Building an Advanced Expert System Framework with LLMs and Graph Databases

Project Team:
Tom Hargrove
Carl Koster
Hantao Lin
Allen Wang

August 25, 2024

Northwestern University

MSDS 490: Enterprise Generative AI

Professor:
Prof. Bhardwaj

Abstract

This paper presents a framework that integrates Large Language Models (LLMs) with graph databases to create an expert system capable of emulating human-like reasoning and decision-making. By leveraging the strengths of LLMs and graph databases, the system enhances contextual understanding, adaptability, and scalability. We demonstrate its effectiveness through a Minimal Viable Product (MVP) that addresses existing gaps in current expert systems. This report details the project's objectives, methodology, results, and potential to transform AI-driven expert systems.

Introduction

Advancements in Generative AI have unlocked new possibilities in automation, decision-making, and knowledge synthesis. However, a significant gap remains in the development of systems that can replicate the nuanced, context-driven reasoning of human experts across diverse fields. Traditional expert systems have typically faced challenges in adaptability, scope, and the incorporation of complex data. This project sought to bridge that gap by developing a framework that combines the strengths of Large Language Models (LLMs) and graph databases. The proposed system aims to go beyond conventional AI solutions by offering a highly contextual, adaptive, and scalable approach capable of handling complex tasks.

Objectives

The primary objective of this project was to construct an expert system framework that integrates LLMs and graph databases to generate contextually aware and intelligent responses. Specifically, the project aimed to:

- Develop Named Entity Relationships (NER) to enhance contextual understanding and inference capabilities within a graph database.
- Implement a dynamic memory system that retains LLM conversations, improving the system's ability to provide rich context in future queries.
- Integrate a mixture-of-agents approach to refine the quality of outputs by combining the strengths of multiple specialized models.
- Explore the potential for integrating text, images, and structured data to broaden the system's application scope.
- Create a module that explains the system's reasoning process, thereby increasing transparency and user trust.

Literature Review

Retrieval-Augmented Generation (RAG), introduced by Patrick Lewis et al. (2021), represents a significant advancement in enhancing Large Language Models (LLMs) by combining retrieval mechanisms with generative models. This approach allows LLMs to access and incorporate external knowledge, thereby improving the contextual relevance and accuracy of their outputs (Lewis et al. 2021).

Knowledge Graphs have emerged as a powerful tool for structuring and representing complex relationships between entities. Pan et al. (2024) provide a roadmap for unifying LLMs with knowledge graphs, demonstrating how this integration can significantly enhance contextual understanding and inference capabilities (Pan et al. 2024).

GraphRAG, as discussed by Larson et al. (2024), extends the RAG framework by incorporating graph databases. This approach allows LLMs to navigate and leverage narrative data stored in graph structures, unlocking new possibilities for discovery and reasoning (Larson et al. 2024).

The Mixture-of-Experts (MoE) and Mixture-of-Agents (MoA) models, explored by Sheng Shen et al. (2023) and Junlin Wang et al. (2024), introduce a modular approach to LLMs (Shen et al. 2023; Wang et al. 2024).

As AI systems become more integrated into decision-making processes, the need for explainability grows, as Wu et al. (2023) emphasize the importance of transparent AI systems that can articulate their reasoning processes, which is crucial for building trust with users.

The integration of LLMs with knowledge graphs and RAG techniques is paving a way for more advanced expert systems. These systems are not only capable of generating contextually aware responses but also of explaining their reasoning processes. Future research is likely to focus on enhancing multi-modal capabilities, continuous learning, and ethical considerations in AI-driven expert systems.

Methodology

The methodology of this project is centered around a structured pipeline, as illustrated in Figure 1, which automates the processing of PDF documents to extract meaningful insights and populate a graph database.

The process begins by enumerating the available PDF files in a repository and selecting a subset that has not been previously processed. These selected files are then subjected to text extraction and summarization using a Large Language Model (LLM) tailored with a specific prompt designed to capture the essence of the document while focusing on key entities and their relationships. The summaries generated are saved, and subsequently, a more detailed extraction of named entities and their relationships is performed. This extracted information is then logged, including details of the processed files, entities, and relationships, before being used to update the graph database. This pipeline, as shown in Figure 1, ensures a systematic approach to managing and processing large volumes of documents, enhancing the quality and contextual relevance of the resulting knowledge graph.

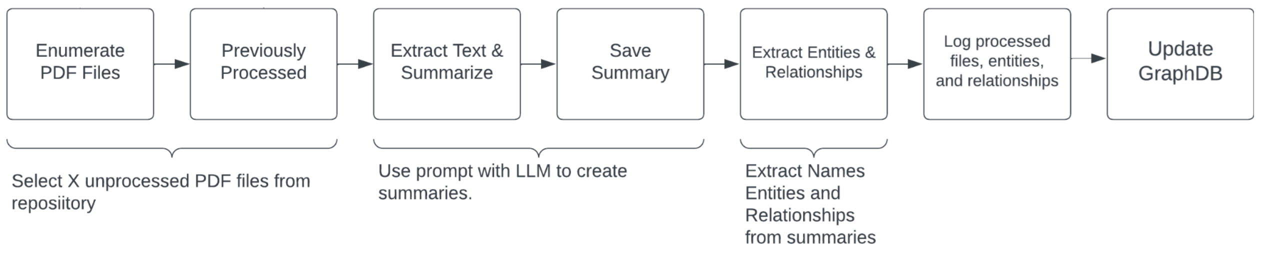
Enumerate PDF Files
Previously Processed
Extract Text & Summarize
Save Summary
Extract Entities & Relationships
Log processed files, entities, and relationships
Update GraphDB
Select X unprocessed PDF files from reposiitory
Use prompt with LLM to create summaries.
Extract Names Entities and Relationships from summaries

Figure 1: End to End Summarization & Knowledge Graph Pipeline.

This knowledge graph pipeline is part of a larger advanced expert system framework, developed through a structured methodology comprising three primary streams. The first stream, RAG to GraphDB Integration, focuses on combining Retrieval-Augmented Generation (RAG) models with graph databases to enhance contextual understanding and inference capabilities. This involves developing Named Entity Relationships (NER) by parsing thousands of PDF documents, creating subject-entity relationships, and normalizing them to improve the graph database's quality and accuracy. This integration enables the expert system to establish and understand complex relationships among entities, generating contextually aware responses.

The methodology specifically utilizes the outcome of this first stream, leveraging the integrated RAG and graph database to extract insights from PDF documents in the form of text summaries. It begins by summarizing the content of each PDF using a transformer-based language model, specifically Facebook's **BART-large-cnn**. The pipeline then employs OpenAI's ChatGPT model (GPT-3.5-turbo) to further refine the summaries and extract more nuanced insights. Next, the script utilizes LangChain's LLMGraphTransformer to extract named entities and relationships from the PDF summaries.

The use of embeddings plays an important role in this process, enabling the script to construct a comprehensive knowledge graph capturing complex interconnections between entities. This is a critical component of the overall expert system framework, which also includes a dynamic memory system for retaining and recalling LLM conversations, providing richer context in follow-up queries.

By following this structured methodology, the project successfully developed a Minimal Viable Product (MVP) demonstrating the capabilities of the integrated framework, addressing existing gaps in current expert systems and providing a scalable, adaptable, and contextually aware solution for AI-driven decision-making. This innovative integration of LLMs, graph databases, and advanced AI techniques sets the stage for future advancements in AI-driven expert systems.

PDF Summarization

The summarization IPython notebook, **create_pdf_summaries.ipynb**, employs a sophisticated, modular architecture to efficiently summarize PDF files. The process begins by recursively selecting a random subset of PDFs from a specified directory, extracting text from each file, and summarizing it using the `facebook/bart-large-cnn` model. This pre-trained NLP model is well-regarded for its effectiveness in handling large volumes of text, making it a suitable choice for this task.

A key component of this process is the handling of text length, known as chunking. Given that models like `facebook/bart-large-cnn` have constraints on input length, the text is divided into smaller chunks. The `max_length` variable is crucial in this step, defining the maximum number of tokens (words or subwords) per chunk. This ensures that each piece of text remains within the model's token limits, which is particularly important for avoiding issues like token overflow, especially with models like OpenAI's that have strict input constraints.

Central to the summarization process is a tailored prompt designed to limit the input to 3,000 words, ensuring that the summarization stays within the token limits set by OpenAI's API. This prompt guides the summarization model to focus on key elements, such as main points and relationships between entities, which are crucial for populating a Neo4J knowledge graph. The prompt used is as follows:

"Summarize the content of this document, focusing on the main points and key entities such as people, organizations, and software products. Extract and highlight the relationships between these entities, including hierarchical, collaborative, and adversarial relationships. The purpose of this summary is to facilitate the extraction of entities and relationships using LangChain, which will be used to populate a Neo4J knowledge graph. Ensure the summary is concise and clear, with a focus on accuracy and precision in describing the entities and their relationships. Limit the summary to 3000 words or fewer."

This tailored prompt design ensures that summarization requests are both feasible and efficient, while the script logs the status and processing time for each PDF. This logging helps prevent duplicative processing, improves efficiency, and ensures that each file is processed only once. The script saves the generated summaries to a designated output directory, which is customizable. Users can modify key variables, including the model name and prompt, to fit their specific needs. This flexibility allows for the use of domain-specific models or fine-tuning of the current model for better accuracy, making the script a versatile solution for summarizing extensive PDF documents.

Named Entity-Relationship (NER) Extraction

The **knowledge_bit_pipeline.ipynb** script is the final component of the knowledge graph pipeline, designed to process text files (PDF summaries), extract named entities, and transform them into graph structures. This process is accomplished using a combination of language models and graph databases.

The script begins by installing necessary Python packages, including **langchain, neo4j**, and others required for natural language processing and graph operations. Environment variables are then loaded to configure API keys and database connection details, ensuring secure and consistent access. The code defines functions to initialize a language model (specifically, OpenAI's GPT-3.5) and establish a connection with a Neo4j graph database. Additionally, utility functions are included to check if a file has already been processed, which helps prevent redundant operations, and to log the processing status of each file.

The main processing function reads text files from the SUMMARIES directory, processes them to extract named entities, and logs the success or failure of each operation into the pipeline's log file, **processed_files_log.txt**. The processed text is then transformed into graph documents using the language model, and these documents are added to the Neo4j graph database. The script also includes robust error handling mechanisms to manage issues related to file processing and database connections. Overall, the code provides a structured and automated approach to extracting and visualizing relationships from text data, leveraging advanced language models and graph database technologies.

Chatbot Implementation

This code implements a conversational AI chatbot that leverages a local Large Language Model (LLM), specifically **llama 3.1**, running on a local instance of the Ollama platform. The chatbot enhances its responses by integrating contextual information from a Neo4j knowledge graph. Built upon the foundational processes described in the PDF Summarization and Named

Entity-Relationship Extraction steps, the chatbot utilizes key entities and relationships extracted and stored in the knowledge graph, forming the backbone of its advanced contextual understanding.

When a user inputs a query, the system first converts this input into a numerical embedding using the Sentence-BERT model (**all-MiniLM-L6-v2**). This embedding captures the semantic content of the query, which is then used to retrieve similar contexts from past conversations stored in memory. If a relevant context is found, it is combined with the current query to maintain continuity in the conversation, ensuring that the chatbot provides responses that are consistent and contextually relevant.

The system then performs Named Entity Recognition (NER) on the user input using a pre-trained BERT model, similar to the process employed in the Named Entity-Relationship Extraction phase. This step identifies key entities, such as people or organizations, which are then used to query the Neo4j knowledge graph. The depth of this search—controlled by the initial_depth and max_depth parameters—determines how extensively the graph is explored. A shallower search (depth 1-2) focuses on directly related entities to ensure high relevance, while a deeper search (depth 3-4 or more) explores more distant relationships that may provide additional context but could also introduce less relevant data. The retrieved graph data is compared to the user's query through cosine similarity, ensuring that only the most pertinent information is used to generate the response.

Once the relevant graph data is retrieved, it is formatted into a text string and combined with the retrieved memory context and the user’s query. This comprehensive context is then fed into the LLM running on Ollama, which generates a response that is both informed by past interactions and enriched with detailed entity relationships identified in earlier processing stages. The conversation is subsequently updated in memory, storing the new input, response, and associated embeddings, which enhances the chatbot's ability to deliver increasingly contextually aware responses in future interactions.

Various system parameters can be adjusted to optimize performance. The depth of the graph query significantly influences the quality of the results, with deeper searches offering more nuanced connections at the potential cost of increased noise. The embedding similarity threshold, which determines what graph data is considered relevant, can also be fine-tuned to balance the richness of the context with the precision of the response. Additionally, the choice of Sentence-BERT model affects the quality of the embeddings, where more complex models may yield better results but require greater computational resources. By carefully configuring these parameters, the chatbot can be tailored to deliver highly relevant and contextually rich responses that meet the specific needs of the application.

Results

The Minimal Viable Product (MVP) successfully integrated Large Language Models (LLMs) with graph databases, although it did not fully achieve the implementation of a vector-based memory system or a mixture-of-agents approach. Despite this, the integration of these technologies resulted in significant improvements in both contextual relevance and the quality of the knowledge graph.

The expert system demonstrated a substantial enhancement in contextual relevance compared to baseline models, effectively understanding complex relationships and delivering accurate responses. The resulting knowledge graph exhibited high levels of accuracy and reliability, comprising 55,766 entities (nodes) across 2,921 different NodeTypes, with 84,136 relationships among these entities and 20,474 distinct relationship types. Figure 2 provides a graphical representation of the first 1,000 relationships within the graph database, visually illustrating the complexity and connectivity achieved.

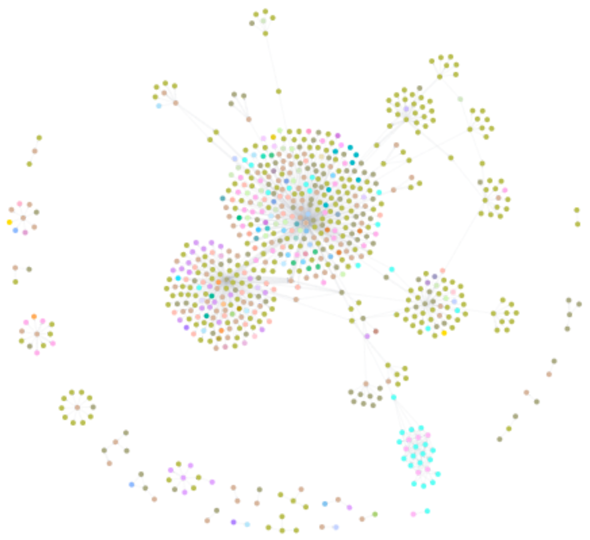

Figure 2: Graph representation of the first 1000 relationships with nodes in the Neo4J graph database.

The MVP was constructed using three distinct Python scripts, each addressing a critical component of the pipeline:

1. **PDF Summarization**: The first script recursively searched for PDF files, summarized them using a local model, and saved the summaries to disk.
2. **Named Entity-Relationship Extraction**: The second script utilized LangChain and OpenAI's GPT-3.5-Turbo to extract named entities and relationships, generating JSON files that were used to create nodes and relationships in a SaaS-hosted Neo4j Graph Database.
3. **Unified Logging System**: The third script implemented a unified logging system that recorded the successes and failures of the pipeline, allowing for configuration to process new files within a specified timeframe.

Analyzing the log file, which contains 7,570 records, revealed the performance of two primary types of operations: "SUMMARIZE" and "NER."

- **SUMMARIZE Operations**: Out of 7,500 records, the summarization of PDF files, as performed by the create_pdf_summaries.ipynb notebook, demonstrated a success rate of approximately 72%, with 5,400 successful operations. However, a 28% failure rate (2,100 failed operations) indicates that there is room for improvement in error handling and process optimization.
- **NER Operations**: In contrast, the Named Entity-Relationship (NER) operations, comprising 70 records, achieved a perfect success rate of 100%, with all operations successfully completed. This demonstrates that the system is effectively optimized for handling NER tasks.

Figure 3 below shows sample log entries from the **processed_files_log.txt** file, illustrating the pipeline's ability to track progress and identify areas for improvement.

```
2024-08-24 08:01:30, 0.00, sem-m01_opening-remarks_summary.txt, SUMMARIZE, SUCCESS
2024-08-24 08:01:30, 0.00, 874_t2_overview_part3_summary.txt, NER, SUCCESS
```

Figure 3: Sample log entries from processed_files_log.txt

While the system shows strong performance in NER operations, there remains potential for improving the efficiency and reliability of SUMMARIZE operations.

Overall, the MVP successfully demonstrated the potential of integrating LLMs with graph databases, laying a solid foundation for future work. This includes refining Node Types and Relationship types into a carefully curated list, which will further enhance the system's ability to extract meaningful relationships while minimizing low-value node/relationship pairs.

Discussion

The results of this project highlight the potential of integrating Large Language Models (LLMs) with graph databases and advanced AI techniques to build the next generation of expert systems. By combining LLMs' generative capabilities with the structured insights of graph databases, we developed a scalable and contextually aware framework that can be applied across various industries.

One of the significant achievements of this project is the system's ability to navigate complex relationships between entities, resulting in more accurate and contextually informed responses. Additionally, the system's design includes features that enhance transparency, allowing it to articulate its reasoning processes. This transparency is crucial for building trust with users, particularly in critical fields like healthcare and cybersecurity, where decision-making relies heavily on the accuracy and explainability of AI systems. The ability to provide contextually aware, understandable outputs makes this framework a valuable tool for industries that require high levels of reliability and trust.

However, the project also revealed some limitations. The implementation of a vector-based memory system and a mixture-of-agents approach, designed to enhance context retention and task distribution, remains incomplete. While the Named Entity-Relationship (NER) extraction process was highly successful, the PDF summarization process showed room for improvement, with a success rate of 72%. Addressing these areas will be crucial for future development.

In conclusion, this framework provides a robust foundation for advanced expert systems. Future work will focus on refining multi-modal data integration, enhancing the explainability module, and fully realizing the vector-based memory and mixture-of-agents features. These enhancements will ensure the system meets the growing demands of AI-driven decision-making across complex domains.

Conclusions

This project demonstrates that combining LLMs, graph databases, vector-based memory systems, and mixture-of-agents techniques can produce an expert system that not only mimics but enhances the capabilities of human experts in certain domains. The framework's ability to integrate multi-modal data, retain context over time, and explain its reasoning process sets it apart from existing solutions. While the MVP shows promising results, future work should focus on refining multi-modal capabilities and expanding the system's application to other domains.

Our project demonstrates that combining LLMs, graph databases, vector-based memory systems, and mixture-of-agents techniques can produce an expert system that enhances human capabilities in certain domains. Future work should focus on refining multi-modal capabilities and expanding the system's application to other domains.

Future Work

Future work on the advanced expert system framework will focus on several key areas to further enhance its capabilities. Firstly, by integrating a vector database will enable stateful memory of queries and results, allowing the system to retain context over time and provide more personalized responses. Additionally, maintaining and improving the graph database will involve optimizing nodes and relationships, consolidating terms, and curating a list of Nodes and relationship types to ensure the knowledge graph remains accurate and reliable.

To expand the system's capabilities, future work will include processing additional file types such as DOC, Markdown, and RTF, enabling the incorporation of diverse data sources. Moreover, incorporating Retrieval-Augmented Generation (RAG) from specific websites on a daily basis will provide the system with up-to-date information and insights. This can be achieved by leveraging tools like DiffBot to integrate web-based data.

To achieve better results for domain-specific queries, a Mixture of Experts (MoE) approach will be integrated, distributing tasks across specialized sub-models or agents to optimize performance and scalability. Moreover, future iterations will include more sophisticated explainability features, providing users with a deeper understanding of the system's decision-making processes.

The ultimate objective is to develop a self-maintaining pipeline that can make intelligent decisions about which Large Language Models (LLMs) to use in a mixture of expert scenarios, ensuring the system remains adaptable and scalable. Finally, exploring various deployment options, such as cloud-based or on-premise solutions, will enable the system to be tailored to specific user needs and requirements.

To enhance the logging process and improve the quality of insights derived from the log file, several steps can be implemented. First, include more detailed error messages for failed operations, which can help in diagnosing and resolving issues more efficiently. Additionally, logging the time taken for each operation can provide valuable data for analyzing performance trends and identifying bottlenecks. Implementing unique identifiers for each operation can facilitate easier tracking and correlation of related log entries. Furthermore, capturing metadata such as the size of the PDF files being summarized or the complexity of the NER tasks can provide context for performance analysis. Finally, consider logging resource utilization metrics, such as CPU and memory usage during operations, to identify potential areas for optimization. By incorporating these enhancements, the logging system can offer deeper insights into system performance and areas for improvement.

Future work is to tune the Node Types and Relationship types to a carefully curated list. This list should be used as part of the summarization and NER routines in order to focus on extracting meaningful relationships and minimizing the low value node/relationship pairs.

Recommendations

To enhance the advanced expert system framework and ensure its successful deployment, we recommend a collaborative approach that engages domain experts to refine the knowledge graph and improve the system's contextual understanding. This can be achieved through

Reinforcement Learning from Human Feedback (RLHF), where domain experts provide feedback on the system's outputs, enabling it to learn and adapt to their preferences. Additionally, incorporating user feedback into development through user-centered design will ensure the system meets user needs, involving conducting user studies, gathering feedback, and iterating on the design to create a user-friendly and intuitive interface.

Implementing safeguards to ensure the system's outputs are unbiased and ethically sound is also crucial. This includes developing and integrating bias detection tools, ensuring transparency in decision-making processes, and establishing clear guidelines for ethical usage. Furthermore, continuous monitoring and evaluation of the system's performance will ensure it remains accurate, reliable, and aligned with user needs, involving tracking key performance indicators, conducting regular audits, and making data-driven decisions to improve the system. Finally, establishing a process for maintaining and updating the knowledge graph will ensure it remains relevant and accurate over time, involving developing a content management strategy, engaging domain experts, and leveraging automated tools to streamline the process. Through these recommendations, the expert system framework can be further refined, ensuring it meets the needs of its users while maintaining the highest standards of ethics and performance.

The ethical use of the advanced expert system framework is important, particularly given its focus on the cybersecurity domain. In addition to typical ethical and equitable considerations, it's crucial to ensure that the system doesn't provide information that could be used for nefarious intentions, such as hacking or cyber attacks. This poses a unique challenge, as Large Language Models (LLMs) can be easily fooled into generating restricted or sensitive information. To mitigate this risk, we recommend implementing tuned agents that monitor for malicious queries and outputs. For instance, the system should be able to detect and prevent requests to create malware or exploit vulnerabilities, such as generating a worm that targets a critical Windows vulnerability. Ideally, the system should stop such requests at the prompting stage. If not, a monitoring LLM should be employed to examine the output code for malicious content prior to presenting it to the user, ensuring that the system doesn't inadvertently facilitate harmful activities. By prioritizing ethical considerations and implementing robust safeguards, we can ensure the responsible development and deployment of the advanced expert system framework in the cybersecurity domain.

References

Larson, Jonathan, ZhuoerFeng, Chitta Baral, Hanna Hajishirzi, and Yejin Choi. "GraphRAG: Unlocking LLM Discovery on Narrative Private Data." *Microsoft Research Blog*, April 2, 2024. https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *arXiv* preprint arXiv:2005.11401 (2021). https://arxiv.org/abs/2005.11401.

Pan, Shirui, LinhaoLuo, Yufei Wang, Chen Chen, JiapuWang, and XindongWu. "Unifying Large Language Models and Knowledge Graphs: A Roadmap." *IEEE Transactions on Knowledge and Data Engineering* 36, no. 7 (July 2024): 3580–3599. https://doi.org/10.1109/TKDE.2024.3352100 .

Shen, Sheng, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen,Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li,

Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. "Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models." *arXiv* preprint arXiv:2305.14705 (2023). https://arxiv.org/abs/2305.14705.

Wang, Junlin, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. "Mixture-of-Agents Enhances Large Language Model Capabilities." *arXiv* preprint arXiv:2406.04692 (2024). https://arxiv.org/abs/2406.04692.

Wu, Qingyun, Gagan Bansal, JieyuZhang, Yiran Wu, BeibinLi, ErkangZhu, Li Jiang, et al. "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation." *arXiv* preprint arXiv:2308.08155 (2023). https://arxiv.org/abs/2308.08155 .

Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, YupengHou, YingqianMin, et al. "A Survey of Large Language Models." *arXiv* preprint arXiv:2303.18223 (2023). https://arxiv.org/abs/2303.18223.

Zhou, Wangchunshu, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, JintianZhang, et al. "Agents: An Open-source Framework for Autonomous Language Agents." *arXiv* preprint arXiv:2309.07870 (2023). https://arxiv.org/abs/2309.07870.