Search Swarm: Multiagent Large Language Models Framework for E-commerce Product Search

Nagim Isyanbaev[1] and **Ilya Makarov**[2,3,4]
[1]Innopolis University
[2]AIRI
[3]Research Center for AI, Innopolis
[4]ISP RAS
n.isyanbaev@innopolis.university

Abstract

Search engines are vital for online e-commerce but often struggle with long, detailed queries. We introduce Search Swarm, a novel multi-agent system designed to improve search engine navigation on platforms like Amazon by accurately locating relevant products based on user instructions. Search Swarm employs multiple large language model (LLM) agents, each with a specific role: query planner, searcher, critic, and attribute selector. These agents collaborate to generate search queries, evaluate results, and identify the best product options tailored to users' needs. Our framework outperforms existing methods like ReAct and Reflexion in the WebShop environment, achieving a reward score of 62.64, compared to scores of 54.1, 59.8, 61.5, and 58.2 for other approaches. Furthermore, in a comparison with a basic rule-based method on Amazon, Search Swarm achieved a score 38.71 points higher and a 41% greater success rate, demonstrating its superior ability to provide relevant product matches over traditional search engines.

1 Introduction

E-commerce search engines have been vital in helping customers find products, yet they encounter several challenges. One major issue is their difficulty in handling extensive and detailed queries that include factors such as buying intent, price range, and user age. As a result, users often have to depend on keyword matching and sift through a multitude of suggestions. Additionally, these search engines frequently overlook user-defined attributes, such as the quantity of items in a pack, necessitating manual selection by the user.

To mitigate these challenges, platforms like Amazon and eBay provide extra filters for attributes such as age group, price, and size. However, the sheer variety of product configurations makes thorough filtering impractical. Another approach involves using LLM-powered chatbots that can interpret natural language queries, but these are limited to search functionalities and do not facilitate the selection of product attributes. In contrast, LLM-based autonomous web agents [Yao *et al.*, 2023; Zhang *et al.*, 2024; Furuta *et al.*, 2024; Shinn *et al.*, 2023; Zhou *et al.*, 2024; Zhao *et al.*, ; Prasad *et al.*, ; Kagaya *et al.*, 2024; Wang *et al.*, 2024; Fu *et al.*, 2024; Yao *et al.*, 2024; Ma *et al.*, 2023; Schroeder *et al.*, 2024] offer a more interactive experience, treating shopping websites as environments to search for products and select their attributes.

Despite their potential, most LLM-based web agents tend to focus on general navigation rather than addressing the specific requirements of e-commerce searches, which diminishes their effectiveness in retrieving relevant product information. To fill this gap, we developed Search Swarm, a multi-agent LLM framework that incorporates task-specific agents and simulates human-like thought processes within its search pipeline.

We assessed Search Swarm using the WebShop simulation benchmark [Yao *et al.*, 2022] and tested it on Amazon through a sim-to-real transfer method, evaluating its capability to retrieve products based on user instructions. This evaluation involved calculating a reward function and success rate, with our results compared to existing solutions to highlight the framework's effectiveness.

The primary contribution of this study is a framework that enhances the functionality of e-commerce search engines by delivering more relevant results for complex queries and making informed decisions about product attributes. Our experiments demonstrate the efficiency of the proposed framework and provide a comparative analysis that validates its superiority.

2 Related Work

In this chapter, we examine contemporary e-commerce product search engines, highlighting their limitations when handling long and detailed queries, as well as existing web agent methods that attempt to address these challenges. We also identify gaps in the current solutions.

2.1 E-commerce Search Engines

Modern search engines enhance user queries before ranking them to improve relevance and recall, particularly for lengthy queries. Modifications may include grammar corrections and personalization. For instance, Amazon's Query Understanding (QU) modifies queries to align with its search grammar, traditionally relying on rules and handcrafted features, which struggle with long-tail queries [Luo *et al.*, 2024]. Similarly, eBay's Cassini algorithm rewrites queries that yield few results by dropping or replacing terms. However, dropping essential terms can lead to incomplete results, and replacing terms using bigram language models [Tan *et al.*, 2017] can falter with longer contexts.

2.2 LLM Agents

Large language models (LLMs) such as GPT-4 [OpenAI, 2024] and Llama 3 [Grattafiori *et al.*, 2024] excel in various tasks. The Retrieval Augmented Generation (RAG) [Lewis *et al.*,] approach allows LLMs to access new information without needing retraining, expanding their applications, including functioning as agents in complex environments like WebShop, where actions are based on textual state representations. Notable examples of LLM agents include ReAct [Yao *et al.*, 2023], which integrates reasoning into action generation, xLAM [Zhang *et al.*, 2024], which emphasizes task-specific models, and WebGUM [Furuta *et al.*, 2024], a multimodal agent that utilizes transformers for web navigation.

2.3 Multi-agent Frameworks

Multi-agent frameworks, where agents assume distinct roles, tend to outperform single-agent systems by distributing tasks and minimizing errors.

For example, ADaPT [Prasad *et al.*,] breaks tasks into subtasks when the main agent cannot complete them, using a controller to manage execution and create new instances for each subtask. TDAG [Wang *et al.*, 2024] similarly decomposes tasks, assigning them to subagents that adapt and further break down tasks if time limits are exceeded, while also expanding a skill library. THREAD [Schroeder *et al.*, 2024] treats task execution as thread management, allowing an LLM to spawn child threads for subtasks, thus providing flexible task handling.

Frameworks like Reflexion [Shinn *et al.*, 2023], ExpeL [Zhao *et al.*,], RAP [Kagaya *et al.*, 2024], AutoGuide [Fu *et al.*, 2024], and Retroformer [Yao *et al.*, 2024] utilize feedback loops for self-reflection and learning, refining actions through iterative feedback and memory systems.

Additionally, LASER [Ma *et al.*, 2023] and LATS [Zhou *et al.*, 2024] employ data structures for task completion, with LASER modeling tasks as state-space exploration and LATS using Monte Carlo Tree Search (MCTS) for planning and reasoning, offering feedback on errors.

While the reviewed solutions present a variety of agent and multi-agent frameworks, they primarily target general web tasks and lack architectures specifically designed for e-commerce searches. To address this gap, we developed Search Swarm, a multi-agent framework optimized for e-commerce product searches, as multi-agent approaches have demonstrated superior performance in this area.

3 Framework

In our framework, we segmented the product search process into three sequential phases: Planning, Data Collection, and Final Decision (refer to algorithms 1, 3, and 2 for the workflow Figure 1). Each phase involves agents that perform specific subtasks based on their designated roles. These agents are large language models (LLMs) that take textual input, enhanced by role-specific prompts, and produce outputs that adhere to a defined JSON schema. While we do not specify the language model in this chapter, we will clarify it in the experiments section. We ensure the validity of the generated sequences using Structured Generation for OpenAI models and the Outlined Python library for Hugging Face models. These tools limit the tokens available for the LLM to sample from in each iteration, ensuring that the responses conform to a specific schema. The complete implementation of the agents pipeline and their prompts can be found in the source code.[1].

The rationale for structuring the search process into these three phases is to replicate human behavior during a search. Typically, individuals start by formulating a query, input it into a search engine, and then browse through the suggested options before selecting the most suitable product from the displayed results.

We designed the prompts such that LLMs adhere to their assigned agent roles. During the development of the framework, we made slight modifications to the prompts to mitigate common errors made by the agents. For instance, we aimed to prevent them from favoring child products solely because they matched the instructions when such products were not explicitly requested, or from overlooking the fact that a product's color is a customizable option when selecting relevant items.

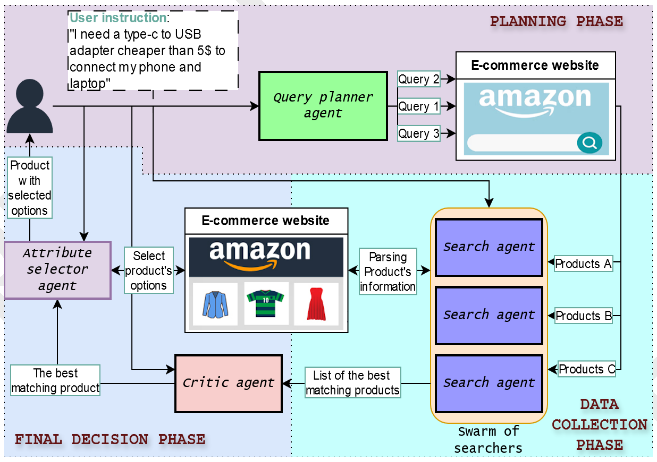

Figure 1: The representation of workflow of the Search Swarm framework. Everything separated into three phases.

Algorithm 1 Planning Phase

```
Input: A user instruction $I$
Output: Product lists for each query $L$
  $Q \leftarrow$ QueryPlannerAgent$(I)$
  $L \leftarrow \emptyset$
  for all $q \in Q$ do
    $O \leftarrow$ Environment.Step("Search[${q}]")
    $L \leftarrow L \cup \{(\text{ParseProductList}(O, q), q)\}$
  end for
```

[1]Code: https://github.com/Nagim123/Multiagent-Web-Search-Search-Swarm

Algorithm 2 Final Decision Phase

```
Input: Top-$k$ relevant products $R$, a user instruction $I$
Output: The most relevant product $x$, best option values $d$
  $x \leftarrow$ CriticAgent$(R, I)$
  Environment.Step(”Search[” + $x$.query + ”]”)
  $O \leftarrow$ Environment.Step(”Click[” + $x$.id + ”]”)
  $d \leftarrow$ AttributeSelectorAgent$(x,$ ParseOptions$(O), I)$
```

Algorithm 3 Data Collection Phase

```
Input: Set of product lists $L$, the count of relevant products $k$, a user instruction $I$
Output: Top-$k$ relevant products $R$
  $R \leftarrow \emptyset$
  $A \leftarrow$ {”Click[Description]”, ”Click[Attributes]”, ”Click[Reviews]”, ”Click[Features]”}
  for all $l \in L$ do
    $J \leftarrow \emptyset$
    for all $p \in l_1$ do
      Environment.Step(”Search[${$l_2$}]”)
      Environment.Step(”Click[${p.id}]”)
      $D \leftarrow \emptyset$
      for all $a \in A$ do
        $D \leftarrow D \cup$ {Environment.Step$(a)$}
        Environment.Step(”Click[< Prev]”)
      end for
      Environment.Step(”Click[< Prev]”)
      $J \leftarrow J \cup$ BuildProductJson$(p, D)$
    end for
    $R \leftarrow R \cup$ SearchAgent$(J, k, I)$
  end for
```

4 Evaluation

We conducted two experiments utilizing the Search Swarm framework: one aimed at validating its effectiveness against existing methods, and the other focused on evaluating its performance on Amazon.

4.1 WebShop Environment

The first experiment was carried out using the WebShop benchmark [Yao *et al.*, 2022], which includes 100 random user instructions. WebShop simulates online shopping with over a million Amazon products and 12,000 crowd-sourced instructions. The agents' goal is to locate and select products that align with user instructions, measuring success through a success rate and reward metrics. The reward score reflects the relevance of the selected product, while the success rate indicates how frequently the maximum reward score—representing a perfect match to the user's goal—is achieved. In this context, we compared Search Swarm to ReAct, Reflexion, LATS, and ADaPT, using Llama-3-8b as the backbone model for consistency. Search Swarm achieved the highest reward score of 62.64, surpassing the other methods, and matched ADaPT's top success rate of 35% Table 1).

4.2 Sim-to-Real Transfer to Amazon

To assess real-world applicability, we implemented a sim-to-real transfer on Amazon, treating interactions similarly to

Method	SR	Reward
Search Swarm	**35**	**62.64**
ADaPT	**35**	58.2
LATS	34	61.5
Reflexion	32	59.8
ReAct	31	54.1

Table 1: Reward scores and success rates (SR) for 100 user instructions in the WebShop environment. All methods were evaluated using the LLama-3-8b model.

Method	SR	Reward
Search Swarm	**60**	**84.51**
Rule based	19	45.8

Table 2: Reward scores and success rates (SR) for 100 user instructions on amazon.com using sim-to-real transfer. Comparison between the rule-based approach and Search Swarm with GPT-4o.

those in WebShop by converting HTML pages into observations and actions into button clicks. Using GPT-4o, we compared Search Swarm to a rule-based baseline that input user instructions into the search engine and selected the first match. Search Swarm outperformed this baseline, achieving a success rate of 60% and a reward score of 84.51, compared to the baseline's 19% and 45.8, respectively (Table 2).

4.3 Results

Search Swarm demonstrated superior performance in both simulated and real-world environments, outperforming other methods in the WebShop benchmark and significantly enhancing product search and selection on e-commerce platforms. These results underscore its potential as a powerful tool for improving online search experiences.

5 Conclusion

Search engines are crucial for locating products on e-commerce websites, yet they encounter difficulties in processing detailed queries and automatically selecting attributes. Several solutions have been proposed, with web agents emerging as a particularly promising approach. However, much of the research has concentrated on general-purpose agents, overlooking the specific requirements of task-oriented searches. To fill this gap, we developed Search Swarm, a framework that incorporates task-specific agents and simulates human-like thought processes within its search pipeline. We conducted two experiments to validate its efficiency and demonstrate its adaptability. Our results indicate that Search Swarm significantly enhances the user experience, representing a significant advancement in the development of intuitive and effective search system extensions.

Acknowledgements

This work was supported by the Ministry of Economic Development of the RF (code 25-139-66879-1-0003).

References

[Fu *et al.*, 2024] Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated Generation and Selection of Context-Aware Guidelines for Large Language Model Agents. In *Workshop on LLMs and Cognition Poster*, 2024. Accessed: November 22, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.08978.

[Furuta *et al.*, 2024] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In *In-Person Poster presentation*, 2024. Accessed: November 20, 2024. [Online]. Available: https://openreview.net/forum?id=efFmBWioSc.

[Grattafiori *et al.*, 2024] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and et al. Alex Vaughan. The llama 3 herd of models, 2024.

[Kagaya *et al.*, 2024] Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. RAP: Retrieval-Augmented Planning with Contextual Memory for Multimodal LLM Agents. In *Workshop Open-World Agents Poster*, 2024. Accessed: November 22, 2024. [Online]. Available: https://arxiv.org/abs/2402.03610.

[Lewis *et al.*,] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, and et al. Rocktäschel, Tim. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*. 2020, pp. 9459–9474. Accessed: November 20, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[Luo *et al.*, 2024] Chen Luo, Xianfeng Tang, Hanqing Lu, Yaochen Xie, Hui Liu, Zhenwei Dai, Limeng Cui, Ashutosh Joshi, Sreyashi Nag, Yang Li, Zhen Li, Rahul Goutam, Jiliang Tang, Haiyang Zhang, and Qi He. Exploring query understanding for amazon product search, 2024.

[Ma *et al.*, 2023] Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. LASER: LLM Agent with State-Space Exploration for Web Navigation. In *Workshop: Foundation Models for Decision Making*, 2023. Accessed: November 20, 2024. [Online]. Available: https://openreview.net/forum?id=sYFFyAILy7.

[OpenAI, 2024] OpenAI. Gpt-4 technical report, 2024.

[Prasad *et al.*,] Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 2024, pp. 4226–4252. Accessed: November 21, 2024. [Online]. Available: https://doi.org/10.18653/v1/2024.findings-naacl.264.

[Schroeder *et al.*, 2024] Philip Schroeder, Nathaniel Morgan, Hongyin Luo, and James Glass. Thread: Thinking deeper with recursive spawning, 2024.

[Shinn *et al.*, 2023] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023.

[Tan *et al.*, 2017] Zehong Tan, Canran Xu, Mengjie Jiang, Hua Yang, and Xiaoyuan Wu. Query rewrite for null and low search results in ecommerce. In *eCOM@SIGIR*, 2017.

[Wang *et al.*, 2024] Yaoxiang Wang, Zhiyong Wu, Junfeng Yao, and Jinsong Su. TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation, 2024.

[Yao *et al.*, 2022] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc., 2022.

[Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *In-Person Poster presentation*, 2023. Accessed: November 20, 2024. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X.

[Yao *et al.*, 2024] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, and et al. Devansh Arpit. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. In *Spotlight*, 2024. Accessed: November 23, 2024. [Online]. Available: https://openreview.net/forum?id=KOZu91CzbK.

[Zhang *et al.*, 2024] Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, and et al. Haolin Chen. xlam: A family of large action models to empower AI agent systems. *CoRR*, abs/2409.03215, 2024.

[Zhao *et al.*,] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence*. 2024, pp. 19632–19642. Accessed: November 21, 2024. [Online]. Available: https://doi.org/10.1609/aaai.v38i17.29936.

[Zhou *et al.*, 2024] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings* *of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR, 21–27 Jul 2024.