



Collaborative Optimization Approach for Workflow Agents in User Behavior Modeling

Xinyu Zhang^{1†}
zhangxinyu1995@ruc.edu.cn
Renmin University of China
Beijing, China

Ran Dou^{1†}
samast_dou@163.com
Huawei Poisson Lab
Hangzhou, Zhejiang, China

Enrui Hu^{1†}
huenrui1@huawei.com
Huawei Poisson Lab
Hangzhou, Zhejiang, China

Minjun Zhao^{1†}
zhaominjun1@huawei.com
Huawei Poisson Lab
Hangzhou, Zhejiang, China

Yangkai Ding
dingyangkai@huawei.com
Huawei Poisson Lab
Hangzhou, Zhejiang, China

Zhicheng Dou*
dou@ruc.edu.cn
Renmin University of China
Beijing, China

Abstract

User behavior modeling is increasingly critical for personalized services and decision-making systems, yet integrating diverse user and product features into a coherent review generation process remains challenging. Thus we propose a novel collaborative optimization approach for workflow agents in user behavior modeling that integrates user and product feature extraction with review and rating generation. Firstly, to solve the difficulty in determining the optimal structure of agent workflows, we employ Monte Carlo Tree Search (MCTS) to optimize the workflow architecture, establishing a high-performance baseline. Meanwhile, to tackle the challenges in generating and optimizing single-agent prompts and demonstrations, we implement a heuristic optimization strategy for joint automated tuning of system prompts and demo cases. Furthermore, through comprehensive analysis of data distributions, we construct a dynamic routing mechanism for the agent workflow, achieving enhanced performance across diverse scenario-specific datasets. We validate the effectiveness of our methods on three real-world datasets, demonstrating significant performance improvements across all proposed techniques. This approach secured second place overall (and first in star-rating prediction) in the user modeling track of the AgentSociety Challenge @ WWW 2025.

CCS Concepts

• Computing methodologies → Multi-agent planning.

Keywords

Workflow Agents; User Behavior Modeling; Automatic Prompt Optimization; In-context Learning; LLM

[†] Equal contribution.

* Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

WWW Companion '25, Sydney, NSW, Australia

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1331-6/2025/04

<https://doi.org/10.1145/3701716.3719228>

ACM Reference Format:

Xinyu Zhang^{1†}, Ran Dou^{1†}, Enrui Hu^{1†}, Minjun Zhao^{1†}, Yangkai Ding, and Zhicheng Dou*. 2025. Collaborative Optimization Approach for Workflow Agents in User Behavior Modeling. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25)*, April 28-May 2, 2025, Sydney, NSW, Australia. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3701716.3719228>

1 Introduction

In the era of information overload, understanding, and simulating user behavior has become a cornerstone of modern recommendation systems, personalized services, and so on. User behavior modeling aims to extract valuable insights and patterns from user behaviors, facilitating customization and adaptation of systems to meet specific users' needs. Traditional user behavior modeling approaches rely on statistical analysis or shallow machine learning techniques and often struggle to capture the slight interplay between user characteristics and item attributes [1, 2]. Recently, large language models (LLMs) have achieved remarkable results in simulating human-like reasoning and text generation. LLM-based agents have demonstrated unprecedented capabilities and effectiveness in modeling complex user behaviors in interactive environments [3, 4].

However, the single agent is not universally effective in the field of user behavior modeling. Taking the scenario of generating user reviews and ratings for items as an example, this task constitutes a complex system engineering project that requires multiple steps to form a complete workflow. Specifically, it necessitates separate analyses of user characteristics and item attributes, followed by synthesizing these analytical results into final reviews and ratings. Consequently, the task needs to be decomposed into processing units suitable for multi-agent collaboration. However, the decomposed workflow may have multiple potential structures and configurations, requiring identification of the optimal workflow that satisfies specified constraints [5].

Moreover, the LLM-based agents within the workflow require both system prompts to define their roles, and supplementary scenario-specific "demo cases" to activate LLM's in-context learning capabilities, thereby enhancing the ability to solve concrete tasks. However, crafting high-quality system prompts presents significant challenges, and manual dynamic adjustments for system prompts based on performance prove difficult [6]. Concurrently, scenario "demo cases" are generally challenging to manually construct [7]. In

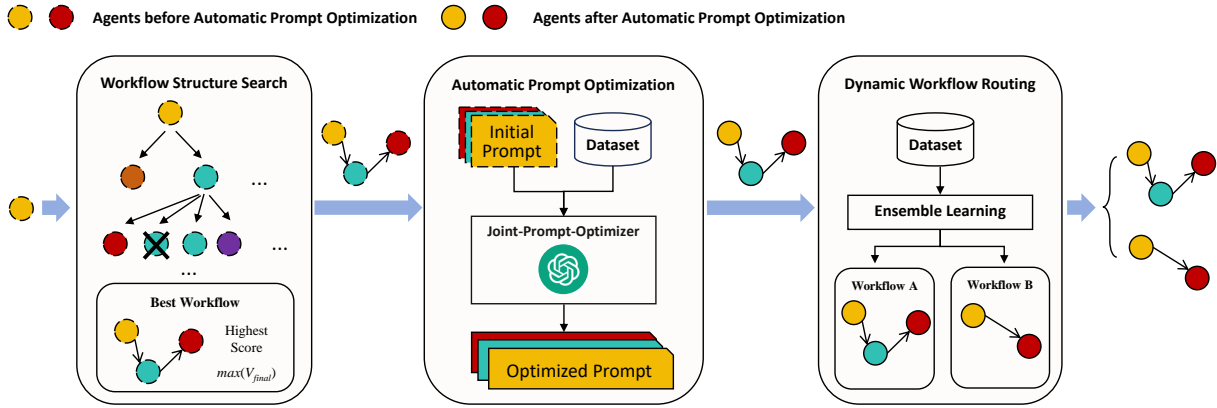


Figure 1: The framework of the proposed collaborative optimization approach for workflow agents

the user review and rating generation scenario, the heterogeneous distributions of users and items coupled with diverse commenting styles make it difficult to manually devise a set of "demo cases" with sufficient coverage and effectiveness. Furthermore, some critical parameters, such as the quantity, sequencing, and distribution of "demo cases," can substantially influence agent performance.

To address these challenges and maximize the effectiveness of user review generation and rating tasks, we propose a novel collaborative optimization approach for workflow agents in user behavior modeling. Firstly, the proposed framework leverages Monte Carlo Tree Search (MCTS) [8] to identify the optimal structure for workflow agents by automatically searching for effective module combinations and execution sequences tailored to multi-stage tasks, such as user preference analysis, product feature extraction, and review synthesis. In addition, we introduce a automatic prompt optimization strategy that utilizes example selection and chain-of-thought generation to jointly refine system prompts and illustrative cases, enhancing the model's performance in complex interactive scenarios. Furthermore, a dynamic Workflow Router is incorporated to adaptively select the most appropriate workflow paths based on dataset characteristics, ensuring both high efficiency and robust performance across various domains. Experiments on three real-world review datasets: Amazon, Goodreads, and Yelp demonstrate that the proposed approach achieves high accuracy in both star-rating prediction and strong relevance in generated reviews.

It is worth mentioning that the proposed method also secured second place overall (first in star-rating prediction) in the user modeling track of the AgentSociety Challenge @ WWW 2025, underscoring the potential of optimizing workflow agents in user behavior modeling.

2 Method

The proposed framework comprises three main components: workflow structure optimization, automatic prompt optimization, and dynamic workflow routing. Our framework is shown in Figure 1.

2.1 Workflow structure search method

In this section, we present a workflow structure optimization methodology aimed at maximizing the performance score while ensuring the execution time constraints are satisfied. Specifically, the structure of a workflow necessitates the clarification of the number

Algorithm 1 Workflow structure search method.

Require: Initial Workflow W_0 , Dataset D , Max iterations K

Ensure: Optimized Workflow W^*

```

1: Initialize  $W_0$ 
2:  $W^* \leftarrow W_0$ 
3: for iteration  $\leftarrow 1$  to  $K$  do
4:    $node \leftarrow \text{Select}$                                 {Using UCT}
5:    $child.workflow \leftarrow \text{Expand}(node)$ 
6:    $score \leftarrow \text{Evaluate}(D)$                         {Both simulation scores and
                                                         execution time}
7:    $\text{Backpropagate}(score)$                             {Update memory}
8:   Update  $W^*$  if improved
9:   if Terminal Condition then
10:    break
11:   end if
12: end for
13: return  $W^*$ 

```

of nodes it contains, the identity of each node, and the manner in which they are interconnected. In this context, we define each node within the workflow as an agent that is composed of a LLM. Consequently, a workflow can be conceptualized as a multi-agent system. Additionally, the dataset provided for this competition comprises three distinct types of data. Due to the variations in data distribution, we will conduct separate workflow structure optimizations for each dataset. Based on AFLOW [9], our key idea is to leverage the tree structure of Monte Carlo Tree Search (MCTS) to optimize workflows. Optimization is performed across each dataset and the complete process is detailed as follows:

Initialization. We first randomly initialize the workflow with a single node, which directly takes the inputs and outputs the prediction. It can also be initialized with a predefined structure.

Selection. We apply the Upper Confidence Bounds for Trees (UCT) formula [10] to select the node, which will serve as the workflow for subsequent expansion. The formula ensures a balance between exploration and exploitation, it is defined as:

$$UCT(s) = V(s) + c \sqrt{\frac{\ln N_{parent}(s)}{N(s)}} \quad (1)$$

where $N(s)$ is the number of visits to node s , $V(s)$ is the value function (expected return) from the subtree of s , c is the exploration weight, and $N_{parent}(s)$ is the visiting count of the parent node of s . Starting from the root node each time select the child node with the highest UCT value for exploration. As the number of visits increases, the UCT value decreases, thereby making it more inclined to select nodes that have not yet been statistically evaluated.

Expansion. We use LLMs to generate new workflows by adding or removing agents, thus creating new workflow configurations. The prompt of the added agent is generated by the LLM. Each selection, expansion, and evaluation will be recorded in memory, serving as the prompt for the LLM used during the expansion phase to assist in making decisions regarding the current expansion action.

Evaluation. The workflow is then evaluated both by the simulation scores and execution times. Due to the efficiency, we design a new evaluation function to simulate the scoring during this phase:

$$V_{final} = w \cdot V_e + (1 - w) \cdot V_t$$

$$V_t = \begin{cases} 1 & \text{if } t \leq 0.9 \times T_{limit} \\ e^{-\alpha(t - 0.9 \times T_{limit})} & \text{if } 0.9 \times T_{limit} < t \leq 1.2 \times T_{limit} \\ 0 & \text{if } t > 1.2 \times T_{limit} \end{cases} \quad (2)$$

where V_e is the value of evaluation of training dataset, V_t is the value of execution time, V_{final} is the final score of evaluation, w is the weight of V_e , t is the average execution time per task and T_{limit} is the average execution time limit for each task. The function of this formula is that when the average execution time for each task is less than 0.9 times the T_{limit} , the dominant factor in evaluation is the simulation scores. However, when the execution time transitions from being close to the T_{limit} to slightly exceeding it, the score will decrease exponentially. To ensure that V_t approaches nearly 0 at $t = 1.2 \times T_{limit}$, the value of α can be determined using the following formula: $\alpha = -\frac{\ln(\epsilon)}{1.2 \times T_{limit} - 0.9 \times T_{limit}} = -\frac{\ln(\epsilon)}{0.3 \times T_{limit}}$ where ϵ is a decimals close to 0.

Backpropagation. After calculating the value in the evaluation phase, the value is propagated back to the root, updating the statistical data of each traversed node.

Terminal Condition. To prevent unnecessary costs due to over-optimization, we implement a simple termination mechanism. If the value does not improve over k consecutive iterations, or if the total time exceeds 1.2 times the limit, the iteration stops. If this mechanism is not triggered, the process concludes after K iterations.

2.2 Automatic Prompt Optimization

We have obtained an optimized workflow structure based on the workflow structure search. In this section, we focus on optimizing the agent prompt within the workflow. We propose a novel self-optimization framework, Joint-Prompt-Optimizer, aimed at achieving both efficient and accurate prompt optimization. The framework is comprised of three core modules: 1) Prompt Build, 2) Example Selection and Chain-of-Thought Generation, and 3) Joint Optimization of Instructions and Examples. As shown in Figure 2, these modules collaboratively generate, optimize, and refine prompts, enhancing task execution accuracy and efficiency.

Preliminary. Given a training dataset $D_{train} = \{(q_i, a_i)\}_{i=1}^n$ of questions q_i and answers a_i , along with a test dataset D_{test} and a

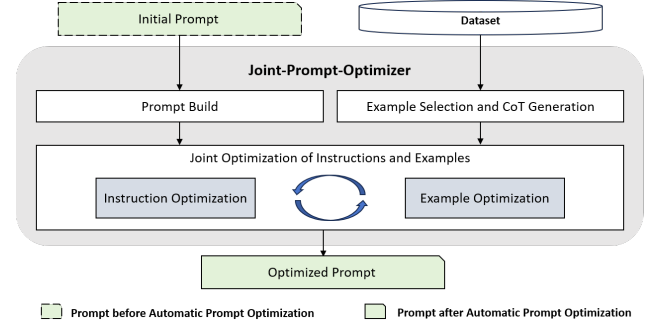


Figure 2: The framework of the Joint-Prompt-Optimizer

score function $s(\cdot)$, the goal is to optimize a prompt p to maximize the task performance. The optimization objective is to find the optimal prompt p^* that maximizes the expected score over D_{test} :

$$p^* = \arg \max_p \mathbb{E}_{(q_i, a_i) \sim D_{test}} [s(M_t(q_i, p), a_i)], \quad (3)$$

Where the prompt p consists of two key components: background knowledge, which provides essential context, problem-solving strategies, and workflows for task comprehension, and use-case knowledge, which includes specific questions and solutions to guide the model's learning. A model M_e is employed to summarize feedback and iteratively refine the prompt, optimizing task performance.

Prompt Build. To optimize prompt generation, we propose a Prompt Build mechanism, inspired by Meta Prompting techniques [11]. This mechanism employs a meta-prompt to guide the creation of an initial prompt. The Prompt Build module combines a unified input I —which integrates the scene description, input-output rules, background information, and an Initial prompt $p^{initial}$ —with the meta-prompt p^{meta} to generate the constructed prompt P , $P = M_e(I, p^{meta}, p^{initial})$, where I encapsulates all task-specific details, and p^{meta} provides a predefined template for structuring the prompt. The objective is to produce a high-quality initial prompt P that serves as a foundation for further refinement and optimization.

Example Selection and Chain-of-Thought Generation. In this module, we utilize an instructive meta-prompt to select the most representative and relevant examples from the training set. The input to this process consists of a prompt p , samples, and associated metadata p^{meta} , while the output is a set of exemplars. The process can be formally expressed as $E_t = M_e(p, B; p^{meta})$, where $E_t = \{e_i\}_{i=1}^m = \{(q_i, a_i, cot_i)\}_{i=1}^m$ represents a set of exemplars e_i , each consisting of a question q_i , an answer a_i , and an optional chain of thought cot_i . The set $B = \{(q_i, \tilde{a}_i, a_i)\}_{i=1}^n$ contains erroneous samples, where q_i is the question, \tilde{a}_i is the model's erroneous answer, and a_i is the true answer.

Joint Optimization of Instructions and Examples. In the final optimization phase, both prompt instructions and selected examples are iteratively refined to enhance performance beyond independent optimization. Inspired by ProTeGi [12], this process involves aggregating feedback from erroneous samples $B = (\tilde{q}_i, \tilde{a}_i)$, where the task model's response diverges from the expected answer \tilde{a}_i . The feedback $F_t = M_e(p_t, B; p^{meta})$ is generated using a meta-prompt p^{meta} , guiding the optimizer in refining the prompt instructions. For multi-stage pipeline optimization without gold labels for intermediate stages, feedback is generated from performance scores and optimization history, following the principles of OPRO[13]. The

feedback $F_t = M_e(p_t; p^{\text{meta}}, S, I)$, where S captures past scores and prompts, and I integrates scene descriptions, input-output rules, and background information. This guides the optimizer to adjust model behavior, refining intermediate stages to align with the overall task, even without predefined ground truth labels. The prompt optimizer then adjusts the prompt p_t based on the feedback, producing an updated prompt $p_{t+1} = M_e(p_t, B, f_t; p^{\text{meta}})$, where $f_t \in F_t$ represents selected feedback, and p^{meta} directs the optimization. This iterative process improves the task model's performance. Similarly, example optimization leverages error-driven self-reflection to generate more diverse and task-relevant examples.

2.3 Dynamic Workflow Routing

Based on workflow structure search and automatic prompt optimization, we observed that data from different sources require distinct workflow configurations for training and optimization. For example, while complex workflows generally produce better results, some datasets perform significantly better with a simple two-stage workflow. To adapt flexibly to varying data characteristics, we propose a dynamic routing mechanism. Inspired by ensemble learning, this mechanism integrates multiple candidate workflows and dynamically selects the most suitable workflow route based on the current data characteristics. This approach effectively reduces biases that may arise from using a single workflow and enhances the robustness and generalization capability of the system. Specifically, through experiments on different datasets, we determine the most suitable workflow for each data type based on test results and combine it with other workflows via the routing mechanism. As a result, the generated workflow can adjust according to the input data, automatically selecting the optimal workflow for inference.

3 Experiment

3.1 User Modeling Task and Datasets

The user modeling task is designed by AgentSociety Challenge @ WWW 2025. The goal of our task is to predict the rating stars and generate a review for a given item (a business or a product) by a specific user. For this task, the agent is provided with historical data about both the user and the item, including previous review information. The agent is designed to capture user behavior and item features in order to simulate real user activity. For efficiency, the average time for processing each task is limited to 1 minute. We evaluated our model on three different datasets: Amazon [14], Goodreads [15] and Yelp¹. Additionally, to optimize the prompt, we split each dataset into a training set (280 reviews) and a test set (120 reviews). All results were evaluated on the test set.

3.2 Evaluation Metrics

Our evaluation focuses on two primary components: preference estimation (PE) and review generation (RG). For the preference estimation, we use the Mean Absolute Error (MAE) to measure the deviation of predicted star ratings from the ground truth. The score for the rating stars is given by $PE = 1 - \frac{1}{N} \sum_i^N |\hat{s}_{ni} - s_{ni}|$, where N is the total number of reviews and \hat{s}_{ni} and s_{ni} are the normalized predicted and ground truth star ratings, respectively. For the review

Table 1: User Modeling Track Results for AgentSociety Challenge @ WWW 2025

Rank	PE	RG	SIM	REAL	OP
1st	0.8587	0.8259	0.9011	0.8031	0.8423
2nd (ours)	0.8613	0.8207	0.9009	0.8010	0.8410
3rd	0.8532	0.8271	0.8943	0.8040	0.8401
4th	0.8573	0.8201	0.9017	0.7967	0.8387
5th	0.8505	0.8251	0.8971	0.7982	0.8378

generation, we evaluated three main aspects: 1. Emotional Tone Error e^{emo} : MAE of normalized emotion score [16]; 2. Sentiment Attitude Error e^{senti} : MAE of normalized sentiment scores, which is evaluated by using the *nlTK* package; and 3. Topic Relevance Error [17] sim^{topic} : Cosine similarity between the review's text embedding and the real topics. The review generation score is determined by $RG = 1 - (0.25 \times e^{\text{emo}} + 0.25 \times e^{\text{senti}} + 0.5 \times sim^{\text{topic}})$. The overall performance (OP) is then evaluated by the average of the preference estimation and review generation $OP = \frac{1}{2}(PE + RG)$.

3.3 Overall Performance

The agent was evaluated using the Qwen2.5-72B-instruct model as part of the AgentSociety Challenge @ WWW 2025, which incorporated 40% simulation data and 60% real data for final evaluation. And our proposed agent earned second place in the final phase with an overall score of 0.8410. The detailed performance is listed in Table 1. We achieved first place on the performance estimation with a score of 0.8613.






3.4 Ablation Study

Workflow Search Results We initiated the structure search by using the baseline provided by the challenge. We set T_{limit} in Equation 2 to 1 minute, in accordance with the challenge requirements, and chose $w = 0.9$ to place more emphasis on the simulation scores. The structure was first selected using the training set and then evaluated using the test set. For each dataset, the search was performed to generate 5 optimized versions, and each version is limited to 10 iterations. The results of the workflow search are presented in Figure 3, with the search conducted separately for each dataset. The results indicate that, among the three datasets, the agent benefits most from the Goodreads dataset. Additionally, further optimization on the training set leads to overfitting on the Amazon dataset, resulting in a decline in performance on the test set. Table 2 presents the details of each version using the Goodreads dataset. In the second version, which employs five agents for the task, V_{final} decreases due to the prolonged execution time. Subsequent optimizations reduce the number of agents to improve time efficiency. Ultimately, the optimized workflow achieves an overall performance of 0.8903 while maintaining efficiency.

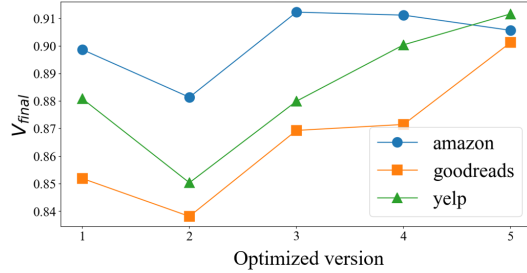
Prompt Optimization Results After selecting the optimal workflow for each dataset, we proceeded with prompt optimization individually. To avoid exceeding the token limits for the agent, we retained two examples for each workflow. The experimental results are presented in Table 3, which demonstrates that the overall performance improves after optimization across all datasets.

¹<https://www.yelp.com/dataset>

Table 2: Workflow search details on Goodreads dataset.

Version No.	Overall Performance	Execution Time (min/task)	V_{final}	Agent Num.	Graph
1	0.8355	0.1988	0.8519	1	
2	0.8485	0.9278	0.8382	5	
3	0.8549	0.7014	0.8694	3	
4	0.8573	0.8972	0.8715	4	
5	0.8903	0.8741	0.9013	3	

Note. ●: predict rating stars and generate review; ●: review summary agent for target item; ●: item information summary agent; ●: user information summary agent; ●: user review summary agent. The prompt for each agent may vary across different versions.

**Figure 3: Workflow search results on the test set.****Table 3: Automatic Prompt Optimization Results**

Optimization	Amazon	Goodreads	Yelp
Before	0.9008	0.8786	0.9068
After	0.9016	0.8849	0.9085

Table 4: Detailed Performance of Non-Routing

	PE	RG	OP
Single Workflow	0.8961	0.8930	0.8945
Workflow Router	0.9011	0.8955	0.8983

Workflow Router From the results of the agent flow optimization, we noticed that different agent flows result in different performances on three datasets. For example, the workflow with an extra agent shows better results on the Goodreads dataset, which summarizes the information of the provided user for the user reviews. While on the other two datasets, the results are better when no process is done on the user reviews. Therefore, for better overall results, we select the best agent flow for each dataset. The results are shown in Table. 4. The results show that with the workflow router, both the performance estimation and the review generation are improved.

4 Conclusion

In this paper, we introduce a novel collaborative optimization approach for workflow agents in user behavior modeling. We utilize Monte Carlo Tree Search (MCTS) to automatically search for the optimal workflow and apply a automatic prompt optimization strategy on the refined workflow. Additionally, we introduce a dynamic Workflow Router to enhance overall performance across three real-world datasets. Our approach achieved second place overall and first place in star-rating prediction in the user modeling track of the

AgentSociety Challenge @ WWW 2025, demonstrating its strong potential for user behavior modeling.

References

- [1] Guy Azov, Tatiana Pelc, Adi Fledel Alon, and Gila Kamhi. Self-improving customer review response generation based on llms. *arXiv preprint arXiv:2405.03845*, 2024.
- [2] Binzong Geng, Zhaoxin Huan, Xiaolu Zhang, Yong He, Liang Zhang, Fajie Yuan, Jun Zhou, and Linjian Mo. Breaking the length barrier: Llm-enhanced ctr prediction in long textual user behaviors. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2311–2315, 2024.
- [3] Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. Llm-esr: Large language models enhancement for long-tailed sequential recommendation. *Advances in Neural Information Processing Systems*, 37:26701–26727, 2025.
- [4] Akira Kasuga and Ryo Yonetani. Cxsimulator: A user behavior simulation using llm embeddings for web-marketing campaign assessment. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3817–3821, 2024.
- [5] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.
- [6] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitit, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.
- [7] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*, 2022.
- [8] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.
- [9] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFLOW: Automating agentic workflow generation. In *ICLR*, 2025.
- [10] Levente Kocsis and Csaba Szepesvari. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML)*, 2006.
- [11] Yifan Zhang. Meta prompting for agi systems. *arXiv preprint arXiv:2311.11482*, 2023.
- [12] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.
- [13] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024.
- [14] Y Hou, J Li, Z He, A Yan, X Chen, and JJ McAuley. Bridging language and items for retrieval and recommendation (2024). *CoRR, abs/2403.03952*.
- [15] Mengting Wan and Julian J. McAuley. Item recommendation on monotonic behavior chains. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 86–94. ACM, 2018.
- [16] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *arXiv preprint arXiv:2010.12421*, 2020.
- [17] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.