SAGE: Scalable Academic Goal-driven Explorer

Introduction

This report presents a proposal for a fully autonomous, agent-based intelligent system named SAGE (Scalable Academic Goal-driven Explorer).

SAGE leverages artificial intelligence through an LLM-based agent chain. Its objective is to automate the research process - finding publications, streamlining referencing, and maintaining comprehensive records - to facilitate academic work such as literature reviews.

Development

Design Proposal

The proposed system implements a multi-agent architecture orchestrated through an LLM interface (Figure 1). Each agent specializes in distinct tasks, with data consistency ensured through strictly defined schemas. The system can leverage either locally deployed or cloud-based LLM solutions, with the final choice depending on cost-effectiveness analysis and performance requirements.

SAGE utilises NLP-based agents rather than symbolic (logic-based) agents. This decision was driven by the maturity and extensive support of modern LLM libraries and frameworks. Although symbolic agents excel at rule-based decision making, they struggle with unstructured data like academic papers and website content (Collins et

al., 2021; Shah, 2022; Sharma, 2024). By contrast, NLP-based agents are better equipped to understand and process natural language queries and academic text, while also offering more streamlined development through established libraries like LangChain (Bhatt and Vaghela, 2024).

Our development approach emphasises well-defined, interconnected components as essential to system efficiency, reliability, and robustness.

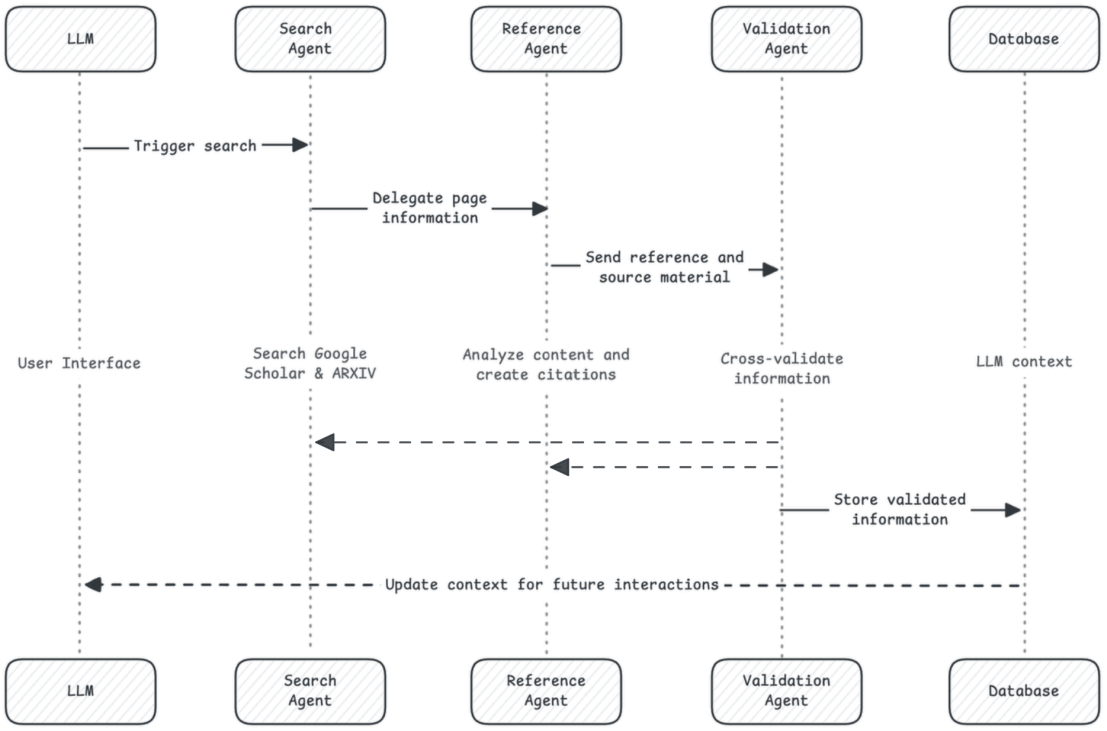

Fig. 1: SAGE system sequence diagram showing agent interactions and data flow.

Agents

SAGE will leverage agentic AI, a broad term encompassing systems that can autonomously pursue goals, make decisions, and take actions over extended periods (Mukherjee and Chang, 2025). As such, the system architecture is based upon three specialised agents working in sequence, each with specific behaviors and capabilities:

- The **Search Agent** independently searches academic databases like Google Scholar and ARXIV. Upon finding valid sources, it delegates the page information to the Reference Agent using predefined structured schemas to ensure data consistency.
- The **Reference Agent** analyses the received content and creates citations following a default referencing guideline. This agent transforms raw source data (webpage html code) into properly formatted academic references.
- The **Validation Agent** performs the critical role of cross-validating both references and page information from the previous agents, ensuring data accuracy and consistency. Upon successful validation, the information is stored in a vector database.

Our agent architecture implements a behavior-oriented approach anchored in a vertically layered, two-pass hybrid system that combines reactive and context-informed behaviour (Wooldridge, 2009). Each agent operates with structured output schemas to ensure data consistency and reliability between agent interactions, tool integration for specific task execution (e.g., API queries), and sequential processing for coordinated behavior chains across system components.

Database

The system utilizes a vector database for information storage, which is particularly effective for LLM-based systems as it enables semantic search and similarity matching of stored content (Rahgozar et al., 2025).

The database will be deployed locally during the initial development phase. This storage approach is integral to the system's efficiency. If the main LLM detects that relevant information already exists in its context (stored in the vector database), it will not initiate the agent sequence, thus preventing redundant searches and processing.

User Interface

The system's primary interface is an LLM that acts as both the user interaction point and workflow orchestrator. When users engage with the system, they communicate naturally with the LLM, expressing their research needs through conversational queries. The LLM interprets these requests and initiates the appropriate agent workflow. As shown in the sequence diagram, the LLM triggers the Search Agent to begin the process. After the Validation Agent stores information in the database, this data becomes part of the main LLM's context, enabling it to provide more informed responses in future interactions without necessarily initiating new searches for previously retrieved information. This architecture allows for a seamless user experience while maintaining complex behind-the-scenes coordination between agents and ensuring efficient resource utilisation.

Requirements

For agents to operate effectively, two key components are essential:

- **Structured output**: ensures data consistency and reliability across the system. Each agent must conform to predefined schema definitions that specify exact output formats - for example, a Search Agent result must always be an object, such as `{ url: string, title: string, summary: string }` (Figure 2). This strict schema enforcement ensures that data can be reliably passed between agents and tools while maintaining consistency and testability.

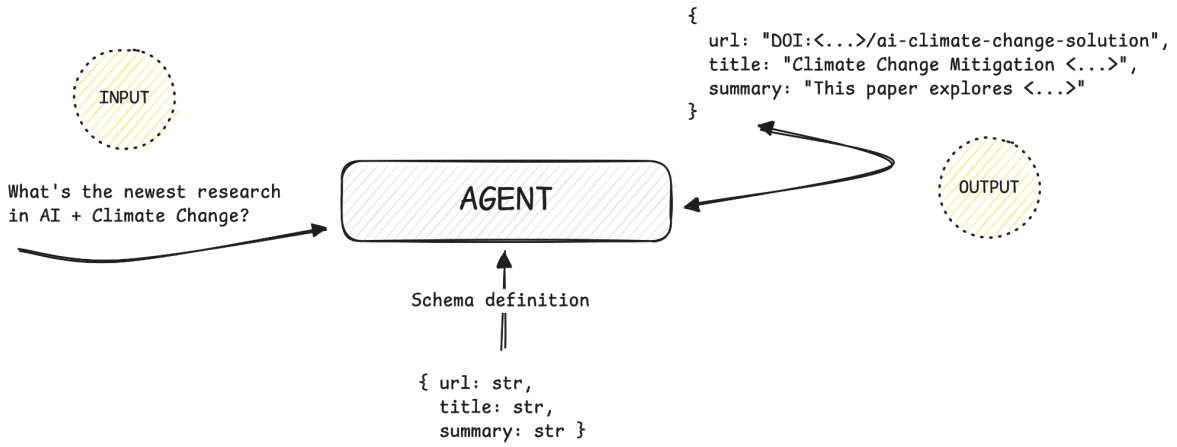

Fig. 2: Example of agent schema enforcement for structured outputs.

- **Tool integration:** extends agent functionality beyond basic model capabilities. Tools are specialised functions that agents can call to perform specific tasks - for example, querying academic databases, parsing PDF documents for reference guidelines, or accessing external APIs. When an agent needs to perform a task, it calls the appropriate tool function, receives the results,

reflects on them, and finally processes those results according to the predefined schema (Figure 3).

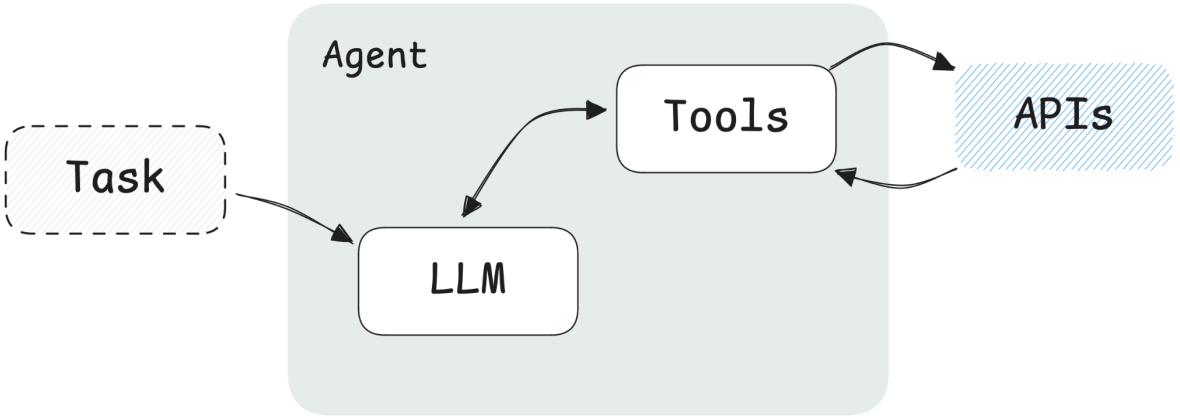

Fig. 3: Agent architecture showing task processing and tool integration.

Without structured outputs, the system cannot ensure consistent data flows between agents (or to tools). Without tooling capabilities, the system would be limited to basic model queries, rather than performing tasks that require real-time access to the internet or other functionalities that the model itself lacks or is not trained on.

SAGE will be developed using Python, leveraging the LangChain library as the system's foundation. LangChain provides a development stack that allows the use of most local and paid models to create reliable LLM agents, as most model wrappers support both structured outputs and tooling (LangChain, 2024).

Expected Challenges

A major technical hurdle comes from LangChain's rapid development cycle. The library is updated frequently, often introducing breaking changes that require code rewrites (Durrani, 2024). Even more challenging is the inconsistent feature support across different LLMs - some models support both structured output and tool use, while others only work with one or neither (LangChain, 2024). This makes it difficult to switch between models or providers without significant code changes.

Conclusion

SAGE aims to automate academic literature search using a multi-agent architecture built upon specialized LLM-based agents for searching, referencing, and validating results.

Its key strengths lie in its structured approach to data handling and tool integration, with strict schema enforcement ensuring reliable communication between components, and its efficient use of vector database storage. The use of an LLM as the primary interface enables natural, conversational interaction while managing the coordination of agents and their specialized tools behind the scenes.

The system faces significant technical challenges, especially with LangChain's frequent updates and inconsistent model support. However, the modular design allows for both adaptation to these changes and future expansion through additional specialised agents, such as plagiarism detection or research gap analysis.

References

- Bhatt, A. and Vaghela, N. (2024) *Med-Bot: An AI-Powered Assistant to Provide Accurate and Reliable Medical Information*. Available at: https://doi.org/10.48550/arXiv.2411.09648.
- Collins, C., Dennehy, D., Conboy, K. and Mikalef, P. (2021) 'Artificial intelligence in information systems research: A systematic literature review and research agenda', *International Journal of Information Management*, 60 (2021), Article 102383. Available at: https://doi.org/10.1016/j.ijinfomgt.2021.102383.
- Durrani, M.M. (2024) *Migrating Away from Langchain: Lessons Learned from Upgrading Our AI Codebase*. Available at: https://mubashirullahd.medium.com/migrating-away-from-langchain-lessons-learned-from-upgrading-our-ai-codebase-2f8fe1cb14f3 (Accessed: 25 February 2025).
- LangChain (2024) *Chat Integrations*. Available at: https://python.langchain.com/docs/integrations/chat/ (Accessed: 25 February 2025).
- Mukherjee, A. and Chang, H.H. (2025) *Agentic AI: Expanding the Algorithmic Frontier of Creative Problem Solving*. Available at: https://arxiv.org/abs/2502.00289.
- Rahgozar, A., Mortezaagha, P., Edwards, J., Manuel, D., McGowen, J., Zwarenstein, M., Fergusson, D., Tricco, A., Cobey, K., Sampson, M., King, M., Richards, D., Bodnaruc, A. and Moher, D. (2025) *An AI-Driven Live Systematic Reviews in the Brain-Heart Interconnectome: Minimizing Research Waste and*

Advancing Evidence Synthesis. Available at: https://doi.org/10.48550/arXiv.2501.17181.

- Shah, W. (2022) *Hybrid AI Models: Combining Symbolic Reasoning and Deep Learning for Enhanced Decision-Making*. Available at: https://doi.org/10.13140/RG.2.2.11493.61920.
- Sharma, A. (2024) 'Bridging Paradigms: The Integration of Symbolic and Connectionist AI in LLM-Driven Autonomous Agents', *Journal of Artificial Intelligence General Science (JAIGS)*, 6(1), pp. 138-150. Available at: https://doi.org/10.60087/jaigs.v6i1.237.
- Wooldridge, M.J. (2009) *An introduction to multiagent systems*. 2nd edn. New York: John Wiley & Sons.