Can We Integrate Items into Models? Knowledge Editing to Align LLMs with Product Catalogs

Ahmadou Wagne[1,*], Julia Neidhardt[1]

[1]Christian Doppler Laboratory for Recommender Systems, TU Wien, Vienna, Austria

Abstract

This study explores the potential of knowledge editing techniques to enhance Large Language Models (LLMs) for Conversational Recommender Systems (CRS). While LLMs like GPT, Llama, and Gemini have advanced conversational capabilities, they face challenges in representing dynamic, real-world item catalogs, often leading to inaccuracies and hallucinations in recommendations. This research preliminarily investigates whether knowledge editing can address these limitations by updating the internal knowledge of LLMs, thereby improving the accuracy of product information without full model retraining. Using the open-source Llama2 model, we apply two knowledge editing methods (GRACE and r-ROME) on a dataset of notebook listings. Our findings demonstrate improvements in the model's ability to accurately represent product features, with r-ROME achieving the highest gain, while not decreasing model efficiency. The study highlights the perspective of utilizing knowledge editing to enhance CRS and suggests future work to explore broader applications and impacts on recommender systems performance.

Keywords

knowledge editing, large language models, conversational recommender systems

1. Introduction

Large Language Models (LLMs) have significantly impacted the conversational capabilities of systems, offering enhanced contextual understanding and world knowledge that support various application scenarios, including Conversational Recommender Systems (CRS). With the rapid evolution of LLMs and the frequent release of new models from model families such as GPT[1], LLama[2] or Gemini[3], CRS can benefit from advanced reasoning capabilities to support e.g. multi-turn dialogue understanding or preference elicitation in order to provide more personalized recommendations as well as clearer and more reliable explanations during conversations.

However, LLM-based recommender systems face challenges in accurately representing items, particularly in real-world scenarios where item catalogs are finite and dynamic. This limitation presents significant challenges for LLM-based CRS, dealing with language-based preference representations that offer various benefits for both users and systems [1]. Moreover, issues such as hallucination [2, 3, 4] occur when relying on the internal representations of products within LLMs, posing particular challenges for text-based generative recommendations [5, 6]. Additionally, products might not be exposed to models during training, further complicating the process.

To address these challenges, the paradigm of generative recommendations [7] has proposed various methods for representing items and knowledge associated with them. These include encoding items as IDs [8] to ensure the recommendation of factual items or utilizing graph representations for knowledge augmentation [9, 10]. Such approaches often require additional retrieval-augmented generation (RAG) components [11], which search large item catalogs at inference time. For use cases such as explainable

Sixth Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop @ RecSys 2024, September 14–18 2024, Bari, Italy.

*Corresponding author.

ahmadou.wagne@tuwien.ac.at (A. Wagne); julia.neidhardt@tuwien.ac.at (J. Neidhardt)
0009-0009-9314-206X (A. Wagne); 0000-0001-7184-1841 (J. Neidhardt)

[1]https://platform.openai.com/docs/models
[2]https://llama.meta.com/docs/model-cards-and-prompt-formats
[3]https://deepmind.google/technologies/gemini/

recommendations, systems would benefit from an accurate representation of knowledge about items inherent to the model.

Furthermore, some proposed methods rely on proprietary models that require sharing data with model providers. In many scenarios, this is not feasible due to concerns about disclosing sensitive data about items or users to third parties. As a result, it is desirable to work with open-source LLMs that can be hosted locally and edited internally. In this work, we conduct experiments with the open-source model Llama2 [12], provided by Meta.

When aiming to achieve factual knowledge about products within a model, retraining the model is not a viable solution for many real-world scenarios due to the associated high costs. To tackle these issues, the field of knowledge editing has emerged. The goal of knowledge editing is to develop effective and efficient methods to alter the knowledge internal to LLMs by introducing new facts or updating outdated ones [13] without requiring complete retraining of the model. This approach holds potential for integrating items and their metadata directly into an existing model, aligning it with the set of items available and their associated features in live systems.

While knowledge editing is still an emerging field, this preliminary study investigates techniques to incorporate knowledge about specific products from an existing item catalog into the model. By editing the knowledge corresponding to item features, we aim to enable the model to provide accurate and up-to-date information about our products. This holds potential for improved item and knowledge representation in LLMs, as knowledge editing has already shown success in other contexts, such as detoxifying LLMs [14].

In the context of recommender systems, knowledge editing has not yet been applied to represent item knowledge. In this study, we conduct early experiments to determine whether dynamic knowledge about items can be updated without resorting to external retrieval and to explore potential implications for CRS in the future. Although various benchmarks exist for the general task of knowledge editing, we focus on applied experiments.

To this end, we use a sample product dataset containing notebook listings from the price comparison platform Geizhals[4]. In the tech domain, accurate technical specifications are essential for providing meaningful recommendations or explanations. We aim to investigate whether knowledge editing techniques can improve the accuracy of factual information about product features returned by a model. Hereby, feature values can range from numeric to textual features.

We conduct experiments to assess the capabilities of aligning LLMs with available product catalogs, to determine whether this approach offers a viable solution for future recommender systems research. In this study, we primarily measure the accuracy of the LLM's output, with future research needing to explore the impact on overall LLM performance, as well as the generalization and portability of edits and their implications on recommender systems.

We first test the accuracy of an LLM in providing correct values for certain features out-of-the-box, as the model may already be aware of certain products that were part of the training data. We then compare this to an edited version of the model, which we manipulate by integrating ground truth data from the product catalog.
Our research provides the following contributions:

- Application of knowledge editing methods to improve the accuracy of product feature representations in LLM outputs, specifically focusing on aligning models with real-world, dynamic item catalogs for use in CRS.
- Evaluation and comparison of feature representation accuracy of an unedited and edited LLM, using product data from a notebook listings dataset.
- Evaluation of two knowledge editing techniques, GRACE and r-ROME, assessing their effectiveness in enhancing item feature representations of LLMs. Including a comparison of the improvement in feature accuracy, inference and editing efficiency in order to inspect trade-offs for the application in live systems.

[4]https://geizhals.at/

In the following, we present related work on both LLMs in recommender systems and the general task of knowledge editing in Section 2. Section 3 provides essential theoretical information on the utilized framework as well as the editing techniques applied. In Section 4, we describe the experimental setup and the evaluation criteria for our study. Section 5 presents the data, experiment conduction, and results obtained. Finally, we reflect on the outcomes of this work and highlight potential future research topics in Section 6.

2. Related Work

2.1. LLMs & Recommender Systems

Following the ongoing research on LLMs, many scholars have proposed recommender systems that exploit the generative, reasoning, and generalization capabilities these models offer [5, 15, 16, 17]. General trends in this research area tend towards employing LLMs to unify various components of recommender systems, such as rating predictions, sequential recommendations, explanation generation, user profile construction, or dialogue management for conversational recommender systems (CRS), instead of separating the process into a modular recommendation pipeline [7, 18]. These systems often require specific fine-tuning or pre-training, which is feasible by using open-source models.

However, many other preliminary studies that employ LLMs as recommender systems are based on proprietary models like ChatGPT [16, 15, 19, 20]. These approaches pose challenges in the area of privacy, both for user and product data, due to the closed-source nature of such models, which also limits the ability to manipulate their internal structures. Furthermore, systems using LLMs to generate items encounter additional difficulties, particularly when generating long textual descriptions or news articles. These challenges include the risk of hallucination, where the model may produce inaccurate or fabricated information, as well as the incongruence of generated content with the factual item catalog. In these cases, there would be a need for an intermediate step, where a database is queried to verify or enhance the generated content.

2.2. Knowledge Editing

To alleviate the necessity of this retrieval or retraining models for every new piece of information, we want to investigate the concept of knowledge editing. Knowledge, in this context, can be understood as the awareness and understanding of facts acquired through experience, conceptualized as an "epistemic contact with reality" [13]. Knowledge editing treats knowledge as a set of atomic elements, representing specific facts or pieces of information. The primary objective is to efficiently update these representations within LLMs without requiring complete model retraining. The ideal outcome is a modified LLM behavior that accurately reflects the updated knowledge while preserving the model's overall performance and ensuring that unrelated outputs remain unaffected.

The process involves defining an edit descriptor z_e, often represented as a pair $[x_e; y_e]$, where x_e specifies the input that triggers the output knowledge to be edited and y_e defines the updated fact. This process is described as editing a base model f_{base} to produce an edited model f_{edit}, such that $f_{\text{edit}}(x_e) = y_e$ [21]. For example, if the knowledge about the current European football champion should be updated from Italy to Spain after the 2024 tournament, a model trained before this event would not reflect this fact. Knowledge editing aims to address this by updating the model to produce the correct output for queries relevant to this changed fact.

Knowledge editing tasks can be categorized into three types: updating, inserting, or deleting knowledge within models [22]. Yao et. al [23] divide methods for knowledge editing into two groups on the highest level. The first one involves directly locating and altering specific layers, neurons, embeddings, or other internal structures of the model. The second involves using auxiliary structures that process or control the output of f_{base} while preserving the original parameters.

Memory-based methods, such as SERAC [21] and IKE [24], fall into the latter group. These methods employ counterfactual models or perform in-context editing. Other methods like GRACE [25] or

CaliNET [26] only add additional parameters to control the output. On the other hand, methods that directly modify the model's parameters include locate-then-edit approaches such as ROME [27], r-ROME [28], or MEMIT [29]. These approaches involve identifying specific parameters, such as neurons, that are associated with particular knowledge and modifying them accordingly. Additionally, there are meta-learning approaches, such as MEND [30], which utilize a hypernetwork that learns the Δ to convert the gradients of fine-tuned models according to the desired edit.

Furthermore, edits can be applied sequentially or singularly, depending on whether they are performed consecutively or as isolated updates. The effectiveness of knowledge edits is typically evaluated using several key measures. *Edit success* assesses the reliability and generalization of the edited model, ensuring that f_{edit} provides the correct output for x_e and continues to do so for similar inputs. *Portability* measures the correct application of the edited knowledge across related queries, such as reasoning over the updated fact. For example, after updating the European football champion, the model should also correctly identify the coach of the new champion. Finally, *locality* evaluates the retention of unrelated knowledge, ensuring that the overall model performance does not degrade [22]. These further evaluation metrics however need tailored locality/portability prompts for the desired edits, which have to be meticulously curated.

Additionally, it is important to consider the efficiency of both the edits and the inference process, particularly when comparing the performance of edited models to RAG methods. In this work, we primarily focus on the reliability of edits, investigating whether we can even successfully update knowledge, such as product features, within models. The broader impact of these edits on the general performance of models remains an underexplored area of research.

3. Background

3.1. EasyEdit Framework

To perform the knowledge edits, we utilize the EasyEdit framework, proposed by Wang et al. [31]. EasyEdit offers a variety of knowledge editing methods compatible with various open-source models to conduct comparative experiments. It provides a shared editor class that processes the input for different types of editing methods (memory-based, locate-then-edit and meta-learning) and models to facilitate unified editing across diverse architectures. It also includes the implementations of the methods following the code provided by the original authors utilizing their optimized parameter settings. The minimum required input is structured as pairs of prompts and targets like:

- Prompt: "Who is the president of the United States?"
- Target: "Joe Biden"

The model output is limited to the token length of the target for the evaluation to facilitate comparison to the ground truth. The framework furthermore provides evaluation benchmarks on various data sets for the general knowledge editing task and information about editing efficiency and hardware requirements to perform certain edits. The evaluation of edits is handled by the framework in a way that metrics are calculated for the output of the model prior to and after the edit in order to determine the success of every singular edit. The metric we focus on in our work, edit success, is calculated differently for varying methods, which is further elaborated in Section 4.

3.2. Applied Knowledge Editing techniques

To highlight different types of editing techniques, we apply both GRACE [25], which preserves the initial parameters and r-ROME [28], a locate-then-edit approach manipulating model weights that improves upon ROME [27] for sequential edits.

General Retrieval Adaptors for Continual Editing (GRACE), as proposed by Hartvigsen et al. [25], is designed to perform sequential edits without degrading the overall model performance. GRACE achieves this by adding a wrapper layer to an existing model that consists of a discrete codebook and a deferral mechanism. The codebook stores a tuple of keys, values, and deferral radii that document the edits. Keys represent activations at a specific model layer in response to a particular prompt x_e. These activations are recorded from the original, unedited model and stored in the codebook. The values correspond to minimal gradient updates that lead the model to generate the target y_e given x_e and are learned during editing. They represent the model's modified behavior when a particular key is triggered by the input. The deferral radius determines a threshold when the value of an edit should be applied during inference, based on the similarity between a triggered activation and the available keys in the codebook. The deferral mechanism triggers after the identified layer, performing a search between the input and the codebook to retrieve the closest key. This mechanism ensures that edits are only applied when the input is sufficiently close to a previously stored entry, allowing the model to fall back to its original behavior when the input does not match known edits. If the input is within the radius, the learned value stored in the codebook is used during inference to generate the output.

The second method, Rebuilding Rank-One Model Editing (r-ROME), addresses issues in the basic ROME implementation that can cause the model to fail during sequential edits. ROME is a parametric method that uses causal mediation analysis [32, 33] to trace the strongest activations responsible for mediating a specific fact about a subject [27] within a language model.

Research has shown that certain mid-layer Multi-Layer Perceptrons are key locations for storing factual knowledge. In ROME, these layers are modified to store a key-value pair, where the key represents an input x_e that activates knowledge about a defined subject, and the value represents the updated fact y_e. This change is implemented via a rank-one matrix update, which alters the model's weights in a targeted way to minimize interference with other parts of the model, allowing for precise, singular edits without disrupting unrelated associations. The edited output is therefore specifically tied to an identified subject that a fact is about. This method performs well for singular edits. R-ROME identifies and addresses irregularities in these updates that can cause failures in subsequent edits, due to asymmetries in the key-value pairs in the original implementation.

4. Methods

In this section, we describe the setup of the experiments conducted in this work. The primary objective is to process product data into a set of $[x_e; y_e]$ pairs, where x_e is a generated prompt corresponding to the desired output, and y_e represents the property to be edited (i.e. the target). Additionally, we extract the subject to which y_e should correspond (i.e. the product name). This information is needed for the r-ROME method, as it utilizes the subject to trace corresponding activations. The experiments focus on a subset of notebook properties and the goal is to input the exact model name of a notebook as a subject and accurately represent its features in our model.

Given our setup and the need for integration with most available methods, we chose Llama2 as the LLM to be edited, as prior work already optimized a variety of knowledge editing methods for this model on benchmark datasets. Specifically, we used the 7B chat model version available on Hugging Face[5]. This choice was made because it is feasible to run the model on a single GPU, which is necessary as some of the methods used do not support parallelism by design. All experiments are conducted on a single NVIDIA A40 GPU. The chat model is selected due to its intended use in a CRS, where we seek conversations as interactions between users and the system.

The parameters for all methods were set according to the default configurations provided in the EasyEdit framework[6][7], as corresponding to the optimal setup identified in the original publications regarding layers to be edited e.g. To demonstrate the capabilities of different types of methods, we utilize the two approaches priorly explained GRACE and r-ROME representing contrasting editing paradigms.

Figure 1 illustrates the experiment setup. For each method, we first extract a pair consisting of a

[5]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[6]https://github.com/zjunlp/EasyEdit/blob/main/hparams/GRACE/llama-7B.yaml
[7]https://github.com/zjunlp/EasyEdit/blob/main/hparams/R-ROME/llama-7b.yaml

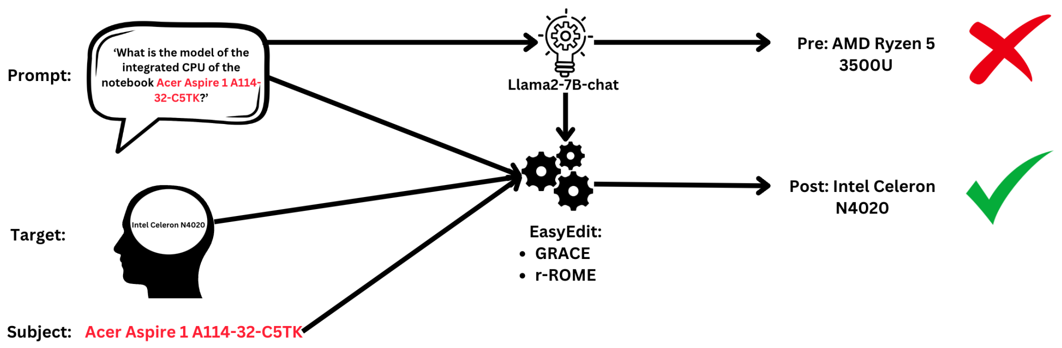
Prompt:
'What is the model of the integrated CPU of the notebook Acer Aspire 1 A114-32-C5TK?'
Llama2-7B-chat
Pre: AMD Ryzen 5 3500U
Target:
Intel Celeron N4020
EasyEdit:
• GRACE
• r-ROME
Post: Intel Celeron N4020
Subject: Acer Aspire 1 A114-32-C5TK

Figure 1: Workflow of the editing process for a single property, taking a prompt and target pair (as well as a subject for r-ROME) as input for the model to edit and comparing the output to the unedited model.

prompt and target and add a subject for r-ROME, which is then passed to the editor. We experiment with different prompts on the base model to generate accurate output formats and create a template for each property separately. The editor processes the output of the vanilla model for evaluation, then performs sequential edits with the selected model for each product and feature, resulting in $n_{\text{products}} \times n_{\text{features}}$ edits. To measure the quality of the edits, we assess the accuracy of the model's responses to the updated facts before and after the edit. Edit success is calculated differently for GRACE and r-ROME, to align with the methodologies used in the original publications [25, 27]. For GRACE, the output given the prompt is generated as text (limited to the token length of the target) and the proportion of matching tokens from the output and the target are calculated and reported. For r-ROME, the raw logits are returned and compared to the target corresponding to the input. Therefore, we analyze the differences in accuracy before and after the edits to interpret the results, as the metrics are not directly comparable. Additionally, we measure the time required to perform the overall editing, as well as the inference over the whole sample.

5. Experiments & Results

5.1. Data

The data used in our experiments consists of a selection of notebook models released before the knowledge cutoff of Llama2's training in July 2022. The properties selected for this study are a variety of numeric and textual features (in the English language), specifically: CPU, RAM, GPU, Camera Resolution, Weight, Display Size, and SSD.

The distinct values for these features vary, with the number of unique values ranging from 106 for Weight to seven for SSD storage capacities. To ensure the precision of our edits, we filtered the notebooks to include only those with exact model names, such as "Acer Aspire 1 A114-32-C5TK", which represent specific products without variants differing in the selected features. Furthermore, we removed information about product features from the title of the subject (e.g. Lenovo IdeaPad Pro 5 14AHP9, Arctic Grey, Ryzen 7 8845HS, 32GB RAM, 1TB SSD).

The final dataset consists of 335 unique notebooks, each serving as a subject for the knowledge editing process. For each feature, we create prompt templates similar to the one depicted in Figure 1, filling them with the corresponding notebook model, which is also added to the subject list. After removing entries with missing values, we extract the relevant features into a target list.

The final dataset, prepared for the editing process, consists of 2265 tuples, with each tuple representing a unique combination of a prompt, target, and subject. This setup results in 2265 consecutive edits performed during the experiments.

Method	pre_acc	post_acc	edit	inference
No edits	–	–	0	7,078
GRACE	0.022	0.21	22,942	7,000
r-ROME	0.449	0.868	24,953	5,800

Table 1
Results reported for the knowledge editing experiment setups. Pre_acc and post_acc measure the average rewrite accuracy reported by EasyEdit for all performed edits. The edit and inference time are reported in seconds.

5.2. Editing

With the prepared dataset, we conduct sequential editing in our pipeline to process each tuple consecutively. The results of our experiments are presented in Table 1. The reported results include the pre- and post-rewrite accuracies as reported by EasyEdit, along with the time required for all edits and the subsequent inference over all prompts. This assessment aims to evaluate how the editing impacts the efficiency of the modified model.

Initially, we use the Llama2 7B base model on our set of prompts to generate the notebook features. The total time to process the entire set is reported at 7,078s (118 minutes), corresponding to an inference time of 3.12s per sample. In contrast, the GRACE model required 7,000 seconds (116.6 minutes) to complete the full inference after all sequential edits, maintaining an inference time of 3.09s per sample. This finding is unexpected, as the original GRACE paper reported a longer inference time due to the necessity of performing similarity searches within the keys of the codebook.

The editing accuracy reported reflects the average accuracy for each individual edit. It is important to note that prior edits may also influence the pre-edit accuracy, given that the model state after the $n - 1$ sequential edit is used for each edit n. For GRACE, the pre-rewrite accuracy is notably low at 2.2%, which improves substantially to 21% following the presentation of the tuple and its inclusion in the codebook. This indicates that the technique effectively influences the target related to a subject, enabling the model to produce accurate results with updated facts. However, the post-rewrite accuracy remains relatively low, which means that it does not do so consistently. The time required for the edits is reported as 22,942 seconds (382.4 minutes), averaging 10.13s per sample.

R-ROME, on the other hand, demonstrates significantly higher average pre- and post-rewrite accuracies, recorded at 44.9% and 86.8%, respectively. The higher pre-edit accuracy can be partially attributed to the specific measurement method used for this approach, although the impact of sequential editing on unrelated inputs—beyond the scope of our analysis—may also play a role. The post-rewrite accuracy suggests a high level of success in identifying activations and updating the target knowledge of the subjects. The total time for editing with r-ROME is 24,953s, with an average of 11,02s per sample. Inference time for the r-ROME model is 5,800 seconds (96.6 minutes), resulting in the fastest per-sample time of 2.56s, further improving over the baseline model.

6. Conclusion & Future Work

In this work, we introduced a new application for knowledge editing on product data for the use in a CRS. Our preliminary results are promising, indicating that our approach has the potential to update and integrate product knowledge of Llama2.

Our experiments indicate that the r-ROME method achieves great improvements in pre- and post-rewrite accuracies, while the GRACE method showed moderate improvement. However, the poor performance prior to the edits is surprising, as one would expect the LLM to possess prior knowledge about some of the products due to them being limited to listings before the knowledge cutoff. This could indicate that the usage of full product names could be too complex as a subject for the model used. Further experiments with more advanced models could give further insights in the future.

Interestingly, the inference time after edits is shorter than expected for the GRACE model, maintaining a similar per-sample inference time as the base Llama2 7B model. The r-ROME model further reduces inference time, making it the fastest among the models tested.

It is important to note that model editing can be a time-consuming process, as demonstrated by the total editing times reported for both GRACE and r-ROME. Despite this, editing with the selected techniques can be scheduled independently from inference time, allowing updates to be applied without significantly degrading the efficiency of live systems for dynamically evolving product catalogs.

An evident next step will be exploring the impact of these editing techniques on recommender systems. We see potential for future work in investigating additional editing methods, adapting the methods to more recent LLMs or introducing a greater variety of data to assess model performance beyond the knowledge cutoff of models. The effect on multi-hop reasoning for edited facts as well as the generalization capabilities have to be also further investigated. Additionally, there are ongoing challenges in effectively modeling users and interactions within these frameworks, but we see great potential in the method e.g. supporting the development of more accurate explainable recommender systems.

Acknowledgments

The financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

[1] S. Sanner, K. Balog, F. Radlinski, B. Wedin, L. Dixon, Large Language Models are Competitive Near Cold-start Recommenders for Language- and Item-based Preferences, 2023. doi:10.48550/arXiv.2307.14225.

[2] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, Q. V. Do, Y. Xu, P. Fung, A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity, in: J. C. Park, Y. Arase, B. Hu, W. Lu, D. Wijaya, A. Purwarianti, A. A. Krisnadhi (Eds.), Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Nusa Dua, Bali, 2023, pp. 675–718. doi:10.18653/v1/2023.ijcnlp-main.45.

[3] G. Marcus, The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence, 2020. doi:10.48550/arXiv.2002.06177.

[4] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu, A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions, 2023. doi:10.48550/arXiv.2311.05232.

[5] J. Li, W. Zhang, T. Wang, G. Xiong, A. Lu, G. Medioni, GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation, 2023.

[6] Z. Cui, J. Ma, C. Zhou, J. Zhou, H. Yang, M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems, 2022. doi:10.48550/arXiv.2205.08084.

[7] L. Li, Y. Zhang, D. Liu, L. Chen, Large Language Models for Generative Recommendation: A Survey and Visionary Discussions, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 10146–10159.

[8] W. Hua, S. Xu, Y. Ge, Y. Zhang, How to Index Item IDs for Recommendation Foundation Models, in: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, SIGIR-AP '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 195–204. doi:10.1145/3624918.3625339.

[9] Y. Wang, Z. Chu, X. Ouyang, S. Wang, H. Hao, Y. Shen, J. Gu, S. Xue, J. Y. Zhang, Q. Cui, L. Li, J. Zhou, S. Li, Enhancing Recommender Systems with Large Language Model Reasoning Graphs, 2024. doi:10.48550/arXiv.2308.10835.

[10] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, C. Huang, LLMRec: Large Language Models with Graph Augmentation for Recommendation, in: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 806–815. doi:10.1145/3616855.3635853.

[11] D. Di Palma, Retrieval-augmented Recommender System: Enhancing Recommender Systems with Large Language Models, in: Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1369–1373. doi:10.1145/3604915.3608889.

[12] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023. doi:10.48550/arXiv.2307.09288.

[13] M. Wang, Y. Yao, Z. Xu, S. Qiao, S. Deng, P. Wang, X. Chen, J.-C. Gu, Y. Jiang, P. Xie, F. Huang, H. Chen, N. Zhang, Knowledge Mechanisms in Large Language Models: A Survey and Perspective, 2024. doi:10.48550/arXiv.2407.15017.

[14] M. Wang, N. Zhang, Z. Xu, Z. Xi, S. Deng, Y. Yao, Q. Zhang, L. Yang, J. Wang, H. Chen, Detoxifying Large Language Models via Knowledge Editing, 2024. doi:10.48550/arXiv.2403.14472.

[15] L. Friedman, S. Ahuja, D. Allen, Z. Tan, H. Sidahmed, C. Long, J. Xie, G. Schubiner, A. Patel, H. Lara, B. Chu, Z. Chen, M. Tiwari, Leveraging Large Language Models in Conversational Recommender Systems, 2023. doi:10.48550/arXiv.2305.07961.

[16] J. Liu, C. Liu, P. Zhou, R. Lv, K. Zhou, Y. Zhang, Is ChatGPT a Good Recommender? A Preliminary Study, 2023. doi:10.48550/arXiv.2304.10149.

[17] J. Ji, Z. Li, S. Xu, W. Hua, Y. Ge, J. Tan, Y. Zhang, GenRec: Large Language Model for Generative Recommendation, 2023. doi:10.48550/arXiv.2307.00457.

[18] S. Geng, S. Liu, Z. Fu, Y. Ge, Y. Zhang, Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5), in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 299–315. doi:10.1145/3523227.3546767.

[19] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, J. Zhang, Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System (2023). doi:10.48550/ARXIV.2303.14524.

[20] S. Dai, N. Shao, H. Zhao, W. Yu, Z. Si, C. Xu, Z. Sun, X. Zhang, J. Xu, Uncovering ChatGPT's Capabilities in Recommender Systems, in: Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1126–1132. doi:10.1145/3604915.3610646.

[21] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, C. Finn, Memory-based model editing at scale, in: International Conference on Machine Learning, PMLR, 2022, pp. 15817–15831.

[22] N. Zhang, Y. Yao, B. Tian, P. Wang, S. Deng, M. Wang, Z. Xi, S. Mao, J. Zhang, Y. Ni, S. Cheng, Z. Xu, X. Xu, J.-C. Gu, Y. Jiang, P. Xie, F. Huang, L. Liang, Z. Zhang, X. Zhu, J. Zhou, H. Chen, A Comprehensive Study of Knowledge Editing for Large Language Models, 2024. doi:10.48550/arXiv.2401.01286.

[23] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, N. Zhang, Editing Large Language Models: Problems, Methods, and Opportunities, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 10222–10240. doi:10.18653/v1/2023.emnlp-main.632.

[24] C. Zheng, L. Li, Q. Dong, Y. Fan, Z. Wu, J. Xu, B. Chang, Can We Edit Factual Knowledge by In-Context Learning?, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 4862–4876. doi:10.18653/v1/2023.emnlp-main.296.

[25] T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim, M. Ghassemi, Aging with grace: Lifelong model editing with discrete key-value adaptors, in: Advances in Neural Information Processing Systems, 2023.

[26] Q. Dong, D. Dai, Y. Song, J. Xu, Z. Sui, L. Li, Calibrating Factual Knowledge in Pretrained Language Models, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 5937–5947. doi:10.18653/v1/2022.findings-emnlp.438.

[27] K. Meng, D. Bau, A. Andonian, Y. Belinkov, Locating and editing factual associations in GPT, Advances in Neural Information Processing Systems 35 (2022).

[28] A. Gupta, G. Anumanchipalli, Rebuilding rome: Resolving model collapse during sequential model editing (2024).

[29] K. Meng, A. Sen Sharma, A. Andonian, Y. Belinkov, D. Bau, Mass editing memory in a transformer, The Eleventh International Conference on Learning Representations (ICLR) (2023).

[30] E. Mitchell, C. Lin, A. Bosselut, C. Finn, C. D. Manning, Fast model editing at scale, in: International Conference on Learning Representations, 2022.

[31] P. Wang, N. Zhang, B. Tian, Z. Xi, Y. Yao, Z. Xu, M. Wang, S. Mao, X. Wang, S. Cheng, K. Liu, Y. Ni, G. Zheng, H. Chen, EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models, in: Y. Cao, Y. Feng, D. Xiong (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 82–93.

[32] J. Pearl, Direct and Indirect Effects, in: Probabilistic and Causal Inference: The Works of Judea Pearl, volume 36, 1 ed., Association for Computing Machinery, New York, NY, USA, 2022, pp. 373–392.

[33] J. Vig, S. Gehrmann, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, S. Shieber, Investigating Gender Bias in Language Models Using Causal Mediation Analysis, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 12388–12401.