PaSa: An LLM Agent for Comprehensive Academic Paper Search

Yichen He[*1] **Guanhua Huang**[*1] **Peiyuan Feng**[1] **Yuan Lin**[†1]
Yuchen Zhang[1] **Hang Li**[1] **Weinan E**[2]

[1]ByteDance Seed [2]Peking University

{hyc,huangguanhua,fpy,linyuan.0}@bytedance.com,
{zhangyuchen.zyc,lihang.lh}@bytedance.com, weinan@math.pku.edu.cn

Abstract

We introduce PaSa, an advanced **Pa**per **Sea**rch agent powered by large language models. PaSa can autonomously make a series of decisions, including invoking search tools, reading papers, and selecting relevant references, to ultimately obtain comprehensive and accurate results for complex scholar queries. We optimize PaSa using reinforcement learning with a synthetic dataset, AutoScholarQuery, which includes 35k fine-grained academic queries and corresponding papers sourced from top-tier AI conference publications. Additionally, we develop RealScholarQuery, a benchmark collecting real-world academic queries to assess PaSa performance in more realistic scenarios. Despite being trained on synthetic data, PaSa significantly outperforms existing baselines on RealScholarQuery, including Google, Google Scholar, Google with GPT-4o for paraphrased queries, ChatGPT (search-enabled GPT-4o), GPT-o1, and PaSa-GPT-4o (PaSa implemented by prompting GPT-4o). Notably, PaSa-7B surpasses the best Google-based baseline, Google with GPT-4o, by 37.78% in recall@20 and 39.90% in recall@50, and exceeds PaSa-GPT-4o by 30.36% in recall and 4.25% in precision. Model, datasets, and code are available at https://github.com/bytedance/pasa.

Demo: https://pasa-agent.ai

1 Introduction

Academic paper search lies at the core of research yet represents a particularly challenging information retrieval task. It requires long-tail specialized knowledge, comprehensive survey-level coverage, and the ability to address fine-grained queries. For instance, consider the query: *"Which studies have focused on non-stationary reinforcement learning using value-based methods, specifically UCB-based algorithms?"* While widely used academic search systems like Google Scholar are effective for general queries, they often fall short when addressing these complex queries (Gusenbauer and Haddaway, 2020). Consequently, researchers frequently spend substantial time conducting literature surveys (Kingsley et al., 2011; Gusenbauer and Haddaway, 2021).

The advancements in large language models (LLMs) (OpenAI, 2023; Anthropic, 2024; Gemini, 2023; Yang et al., 2024) have inspired numerous studies leveraging LLMs to enhance information retrieval, particularly by refining or reformulating search queries to improve retrieval quality (Alaofi et al., 2023; Li et al., 2023; Ma et al., 2023; Peng et al., 2024). In academic search, however, the process goes beyond simple retrieval. Human researchers not only use search tools, but also engage in deeper activities, such as reading relevant papers and checking citations, to perform comprehensive and accurate literature surveys.

In this paper, we introduce PaSa, a novel paper search agent designed to mimic human behavior for comprehensive and accurate academic paper searches. As illustrated in Figure 1, PaSa consists of two LLM agents: the Crawler and the Selector. For a given user query, the Crawler can autonomously collect relevant papers by utilizing search tools or extracting citations from the current paper, which are then added to a growing *paper queue*. The Crawler iteratively processes each paper in the paper queue, navigating citation networks to discover increasingly relevant papers. The Selector carefully reads each paper in the paper queue to determine whether it meets the requirements of the user query. We optimize PaSa within the AGILE, a reinforcement learning (RL) framework for LLM agents (Feng et al., 2024).

[*]Equal contribution.
[†]Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11663–11679

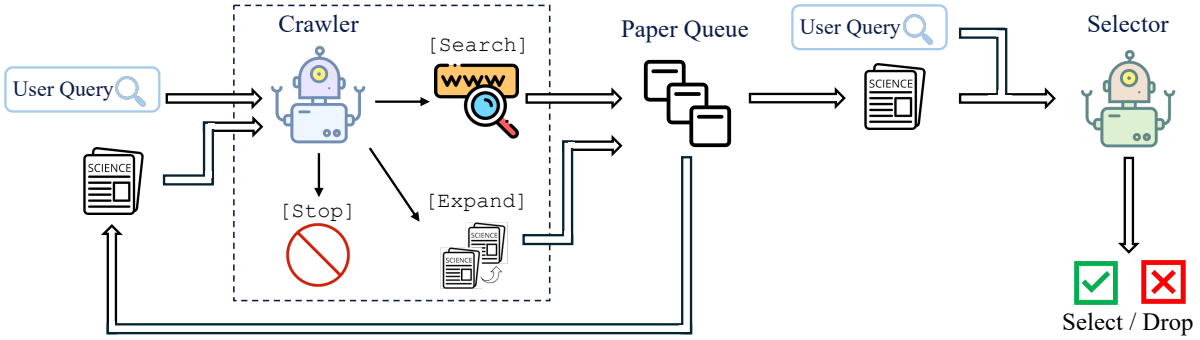

Figure 1: Architecture of PaSa. The system consists of two LLM agents, Crawler and Selector. The Crawler processes the user query and can access papers from the paper queue. It can autonomously invoke the search tool, expand citations, or stop processing of the current paper. All papers collected by the Crawler are appended to the paper queue. The Selector reads each paper in the paper queue to determine whether it meets the criteria specified in the user query.

Effective training requires high-quality academic search data. Fortunately, human scientists have already created a vast amount of high-quality academic papers, which contain extensive surveys on a wide range of research topics. We build a synthetic but high-quality academic search dataset, AutoScholarQuery, which collects fine-grained scholar queries and their corresponding relevant papers from the related work sections of papers published at ICLR 2023 [1], ICML 2023 [2], NeurIPS 2023 [3], ACL 2024 [4], and CVPR 2024 [5]. AutoScholarQuery includes 33,511 / 1,000 / 1,000 query-paper pairs in the training / development / test split.

Although AutoScholarQuery only provides query and paper answers, without demonstrating the path by which scientists collect the papers, we can utilize it to perform RL training to improve PaSa. In addition, we design a new session-level PPO (Proximal Policy Optimization (Schulman et al., 2017)) training method to address the unique challenges of the paper search task: 1) sparse reward: The papers in AutoScholarQuery are collected via citations, making it a smaller subset of the actual qualified paper set. 2) long trajectories: The complete trajectory of the Crawler may involve hundreds of papers, which is too long to directly input into the LLM context.

To evaluate PaSa, besides the test set of AutoScholarQuery, we also develop a benchmark, RealScholarQuery. It contains 50 real-world academic queries with annotated relevant papers, to assess PaSa in real-world scenarios. We compare PaSa with several baselines including Google, Google Scholar, Google paired with GPT-4o for paraphrased queries, ChatGPT (search-enabled GPT-4o), GPT-o1 and PaSa-GPT-4o (PaSa agent realized by prompting GPT-4o). Our experiments show that PaSa-7b significantly outperforms all baselines. Specifically, for AutoScholarQuery test set, PaSa-7b achieves a 34.05% improvement in Recall@20 and a 39.36% improvement in Recall@50 compared to Google with GPT-4o, the strongest Google-based baseline. PaSa-7b surpasses PaSa-GPT-4o by 11.12% in recall, with similar precision. For RealScholarQuery, PaSa-7b outperforms Google with GPT-4o by 37.78% in Recall@20 and 39.90% in Recall@50. PaSa-7b surpasses PaSa-GPT-4o by 30.36% in recall and 4.25% in precision.

The main contributions of this paper are summarized as follows:

- We introduce PaSa, a comprehensive and accurate paper search agent that can autonomously use online search tools, read entire papers, and navigate citation networks.
- We develop two high-quality datasets for complex academic search, AutoScholarQuery and RealScholarQuery.
- Although PaSa is trained solely on synthetic data, it achieves remarkable real-world performance. Experiments demonstrate that PaSa, built on 7B LLM, significantly outperforms all baselines, including GPT-4 agent, Google-based search, and ChatGPT.

[1] https://iclr.cc/Conferences/2023
[2] https://icml.cc/Conferences/2023
[3] https://neurips.cc/Conferences/2023
[4] https://2024.aclweb.org/
[5] https://cvpr.thecvf.com/Conferences/2024

2 Related Work

LLMs in Scientific Discovery LLMs have been applied across various stages of scientific discovery (Van Noorden and Perkel, 2023; Lu et al., 2024; Messeri and Crockett, 2024; Liao et al., 2024), such as brainstorming ideas (Girotra et al., 2023; Wang et al., 2024a; Baek et al., 2024), designing experiments (M. Bran et al., 2024), writing code (Xu et al., 2022), and generating research papers (Shao et al., 2024; Agarwal et al., 2024; Wang et al., 2024b). One of the most fundamental yet critical stages in research is conducting academic surveys. Despite its importance, current tools like Google Scholar are often insufficient, leading researchers to spend considerable time on literature review tasks (Kingsley et al., 2011; Gusenbauer and Haddaway, 2021, 2020). This challenge motivates us to develop PaSa, an LLM agent designed to autonomously and comprehensively assist researchers in collecting relevant research papers for complex scholarly queries.

LLM Agents LLM Agents combine LLMs with memory, tool use, and planning, enabling them to perform more complex tasks such as personal copilots (Stratton, 2024), travel planning (Gundawar et al., 2024), web operations (Deng et al., 2024), software development (Qian et al., 2023), and scientific experimentation (Bran et al., 2023). In addition to realizing LLM Agents through prompt engineering (Park et al., 2023; Yao et al., 2023; Shinn et al., 2024; Chen et al., 2023), recent research has focused on optimizing and training these agents (Feng et al., 2024; Putta et al., 2024; Liu et al., 2023). Among these efforts, AGILE (Feng et al., 2024), a reinforcement learning framework for LLM agents, allows the joint optimization of all agent skills in an end-to-end manner. In our work, we adopt the AGILE framework to implement PaSa. Specifically, we design a novel session-level PPO algorithm to address the unique challenges of the paper search task, including sparse rewards and long trajectories.

3 Datasets

3.1 AutoScholarQuery

AutoScholarQuery is a synthetic but high-quality dataset of academic queries and related papers, specifically curated for the AI field.

To construct AutoScholarQuery, we began by collecting all papers published at ICLR 2023, ICML 2023, NeurIPS 2023, ACL 2024, and CVPR 2024. For the Related Work section of each paper, we prompted GPT-4o (Hurst et al., 2024) to generate scholarly queries, where the answers to these queries correspond to the references cited in the Related Work section. The prompt used is shown in Appendix H.1. For each query, we retained only the papers that could be retrieved on arXiv[6], using their `arxiv_id` as the unique article identifier in the dataset. We adopt the publication date of the source paper as the query date. During both training and testing, we only considered papers published prior to the query date.

The final AutoScholarQuery dataset comprises 33,551, 1,000, and 1,000 instances in the training, development, and testing splits, respectively. Each instance consists of a query, the associated paper set, and the query date, with queries in each split derived from distinct source papers. Table 1 provides illustrative examples from AutoScholarQuery, while additional dataset statistics are summarized in Table 2.

To evaluate the quality of AutoScholarQuery, we sampled 100 query-paper pairs and assessed the rationality and relevance of each query and the corresponding paper. A qualified query should be meaningful and unambiguous. A qualified paper should match the requirements of the scholarly query. Detailed evaluation criteria are provided in Appendix. A. Three authors manually reviewed each pair, determining that 94.0% of the queries were qualified. Among these qualified queries, 93.7% had corresponding papers that were deemed relevant and appropriate.

3.2 RealScholarQuery

To evaluate PaSa in more realistic scenarios, we constructed RealScholarQuery, a test dataset consisting of 50 real-world research queries. After launching the demo of PaSa, we invited several AI researchers to use the system. From the queries they provided, we randomly sampled a subset of queries and manually filtered out overly broad topics (e.g., "multimodal large language models," "video generation"). Ultimately, we collected 50 fine-grained and realistic queries.

For each query, we first manually gathered relevant papers to the best of our ability. To ensure comprehensive coverage, we then applied multiple methods to retrieve additional papers, including

[6] https://arxiv.org/

Query: Could you provide me some studies that proposed hierarchical neural models to capture spatiotemporal features in sign videos? **Query Date:** 2023-05-02 **Answer Papers:** [1] TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation (2010.05468) [2] Sign Language Translation with Hierarchical Spatio-Temporal Graph Neural Network (2111.07258) **Source:** SLTUnet: A Simple Unified Model for Sign Language Translation, ICLR 2023
Query: Which studies have focused on nonstationary RL using value-based methods, specifically Upper Confidence Bound (UCB) based algorithms? **Query Date:** 2023-08-10 **Answer Papers:** [1] Reinforcement Learning for Non-Stationary Markov Decision Processes: The Blessing of (More) Optimism (2006.14389) [2] Efficient Learning in Non-Stationary Linear Markov Decision Processes (2010.12870) [3] Nonstationary Reinforcement Learning with Linear Function Approximation (2010.04244) **Source:** Provably Efficient Algorithm for Nonstationary Low-Rank MDPs, NeurIPS 2023
Query: Which studies have been conducted in long-form text generation, specifically in story generation? **Query Date:** 2024-01-26 **Answer Papers:** [1] Strategies for Structuring Story Generation (1902.01109) [2] MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models (2010.00840) **Source:** ProxyQA: An Alternative Framework for Evaluating Long-Form Text Generation with Large Language Models, ACL 2024

Table 1: Examples of queries and corresponding papers in AutoScholarQuery.

Conference	$\|P\|$	$\|Q\|$	$Ans(/Q)$	Ans-50	Ans-90
ICLR 2023	888	5204	2.46	2.0	5.0
ICML 2023	981	5743	2.37	2.0	5.0
NeurIPS 2023	1948	11761	2.59	2.0	5.0
CVPR 2024	1336	9528	2.94	2.0	6.0
ACL 2024	485	3315	2.16	2.0	4.0

Table 2: Statistics of AutoScholarQuery. $|P|$ and $|Q|$ represent the total number of papers and queries collected for each conference. $Ans(/Q)$ denotes the average number of answer papers per query. Ans-50 and Ans-90 refers to the 50th and 90th percentiles of answer paper counts per query.

PaSa, Google, Google Scholar, ChatGPT (search-enabled GPT-4o), and Google paired with GPT-4o for paraphrased queries. As these methods also serve as baselines for comparison with PaSa, implementation details are deferred to Section 5.2. The results from all methods were aggregated into a pool of candidate papers. Finally, professional annotators reviewed all candidate papers for each query, selecting those that met the specific requirements of the query to create the final set of relevant papers. Annotation guidelines and quality control procedures are detailed in Appendix. B. The query date of all instances in RealScholarQuery is 2024-10-01. Table 9 in Appendix C provides an example from RealScholarQuery.

The annotators included professors from the Department of Computer Science at a top-tier university in China. On average, each query required the annotators to review 76 candidate papers. We paid $4 per data entry (a query-paper pair), resulting in an average of $304 per query. Given the high annotation cost, we completed this process for only 50 instances. On average, each query is associated with 15.82 answer papers. The 50th percentile of answer counts per query is 9, while the 90th percentile reaches 37.

4 Methodology

4.1 Overview

As illustrated in Figure 1, the PaSa system consists of two LLM agents: Crawler and Selector. The crawler reads the user's query, generates multiple search queries, and retrieves relevant papers. The retrieved papers are added to a *paper queue*. The Crawler further processes each paper in the paper queue to identify key citations worth exploring further, appending any newly relevant papers to the paper queue. The selector conducts a thorough review of each paper in the paper queue to assess whether it fulfills the user's query requirements.

In summary, the Crawler is designed to maximize the recall of relevant papers, whereas the Selector emphasizes precision in identifying papers that meet the user's needs.

4.2 Crawler

In RL terminology, the Crawler performs a token-level Markov Decision Process (MDP). The action space $\mathcal{A}$ corresponds to the LLM's vocabulary, where each token represents an action. The LLM

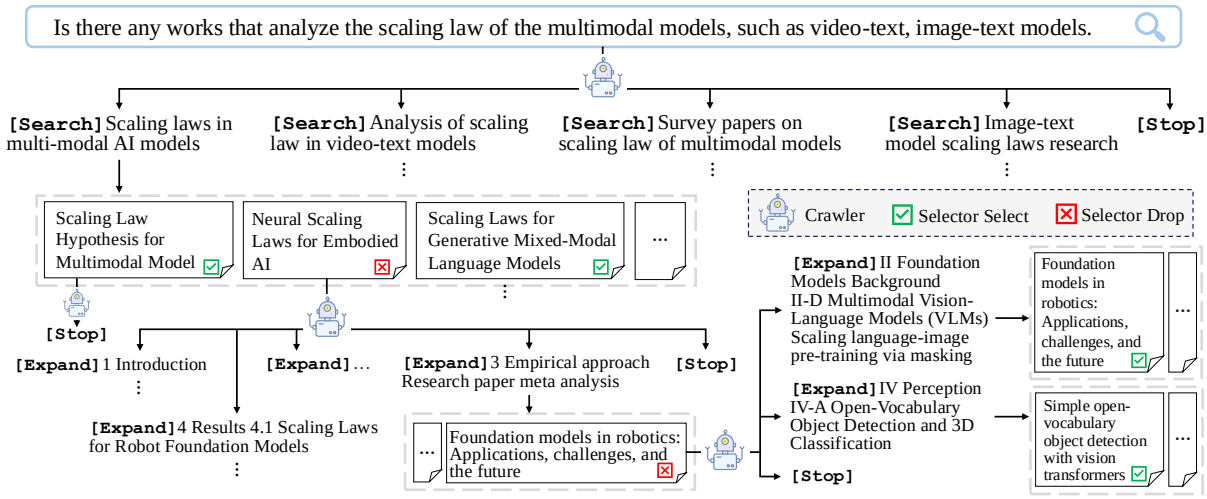

Figure 2: An example of the PaSa workflow. The Crawler runs multiple [Search] using diverse and complementary queries. In addition, the Crawler can evaluate the long-term value of its actions. Notably, it discovers many relevant papers as it explores deeper in the citation network, even when intermediate papers along the path do not align with the user query.

Name	Implementation
[Search]	Generate a search query and invoke the search tool. Append all resulting papers to the paper queue.
[Expand]	Generate a subsection name, then add all referenced papers in the subsection to the paper queue.
[Stop]	Reset the context to the user query and the next paper in the paper queue.

Table 3: Functions of the Crawler.

functions as the policy model. The agent's state is defined by the current LLM context and the paper queue. The Crawler operates with three registered functions, as outlined in Table 3. When an action matches a function name, the corresponding function is executed, further modifying the agent's state.

For example, as Figure 2 shows, the agent begins by receiving a user query, incorporating it into its context, and initiating actions. If the token generated is [Search], the LLM continues to generate a search query, and the agent invokes a search tool to retrieve papers, which are then added to the paper queue. If the token is [Expand], the LLM continues to extract a subsection name from the current paper in its context. The agent then extracts all referenced papers within that subsection, adding them to the paper queue. If the token is [Stop], the agent resets its context to the user query and information of the next paper in the paper queue. This information includes the title, abstract, and an outline of all sections and subsections.

The training process for the Crawler comprises two stages. In the first stage, we generate trajectories for a small subset of the training data and then perform imitation learning (see Appendix D.1 for details). In the second stage, reinforcement learning is applied. The details of the RL training implementation are described below.

Reward Design We conduct RL training on the AutoScholarQuery training set, where each instance consists of a query q and a corresponding paper set $\mathcal{P}$. Starting with a query q, the Crawler generates a trajectory $\tau = (s_1, a_1, \cdots, s_T, a_T)$. At each time step t, we denote the current paper queue as $\mathcal{Q}_t$. Upon taking action a_t, the Crawler appends a set of new papers $(p_1, p_2, \cdots, p_{n_t})$ to the paper queue. If a_t = [Stop], the set is empty and no papers are added.

The reward of executing action a_t in state s_t is defined as

$$r(s_t, a_t) = \alpha \times \sum_{i=1}^{n_t} \mathbb{I}(q, p_i, t) - c(a_t), \quad (1)$$

where $\mathbb{I}(q, p_i, t) = 1$ if p_i matches the query q and is not already in $\mathcal{Q}_t$, and $\mathbb{I}(q, p_i, t) = 0$ otherwise. Here, α is a reward coefficient, and $c(a_t)$ is the cost of action a_t.

The indicator function $\mathbb{I}(q, p_i, t)$ can be determined by checking if p_i belongs to $\mathcal{P} - \mathcal{Q}_t$. However, it is important to note that the AutoScholarQuery may only include a subset of the ground-truth papers, as citations often emphasize a limited

number of key references. If the Crawler receives rewards solely based on matching papers in AutoScholarQuery, this could lead to sparse rewards during training. To mitigate this, we use the Selector as an auxiliary reward model for the Crawler. The revised definition of $\mathbb{I}(q, p_i, t)$ is:

$$\mathbb{I}(q, p_i, t) = \begin{cases} 1, & \text{if (Selector}(q, p_i) = 1 \text{ or } p_i \in \mathcal{P}) \\ & \quad \text{and } p_i \notin \mathcal{Q}_t, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Here Selector$(q, p_i) = 1$ if paper p_i is identified as correct to meet the query q by the Selector, and Selector$(q, p_i) = 0$ otherwise.

RL Training A key challenge in training the Crawler with RL is the significant time required to sample a complete trajectory for a given query. This is due to each `[Search]` or `[Expand]` action adding multiple papers to the paper queue, resulting in hundreds or even thousands of papers in the final paper queue.

To address this issue, we define a *session* as a sub-trajectory that ends with the `[Stop]` action, after which a new session begins. We identify two types of initial states for such sub-trajectories: S_q, containing only the user query, and S_{q+p}, containing both the query and a paper. S_q represents the task's starting point, where the LLM context includes only the query. In contrast, S_{q+p} arises after a `[Stop]` action, where the LLM context is reset to the query and the next paper in the queue.

Formally, we model the Crawler as a policy $\pi_\theta(a_t|s_t)$. We partition the entire trajectory $\tau = (s_1, a_1, \cdots, s_T, a_T)$ into a sequence of sessions: $(\tau_{t_1:t_2-1}, \tau_{t_2:t_3-1}, \cdots)$. Each session is $\tau_{t_i:t_{i+1}-1} = (s_{t_i}, a_{t_i}, \cdots, s_{t_{i+1}-1}, a_{t_{i+1}-1})$, where the initial state s_{t_i} is either belonging to type S_q or S_{q+p}, and the final action $a_{t_{i+1}-1}$ is `[STOP]`.

Sampling such a sub-trajectory from these session initial states is computationally efficient. During the PPO training, at time step $t \in [t_i, t_{i+1})$, we estimate the return in the session using Monte Carlo sampling:

$$\begin{aligned} \hat{R}_t \quad = \quad & \sum_{k=t}^{t_{i+1}-1} \gamma_0^{k-t} \Big[r(s_k, a_k) \\ & + \gamma_1 \sum_{j=1}^{n_k} \hat{V}_\phi(S_{q+p_j}) \Big] - \beta \cdot \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{sft}}(a_t|s_t)} \end{aligned} \tag{3}$$

Here, γ_0 is the in-session discount factor, and γ_1 is the across-session discount factor. $\hat{V}_\phi(\cdot)$ is the value function model to approximate the state value. After executing a_k, the paper queue is updated to include the newly found papers $(p_1, p_2, \cdots, p_{n_k})$. Since the Crawler will subsequently initiate new sessions to process these additional papers, their associated reward-to-go should be incorporated into the return estimate. In addition, we include a per-token KL penalty term from the learned policy π_θ to the initial policy π_{sft} obtained through imitation learning at each token to mitigate over-optimization. This term is scaled by the coefficient β.

Then the advantage function can be approximated by

$$\hat{A}(s_t, a_t) \quad = \quad \hat{R}_t - \hat{V}_\phi(s_t). \tag{4}$$

Finally, the policy and value objectives can be given by

$$\begin{aligned} \mathcal{L}_{\text{policy}}(\theta) = & \mathbb{E}_{\tau' \sim \pi_\theta^{\text{old}}} \Bigg[\min \Bigg(\frac{\pi_\theta(a_t|s_t)}{\pi_\theta^{\text{old}}(a_t|s_t)} \hat{A}(s_t, a_t), \\ & \text{clip}\Big(\frac{\pi_\theta(a_t|s_t)}{\pi_\theta^{\text{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \Big) \hat{A}(s_t, a_t) \Bigg) \Bigg] \end{aligned} \tag{5}$$

and

$$\begin{aligned} \mathcal{L}_{\text{value}}(\phi) = \quad & \mathbb{E}_{\tau' \sim \pi_\theta^{\text{old}}} \Bigg[\max \Bigg(\Big(\hat{\mathrm{R}}_\mathrm{t} - \hat{\mathrm{V}}_\phi(\mathrm{s_t}) \Big)^2, \\ & \Big(\hat{R}_t - \hat{V}_\phi^{\text{clip}}(s_t) \Big)^2 \Bigg) \Bigg], \end{aligned} \tag{6}$$

respectively, where

$$\hat{V}_\phi^{\text{clip}}(s_t) = \quad \text{clip}\Big(\hat{V}_\phi(s_t), V_\phi^{\text{old}}(s_t) - \epsilon, V_\phi^{\text{old}}(s_t) + \epsilon \Big). \tag{7}$$

Here, π_θ^{old} and V_ϕ^{old} is used for sampling and τ' is session trajectory. We then combine these into the unified RL loss:

$$\mathcal{L}_{\text{RL}}(\theta, \phi) = \mathcal{L}_{\text{policy}}(\theta) + \eta \cdot \mathcal{L}_{\text{value}}(\phi) \tag{8}$$

where η is the coefficient of the value objective.

4.3 Selector

The Selector is an LLM agent that takes two inputs: a scholar query and a research paper (including its title and abstract). It generates two outputs: (1) a single decision token d, either "True" or "False", indicating whether the paper satisfies the query, and (2) a rationale $r = (r_1, r_2, ..., r_m)$ containing m tokens that support this decision. The rationale

serves two purposes: enhancing decision accuracy by jointly training the model to generate decisions and explanations, and improving user trust by providing the reasoning in PaSa application.

To optimize training efficiency for the Crawler, the decision token is presented before the rationale, allowing the Selector to act as a single-token reward model during the Crawler training. Additionally, the token probability of the decision token can be used to rank search results. At last, as shown in Table 6, the order of the decision and rationale does not affect the Selector's performance.

We perform imitation learning to optimize the Selector. See Appendix E for training data collection and training details.

5 Experiments

5.1 Experimental Setting

We sequentially trained the Selector and Crawler, both based on the Qwen2.5-7b (Yang et al., 2024), to develop the final agent, referred to as PaSa-7b.

Selector The Selector was fine-tuned using the training dataset described in Appendix E. We conducted supervised fine-tuning for one epoch with a learning rate of 1e-5 and a batch size of 4. The training runs on 8 NVIDIA-H100 GPUs.

Crawler The training process involves two stages. First, we perform imitation learning for 1 epoch on 12,989 training data with a learning rate of 1e-5 and batch size of 4 per device, using 8 NVIDIA H100 GPUs. In the second stage, we apply PPO training. To ensure stability, we first freeze the policy model and train the value model, followed by co-training both the policy and value models. The hyperparameters used during the training process are listed in Table 12 in Appendix D.2.

During imitation learning, the model encounters 5,000 queries, while during the RL training phase, the model processes a total of 16,000 queries. For more details please refer to Appendix D.1 for the imitation learning data construction and Appendix D.2 for the PPO training data sampling.

Implementation of `[Search]` The LLM predicts a query based on the context. Then the agent calls Google[7] with the parameters `site:arxiv.org` and `before:query_date`, restricting search results by source and publication time.

Paper Management We developed a database to manage and restore research papers. PaSa retrieves paper information from the database. If no matching record is found, we use ar5iv[8] to obtain the full paper content, including citations, and then parse this data and store it in the database.

5.2 Baselines and Evaluation

We evaluate our paper search agent on both the test set of AutoScholarQuery and RealScholarQuery. We compare PaSa-7b against the following baselines:

- **Google.** We use Google to search the query directly, with the same parameter settings in Section 5.1.
- **Google Scholar.** Queries are submitted directly to Google Scholar[7], with the same parameter settings in Section 5.1.
- **Google with GPT-4o.** We first employ GPT-4o to paraphrase the scholar query. The paraphrased query is then searched on Google.
- **ChatGPT.** We submit scholar query to ChatGPT[9], powered by search-enabled GPT-4o.
- **GPT-o1.** Prompt GPT-o1 to process the scholar query. Note that it does not have access to external search tools.
- **PaSa-GPT-4o.** Implement PaSa as illustrated in Figure 1 by prompting GPT-4o. It can perform multiple searches, paper reading, and citation network crawling.

We carefully designed prompts for all baselines and they are shown in Appendix H.2. All baselines, except PaSa-GPT-4o, represent the best-known scholar search methods. These comparisons highlight the effectiveness of our agentic approach. The comparison with PaSa-GPT-4o isolates the impact of RL training.

As shown in Figure 2, the crawling process of PaSa can be visualized as a paper tree. In practice, considering the computational expense, we limit the Crawler's exploration depth to three for both PaSa-7b and PaSa-GPT-4o.

For Google-based baselines, we evaluate recall using Recall@20, Recall@50, and Recall@100 metrics for the top-20, top-50, and top-100 search

[7] Accessed via the Google Search API provided by https://serper.dev.

[8] https://ar5iv.org/

[9] https://chatgpt.com

Method	Crawler Recall	Precision	Recall	Recall@100	Recall@50	Recall@20
Google	-	-	-	0.2015	0.1891	0.1568
Google Scholar	-	-	-	0.1130	0.0970	0.0609
Google with GPT-4o	-	-	-	0.2683	0.2450	0.1921
ChatGPT*	-	0.0507	0.3046	-	-	-
GPT-o1	-	0.0413	0.1925	-	-	-
PaSa-GPT-4o	0.7565	**0.1457**	0.3873	-	-	-
PaSa-7b	**0.7931**	**0.1448**	**0.4834**	**0.6947**	**0.6334**	**0.5301**
PaSa-7b-ensemble	**0.8265**	0.1410	**0.4985**	**0.7099**	**0.6386**	**0.5326**

Table 4: Results on AutoScholarQuery test set. *: Due to the need for manual query submission, the ChatGPT baseline is evaluated on 100 randomly sampled instances. Results for all methods on this subset are reported in Table 14.

Method	Crawler Recall	Precision	Recall	Recall@100	Recall@50	Recall@20
Google	-	-	-	0.2535	0.2342	0.1834
Google Scholar	-	-	-	0.2809	0.2155	0.1514
Google with GPT-4o	-	-	-	0.2946	0.2573	0.2020
ChatGPT	-	0.2280	0.2007	-	-	-
GPT-o1	-	0.058	0.0134	-	-	-
PaSa-GPT-4o	0.5494	0.4721	0.3075	-	-	-
PaSa-7b	**0.7071**	**0.5146**	**0.6111**	**0.6929**	**0.6563**	**0.5798**
PaSa-7b-ensemble	**0.7503**	**0.4938**	**0.6488**	**0.7281**	**0.6877**	**0.5986**

Table 5: Results on RealScholarQuery.

results, respectively. For other baselines that do not produce rankings, we assess precision and recall for the final retrieved papers. Additionally, we compare Crawler recall between PaSa-GPT-4o and PaSa-7b, defined as the proportion of target papers collected by the Crawler. This measures how many target papers are successfully included in the paper queue generated by the Crawler.

Method	Precision	Recall	F1
GPT-4o	0.96	0.69	0.80
Qwen-2.5-7b	1.0	0.38	0.55
PaSa-7b-Selector	0.95	0.78	**0.85**
PaSa-7b-Selector (Reason First)	0.94	0.76	**0.84**

Table 6: Selector Evaluation.

5.3 Main results

As shown in Table 4, PaSa-7b outperforms all baselines on AutoScholarQuery test set. Specifically, compared to the strongest baseline, PaSa-GPT-4o, PaSa-7b demonstrates a 9.64% improvement in recall with comparable precision. Moreover, the recall of the Crawler in PaSa-7b is 3.66% higher than that in PaSa-GPT-4o. When compared to the best Google-based baseline, Google with GPT-4o, PaSa-7b achieves an improvement of 33.80%, 38.83% and 42.64% in Recall@20, Recall@50 and Recall@100, respectively.

We observe that using multiple ensembles of Crawler during inference can improve performance. Specifically, we use sampling decoding to run Crawler twice in the PaSa-7b-ensemble setting, which boosts Crawler recall by 3.34% on AutoScholarQuery and increases the final recall by 1.51%, with no significant change in precision.

To evaluate PaSa in a more realistic setting, we assess its effectiveness on RealScholarQuery. As illustrated in Table 5, PaSa-7b exhibits a greater advantage in real-world academic search scenarios. Compared to PaSa-GPT-4o, PaSa-7b achieves improvements of 30.36% in recall and 4.25% in precision. Against the best Google-based baseline on RealScholarQuery, Google with GPT-4o, PaSa-7b outperforms Google by 37.78%, 39.90%, and 39.83% in recall@20, recall@50 and recall@100, respectively. Additionally, the PaSa-7b-ensemble further enhances crawler recall by 4.32%, contributing to an overall 3.52% improvement in the recall of the entire agent system.

As both the final decision-maker and auxiliary reward model in RL training for the Crawler, the performance of the Selector is crucial. To evaluate its effectiveness, we collected a dataset of 200 query-paper pairs, annotating whether each paper meets the query's requirements. This dataset serves

Method	AutoScholarQuery			RealScholarQuery		
	Crawler Recall	Precision	Recall	Crawler Recall	Precision	Recall
w/o [Expand]	0.3355	0.1445	0.2536	0.3359	**0.6738**	0.2890
w/o RL training	0.6556	0.1476	0.4210	0.4847	0.5155	0.4115
w/o Selector as RM	0.7041	**0.1535**	0.4458	0.5994	0.5489	0.5148
PaSa-7b	**0.7931**	0.1448	**0.4834**	**0.7071**	0.5146	**0.6111**

Table 7: Ablation study results on AutoScholarQuery test set and RealScholarQuery.

as the benchmark for evaluating the Selector (see Appendix F for details). We then compared our Selector against GPT-4o (Hurst et al., 2024) and Qwen-2.5-7b (Yang et al., 2024), as shown in Table 6. The results show that our Selector achieves an F1 score of 85%, outperforming GPT-4o by 5% and Qwen-2.5-7b by 30%. Additionally, when compared to a setting where reasoning precedes decision token generation, the performance is comparable. Lastly, the Selector's precision reaches 95%, confirming its effectiveness as an auxiliary reward model for the Crawler RL training.

5.4 Ablation study

We perform ablation studies in Table 7 to evaluate the individual contributions of exploring citation networks, RL training, and using the Selector as the reward model. The results indicate that removing the [Expand] action from the Crawler leads to a significant drop in the recall: a decrease of 22.98% on AutoScholarQuery and 32.21% on RealScholarQuery. Furthermore, RL training enhances recall by 6.24% on AutoScholarQuery and 19.96% on RealScholarQuery. The RL training curves are depicted in Figure 3 in Appendix D.2, where the training curves show a steady increase in return with the training steps, eventually converging after 200 steps. Finally, removing the Selector as an auxiliary reward model results in a 3.76% recall drop on AutoScholarQuery and a 9.63% drop on RealScholarQuery.

We investigate how to control agent behavior by adjusting the rewards in RL training. Experiments are conducted with varying reward coefficients α in Equation 1, and results are presented in Table 8. We report two metrics: crawler recall and crawler action. The crawler action refers to the total number of [Search] and [Expand] actions throughout the Crawler's entire trajectory. As the reward increases, both crawler recall and crawler action increase, suggesting that adjusting rewards in RL training can effectively influence PaSa's behavior.

α	Crawler Recall	Crawler Actions	Precision	Recall
0.5	0.7227	175.9	0.1458	0.4602
1.0	0.7708	319.8	0.1451	0.4792
1.5	0.7931	382.4	0.1448	0.4834
2.0	0.8063	785.5	0.1409	0.4869

Table 8: Performance of the Crawler trained on different reward coefficient α on AutoScholarQuery test set.

6 Conclusion

In this paper, we introduce PaSa, a novel paper search agent designed to provide comprehensive and accurate results for complex academic queries. PaSa is implemented within the AGILE, a reinforcement learning framework for LLM agents. To train PaSa, we developed AutoScholarQuery, a dataset of fine-grained academic queries and corresponding papers drawn from top-tier AI conference publications. To evaluate PaSa in real-world scenarios, we also constructed RealScholarQuery, a dataset of actual academic queries paired with annotated papers. Our experimental results demonstrate that PaSa outperforms all baselines, including Google, Google Scholar, and Google with GPT-4o, ChatGPT, GPT-o1, and PaSa-GPT-4o. In particular, PaSa-7B surpasses Google with GPT-4o by 37.78% in recall@20 and 39.90% in recall@50, while also exceeding PaSa-GPT-4o by 30.36% in recall and 4.25% in precision. These findings underscore that PaSa significantly improves the efficiency and accuracy of academic search.

Limitations

(1) Our dataset collection and experiments were primarily focused on the field of machine learning. Although our proposed method is general, we did not explore its performance in other scientific fields. We leave to investigate its applicability to other domains in future work.

(2) Due to resource constraints, our experiments primarily use LLMs with 7b parameters. We expect

that scaling up to larger models will lead to more powerful agents. Expanding PaSa to leverage larger LLMs is our future work.

Acknowledgments

The authors thank Yaohua Fang, Zheng Li, Qiang Lu, Yelong Shi, Xuguang Wei, and Tingshuai Yan from ByteDance for their support in developing the PaSa demo. We also thank Jianghui Xie at ByteDance for her assistance with the release of the PaSa demo. Finally, we thank the anonymous reviewers for their valuable suggestions that helped improve this work.

References

Shubham Agarwal, Issam H Laradji, Laurent Charlin, and Christopher Pal. 2024. Litllm: A toolkit for scientific literature review. *arXiv preprint arXiv:2402.01788*.

Marwah Alaofi, Luke Gallagher, Mark Sanderson, Falk Scholer, and Paul Thomas. 2023. Can generative llms create query variants for test collections? an exploratory study. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 1869–1873.

A Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku; 2024. *URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf*.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2024. Researchagent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. Agile: A novel reinforcement learning framework of llm agents. *Advances in Neural Information Processing Systems*, 37:5244–5284.

Team Gemini. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Karan Girotra, Lennart Meincke, Christian Terwiesch, and Karl T Ulrich. 2023. Ideas are dimes a dozen: Large language models for idea generation in innovation. *Available at SSRN 4526071*.

Atharva Gundawar, Mudit Verma, Lin Guan, Karthik Valmeekam, Siddhant Bhambri, and Subbarao Kambhampati. 2024. Robust planning with llm-modulo framework: Case study in travel planning. *arXiv preprint arXiv:2405.20625*.

Michael Gusenbauer and Neal R Haddaway. 2020. Which academic search systems are suitable for systematic reviews or meta-analyses? evaluating retrieval qualities of google scholar, pubmed, and 26 other resources. *Research synthesis methods*, 11(2):181–217.

Michael Gusenbauer and Neal R Haddaway. 2021. What every researcher should know about searching–clarified concepts, search advice, and an agenda to improve finding in academia. *Research synthesis methods*, 12(2):136–147.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Karl Kingsley, Gillian M Galbraith, Matthew Herring, Eva Stowers, Tanis Stewart, and Karla V Kingsley. 2011. Why not just google it? an assessment of information literacy skills in a biomedical science curriculum. *BMC medical education*, 11:1–8.

Minghan Li, Honglei Zhuang, Kai Hui, Zhen Qin, Jimmy Lin, Rolf Jagerman, Xuanhui Wang, and Michael Bendersky. 2023. Generate, filter, and fuse: Query expansion via multi-step keyword generation for zero-shot neural rankers. *arXiv preprint arXiv:2311.09175*.

Zhehui Liao, Maria Antoniak, Inyoung Cheong, Evie Yu-Yen Cheng, Ai-Heng Lee, Kyle Lo, Joseph Chee Chang, and Amy X Zhang. 2024. Llms as research tools: A large scale survey of researchers' usage and perceptions. *arXiv preprint arXiv:2411.05025*.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. 2023. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, pages 1–11.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Lisa Messeri and MJ Crockett. 2024. Artificial intelligence and illusions of understanding in scientific research. *Nature*, 627(8002):49–58.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 20–28.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, Mexico City, Mexico. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Jess Stratton. 2024. An introduction to microsoft copilot. In *Copilot for Microsoft 365: Harness the Power of Generative AI in the Microsoft Apps You Use Every Day*, pages 19–35. Springer.

Richard Van Noorden and Jeffrey M Perkel. 2023. Ai and science: what 1,600 researchers think. *Nature*, 621(7980):672–675.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. 2024a. SciMON: Scientific inspiration machines optimized for novelty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–299, Bangkok, Thailand. Association for Computational Linguistics.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, et al. 2024b. Autosurvey: Large language models can automatically write surveys. *arXiv preprint arXiv:2406.10252*.

Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 1–10.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: synergizing reasoning and acting in language models (2022). *arXiv preprint arXiv:2210.03629*.

A Quality Evaluation of AutoScholarQuery

To assess the quality of AutoScholarQuery, we sampled 100 query-paper pairs and evaluated the rationality and relevance of each query and its corresponding paper. The detailed evaluation criteria are as follows:

- A qualified query should be a complete and understandable sentence. For example, incomplete or fragmented sentences are not acceptable.
- A query that misrepresents the meaning of the source paper, leading to irrelevant citations, is not qualified. This includes queries that exaggerate the scope or introduce incorrect conditions.
- A query is ambiguous if there is insufficient context and additional information is needed. For instance, abbreviations with multiple meanings can create ambiguity, leading to the corresponding citations being incomplete answer lists.

Query: Give me papers about how to rank search results by the use of LLM
Query Date: 2024-10-01
Answer Papers:
[0] Instruction Distillation Makes Large Language Models Efficient Zero-shot Rankers (2311.01555)
[1] Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels (2310.14122)
[2] Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting (2306.17563)
[3] A Setwise Approach for Effective and Highly Efficient Zero-shot Ranking with Large Language Models (2310.09497)
[4] RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models (2309.15088)
[5] PaRaDe: Passage Ranking using Demonstrations with Large Language Models (2310.14408)
[6] Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents (2304.09542)
[7] Large Language Models are Zero-Shot Rankers for Recommender Systems (2305.08845)
[8] TourRank: Utilizing Large Language Models for Documents Ranking with a Tournament-Inspired Strategy (2406.11678)
[9] ExaRanker: Explanation-Augmented Neural Ranker (2301.10521)
[10] RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs (2407.02485)
[11] Make Large Language Model a Better Ranker (2403.19181)
[12] LLM-RankFusion: Mitigating Intrinsic Inconsistency in LLM-based Ranking (2406.00231)
[13] Improving Zero-shot LLM Re-Ranker with Risk Minimization (2406.13331)
[14] Zero-Shot Listwise Document Reranking with a Large Language Model (2305.02156)
[15] Consolidating Ranking and Relevance Predictions of Large Language Models through Post-Processing (2404.11791)
[16] Re-Ranking Step by Step: Investigating Pre-Filtering for Re-Ranking with Large Language Models (2406.18740)
[17] Large Language Models for Relevance Judgment in Product Search (2406.00247)
[18] PromptReps: Prompting Large Language Models to Generate Dense and Sparse Representations for Zero-Shot Document Retrieval (2404.18424)
[19] Passage-specific Prompt Tuning for Passage Reranking in Question Answering with Large Language Models (2405.20654)
[20] When Search Engine Services meet Large Language Models: Visions and Challenges (2407.00128)
[21] RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! (2312.02724)
[22] Rank-without-GPT: Building GPT-Independent Listwise Rerankers on Open-Source Large Language Models (2312.02969)
[23] MuGI: Enhancing Information Retrieval through Multi-Text Generation Integration with Large Language Models (2401.06311)
[24] Discrete Prompt Optimization via Constrained Generation for Zero-shot Re-ranker (2305.13729)
[25] REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering (2402.17497)
[26] Agent4Ranking: Semantic Robust Ranking via Personalized Query Rewriting Using Multi-agent LLM (2312.15450)
[27] FIRST: Faster Improved Listwise Reranking with Single Token Decoding (2406.15657)
[28] Leveraging LLMs for Unsupervised Dense Retriever Ranking (2402.04853)
[29] Unsupervised Contrast-Consistent Ranking with Language Models (2309.06991)
[30] Enhancing Legal Document Retrieval: A Multi-Phase Approach with Large Language Models (2403.18093)
[31] Found in the Middle: Permutation Self-Consistency Improves Listwise Ranking in Large Language Models (2310.07712)
[32] Fine-Tuning LLaMA for Multi-Stage Text Retrieval (2310.08319)
[33] Zero-shot Audio Topic Reranking using Large Language Models (2309.07606)
[34] Uncovering ChatGPT's Capabilities in Recommender Systems (2305.02182)
[35] Cognitive Personalized Search Integrating Large Language Models with an Efficient Memory Mechanism (2402.10548)
[36] Towards More Relevant Product Search Ranking Via Large Language Models: An Empirical Study (2409.17460)
[37] Pretrained Language Model based Web Search Ranking: From Relevance to Satisfaction (2306.01599)
[38] Open-source large language models are strong zero-shot query likelihood models for document ranking (2310.13243)

Table 9: Examples of queries and corresponding papers in RealScholarQuery.

- An answer paper is considered qualified if it aligns with the requirements of the query. The paper should address all or most of the essential factors that make it a suitable response.

Our quality check found that 94.0% of the queries were qualified. Among them, 93.7% of the corresponding answer papers were also qualified. The primary reason for unqualified papers was inaccurate citations in the source paper.

B Annotation details

The annotators of RealScholarQuery include professors from the Department of Computer Science at a top-tier university in China. They are paid $4 per data entry, which consists of a user query and a research paper. Their task is to determine whether the paper satisfies the query.

B.1 Annotation Instructions

For each <user query, paper> pair, carefully assess whether the paper address the user query. Write your decision and provide a brief explanation (1-2 sentences). Specific guidelines are as follows:

- You may read the entire paper to determine whether it satisfies the user query.
- Exclude survey papers unless the user query

The prompt for search query generation
You are an elite researcher in the field of AI, please generate some mutually exclusive queries in a list to search the relevant papers according to the User Query. Searching for a survey paper would be better. User Query: {user_query} The semantics between generated queries are not mutually inclusive. The format of the list is: ["query1", "query2", ...] Queries:

Table 10: The prompt for GPT-4o to generate search queries from the user query.

	Search Session starting from S_q	**Expand Session starting from S_{q+p}**
prompt	Please, generate some mutually exclusive queries in a list to search the relevant papers according to the User Query. Searching for survey papers would be better. User Query: {user_query}	You are conducting research on '{user_query}'. You need to predict which sections to look at to get more relevant papers. Title: {title} Abstract: {abstract} Sections: {sections}
response	[Search] {query 1} [Search] {query 2} ... [Stop]	[Expand] {section 1} [Expand] {section 2} ... [Stop]

Table 11: The session trajectory templates of the Crawler.

specifically requests them.

- All conditions in the user query must be met for the paper to be considered qualified.

B.2 Quality control

The annotation process follows the following quality control measures:

- Stage 1: Annotators work in batches of 20. Authors review 100% of the annotations. Once the consistency rate on the first pass reaches 90%, the process moves to Stage 2.
- Stage 2: Annotators work in batches of 50. Authors randomly check 40% of the annotations. If the consistency rate is below 90%, the entire batch is re-annotated and re-checked. Once the batch meets the 90% consistency rate on the first pass, the process moves to Stage 3.
- Stage 3: Annotators work in batches of 100. Authors randomly check 20% of the annotations. If the consistency rate is below 90%, the entire batch is re-annotated and re-checked.

Two authors conducted the quality control.

C Example in RealScholarQuery

Table 9 presents an example query and corresponding papers from RealScholarQuery.

D Implementation Details of the Crawler

D.1 Imitation learning data generation

We generate training data for imitation learning on a session-by-session basis. There are two types of sessions: *search session* (starting from state S_q) and *expand session* (starting from state S_{q+p}).

For search sessions starting from S_q, we sample user queries from the AutoScholarQuery training set and prompt GPT-4o to generate corresponding search queries. The prompt template is shown in Table 10. The session trajectory is constructed by adding a `[Search]` token before each query, concatenating the queries, and appending a `[Stop]` token at the end, as shown in Table 11. A total of 3,011 search session trajectories are generated.

For expanded sessions starting from S_{q+p}, we continue by searching for the generated queries using Google. We then sample papers from the search results and obtain the initial state, which includes both the query and a paper. To build the session trajectory, we examine each sub-section of the paper. If the sub-section references at least one paper in the AutoScholarQuery training set corresponding to the query, the sub-section is selected. Otherwise, the sub-section is selected with a 10% probability to enhance trajectory diversity. The selected sections are filled into the template in Table 11, completing the session trajectory. In total, 9,978 expanded session trajectories are constructed.

D.2 PPO training

During PPO training, each device processes 4 user queries in each step, generating a search session for each user query. Then, 6 expansion sessions are created by randomly sampling 6 papers from the search results. This process is repeated with the expanded citation results, yielding 6 additional expanded sessions. In total, 16 session trajectories are generated per step.

Name		Value
α	(Equation 1)	1.5
c([Search])	(Equation 1)	0.1
c([Expand])	(Equation 1)	0.1
c([Stop])	(Equation 1)	0.0
γ_0	(Equation 3)	1.0
γ_1	(Equation 3)	0.1
β	(Equation 3)	0.1
ϵ	(Equation 5, Equation 6)	0.2
η	(Equation 8)	10
learning rate		1e-6
epoch per step		2
forward batch size		1
accumulate batch size		16
NVIDIA H100 GPU		16
policy freezing step		50
total step		250

Table 12: The hyperparameters used in PPO training.

Table 12 lists the hyperparameters used during the training process. Figure 3 depicts the RL training curves, which show a steady increase in return with the training steps, eventually converging after 200 steps.

E Implementation Details of the Selector

We begin by sampling user queries from the AutoScholarQuery training set. For each user query and one of its corresponding papers in the AutoScholarQuery training set, we prompt GPT-4o to generate a decision token and rationale (see Table 13 for the prompt). We reject any data where the decision token is "False", as this contradicts the AutoScholarQuery label. The remaining data are retained as positive <user query, paper> pairs.

Next, we simulate a partial paper search using PaSa-GPT-4o. In this simulation, each paper has a 50% probability of being added to the paper queue. Pairs where the paper is not selected by GPT-4o and is not in the AutoScholarQuery training set are labeled as negative examples.

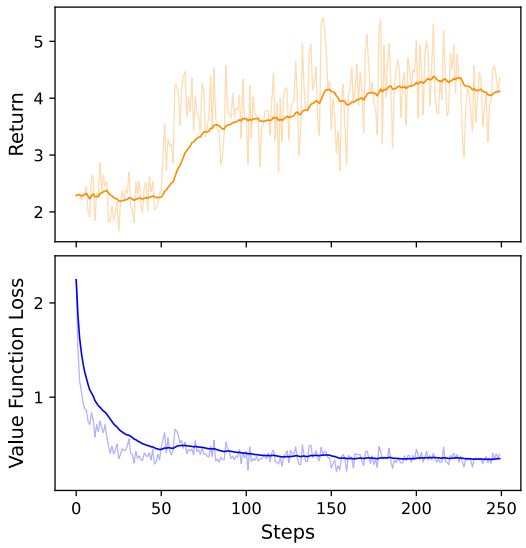

Figure 3: Return and value function loss curves during the PPO training process. The smoothing method of the curve in the figures is the exponential moving average(EMA) formula that aligns with the one used in TensorBoard, and the smoothing weight is set to 0.95.

The final training dataset consists of 19,812 <user query, paper> pairs, each with a decision token and rationale generated by GPT-4o, drawn from 9,000 instances in the AutoScholarQuery training set.

F Selector Test Dataset

We select 200 queries from the AutoScholarQuery development set. For each query, we perform a Google search and randomly choose one paper from the union of the search results and the relevant paper set in AutoScholarQuery. This yields a set of <user query, paper> pairs. Annotators then evaluate whether each paper aligns with the requirements of the user query. The final test dataset consists of 98 positive samples and 102 negative samples.

G Additional Experimental Results

G.1 Results on 100-sample subset of AutoScholarQuery

To ensure a fair comparison with the ChatGPT baseline, which is evaluated on only 100 samples from AutoScholarQuery test, we report the performance of all methods on the same subset in Table 14. The results align with those in Table 4, confirming that PaSa-7b consistently outperforms all baselines.

The prompt for paper selection
You are an elite researcher in the field of AI, conducting research on {user_query}. Evaluate whether the following paper fully satisfies the detailed requirements of the user query and provide your reasoning. Ensure that your decision and reasoning are consistent. Searched Paper: Title: {title} Abstract: {abstract} User Query: {user_query} Output format: Decision: True/False Reason:... Decision:

Table 13: Prompt used by PaSa Selector or GPT-4o to evaluate paper relevance.

Method	**Crawler Recall**	**Precision**	**Recall**	**Recall@100**	**Recall@50**	**Recall@20**
Google	-	-	-	0.2101	0.2010	0.1788
Google Scholar	-	-	-	0.0801	0.0739	0.0612
Google with GPT-4o	-	-	-	0.2101	0.2010	0.1788
ChatGPT	-	0.0507	0.3046	-	-	-
GPT-o1	-	0.0374	0.2006	-	-	-
PaSa-GPT-4o	0.7595	0.1817	0.4488	-	-	-
PaSa-7b	**0.7752**	**0.1881**	**0.5328**	**0.6932**	**0.6543**	**0.5494**
PaSa-7b-ensemble	**0.8244**	**0.1822**	**0.5568**	**0.7041**	**0.6795**	**0.5535**

Table 14: Results on 100-sample subset of AutoScholarQuery test.

G.2 Action cost

We incorporate action costs to prevent the agent from taking excessive, unproductive actions. Without such costs, the total number of actions would increase significantly without yielding meaningful outcomes.

The key consideration is the reward coefficient α and the action cost $c(a_t)$ in Equation 1. In Table 8, we fix $c(a_t)$ and analyze how varying α affects performance.

Additionally, Table 15 shows how different values of $c(a_t)$ affect the final performance.

$c(a_t)$	**Crawler Recall**	**Crawler Actions**	**Precision**	**Recall**
0	0.8239	1296.3	0.1388	0.4852
0.1	0.7931	382.4	0.1448	0.4834
0.2	0.7478	230.1	0.1489	0.4764

Table 15: Performance of the Crawler trained on different action cost $c(a_t)$ on AutoScholarQuery test set.

H Prompt Templates

H.1 Prompts used for data synthesis in AutoScholarQuery

Table 16 presents the prompt template used with GPT-4o to automatically generate AutoScholarQuery. For each paper, we extract its *Related Work* section, input it into GPT-4o, and use the prompt to generate scholarly queries along with their corresponding paper answers.

H.2 Prompts for baselines

Table 17 presents the search query paraphrasing prompt used for the baseline Google with GPT-4o.

Table 18, 19 and 20 show the prompts used for the ChatGPT baseline (search-enabled GPT-4o), the GPT-o1 baseline and PaSa-GPT-4o, respectively.

The prompt for AutoScholarQuery generation
You are provided a 'Related Work' section of a research paper. The researcher reviewed the relevant work, conducted a literature survey, and cited corresponding references in this text (enclosed by '\cite' tags with IDs). Can you guess what research questions the researcher might have posed when preparing this text? The answers to these questions should be the references cited in this passage. Please list questions and provide the corresponding answers. [Requirements:] 1. Craft questions similar to those a researcher would pose when reviewing related works, such as "Which paper studied ...?", "Any works about...?", "Could you provide me some works...?" 2. Construct the question-answer pairs based on [Section from A Research Paper]. The answer should be the cited papers in [Section from A Research Paper]. 3. Do not ask questions including "or" or "and" that may involve more than one condition. 4. Clarity: Formulate questions clearly and unambiguously to prevent confusion. 5. Contextual Definitions: Include explanations or definitions for specialized terms and concepts used in the questions. 6. Format the output as a JSON array containing five objects corresponding to the three question-answer pairs. Here are some examples: [Begin of examples] {Section from A Research Paper-1} {OUTPUT-1} {Section from A Research Paper-2} {OUTPUT-2} {Section from A Research Paper-3} {OUTPUT-3} [End of examples] {Section from A Research Paper} [OUTPUT]:

Table 16: The prompt used with GPT-4o to automatically synthesize AutoScholarQuery.

The prompt for search query paraphrase
Generate a search query suitable for Google based on the given academic paper-related query. Here's the structure and requirements for generating the search query: Understand the Query: Read and understand the given specific academic query. Identify Key Elements: Extract the main research field and the specific approaches or topics mentioned in the query. Formulate the Search Query: Combine these elements into a concise query that includes terms indicating academic sources. Do not add any site limitations to your query. [User's Query]: {user_query} [Generated Search Query]:

Table 17: The search query paraphrasing prompt used for the Google with GPT-4o baseline.

The prompt for ChatGPT (search-enabled GPT-4o)
[User's Query] You should return the Arxiv papers. You should provide more than 10 papers you searched in JSON format: {"paper_1": {"title": , 'authors': , 'link': }, "paper_2": {"title": , 'authors': , 'link': }}

Table 18: The prompt for ChatGPT baseline (search-enabled GPT-4o).

The prompt for GPT-o1
{user_query} You should return arxiv papers. You should provide more than 10 paper you searched in JSON format: {"paper_1": {"title": , "authors": , "link": }, "paper_2": {"title": , "authors": , "link": }}. Do not return any other description.

Table 19: The prompt for GPT-o1 baseline.

The prompt for search session of Crawler

You are an elite researcher in the field of AI, please generate some mutually exclusive queries in a list to search the relevant papers according to the User Query. Searching for a survey paper would be better.
User Query: {user_query}
The semantics between generated queries are not mutually inclusive. The format of the list is: [“query1”, “query2”, ...]
Queries:

The prompt for the expand session of Crawler

You are an elite researcher in the field of AI, currently conducting research on the [topic] specified below. Your task is to determine if the paper specified below likely contains highly relevant citations for the [topic] and, if so, to identify specific sections where these citations might appear.
Task Instructions:
1. Relevance Assessment: Decide if the paper is likely to include citations highly relevant to the given [topic]. Output "Yes" or "No" on the first line.
2. Section Selection: If you answered "Yes" in step 1, identify which sections of the paper are likely to contain these relevant citations. From the list of provided sections, select only those you think may contain relevant citations. If no sections seem relevant even if your answer to step 1 was "Yes," leave this empty. Output the selected sections in JSON format on the second line.
[topic]: {user_query}
[paper title]: {title}
[paper abstract]: {abstract}
[paper sections]: {sections}
Output Format: Output exactly two lines:
1. The first line: Either "Yes" or "No" based on the relevance assessment.
2. The second line: A JSON string with selected sections, e.g., {{"selected_section_1": section_name_1, "selected_section_2": section_name_2}}. If no sections are selected, output {{}}.

The prompt for Selector

You are an elite researcher in the field of AI, conducting research on {user_query}. Evaluate whether the following paper fully satisfies the detailed requirements of the user query and provide your reasoning. Ensure that your decision and reasoning are consistent.
Searched Paper:
Title: {title}
Abstract: {abstract}
User Query: {user_query}
Output format: Decision: True/False
Reason:...
Decision:

Table 20: The prompts for PaSa-GPT-4o.