Literature Review using multi-Agent system

Chaitrashree Venkatesh Bhat , Matriculation Numeber- 430875
RPTU Kaiserslautern, Department of Computer Science

Note: *This report contains a project documentation and reflection on the portfolio task submitted for the lecture Engineering with Generative AI in WiSe 2024-25. This report is an original work and will be scrutinised for plagiarism and potential LLM use.*

1 Portfolio documentation

The task had majorly three phases.

- Research phase
- Design phase
- Implementation phase

Below is the steps followed in each phase and the choice made for each phase to achieve the maximum output

- **Research Phase** -
  - We are asked to choose the free and open source model. I have choosed "llama-3.3-70b-versatile" model because it offers high capacity, versatility, better comprehension of academic language, and the ability to produce high-quality, coherent summaries, especially for longer or more complex texts. Its large size and training data make it well-suited for understanding the detailed content of research papers, delivering summaries that highlight key findings and insights effectively.

  - Collecting research paper and writting the literature review.Below are list of 6 research paper which I used for generating literature review.
    * Large Language Model Enhanced Multi-Agent Systems for 6G Communications
    * LLM experiments with simulation: Large Language Model Multi-Agent System for Simulation Model Parametrization in Digital Twins
    * Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems
    * Self-Adaptive Large Language Model (LLM)-Based Multiagent Systems
    * Navigating Complexity: Orchestrated Problem Solving with Multi-Agent
    * Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission

  - Numerical evaluation metrix-I have used rouge-score to compare the literature review. It provides a clear, reliable measure of how well the generated summary reflects the key content of the original paper. It evaluates both the precision and recall of important terms, phrases, and concepts, ensuring that the literature review is informative and accurate.

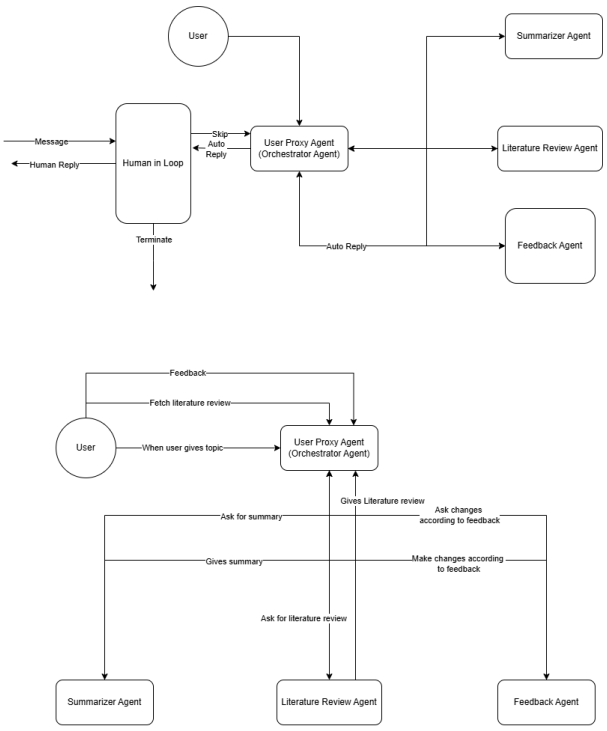

Figure 1: Design Overview

- **Design Phase**- Refer the Design Overview
  - User
    - Initiates the interaction by providing a topic or query.
    - Provides feedback for refinements.
  - User Proxy Agent (Orchestrator Agent)
    - Acts as the central orchestrator for all interactions.
    - Routes user requests to the appropriate agents (Summarizer Agent, Literature Review Agent, Feedback Agent).
    - Manages human-in-the-loop intervention and automation
  - Human in Loop
    - Acts as a manual intervention step where a human can review and modify responses.
    - Can either provide a reply or allow the system to auto-reply.
  - Summarizer Agent
    - Generates summaries when requested by the Orchestrator Agent.
    - Provides concise outputs of literature reviews.
  - Literature Review Agent
    - Retrieves and processes literature reviews based on the user's topic.
    - Returns the processed literature review to the Orchestrator Agent
  - Feedback Agent
    - Incorporates user feedback into the system
    - Suggests modifications to the Literature Review Agent and Summarizer Agent based on user feedback.

- **Implementation phase**-
  - *Integrated the LLM with the framework and tested the connection.* The Autogen framework is integrated by importing the necessary components (AssistantAgent and UserProxyAgent) from the autogen module. To ensures that the language model (LLM) is connected and ready to perform tasks. The Groq API key is used to connect the LLM with the system. The system loads the API key for connecting to the language model (LLM) service and verifies its availability. This is essential for connecting the LLM to the framework, ensuring that it can perform language-related tasks, like summarizing papers or generating literature reviews.

  - *Integrating tools-set up and connect the necessary tools required* : I have used Streamlit, to provide a user interface (UI). Streamlit is used to create the web app that allows the user to interact with the system. Streamlit helps facilitate user input.

  - *Building the agent — Integrate each agent one at a time and link them to LLM and tools.*I have defined the agents (like summarizer agent, feedback agent, and literature review agent) and integrate them into the system one by one. Each agent is responsible for a specific task, and they are linked to the LLM (Autogen framework) and

connected with Streamlit to facilitate interaction. Each agent is designed to handle specific tasks. For example:

  * Summarizer agent: Summarizes individual papers
  * Literature review agent: Generates a consolidated literature review based on multiple paper summaries.
  * Feedback agent: Refines the literature review based on user feedback.

- *Development of the chat mechanism, ensuring it uses max message limit for termination and human interaction in the loop.* The chat mechanism in this context allows the user to interact with the system through Streamlit. The system is designed to handle maximum message limits to prevent long loops, and it ensures that the user can provide feedback or interact during the process. After fetching and summarizing papers, the user is given the option to provide feedback on the literature review. The user can input feedback, and the system will use this feedback to refine the output (such as improving the literature review). This ensures a human-in-the-loop process, where the system can improve based on user guidance.

- *Test it using the papers in the research phase, include the human in the process, and record the process.*The system fetches research papers based on the user's topic input, summarizes them using the agents, and presents the results in the Streamlit UI. The user can interact with the literature review and give feedback for refinement. The system performs these tasks automatically, but it also allows the user to be involved, refining the process by providing feedback at any stage.

- *Use the metrics to evaluate the difference between your written process and the system-generated process.* The system compares the human-written summaries and system-generated summaries using metrics. The goal is to evaluate how well the system performs based on user feedback and compare the generated summaries with human expectations. As mentioned above I have choosed rouge-score to comapre the the literature review written by me and the system generated one. I am able to get Cosine Similarity of 0.6883 indicates a moderate similarity, implying the generated literature review is somewhat aligned with the reference but could still be improved in terms of exact content and word choice.

  Interpreting the Results: ROUGE-1 is reasonably good, with a precision of about 52.45percentage and recall of 47.77percentage, showing that the generated summary captures a decent amount of relevant content. ROUGE-2 scores are lower, indicating that the generated summary has a smaller overlap at the bigram level. This suggests that the generated text might be missing some higher-order relationships between words. ROUGE-L is moderate, reflecting that the generated summary has some structure and order that aligns with the reference summary, but it's not perfect. Cosine Similarity of 0.6883 indicates a moderate similarity, implying the generated summary is somewhat aligned with the reference but could still be improved in terms of exact content and word choice.

2 Reflection

1. What was the most interesting thing you learned while working on the portfolio? What aspects did you find interesting or surprising?

   Answer: I learned how to use multi-agent systems to achieve specific outputs and outcomes. It was fascinating to see how multiple agents can collaborate or work independently to reach a goal, and how the interactions between agents can be tailored to produce effective results One of the most interesting aspects was giving prompts to the agents and watching them work according to the specific instructions. I was surprised by how accurately the system could follow detailed commands and produce results based on those inputs. Additionally, finding the right modules to ensure everything worked efficiently was a rewarding challenge.

2. which part of the portfolio are you proud of why? what were the challenges you faced and how did you overcome them?

   Answer: I am particularly proud of the successful implementation of the multi-agent system to generate summaries. Initially, I tried using the Hugging Face module, but it didn't work as expected. Instead of summarizing the content, it gave me the entire paper as a summary. I faced additional challenges when I tried running these processes in Colab. I ended up losing account credits, even after creating multiple Google accounts, and reached the model credit limits. I then turned to OpenAI, but encountered similar issues. The prompts didn't give me the desired output, and I was only able to retrieve the first few lines of the paper as the literature review. The agents weren't properly understanding the content of the paper. Later, I downloaded VS Code to my system and used Groq to access the API, which worked much better. However, I still faced challenges when trying to extract literature summary from a set of six research papers. Despite these obstacles, I am proud of the fact that I was able to enhance the process and take the feedback from user and update the literature review accordingly and also this system can help to give literature review of any topic given by user.

3. What adjustments to your design and implementation were necessary during the implementation phase?

   - Refining AI Agent :Initially, the interaction between the summarizer agent, feedback agent, and literature review agent was not seamless. The system sometimes failed to properly refine summaries based on feedback, as the agents were operating somewhat independently. Implemented a structured message-passing system between agents to ensure each agent correctly received and processed the previous agent's response.Improved the feedback loop by aggregating multiple rounds of user feedback before modifying the literature review, preventing unnecessary reiterations.
   - Optimizing API Calls and Performance : The system was taking too long to process and summarize multiple research papers, leading to delays in generating the literature

review. Implemented batch processing for summarizing papers instead of processing them one by one.Optimized the LLM configuration by adjusting parameters to ensure efficient processing without sacrificing quality.

- Improving the Literature Review Generation Process:The literature review lacked structure and coherence when combining multiple research paper summaries. It often felt like a collection of independent summaries rather than a well-connected review. Implemented a structured format for literature reviews, ensuring that research trends, findings, gaps, and future directions were clearly highlighted. Adjusted the system message for the literature review agent to explicitly guide it in generating more structured reviews rather than listing disconnected summaries.
- Handling User Feedback More Effectively. The system was not incorporating user feedback effectively, as it only refined the last-generated review instead of considering all previous feedback. Implemented a feedback history system, storing all user feedback in st.session state.feedback history. Modified the feedback agent's prompt design to incorporate cumulative feedback rather than refining only the latest input.

4. From the lecturer which topic excited you the most? Why? What would you like to learn more and why?

   Answer:From the course point of view, the topic that excited me the most was Multiagent Systems in Large Language Models (LLMs).This is one of the reasons I chose this task for my portfolio. I already had some knowledge of LLMs and their workings from my BE project in the final year. However, I was not familiar with the concept of multiagents. Through working on Assignment 3 and the multiagent tasks, I gained a deeper understanding of the topic. It not only helped me expand my knowledge but also provided hands-on experience with implementing multiagent systems, which was truly exciting and rewarding. From guest lecturer the topic which cought my interest was AI or ML generative AI at Amazon. This topic is fascinating because it demonstrates how AI is revolutionizing industries—from retail and cloud computing to AI-driven customer service. The scale at which Amazon implements AI is impressive, impacting millions of users worldwide. They talked about Real-World Impact, Scalability and Innovation, Future of AI in E-Commerce. I would like to learn more on real world implementation of LLM.

5. How did you find working with the DIFY during the course of work and exam? Do you recommend it using it while learning Generative AI technology? Why? What is the best way of learning Generative AI either by python code or no code platform?

   Answer:Working with DIFY during the course of work and the exam was an interesting and efficient experience. The platform provided an intuitive interface that made it easier to implement and test various concepts related to Generative AI. It helped streamline some of the more complex tasks, allowing me to focus on the learning and development aspects without getting bogged down by low-level coding challenges. The hands-on nature of DIFY helped reinforce my understanding of the concepts by providing instant feedback and

results, which was particularly useful during exam preparation.Yes, I would recommend using DIFY when learning Generative AI technology. The platform provides a practical approach to implementing AI models without requiring extensive coding expertise. This is beneficial for beginners and those looking to quickly grasp the basics of Generative AI. DIFY allows users to experiment and prototype models, accelerating the learning process while offering valuable insights into the technology. It helps bridge the gap between theoretical knowledge and practical application. The best way to learn Generative AI depends on the learner's background and goals. If you have a solid foundation in programming and want to deeply understand how the models work, learning through Python code is highly recommended. This will allow you to gain more control over the models, customize them, and understand the underlying algorithms. However, if you're new to AI or want to quickly experiment with different models and concepts without delving into complex coding, no-code platforms like DIFY can be a great starting point. They provide a user-friendly environment where you can focus on learning the theory and practical application without getting overwhelmed by the coding details. For a more holistic understanding, I recommend using a combination of both: start with no-code platforms for a broader overview and transition to Python-based development as you build confidence and want to explore more advanced topics.

6. How did you find the assignment and the course how did they helped you for the portfolio exam?

   Answer: Assignment 2 and Assignment 3 were very helpful and allowed me to learn more about the concepts while gaining hands-on experience. I was able to get full marks and pass Assignment 2. I tried my best to understand Assignment 3 and worked hard on it. Since I chose Task 3 for my assignment, it helped me a lot. Assignment 3 provided me with a basic understanding of how to approach Task 3

[1] [2] [3] [4] [5] [6].

References

[1] R. Li, Z. Zhao, X. Wang, and M. Debbah. Intelligent 6g: When machine learning meets wireless networks. *IEEE Wireless Communications*, 27:56–52, 2020.

[2] Lyle Wallis and Mark Paich. Integrating artificial intelligence with anylogic simulation.

[3] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, and Percy Liang. Generative agents: Interactive simulacra of human behavior. *arXiv preprint*, arXiv:2304.03442, 2023.

[4] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, and Izhak Shafran. React: Synergizing reasoning and acting in language models. *arXiv preprint*, arXiv:2210.03629, 2022.

[5] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, and Thorsten Holz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *arXiv preprint*, arXiv:2302.12173, 2023.

[6] Microsoft. Autogen use cases: Agent chat, 2025.