly, ZESRec [5] applies BERT to convert item descriptions into continuous representations and uses them as universal item embeddings for zero-shot recommendation. UniSrec [16] takes user behavior sequences as input and learns universal sequence representations using pretrained language models. VQ-Rec [15] proposes to first obtain text encodings via language models and then map them to discrete codes for embedding lookup, balancing the semantic features and domain features. However, these methods simply use LMs as fixed encoders to convert texts to dense vectors, where no new textual information is generated.

2.2.2 LM as Knowledge Enhancer. Pretrained language models (*e.g.,* BERT [23], GPT [35], T5 [36]) encode knowledge from training corpora and can be exploited for recommendations. LMRecSys [50] is one of the earliest attempts to transfer the session-based recommendation task into prompts and evaluate the performance of BERT [23] and GPT-2 [35] in movie recommendation. Empirical results show that language models tend to underperform traditional recommenders such as GRU4Rec [14]. P5 [9] and M6-Rec [3] finetune pretrained language models (T5 [36] or M6 [29]) by converting multiple recommendation tasks to natural language sequences to incorporate knowledge and semantics inside the training corpora for personalization. More recently, Zhang *et al.* [48] propose to formulate recommendations as instruction followed by language models and design 39 instruction templates for user behavior modeling. In this earlier stage, the sizes of the adopted language models for recommendation are relatively small (*e.g.,* billions of parameters), and finetuning is usually involved for better performance.

With the scaling of the model and corpus size, especially with the emergence of ChatGPT [31], LLMs have shown uncanny capability in a wide variety of tasks. One of the unique abilities of LLMs is reasoning, which becomes apparent only when the size of the model surpasses a certain threshold. The knowledge scope of LLMs has also significantly broadened compared to earlier language models. Zero-shot learning or in-context learning is widely used since retraining LLMs is impractical and unnecessary. Several studies apply LLMs as recommenders and achieve some preliminary results [4, 8, 17, 22, 30, 44]. ChatRec [8] employs LLMs as a recommender system interface for conversational multi-round recommendations. Liu *et al.* [30] study whether ChatGPT can serve as a recommender with task-specific prompts and report the zero-shot performance. Hou *et al.* [17] further report the zero-shot ranking performance of LLMs with historical interaction data.

However, directly using LLMs as recommenders generally fall behind state-of-the-art recommendation algorithms, implying the importance of domain knowledge and collaborative signals for recommendation tasks [4, 22, 28]. Moreover, these preliminary studies mainly overlook the inference latency and the compositional gap problem of LLMs. In this work, we propose to extract both reasoning and factual knowledge from LLMs to enhance the user and the item representations for recommendation models. The open-world knowledge is effectively extracted and adapted to the recommendation domain as augmented representation vectors, which are compatible with any recommendation algorithms. Besides, prestored representations also allow for fast training and inference.

3 PRELIMINARIES: CTR PREDICTION

Click-Through Rate (CTR) prediction aims at accurately predicting the probability of a user clicking an item, which is the core task for recommender systems. Therefore, we mainly focus on the CTR prediction task, and give the problem formulation and representative network structures in this section. Our proposed framework can be easily extended to other tasks like rating prediction or sequential recommendation.

CTR prediction can be 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 represent the feature 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 the corresponding field.

[1]Code and knowledge will be available at https://gitee.com/mindspore/models/tree/master/research/recommend/KAR

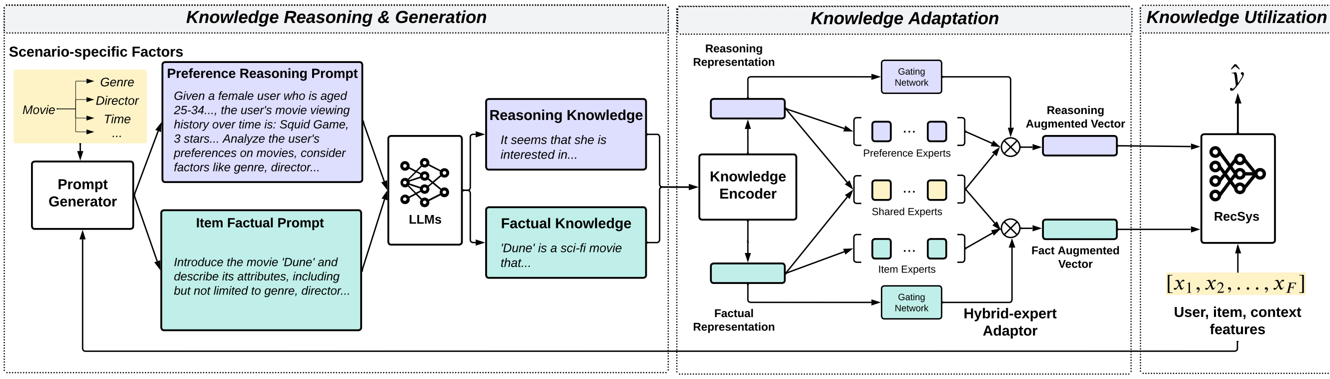

Figure 2: The overall framework of KAR, which consists of three stages: (1) Knowledge reasoning and generation; (2) Knowledge adaption; and (3) Knowledge utilization.

CTR models are trained to learn a function that can accurately predict the click probability $P(y_i = 1|x_i)$ for each sample x_i. Earlier work focuses on the feature interactions between different fields such as DCN [45], DeepFM [10], FibiNet [19], AutoInt [39], which can be formulated as

$$\hat{y_i} = f(x_i; \theta)\,, \tag{1}$$

where f is the learned function with parameters θ. Recent studies emphasize sequential behavior modeling like DIN [54] or DIEN [53] and incorporate the user behavior sequence h_i together with the current sample x_i. Hence the task can be extended to

$$\hat{y_i} = f(x_i, h_i; \theta)\,. \tag{2}$$

Typically, the CTR models may have the following four layers: (1) *embedding layer*; (2) *feature interaction layer*; (3) *user behavior modeling layer*; and (4) *output layer* [49]. The categorical features are first converted into low-dimensional dense embeddings in the embedding layer. Then the embeddings are fed into the feature interaction layer to capture complex relationships and high-order interactions, enabling the model to learn non-linear patterns. Meanwhile, the user behavior modeling layer models captures user interests and preferences based on historical behavior or contextual information, utilizing techniques like RNNs or attention mechanisms to capture temporal dependencies. Finally, the output layer takes the processed features and generates the final prediction of the click probability, usually using a linear or MLP layer followed by a sigmoid activation function. After obtaining the predictions from the output layer, the model is typically trained with Binary Cross-Entropy (BCE) loss:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\,. \tag{3}$$

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

4 METHODOLOGY

We first provide an overview of our proposed Open-World Knowledge Augmented Recommendation Framework with Large Language Models, and then elaborate on the details of each component.

4.1 Overview

To extract open-world knowledge from LLMs and incorporate it into classical recommendation systems (RSs), we design KAR, as shown in Figure 2. This framework 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 the open-world knowledge beyond the original recommendation dataset.

Knowledge Adaptation Stage converts textual knowledge into compact and relevant representations suitable for recommendation, bridging the gap between LLMs and RSs. First, the reasoning and factual 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.

Knowledge Utilization Stage integrates the reasoning and fact augmented vectors into an existing recommendation model, enabling it to leverage both domain knowledge and open-world knowledge during the recommendation process.

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 [18, 33]. This opens up new possibilities for incorporating reasoning knowledge for user preferences and factual knowledge for candidate items in recommendation systems. However, it is non-trivial to extract the reasoning knowledge and corresponding factual knowledge from LLMs due to the following two challenges.

Considering the reasoning knowledge, according to [32], LLMs often suffer from the *compositional gap* where the model fails at generating the correct answer to the compositional question but

[2]the embedding space from language models

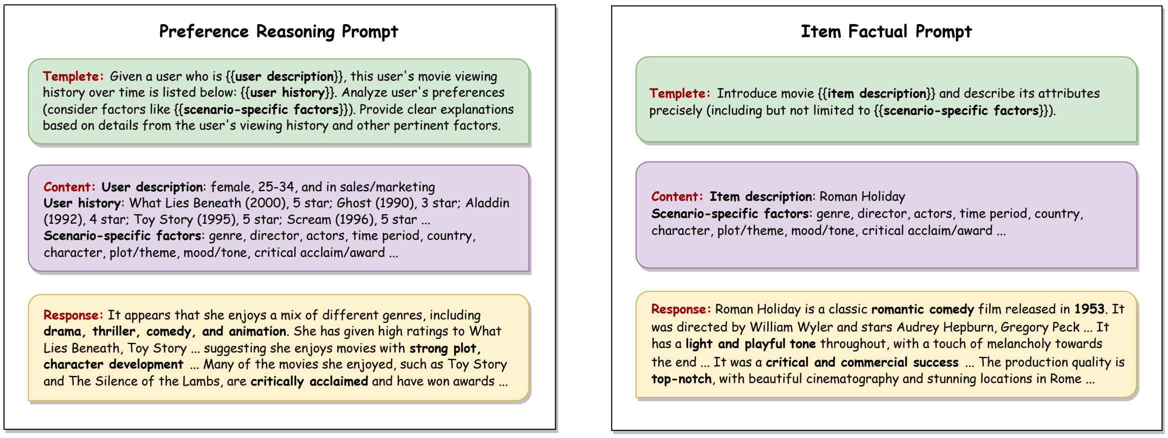

Figure 3: Example prompts for KAR. The green, purple, and yellow text bubbles represent the prompt template, the content to be filled in the template, and the response generated by LLMs, respectively (some text has been omitted due to the page limits).

can correctly answer all its sub-questions. User's clicks on items are motivated by multiple key aspects and user's interests are diverse and multifaceted, which involve multiple reasoning steps. To this end, LLMs may not be able to directly produce accurate reasoning knowledge. Expecting LLMs to provide precise recommendations in one step as in previous work [8, 17, 30] might be overly ambitious.

As for the factual knowledge, LLMs contain massive world knowledge, yet not all of it is useful for recommendation. When the request to an LLM is too general, the generated factual knowledge may be correct but useless, as it may not align with the inferred user preferences. For example, an LLM may infer that a user may prefer highly acclaimed movies that have received multiple awards, while the generated factual knowledge is about the storyline of the target movie. This mismatch between preference reasoning knowledge and item factual knowledge may limit the performance of RSs.

Therefore, inspired by the success of Factorization Machines [37] 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 specific 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 are able to provide related factors. After obtaining the potential factors, 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 the factors like *topic, source, region, style, freshness, clarity, and impact.* This collaborative process between LLMs and experts ensures that the chosen factors encompass the critical dimensions of user preference and item characteristics for each scenario.

4.2.2 LLM as Preference Reasoner & Knowledge Provider. After obtaining the scenario-specific factors, we introduce them into our prompt engineering. To extract reasoning and factual knowledge from the open world, we propose to apply the LLM as a *preference reasoner* to infer user preferences, and a *knowledge provider* to acquire external factual knowledge for candidate items. Therefore, we design two types of prompts accordingly: 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 preference reasoning prompt, where the user profile description and behavior history provide LLM with the necessary context and user-specific information to understand the user's preferences. Scenario-specific factors can instruct the LLM to analyze the user preference from different facets and allow the LLM to recall the relevant knowledge more effectively and comprehensively. We can observe that in the generated response, the LLM analyzes and identifies the user's preferences for the corresponding factors based on the user's profile and behavior history, which is beneficial for recommendations.

Item factual prompt is designed to fill in 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. With the designed prompt, LLM provides external knowledge that aligns with user preferences, allowing for more accurate and personalized recommendations.

By combining the two kinds of prompts, we enable the LLM to act as both a preference reasoner and a knowledge provider, thereby extracting the open-world knowledge from LLMs and expanding the knowledge scope of RSs.

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 [21].

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 and aggregates them effectively. 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 the generated knowledge and bridges the gap between LLMs and RSs.

4.3.1 Knowledge Encoder. To harness the potential of textual knowledge generated by LLMs, we employ a knowledge encoder, *e.g.*, BERT, to obtain the encodings for each token within the text. Then, we require an aggregation process that combines the information from 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

$$\begin{aligned} r_i^p &= \mathrm{Aggr}(\mathrm{Encoder}(klg_i^p))\,, \\ r_i^\iota &= \mathrm{Aggr}(\mathrm{Encoder}(klg_i^\iota))\,, \end{aligned} \tag{4}$$

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, but we also explore the effectiveness of other aggregation methods. Note that the knowledge encoder is devised for situations where we only have access to the textual outputs of LLMs. If the dense vector outputs from LLMs are available, a separate knowledge encoder can be eliminated.

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) [20] approach. This allows us to fuse knowledge from different facets and benefits 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

$$\begin{aligned} \alpha_i^p &= \mathrm{Softmax}(g^p(r_i^p))\,, \quad \alpha_i^\iota = \mathrm{Softmax}(g^\iota(r_i^\iota))\,, \\ \hat{r}_i^p &= \sum\nolimits_{e \in \mathcal{S}_s} \alpha_{i,e}^p \times e(r_i^p) + \sum\nolimits_{e \in \mathcal{S}_p} \alpha_{i,e}^p \times e(r_i^p)\,, \\ \hat{r}_i^\iota &= \sum\nolimits_{e \in \mathcal{S}_s} \alpha_{i,e}^\iota \times e(r_i^\iota) + \sum\nolimits_{e \in \mathcal{S}_\iota} \alpha_{i,e}^\iota \times e(r_i^\iota)\,, \end{aligned} \tag{5}$$

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 Multi-Layer Perceptron (MLP), facilitating dimensionality reduction and space transformation.

4.4 Knowledge Utilization

Once we have obtained the reasoning augmented vector and the fact augmented vector, we can then incorporate them into backbone CTR models. In this section, we explore a straightforward approach where these augmented vectors are directly treated as additional input features. Specifically, we use them as additional feature fields in the feature interaction layer (introduced in Section 3) of a CTR model, allowing them to explicitly interact with other categorical features. During training, the hybrid-expert adaptor module is jointly optimized with the backbone model to ensure that the transformation process adapts to the current data distribution. Generally, KAR can be formulated as

$$\hat{y}_i = f(x_i, h_i, \hat{r}_i^p, \hat{r}_i^\iota; \theta)\,. \tag{6}$$

which is enhanced by the the reasoning augmented vector $\hat{r}_i^p$ and the fact augmented vector $\hat{r}_i^\iota$. Importantly, KAR only modifies the input of the backbone model and is independent of the design and loss function of backbone model, so it is flexible and compatible with various backbone model designs. Furthermore, it can be extended to other recommendation tasks, such as sequential recommendation and direct recommendation, by simply adding two augmented vectors in the input, similar to Eq. (6). By incorporating the knowledge augmented vectors into the input layer, KAR combines both the open-world knowledge and the recommendation domain knowledge in a unified manner to provide more informed and personalized recommendations.

4.5 Speed-up Approach

Our proposed KAR framework adopts LLMs to generate reasoning knowledge for user preferences and factual knowledge for candidate items. 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 the knowledge representation r_i^p and r_i^ι generated by the knowledge encoder or the LLM into a database. As such, we only use the LLM and knowledge encoder once before the training of backbone models. During the training and inference stage of the backbone model, relevant representations are retrieved from the database.

If we have stricter requirements for inference time or storage efficiency, we can detach the adaptor from the model after training and further prestore the reasoning and factual 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 5.4. 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. In our framework, the inference time complexity is $O(f(n+2,m)) = O(f(n,m))$, which is equivalent to the complexity of the original model.

Since item features are relatively fixed and do not change frequently, it is natural and feasible to prestore the item factual knowledge for further use. Moreover, user behaviors evolve over time, making it challenging for LLMs to provide real-time reasoning knowledge about behavioral changes. However, considering that long-term user preferences are relatively stable, and the backbone model already emphasizes modeling recent user behaviors, it is unnecessary to require LLMs to have access to real-time behaviors. Therefore, LLM can infer long-term preferences based on users' long-term behaviors, allowing for conveniently prestoring the generated knowledge without frequent updates. The inference overhead of LLMs can also be significantly reduced. The backbone models can capture ever-changing short-term preferences with timely model updates. This can take better advantage of both LLMs and recommendation models.

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 different backbone CTR models?
- **RQ2:** How does KAR perform compared with other methods with pretraining techniques?
- **RQ3:** How do the reasoning knowledge and factual knowledge generated by the LLM contribute to performance improvement?
- **RQ4:** How do different knowledge adaptation approaches, such as different knowledge encoders and semantic transformation approaches, impact the performance?
- **RQ5:** 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 a public dataset[3], MovieLens-1M dataset[4], which contains 1 million ratings provided by 6000 users for 4000 movies. Following the data processing similar to DIN [54], 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 input 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.

5.1.2 Backbone Models. We implement 9 representative CTR prediction models as our backbone models, which can be categorized into feature interaction models and user behavior models.

Feature Interaction Models focus on modeling feature interactions between different feature fields. **DeepFM** [10] is a classic CTR model that combines factorization machine (FM) and neural network to capture low-order and high-order feature interactions. **xDeepFM** [26] leverages the power of both deep network and Compressed Interaction Network to generate feature interactions at the vector-wise level. **DCN** [45] incorporates cross-network architecture and the DNN model to learn the bounded-degree feature interactions. **DCNv2** [46] is an improved framework of DCN which is more practical in large-scale industrial settings. **FiBiNet** [19] can dynamically learn the feature importance by Squeeze-Excitation network and fine-grained feature interactions by bilinear function. **FiGNN** [25] converts feature interactions into modeling node interactions on the graph for modeling feature interactions in an explicit way. **AutoInt** [39] adopts a self-attentive neural network with residual connections to explicitly model the feature interactions.

User Behavior Models emphasize on modeling sequential dependencies of user behaviors. **DIN** [54] utilizes attention to model user interests dynamically with respect to a certain item. **DIEN** [53] extends DIN by introducing an interest evolving mechanism to capture the dynamic evolution of user interests over time.

5.1.3 Pretraining Baselines. We compare KAR with some baselines that leverage pretraining techniques with text or behavioral data to enhance the recommendation models, such as P5 [9], UniSRec [16], VQRec [15]. **P5** [9] is a text-to-text paradigm that unifies recommendation tasks and learns different tasks with the same language modeling objective during pretraining. **UniSRec** [16] designs a universal sequence representation learning approach for sequential recommenders, which introduces contrastive pretraining tasks to effective transfer across scenarios. **VQ-Rec** [15] uses Vector-Quantized item representations and a text-to-code-to-representation scheme, achieving effective cross-domain and cross-platform sequential recommendation. 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.

5.1.4 Evaluation Metrics. To validate the effectiveness of our framework, we employ AUC (Area under the ROC curve) and Logloss (binary cross-entropy loss) as the evaluation metrics, which are widely used in the field. A higher AUC value or a lower Logloss value, even by a small margin (e.g., 0.001), can be considered as a significant improvement in CTR prediction performance, as indicated by previous studies [26, 46].

[3]Considering the large API cost of the LLMs, we only conduct experiments on one public dataset.
[4]https://grouplens.org/datasets/movielens/1m/

Table 1: Comparison between KAR and backbone models.

Backbone Model	AUC			Logloss		
	base	KAR	improv.	base	KAR	improv.
DCNv2	0.7924	**0.8049***	1.58 %	0.5451	**0.5315***	2.50 %
DCN	0.7929	**0.8043***	1.46 %	0.5457	**0.5319***	2.53 %
DeepFM	0.7928	**0.8041***	1.44 %	0.5462	**0.5321***	2.57 %
FiBiNet	0.7925	**0.8051***	1.59 %	0.5450	**0.5310***	2.56 %
AutoInt	0.7934	**0.8060***	1.59 %	0.5440	**0.5297***	2.65 %
FiGNN	0.7944	**0.8054***	1.39 %	0.5424	**0.5307***	2.16 %
xDeepFM	0.7942	**0.8041***	1.25 %	0.5457	**0.5317***	2.57 %
DIEN	0.7960	**0.8059***	1.25 %	0.5469	**0.5298***	3.13 %
DIN	0.7975	**0.8066***	1.15 %	0.5387	**0.5304***	1.55%

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

5.1.5 Implementation Details. We utilize API of a widely-used LLM for generating reasoning and factual knowledge. Due to the unavailability of the LLM's parameters, ChatGLM [6] is employed as the knowledge encoder to encode the knowledge, followed by average pooling as the aggregation function in Eq. (4). 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 shared experts and 5-6 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. In order to ensure fair comparisons, the parameters of the base backbone model and the baselines are also tuned to achieve their optimal performance.

5.2 Effectiveness Comparison

5.2.1 Improvement over Backbone Models (RQ1). We first compare the performance of our proposed framework, KAR, with the backbone models, and the results are shown in Table 1. From the results, we can have the following observations: (i) Applying KAR significantly improves the performance of backbone models. For example, when using DCNv2 as the backbone model, KAR achieves a 1.58% increase in AUC and a 2.50% decrease in Logloss, demonstrating the effectiveness of incorporating 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 the equipment of KAR, the selected 9 representative CTR models in the table all achieve an AUC improvement of 1.1-1.6%, indicating the universality of the KAR. (iii) KAR shows greater improvement in feature interaction models compared to user behavior models. This may be because the knowledge augmented vectors generated by KAR are utilized more effectively by the feature interaction layer than the user behavior modeling layer. The dedicated feature interaction designed in the feature interaction layer may better exploit the information contained in the knowledge vectors.

5.2.2 Improvement over Baselines (RQ2). Next, we compare KAR with recent baselines that employ pretraining techniques [5]. The results are presented in Table 2, from which we make the following observations: (i) Our framework, KAR, significantly outperforms models based on text pretraining or user sequence pretraining. For instance, with DIN as the backbone, KAR achieves a 1.92% improvement in AUC and a 2.79% improvement in Logloss over VQ-Rec. (ii) Pretraining models based on user sequences, such as UniSRec and VQ-Rec, perform poorly, and even fail to surpass the basic model, DIN. This could be attributed to the fact that the two studies originally focus on the sequential recommendation domain with little exploration on feature interaction, while CTR prediction heavily relies on feature interaction.

[5]We did not include the results of P5, because when adapting its official code to CTR task, its metrics are notably worse than others even after careful parameter tuning.

Table 2: Comparison between KAR and baselines.

Model	AUC	Logloss
UnisRec	0.7891	0.5496
VQ-Rec	0.7914	0.5456
base(DIN)	<u>0.7975</u>	<u>0.5387</u>
KAR(DIN)	**0.8066***	**0.5304***

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

5.3 Ablation Study

5.3.1 Reasoning and Factual Knowledge (RQ3). To study the impact of knowledge generated by LLMs, we conduct an ablation study on reasoning and factual knowledge. We select DIN, AutoInt, and DCNv2 as backbone models and compare their performance with different knowledge enhancements, as shown in Figure 4. 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.

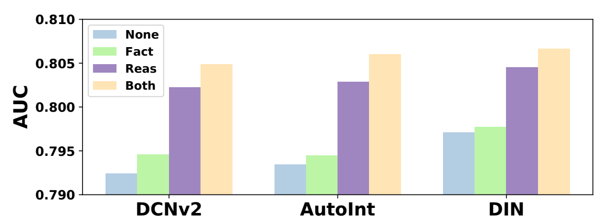

Figure 4: Ablation on reasoning and factual knowledge.

From Figure 4, 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 external knowledge could not be matched with candidate items. However, combining the externally generated factual knowledge on items from the LLMs aligned with reasoning knowledge allows the recommendation model to gain a better understanding of items according to user preferences.

Table 3: Impact of different knowledge encoders.

Variant	BERT		ChatGLM	
	AUC	**Logloss**	**AUC**	**Logloss**
base(DIN)	0.7975	0.5387	0.7975	0.5387
KAR(LR)	0.7746	0.5699	0.7903	0.5490
KAR(MLP)	0.7882	0.5518	0.7986	0.5399
KAR-MLP	<u>0.8040</u>	0.5345	0.8042	0.5318
KAR-MoE	**0.8046**	**0.5322**	<u>0.8052</u>	<u>0.5305</u>
KAR	**0.8046**	<u>0.5324</u>	**0.8066**	**0.5304**

5.3.2 Knowledge Encoders (RQ4). We employ two different language models, BERT [23] and ChatGLM [6], to investigate the impact of different knowledge encoders on model performance. Additionally, we designed several variants to demonstrate how the representations generated by knowledge encoders are utilized. **KAR(LR)** applies average pooling to token representations from the knowledge encoder and directly feeds the result into a linear layer to obtain prediction scores, without utilizing a backbone CTR model. **KAR(MLP)** replaces the linear layer of KAR(LR) with an MLP. **KAR-MLP** and **KAR-MoE** replace the hybrid-expert adaptor with an MLP and a Mixture-of-Experts (MoE), respectively. The original KAR and the two variants, KAR-MLP and KAR-MoE, all adopt DIN as the backbone model. The performance of these variants under two knowledge encoders is presented in Table 3, from which we draw the following conclusions.

Firstly, we can observe that, overall, variants with ChatGLM as the knowledge encoder outperform those with BERT. The performance of KAR(LR) and KAR(MLP) can be considered as a measure of the quality of the encoded representations, since they directly adopt the representations for prediction. Considering KAR(LR) and KAR(MLP), the superior performance of ChatGLM over BERT indicates that ChatGLM performs better in preserving the information within knowledge from LLMs, which may be attributed to the larger size and better text comprehension of ChatGLM (6 billion) compared to BERT (110 million). With ChatGLM, the performance of KAR(MLP) even surpasses base(DIN), validating the effectiveness of our generated open-world knowledge.

Secondly, only leveraging the open-world knowledge generated by LLMs for recommendation can lead to some improvements in certain cases, but integrating it with recommendation domain knowledge yields more promising results. Amongst the methods directly utilizing encoded representations, only KAR(MLP) with ChatGLM gains a modest improvement (0.14% in AUC) over the base(DIN). However, combining the generated open-world knowledge with the domain knowledge in the classical RSs, as in KAR, brings significant enhancements (1.15% in AUC). We attribute this improvement to KAR successfully bridging the open-world knowledge and the recommendation domain knowledge.

Lastly, the performance of KAR benefits from complex semantic transformation structures, but the knowledge encoder also limits it. The results on ChatGLM show that a simple MLP is less effective than MoE and our designed hybrid-expert adaptor outperforms the MoE, indicating that the transformation from semantic space to recommendation space entails a complex network structure. However, with BERT as the knowledge encoder, KAR-MoE and KAR-MLP exhibit similar performance, suggesting that the information from BERT is limited and only employing an MoE is sufficient.

Table 4: Impact of different aggregation and semantic transformation approaches.

Variant	DIN		DCNv2		AutoInt	
	AUC	**Logloss**	**AUC**	**Logloss**	**AUC**	**Logloss**
base	0.7975	0.5387	0.7924	0.5451	0.7934	0.5440
KAR-last	0.8037	0.5320	0.8021	0.5346	0.8006	0.5366
KAR-wavg	0.8046	0.5335	0.8014	0.5355	0.8023	0.5344
KAR-MLP	0.8042	0.5318	0.8029	0.5331	0.8031	0.5349
KAR-MoE	<u>0.8052</u>	<u>0.5305</u>	<u>0.8036</u>	<u>0.5329</u>	<u>0.8041</u>	<u>0.5325</u>
KAR	**0.8066**	**0.5304**	**0.8049**	**0.5315**	**0.8060**	**0.5297**

5.3.3 Aggregation and Semantic Transformation (RQ4). After obtaining representations from knowledge encoder, it is crucial to explore how the aggregation and semantic transformation affect the performance. Therefore, we designed several variants with different aggregation and semantic transformation approaches to investigate their impact. In the original version of **KAR**, the aggregation function is average pooling over representations of all tokens. **KAR-last** only utilizes the representation corresponding to the last token, while **KAR-wavg** applies a weighted average over all the tokens, with higher weights assigned to tokens towards the end of the sequence. Similar to Section 5.3.2, **KAR-MLP** and **KAR-MoE** replace the hybrid-expert adaptor with an MLP and an MoE, respectively. To ensure a fair comparison, all the variants and KAR use ChatGLM as knowledge encoder and DIN/DCNv2/AutoInt as backbone models, and the comparison is presented in Table 4.

First, compared to KAR, the performances of all variants decline to some extent, yet they still exhibit significant improvement over backbone models. This suggests that different aggregation and semantic transformation methods do influence performance, but they are still capable of combining the knowledge from LLMs with RSs. Furthermore, the performance order of KAR, KAR-wavg, and KAR-last indicates that the useful information for RS is distributed across the representation of various tokens and is not biased toward the end of the sentence. Finally, the results of KAR, KAR-MoE, and KAR-MLP across different backbone models also confirm the finding in Section 5.3.2 that complex transformation structures can better exploit representations generated by the knowledge encoder.

5.4 Efficiency Study (RQ5)

To quantify the actual time complexity of KAR, we compare the inference time of KAR based on DIN with the API of an LLM and the base DIN model in Table 5. For **LLM API**, we follow the zero-shot user rating prediction in [22] that provides the user viewing history and ratings as prompt and invokes a mainstream AI chatbot API to predict user ratings on candidate items. Since this approach does not allow setting a batch size, the table presents the average

Table 5: The comparison of inference time.

Model	Inference time (s)
LLM API	5.54
$\text{KAR}_{w/\ apt}$	8.08×10^{-2}
$\text{KAR}_{w/o\ apt}$	6.64×10^{-3}
base	6.42×10^{-3}

response time per sample. For KAR, we evaluate the two acceleration strategies as introduced in Section 4.5: $\textbf{KAR}_{w/\ apt}$, where the adaptor participates in the inference stage, and $\textbf{KAR}_{w/o\ apt}$, 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. We present the average inference time per batch in Table 5, from which we draw the following conclusions.

Firstly, adopting a large-scale LLM, such as calling an API, for direct inference is not feasible for RSs due to its large computational latency. The response latency of LLM API is 5-6 seconds, which does not meet the real-time requirement of RSs typically demanding a response latency of within 100ms. 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.*, $\text{KAR}_{w/o\ apt}$, the actual inference time is nearly the same with that of the backbone model. This demonstrates the effectiveness of our proposed KAR and acceleration strategies.

6 CONCLUSION

Our work presents KAR, a framework for effectively incorporating the 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 the 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 directly used to enhance the performance of any recommendation model. Additionally, efficient inference is achieved through preprocessing and prestoring the LLM knowledge. The framework demonstrates superior performance compared to the state-of-the-art methods and is compatible with various recommendation algorithms.

REFERENCES

[1] 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).

[2] Wei Chen, Wynne Hsu, and Mong Li Lee. 2013. Making recommendations from multiple domains. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 892–900.

[3] 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).

[4] 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).

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

[6] 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.

[7] 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).

[8] 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).

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] 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.

[16] 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.

[17] 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).

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

[19] 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.

[20] 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.

[21] 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.

[22] 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).

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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).

[29] Junyang Lin, Rui Men, An Yang, Chang Zhou, Ming Ding, Yichang Zhang, Peng Wang, Ang Wang, Le Jiang, Xianyan Jia, et al. 2021. M6: A chinese multimodal pretrainer. *arXiv preprint arXiv:2103.00823* (2021).

[30] 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).

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

[32] 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]

[33] 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]

[34] 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.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

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

[38] 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.

[39] 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.

[40] 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).

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

[42] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The world wide web conference*. 2000–2010.

[43] 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.

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

[45] 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.

[46] 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.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903* (2022).

[48] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *arXiv preprint arXiv:2305.07001* (2023).

[49] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep Learning for Click-Through Rate Estimation *(IJCAI '21)*.

[50] Yuhui Zhang, HAO DING, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language Models as Recommender Systems: Evaluations and Limitations. In *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*.

[51] 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).

[52] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.

[53] 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.

[54] 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.