New Generation Conversational Recommendation System Powered by Large Language Model Agent

Tianyi Li 23020241154414[1*], Shaowen Ye 36920241153272[2], Guanxi Lin 36920241153233[2], Yu Zhang 36920241153282[2],

[1]School of Informatics Xiamen University
[2]Xiamen University Artificial Intelligence Research Institute
[*]Team Leader

Abstract

Recommender systems (RS) play a vital role in digital applications by providing personalized recommendations based on user preferences and behavior. With the rise of large language models(LLMs), conversational recommender systems (CRS) have developed fastly. However, integrating LLMs into CRS introduces challenges, including limited context length, long inference time, and insufficient integration of collaborative filtering (CF) knowledges. This paper proposes a new generation of CRS powered by LLMs(LLMRecAgent) to address these limitations.This architecture adopts a two-stage retrieval-and-ranking recommendation paradigm to effectively reduce the context length input to the large model, meanwhile reducing the computing time. We conducted experiments on three datasets and the results outperform the baseline models, demonstrating the effectiveness of our approach. This hybrid approach significantly improves user experience and recommendation effectiveness in real-time interactions.

Introduction

Recommender systems (RS) have become indispensable technologies in the fields of e-commerce(Alamdari et al. 2020), online entertainment, and other digital applications. By analyzing user preferences, historical behavior data, and contextual information, recommender systems can provide personalized recommendations, greatly enhancing user experience and satisfaction. In the early stages, traditional recommendation systems relied on collaborative filtering(Wang et al. 2017) and content-based recommendation algorithms. With the development of deep learning, they have gradually evolved into more complex hybrid models.

In recent years, users have increasingly preferred to interact with systems through natural language dialogue interfaces, which has constantly raised the design requirements for recommender systems. Compared to simple recommendation lists, users expect systems to understand and respond to their complex and changing needs in real time. This trend has driven the development of conversational recommender systems (CRS), transforming recommender systems from mere information delivery tools into intelligent dialogue partners capable of understanding user intents and engaging in multi-turn interactions.

Although large language models (LLMs) have shown great potential in recommender systems, their application still faces several challenges and limitations. First, LLMs are typically pretrained on vast amounts of general text data from the internet, which gives them broad cross-domain understanding and rich world knowledge. However, this generalization can also mean that they may fall short when dealing with fine-grained domain-specific knowledge, especially in highly specialized and complex areas. For instance, in fields like healthcare, law, or finance, user behavior, and needs are often highly specialized, and the precise knowledge required in these domains may not be adequately reflected in the LLMs' pre-trained data, resulting in poor performance when capturing and processing such specific information.

Large language models (LLMs), such as GPT-3(Brown 2020) and PaLM(Chowdhery et al. 2023), have made significant breakthroughs in the field of natural language generation. As users' demand for personalization and natural interaction continues to increase, integrating LLMs into recommender systems have become a promising solution. This integration not only improves the user experience but also enhances the quality of recommendations. For example, through dialogue with LLMs, the system can better understand users' deep-level needs and provide more targeted and persuasive recommendations. Such interaction not only makes users feel more fluent in conversing with the system but also reduces information asymmetry issues during the recommendation process, enhancing user trust and reliance on the system.

Secondly, LLMs face difficulties in retaining user preference information over multi-turn conversations. In conversational recommender systems, users' preferences are often dynamic, becoming clearer as the dialogue progresses. However, LLMs lack effective mechanisms to retain and track user preference information throughout the conversation, especially in extended multi-turn dialogues. This loss of information leads to a reduction in the level of personalization in recommendations, negatively impacting the user experience. Additionally, since LLMs must reprocess the dialogue context each time recommendations are generated, this can result in unnecessary redundant computations and inefficient recommendation results.

Third, LLMs have limitations in incorporating traditional collaborative filtering (CF) information. CF is one of the core methods in recommender systems, generating personalized recommendations by analyzing similarities in user behaviors. However, LLMs, while excelling in natural language processing, do not naturally handle the logic of collaborative filtering. They rely more on language generation and semantic understanding, failing to effectively integrate interaction data between users, which limits the improvement of recommendation performance.

To address these challenges, we aim to propose a conversational recommendation system powered by large language model agent.

First, the system utilizes external tools to compensate for LLMs' deficiencies in domain-specific knowledge. By integrating domain-specific models or knowledge bases, the system can dynamically acquire and apply specialized knowledge, enhancing the accuracy of recommendations.

Secondly, the system introduces a memory mechanism to store and track user preference information. This mechanism allows the system to record and update users' historical behaviors and conversational content throughout multi-turn dialogues, ensuring that the model can offer personalized recommendations based on the user's specific needs in subsequent conversations. This not only improves the user experience but also reduces the burden on users to repeat the same information in each interaction. Additionally, the memory mechanism ensures the continuity of user information across long-term interactions, enabling the system to deliver more customized services.

Lastly, this paper proposes embedding collaborative filtering outputs into the LLMs' prompts, thus combining the strengths of LLMs' language understanding capabilities with the advantages of collaborative filtering. By incorporating collaborative filtering results into the contextual input for LLMs, the system can generate recommendations that not only reflect the user's current conversational needs but also leverage similar users' behavior patterns, resulting in more personalized and relevant recommendations. Collaborative filtering results serve as an additional context that helps the LLM consider both the user's historical preferences and the behaviors of others, producing recommendations that are better aligned with the user's individual needs.

Related Works

In recent years, with the rapid development of LLMs in the field of natural language processing, an increasing number of researchers have started to explore their potential applications in recommendation systems. LLMs' strong capabilities in semantic understanding, feature extraction, and interactivity can effectively enhance the performance of recommendation systems. In the field of recommendation systems, the applications of LLMs can currently be divided into two categories: traditional recommendation systems and conversational recommendation systems.

Traditional Recommender System

In traditional recommendation systems, the recommendation mechanism primarily relies on users' past behavior data to predict their preferences for specific items. In this context, LLMs are usually integrated into the recommender systems as feature extractors.(Zhao et al. 2024) By analyzing the contextual information of items and users' historical behavior, embedding vectors are generated to represent users' interests and item characteristics, thereby improving the accuracy and personalization level of recommendations. This approach effectively combines the natural language understanding capability of LLMs with traditional recommendation mechanisms, further enriching the feature dimensions and user profiles of the recommender systems.(Chen, Chen, and Wang 2015)

Additionally, some methods directly utilize LLMs as recommendation engines, which means the input sequence can include users' profiles, item descriptions, and interaction history, among other information(Li et al. 2018). The model generates a list of recommended items directly after receiving the input. This method not only makes the recommendation system more flexible but also shows great potential in generating recommendation systems(Lu et al. 2015). However, such applications also face challenges in terms of the high computational resource demands of LLMs and real-time optimization.

Conversational Recommender System

Compared to traditional recommender systems, conversational recommender systems (CRS) have higher requirements in terms of interaction experience and personalized demand fulfillment. CRS not only needs to accurately capture user preferences but also dynamically generate interactive content that meets their needs .Despite advancements, current conversational recommender systems often rely on static rules or predefined models that lack adaptability to dynamic user preferences and complex dialogue contexts. This limitation means CRS struggles to adjust in real-time to diverse user requests and preferences as they evolve during the conversation. As a result, they often fall short of delivering highly personalized recommendations that align with the nuanced and evolving needs of individual users. However, these methods cannot achieve accurate personalized recommendations. In this study, we will focus on exploring the application of LLMs in conversational recommender systems, aiming to enhance the system's language understanding and generation capabilities in multi-turn interactions, making it more flexible and intelligent in complex dialogue scenarios.

Enhancing LLMs

The scaling up of parameters and data has led to significant advancements in the capabilities of large language models (LLMs), including in-context learning(Rubin, Herzig, and Berant 2021), instruction following(Zhou et al. 2024), and improved planning and reasoning abilities. In recommender systems, the application of LLMs is rapidly emerging as a growing trend.

As models exhibit emergent intelligence, researchers have begun exploring the potential of LLMs as autonomous agents(Jin et al. 2024), enhanced with memory modules, planning capabilities, and tool-using skills. For instance, recent studies have integrated external memory into LLMs,

enabling them to grow and learn over time. In terms of planning, methods like Chain-of-Thought (CoT) reasoning and ReAct(Yao et al. 2022)promote step-by-step reasoning, while approaches like Tree-of-Thought (ToT) and Graph-of-Thought (GoT)(Besta et al. 2024) introduce multi-path reasoning to ensure consistency and accuracy. Techniques like Self-Refine (Madaan et al. 2024) and Reflexion(Shinn et al. 2024) help LLMs reflect on mistakes to enhance their future problem-solving abilities.

To further extend domain-specific skills, several studies have explored guiding LLMs to use external tools, such as web search engines, mathematical tools(Shinn et al. 2024), code interpreters, and visual models(Liu et al. 2024). To the best of our knowledge, this paper is the first to explore the LLM + tools paradigm within the field of recommender systems.

Methods

In the most classic recommendation task—sequential recommendation—fine-tuning pre-trained large models for generative recommendation has become a very popular and cutting-edge method. This approach leverages the powerful capabilities of large models to generate recommended items that match users' interests, significantly improving the effectiveness and accuracy of the recommendation system. However, directly using large models to generate recommended items faces several key challenges:

First, user historical interaction sequences are typically very long, resulting in excessively large item sequences being input into the large model. This not only makes the input difficult for the model to handle, but may also lead to information loss or computational overload during the model's inference process. Secondly, the inference speed of large models is relatively slow, particularly when faced with a large number of user requests, making it difficult to respond quickly and thereby impacting user experience and system responsiveness.

To address these issues, we propose LLMRecAgent. This architecture adopts a two-stage retrieval-and-ranking recommendation paradigm to effectively reduce the context length input to the large model. First, a retrieval module filters out a subset of items most relevant to the user's historical interactions from a large pool of candidate items. Then, the large language model ranks and generates recommendations based on this refined subset. This two-stage design not only reduces the length of the item sequence input to the large model, avoiding the computational burden caused by excessively long contexts, but also improves inference speed while ensuring high recommendation quality.

Overall Architecture

The overall process is divided into six stages:

User Request: The user initiates a recommendation request through natural language interaction, such as "Can you recommend me a movie?". At this stage, the large language model (LLM) agent receives and processes the user's request. This step serves as the entry point for generating personalized recommendations by interpreting the user's intent.

Recall Stage: After identifying relevant item categories (e.g., "movies") from the user's request, the LLM agent queries a large-scale item database to recall all items related to the user's needs. This step ensures that the system retrieves a comprehensive set of candidates as the basis for subsequent processing.

Retrieval Stage: The recalled candidate set is input into the retrieval module, which utilizes an efficient pre-trained sequential recommendation model, such as SASRec, to further filter and refine the candidate items. SASRec leverages the user's historical interaction data to quickly generate a concise subset of candidate items. Compared to the LLM, SASRec offers the advantage of faster response times, making it well-suited for real-time recommendations. Additionally, the retrieval module incorporates user interaction history to construct prompts, which are used as critical contextual inputs for the next stage.

Ranking Stage: The refined candidate item subset, along with the user's historical interaction data, is structured into prompts and input into the LLM for ranking. The LLM performs deep semantic understanding of the user's historical behavior and comprehensive semantic modeling of the candidate items. By capturing fine-grained correlations between user needs and candidate items, the LLM re-ranks the items to determine their relevance and priority.

Recommendation Generation: The highly ranked item subset is used by the LLM to generate natural language recommendation results. These results are presented to the user in a conversational manner, such as "Sure! I guess you like Star Wars...". This stage not only delivers personalized recommendation content but also enhances the naturalness and interpretability of the system through the use of conversational language.

User Feedback and Iterative Optimization. The user's interactions with the recommended results, such as clicks, ratings, or skips, are recorded and incorporated into the user's historical interaction data. This feedback is fed back into the system to update user profiles and optimize the recall, retrieval, and ranking stages. By incorporating an iterative feedback mechanism, the system dynamically adapts to changes in user preferences, enabling continuous improvement in recommendation quality.

Text Enhanced Retriever

Traditional recommendation systems often rely solely on item ID features while neglecting the rich semantics contained in textual information about items. This limitation may lead to suboptimal performance in terms of recommendation diversity and novelty. To address this issue, this study proposes leveraging item textual information (e.g., titles, descriptions) to enhance the effectiveness of recommendation systems. The specific approach is as follows.

Text Feature Extraction: A text encoder (in this study, distilbert-base-uncased) is used to encode item textual information and extract its semantic features. These features capture the latent semantic relationships between items, thereby compensating for the limitations of relying solely on item ID features.

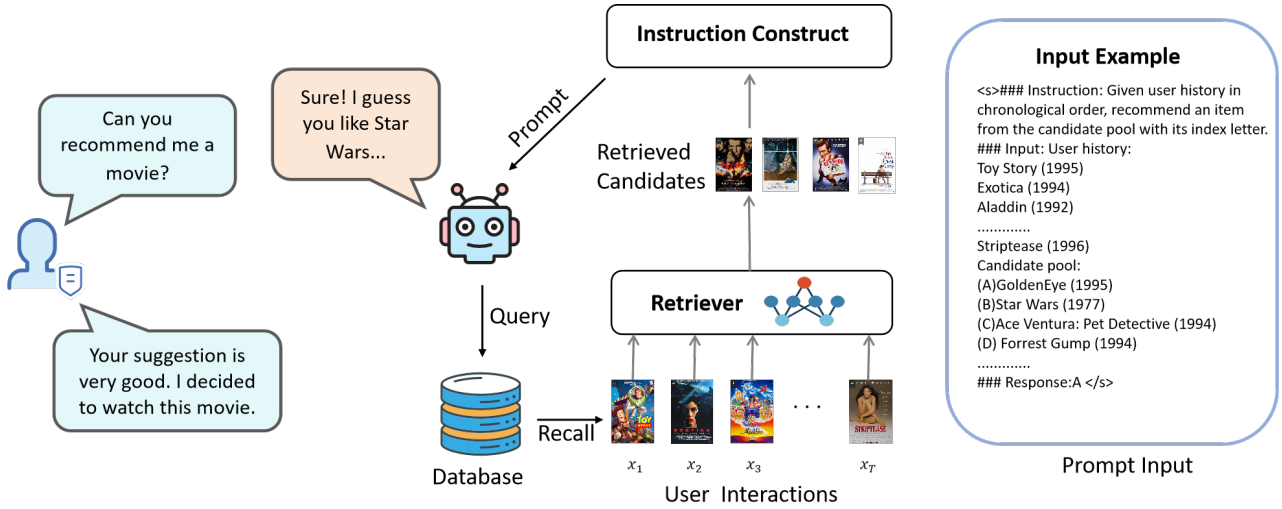

Figure 1: Overall process of Conversational Recommendation System Powered by Large Language Model Agent.

Embedding Layer Initialization: The extracted text features are used to initialize the embedding layer of the retrieval module. This ensures that the retrieval module incorporates textual semantic information from the beginning, providing a richer feature representation for subsequent model training.

Joint Training: During model training, the retrieval module and the embedding layer are jointly optimized. Through backpropagation, both the text embeddings and the parameters of the retrieval module are gradually updated. This process not only integrates the textual features into the retrieval module but also aligns them closely with the recommendation task, further enhancing the retrieval performance.

By incorporating textual information into the recommendation system, the proposed method improves the system's ability to capture semantic relationships between items, enhances its handling of long-tail items and cold-start scenarios, and ultimately boosts overall recommendation performance.

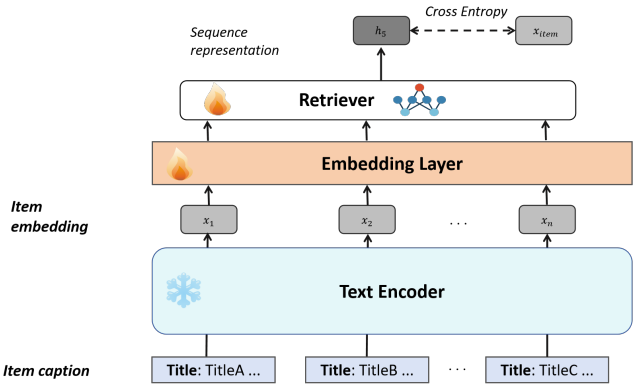

Figure 2: Using the text enhanced retrieval module

LLMRecAgent Pretraining

Since large language models are not inherently trained on recommendation system data, they may struggle to fully capture the fine-grained correlations between user historical interactions and candidate items when performing specific ranking tasks, resulting in suboptimal ranking performance. To address this issue, the large model can be fine-tuned on recommendation system-specific data, enabling it to better adapt to and understand ranking tasks. In this study, we fine-tune the model using constructed prompts as instructions and positive samples as responses, optimizing the model for instruction-based tasks. Additionally, the LoRA(Hu et al. 2021) method is employed to reduce the number of trainable parameters during fine-tuning.

Experiments

Datasets

Our model is evaluated on the following datasets:

- **ML-100k**: A benchmark dataset for movie recommendation with around 100k user-item interactions (Harper and Konstan 2015).
- **Beauty**: A product review dataset from the Amazon website consisting of user feedback on Beauty products (He and McAuley 2016; McAuley et al. 2015).
- **Games**: A video game dataset from Amazon with user reviews and ratings on video game products (He and McAuley 2016; McAuley et al. 2015).

For preprocessing, we follow (Yue et al. 2022; Yang et al. 2023) to construct input sequences in chronological order and iteratively filter users and items that are fewer than 5 interactions (i.e., 5-core). Items without metadata (i.e., title) are also filtered. We report the statistics (i.e., users, items, interactions, sequence length, and dataset density) in Table 2.

	ML-100k					Beauty					Games				
	GRU	BERT	LRU	SAS	SAS-T	GRU	BERT	LRU	SAS	SAS-T	GRU	BERT	LRU	SAS	SAS-T
M@5	0.0276	0.0514	0.0337	0.0447	**0.0523**	0.0271	0.0215	0.0400	0.0433	**0.0485**	0.0432	0.0281	0.0506	0.0535	**0.0600**
N@5	0.0317	0.0604	0.0393	0.0503	**0.0633**	0.0325	0.0249	0.0460	0.0501	**0.0550**	0.0513	0.0330	0.0598	0.0642	**0.0714**
R@5	0.0448	0.0881	0.0560	0.0673	**0.0922**	0.0491	0.0355	0.0644	0.0707	**0.0748**	0.0762	0.0479	0.0880	0.0970	**0.1061**
M@10	0.0321	0.0551	0.0378	0.0508	**0.0586**	0.0301	0.0236	0.0433	0.0470	**0.0528**	0.0488	0.0317	0.0568	0.0604	**0.0671**
N@10	0.0433	0.0689	0.0495	0.0659	**0.0759**	0.0399	0.0302	0.0541	0.0592	**0.0654**	0.0651	0.0419	0.0749	0.0812	**0.0887**
R@10	0.0817	0.1137	0.0881	0.1169	**0.1224**	0.0722	0.0522	0.0897	0.0990	**0.1071**	0.1193	0.0755	0.1347	0.1497	**0.1599**
M@20	0.0362	0.0570	0.0442	0.0560	**0.0633**	0.0324	0.0251	0.0456	0.0495	**0.0528**	0.0525	0.0345	0.0614	0.0652	**0.0723**
N@20	0.0582	0.0762	0.0718	0.0849	**0.0893**	0.0482	0.0358	0.0627	0.0682	**0.0754**	0.0786	0.0519	0.0917	0.0985	**0.1052**
R@20	0.1410	0.1426	0.1746	0.1923	**0.2034**	0.1050	0.0741	0.1241	0.1348	**0.1471**	0.1729	0.1151	0.2015	0.2182	**0.2273**

Table 1: Performance comparison of different models on ML-100k, Beauty, and Games datasets, with the best results marked in bold and second best results underlined.

Datasets	Users	Items	Interact.	Length	Density
ML-100k	610	3,650	100k	147.99	4e-2
Beauty	22,332	12,086	198K	8.87	7e-4
Games	15,264	7,676	148K	9.69	1e-3

Table 2: Dataset statistics after preprocessing.

Baseline Methods

We adopt multiple state-of-the-art sequential recommenders, which include RNN models (i.e. GRU4Rec), transformer-based recommenders (i.e. SASRec, BERT4Rec) and linear recurrence-based LRURec.

- **GRU4Rec**: GRU4Rec is an RNN-based model that leverages GRU modules for sequential recommendation (Hidasi et al. 2016).
- **SASRec**: SASRec adopts unidirectional attention to process input at a sequence level to generate the next items (Kang and McAuley 2018).
- **BERT4Rec**: A bidirectional attention-based recommender model, BERT4Rec is trained via predicting masked items (Sun et al. 2019).
- **LRURec**: An efficient sequential recommender based on linear recurrence.(Yue et al. 2024).

Evaluation

In our evaluation, we follow the leave-one-out strategy and in each data example, we use the last item for testing, the second last item for validation, and the rest items for training. The evaluation metrics are mean reciprocal rank (MRR@k), normalized discounted cumulative gain (NDCG@k) and recall (Recall@k) with $k \in \{5, 10, 20\}$. We save the model with the best validation scores for evaluation (Recall@10 for retrieval and NDCG@10 for ranking), where predictions are ranked against all items in the dataset.

Implementation

We choose Llama2-7b(Touvron et al. 2023) as the rerank model. The retriever models are trained with AdamW optimizer using the learning rate of 0.001 and the maximum epoch of 100. Validation is performed every 500 iterations and early stopping is triggered if validation performance does not improve in 20 consecutive rounds. We used 200 as maximum length for ML-100k and 50 for the other datasets. For our ranker, we use maximum 20 history items and rank the top-20 candidates from the retriever model. The title length is truncated if it exceeds 32 tokens. We adopt QLoRA to quantize the Llama 2-based ranker and adopt 8 as LoRA dimension, 32 as α as well as 0.05 dropout. The LoRA learning rate is 1e-4 with target modules being the Q and V projection matrices. The model is tuned for 1 epoch and validated every 100 iterations. Similarly, the model with the best validation performance is saved for test set evaluation.

Retriver Comprision Results

The table 1 compares the performance of five recommendation models (GRU, BERT, LRU, SAS, and SAS-T) across three datasets: ML-100k, Beauty, and Games. The models are evaluated using multiple metrics. In the experimental results, we can observe that SASRec performs the best. Note that our retrieval module is model-agnostic, meaning we can choose any high-performing and efficient model for the retrieval task. Therefore, in this paper, we select the top-performing SASRec as the retrieval module for text augmentation, and the model after text augmentation is labeled as SAS-T.

Across all three datasets, SAS-T consistently outperforms the other models, achieving the highest scores in most metrics. SAS follows closely behind, often securing second place, while BERT also shows competitive performance, particularly in the ML-100k and Games datasets. GRU and LRU generally perform the weakest, with GRU especially trailing in many categories.

Overall, SAS-T is the most effective model for recommendation tasks, demonstrating strong performance in both

precision and recall across different datasets, while SAS and BERT offer solid alternatives. GRU and LRU are less competitive in comparison.

	ML-100k					
	GRU	BERT	LRU	SAS	SAS-T	LLMRecAgent
M@5	0.1639	0.0887	0.1449	0.1965	<u>0.2022</u>	**0.2184**
N@5	0.1607	0.1032	0.1793	0.2356	<u>0.2498</u>	**0.2693**
R@5	0.2329	0.1477	0.2833	0.3544	<u>0.3879</u>	**0.4227**
M@10	0.1485	0.1054	0.1596	0.2384	<u>0.2477</u>	**0.2623**
N@10	0.1886	0.1435	0.2147	0.3367	<u>0.3508</u>	**0.3766**
R@10	0.3188	0.2720	0.3927	0.6654	<u>0.6972</u>	**0.7560**

Table 3: Recommendation performance on the valid retrieval subset, in which the ground truth item is among the top 20 retrieved items. The best results are marked in bold and second best results are underlined.Note that SAS-T is the text-enhanced SASRec proposed in this paper.

Ranker Comprision Results

We perform further ranking on a subset of 20 candidate samples that include positive samples, and the experimental results are shown in the table 3. The table 3 presents the recommendation performance of different models on the ML-100k dataset. Across all metrics, LLMRecAgent outperforms the other models by a significant margin. It achieves the highest scores for both M@5 (0.2184), N@5 (0.2693), and R@5 (0.4227), as well as for the top-10 recommendations, with scores of M@10 (0.2623), N@10 (0.3766), and R@10 (0.7560). This consistent performance across both top-5 and top-10 recommendations demonstrates the superior effectiveness of LLMRecAgent in capturing relevant items and placing them higher in the ranking.

SAS-T follows closely behind, securing second place in most cases. It achieves the second-best results for M@5 (0.2022) and N@5 (0.2498), as well as for M@10 (0.2477), N@10 (0.3508), and R@10 (0.6972). Although slightly behind LLMRecAgent, SAS-T still shows strong performance, particularly in ranking metrics like R@10, where it maintains a substantial gap over the other models.

The experimental results demonstrate that the proposed LLMRecAgent significantly improves the model's performance after ranking, indicating that large models can effectively capture personalized information from text. By utilizing the retrieval module to search through the entire candidate item set, the context length inputted into the large model is effectively reduced, thereby improving the inference speed of the large model.

Conclusion

To address the issues of excessive context length and slow inference speed in LLM-based conversational recommendation systems, an innovative two-stage architecture has been proposed—LLMRecAgent. This architecture uses a retrieval module to filter out a subset of items relevant to the user's historical interactions, reducing the context length input to the large model. Then, the large language model is used to rank and generate recommendations based on this refined subset. This approach not only reduces the computational burden but also improves inference speed, while ensuring high recommendation quality.

Acknowledgments

Thank you to both the teacher and the teaching assistants for all your hard work this semester. The teacher's lectures have always been able to break down complex deep learning concepts in a clear and engaging way. Even though some of the content was challenging, your sense of humor made it easier for us to absorb the knowledge without even realizing it. Your teaching style is approachable and really helped us better understand many of the concepts.

The teaching assistants' support has been invaluable as well. Your careful grading of assignments and willingness to answer students' questions ensured that we could keep up with the course material. Your responsible and dedicated attitude made us feel well-supported throughout the learning process. Thank you for all your help!

References

Alamdari, P. M.; Navimipour, N. J.; Hosseinzadeh, M.; Safaei, A. A.; and Darwesh, A. 2020. A systematic study on the recommender systems in the E-commerce. *Ieee Access*, 8: 115694–115716.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.

Brown, T. B. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, L.; Chen, G.; and Wang, F. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25: 99–154.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Harper, F. M.; and Konstan, J. A. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, 5(4).

He, R.; and McAuley, J. 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*, WWW '16, 507–517. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. ISBN 9781450341431.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jin, H.; Huang, L.; Cai, H.; Yan, J.; Li, B.; and Chen, H. 2024. From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, 197–206. IEEE.

Li, Z.; Zhao, H.; Liu, Q.; Huang, Z.; Mei, T.; and Chen, E. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1734–1743.

Liu, X.; Zhou, T.; Wang, C.; Wang, Y.; Wang, Y.; Cao, Q.; Du, W.; Yang, Y.; He, J.; Qiao, Y.; et al. 2024. Toward the unification of generative and discriminative visual foundation model: a survey. *The Visual Computer*, 1–42.

Lu, J.; Wu, D.; Mao, M.; Wang, W.; and Zhang, G. 2015. Recommender system application developments: a survey. *Decision support systems*, 74: 12–32.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

McAuley, J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 43–52. New York, NY, USA: Association for Computing Machinery. ISBN 9781450336215.

Rubin, O.; Herzig, J.; and Berant, J. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450.

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

Wang, H.; Zhang, P.; Lu, T.; Gu, H.; and Gu, N. 2017. Hybrid recommendation model based on incremental collaborative filtering and content-based algorithms. In *2017 IEEE 21st international conference on computer supported cooperative work in design (CSCWD)*, 337–342. IEEE.

Yang, F.; Chen, Z.; Jiang, Z.; Cho, E.; Huang, X.; and Lu, Y. 2023. PALR: Personalization Aware LLMs for Recommendation. arXiv:2305.07622.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yue, Z.; Wang, Y.; He, Z.; Zeng, H.; McAuley, J.; and Wang, D. 2024. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 930–938.

Yue, Z.; Zeng, H.; Kou, Z.; Shang, L.; and Wang, D. 2022. Defending Substitution-Based Profile Pollution Attacks on Sequential Recommenders. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, 59–70. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392785.

Zhao, Z.; Fan, W.; Li, J.; Liu, Y.; Mei, X.; Wang, Y.; Wen, Z.; Wang, F.; Zhao, X.; Tang, J.; et al. 2024. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*.

Zhou, W.; Ou, Y.; Ding, S.; Li, L.; Wu, J.; Wang, T.; Chen, J.; Wang, S.; Xu, X.; Zhang, N.; et al. 2024. Symbolic learning enables self-evolving agents. *arXiv preprint arXiv:2406.18532*.