Memory Assisted LLM for Personalized Recommendation System

Jiarui Chen
National University of Singapore
Singapore
e0893429@u.nus.edu

ABSTRACT

Large language models (LLMs) have demonstrated significant potential in solving recommendation tasks. With proven capabilities in understanding user preferences, LLM personalization has emerged as a critical area for providing tailored responses to individuals. Current studies explore personalization through prompt design and fine-tuning, paving the way for further research in personalized LLMs. However, existing approaches are either costly and inefficient in capturing diverse user preferences or fail to account for timely updates to user history. To address these gaps, we propose the Memory-Assisted Personalized LLM (MAP). Through user interactions, we first create a history profile for each user, capturing their preferences, such as ratings for historical items. During recommendation, we extract relevant memory based on similarity, which is then incorporated into the prompts to enhance personalized recommendations. In our experiments, we evaluate MAP using a sequential rating prediction task under two scenarios: single domain, where memory and tasks are from the same category (e.g., movies), and cross-domain (e.g., memory from movies and recommendation tasks in books). The results show that MAP outperforms regular LLM-based recommenders that integrate user history directly through prompt design. Moreover, as user history grows, MAP's advantage increases in both scenarios, making it more suitable for addressing successive personalized user requests.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**.

KEYWORDS

Personalized Large Language Model, Recommendation System

ACM Reference Format:
Jiarui Chen. 2018. Memory Assisted LLM for Personalized Recommendation System. In *Proceedings of XXXXXXXX (WWW).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The recent rapid development of large language models (LLMs) has garnered considerable attention for their wide range of applications. LLMs have demonstrated remarkable zero-shot learning capabilities across various traditional machine learning tasks, including recommendation systems (RS). In the RS domain, LLMs are tasked with providing recommendations based on user-item interaction data, where the tasks are presented as prompts [5, 9]. However, these "one-size-fits-all" LLMs have proven insufficient for addressing personalized individual requests [2]. As a result, the personalization of LLMs has emerged as a critical area of focus, given its essential role in meeting users' increasingly customized needs. Personalization in LLMs refers to the generation of tailored model responses by incorporating personal data, such as an individual's historical behaviors [7, 23]. By learning user preferences from these personal data, LLMs can significantly enhance their performance across fields that involve extensive individual requests, such as recommendation tasks [12, 25].

Researchers have extensively explored the implementation of personalized large language models (LLMs), focusing on two primary approaches: fine-tuning and prompt design. On one hand, personalized LLMs with fine-tuning have demonstrated advantages in various personalized tasks by adjusting LLM parameters based on users' behavioral histories [24]. However, fine-tuning faces significant challenges, particularly regarding the storage requirements associated with millions of distinct user histories that need to be fine-tuned [20]. Furthermore, compared to zero-shot in-context learning, fine-tuning is not only more computationally expensive but also prone to issues such as catastrophic forgetting of pre-trained knowledge and difficulties in aligning the model during the personalization process [27]. On the other hand, LLM recommenders that incorporate user history via prompt templates have proven to be efficient, especially when enhanced with additional strategies such as limiting the recommendation space, filtering user histories, and applying few-shot prompt strategies [3, 20, 26, 30]. Despite their efficiency, these approaches fail to account for the dynamic nature of user history. As a result, they have not adequately evaluated LLM-based recommendation systems across different sizes of user histories and exhibit inefficiencies in learning preferences from extended user histories.

To address the limitations of existing research, we propose the **M**emory **A**ssisted LLM-based **P**ersonalized recommendation system (MAP), an approach that leverages prompt design in zero-shot settings, allowing users to continuously update their behavioral history with each new interaction. This design enables LLM-based recommendation systems to capture temporal changes in user behavior, becoming progressively more user-centric as the interactions between the individual and the system increase. Additionally, MAP incorporates a history retrieval mechanism that filters the user history to extract a "memory" of the most relevant past interactions.

For demonstration and evaluation, we plan to test MAP in both single-domain and cross-domain recommendation scenarios. In the

WWW, XXXX,XXXX, XX,XX

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

single-domain case, we will focus on movies, while in the cross-domain setting, we will use books as the recommendation target and movies as the cross-domain history. However, since the LaMP benchmark does not account for the effect of varying user history sizes on recommendation systems, we have defined a new task based on one of the benchmark's existing tasks, LaMP-3 Personalized Product Rating [20], to assess MAP with an increasing user history. Following prior research [9, 12], we treat item rating prediction as a multi-class classification task. In this new task, we conduct rating predictions iteratively, varying only two aspects: the size of the user history and the target of the prediction task. This approach allows us to investigate whether increasing user history enhances the system's prediction accuracy. For cross-domain item rating prediction, we follow a similar iterative process, with the prediction target remaining the same for each user in every round. The objective is to determine if MAP's prediction accuracy for book ratings improves as more movie rating history is incorporated. Experimental results show that the MAP framework enhances recommendation system performance, with improvements becoming more significant over time. This trend holds not only in single-domain item recommendations but also in cross-domain scenarios, further demonstrating MAP's effectiveness in long-term personalized recommendation tasks.

2 RELATED WORK

2.1 Recommendation System

Traditional RS. Traditional recommendation systems are primarily categorized into three core approaches: Collaborative Filtering (CF), which predicts item ratings based on the preferences of similar users; content-based recommendation, which compares item features to users' preferences to suggest relevant items; and hybrid methods that combine these two techniques [8, 19, 21]. However, interest in these traditional approaches has declined, as their limitations have become more evident with the emergence of large language models (LLMs) in the recommendation field. These limitations include difficulties in adapting to new users, poor performance when handling large user history datasets, and the laborious task of obtaining item descriptions through manual web scraping [1, 14]. A key reason for the shift in researchers' focus is the demonstrated superiority of LLM-based recommendation systems. These systems have shown better performance than traditional RSs, often requiring only minimal training data to achieve competitive results [3]. As a result, traditional RSs are increasingly viewed as less effective in solving modern recommendation tasks.

LLM-based RS. Recently, the application of LLM in the recommendation field has gained unprecedented attention, and these approaches can be broadly categorized into two major classes: fine-tuning LLMs for downstream recommendation tasks and direct prompting of LLMs. Fine-tuning approaches typically adjust model parameters or instructions using user history pairs, while direct prompting, such as zero-shot recommendations, generates outputs solely based on prompt inputs without altering the model [15, 28]. In the fine-tuning paradigm, LLMs are able to acquire more domain-specific knowledge by optimizing their parameters and weights through task-specific datasets. This allows the model to become more specialized for particular recommendation tasks. However, fine-tuning is also widely recognized as computationally expensive and time-consuming, even when employing more efficient methods like Parameter-Efficient Fine-Tuning (PEFT), which reduces the resource requirements compared to full-model fine-tuning [27]. On the other hand, the direct prompting paradigm for recommendation systems is more flexible and lightweight, as it bypasses the need for extensive model updates, offering a more efficient alternative for generating recommendations [29].

2.2 Personalization in Large language Model

Personalizing LLMs remains an important yet underexplored research challenge. Early studies have demonstrated that LLMs can outperform popular recommendation algorithms when integrated with recommendation systems [9, 13]. Subsequent research efforts in personalizing LLMs have largely focused on designing prompts that incorporate user-item interaction histories. For example, Lyu et al. [16] explored four prompting strategies for RS testing, showing that strategically guiding the LLM toward specific queries can enhance recommendation accuracy. Similarly, Dai et al. [3] used ChatGPT to explore its capabilities across three types of ranking tasks: point-wise ranking (predicting product ratings), pair-wise ranking (choosing between two product pairs), and list-wise ranking (ordering a list of products based on user preferences). Salemi et al. [20] proposed the LaMP benchmark as an evaluation framework for LLM-based recommenders, building on existing natural language processing (NLP) benchmarks, which do not allow for model development to adapt to specific user needs. The LaMP benchmark includes seven personalized tasks, with three involving text classification and four involving text generation, covering tasks like tagging, rating, and headline generation. To address the limitations of most LLMs' context lengths, the benchmark introduces Retrieval Augmentation, a method for retrieving the most relevant user behavior before prompting the LLM. This approach increases the likelihood that the LLM will accurately learn user preferences for a given item. However, as mentioned earlier, the benchmark does not allow for evaluations based on varying sizes of user histories, which limits its ability to assess the performance of personalized LLMs in different historical contexts.

3 PROBLEM STATEMENT

In this section, we present the problem statement and describe the two personalization settings our study focuses on.

In the single-domain setting, each user provides a history composed of a sequence of item ratings, $(H_1, H_2, H_3, ..., H_n)$, where each rating is represented as a $\{(item, rating)\}$ pair. During the experiment, at iteration r, the model predicts the rating for the $(r+1)$th item based on the previous r item ratings. For instance, at the beginning of the experiment, the model is tasked with predicting the rating of H_2 given the item in H_2 and the history (H_1). In the next iteration, this history expands to (H_1, H_2), and the model then predicts the rating for H_3. Consequently, for each user, the model generates a sequence of predictions $\left(P_1, P_2, P_3, ..., P_{(n-1)}\right)$, where each prediction P_i is dependent on the history $\left(H_1, ..., H_{(i-1)}\right)$.

In the cross-domain setting, each user has a history of item ratings $(H_1, H_2, H_3, ..., H_n)$ in domain A, and this history is used to predict ratings for items in domain B. At each iteration, the model predicts the rating for a specific item in domain B, producing a prediction P_i based on a subset of the history $(H_1, ..., H_i)$ for domain A. Thus, the final output for each user in the cross-domain setting is a sequence of predictions $(P_1, P_2, P_3, ..., P_n)$.

4 METHOD: MEMORY ASSISTED LLM-BASED RECOMMENDATION SYSTEM

In this section, we present the pipeline of the proposed Memory-Assisted LLM for Personalized Recommendation System (MAP). The core idea of memory assistance is to maintain a retrievable user profile, stored separately from the language model, functioning as a dynamic memory. This memory is continuously updated based on user interactions and retrieved to refine recommendation outputs by incorporating relevant user preferences. The primary objective is to enable recommendation systems (RSs) to generate more personalized responses tailored to each individual user. The MAP system consists of five key components: the user profile, detection module, retrieval module, update module, and a basic large language model. The user profile stores information on the user's historical interactions and preferences. The detection module monitors user interactions to identify changes or new inputs. The retrieval module accesses relevant portions of the user profile to be used for making predictions. The update module ensures the memory is kept current by integrating new user data after each interaction. Finally, the base large language model processes the inputs and generates personalized recommendations, enhanced by the retrieved user history. The flow of this system is illustrated in Fig. 1.

4.1 User Profiles

The user profile serves as the system's memory storage, maintaining a comprehensive and categorized collection of data that reflects each user's preferences across various domains. This data includes movie ratings, product reviews, and other interactions, providing a detailed historical record that allows the system to access relevant user preferences whenever a recommendation or prediction is needed.

For each user, this structured memory is organized in a table format. In the case of movie ratings, for example, the table may include columns such as movie title, genre, date watched, and the user's rating. Each row represents a distinct interaction, recording the user's experience with a specific movie. Similar tables are maintained for other domains, such as product reviews or book ratings, ensuring that user preferences are stored systematically.

This structured format allows the system to efficiently retrieve specific entries relevant to a given query, enabling it to concentrate on the user's preferences within a particular context or genre. This enhances the system's ability to provide highly personalized recommendations based on the user's past interactions.

4.2 Detection Module

Compared to the original LLM, our MAP system introduces two additional behaviors specifically tailored for recommendation tasks: (1) constructing or updating the memory, which serves as the user profile, and (2) retrieving this memory to generate personalized recommendations. All other general functionalities of the LLM remain unchanged. To handle these new behaviors, a detection module is employed to classify the type of action required for each incoming query during user interactions.

We categorize incoming queries into three types:

- **Type A:** A request for a recommendation. This type requires retrieving the memory from user profile to generate personalized suggestions.
- **Type B:** A message that reflects a new user preference, which prompts the system to update the user profile in memory with the new information.
- **Type C:** A query unrelated to recommendations, where the LLM responds directly without utilizing the memory system.

For instance, when a user asks for a movie recommendation, the system classifies this as a Type A query, since it must access the user's preferences, such as favorite movie genres or past ratings, to provide relevant suggestions. In this case, the user's previous interactions, stored in the user profile, are passed to the retrieval module to assist in generating personalized recommendations. On the other hand, if the user rates or selects a movie, this is classified as a Type B query, as it reflects a new preference that needs to be integrated into the user profile for future recommendations. This update ensures that the system continually refines its understanding of the user's tastes. To manage this process, the detection module relies on a prompt template, as illustrated in Fig. 2. For each incoming query, we prompt the LLM with additional steps to first identify the type of query—whether it is Type A, B, or C—before proceeding with the corresponding action. This ensures that the system correctly processes and responds to various user inputs, enhancing both the personalization and efficiency of the recommendation process.

4.3 Memory retrieval and Memory Assisted Recommendation

When a query involves generating a recommendation, the retrieval module first loads the user's stored history from memory, which typically consists of a list of items, including details such as the name, description, and historical rating. However, loading the entire user history every time a recommendation is requested can be inefficient, as many entries may be irrelevant to the current query. Moreover, large language models (LLMs) have inherent limitations, including high computational costs for processing long inputs and constraints on maximum sequence length.

To address these challenges, we draw inspiration from previous studies that have shown the effectiveness of in-prompt retrieval augmentation for personalizing LLMs [20]. In our memory-assisted approach, rather than loading all user data, the retrieval module selectively extracts the most relevant data points from the user profile based on the current query. This strategy enables the system to focus on the most pertinent subset of data, improving both the efficiency of processing and the quality of the generated recommendations. By limiting the input to only the most relevant user preferences, the system enhances its ability to produce accurate, personalized recommendations without overwhelming the LLM with unnecessary data.

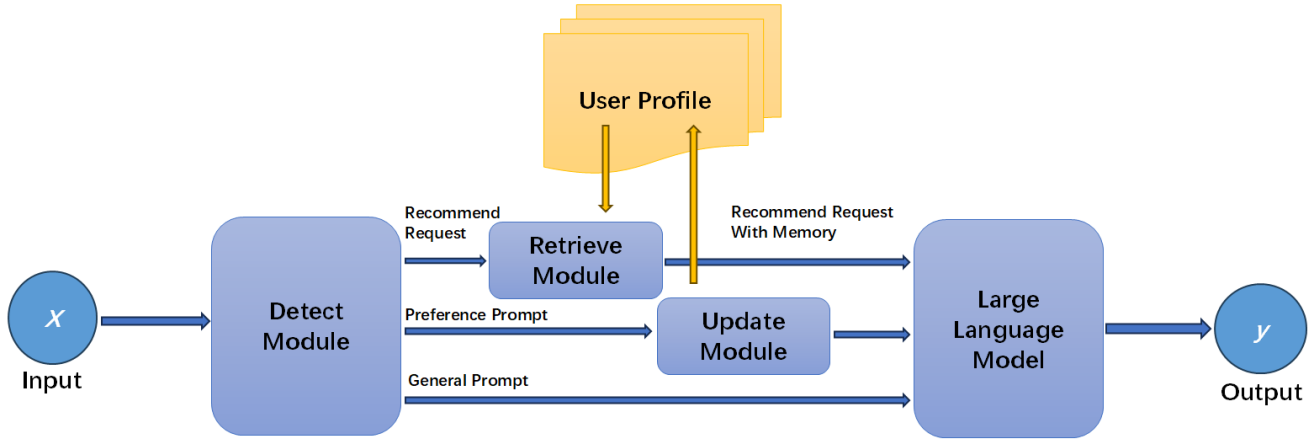

Figure 1: Pipeline of our proposed Memory-Assisted LLM-based recommendation system for personalization. The system consists of five key components: the detection module, user profile, retrieval module, update module, and the language model. The detection module classifies user queries, triggering either memory retrieval for generating recommendations or profile updates based on user preferences. The retrieved memory data is then passed to the language model to provide personalized recommendations.

If the text delimited by <<>> is asking for a recommendation, your output: A
if the text reflects a new preference, such as picking a recommendation, your output: B
if the text is irrelevant to recommendations, your output: C
<<{input}>>

Figure 2: The prompt used for detect module.

Since each item in the user's history contains a detailed description, we use the similarity between the item that needs to be predicted and the historical items to filter and rank the retrieved memory. This similarity is computed using a scoring mechanism that compares the description and attributes of the item to be predicted with those of each item in the user's history. The similarity score for each historical item is calculated as follows:

$$\text{Score}_i = \text{Similarity}(item_i, x) \tag{1}$$

where $item_i$ represents each instance stored in memory, and x is the input for the task requiring a rating prediction. The function Similarity is a model that computes the similarity score between the historical items and the target item. By applying this scoring mechanism, the system can prioritize and retrieve the most relevant historical items, ensuring that only the most similar and relevant data points are used to inform the recommendation, thereby improving the quality and efficiency of the prediction process.

In our method, we explore different implementations for the similarity module depending on the type of description associated with each item. For items that include a genre list in their description, we calculate the similarity by comparing the intersection of

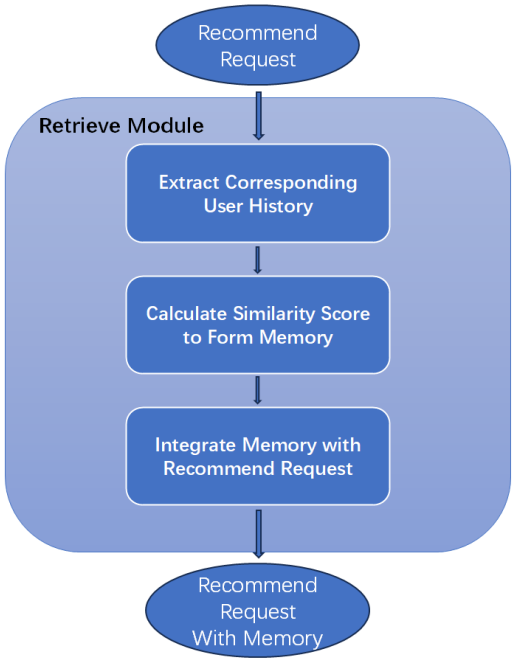

Figure 3: Inner Structure of the Retrieval Module. The retrieval module consists of three key steps: 1) loading the stored original user profiles; 2) calculating similarity scores between stored items and the current query, and 3) selecting the most relevant items from the user's history. These relevant items are then fed into the language model to assist in generating personalized recommendations.

The text delimited by <<>> is the client's movie rating history, with rating ranged from 1 to 5, where 1 means dislike and 5 means like.
<<{his}>>
The text delimited by <<<>>> is the movie that the client is going to rate, based on the rating history, predict the client's rating on this movie.
You should only output the predicted rating, do not provide any other explanation. Output example: 5
<<<{task}>>>

Figure 4: The prompt template which combines the retrieved memory and conducts recommendations for a new item.

genres between the predicted item and the historical items. This approach helps prioritize recommendations that align with the user's preferences in terms of genre.

Beyond simple attribute matching, we also employ a more advanced technique using a pre-trained language model, such as BERT [4], for text feature extraction. By embedding the text descriptions of both the historical and predicted items, we compute the cosine similarity between their feature vectors. This allows for a semantic comparison, enabling the system to identify similarities between items even if they are described using different terms.

Once the most relevant items are identified and ranked, the processed memory is integrated into the LLM as part of the prompt. The template for the prompt is shown in Fig. 4. This augmented input enables the LLM to generate more personalized and contextually accurate recommendations by focusing on the user's most relevant historical preferences. By narrowing the input to the most pertinent data, the model reduces computational costs while enhancing the quality of the predictions, ensuring a more efficient and personalized recommendation process.

4.4 Memory update

The updating module integrates new data into the user profile whenever a new preference is detected from the input. This process is triggered when the system identifies a Type B query.

When a user interacts with the system in ways that indicate new preferences—such as submitting a new rating or review—the system processes and stores this information within the user profile. The prompt template used for updating is shown in Fig. 5. The newly gathered data is structured in the same format as the existing entries in the user profile, maintaining consistency across all stored information. This ensures that the system can seamlessly retrieve the updated data for future queries, keeping the user profile reflective of the most current and accurate preferences. For example, if a user rates a movie after watching it or leaves a review detailing their experience, the updating module records this information, including attributes such as the item's name, description, rating, and any additional context like genre or category tags. By capturing this data, the system becomes more capable of delivering personalized recommendations that reflect the user's evolving preferences.

5 EXPERIMENTS

In this section, we evaluate the effectiveness of our Memory-Assisted Personalized Recommendation System (MAP) through a series of

The text delimited by <<>> is the client's new preference, summarize them in JSON format with item title, description, rating as keys.
Output example: "Movie": "The Dark Knight", "Genres": "Action", "Rating": "5"
<<{input}>>

Figure 5: The prompt used for Update Module

experiments. The experiments are designed to assess the system's performance in two key scenarios:

- **Single-domain movie rating prediction:** In this scenario, we evaluate how well MAP predicts movie ratings based on a user's past interactions within the movie domain. The system uses the user's movie rating history to generate personalized recommendations for new movies.
- **Cross-domain book rating prediction based on movie ratings:** In this scenario, we assess MAP's ability to predict book ratings by leveraging the user's movie rating history. This cross-domain evaluation examines whether MAP can effectively transfer learned user preferences from one domain (movies) to another (books), showcasing its versatility in handling multi-domain recommendation tasks.

By conducting these experiments, we aim to determine the impact of memory-assisted retrieval and personalized prompt design on recommendation accuracy and user satisfaction in both single-domain and cross-domain contexts.

5.1 Experimental Details

We use the GPT-3.5-turbo model [10] for our study, as prior research has demonstrated the superiority of ChatGPT in similar recommendation tasks [3, 6, 22]. Our study includes two experiments: a movie rating prediction experiment and a cross-domain rating prediction experiment. Both experiments utilize a retrieval approach to select relevant rating histories as memory to assist the LLM model in learning user preferences.

In the movie rating prediction experiment, we sort the user's rating history based on the number of matching movie genres, extracting ratings for movies with the highest genre overlap as part of the prompt input for the model. This approach ensures that the memory provided to the LLM focuses on items most similar to the current prediction task, thereby improving the relevance of the personalized recommendations.

In the cross-domain prediction experiment, we select movies and books as the two domains for testing. The primary focus is to determine whether the accuracy of book rating predictions by the memory-assisted LLM improves with an increasing amount of movie rating history used as memory. Since genres differ between these domains, we utilize a pre-trained BERT tokenizer to encode the genres of both books and movies. We then calculate the cosine similarity between these encoded representations. Similar to the first experiment, the memory consists of ratings for movies with high cosine similarity to the book being predicted.

For evaluation, we use mean absolute error (MAE)—a commonly employed metric in rating prediction—to assess the performance

in both experiments. This metric helps us quantify the accuracy of the system's predictions by measuring the average magnitude of the errors between the predicted and actual ratings.

5.2 Baseline Model

Based on the approach in [3], we developed a baseline model that directly prompts the LLM with the unprocessed rating history, without using extracted memory. This baseline model serves as a comparison for evaluating the effectiveness of our memory-assisted approach using the GPT-3.5-turbo model.

In the movie rating prediction task, the baseline model treats the user's past ratings as a series of historical messages. The model is prompted with a prediction task query and an updating prompt for each iteration. The updating prompt informs the model of the actual rating for that round's query. For subsequent iterations, the baseline model is given both the current query and the past queries along with their corresponding updating prompts, essentially providing the model with the user's historical interactions as messages. The baseline model also includes an additional "history prompt," ensuring that every starting iteration for each user begins with an identical input structure. This design simulates a scenario in which users repeatedly query a RS without the assistance of memory extraction or selection mechanisms. The model essentially recalls the entire interaction history as flat sequences of messages.

For cross-domain rating prediction, where the task is to predict book ratings based on movie ratings, we adjust the baseline model by removing the updating prompts from each iteration. This is because the prediction task (a single book rating) remains constant across iterations. Instead, the model receives one book as the prediction object and the user's movie rating history. In the baseline model, the user's movie rating history grows incrementally over iterations, with each rating presented as a separate sentence. Unlike the memory-assisted model, which filters and extracts the most relevant historical ratings for each iteration, the baseline model feeds one movie rating at a time into the LLM as historical messages alongside the task query for each iteration. This results in a more fragmented input structure compared to the memory-assisted model, where movie rating history and task queries are integrated into a complete prompt for each iteration.

5.3 Single-domain Recommendations

We first evaluate the single-domain recommendation task using movie rating prediction. In this scenario, each user is provided with a list of historical movie ratings reflecting their preferences, and the goal is to predict the next movie rating. For each user, the historical data starts with 1 entries, and with each iteration, additional rating history is incrementally added, growing from 1 to 19 entries.

Dataset. The memory-assisted model is evaluated using the widely adopted MovieLens 100k dataset [11], which contains 944 users and 1,683 movies. To ensure sufficient data per user, we exclude users with fewer than 20 movie ratings and limit each user to exactly 20 ratings. After preprocessing, the dataset consists of 834 users and a total of 16,680 movie ratings. For evaluation, we compare the predicted rating for each user's next movie with the corresponding ground truth rating.

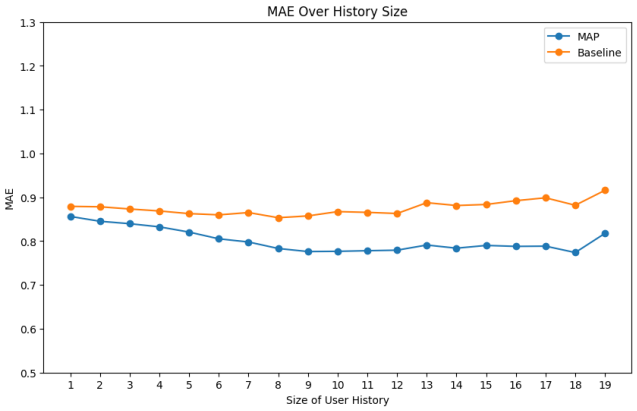

Figure 6: MAE trends over different user history sizes for the our MAP and baseline for the single-domain recommendation. Our MAP shows a consistent decline in MAE as the user history increases, while the baseline model maintains stable performance.

Results. For each user, the model generates 19 predictions for 19 different movies. We compute the absolute difference between the predictions and the users' actual ratings and calculate the Mean Absolute Error (MAE) across the 834 users. From the visualization in Fig. 6, the memory-assisted model consistently outperforms the baseline starting from the second iteration, with similar performance to the baseline during the first iteration.

A notable trend is the decreasing MAE for the MAP as the amount of user history increases, while the baseline model's MAE remains relatively stable across all iterations. As shown in Table 1, the memory-assisted model (MAP) provides an average improvement of 4.89% in MAE when the user history size reaches 5 entries, and this improvement grows to 12.27% when the user history size increases to 17 entries. This results in a widening gap between the two models' MAE values as the user's movie rating history grows. These findings indicate that incorporating memory enables the LLM-based recommendation system to deliver more accurate predictions as more user history is accumulated, significantly enhancing personalization.

Table 1: The comparison of Mean Absolute Error (MAE) of single-domain recommendation under different user history sizes

User's history size	5	9	13	17
Vanilla GPT	0.8628	0.8576	0.8878	0.8989
MAP (Ours)	**0.8206**	**0.7763**	**0.7911**	**0.7886**

Costs. Beyond performance, we also compare the cost of running the recommendation models using the same API of the large language model (see Table 2). For both our model and the baseline model, we use the pricing of GPT-3.5-turbo [18]. For the MovieLens dataset, we calculate the average number of tokens processed per user history and multiply this by the API's cost per token. The

Table 2: The average cost of running when increasing every 10 history for one user. This is estimated by the average token used for each piece of user history from MovieLens dataset and the cost of API for GPT model.

Method	Cost of dollar / 10 history per user
Vanilla GPT	$0.00325
MAP (Ours)	**$0.00086**

baseline model (Vanilla GPT) requires more tokens to generate recommendations due to its less efficient handling of user history, resulting in a higher operational cost of $0.00325 per 10 histories per user. In contrast, our proposed MAP model optimizes token usage by summarizing and retrieving relevant user history, thereby reducing the cost to $0.00086 per 10 histories per user. This significant cost reduction demonstrates the economic advantage of our approach, making it more scalable and sustainable for large-scale deployments.

5.4 Cross-domain Recommendation

To evaluate the model's ability to perform cross-domain recommendations, we designed an experiment where the task is to predict book ratings based on a user's movie rating history. This experiment tests the model's ability to transfer knowledge from one domain (movies) to another (books) and still provide relevant recommendations. For each user, the experiment is conducted over 19 iterations, with the number of movie ratings used in each iteration incrementally increasing from 1 to 19. The memory-assisted model retrieves relevant movie ratings from the user's profile and calculates the cosine similarity between movie genres and book genres. The most similar movie ratings are used to assist in predicting the user's book rating.

Dataset. For this cross-domain experiment, we use movie and book ratings from the Amazon Review Data [17]. We employ the "ratings only" version of the dataset and merge it with item metadata to ensure both movie and book ratings are aligned with their respective genres. After filtering out users with fewer than 20 movie interactions and capping the number of movie ratings at 19 per user, the refined dataset consists of 418 users, 418 book ratings, and 7,942 movie ratings.

Table 3: The comparison of Mean Absolute Error (MAE) of cross-domain recommendation under different user history sizes.

User's history size	5	9	13	17
Vanilla GPT	0.8218	0.8160	0.8206	0.8162
MAP (Ours)	**0.7634**	**0.7459**	**0.7258**	**0.7037**

Results. For each user, the model generates 19 predictions for a single book, each based on an incrementally growing history of movie ratings ranging from 1 to 19 entries. The absolute difference between each predicted book rating and the actual rating is calculated, and the Mean Absolute Error (MAE) is computed across all 418 users. To ensure robustness, we shuffle the order of the movie history for each user and repeat the experiment for both the memory-assisted model and the baseline model. The final MAE is averaged across the shuffled and unshuffled results to eliminate potential biases from the order of history inputs.

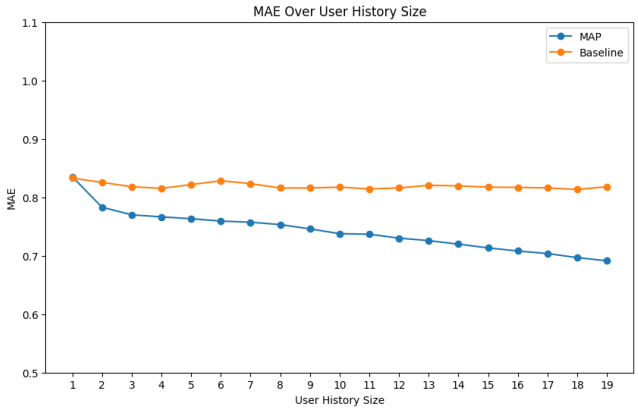

Figure 7: MAE results for cross-domain recommendation tasks. The memory-assisted model shows a clear improvement as the number of movie ratings increases, compared to the baseline model which struggles to maintain its initial performance as the user history grows.

As shown in the smoothed graph in Fig. 7, the memory-assisted model consistently outperforms the baseline (Vanilla GPT) on average. The memory-assisted model shows significant improvement as more movie rating history becomes available, especially in later iterations. The MAE for the MAP steadily declines as the movie history grows, while the baseline model exhibits a temporary dip in error during the first five iterations but returns to its initial performance levels afterward. Table 3 provides more detailed values, showing that MAP achieves an average improvement of 7.10% over the baseline when there are 5 user histories and an even higher average improvement of 13.78% when the history size reaches 17 entries.

This performance suggests that the baseline model, which handles larger user history directly within its prompt, struggles with efficiently managing increasing amounts of input data, leading to a degradation in performance. In contrast, the memory-assisted model maintains its ability to personalize recommendations more effectively as the user history expands, even in a cross-domain setting, where movie ratings are used to predict book ratings.

Table 4: The average cost of running when increasing every 10 history for one user.

Method	Cost of dollar / 10 history per user
Vanilla GPT	$0.00322
MAP (Ours)	**$0.00156**

Costs. In addition to performance improvements, the memory-assisted approach significantly reduces computational costs, as shown in Table 4. Costs were measured by calculating the average expense incurred when adding 10 additional pieces of user history data for each user.

The cost per user for the Vanilla GPT model increases substantially as more user history is added, averaging $0.00322 per 10 history entries. In contrast, the memory-assisted model (MAP) reduces this cost to $0.00156 per 10 history entries. This cost reduction is achieved by focusing only on the most relevant user history through memory retrieval, minimizing the data the LLM processes.

This cost-efficiency is especially valuable in large-scale applications that manage thousands or millions of users with substantial historical data. By using the memory-assisted approach, systems can maintain or improve performance while reducing the operational costs associated with recommendation tasks, making it a scalable and sustainable solution.

6 CONCLUSION

In the context of the rapid development of large language models (LLMs), increasing attention has been drawn to the field of personalized LLMs. In this paper, we introduced MAP (Memory-Assisted Personalized LLM), a model that integrates users' past behaviors while updating with new user requests, demonstrating its advantages in supporting long-term interactions between users and recommender systems in both single-category and cross-domain recommendation tasks. By adopting a retrieval framework to filter and prioritize the most relevant historical data, MAP effectively captures patterns in user preferences, understanding their likes and dislikes over time. To further evaluate MAP in a personalized setting with progressively increasing user history, we designed a new task building upon the LaMP benchmark [20]. The proposed MAP structure taps into new potentials for LLM personalization and provides a framework that could guide the design of future personalized LLMs. Our findings underscore the importance of memory-assisted architectures in enhancing both the accuracy and efficiency of personalized recommendations.

REFERENCES

[1] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. Llm based generation of item-description for recommendation system. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1204–1207.

[2] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. 2024. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web* 27, 4 (2024), 42.

[3] 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. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1126–1132.

[4] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Dario Di Palma. 2023. Retrieval-augmented recommender system: Enhancing recommender systems with large language models. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1369–1373.

[6] Dario Di Palma, Giovanni Maria Biancofiore, Vito Walter Anelli, Fedelucio Narducci, Tommaso Di Noia, and Eugenio Di Sciascio. 2023. Evaluating chatgpt as a recommender system: A rigorous approach. *arXiv preprint arXiv:2309.03613* (2023).

[7] Joel Eapen and VS Adhithyan. 2023. Personalization and customization of llm responses. *International Journal of Research Publication and Reviews* 4, 12 (2023), 2617–2627.

[8] Lei Fu and XiaoMing Ma. 2021. An improved recommendation method based on content filtering and collaborative filtering. *Complexity* 2021, 1 (2021), 5589285.

[9] 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).

[10] gpt3.5turbo 2022. gpt-3-5-turbo. https://platform.openai.com/docs/models/gpt-3-5-turbo.

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[12] 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).

[13] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: an efficient all-round LLM-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1395–1406.

[14] Balraj Kumar and Neeraj Sharma. 2016. Approaches, issues and challenges in recommender systems: a systematic review. *Indian journal of science and technology* 9, 47 (2016), 1–12.

[15] Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2023. Pre-train, Prompt, and Recommendation: A Comprehensive Survey of Language Modeling Paradigm Adaptations in Recommender Systems. *Transactions of the Association for Computational Linguistics* 11 (2023), 1553–1571.

[16] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. 2023. Llm-rec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780* (2023).

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

[18] OpenAI Pricing 2024. OpenAI Pricing. https://openai.com/api/pricing/.

[19] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE international conference on data mining*. IEEE, 502–511.

[20] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406* (2023).

[21] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 291–324.

[22] Ítallo Silva, Leandro Marinho, Alan Said, and Martijn C Willemsen. 2024. Leveraging ChatGPT for Automated Human-centered Explanations in Recommender Systems. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 597–608.

[23] Zhaoxuan Tan and Meng Jiang. 2023. User modeling in the era of large language models: Current research and future directions. *arXiv preprint arXiv:2312.11518* (2023).

[24] Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024. Democratizing large language models via personalized parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.04401* (2024).

[25] Alicia Y Tsai, Adam Kraft, Long Jin, Chenwei Cai, Anahita Hosseini, Taibai Xu, Zemin Zhang, Lichan Hong, Ed H Chi, and Xinyang Yi. 2024. Leveraging LLM Reasoning Enhances Personalized Recommender Systems. *arXiv preprint arXiv:2408.00802* (2024).

[26] Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153* (2023).

[27] Stanisław Woźniak, Bartłomiej Koptyra, Arkadiusz Janz, Przemysław Kazienko, and Jan Kocoń. 2024. Personalized large language models. *arXiv preprint arXiv:2402.09269* (2024).

[28] 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).

[29] Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. 2024. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering* (2024).

[30] Aakas Zhiyuli, Yanfang Chen, Xuan Zhang, and Xun Liang. 2023. Bookgpt: A general framework for book recommendation empowered by large language model. *arXiv preprint arXiv:2305.15673* (2023).