A COLLABORATIVE MULTI-AGENT LLM APPROACH FOR KNOWLEDGE GRAPH CURATION AND QUERY FROM MULTI-MODAL DATA SOURCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-Augmented Generation (RAG) systems have demonstrated considerable effectiveness in querying private, short, unstructured data; however, they often encounter challenges in delivering accurate factual answers when working with larger corpora, frequently lacking context and failing to establish domain relationships. In this paper, we introduce a novel collaborative multi agent Retrieval-Augmented Generation (CoMaKG-RAG) framework designed to enhance the capabilities of large language models (LLMs) in complex information retrieval scenarios involving multimodal data sources.Our framework comprises a pool of customized collaborative agents, including a query generator agent, a domain model generator agent, a domain model populator agent, a knowledge graph curator agent, and a knowledge graph query agent, each tailored through a developed customization model and historical domain questions. The query generator formulates relevant queries related to text and image chunks within documents, while the domain model generator constructs a structured domain model based on these queries. The domain model populator agent enriches the model by integrating additional text and image fragments, and the knowledge graph generator assembles a comprehensive unified knowledge graph using Neo4j.Each agent interacts with one another, evaluates outputs, and provides feedback to enhance the overall process. Ultimately, user queries are transformed into cipher queries using the knowledge graph query agent, processed by a unified knowledge graph engine, and converted back into natural language responses. This approach enhances information retrieval from multimodal sources by mitigating hallucinations, generic responses, incomplete responses, and factual inaccuracies. We evaluated our method against the publicly available technical report ”Operations Maintenance Best Practices” and state-of-the-art knowledge graph generation and query software, Neo4j Graph Builder. Our results demonstrate that our method identifies a substantially higher number of entities and uncovers unique, contextually significant relationships, surpassing the performance of the graph builder in both the quantity and quality of extracted information. The proposed agentic graph RAG system was evaluated on both factual and descriptive queries and was able to provide accurate responses for both text and image-based questions, whereas the Neo4j graph performed sub optimally.

1 INTRODUCTION

The proliferation of digital documents, particularly in PDF format, has created significant challenges in information retrieval and knowledge management across various industries. While Large Language Models (LLMs) have shown remarkable capabilities in natural language processing tasks, their application to complex information extraction from unstructured documents remains an area of active research.

Traditional Retrieval-Augmented Generation (RAG) approaches have made significant strides in enhancing document querying by combining the strengths of LLMs with information retrieval techniques. However, these methods often struggle with maintaining context over long documents, handling complex multi-hop queries, and providing transparent reasoning paths. Moreover, the flat structure of typical RAG systems limits their ability to capture and utilize the hierarchical nature of information present in many documents.

Knowledge Graph (KG) approaches offer a superior alternative for document querying and information retrieval. By representing information as interconnected entities and relationships, KGs can capture the semantic structure of documents more effectively. This hierarchical representation enables more nuanced and context-aware querying, supports multi-hop reasoning, and provides clear provenance for extracted information. Furthermore, KGs allow for the integration of domain-specific knowledge and ontologies, enhancing the overall quality and relevance of query responses.

However, the generation of high-quality knowledge graphs from unstructured text remains a significant challenge. Existing approaches Sukumar et al. (2023); Schatz et al. (2023) for KG creation often produce graphs with superficial or less meaningful relationships between nodes, resulting in limited utility for complex querying and knowledge discovery. The main difficulties lie in accurately identifying relevant entities, establishing meaningful relationships, and capturing the hierarchical structure of information present in the source documents. To address these challenges, this paper presents a novel multi-agent framework that integrates LLMs with RAG techniques to enhance knowledge graph curation and document querying. Our approach aims to bridge the gap between traditional RAG systems and KG-based information retrieval by introducing a collaborative multi-agent system that not only automates and enhances the knowledge extraction and query process but also generates more semantically rich and hierarchically connected knowledge graphs.

The proposed framework consists pool of five specialized agents working in concert: a query generator agent, a domain model generator agent, a domain model populator agent, a knowledge graph generator agent,and a knowledge graph query agent

The key innovation in our approach lies in the introduction of intermediate layers, particularly the question generation and domain model creation steps. These crucial components enable the generation of more meaningful and relevant relationships between nodes, resulting in a hierarchically connected graph rather than a collection of isolated nodes. This structure significantly enhances the graph's utility for complex querying and knowledge discovery.

By integrating LLMs with RAG techniques and advanced knowledge graph technologies, our approach offers a promising solution for enhancing information retrieval processes in data-intensive environments. This paper details the architecture of our multi-agent system, its implementation using Neo4j for knowledge graph storage, and presents experimental results demonstrating its effectiveness in reducing manual reading time, improving query response accuracy, and generating more semantically meaningful knowledge representations.

The rest of this paper is organized as follows: Section (II) provides a background on LLMs, RAG, and knowledge graphs, as well as a review of existing approaches to KG generation from text. Section (III) details our multi-agent framework and methodology, particularly emphasizing the domain model generation process. Section (IV) presents our experimental setup and results, including comparisons with traditional KG generation methods. Section (V) discusses the implications of our findings, potential applications, and the advantages of our hierarchical graph structure. Finally, Section (VI) concludes the paper and outlines directions for future research.

2 BACKGROUND AND RELATED WORK

2.1 LARGE LANGUAGE MODELS (LLMS)

Large Language Models are neural networks trained on vast amounts of text data to perform a wide range of natural language processing tasks. These models, such as GPT (Generative Pre-trained Transformer) series, BERT, and T5, have demonstrated remarkable capabilities in understanding and generating human-like text Asmitha et al. (2024). LLMs have shown proficiency in tasks including text summarization, question answering, and language translation Omrani et al. (2024). However, their application to complex information retrieval and knowledge structuring tasks remains an active area of research.

2.2 KNOWLEDGE GRAPHS

Knowledge Graphs Knowledge Graphs (KGs) are structured representations of information that capture entities and their relationships in a graph format Osman & Barukub (2020). KGs have become fundamental in various applications, including semantic search, question answering systems, and recommender systems. They provide a means to organize and query complex, interconnected information efficiently.

2.3 RETRIEVAL-AUGMENTED GENERATION (RAG)

Retrieval-augmented generation is a hybrid approach that combines the strengths of retrieval-based and generation-based models . In RAG systems, a retrieval component first accesses relevant information from a knowledge base, which is then used to augment the input to a generative language model. This approach enhances the model's ability to produce accurate and contextually relevant outputs by grounding its generations in retrieved factual information.

2.4 EXISTING APPROACHES FOR KG GENERATION

Several approaches have been proposed for automatically generating knowledge graphs from unstructured text:

1. Rule-Based Methods: These approaches use predefined patterns and rules to extract entities and relationships from text Khaing et al. (2019). While effective for specific domains, they often lack flexibility and require significant manual effort to create and maintain rules.
2. Supervised Learning Methods: These techniques use machine learning models trained on annotated datasets to identify entities and relations in text Zhao et al. (2024). They can be more adaptable than rule-based methods but require large amounts of labelled training data.
3. Unsupervised and Semi-Supervised Methods: These approaches attempt to extract knowledge graph elements with minimal or no labeled data, often using techniques like clustering or distant supervision Zhao et al. (2019). They can be more scalable but may suffer from lower precision.
4. Neural Network-Based Methods: Recent approaches leverage deep learning models, including LLMs, for KG construction Zhu et al. (2024). These methods have shown promise in capturing complex semantic relationships but often struggle with producing meaningfully structured and hierarchical knowledge representations.

2.5 MULTI-AGENT APPROACHES IN AI AND NLP

Recent advancements in AI have shown the potential of multi-agent systems for tackling complex tasks. These approaches distribute cognitive load across multiple customized agents, often leading to more robust and effective solutions.

1. CoMM Framework: Chen et al. introduced the Collaborative Multi-Agent, Multi-Reasoning-Path (CoMM) prompting framework Yang et al. (2024). This approach prompts Large Language Models (LLMs) to play different roles in a problem-solving team, encouraging collaborative problem-solving. CoMM applies different reasoning paths for different roles, effectively implementing few-shot prompting in multi-agent scenarios. This work demonstrated significant improvements in solving complex college-level science problems
2. AutoGen: Li et al. presented AutoGen, a framework for building multi-agent systems with LLMs Wu et al. (2023). This work highlights how different agents can be assigned specific roles and collaborate to solve complex tasks, providing a foundation for multi-agent systems in various domains.
3. ChatDev: Qian et al. proposed ChatDev, a collaborative software development framework using multiple LLM-based agents Qian et al. (2024). This system demonstrates how multi-agent approaches can be applied to complex creative tasks like software development.

4. Task-Oriented Dialogue Systems: Zhang et al. developed a multi-agent framework for task-oriented dialogue systems Sun et al. (2025). Their work illustrates how multiple specialized agents can work together to handle complex conversational tasks.

These multi-agent approaches offer valuable insights into designing collaborative AI systems. They demonstrate the potential for breaking down complex tasks into subtasks handled by specialized agents, a principle we adapt in our framework for knowledge graph generation. Our work builds upon these multi-agent concepts, particularly drawing inspiration from the CoMM framework, and applies them specifically to the challenge of generating semantically rich and hierarchically structured knowledge graphs from unstructured documents. By leveraging the strengths of multiple specialized agents, we aim to overcome the limitations of existing KG generation methods and produce more meaningful and useful knowledge representations.

Key features of our collaborative multi-agent system include:

- Adaptive learning capabilities, improving performance with exposure to diverse manual content
- Seamless integration between agents, with feedback loops for continuous improvement
- Customizable options for different domains, operational needs, and industry-specific requirements
- Strong focus on data quality, ensuring accuracy, completeness, and relevance of the generated knowledge graph

3 COLLABORATIVE MULTI-AGENT ARCHITECTURE AND METHODOLOGY

3.1 Overview of the Multi-Agent System

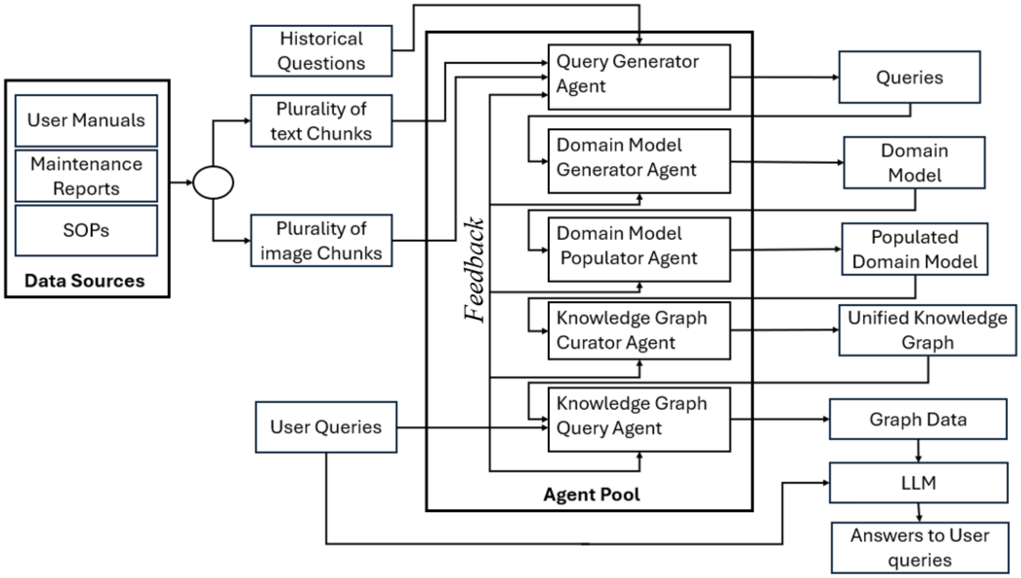

Figure 1: The multi-agent system Architecture

3.2 Agent Architecture

Each agent in our system is customized by a comprehensive JSON model that outlines its role, capabilities, inputs, outputs, and specialized tasks. This structured approach ensures clear delineation of responsibilities and facilitates seamless interaction between agents. The JSON model for each agent includes:

1. Name and Role: Clearly defines the agent's identity and primary function within the system
2. Domain Knowledge: Specifies the agent's area of expertise and specialization.
3. Interface: Describes how the agent interacts with other components of the system.
4. Input and Output: Defines the expected input format and the produced output structure.
5. Tasks: Lists both primary and secondary tasks the agent can perform.
6. Tools: Enumerates integrated and external tools the agent can utilize
7. Additional Attributes: Includes information on experience, learning capabilities, and error handling.
8. Collaboration Capabilities: Outlines how the agent interacts and cooperates with other agents in the system.

The multi-agent system architecture Fig 1 is designed to effectively process user queries by utilizing specialized agents that communicate and learn from one another. Each agent is customized using JSON models as stated above, enabling adaptability to various tasks and continuous improvement through inter-agent feedback. The system aims to transform user queries into structured questions that can be efficiently answered using agentic graph RAG.

3.3 THE MULTI AGENT ARCHITECTURE HAS FOLLOWING COMPONENTS

1. Data Sources
    - User Manuals, Maintenance Reports, SOPs: Essential documents providing foundational knowledge for the system.
    - Plurality of Text Chunks: Text data is segmented for easier retrieval and processing.
    - Plurality of Image Chunks: Visual data is also segmented, facilitating effective handling of image-related information.
2. User Queries
    - These are the natural language questions that users input into the system. The goal of the architecture is to process these queries and generate structured response in natural language.
3. Agent Pool
    - A central hub housing various specialized agents, each responsible for distinct functions. All agents are designed to communicate with one another and provide feedback, enhancing the system's performance.
4. Agents
    - Query Generator Agent:
        - The Question Generator is designed to formulate relevant questions based on input documents, thereby guiding the information extraction process. This agent utilizes historical domain-specific questions as input to generate contextual questions that enhance understanding and exploration of the subject matter.
        - Feedback Loop: Receives feedback from subsequent agents to refine its question generation process.
    - Domain Model Generator Agent:
        - Creates a structured domain model using the generated questions and document content.
        - Inter-agent Feedback: Adjusts its model based on insights from the query generator and other agents.
    - Domain Model Populator Agent:
        - Domain Model Populator: Enriches the initial domain model with detailed information extracted from the document.
        - Feedback Mechanism: Takes input from the domain model generator to improve data population strategies and other agents.
    - Knowledge Graph Curator Agent:

    - Knowledge Graph Generator: Transforms the populated domain model into a fully connected knowledge graph in Neo4j.
    - Feedback Integration: Continuously refines the graph based on queries executed by the knowledge graph query agent and other agents.
  - Knowledge Graph Query Agent:
    - Interfaces with the knowledge graph to interpret and answer user queries.
    - Learning from Feedback: Adapts its querying approach based on the results and feedback from other agents and other agents.

5. Output
   - The processed information is synthesized through a Language Model (LLM), which generates coherent answers to the user queries.
   - The entire system operates on a continuous feedback loop, where all agents learn from interactions and improve their performance over time.

3.4 DETAILED EXAMPLE: KNOWLEDGE GRAPH GENERATOR AGENT

To illustrate our agent architecture in depth, we focus on the Knowledge Graph Generator (Agent 4). Its JSON model includes the following key components:

1. Name and Role: ai_agent04, serving as the Knowledge Graph Generator
2. Domain Knowledge: Specializes in creating fully connected Knowledge Graphs in Neo4j from populated domain models
3. Interface: Text-based, capable of structured data processing and Cipher query generation
4. Input: Populated domain model from the previous agent
5. Output: Fully connected Knowledge Graph in Neo4j, along with Cypher queries and performance metrics
6. Tasks:
   - Primary- Include converting domain models to graph structures, generating Cypher queries, and ensuring full connectivity
   - Secondary- Evaluate populated domain model for completeness and accuracy. Provide feedback in case of any discrepancies.
7. Tools: Integrates Neo4j driver, Cypher query generator, and data cleaning tools
8. Additional Attributes: Features adaptive learning capabilities, collaboration features, and robust error handling

4 EXPERIMENTAL SETUP AND RESULTS

4.1 DATASET

For our experiments, we utilized Chapter 9.2 of the publicly available technical report ”Operations Maintenance Best Practices: A Guide to Achieving Operational Efficiency” published by the Pacific Northwest National Laboratory Laboratory (2022). This chapter specifically focuses on boilers, their types, components, maintenance, and efficiency.

4.2 EXPERIMENTAL PROCESS DOCUMENT PREPROCESSING

The content of Chapter 9.2 PDF was extracted to a text and image file, preserving the document’s structure.

4.3 LARGE LANGUAGE MODEL (LLM)

In this experiment, GPT-4o is utilized to build a customized LLM agent tailored for specific tasks.

4.4 MULTI-AGENT SYSTEM ARCHITECTURE

Our framework employs a team of specialized agents, each designed to perform specific tasks in the knowledge graph generation process. The agents include:

1. Question Generator (ai_agent01): This agent simulates a field engineer or operator, generating relevant questions about the domain. For our boiler maintenance example, key capabilities include:
    (a) Formulating questions about maintenance procedures for various boiler types
    (b) Generating queries related to troubleshooting common boiler issues
    (c) Creating questions about inspection processes, including Non-Destructive Examination (NDE) methods
    (d) Asking about safety protocols and compliance guidelines

    It's important to note that this agent is highly customizable and can be adapted to generate questions for any industry or domain, not just boiler maintenance.
2. Comprehensive Domain Model Generator (ai_agent02): This agent creates detailed and hierarchical domain models from textual input. Key capabilities include
    (a) Detailed entity extraction
    (b) Multi-level relationship identification
    (c) Hierarchical structure creation
    (d) Attribute detailing
    (e) Ontology integration and cross-domain connection identification

    This agent is essential in our framework, creating a detailed domain model with multi-level relationships and attributes that enable the construction of a precise and semantically rich knowledge graph.
3. Domain Model Populator Agent (ai_agent03): This agent populates the domain model with content from the extracted manual text. Key capabilities include:
    (a) Content extraction and mapping to the domain model
    (b) Relationship population
    (c) Contextual analysis and Implicit Information Inference
    (d) Consistency validation
4. Knowledge Graph Generator (ai_agent04): This agent creates fully connected, hierarchical Knowledge Graphs in Neo4j from populated domain models. Key capabilities include:
    (a) Converting domain models to graph structures
    (b) Generating Cypher queries for graph creation
    (c) Ensuring full connectivity and hierarchy
    (d) Performing data cleaning and normalization
    (e) Optimizing graph structure and implementing advanced graph algorithms

The agents collaborate leveraging natural language processing and structured data mapping capabilities. They interact with integrated tools such as text analysis algorithms, graph modeling tools, and data cleaning systems, as well as external APIs for industry standards and graph visualization.

This multi-agent approach allows for specialized knowledge to be applied at each stage of the knowledge graph generation process, from initial question generation to the creation of complex, hierarchical relationships between concepts. The system is designed to be scalable, handling large volumes of text input and complex domain structures efficiently.

Importantly, while our experiment focused on boiler maintenance, the multi-agent collaboration system is designed to be highly flexible and can be leveraged for any industry or domain. By customizing the Question Generator agent and adjusting the domain knowledge of other agents, this system can be applied to diverse fields such as healthcare, finance, manufacturing, or any other area where structured knowledge extraction from technical documents is required.

This architecture enables us to generate comprehensive, hierarchical knowledge graphs that accurately represent complex domains, as described in source documents, while being adaptable to a wide range of industries and applications.

5 RESULTS AND ANALYSIS

5.1 QUESTION GENERATION BY AGENT 1

Total Number of Questions Generated by Agent 1: Agent 1 successfully generated a total of 100 questions from each page of document, designed to probe various aspects of boiler operation and efficiency. The total number of questions can be adjusted based on user requirements. Distribution of Question Types:

- Maintenance:
    - Example: ”What equipment is used to determine combustion efficiency in a boiler?” and ”How can scale formation in a boiler be prevented?”
- Safety:
    - Example: ”What are the potential consequences of having deficient air in a boiler?” and ”What are the safety implications of improper boiler maintenance?”
- Troubleshooting:
    - Example: ”What is a major problem associated with heat recovery in flue gas?” and ”How can the energy from blowdown be recovered in a boiler?”
- Other:
    - Example: ”What is cogeneration in the context of boiler operation?” and ”How can the vertical temperature in a boiler room indicate air stratification?”

5.2 DOMAIN MODEL GENERATION BY AGENT 2

The following JSON snippet demonstrates the initial hierarchical structure of the domain model for the ”Boilers” entity. At this stage, attributes such as boiler horsepower and maximum pressure are placeholders awaiting further population by Agent 3.

```
{
    "Boilers": {
        "description": "Fuel-burning appliances that produce either hot water or steam for heating or process uses.",
        "attributes": {},
        "sub-entities": {
            "Types of Boilers": {
                "description": "Classifications of boiler designs.",
                "attributes": {},
                "sub-entities": {
                    "Fire-Tube Boilers": {
                        "description": "Boilers where hot gases circulate through tubes submerged in water.",
                        "attributes": {
                            "Boiler Horsepower": "",
                            "Maximum Pressure": ""
                        },
                        "relationships": {
                            "Efficiency Institute": {
                                "type": "Reprinted with permission of",
                                "details": "The Boiler Efficiency Institute, Auburn, Alabama"},
```

5.3 DOMAIN MODEL POPULATION BY AGENT 3

The following JSON snippet demonstrates the populated data for ”Fire-Tube Boilers,” a subtype within the ”Boilers” entity.

```
{
  "Boilers": {
    "description": "Fuel-burning appliances that produce either
       hot water or steam for heating or process uses.",
    "attributes": {},
    "sub-entities": {
      "Types of Boilers": {
        "description": "Classifications of boiler designs.",
        "attributes": {},
        "sub-entities": {
          "Fire-Tube Boilers": {
            "description": "Boilers where hot gases circulate
               through tubes submerged in water.",
            "attributes": {
              "Boiler Horsepower": "20 through 800 bhp",
              "Maximum Pressure": "150 psi"
            },
            "relationships": {
              "Efficiency Institute": {
                "type": "Reprinted with permission of",
                "details": "The Boiler Efficiency Institute,
                   Auburn, Alabama"
              },
```

5.4 KNOWLEDGE GRAPH CURATION BY AGENT 4

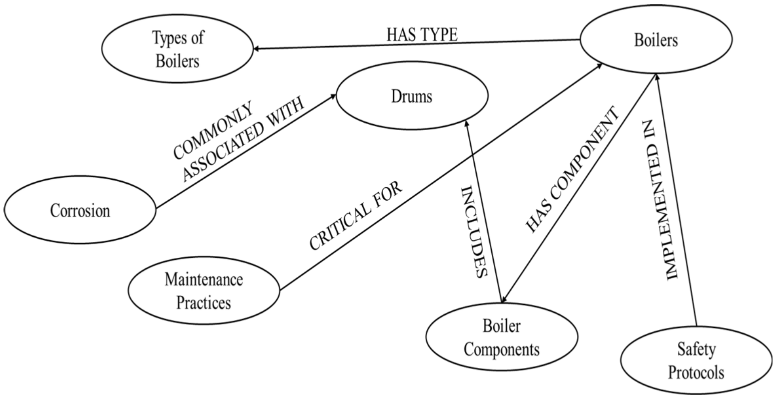

Figure 2: Sample of Graph created by Multi-Agent System

5.5 COMPARATIVE ANALYSIS

5.5.1 AGENTIC GRAPH RAG AND NEO4J GRAPH BUILDER RESPONSE ON USER QUERIES

As shown in the Table 1, five relevant questions were posed to both CoMaKG-RAG and Neo4j GraphRAG. The CoMaKG-RAG was able to respond accurately and precisely to all questions using data from graphs, text, and tables of a document, while Neo4j GraphRAG failed to answer four questions correctly.The response time is shown against each answer. This difference in performance can be attributed to CoMaKG-RAG’s ability to generate a contextually rich knowledge graph from multimodal data sources, significantly enhancing the effectiveness of the graph RAG system

Table 1: Agentic graph RAG and Neo4j graph builder response comparison

Question	Human	Neo4j Grpah Builder	CoMaKGRAG	Source
List the reasons for incidents and number of instances of occurrence?	Maintenance Errors: 11 occurrences Design Flaws in Boiler System: 4 occurrences Incorrect Installation or Repair Practices: 3 occurrences Failures in Low Water or Limit Control Systems: 2 occurrences Failures of the Burner System: 2 occurrences	I'm sorry, but I don't have that information.(17.96 s)	Incidents attributed to errors made during maintenance or operation: 11 occurrences. Incidents resulting from design flaws in the boiler system: 4 occurrences. Incidents caused by incorrect installation or repair practices: 3 occurrences. Incidents related to failures in low water or limit control systems: 2 occurrences. Incidents involving failures of the burner system: 2 occurrences.(1 s)	Image -Graph
when visual inspection of boiler is needed?	Operators and maintenance technicians must conduct a visual inspection of a boiler, especially during start-ups and running operations.	A visual inspection of the boiler is recommended to be completed weekly to ensure that all equipment is operating correctly and that safety systems are in place.(10.33 s)	Visual inspections are conducted during start-ups and running operations. These regular checks are performed to ensure that the boiler is functioning properly.(1s)	Text
what was the capacity of boiler installed in public school of Canada	A 300-hp boiler installed at a public school in Canada	The boiler installed at a public school in Canada had a capacity of 300 horsepower (hp). (20.14 s)	The capacity of the boiler installed in the public school in Canada was 300 horsepower.(1 s)	Text
How frequently check the motor condition temperatures ?	Daily	The document does not specify an exact frequency for checking motor condition temperatures. (11.12 s)	You should check the motor condition temperatures daily.(1 s)	Table
How scale deposits cause damage to tube?	It cause damage to tube structure due to overheating and corrosion.	Scale deposits act as insulators on the water side of a boiler, reducing heat transfer efficiency. (15.52 s)	Scale deposits, which are mineral build-ups on the water side of a boiler, can cause damage to tubes by leading to overheating and corrosion.(1 s)	Text

6 CONCLUSION

In this paper, we introduced a collaborative novel multi-agent framework that integrates Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) CoMaKG-RAG techniques to enhance knowledge graph curation and document querying from multi model sources. By leveraging five specialized agents—question generation, domain model generation, domain model population, knowledge graph creation, and query generation—our approach addresses the limitations of traditional RAG systems and flat knowledge graph structures. The introduction of intermediate layers such as question generation and domain model creation enabled the generation of more semantically rich, hierarchically connected knowledge graphs, which significantly improved the quality of information retrieval and complex query responses. The experimental results demonstrated that our multi-agent system can generate more meaningful relationships between entities and reduce the time spent on manual document reading while improving the accuracy of query responses. By incorporating domain-specific knowledge and creating a more interconnected and context-aware knowledge graph, our approach provides a more powerful tool for knowledge management tasks in data-intensive environments. This work highlights the potential of LLM-driven multi-agent systems in transforming information retrieval processes and points towards further opportunities in refining knowledge graph generation, particularly in handling larger-scale documents and more intricate queries. Future research may focus on expanding this framework to accommodate diverse domains and improving the automation of knowledge graph curation to enhance real-world applications in various industries

REFERENCES

M. Asmitha, C. R. Kavitha, and D. Radha. Summarizing news: Unleashing the power of bart, gpt-2, t5, and pegasus models in text summarization. In *2023 4th International Conference on Intelligent Technologies (CONIT)*, pp. 1–6, Bangalore, India, 2024. doi: 10.1109/CONIT61985.2024.10626617.

P. Chen, B. Han, and S. Zhang. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv*, 2404.17729v1, 2024. URL `https://arxiv.org/abs/2404.17729`. [cs.CL], Apr. 26, 2024.

E. T. Khaing, M. M. Thein, and M. M. Lwin. Stock trend extraction using rule-based and syntactic feature-based relationships between named entities. In *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 78–83, Yangon, Myanmar, 2019. doi: 10.1109/AITC.2019.8920986.

Pacific Northwest National Laboratory. Operations maintenance best practices: A guide to achieving operational efficiency. `https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-19634.pdf`, 2022. Published on Monday, August 15, 2022.

P. Omrani, A. Hosseini, K. Hooshanfar, Z. Ebrahimian, R. Toosi, and M. Ali Akhaee. Hybrid retrieval-augmented generation approach for llms query response enhancement. In *2024 10th International Conference on Web Research (ICWR)*, pp. 22–26, Tehran, Iran, 2024. doi: 10.1109/ICWR61162.2024.10533345.

A. H. Osman and O. M. Barukub. Graph-based text representation and matching: A review of the state of the art and future challenges. *IEEE Access*, 8:87562–87583, 2020. doi: 10.1109/ACCESS.2020.2993191.

C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun. Chatdev: Communicative agents for software development. *arXiv*, 2307.07924, 2024. URL `https://arxiv.org/abs/2307.07924`.

K. Schatz, P.-Y. Hou, A. V. Gulyuk, Y. G. Yingling, and R. Chirkova. Build-kg: Integrating heterogeneous data into analytics-enabling knowledge graphs. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 2965–2974, Sorrento, Italy, 2023. doi: 10.1109/BigData59044.2023.10386570.

S. T. Sukumar, C.-H. Lung, and M. Zaman. Knowledge graph generation for unstructured data using data processing pipeline. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 466–471, Torino, Italy, 2023. doi: 10.1109/COMPSAC57700.2023.00068.

J. Sun, J. Kou, W. Hou, and Y. Bai. A multi-agent curiosity reward model for task-oriented dialogue systems. *Pattern Recognition*, 157:110884, 2025. doi: 10.1016/j.patcog.2024.110884.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL `https://arxiv.org/abs/2308.08155`.

L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3091–3110, July 2024. doi: 10.1109/TKDE.2024.3360454.

Y. Zhao, X. Jin, Y. Wang, and X. Cheng. Semi-supervised auto-encoder based event detection in constructing knowledge graph for social good. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 478–485, Thessaloniki, Greece, 2019.

Y. Zhao et al. Research on machine learning-based knowledge graph completion technique for power grid contingency planning. In *2024 36th Chinese Control and Decision Conference (CCDC)*, pp. 1263–1267, Xi'an, China, 2024. doi: 10.1109/CCDC62350.2024.10587969.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv*, 2305.13168v2, 2024. URL `https://arxiv.org/abs/2305.13168`. [cs.CL], Feb. 22, 2024.

Sukumar et al. (2023) Schatz et al. (2023) Asmitha et al. (2024) Omrani et al. (2024) Osman & Barukub (2020) Khaing et al. (2019) Zhao et al. (2024) Zhao et al. (2019) Zhu et al. (2024) Yang et al. (2024) Chen et al. (2024) Qian et al. (2024) Sun et al. (2025) Wu et al. (2023)