Towards Autonomous Web Navigation with LLM-based Agents

Mario Izquierdo Álvarez[a,*]

[a]FDS, a DXC Technology Company, Santiago de Compostela, Spain.

[b]Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Santiago de Compostela, Spain.

***Abstract*—Autonomous web navigation remains a complex challenge, particularly due to the dynamic, diverse and unstructured nature of web environments. Traditional web scraping techniques, while effective, require rigid configurations tied to specific website structures, limiting their generalizability. To address these challenges, this work explores the usage of autonomous agents powered by Large Language Models for autonomous web navigation, focusing on the retrieval of academic publications from webs of preprint repositories.**

The proposed solution, based on hyperlink exploration, is designed as a component of a potentially broader system for AI-driven paper search assistance. It leverages a multi-agent architecture and a structured tree-traversal like approach to explore and extract relevant documents. Each agent is assigned a specific role, including relevant URL extraction, document collection, planning, presentation and quality control. The system is implemented using AutoGen, which enables flexible agent interactions and modular design. Unlike traditional web information extraction techniques, this approach generalizes navigation patterns across different websites without relying on predefined HTML selectors, allowing its usage on different websites.

Experimental results are promising, demonstrating the system's effectiveness in retrieving relevant academic content. However, challenges such as increased response times and occasional hallucinations indicate areas for refinement. Future work aims to enhance interactivity by integrating advanced form-based search capabilities, optimize retrieval efficiency, and implement more robust evaluation frameworks. These improvements could contribute to fully automated AI-driven web exploration, facilitating the development of more generalizable autonomous web navigation tools.

***Index Terms*—LLM, Agents, Web Scraping, Autonomous Navigation**

I. Introduction

In recent years, artificial intelligence (AI) and natural language processing (NLP) have made considerable progress, capturing significant attention, and driving extensive research [1]–[3]. A key event in this evolution was the introduction of the transformer architecture, presented in the influential paper "Attention Is All You Need" [4]. This innovation changed how models process sequential data, by using self-attention mechanisms, removing the need for recurrence and convolution, and enabling greater scalability and efficiency. Transformers enabled the development of a new class of models, known as Large Language Models (LLMs), which have become central to the field of NLP and one of the hot-topics in AI nowadays.

The release of GPT-3 marked a significant milestone for LLMs, demonstrating remarkable capabilities in understanding and generating human-like text, significantly improving task-agnostic, few-shot performance, and sometimes even achieving competitiveness with prior state-of-the-art fine-tuning approaches [3]. It showcased the potential of LLMs in diverse applications, from text completion to creative content generation.

Building on this, the debut of ChatGPT brought generative AI and LLMs to mainstream attention, collecting an unprecedented user base of one million users within just five days of launch [5]. This rapid adoption highlighted the growing interest in exploring the applications of LLMs. These models quickly gained attention for their versatility and emerging capabilities, motivating researchers and industries to explore new applications and opportunities [6]. Just a few months later, the release of GPT-4 marked a major leap forward, highlighting the impact of scaling in model architecture, dataset size, and computational resources. GPT-4 demonstrated improved reasoning, contextual understanding, and task generalization, sometimes even reaching human-level performance on various academic and professional benchmarks, solidifying the role of LLMs in AI and encouraging further research on their abilities and limitations [7].

Despite their remarkable capabilities, foundational models (FMs) based on LLMs exhibit inherent limitations, particularly in handling complex tasks that require autonomous behavior, decision-making, and action execution. While they excel at generating text and providing contextually relevant responses, FMs rely on explicit human input for each step, lacking the ability to independently plan, execute tasks, or interact with external systems. This dependency on user prompts for every action makes their operation inefficient and increases the likelihood of errors [8], [9].

This gap has led to a growing interest in the development of FM-based autonomous agents, designed to address these limitations by integrating LLMs into frameworks that enable autonomy and goal-oriented functionality.

LLM agents represent an innovative approach that integrates the language understanding and reasoning capabilities of FMs with the ability to interact with other agents, modules for planning, decision-making, code execution, and the use of external tools. This combination enables them to adapt to dynamic environments, autonomously decompose high-level objectives into manageable tasks, and effectively orchestrate their execution to achieve specified goals [8].

Currently, LLM agents represent an emerging field of research, still in its early stages, but with a high potential to address challenges requiring autonomy, complex planning, language and contextual understanding, and goal-oriented

*Work supervised by Francisco José Morón[a] and Marcos Fernández[b].

functionality. Early explorations highlight applications such as software engineering, where agents assist in automating tasks such as code generation and debugging [10], [11], medical assistance, enabling personalized healthcare services and informed decision-making [12], [13], conversational database querying, improving data accessibility by enabling natural language interactions with databases [14] among others.

Additionally, one of the most compelling improvements of FMs is to equip them with tools that allows them to access and retrieve up-to-date information from the web. This addresses one of their primary limitations: their knowledge is constrained to the data available during their training, making them unaware of recent information. Recent efforts have explored frameworks for enabling LLMs to interact dynamically with online content [15]. A prominent implementation of this concept is OpenAI's "GPT Search", which equips ChatGPT with real-time web search functionality, extending its knowledge base to include the latest available information [16].

Nonetheless, while some applications aim to provide LLMs with broad web knowledge, other research efforts focus on more targeted or specialized searches, enabling in-depth exploration of specific domains or websites. These approaches often require the ability to navigate internally within websites and interact with web elements to achieve predefined objectives, often requiring more advanced planning and interaction capabilities, which aligns with the capacities of LLM agents. While this is still a very recent research field, it has captured significant interest, with several recent studies focusing on enhancing the web interaction capabilities of agents [17]–[21]. However, despite these advancements, the field remains far from achieving human level performance, requiring substantial progress in areas such as web navigation and interaction capabilities.

It is worth noting that, while these advancements still require further refinement and research, LLM agents hold significant promise for information extraction from specific websites and targeted searches, since FMs capabilities offer solutions to some inherent limitations in traditional automated web information extraction systems.

Traditionally, web information extraction relies heavily on techniques such as web scraping and web crawling. These methods, while efficient, are inherently tied to the specific design and structure of each website. They require specific configurations for each website and are highly dependent on how the HTML is written. Moreover, they often necessitate constant maintenance to adapt to structural changes, which can disrupt automated workflows and limit scalability across different domains [22], [23].

In contrast, LLM-based web navigation agents leverage their contextual understanding and planning abilities to achieve greater abstraction and generalization. This allows these systems to generalize similar tasks across websites with varying HTML structures without relying on rigid DOM selectors or fixed layouts. By using LLM's contextual comprehension, these agents can make informed navigation decisions, dynamically adapting to new environments. This setup provides a more flexible and adaptive approach to web information extraction, potentially reducing web-specific dependency and manual adjustments [22], [23].

In this context, this work explores the application of common design and development patterns for LLM-based agent systems that have been used to address various tasks, focusing specifically on their adaptation to the task of extracting academic papers from preprint repositories. This task is often approached by humans using similar interaction flows across different websites. However, the heterogeneity and variability in the HTML structures of these sites make it unfeasible to develop tools capable of functioning effectively across multiple websites using traditional information extraction methods.

To address this challenge, the proposed system relies on LLM agents capable of autonomously generating plans, making decisions, executing actions, and navigating websites based on natural language queries.The proposed solution employs a stateful multi-agent system with a partially restricted interaction flow. In this setup, the agents maintain a dynamic search state, enabling them to adapt their actions to observations at each step. During the navigation process, relevant information is gathered and stored. Once the search is complete, this information is ranked based on its semantic similarity to the initial query and presented to the user through a conversational chat interface.

II. Related Work

As previously mentioned, multi-agent LLMs systems for autonomous web navigation is still an emerging and promising area of research. However, despite being in its early stages, it has generated significant interest due to its potential to address the limitations of traditional web content extraction methods and to embrace new opportunities enabled by recent advancements in AI and NLP, progressing toward more powerful systems for automated web interaction. Thus, recent studies have explored various approaches to this problem, aiming to improve task completion rates, enhance generalizability, and leverage structured and flexible methodologies to navigate the web effectively.

A. Reinforcement Learning Methods

One of the earliest line of research on web agents focuses on reinforcement learning methods. For instance, Liu et al. [24] propose a workflow-guided exploration framework that leverages demonstrations to constrain an RL agent's exploration paths, reducing the complexity of sparse reward scenarios by introducing "workflows". Their method uses a neural network architecture, DOMNET, specifically designed to perform flexible relational reasoning over the tree-structured HTML representation of websites. Similarly, following the RL strategy, a more recent work Yao et al. [25] introduce WebShop, a large-scale simulation environment designed to train RL agents for real-world e-commerce tasks. The authors combined RL and imitation learning, along with pre-trained language and vision models, replacing pixel-level mouse click interaction of other works with high-level semantic actions. Nevertheless, the authors stated that success rate remains well below human performance, being the action selection a bottleneck for the models with great room for improvement.

B. HTML fine-tuned

Another significant and remarkably recent contribution comes from WebAgent by Gur et al. [26], which addresses challenges in web automation through specialized models and a modular architecture. At its core is HTML-T5, a new pre-trained LLM fine-tuned for long HTML documents using local and global attention, capable of planning tasks and summarizing long web pages. WebAgent interacts with web pages programmatically, using Flan-U-PaLM for python code generation. Additionally, leveraging self-experience supervision, the system aligns its models with real-world web interactions, improving generalization and reducing human involvement in manual annotation. Authors state that WebAgent achieves a 50% improvement in task success rates on real websites and SoTA performance on benchmarks like Mind2Web. This work demonstrated how modularity is highly beneficial for web automation, nonetheless, the authors also mentioned how this modular strategy may increase latency and computational overheads. They also mentioned additional limitations, including challenges with the generalization capabilities of the system and the complexity and high costs associated with evaluating autonomous agents in real-world scenarios. It is also important to note that while the execution of generated code offers flexibility, it may be less reliable and secure in production environments, where harmful operations or dangerous behaviors could unexpectedly arise.

C. World Model Approaches

In contrast, other even more recent and promising works adopt world model-based approaches, formulating web navigation as an environment dynamics problem. Chae et al. [18] propose a World-Model-Augmented (WMA) framework that equips LLM agents with the ability to simulate outcomes of their actions before execution, predicting the next state of the system, thus enhancing decision-making and exploration. Their solution introduces a novel transition-focused observation abstraction, which simplifies the training of world models by summarizing state changes in natural language rather than relying on full HTML structures, allowing agents to focus on key differences between states, minimizing computational overhead. The agent leverages WMA simulated outcomes to select the most optimal action during inference, without requiring additional training of policy models. Evaluations of this approach showed promising results. Nonetheless, the authors highlight some limitations and potential improvements: The framework focuses solely on text-based models, overlooking the integration of visual information that could enhance web navigation. Additionally, while it performs well in single-step decision-making, it lacks multistep planning capabilities. Future work could address these by incorporating visual data and exploring advanced planning techniques, such as Monte Carlo Tree Search, to improve long-term decision-making.

Similarly, Gu et al. [17] introduce WEBDREAMER, a framework that leverages LLMs as implicit world models for web navigation. Following a strategy similar to that of previous work, the system uses LLMs to simulate and evaluate action outcomes before execution, using two modules for multistep planning. WEBDREAMER generates concise natural language descriptions to represent predicted state changes, enabling efficient trajectory planning and action simulation while reducing the risks associated with direct interactions on live websites. Evaluations reveal significant performance gains over reactive strategies. Moreover, the authors highlight the advantages over tree search methods, which, although achieving slightly higher accuracy in controlled environments, are impractical for real-world web settings due to the possibility to perform irreversible actions. Alongside its advancements, the authors acknowledge certain limitations in their approach. The simplicity of the planning algorithm, while effective in demonstrating the concept, leaves room for refinement through more sophisticated techniques like Monte Carlo Tree Search. Additionally, the reliance on state-of-the-art models such as GPT-4 results in high computational costs, highlighting the need for future work to explore more cost-efficient alternatives, such as fine-tuned models designed for simulation tasks.

D. Web Navigation as a State-Machine

A further line of research, more aligned with the methodology of the presented work, focuses on modeling web navigation as a state machine, treating websites as deterministic systems where transitions occur via predefined action spaces. Deng et al. [27] present MIND2WEB, a dataset and framework that defines web navigation tasks as sequences of state transitions, where each state is represented by a webpage and actions correspond to interactions like clicking, typing, or selecting elements. Their methodology involves a two-stage system, MINDACT, which first filters relevant DOM elements using a smaller LM to create a small and relevant snippet of the website. Then, it employs a larger LM to predict the optimal action and the target element for interaction. In this case, the authors also highlight certain limitations and challenges faced by their proposal. One key issue lies in the diversity and representativeness of the data, as most web pages and tasks included in the dataset focus on English-speaking contexts and are carried out by users with high technological proficiency. Additionally, the system currently relies solely on textual information, overlooking the potential of visual data to enrich webpage representations. They also point out the need for more effective modeling of interaction dynamics, such as changes in the environment following previous actions, and for incorporating more interactive configurations between the agent and humans, since the system only relies on a single description of the task goal.

A similar approach is LASER, proposed by Ma et al. [21], which models interactive tasks as state-space exploration, explicitly defining a finite set of high-level states that encapsulate the structure of web environments. Each state includes a restricted action space tailored to the state's context, reducing the likelihood of invalid actions and ensuring the agent remains within the bounds of valid transitions. This state-action mapping is complemented by a thought-and-action framework inspired by ReAct, where the agent generates intermediate "thoughts" to guide decisions and stores intermediate results in a memory buffer for backtracking and

error recovery. LASER demonstrated improved performance over previous methods, but still inferior to human baselines. Additionally, the authors describe several limitations of the system. The followed strategy is task-specific, and LASER's scope is restricted to finding target items in the shopping domain, leaving out other common e-commerce tasks such as tracking orders or checking order history. Moreover, the system relies on manual annotation of possible states and their descriptions, which limits its applicability to specific domains and makes it less suitable for open-world scenarios, suggesting the usage of a hierarchical multi-agent system where domain-specific agents like LASER collaborate with a general open-world agent.

Another interesting contribution presents WEBARENA [28], a benchmark and highly realistic environment designed to evaluate web navigation agents in tasks like e-comerce, social forum discussion etc. The research work also introduces an agent-based system which relies on modeling web interactions as transitions within a deterministic state space. Each state represents a webpage, defined through observations such as HTML DOM trees, accessibility trees, or screenshots, and transitions occur based on predefined actions like clicking, typing, or navigating through URLs or tabs.

In their experiments, the authors evaluate several state-of-the-art agents, including GPT-4 and GPT-3.5, using both direct action prediction and chain-of-thought prompting. Despite advancements in model capabilities, the results reveal a pronounced performance gap between agents and human benchmarks. The best-performing agent, GPT-4 with chain-of-thought prompting, achieves a success rate of only 14.41%, far behind human performance at 78.24%. This disparity becomes even more pronounced in complex, long-horizon tasks.

Additionally, the authors pointed several limitations of LLM-based agent systems: One prominent issue is the frequent misclassification of achievable tasks as unachievable, with GPT-4 erroneously marking 54.9% of such tasks as impossible. Additionally, agents often fail to generalize across tasks derived from the same template. These results illustrate the weaknesses of current agents in handling task variance, dynamic content, and multistep planning. Moreover, agents exhibit a lack of robust error recovery, often repeating invalid actions or stopping prematurely when encountering ambiguities.

These limitations underscore the complexity of this task and the challenges associated with navigating the dynamic and highly diverse nature of web environments. The dynamic content of real-world websites, the variability in user interfaces, and the need for long-horizon planning present significant obstacles to scalability and robustness. Despite recent advances, the performance of autonomous agents in realistic web environments remains suboptimal, requiring further exploration of multi-agent systems, state-based modeling, and innovative methodologies to close the gap between human and agent performance.

E. Current SOTA and Limitations

As mentioned, research into the use of LLM-based agents for web navigation is still in its early stages. Due to the growing interest in this area, several recent studies, such as the discussed ones, aimed to establish foundational methodologies and benchmarks to address the challenges inherent in these tasks. However, most existing works highlight significant limitations that underscore the complexity of the problem, including the dynamic and ever-changing content of real-world websites, the considerable variability in websites, and the necessity for long-horizon planning. These factors create substantial obstacles to achieving accuracy, scalability and robustness, even for state-of-the-art systems.

Consequently, while the field has made promising advances in developing structured frameworks, there remains considerable room for improvement. Current systems, despite their advancements, fall short of replicating human-level performance in web navigation tasks. Thus, further research is needed, leaving opportunities for innovation and refinement.

III. METHODOLOGY

A. Proposed Task

Web navigation is an essential, generic skill underlying a multitude of specific tasks commonly performed online. These tasks include a variety of domains, such as e-commerce (e.g., browsing products and completing purchases), reservation systems (e.g., booking flights, hotels, or restaurants), research activities (e.g., retrieving academic papers or exploring literature), and information retrieval (e.g., finding updates on specific topics, acquiring technical documentation or seeking for specific data). Despite their differences, these tasks share a foundational requirement: the ability to navigate through and interact with web environments effectively.

To address autonomous web navigation agents, some research efforts aim to develop universal agents with maximum flexibility and autonomy [17], [27]. These agents are designed as general-purpose systems capable of solving any web navigation and interaction task, regardless of domain. The primary goal of such universal agents is to achieve a high level of generalization, enabling them to dynamically adapt to new tasks and environments without requiring task-specific tuning.

In contrast, other approaches focus on more targeted solutions, designing agents specialized in solving domain-specific or task-specific problems. For instance, LASER [21] concentrates on the particular domain of e-commerce. Such domain-specific systems emphasize task precision, often achieving higher performance within their narrow scope. Additionally, to scale these tasks-specific systems, there are suggestions to integrate them into hierarchical architectures controlled by a universal agent, balancing specialization and flexibility by allowing domain-specific agents to execute their respective tasks while being orchestrated by a higher-level, general-purpose agent.

The task of web navigation is extremely complex, even for current state-of-the-art systems, thus, a divide-and-conquer approach may prove beneficial, decomposing the problem into smaller, task-oriented solutions. This strategy exploits that, while the same task performed across different websites exhibits significant variability, it also possesses many similarities. By focusing on these shared characteristics, systems

can achieve superior performance on addressing specific tasks, maintaining website generalization. For instance, the process of booking a flight shares fundamental interaction patterns, regardless of the specific airline's website design. Leveraging these patterns allows systems to abstract the task and adapt effectively to varying contexts.

Furthermore, different web navigation tasks often share numerous interaction similarities, such us hyperlink navigation, form usage and information seeking. A system designed to perform one specific task correctly can provide valuable insights into designing systems for other related tasks. For example, an agent developed to navigate e-commerce sites for product selection may contribute to the development of systems capable of handling reservation processes. These shared interaction flows highlight the potential for cross-task generalization and the reuse of successful methodologies across domains.

Considering these factors, this work is framed as a domain-specific approach, focusing on the application of autonomous agents to a concrete task: retrieving academic papers from open scientific repository websites such as arXiv, bioRxiv, medRxiv etc. However, it is important to note that while this specific task serves as the primary use case, the underlying objective of this study is to explore the development and implementation of autonomous web LLM-based agents. The chosen task provides a scenario to test and evaluate the system's capabilities.

One of the objectives of this work is to be tested in real web environments. However, one of the most limiting and challenging factors of real-world websites is the dynamic content generated with JavaScript, as well as restrictive entry conditions such as login or registration forms and access policies that block automated interactions. These challenges significantly increase the complexity of developing and testing such systems.

To create a realistic yet simplified experimental environment, this study focuses on exploring and navigating preprint repository websites, since, while these websites differ substantially in their specific HTML structures and DOM selectors, they offer several advantages for this work. Many of them are openly accessible, eliminating the need for complex login forms, and their content is publicly available for exploration. These characteristics make them a suitable choice for a project of this nature, avoiding some web complexities out of the project's scope.

Thus, the proposed use case is an automated web navigation system that employs LLM agents to search for, collect, and present academic publications to users, addressing queries expressed in natural language. Additionally, the proposed system is designed to generalize to different websites, decoupling itself from the website-specific dependencies that traditional systems often face.

The proposed system, as a web navigation and interaction framework, is designed to operate within specific given websites, performing focused and bounded searches. This differentiates it from other systems that aim to conduct general web searches through the World Wide Web, to answer broad-domain questions. In the developed system, exploration occurs within a given domain by interacting with and navigating the specified website. However, to demonstrate the system's generalization capabilities, in scenarios where no specific website is predefined for exploration, the agents are designed to select appropriate websites to address the given query. This is achieved by accessing a pool of websites from various domains and conducting the search across one or multiple selected ones, depending on the specific configuration.

B. Proposed Task: Additional Considerations

As outlined, the proposed system focuses on solving a domain-specific task, resembling the approach used in LASER [21]. However, it is key to emphasize that the primary objective of this work is not to resolve the task of retrieving academic papers itself. Instead, the task serves as a convenient use case to explore the autonomous web navigation capabilities of the agents. This selection aligns with the project's scope and addresses the complexities of real-world web environments in a manageable way.

However, along with the advantages of the proposed task, there are some challenges and considerations:

1) Long-Horizon Research Task: Retrieving highly relevant results from preprint repository websites presents a particularly challenging problem for autonomous web agents. The large number of potential candidates makes exploratory searches complex, requiring decisions on when to stop searching, whether the results found are sufficiently relevant, or whether to continue exploring. Additionally, if a search proves unproductive, the agent must determine whether to give up entirely or refine its strategy. Such decisions represent significant challenges frequently encountered in long-horizon tasks, as noted in prior studies [28], and are particularly pronounced in the proposed task.

2) API Usage: As discussed, exploratory navigation through websites may not be the most efficient method for retrieving publications due to the long-horizon challenges described earlier, which arise from the sequential or batch retrieval of articles. However, many preprint repositories provide APIs that offer a more efficient alternative, leveraging their internal search engines to query entire document databases. Despite this, several critical considerations arise:

- A system that relies on API usage instead of web interactions, lacks generalizability to other domains, tasks, or websites that do not offer APIs, conflicting with the primary objective of this work.
- Furthermore, interacting with APIs frequently requires executing code generated by LLMs to communicate with the APIs. However, this process relies heavily on the model's understanding of the API specifications, which depends on its training data and may not always reflect the most up-to-date information. Additionally, LLMs can produce hallucinations, potentially generating code that is incorrect or harmful. This introduces significant security risks, as the generated code could unintentionally cause vulnerabilities, lead to resource misuse, or even execute malicious actions if not properly validated.

For these reasons, this study avoids the use of APIs, opting for an approach that does not rely on LLM-generated code execution and is more generalizable across domains.

C. Autogen: Development Framework

AutoGen [29], [30], developed by Microsoft in collaboration with academic institutions such as Penn State University and the University of Washington, is an open-source framework designed to streamline the development of AI systems powered by LLMs. It is particularly suited for creating multi-agent systems, supporting both cloud-based and local execution environments. AutoGen is very flexible, enabling to build complex agent-based applications, leveraging typical design patterns and facilitating customization.

One of the standout features of AutoGen is its support for integrating tools and enabling diverse conversational patterns between agents:

- *Two-Agent Chat:* This is the simplest interaction pattern, involving a straightforward exchange between two agents. It is ideal for tasks requiring minimal collaboration.
- *Sequential Chat:* This pattern allows a sequence of interactions, where the output or summary of one conversation is passed as context to the next. It is particularly useful for workflows broken into sequential subtasks, enabling continuity and context retention.
- *Group Chat:* AutoGen supports multi-agent conversations, orchestrated by a *GroupChatManager*. This manager broadcast messages to other agents and decides the next speaker in the group using strategies such as round-robin, random selection, manual intervention, or LLM-based decision-making. This flexibility allows dynamic collaboration among agents, adapting to the needs of the task.
- *Constrained Speaker Selection:* For applications requiring stricter control, AutoGen enables developers to define allowed or disallowed transitions between agents, effectively modeling deterministic workflows.

AutoGen not only facilitates the creation of state machines or transition graphs, but also enables the development of custom interaction flows targeted to specific tasks, offering further adaptability. This capability is particularly valuable for projects like the one presented here, since as stated in [21], the web navigation task may benefit from structured and controlled interaction flows.

In this context, and given the characteristics of this project, AutoGen has been chosen as the primary development framework due to its flexibility, different levels of customization in agent interactivity, and its focus on modularity and extensibility. This allows workflows to be encapsulated within agents, which can then seamlessly participate in larger workflows, thus enabling reuse and composability. Additionally, it is important to highlight that, as the field of agent-based systems is still relatively new, frameworks like AutoGen are under constant evolution, introducing new features but also presenting challenges, such as occasional bugs or outdated documentation that may hinder development efforts.

D. Used LLMs

In recent years, the surge of interest in LLMs has led to the development of a wide range of different models, both proprietary, such as OpenAI's GPT series, and open-source alternatives like Meta's LLaMA [31] and Alibaba's Qwen [32]. Each model offers different advantages and trade-offs in terms of performance, cost, and usability, targeted to different use cases and application requirements.

In the context of this project, which focuses on developing an autonomous web navigation system, several key factors influenced the choice of LLMs:

- *Speed*: Web navigation tasks are inherently token-intensive due to the need for processing large HTML documents to extract the contextual information and web structure needed for decision-making. Additionally, after executing actions, the system must observe changes in the environment to assess the effects of its interactions. These iterative processes demand high text-processing speed, as slow models could significantly hinder the efficiency of web navigation tasks.
- *Reasoning*: The system must interpret the environment, make decisions, and generate plans to achieve user-defined objectives. Advanced reasoning capabilities are crucial for evaluating alternatives, aligning actions with goals, and overcame errors or inefficient navigation paths. Models with robust reasoning abilities can significantly enhance the system's effectiveness.
- *World Knowledge*: Successful interaction with the web requires the ability to interpret webpages, including their structure and functionality. This knowledge can be obtained from specialized models, fine-tuned on HTML data, or from large-scale datasets containing web-related content. Such expertise enables the system to accurately model and understand the dynamics of web environments.
- *Resource Constraints*: Due to the limited hardware resources available, deploying large-scale LLMs locally is not feasible for this project. This constraint leads to reliance on cloud-executed models, which can accommodate the computational demands of web navigation tasks while offering scalability.
- *Cost*: As previously mentioned, web navigation tasks are highly token-intensive, emphasizing the importance of balancing cost and performance. Selecting models that optimize this trade-off is essential for sustainable deployment.

Based on the previously stated considerations, OpenAI's GPT models were selected as primary candidates for this project. Their performance and versatility make them well-suited to address the requirements of this work. Thus, the following models were selected:

- *GPT-4o*: It represents one of OpenAI's state-of-the-art models, excelling in advanced reasoning and knowledge-intensive tasks, outperforming vanilla GPT-4 with a more optimized processing. It has been employed in this project for complex scenarios requiring strategic planning, decision-making, and in-depth contextual understanding. GPT-4o's capabilities have been benchmarked

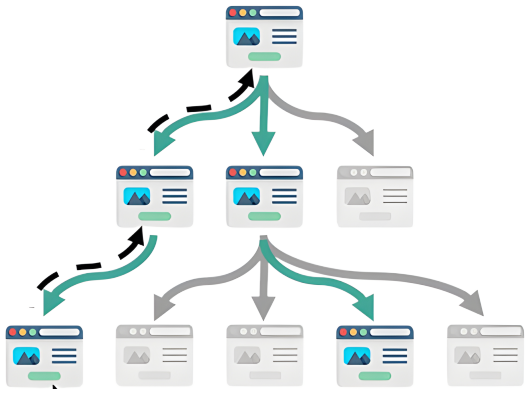


Fig. 1: Tree-search based Web Navigation. [17]

extensively, achieving SOTA performance in reasoning-centric evaluations such as the MMLU [33] benchmark, proving its suitability for tasks with high complexity [34].
- *GPT-4o-mini*: It is a smaller variant of GPT-4o, optimized for faster token processing and reduced operational costs. While it offers inferior reasoning capabilities compared to its larger counterpart, its performance remains adequate for simpler tasks that are token-intensive and require less reasoning complexity. In this project, GPT-4o-mini was employed for specific subtasks, particularly those involving extensive text processing or repetitive observations, where cost-efficiency and speed were prioritized over advanced reasoning [35].

While other LLM alternatives could potentially address the requirements of this project, the integration capabilities of OpenAI's GPT models with the AutoGen framework presented a noticeable advantage. Therefore, the selection of GPT-4o and GPT-4o-mini aligns with the project's goals and constraints, providing a balance of high reasoning capabilities, token-processing speed, and cost-effectiveness. These attributes, combined with their integration with AutoGen, display the suitability of these models for addressing the challenges of this work.

E. Explorative Search

As previously discussed, the proposed system of autonomous agents is designed to interact with specific websites to explore and extract information required by the user, while maintaining generalizability across different web environments. Humans perform this type of exploration frequently and with ease, adapting effortlessly to diverse user interfaces. This adaptability is possible in part due to core similarities present in most websites, enabling generalizable interaction flows. When interacting with preprint repositories to gather academic publications, there are two main search methods:

1) Search Forms and Advanced Search Mechanisms: Many preprint repositories provide search forms or advanced search tools that allow users to query indexed documents using standard information retrieval techniques, such as keyword searches. These tools often include advanced options for filtering and sorting results, enabling more efficient and precise searches. For a web-based agent to leverage these search mechanisms, it must be capable of:

- *Accessing Search Forms*: In some cases, search forms or advanced search interfaces are not directly accessible from the website's index page, requiring the agent to navigate through the website to locate them.
- *Identifying Interactive Elements*: The agent must correctly identify and interpret the various interactive elements within the forms, such as text inputs, date pickers, radio buttons, dropdown menus, submit buttons, etc.
- *Interacting with Form Elements*: The system must possess the ability to interact with these elements appropriately, ensuring that inputs are correctly formatted and consistent with the fields' expected data types.

Given these requirements, implementing a system capable of dealing with advanced search forms represents a significantly complex task. It demands high levels of interactivity, robust web comprehension, and consistency when filling forms, as errors or mismatches in input formats can compromise the search process.

2) Search Through Exploration of Accessible Hyperlinks: The second common search pattern involves exploring hyperlinks within a website to locate relevant content. This iterative process typically begins on the website's main page, where users evaluate navigation options, click on hyperlinks to access specific sections, and analyze the content of each page. This approach can be compared to a tree traversal process as shown in Fig.1, where the main page acts as the root node, and the hyperlinks represent edges connecting to child nodes (i.e., subpages). Users navigate the website by exploring the tree, where each click to a hyperlink is equivalent to moving to a child node and returning to a previous page mirrors backtracking. An autonomous agent capable of leveraging this navigation process require several capacities:

- *Evaluating Page Content*: Agents must assess whether the accessed page contains the objective information.
- *Identifying Navigation Paths*: They must analyze available hyperlinks or navigation entries to decide which path might lead to the target content.
- *Backtracking When Necessary*: If a chosen path proves ineffective, agents should backtrack to explore unvisited routes, reconsidering their navigation strategy.

This exploratory pattern is widely applicable across most websites on the internet, and preprint repositories are no exception. These repositories generally organize their content into categories and subcategories, which are often accessible through URLs. It is noteworthy that, while this approach is less efficient than advanced search tools for retrieving relevant academic publications, exploratory navigation is sometimes essential. For instance, advanced search forms may not always be available directly on the main page, so an initial exploration is needed to find them. Additionally, not all information on a website is indexed by its search engine, certain content may only be accessible through navigation.

Therefore, although the specific proposed task in this work is retrieval of academic papers, the real objective of the project

is to explore autonomous agents navigating the web. As such, this exploratory method is considered a priority.

Moreover, since the creation of a fully interactive system capable of solving all these challenges exceeds the scope of this project due to its complexity, this work focuses on addressing only the second pattern: Navigation through the exploration of accessible hyperlinks.

F. Search Context

In the proposed solution, the autonomous agents iteratively explore a given website by following hyperlinks, navigating a tree-like structure as outlined in the previous section. At each stage, the agent analyzes the content of the current page, determines whether relevant information is present, and—if necessary—follows embedded hyperlinks to potentially reach more pertinent sections. To accomplish this, the agent must maintain a record of which pages have been accessed and which potentially relevant links remain unexplored, as well as any relevant documents collected so far. This record has been called *Web Context*, denoted by W.

1) Web Context: Let's define W as a structure that captures the local state of exploration for a single website. Specifically, it tracks:

- The set of already discovered and not yet visited, potentially relevant URLs.
- The set of already visited URLs,
- Any relevant papers identified so far.
- A priority queue of URLs awaiting further exploration.

Thus, let $\mathcal{U}$ represent the set of all possible URLs, and let $\mathcal{P}$ be the set of URLs pointing to paper entries. Then, W can be expressed as:

$$W = \langle C,\ CP,\ DP,\ DU,\ VU \rangle,$$

where:

- $C \subset \mathcal{U}$ is the priority queue of *collected URLs* that are pending analysis.
- $CP : \mathcal{P} \rightarrow$ Desc., is a partial function storing *collected papers* and their descriptions.
- $DP \subset \mathcal{P}$ is the set of *discovered paper URLs*, used to avoid duplicates in CP.
- $DU \subset \mathcal{U}$ is the set of *discovered URLs* (whether visited or not).
- $VU \subset \mathcal{U}$ is the set of already *visited URLs*.

This structure ensures that each page is visited at most once and that the system can easily identify new hyperlinks. It also enables storage of all relevant encountered papers without repetition, and provides a mechanism for returning to previously discovered nodes to explore alternative routes if a chosen path proves unproductive. Essentially, whenever an agent follows one hyperlink but finds minimal relevant content, the other stored links from earlier pages remain in the priority queue, allowing the agent to *backtrack* and switch to a different path.

2) Extending to Multiple Websites: Search Context: While the *Web Context* (W) tracks the state for only one website, the proposed system also handles scenarios in which no specific website is provided. In such cases, the system dynamically chooses one or more relevant websites for the given query. To manage the broader state of the search across multiple sites, a *Search Context* is defined, denoted by S.

Let Σ^* be the set of all possible query strings, and let $\mathcal{U}$ again denote the universe of URLs. Moreover, let $\mathcal{P}_{\text{pool}} \subset \mathcal{U}$ represent the *pool* of websites from which the agent may choose based on relevance to the given user query. Thus, S is defined as:

$$S = \langle Q,\ P,\ \alpha,\ \beta,\ \gamma \rangle,$$

where:

- $Q \in \Sigma^*$ is the *user query* string.
- $P \subseteq \mathcal{P}_{\text{pool}}$ is priority queue of *pages to visit*, each drawn from the broader pool of accessible websites.
- $\alpha \in \mathcal{U} \cup \{\perp\}$ is the *current page* being analyzed or crawled (e.g., ArXiv).
- $\beta \in \mathcal{U} \cup \{\perp\}$ is the *previous page* visited.
- $\gamma \in \{\text{false}, \text{true}\}$ is the *completion state*, indicating whether the overall search has ended successfully or been interrupted due to an error or exceeding optional search limitations (e.g., the maximum number of pages to visit).

In this way, S provides a broader, query-level perspective of the search process. While each individual W controls the traversal of a single website, S maintains the global navigation flow, ensuring the agent can sequentially explore multiple candidate websites and terminate the search successfully once the requested content has been found or when no further progress is possible.

G. Agent Organizational Chart

To navigate the web and dynamically update the search context, the proposed solution employs a multi-agent system composed of LLM-powered agents. Each agent serves a specific role within the system for efficient and accurate data collection, processing, and presentation. This section presents a detailed description of each agent and their responsibilities within the system.

1) Web Selector Agent: Serves as the entry point for the multi-agent system, being responsible for identifying the most relevant web resources to initiate the search, which can be gathered directly from the user query or from a pre-defined website pool (i.e., P_{pool}).

Given that the task performed by this agent does not require high token consumption but demands a strong understanding of the user's query and objectives to select the appropriate websites, the LLM supporting this agent is GPT-4o.

2) Paper Collector Agent: It is the system's dedicated web scraper, tasked with retrieving and storing the main entries of research papers from a specified URL, by updating the *Web Context*. This agent plays a key role by identifying pages where abstracts, authors, publication dates, and other essential details are usually available. It uses the existing contextual information, such as paper titles, to ensure that only highly relevant papers are stored, filtering them based on their alignment with the user's query and scoring their relevance on a scale from 0 to 10. Finally, the agent briefly reports the collected papers. However, if no relevant papers are found,

the agent provides a brief summary of the page's content and suggests further navigation strategies.

It is noteworthy that the task assigned to this agent generally requires a high input token consumption, as it processes lengthy web pages seeking for entries to relevant papers. For this reason, GPT-4o-mini would appear to be a suitable choice to support this agent. However, experimental results have shown that this model tends to generate false positives when identifying relevant documents, hindering the search process and misaligning it with its main objective. This issue has been resolved by using GPT-4o, which is why this latter model is the selected one to support this agent.

3) Crawler Agent: It is the system's URL Extractor and Evaluator. Its primary role is to identify, evaluate and collect hyperlinks within a given page that may lead to relevant resources for the bibliographic search. This agent processes all navigation links on a given page, categorizing them based on their relevance to the user's query and assigning a relevance score. This agent ensures that the system can dynamically update the *Web Context*, enabling the discovery of potentially relevant navigation paths and the efficient traversal of the web's tree-like structure.

This agent, similarly to the Paper Collector, also has a high input token consumption. However, contrary to the previous case, GPT-4o-mini has proved to be powerful enough to perform this task effectively.

4) Planner Agent: It acts as the orchestrator of the navigation task, managing the workflow and coordinating the actions of other agents. It begins by breaking down the user's query into actionable steps and iteratively consulting the top k most relevant URLs from the priority list W_C. At each step, it selects which page to explore next to continue the search. This way, the Planner coordinates navigation, leveraging the capabilities of other agents and receiving reports on their tasks.

When sufficient data has been gathered, it delegates the responsibility to the Supervisor Agent for final evaluation. By overseeing the entire process, the Planner Agent ensures that the system operates efficiently and effectively, aiming to minimize unnecessary navigation while maximizing the quality of the search results.

As the main coordinator and decision-maker of the navigation process, the LLM used for this agent must possess high reasoning skills, broad world knowledge and strong contextual comprehension, as stated in Section III-D. Thus, the selected LLM to support this agent is GPT-4o. However, one of the challenges of using this model is its higher cost. Therefore, to mitigate this, the agent maintains a very low token consumption, as it does not directly consult the web pages. Instead, it obtains information about their content through summaries generated by the Paper Collector Agent.

It is worth mentioning that, although the Planner can consult the top k most relevant URLs according to the scores assigned by the Crawler in order to select the path to follow, it does not have direct access to these scores. This design choice is intentional, as the Planner has a broader context of the overall search process, and it also serves as a double confirmation mechanism for the relevance of the hyperlink. Furthermore, as previously noted, since the Planner is supported by GPT-4o, this double confirmation using a more powerful model helps to avoid paths misclassified as relevant by the weaker GPT-4o-mini used by the Crawler Agent.

5) Supervisor Agent: It is responsible for assessing the completeness and relevance of the gathered information. It uses a provided tool to retrieve both the list of collected papers and the user's query, this agent evaluates whether the search objectives have been met. If the collected papers fully satisfy the query, the Supervisor Agent approves the completion of the task. Otherwise, it provides constructive feedback, identifying gaps in the collected information and suggesting further actions. This agent ensures that the system's output meets the standards of quality and relevance before presenting it to the user.

Given the relevance of identifying correctly if the navigation task has been fully completed, the chosen LLM for this agent is GPT-4o.

6) Presenter Agent: Once the search is completed and approved by the Supervisor, the Presenter Agent retrieves the collected papers and compiles a detailed report, ensuring that the information is complete, accurate, and well-structured. It presents the data in Markdown format for clarity and readability, including essential details such as titles, authors, publication dates if available, and links to the sources. The Presenter Agent ensures that the final output is adjusted to the user's query, providing all relevant information in a clear and structured format, also aligning the result with the user's language.

It is noteworthy that, to compile all the information into a single report, the Presenter Agent must process extensive data, requiring a long context window and, ideally, a low token processing cost. For this reason, GPT-4o-mini has been chosen to support this agent.

7) Critique Agent: It is the final layer of quality control, tasked with reviewing the report generated by the Presenter Agent. This agent evaluates the report's structure, content, and references, providing actionable feedback to enhance its quality. It identifies any missing or weak points, such as incomplete citations or gaps in coverage, and suggests improvements to ensure the report fully addresses the user's query. By offering constructive critique, this agent helps the system to provide comprehensive, accurate, complete and well referenced outputs.

This task is key, as the feedback provided by this agent directly contributes to the generation of the final report delivered to the user. As mentioned, the report is created by the Presenter Agent using GPT-4o-mini due to its high token consumption. However, to enhance the results produced by this model, GPT-4o has been used as the Critique Agent, thereby improving the baseline outputs of GPT-4o-mini.

H. Hierarchical Organization

To allow interaction among the agents described in the previous section, they are arranged in a hierarchical structure, where the first level is organized as an AutoGen *Group Chat* [29], illustrated in Fig.2. This group of agents consists of two primary types:

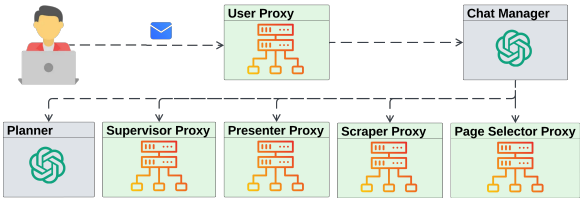


Fig. 2: Group Chat. Gray boxes indicate LLM-powered agents, while green boxes represent non-LLM proxies for accessing real agents or hidden workflows.

1) *LLM-Based Agents*: These include agents such as the Planner, which leverage the capabilities of LLMs.
2) *Proxy Agents*: These agents do not utilize LLMs directly. Instead, they serve as abstractions for more complex workflows, acting as proxies or entry points to access these workflows. Additionally, Proxy Agents can act as intermediaries for external elements, such as the user or tools.

Some of these proxies serve as entry points to a pipeline or sequence of actions known as a *Nested Chat* [29]. A Nested Chat is a feature of AutoGen that enables the encapsulation of agent conversations organized sequentially, hiding these interactions from the main chat. This makes it particularly suitable for scenarios where:

- Within a group chat, a specific event should trigger a task that can be divided into sequential subtasks, and it is preferable to isolate this task from the general conversation. Useful for hierarchical structures.
- Tool calls need to be encapsulated within a single agent for better organization and clarity.

For better understanding, the following subsections detail the purpose of the different proxies.

1) User Proxy: This proxy serves as the intermediary between the user and the system. Its main purpose is to process user inputs and forward them to the group. Additionally, it includes some automated messages that allow the system to receive instructions on behalf of the user in special situations (e.g., notifying of a forced interruption due to reaching the limit of subpages to visit on a single website).

2) Encapsulation proxies: Both the *Supervisor Proxy* and the *Page Selector Proxy* have a different purpose. In this case, since the tasks performed by these agents are independent and do not require external context about the navigation process, their operations are encapsulated within a Nested Chat. This approach prevents these agents from needing to process the entire navigation history to carry out their tasks. Furthermore, it allows for decoupling their tasks from the main conversation, simplifying the message history and isolating tool calls, making the latter transparent to the rest of the agents.

3) Scraper Nested Chat: Whenever the *Scraper Proxy* is invoked with a target URL from which to extract information and a query to guide the process, a sequential two-step procedure is triggered, illustrated in Fig.3a.

In the first step, the Crawler uses its tools to access the specified webpage and extract its content. The Crawler inspects the retrieved content, evaluating and collecting URLs that might lead to relevant resources for addressing the user's query. Subsequently, it uses a dedicated tool to update the *Web Context*, storing potentially relevant hyperlinks. These hyperlinks will later be accessed by the Planner for further navigation.

Once the Crawler completes its task, the Paper Collector performs a similar operation by accessing the cached web content to locate relevant academic paper entries. Using its assigned tool, the Scraper stores any relevant paper entries found. If no papers are identified, it reports their absence. Additionally, the Scraper generates a brief description of the visited page to provide context for the Planner, reporting either the papers collected or the lack of them.

It is worth noting that the provided tools programmatically manage the *Web Context* to ensure correct usage, relieving the agents of this responsibility and thus avoiding potential inconsistencies.

4) Presenter Nested Chat: Similarly, once the search process is completed and the Presenter Proxy is invoked, a sequential process detailed in Fig.3b is triggered.

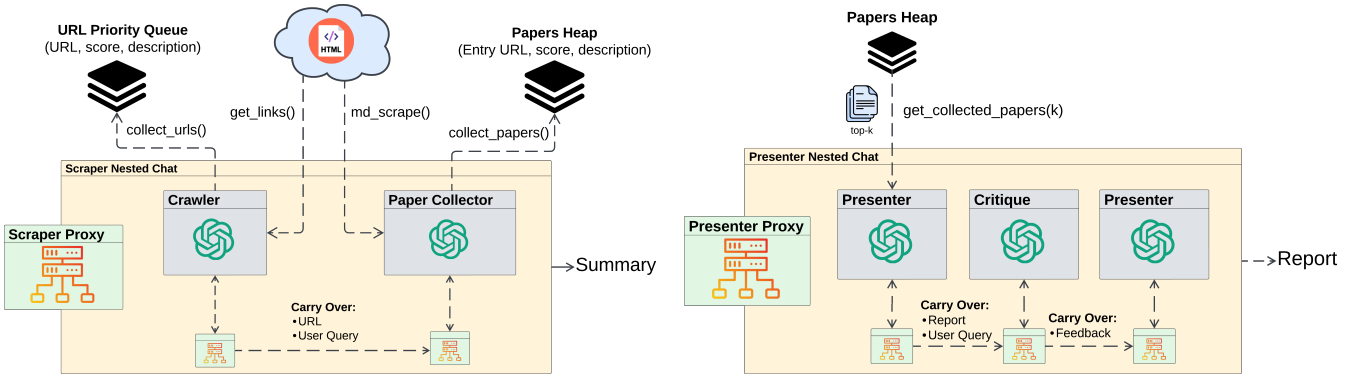
Since the search process often involves multiple pages, there are generally more papers collected than initially requested, in those cases where a specific number was provided (except for high numbers, where the search process may have been interrupted before reaching the target). Therefore, the first step of the Presenter Agent is to retrieve the top k most relevant papers. This is achieved using a dedicated tool that accesses each paper entry and extracts its content, usually including information such as title, abstract, authors, publication dates, and relevant links (e.g., direct access to the full content in PDF format, among others).

At this stage, the previously assigned scores for each paper are no longer relevant for ranking purposes, as these scores were determined based only on the publications within the contextual window of the LLM at the time of scoring. To compare all the collected papers simultaneously and rank them by relevance, the system employs the *BAAI/bge-reranker-v2-m3* [36] [37], a lightweight cross-encoder model for document retrieval ranking. This model is particularly suited for the task due to its strong multilingual capabilities and fast inference performance. It was chosen specifically for its high efficiency in multilingual tasks, as reflected in the *MTEB* benchmark [38] [39], enabling the system to maintain the multilingual capabilities of an LLM-based architecture while ensuring fast, local execution.

After obtaining the top k papers ranked by the cross-encoder, the Presenter Agent generates a preliminary report. This report is subsequently reviewed by the Critique Agent, which provides the necessary feedback to produce the final report to be delivered to the user.

I. Interaction Flow

In order to improve the system's reliability, making it more robust and less error-prone, a state-based flow was designed to select the next speaker depending on the previous speaker and the status of the *Search Context*. The flow begins with the user submitting a query through the *User Proxy*, which is first processed by the *Web Selector* to initialize S_P with the most relevant websites for exploration.



(a) Scraper Nested Chat (b) Presenter Nested Chat

Fig. 3: Sequential Nested Chats. [29]

Subsequently, the Planner initiates a navigation process, assisted by the Scraper: In this phase, each page is explored to collect papers until the Supervisor grants approval or the configured per-page exploration limit is reached. This procedure is repeated for all pages in S_p. Once these pages have been processed, the Presenter compiles all gathered information into the final report. High-level descriptions of this process are shown in Fig.4 and the pseudocode in Alg.1.

J. Additional Details

1) HTML to Markdown: As discussed, web navigation through the processing of HTML pages with LLMs is highly token-intensive. HTML files are inherently verbose, with a significant portion of tokens dedicated to structural elements such as tags, attributes, and inline scripts. This verbosity not only increases token consumption, leading to higher computational costs, but also introduces noise that can hinder the model's ability to focus on the relevant textual content.

To address this challenge, the proposed strategy is to parse HTML to Markdown before it is processed by the agents. This approach preserves all the relevant text content from the webpages, including hyperlinks, while significantly reducing the number of tokens required to represent the same information. This strategy is particularly viable in this scenario, as direct interaction with HTML elements, such as forms or dynamic components, is not necessary. In cases where such interactions are required, an alternative strategy would be needed, potentially integrated as a separate module.

For the web content extraction and the Markdown conversion, the system relies on *Crawl4AI* [40], an open-source project designed for efficient and lightweight web scraping and data extraction. *Crawl4AI* employs a custom conversion strategy that does not rely on language models, making it both fast and cost-effective. This ensures that the process remains scalable even when dealing with a high volume of webpages. By using this approach, we can efficiently process relevant information from webpages while minimizing token consumption and improving the overall performance of the system.

Algorithm 1 High-Level Interaction Flow Pseudocode

```
function SELECTSPEAKER(lastSpeaker, S)
    if lastSpeaker is PresenterProxy then
        return Report
    end if
    if Search in S_α is stuck then                ▷ Abort search in S_α
        if S_P then
            Move to next page
        else
            All pages done → Presenter compiles results.
        end if
    end if
    if lastSpeaker is UserProxy then
        if not S_P then                           ▷ First message. Select Webs
            Invoke WebSelector
        else
            Invoke Planner
        end if
    else if lastSpeaker is WebSelector then
        Invoke Planner
    else if lastSpeaker is Planner then           ▷ Planner decides
        Invoke ScraperProxy or Supervisor
    else if lastSpeaker is ScraperProxy then
        if per-page scraping limit reached then
            if S_P then
                Move to next page
            else
                All pages done → Presenter compiles results
            end if
        else
            Return control to Planner
        end if
    else if lastSpeaker is Supervisor then
        if S_α search completed then
            if S_P then
                Move to next page
            else
                Search Completed → Presenter compiles results
            end if
        else
            Return control to Planner
        end if
    end if
end function
```

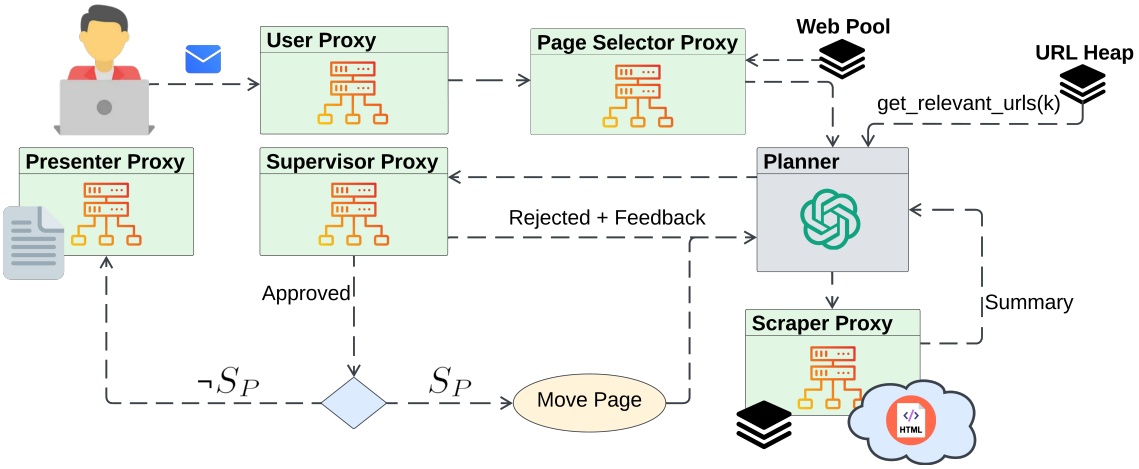


Fig. 4: High-Level Interaction Flow Representation.

2) Hint Injections: One observed problematic behavior in LLMs is the loss of focus on the task or specific details when processing long contexts. For instance, when the Planner performs navigation actions continuously, the past context passed to the model can introduce a bias, making it more likely to continue navigating instead of stopping, even if the search process has already been sufficiently productive. This behavior often leads to unnecessary repetition of previous patterns and inefficient use of resources.

This work propose a strategy to address this issue called *hint injection*. This technique involves injecting instructions or prompts that guide or reorient the model during critical situations. In our case, hint injection is primarily applied during function calls. When an agent invokes a tool and expects a response, the response is typically accompanied by a hint containing key instructions. Since these hints appear in the more recent part of the context, they are more likely to influence the model's behavior effectively.

For example, when the Planner retrieves URLs, it is explicitly instructed via a hint that if the search process is completed, control should be transferred to the Supervisor, avoiding unnecessary navigation. At the same time, in cases of early termination, the Supervisor can return control to the Planner, ensuring that the workflow remains efficient and avoids redundant actions.

Thus, by embedding hints into function calls, the effects of long-context bias are mitigated, improving task focus and the overall reliability of the system.[1]

IV. Evaluations and Results

A. Evaluation Methods

Evaluating the proposed system presents several challenges and limitations. One of the initial goals of this project was to test and evaluate the system in real-world web environments. As discussed, building a fully interactive autonomous web navigation system, capable not only of navigating through relevant URLs but also interacting with other web elements such as advanced search forms, is considerably complex, challenging even for SOTA works. For this reason, as mentioned in Section III-E, this work has focused solely on exploratory navigation through hyperlinks, treating it as a first step toward a more complete and fully interactive system.

This interaction constraint makes evaluation using some existing web agent benchmarks premature and challenging. Several of these benchmarks assess open-domain tasks, such as *WebArena* [27], including diverse tasks designed for general-purpose web agents, like e-commerce, booking, and information extraction tasks. In contrast, this work focuses on a task-specific solution. On the other hand, other benchmarks designed for specialized tasks, such as *WebShop* [25] for e-commerce, assume broader interaction capabilities than those presented in this work. Therefore, evaluating the system with these benchmarks would not be appropriate at this stage. However, as the system evolves and incorporates more advanced interaction capabilities, benchmarking against these standardized evaluation frameworks would be appropriate and could be considered in future work.

Given this context, an evaluation methodology that allows to assess the system's navigation capabilities in real-world preprint repositories is considered. There are two primary approaches for evaluating an agent-based system like the one proposed:

1) *Task-level evaluation:* This approach assesses system performance based on its success rate in completing the assigned task. It focuses on evaluating how well the final outcome satisfies the task requirements.
2) *Action-level evaluation:* Multi-agent systems typically involve a sequence of actions for task resolution. This evaluation method is more granular and aims to measure the correctness of each individual action, rather than just the final result. By analyzing the decision-making process at each step, this method provides deeper insights into the system's reasoning, limitations, and weaknesses.

Although action-level evaluation is generally more informative and provides a better understanding of the system's performance, its implementation often requires a corpus of an-

[1] Hint Injection Example: *"Select one of the above URLs to navigate. But remember: It's not necessary to scrape all URLs. If enough info has been collected, give the turn to the Supervisor."*

notated navigation paths or action sequences for comparison. Additionally, integrating this type of evaluation into AutoGen's workflow presents technical challenges, as it requires capturing and recording all actions taken by the agents. Due to these complexities, this work adopts the first evaluation approach, focusing on task-level assessment. Specifically, the system's performance is measured based on its ability to retrieve and present relevant publications that match the user query.

B. Used Metrics

An important aspect of the evaluation process, which introduces additional challenges, is the dynamic and ever-changing nature of the web. This, combined with the large number of relevant papers that could be potential candidates for retrieval, results in a scenario where multiple different responses could be equally valid and correct. In other words, the proposed task lacks a fixed *ground truth* or *gold standard* against which the system's responses can be directly compared.

This challenge is commonly encountered in information retrieval systems operating over large or continuously evolving document corpora. Due to these similarities, three widely used metrics from RAG systems have been selected to evaluate the proposed solution. These metrics, often referred to as the *"RAG triad"* [41], allow for the use of *LLM-based evaluators* to mitigate the absence of a predefined ground truth. They are defined as follows:

- *Answer Relevance:* Measures how closely the generated answer aligns with the user's original query, ensuring that the response is both directly relevant and sufficiently complete to fully address the request.
- *Context Relevance:* Evaluates how well the retrieved context matches the user's query, ensuring that the extracted information is relevant for answering the question.
- *Groundedness:* Assesses the extent to which the generated answer is supported by the retrieved documents. This metric helps determine factual accuracy and identify potential hallucinations.

Thus, an LLM with chain-of-thought (CoT) prompting is employed to evaluate each query across all metrics. To ensure evaluation consistency, the metrics rely on a binary success/failure approach for each query, rather than a broad scoring range, mitigating variability introduced by the LLM evaluator. Thus, aggregating results across multiple queries allows these metrics to be interpreted as a success rate. Additionally, each evaluation includes the LLM-generated CoT as a justification, enabling a more detailed analysis of the system's weaknesses. Notably, while LLM-based evaluation has limitations, prior research in information retrieval has shown that LLM-generated relevance labels can match human annotators in accuracy [42], making them a viable alternative when no fixed ground truth is available.

C. Experimental Setup

To automate the evaluation process, *LangSmith* [43] has been used. LangSmith is a framework-agnostic evaluation tool, making it suitable for various applications. Furthermore, to maximize the quality and reliability of the evaluations, GPT-4o has been selected as the LLM evaluator. Additionally, due to inference costs, and to ensure that searches do not become excessively long, the search per page has been limited to a maximum of five subpages to explore.

D. Evaluation Queries

To assess the system's performance, a set of 60 queries in English was created, each containing specific requests targeted at five different web pages. To ensure a controlled evaluation of system performance across diverse pages, all searches were conducted exclusively within the specified websites, avoiding multipage searches.

An important aspect to consider is the inherent difficulty of the task given the interaction limitations at the current stage. Since the system is unable to use advanced search forms, tasks requiring sorting or filtering of search results are not yet feasible. Additionally, queries that involve highly specific topics become nearly impossible to resolve solely through hyperlink navigation. These cases often require extensive exploration, making the search process extremely complex even for humans, given the same navigation constraints. However, it is important to highlight that these limitations could be mitigated in the future by adding form usage capabilities.

To properly evaluate the system performance in its current state, the queries were designed to focus on relatively general topics that allow for relevant retrieval through URL-based navigation. Specifically, for each website, 12 queries were proposed: Six *general-topic* queries, which facilitate information retrieval through standard navigation, and six more challenging and specific queries, which require more exhaustive searches, as their content is harder to locate and not always immediately accessible.

E. Experimental Results

The results in Table I show that the proposed system effectively navigates different websites, highlighting its generalization capabilities and its ability to retrieve relevant information in most cases. The system achieves an average *Context Relevance* score of *0.82*, indicating that it consistently retrieves contextually relevant content to user queries. A qualitative query-level analysis reveals that the system usually finds relevant information for more general queries, while its performance naturally declines for more specific ones. However, even when the system fails to locate relevant information, its navigation and decision-making processes are generally well-reasoned and appropriate, and it should be considered that the search performance is constrained by the imposed per-page search limitation and the available results in the sections. An interesting observation is that the *Planner Agent* often detects unsatisfactory search outcomes and suggests the use of an advanced search form, a functionality currently unavailable due to the absence of a module to handle such interactions. This suggests that extending the system with advanced search capabilities could significantly improve its effectiveness in future iterations.

On the other hand, the metric that exhibits the weakest performance is *Answer Relevance*, which averages *0.73* and shows extra variability across different websites. Notably, the system performs significantly better on *ArXiv*, which can likely be attributed to the platform's well-structured category-based URL navigation. ArXiv provides a broader range of categorized access points, facilitating more effective searches through URL-based navigation. Additionally, an analysis of failed cases reveals that some Answer Relevance failures are related to formatting issues rather than retrieval errors. Specifically, some queries request presentation in a particular format, such as displaying only titles and dates. However, in some cases, the *Presenter Agent* prioritizes its system prompt instructions over the user's query. The system prompt is designed to ensure that responses are complete and well-referenced when no specific formatting details are provided. However, in certain instances, it overrides explicit user instructions, leading to failures in Answer Relevance.

Finally, *Groundedness* is a critical metric as it assesses whether the system hallucinates information. Overall, the system appears relatively robust, achieving a Groundedness score of *0.92*. However, while this result is high, it leaves room for improvement given the importance of this metric. A closer examination of failed cases reveals that most hallucinations occur when the system provides data explicitly requested in the query but not found in the search. An example involves publication dates, which are sometimes hallucinated or contain minor transcription errors. More severe cases include hallucinated research papers. In some instances, if a precise number of papers is specified and not enough are found, the *Presenter* may correctly present the retrieved ones but hallucinate additional ones to meet the requested count. While this behavior appears to be infrequent based on the reported metrics, it remains a critical issue. Addressing this problem could involve implementing a hallucination detection module, which would enhance the system's robustness and reliability.

1) Additional Limitations: During the evaluations, several additional challenges have been identified that are worth mentioning:

- *Web Compatibility:* As previously discussed, the system relies on *Crawl4AI* [40] for web content extraction. While the system has been tested on various websites, incompatibilities have been observed in some cases due to outdated simulated browsers or JavaScript rendering issues on certain pages.
- *Response Time:* Restricting the system to URL-based navigation limits its capabilities and efficiency, increasing response times. Additionally, the sequential nature of the process, partially constrained by the Tokens Per Minute (TPM) limit imposed by OpenAI's API, further slows down the search process.
- *Cost:* Due to the high token consumption resulting from the observation-action pattern, the system incurs a non-trivial usage cost. This cost varies depending on the explored websites and configuration parameters, such as per-page search limitations.

TABLE I: Retrieval Results for Different Sources

Source	Answer Relevance	Context Relevance	Groundedness
Arxiv	0.92	1.00	0.75
Biorxiv	0.67	0.83	1.00
Chemrxiv	0.58	0.75	0.92
Medrxiv	0.67	0.83	0.92
Springer	0.83	0.67	1.00
Overall	**0.73 $\pm$ 0.14**	**0.82 $\pm$ 0.12**	**0.92 $\pm$ 0.10**

V. Conclusions and Future Work

This work presents a component of a potentially broader system for autonomous web navigation aimed at retrieving academic publications from scientific repositories. The proposed system focuses on hyperlink navigation, as it is a generalizable and fundamental task across multiple websites. The solution follows a domain-specific approach and implements a tree-traversal search pattern to explore relevant content.

While the current implementation exhibits limitations and challenges, some of these can be mitigated by incorporating additional modules to enhance interactivity, such as handling search forms. Despite the complexity of the task, the results are favorable and the findings suggest that LLM-based agents provide a promising solution for achieving autonomous web navigation, exhibiting adaptability across different real-world websites.

To further enhance the system, several improvements and extensions are suggested:

- *Enhancing Interaction Capabilities:* Introducing modules that allow interaction with search forms would enable the system to leverage built-in search engines and filtering mechanisms of academic repositories, significantly improving retrieval efficiency and effectiveness.
- Leverage *more Robust Evaluations* which not only assess final task completion but also analyze action-level decision-making. Once the system's capabilities expand, evaluating it against web navigation benchmarks will provide a more comprehensive comparison with SOTA approaches.
- *Optimization:* As access to SOTA LLMs with higher TPM limits becomes more affordable, optimizing search strategies—such as parallelizing searches across multiple relevant sites—could significantly reduce retrieval times.
- Implementing a *hallucination detection* mechanism would enhance the reliability of retrieved results, checking factual accuracy and minimizing generated errors
- *Exploring Multimodality* by integrating visual information processing, rather than solely relying on textual content, could improve navigation and interaction with dynamically generated webpages.
- Exploring new *emergent models* such as OpenAI's o1 [44] or DeepSeek [45], which leverage Test-Time Compute [46] for enhanced reasoning, could unlock more powerful interactions. Incorporating these models into key agent roles, such as the *Planner Agent*, may notably improve decision-making.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI*, 2019, accessed: 2024-11-15. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah *et al.*, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[5] Techopedia, "Todo sobre chatgpt: Estadísticas, tendencias, uso y predicciones," 2024, acceded: 18-01-2025. [Online]. Available: https://www.techopedia.com/es/estadisticas-chatgpt

[6] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman *et al.*, "On the opportunities and risks of foundation models," 2022. [Online]. Available: https://arxiv.org/abs/2108.07258

[7] OpenAI, J. Achiam, S. Adler *et al.*, "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[8] Q. Lu, L. Zhu, X. Xu, Z. Xing, S. Harrer, and J. Whittle, "Towards responsible generative ai: A reference architecture for designing foundation model based agents," 2024. [Online]. Available: https://arxiv.org/abs/2311.13148

[9] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021. [Online]. Available: https://arxiv.org/abs/2107.13586

[10] H. Jin, L. Huang, H. Cai, J. Yan, B. Li, and H. Chen, "From llms to llm-based agents for software engineering: A survey of current, challenges and future," 2024. [Online]. Available: https://arxiv.org/abs/2408.02479

[11] Y. Wang, W. Zhong, Y. Huang, E. Shi, M. Yang, J. Chen, H. Li, Y. Ma, Q. Wang, and Z. Zheng, "Agents in software engineering: Survey, landscape, and vision," 2024. [Online]. Available: https://arxiv.org/abs/2409.09030

[12] Y. Zhu, S. Wei, X. Wang, K. Xue, X. Zhang, and S. Zhang, "Menti: Bridging medical calculator and llm agent with nested tool calling," 2024. [Online]. Available: https://arxiv.org/abs/2410.13610

[13] M. Abbasian, I. Azimi, A. M. Rahmani, and R. Jain, "Conversational health agents: A personalized llm-powered agent framework," 2024. [Online]. Available: https://arxiv.org/abs/2310.02374

[14] L. Nan, E. Zhang, W. Zou, Y. Zhao, W. Zhou, and A. Cohan, "On evaluating the integration of reasoning and action in llm agents with database question answering," 2023. [Online]. Available: https://arxiv.org/abs/2311.09721

[15] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," 2022. [Online]. Available: https://arxiv.org/abs/2203.05115

[16] OpenAI, "Introducing chatgpt search," https://openai.com/index/introducing-chatgpt-search/, 2024, accedido: 19 de enero de 2025.

[17] Y. Gu, B. Zheng, B. Gou, K. Zhang, C. Chang, S. Srivastava, Y. Xie, P. Qi, H. Sun, and Y. Su, "Is your llm secretly a world model of the internet? model-based planning for web agents," 2024. [Online]. Available: https://arxiv.org/abs/2411.06559

[18] H. Chae, N. Kim, K. T. iunn Ong, M. Gwak, G. Song, J. Kim, S. Kim, D. Lee, and J. Yeo, "Web agents with world models: Learning and leveraging environment dynamics in web navigation," 2024. [Online]. Available: https://arxiv.org/abs/2410.13232

[19] H. Cai, Y. Li, W. Wang, F. Zhu, X. Shen, W. Li, and T.-S. Chua, "Large language models empowered personalized web agents," 2024. [Online]. Available: https://arxiv.org/abs/2410.17236

[20] K. Yang, Y. Liu, S. Chaudhary, R. Fakoor, P. Chaudhari, G. Karypis, and H. Rangwala, "Agentoccam: A simple yet strong baseline for llm-based web agents," 2024. [Online]. Available: https://arxiv.org/abs/2410.13825

[21] K. Ma, H. Zhang, H. Wang, X. Pan, W. Yu, and D. Yu, "Laser: Llm agent with state-space exploration for web navigation," 2024. [Online]. Available: https://arxiv.org/abs/2309.08172

[22] A. Ahluwalia and S. Wani, "Leveraging large language models for web scraping," 2024. [Online]. Available: https://arxiv.org/abs/2406.08246

[23] N. Corcuera, "Enhancing web scraping with large language models," https://dzone.com/articles/enhancing-web-scraping-with-large-language-models, 2024, accessed: 2025-01-19.

[24] E. Z. Liu, K. Guu, P. Pasupat, and P. Liang, "Reinforcement learning on web interfaces using workflow-guided exploration," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryTp3f-0-

[25] S. Yao, H. Chen, J. Yang, and K. R. Narasimhan, "Webshop: Towards scalable real-world web interaction with grounded language agents," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=R9KnuFlvnU

[26] I. Gur, H. Furuta, A. V. Huang, M. Safdari, Y. Matsuo, D. Eck, and A. Faust, "A real-world webagent with planning, long context understanding, and program synthesis," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=9JQtrumvg8

[27] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su, "Mind2web: Towards a generalist agent for the web," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. [Online]. Available: https://openreview.net/forum?id=kiYqbO3wqw

[28] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig, "Webarena: A realistic web environment for building autonomous agents," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=oKn9c6ytLx

[29] Microsoft, *AutoGen: An Open-Source Programming Framework for Agentic AI*, 2025, version 0.2. [Online]. Available: https://microsoft.github.io/autogen/0.2/

[30] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu *et al.*, "AutoGen: Enabling next-gen LLM applications via multi-agent conversations," *arXiv preprint arXiv:2308.07835*, 2024.

[31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[32] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng *et al.*, "Qwen technical report," 2023. [Online]. Available: https://arxiv.org/abs/2309.16609

[33] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," 2021. [Online]. Available: https://arxiv.org/abs/2009.03300

[34] OpenAI, "Hello gpt-4o," 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o/

[35] OpenAI, "Gpt-4o mini: advancing cost-efficient intelligence," 2024. [Online]. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[36] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," 2024.

[37] C. Li, Z. Liu, S. Xiao, and Y. Shao, "Making large language models a better foundation for dense retrieval," 2023.

[38] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," 2023. [Online]. Available: https://arxiv.org/abs/2210.07316

[39] HuggingFace, "Mteb leaderboard," https://huggingface.co/spaces/mteb/leaderboard.

[40] Unclecode, "Crawl4ai: Open-source llm friendly web crawler & scraper," https://github.com/unclecode/crawl4ai, 2025.

[41] L. Madzou, "What is the rag triad?" 2023. [Online]. Available: https://truera.com/ai-quality-education/generative-ai-rags/what-is-the-rag-triad/

[42] P. Thomas, S. Spielman, N. Craswell, and B. Mitra, "Large language models can accurately predict searcher preferences," 2024. [Online]. Available: https://arxiv.org/abs/2309.10621

[43] LangChain, "Langsmith sdk," 2025. [Online]. Available: https://github.com/langchain-ai/langsmith-sdk

[44] OpenAI, "Learning to reason with llms," *OpenAI*, 2024. [Online]. Available: https://openai.com/index/learning-to-reason-with-llms/

[45] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[46] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling llm test-time compute optimally can be more effective than scaling model parameters," 2024. [Online]. Available: https://arxiv.org/abs/2408.03314