Domain-Specific Retrieval-Augmented Generation Using Vector Stores, Knowledge Graphs, and Tensor Factorization

Ryan C. Barron[†‡‖], Vesselin Grantcharov[§‖], Selma Wanna[*¶], Maksim E. Eren[*‡],
Manish Bhattarai[†], Nicholas Solovyev[†], George Tompkins[**],
Charles Nicholas[‡*], Kim Ø. Rasmussen[†], Cynthia Matuszek[‡*], and Boian S. Alexandrov[†]
[**]Analytics, Intelligence and Technology Division, Los Alamos National Laboratory, Los Alamos, New Mexico, USA.
[§]University of New Mexico. [¶]University of Texas at Austin. [‡]University of Maryland Baltimore County.
[*]Advanced Research in Cyber Systems, Los Alamos National Laboratory, New Mexico, USA.
[†]Theoretical Division, Los Alamos National Laboratory, New Mexico, USA.

Abstract—**Large Language Models (LLMs) are pre-trained on large-scale corpora and excel in numerous general natural language processing (NLP) tasks, such as question answering (QA). Despite their advanced language capabilities, when it comes to domain-specific and knowledge-intensive tasks, LLMs suffer from hallucinations, knowledge cut-offs, and lack of knowledge attributions. Additionally, fine tuning LLMs' intrinsic knowledge to highly specific domains is an expensive and time consuming process. The retrieval-augmented generation (RAG) process has recently emerged as a method capable of optimization of LLM responses, by referencing them to a predetermined ontology. It was shown that using a Knowledge Graph (KG) ontology for RAG improves the QA accuracy, by taking into account relevant sub-graphs that preserve the information in a structured manner. In this paper, we introduce SMART-SLIC, a highly domain-specific LLM framework, that integrates RAG with KG and a vector store (VS) that store factual domain specific information. Importantly, to avoid hallucinations in the KG, we build these highly domain-specific KGs and VSs without the use of LLMs, but via NLP, data mining, and nonnegative tensor factorization with automatic model selection. Pairing our RAG with a domain-specific: (i) KG (containing structured information), and (ii) VS (containing unstructured information) enables the development of domain-specific chat-bots that attribute the source of information, mitigate hallucinations, lessen the need for fine-tuning, and excel in highly domain-specific question answering tasks. We pair SMART-SLIC with chain-of-thought prompting agents. The framework is designed to be generalizable to adapt to any specific or specialized domain. In this paper, we demonstrate the question answering capabilities of our framework on a corpus of scientific publications on malware analysis and anomaly detection.**

Index Terms—**Artificial Intelligence, Retrieval Augmented Generation, Knowledge Graph, Natural Language Processing, Non-Negative Tensor Factorization, Topic Modeling, Agents**

I. Introduction

The expanding volumes of data across large databases and information collections necessitate the specialized extraction of pertinent knowledge, often without an in-depth understanding of the underlying database resources. Recent advancements in Large Language Models (LLMs) have facilitated developments that enable users to engage in dialogues with LLM-powered chat-bots to discover information. Despite these models' impressive handling of general queries, their application in domain-specific tasks is hindered by several limitations. These include the production of factually incorrect responses (”hallucinations”) [1], unawareness of recent developments or events beyond their training data (”knowledge cutoff”) [2], failure to accurately attribute sources of information (”implicit knowledge”) [3], and a lack of specific technical knowledge required for specialized fields [4].

Fine-tuning is a common strategy employed to tailor these general models to specific domains. However, this approach is resource-intensive, demanding significant amounts of data, extensive computational power, and considerable time, which makes it impractical for many domain-specific applications. These limitations pose significant challenges in interpreting and validating the knowledge generated by LLMs, as well as in referencing their sources. Consequently, this reduces the trustworthiness of LLMs and limits their effectiveness in highly specialized scientific contexts where accuracy and reliability are paramount. The ongoing challenges underscore the need for more sophisticated solutions that can bridge the gap between general-purpose LLMs and the nuanced requirements of domain-specific applications.

Retrieval-Augmented Generation (RAG) with Knowledge Graphs (KGs) and vector stores (VS) significantly enhances the context of LLMs, mitigating the need to fine-tune these models to specific domains [5], [6]. KGs provide a structured way to store factual information, making it easier to access and use, while VSs allow storing unstructured documents and preserving the semantics of the text. This integration allows LLMs to tap into both domain-specific and updated information, effectively addressing the traditional limitations of generative models.

Despite these improvements, challenges remain in the practical implementation of domain-specific RAG systems. Ex-

[‖] The first two authors contributed equally to this work.

tracting accurate and representative domain-specific ontologies to build KGs and VSs is a complex task. Additionally, curating datasets with specific text data for constructing both KGs and VSs is equally demanding. These steps are critical for ensuring that the augmented LLMs can reliably produce high-quality, relevant responses across different domains.

In this paper, we introduce a framework designed for constructing domain-specific corpora of scientific articles through advanced techniques, including: text mining, information retrieval, dimension reduction, nonnegative tensor factorization, citation graphs, and human-in-the-loop strategies. We introduce a novel framework, which we call **SMART-SLIC**, for developing KG's ontologies, utilizing both metadata and full texts from open-source scientific publications, as well the latent structures of these corpora, extracted through nonnegative tensor factorization, enhanced with automatic model determination. **SMART-SLIC** facilitates topic modeling [7], and determination of the optimal number of topics [8], [9] for effective document classification. Our new framework underpins the creation of a precisely tailored corpus of domain-specific scientific articles, which is crucial for our AG approach and supports the development of a chat-bot adept at answering domain-specific technical inquiries. Further, the framework is versatile, allowing for its application to any domain of documents. In this paper, we illustrate the effectiveness of our framework, **SMART-SLIC**, with a case study where we construct a domain-specific corpus, KG, and VS, focused on malware analysis and anomaly detection, and apply our enhanced question-answering framework for scientific queries related to this corpus. Our contributions are summarized as follows:

- We detail the development of a framework for building domain-specific scientific corpora using a blend of text mining, information retrieval, artificial intelligence (AI), and human-in-the-loop techniques.
- We describe the creation of a domain-specific KG & VS ontology that leverages both observable metadata, and full texts of the corpus of domain-specific open-source scientific articles, as well as its latent structure extracted by non-negative tensor/matrix factorization with automatic model selection.
- We demonstrate the enhanced capabilities of **SMART-SLIC**'s, RAG-enhanced LLM system, which utilizes chain-of-thought prompting with LLM agents to proficiently address scientific questions.

II. Related Works

Recent methods for building RAG-assisted [6] chatbot applications rely on unstructured text stored in vector databases for question answering (QA) tasks [10]. Although the integration of knowledge graphs (KGs) in AI systems is not novel [11], increasingly, researchers are leveraging them to improve LLM reasoning while simultaneously addressing the reliability issues discussed in Section I [12]–[14]. Despite the benefits, integrating domain-specific knowledge into chatbots requires substantial effort. Here, we review the prior work for common chatbot designs, the integration of domain-knowledge in RAG pipelines, and the steps required for constructing KGs.

A. KGs in RAG Pipelines

Building a sophisticated chatbot requires the knowledge of a wide range of research fields; hence, rarely do prior works present a fully engineered system like ours. Instead, most efforts focus on improving specific aspects of RAG pipelines, e.g., retriever design [15], [16], query intent recognition [17], and KG reasoning [18]–[21]. Our approach resembles past methods which leverage chain-of-thought [22] prompting on KGs [19], [20]; in conjunction with LLM-agents to enhance reasoning capabilities [23]–[25]. In addition to incorporating these state-of-the-art techniques, we improve our RAG pipeline by modifying our retrieval method to use K-Nearest Neighbors with the Levenshtein metric instead of cosine distance as an entry point for context search. We also construct a "highly-specific" knowledge base for targeted QA tasks.

Although expensive and time-consuming, a handful of prior works incorporate domain-knowledge into their RAG pipelines [26]–[29]; however, the majority either use existing KGs built broadly on medical literature [26], [28]; or do not disclose any details regarding their dataset construction [29]. We emphasize that our method is "highly-specific" because it was driven by subject matter expertise which informed our dataset curation and cleaning techniques [30], [31].

B. KG Development

At a minimum, the development of knowledge graphs requires building a corpus, defining an ontology, and extracting the relevant entity-relation triplets from unstructured text.

Corpus Building. Here we define the term "highly-specific" and explain our dataset collection method. A key feature of our dataset collection is the use of unsupervised methods [31] to decompose corpora into document clusters to finer specificity than the author-provided tags available on open access websites. This differs significantly from prior approaches [27], [32], [33]. We leverage latent-topic information from our NMFk method to filter and select the best data for our knowledge base, and prune documents based on citation information and embedding distances. Our text cleaning pipeline is informed by subject matter experts (SME) [31], [34], thus going beyond standard methods by incorporating expert-derived rules for document cleaning, e.g, acronym and entity standardization.

KG Construction. Our ontology is shaped by traditional methods, i.e., relying on SME design and capturing task-specific features. However, we innovate by incorporating latent information from our decomposition process [31] into our KG as entities. For entity and relation extraction, we move beyond conventional learning-based techniques [35]; and instead, leverage recent advancements which use LLM-agents [10], [36] as opposed to other LLM prompting methods [37]–[40]. This approach yields non-sparse KGs, meaning, the average out-degree of entities [41], [42] is high. To our knowledge, no prior work integrates all of these methods into their knowledge graph construction process.

III. Methods

This section outlines our framework, covering corpus extraction, KG ontology, VS construction, and the RAG process.

A. Domain-Specific Dataset

Overview of of our system is summarized in Figure 1. To collect the dataset, we began with a set of core documents selected by subject matter experts (SMEs). Here, these core documents represent the specific domain in which we want to built our corpus on. These core documents were used to build a citation and reference network, which allowed for the expansion of the dataset through the authorized APIs: SCOPUS [43], Semantic Scholar (S2) [44], and Office of Scientific and Technical Information (OSTI) [45]. We also extract common bigrams from the core documents to query these APIs to search for relevant documents. As we expand on the the corpus starting from the core documents, it is possible to add documents that do not directly relate to the information in the core documents. To maintain the central quality and thematic coherence of the core dataset, we employed several pruning strategies to remove these irrelevant documents to preserve the speciality specific to the targeted domain. These strategies focused on removing documents that diverge from the central theme of the core. Pruning was performed through two methods from [34]:

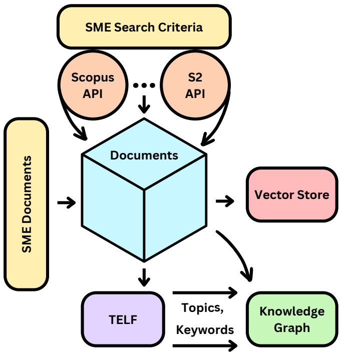

Fig. 1. User query routing overview.

- **Human-in-the-Loop Pruning**: SMEs manually review and select a handful documents that align with the core theme. Here, we reduce the document's TF-IDF matrix to two dimensions with UMAP and let the SME look at the documents that are at the centroids of the given clusters. SME can then select which documents to remove.
- **Automatic Pruning of Document Embeddings**: Based on the SME selections from the previous step, we next remove the document that are certain distance away from the selected and the core documents. Documents were transformed into embeddings with SCI-NCL [46], a BERT based model fine-tuned on scientific literature, to measure semantic similarity with core and SME selected documents. Those outside a set similarity threshold were removed, ensuring only the documents relevant to the core documents and SME selections remained.

Although a human is in the loop, the system remains scalable by clustering documents. One review per cluster allows the operator to decide on all documents in the group, making it efficient even with large datasets without limit on cluster size.

Additionally, we applied pre-processing techniques using a publicly available Python library, **T**ensor **E**xtraction of **L**atent **F**eatures (**T-ELF**)[1] [31]. The cleaning procedures involved the following pre-processing steps:

- Exclude non-English, copyrights, and non-essential elements: stop phrases, formulas, and email addresses.
- Remove formatting artifacts like next-line markers, parentheses, brackets, accents, and special characters.
- Filter out non-ASCII characters and boundaries, HTML tags, stop words, and standalone numbers.
- Eliminate extra whitespace and words ≤ 2 characters.
- Standardize punctuation variations, particularly hyphens.

These pre-processing cleaning and standardization efforts are essential for preparing the dataset for further analysis, thereby enhancing the quality and consistency of the data.

B. Dimension Reduction

The extraction of the latent structure from the dataset is accomplished through the following approach. Initially, the data is prepared and the necessary computational framework is established through these steps:

- Creation of the TF-IDF matrix, $\mathbf{X}$, of the cleaned corpus
- $\mathbf{X}$ is decomposed using nonnegative tensor factorization from **T-ELF** enhanced with our new binary search strategy [47], to classify document clusters.

T-ELF allows us to extract highly specific features from the data. This method identifies latent topics within the corpus, grouping documents into clusters based on shared themes. To avoid over/under-fitting, automatic model determination is used where the final cluster counts are determined by achieving the highest silhouette scores above a predetermined threshold using the Binary Bleed method [47]. This method employs a binary search strategy across k values, selectively skipping those k values that do not surpass the silhouette threshold. The search criterion for an optimal k is defined as $k_{\text{optimal}} = \max\{k \in \{1, 2, \ldots, K\} : S(f(k)) > T\}$, where $S(f(k))$ denotes the silhouette score of the k-th configuration and T the threshold. Importantly, even after identifying an initial "optimal" k, higher k values are visited regardless to ensure no better configuration is overlooked.

The factorization of $\mathbf{X}$ yields two non-negative factor matrices $\mathbf{W} \in \mathbb{R}_{+}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}_{+}^{k \times n}$, ensuring $X_{ij} \approx \sum_{s} W_{is} H_{sj}$. Distribution of words over topics are captured in $\mathbf{W}$. The matrix $\mathbf{H}$ shows the topic distribution across documents, and is used to identify the predominant topic for each document in post-processing. Full tensor and matrix factorization implementations of various algorithms are available in **T-ELF** [2].

C. Knowledge Graph Ontology

Features from **T-ELF** and document metadata is mapped into series of head, entity, and tail relations, forming directional triplets, then injected into a Neo4j [48] KG.

Our KG incorporates document metadata as well as the latent features. The primary source of information in the KG comes from documents, which are injected into the graph along with related attributes. Each document node contains information such as DOI, title, abstract, and source API document identifiers. Additional node labels include authors,

[1] T-ELF is available at https://github.com/lanl/T-ELF

[2] Several tensor and matrix factorization algorithms: https://github.com/lanl/T-ELF/tree/main/TELF/factorization

publication year, Scopus category, affiliations, affiliation country, acronyms, publisher, topics, topic keywords, citations, references, and a subset of NER entities produced from spaCy's NER labels [49]. These NER labels cover events, persons, locations, products, organizations, and geopolitical entities.

The KG nodes represent documents and their associated metadata, while the edges capture the relationships between these entities, such as citations, co-authorships, and topic associations, enabling logical query and retrieval capabilities for the RAG.

D. Vector Store Assembly

To augment the RAG, we introduced a vector database for the original documents using Milvus [50]. Additionally, a subset of documents' full texts were vectorized and incorporated into the vector store. Full texts, when available, are segmented into smaller paragraphs, each assigned an integer ID to indicate its position within the original document. These paragraphs are then vectorized through the into embeddings using OpenAI's text-embedding-ada-002 [51] model and imported to the vector store to support the RAG process.

The RAG application can query the vector store to find relevant paragraph chunks from these full texts. If the retrieved text contains the needed information, the LLM can answer the posed question and include a citation of the document, precisely indicating the exact paragraph. If further related information is needed, the application can use document metadata (e.g., DOI, author) to expand its search through the KG. This approach allows us to preserve the semantics of the original documents and provide relevant responses.

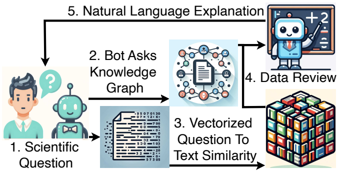
5. Natural Language Explanation
2. Bot Asks Knowledge Graph
4. Data Review
3. Vectorized Question To Text Similarity
1. Scientific Question

Fig. 2. The RAG pipeline. Images generated with DALL·E [52].

E. Retrieval Augmented Generation

RAG is an NLP method that mixes retrieval and generation techniques to improve the accuracy and relevance of responses in generative AI. It works by first gathering information from an external knowledge base based on a user's query. This retrieved information is then used to guide and enhance the outputs of the generative model, leading to more relevant and context-aware responses. By integrating these tactics, RAG addresses the limitations of purely generative models and provides an adaptable framework suitable for applications demanding detailed and current information.

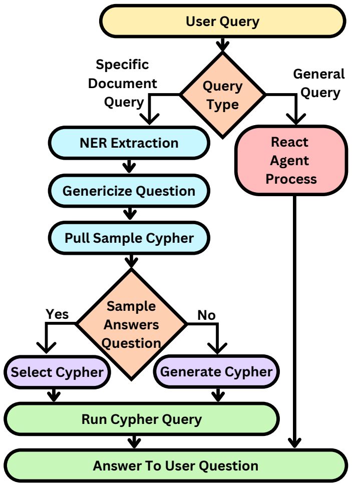

Fig. 3. User query routing overview.

Figure 2 demonstrates the data pipeline operated throughout the work for RAG. The process begins with a user query, which the LLM then uses to query the knowledge graph. The LLM transforms the query into a vector embedding. This embedding is compared to existing texts to find the most similar text. The retrieved information is appended to the original query, and the LLM produces a relevant answer using this context. Finally, the LLM constructs a final answer in natural language to explain the answer to the user's question.

To optimally leverage RAG, accurately understanding the user's question is crucial. Our RAG approach includes multiple potential routes depending on a user's question. The question routing pipeline may be a **General Query**, which calls the *ReAct Agent Process* [23], or a **Specific Document Query**, which calls either a *Retrieved Query* or a *Synthesized Query*. Understanding the question directs the information to the appropriate toolset and subsequent process. The routing process overview, as described below, can be seen in Figure 3.

Specific Document Query: If a user's question requires information from a specific document's text (title + abstract), it is better suited for a traditional RAG application in which the LLM interacts with the VS to find the needed text. In our case, we use a ReAct agent where the VS search is the sole tool, allowing the LLM to make multiple search requests as required. Specifically, a ReAct agent means the LLM has distinct steps for reasoning and acting after determining the input meaning. We use langgraph [53] to define an execution graph with three nodes, as illustrated in Figure 4: (1) the ReAct agent, (2) the tool executor, and (3) the end.

ReAct Agent Process: The agent node is the central part of the ReAct graph, where the LLM calls are encapsulated. The ReAct agent is responsible for collecting inputs, making actionable decisions, and explaining the results. The four prompt parts are:

a. Instructions
b. User query
c. Tool names, data
d. Tool Scratchpad

The agent is informed how to answer a user's query from the instructions, including answer formulations and tool usage. The query aids tool selection or answer directly. The tools have specific descriptions and parameters required for their calls, including schemas if interacting with databases. The scratchpad serves as temporary storage for tool calls, responses, and the LLM's reasoning, allowing the agent to iteratively solve complex problems.

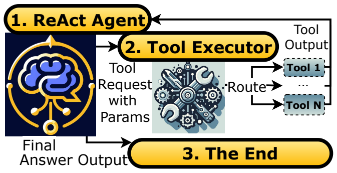

Fig. 4. Nodes and tools of the ReAct agent. Images from DALL·E [52].

The tool executor takes the tool name and input parameters from the agent node, routes to the corresponding function, and returns the output. It handles execution logistics, error handling, logging, and status updates.

The end node signals that the Reason-Act loop has completed. The final output from the LLM after the retrieval augmented generation is returned to the user.

General Query: If the user asks a broader question, such as those about trends, the required information is found within the KG. In this route, we start with a preprocessing step in which the LLM performs NER to decouple specific data from the genericized question. After this, we send the genericized query to a smaller vector store containing pairs of cypher queries and descriptions of the information they return, with embedding vectors generated from the descriptions. From here, there are two possible subroutes.

Retrieved Query: If a retrieved query is able to answer the question, we execute it's cypher before making a final LLM call to return the result. If no existing queries are able to answer, we synthesize a new cypher query.

Synthesized Query: If the LLM opts for "synthesis," it generates a new cypher query using the graph's schema and retrieved examples. For reliability, the LLM audits this generated query. First, we retrieve the query's execution plan and profile by using the cypher keyword "PROFILE," which lists the operators used on the knowledge graph. We also provide descriptions of the relevant low-level operators from Neo4j's official documentation. Once we obtain the detailed execution plan, the LLM performs two steps: it translates the plan into plain language and assesses if it addresses the user's question. Valid generations proceed as if retrieved queries.

IV. Results

In this section, we discuss identification of optimal clusters for tensor decomposition, vectorization of the dataset, construction of KG, and compare the system using the with GPT-4-instruct [51] as the operating model of **SMART-SLIC** to answer research questions. The same model was used to answer without RAG as well. Our findings highlight the accuracy and reliability of the **SMART-SLIC**'s RAG.

A. Dataset

Initially, 30 documents specializing on large-scale malware analysis and anomaly detection with tensor decomposition fields were selected by the SME as the core documents to construct the data. These documents were expanded along the citation/reference network 2 times. The final dataset was enumerated at 8,790 scientific publications. From the cleaned corpus, the tensor object was generated.

B. Extraction of Latent Features

After setting up the tensor, the most coherent grouping is determined by iterating through a range of $k = \{1, 2, 3, \ldots, 45\}$ clusters to decompose. Our analysis determined that 25 topic-clusters represented the optimal division across all evaluated k values. The decomposition itself was executed using **T-ELF** on high-performance computing resources, specifically two AMD EPYC 9454 48-Core Processors. This setup provided a total of 192 logical CPUs, enabling us to complete the entire decomposition process in approximately 2 hours. Following the decomposition, post-processing refined and defined clusters for the topics, which are listed in Table I.

TABLE I
Labels for Topic Clusters

#	Label	# Docs.	Percent
0	Malware Behavioral Analysis	158	1.80
1	Cybersecurity Challenges	305	3.47
2	Cybersecurity Research	114	1.30
3	Botnet Detection Techniques	142	1.62
4	Malware Feature Selection And Extraction	353	4.02
5	Network Intrusion Detection	134	1.52
6	Evaluation of Malware Classifiers	301	3.42
7	Malicious Code Analysis	827	9.41
8	Artificial Intelligence for Malware	888	10.10
9	Nonnegative Matrix Decomposition	520	5.92
10	Security Threat Mitigation	180	2.05
11	Deep Learning for Malware	113	1.29
12	Machine Learning Techniques	275	3.13
13	Education Technology	447	5.09
14	Unsupervised Anomaly Detection	372	4.23
15	Ransomware Prevention	147	1.67
16	Temporal Graph Forecast	307	3.49
17	Mobile Malware Detection	230	2.62
18	Adversarial Defense Strategy	358	4.07
19	IoT Security	238	2.71
20	Privacy Protection Challenge	628	7.14
21	Sparse Tensor Decomposition	212	2.41
22	Backdoor Detection	350	3.98
23	Neural Network Architecture	581	6.61
24	Malware Analysis Techniques	610	6.94

C. Vector Store

The 8,790 documents were vectorized and ingested into the Milvus vector store. When questions are posed to the framework, they are also vectorized using this model. Of the total documents, 22% had full-texts available, which were vectorized into the Milvus. Each document and full-text had a DOI, with the full-texts also including paragraph identifiers.

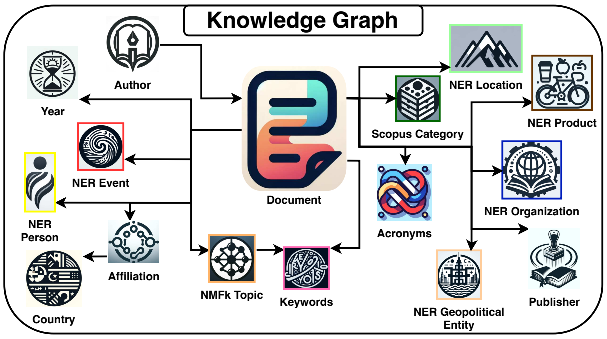

Fig. 5. The KG schema. Images generated with DALL·E [52].

D. Knowledge Graph

From the 25 clusters output form **T-ELF**, we formatted the the data into 1,457,534 triplets. Once injected into the knowledge graph, there were 321,122 nodes and 1,136,412 edge relationships. The nodes injected into the graph are represented in Figure 5, where they are organized into 16 base categories, referred to as labels, that define the foundational classes for the injection process. Once the graph was built was directly queried for information as Structured Query Language (SQL) is directly queriable outside of an application. In Figure 6, the knowledge graph is queried for the SME keyword related to cybercrime. The query is structured as:

```
MATCH  (k:Keyword)-[r1]-(d:Document)-[r2]
-(aff:Affiliation)-[r3]-(c:Country )
WHERE k.term CONTAINS 'cybercrime'
RETURN k,r1,d,r2,aff,r3,c
```

To retrieve the country nodes from a keyword, several relationships were navigated. First, from the keyword to documents, then from documents to affiliations and finally from the affiliations to the countries. In the cypher query, these links are the denoted as an r with a following integer, where r is the relationship identifier. The syntax is ()-[]-()-[]-()-[]-(), where brackets are relations and parenthesis are nodes. In the first part of the "where" clause, the keyword label is further tailored to the keyword node, such that it must contain "cybercrime." Overall this can answer the question, "which countries have published papers that mention cybercrime?" The question's retrieved nodes in Figure 6 has 29 countries in red, 99 affiliated institutions in yellow, and 65 published documents in blue.

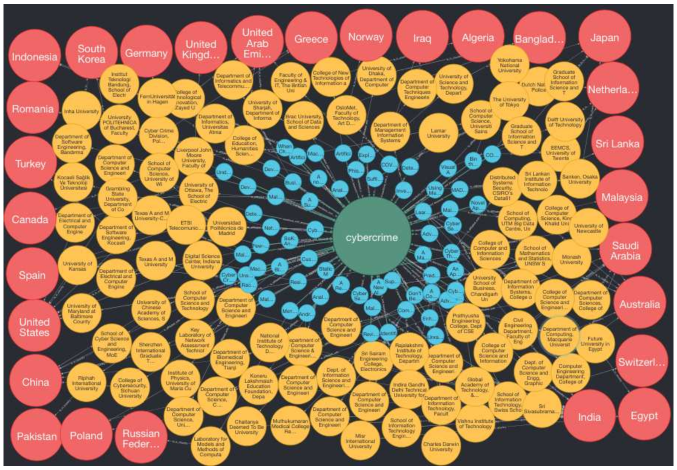

Fig. 6. Keyword 'cybercrime' graph search. A single keyword (green), along with linked documents (light blue) are returned. The documents also link affiliated institutions (yellow), and the country of the institutions (red).

E. Question Answering Validation

The raw data collected was analyzed using document-specific questions in Zero-Shot Conditioning, including:

- How many citations are there for *DOI*?
- How many references are there for *DOI*?
- How many authors are there for *DOI*?
- What year was *DOI* published?
- Which publisher published *DOI*?
- How many scopus categories are assigned to *DOI*?
- What is the title of *DOI*?

After document specific questions, we then examined topic specific questions, which included year variations, as in:

- How many papers are there on the topic of *Topic*?
- How many papers were written related to *Topic* in *Year*?

In total, there were 200 questions in this set. Using these questions, in this study, we compare the performance of GPT-4-instruct [51] with and without our RAG framework on both topic-specific and document metadata questions. As shown in Figure 7, our findings indicate that GPT-4 with RAG answers all questions with a 97% accuracy rate. In contrast, without RAG, GPT-4 abstains from answering 40% of the questions, and the accuracy of the answered questions drops to 20%. A similar trend is observed for topic-based questions, where the specialized RAG significantly enhances the retrieval of correct answers. The topic questions attempted with RAG was also 100%, but without was only 36%. In consideration of only the attempted questions, the system with RAG answered the topic questions correctly 92%. Without RAG, the LLM answered the topic questions with 27.77% accuracy.

Without RAG, several questions about years were answered incorrectly, with the system stating the year didn't exist. The LLM also struggled with author and reference details, often asking for more information or recommending consulting a human expert. In some cases, it noted its lack of internet access but later suggested using Google Scholar, yet still provided inaccurate responses.

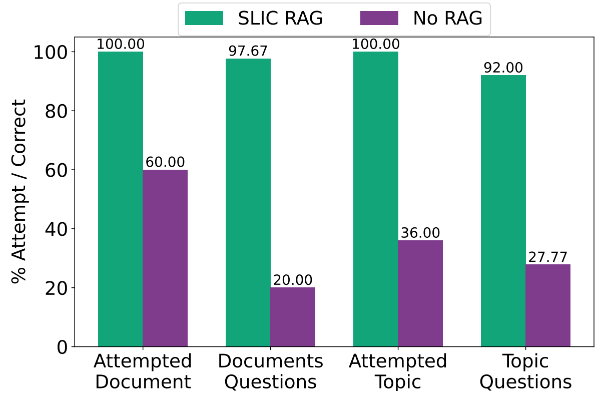

Fig. 7. Two question types, document and topic, showing LLM percent attempts and correct with **SMART-SLIC** RAG and without RAG.

The analysis highlights the effectiveness of **SMART-SLIC** in accurately answering nearly all 200 research queries, covering both document-specific and topic-specific questions. In contrast, GPT-4-instruct without RAG struggled. **SMART-SLIC** excelled in handling complex queries related to citations, references, authorship, and topical relevance, underscoring the advantages of integrating its methodologies for more reliable AI-driven research tools and improved information retrieval in scientific fields.

We then shifted to more complex questions requiring thorough searches through various retrieval methods, using GPT-4-instruct with and without RAG. The SME posed the following questions, and the responses are from the **SMART-SLIC** RAG process, with DOI citations provided in square brackets:

1) *What challenge is associated with outlier detection in high-dimensional data?* The challenge associated with outlier detection in high-dimensional data is the "curse of dimensionality," which makes it difficult to detect outliers in the full data space [10.1109/ICCC54389.2021.9674561].
2) *Can tensor decomposition methods be leveraged to detect SCADA anomalies?* Yes, tensor decomposition methods can be leveraged to detect SCADA anomalies. The use of non-negative tensor decomposition, specifically the Canonical Polyadic Alternating Poisson Re-

gression (CP-APR) method, has been applied to model the complex and multi-faceted activity details of SCADA events, improving the specificity and accuracy of identifying anomalies in electrical grid systems [10.1109/MILCOM58377.2023.10356348].

3) *What does MADHAT stand for?* The acronym MADHAT stands for Multidimensional Anomaly Detection fusing HPC, Analytics, and Tensors [10.1109/HPEC.2019.8916559].
4) *What's the most advanced anomaly detection method on the LANL Unified Host and Network Dataset?* The most advanced anomaly detection method on the Los Alamos National Laboratory Unified Host and Network Dataset is the tensor factorization model that exploits the multi-dimensional and sparse structure of user authentication logs for improved detection of penetration testing events [10.1109/ISI49825.2020.9280524].

For each response, the **SMART-SLIC** agent selected DOIs that the SME also chose, demonstrating the agent's accuracy in retrieving relevant sources. The consistency in DOI selections highlights the robustness of the retrieval mechanisms, ensuring reliable and pertinent information for the user's questions.

The same questions were asked without RAG, and the results varied. The LLM answered the first general question accurately, but while the initial response to the second question was correct, its elaboration missed key details. The third and fourth responses were entirely wrong, with fabricated answers like "Malware and Attack Detection Hunting and Analysis Team" and "Long Short-Term Memory." Additionally, none of the responses included DOI citations, reducing the credibility of the information by omitting source references.

The evaluation of **SMART-SLIC** and GPT-4-instruct, with and without RAG, highlights the importance of retrieval systems for accurate research output. **SMART-SLIC**'s RAG excelled in selecting relevant DOI citations for complex queries, while GPT-4-instruct struggled with fabrications, showing the need for advanced systems like **SMART-SLIC**. Its strength lies in using high-quality, domain-specific corpora for strong performance in defined research areas, while also offering potential for further exploration in less-defined domains.

V. Conclusion

Our **SMART-SLIC** framework leverages advanced language models and specialized tools to effectively address user queries by categorizing them into Specific Document Queries and General Queries for efficient processing. The ReAct agent manages general inquiries, while NER and cypher query generation handle document-specific questions.

LLMs excel in general NLP tasks but struggle in domain-specific areas due to hallucinations, knowledge cut-offs, and lack of attribution. Our system addresses this by integrating RAG with a domain-specific KG and VS, enhancing reliability without fine-tuning. Built using NLP, data mining, and non-negative tensor factorization, this setup enables accurate attributions, reduces hallucinations, and excels in domain-specific queries, as shown in malware analysis research.

The framework significantly enhances query response accuracy and reliability, making it adaptable to various applications. Future work will expand the framework's use across domains like robotics, materials science, legal cases, and quantum computing. Enhancements in graph completion, entity linking, and link prediction will further interconnect graphs, reveal hidden connections, and support LLMs in information clarification, keeping **SMART-SLIC** at the forefront of intelligent information retrieval and generation.

References

[1] Y. A. Yadkori, I. Kuzborskij, A. György, and C. Szepesvári, "To believe or not to believe your llm," *arXiv preprint arXiv:2406.02543*, 2024.

[2] N. Harvel, F. B. Haiek, A. Ankolekar, and D. J. Brunner, "Can llms answer investment banking questions? using domain-tuned functions to improve llm performance on knowledge-intensive analytical tasks," in *Proceedings of the AAAI Symposium Series*, vol. 3, no. 1, 2024, pp. 125–133.

[3] N. Zhang, Y. Yao, B. Tian, P. Wang, S. Deng, M. Wang, Z. Xi, S. Mao, J. Zhang, Y. Ni *et al.*, "A comprehensive study of knowledge editing for large language models," *arXiv preprint arXiv:2401.01286*, 2024.

[4] S. K. Freire, C. Wang, and E. Niforatos, "Chatbots in knowledge-intensive contexts: Comparing intent and llm-based systems," *arXiv preprint arXiv:2402.04955*, 2024.

[5] A. Bertsch, M. Ivgi, U. Alon, J. Berant, M. R. Gormley, and G. Neubig, "In-context learning with long-context models: An in-depth exploration," *arXiv preprint arXiv:2405.00200*, 2024.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[7] R. Vangara, E. Skau, G. Chennupati, H. Djidjev, T. Tierney, J. P. Smith, M. Bhattarai, V. G. Stanev, and B. S. Alexandrov, "Semantic nonnegative matrix factorization with automatic model determination for topic modeling," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 328–335.

[8] B. T. Nebgen, R. Vangara, M. A. Hombrados-Herrera, S. Kuksova, and B. S. Alexandrov, "A neural network for determination of latent dimensionality in non-negative matrix factorization," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025012, 2021.

[9] R. Vangara, M. Bhattarai, E. Skau, G. Chennupati, H. Djidjev, T. Tierney, J. P. Smith, V. G. Stanev, and B. S. Alexandrov, "Finding the number of latent topics with semantic non-negative matrix factorization," *IEEE access*, vol. 9, pp. 117 217–117 231, 2021.

[10] J. Liu, "LlamaIndex," 11 2022. [Online]. Available: https://github.com/jerryjliu/llama_index

[11] R. F. Simmons, "Natural language question-answering systems: 1969," *Commun. ACM*, vol. 13, no. 1, p. 15–30, jan 1970. [Online]. Available: https://doi.org/10.1145/361953.361963

[12] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024.

[13] S. Pan, Y. Zheng, and Y. Liu, "Integrating graphs with large language models: Methods and prospects," *IEEE Intelligent Systems*, vol. 39, no. 1, pp. 64–68, 2024.

[14] Y. Li, Z. Li, P. Wang, J. Li, X. Sun, H. Cheng, and J. X. Yu, "A survey of graph meets large language model: Progress and future directions," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-234*, 2023.

[15] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1870–1879. [Online]. Available: https://aclanthology.org/P17-1171

[16] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550

[17] J. Tan, Z. Dou, Y. Zhu, P. Guo, K. Fang, and J.-R. Wen, "Small models, big insights: Leveraging slim proxy models to decide when and what to retrieve for llms," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, 2024.

[18] J. Jiang, K. Zhou, X. Zhao, Y. Li, and J.-R. Wen, "ReasoningLM: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3721–3735. [Online]. Available: https://aclanthology.org/2023.emnlp-main.228

[19] L. LUO, Y.-F. Li, R. Haf, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=ZGNWW7xZ6Q

[20] B. Jin, C. Xie, J. Zhang, K. K. Roy, Y. Zhang, Z. Li, R. Li, X. Tang, S. Wang, Y. Meng, and J. Han, "Graph chain-of-thought: Augmenting large language models by reasoning on graphs," 2024. [Online]. Available: https://arxiv.org/abs/2404.07103

[21] M. Li, H. Yang, Z. Liu, M. M. Alam, Ebrahim, H. Sack, and G. A. Gesese, "KGMistral: Towards boosting the performance of large language models for question answering with knowledge graph integration," in *Workshop on Deep Learning and Large Language Models for Knowledge Graphs*, 2024. [Online]. Available: https://openreview.net/forum?id=JzL0qm3YA8

[22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[23] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X

[24] J. Lála, O. O'Donoghue, A. Shtedritski, S. Cox, S. G. Rodriques, and A. D. White, "Paperqa: Retrieval-augmented generative agent for scientific research," *arXiv preprint arXiv:2312.07559*, 2023.

[25] D. Sanmartin, "Kg-rag: Bridging the gap between knowledge and creativity," 2024. [Online]. Available: https://arxiv.org/abs/2405.12035

[26] K. Soman, P. W. Rose, J. H. Morris, R. E. Akbas, B. Smith, B. Peetoom, C. Villouta-Reyes, G. Cerono, Y. Shi, A. Rizk-Jackson *et al.*, "Biomedical knowledge graph-enhanced prompt generation for large language models," *arXiv preprint arXiv:2311.17330*, 2023.

[27] C. Edwards, "Hybrid context retrieval augmented generation pipeline: Llm-augmented knowledge graphs and vector database for accreditation reporting assistance," 2024.

[28] N. Matsumoto, J. Moran, H. Choi, M. E. Hernandez, M. Venkatesan, P. Wang, and J. H. Moore, "KRAGEN: a knowledge graph-enhanced RAG framework for biomedical problem solving using large language models," *Bioinformatics*, vol. 40, no. 6, p. btae353, 06 2024. [Online]. Available: https://doi.org/10.1093/bioinformatics/btae353

[29] Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, and Z. Li, "Retrieval-augmented generation with knowledge graphs for customer service question answering," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2905–2909. [Online]. Available: https://doi.org/10.1145/3626772.3661370

[30] N. Solovyev, R. Barron, M. E. Eren, K. O. Rasmussen, M. Bhattarai, I. D. Boureima, and B. S. Alexandrov, "Slic: Scientific leadership identification and characterization: Interactive distillation of large single-topic corpora of scientific papers," DOE Data Days (D3) at Lawrence Livermore National Laboratory, LA-UR-23-30223, Tech. Rep., 2023.

[31] M. Eren, N. Solovyev, R. Barron, M. Bhattarai, D. Truong, I. Boureima, E. Skau, K. O. Rasmussen, and B. Alexandrov, "Tensor Extraction of Latent Features (T-ELF)," Oct. 2023. [Online]. Available: https://github.com/lanl/T-ELF

[32] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *Journal of Network and Computer Applications*, vol. 185, p. 103076, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521000990

[33] S. Yu, Z. Yuan, J. Xia, S. Luo, H. Ying, S. Zeng, J. Ren, H. Yuan, Z. Zhao, Y. Lin, K. Lu, J. Wang, Y. Xie, and H.-Y. Shum, "Bios: An algorithmically generated biomedical knowledge graph," 2022. [Online]. Available: https://arxiv.org/abs/2203.09975

[34] N. Solovyev, R. Barron, M. Bhattarai, M. E. Eren, K. O. Rasmussen, and B. S. Alexandrov, "Interactive distillation of large single-topic corpora of scientific papers," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023, pp. 1000–1005.

[35] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2022.

[36] H. Ye, H. Gui, A. Zhang, T. Liu, W. Hua, and W. Jia, "Beyond isolation: Multi-agent synergy for improving knowledge graph construction," 2023. [Online]. Available: https://arxiv.org/abs/2312.03022

[37] S. Wadhwa, S. Amir, and B. Wallace, "Revisiting relation extraction in the era of large language models," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 15 566–15 589. [Online]. Available: https://aclanthology.org/2023.acl-long.868

[38] M. Marinov, Y. Benkhedda, G. Nenadic, and R. Batista-Navarro, "Relation extraction for constructing knowledge graphs: Enhancing the searchability of community-generated digital content (CGDC) collections," in *Workshop on Deep Learning and Large Language Models for Knowledge Graphs*, 2024. [Online]. Available: https://openreview.net/forum?id=ZOKivqqTjg

[39] V. Zavarella, J. C. Gamero, and S. Consoli, "A few-shot approach for relation extraction domain adaptation using large language models," in *Workshop on Deep Learning and Large Language Models for Knowledge Graphs*, 2024. [Online]. Available: https://openreview.net/forum?id=rBUbEKOECY

[40] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," 2024. [Online]. Available: https://arxiv.org/abs/2404.16130

[41] X. Lv, X. Han, L. Hou, J. Li, Z. Liu, W. Zhang, Y. Zhang, H. Kong, and S. Wu, "Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 5694–5703. [Online]. Available: https://aclanthology.org/2020.emnlp-main.459

[42] W. Chen, Y. Cao, F. Feng, X. He, and Y. Zhang, "Hogrn: Explainable sparse knowledge graph completion via high-order graph reasoning network," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–13, 2024.

[43] Elsevier, "Scopus," 2024, accessed: 2024-07-20. [Online]. Available: https://www.scopus.com

[44] Allen Institute for AI, "Semantic scholar," 2024, accessed: 2024-07-20. [Online]. Available: https://www.semanticscholar.org

[45] U.S. Department of Energy, "Office of scientific and technical information (osti)," 2024, accessed: 2024-07-20. [Online]. Available: https://www.osti.gov

[46] M. Ostendorff, N. Rethmeier, I. Augenstein, B. Gipp, and G. Rehm, "Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings," in *The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*. Abu Dhabi: Association for Computational Linguistics, December 2022, 7-11 December 2022. Accepted for publication.

[47] R. Barron, M. E. Eren, M. Bhattarai, I. Boureima, C. Matuszek, and B. S. Alexandrov, "Binary bleed: Fast distributed and parallel method for automatic model selection," 2024. [Online]. Available: https://arxiv.org/abs/2407.19125

[48] Neo4j, Inc., "Neo4j: The #1 platform for connected data," https://neo4j.com/, 2023.

[49] Explosion AI, "spacy english core web transformer model," https://spacy.io/models/en#en_core_web_trf, 2023.

[50] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu *et al.*, "Milvus: A purpose-built vector data management system," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2614–2627.

[51] OpenAI, "Openai api," 2024, accessed: 2024-07-28. [Online]. Available: https://www.openai.com/api/

[52] OpenAI's DALL·E, "Visual representations of llms, kg, & rag concepts," 2024.

[53] L. Inc., "Langgraph: Building language agents as graphs," https://langchain-ai.github.io/langgraph/, 2024, version 1.0.