LLMs for Closed-Library Multi-Document Query, Test Generation, and Evaluation

Claire Randolph, Maj[*], Adam Michaleas[†], Darrell O. Ricke, Ph.D.[†]
[*]Department of the Air Force, Artificial Intelligence Accelerator, Cambridge, MA, USA
{claire.randolph}@us.af.mil
[†]MIT Lincoln Laboratory, Lexington, MA, USA
{adam.michaleas, darrell.ricke}@ll.mit.edu

***Abstract*—Learning complex, detailed, and evolving knowledge is a challenge in multiple technical professions. Relevant source knowledge is contained within many large documents and information sources with frequent updates to these documents. Knowledge tests need to be generated on new material and existing tests revised, tracking knowledge base updates. Large Language Models (LLMs) provide a framework for artificial intelligence-assisted knowledge acquisition and continued learning. Retrieval-Augmented Generation (RAG) provides a framework to leverage available, trained LLMs combined with technical area-specific knowledge bases. Herein, two methods are introduced, which together enable effective implementation of LLM-RAG question-answering on large documents. Additionally, the AI tools for knowledge intensive tasks (AIKIT) solution is presented for working with numerous documents for training and continuing education. AIKIT is provided as a containerized open source solution that deploys on standalone, high performance, and cloud systems. AIKIT includes LLM, RAG, vector stores, relational database with a Ruby on Rails web interface.**

***Index Terms*—Artificial Intelligence (AI), Large Language Model (LLM), Retrieval-Augmented Generation (RAG), Document as a Dictionary (DaaDy), Structured Question Answer Dictionary (SQuAD)**

I. Introduction

Some highly technical professions require learning and retention of complex, detailed, and evolving knowledge from multiple relevant documents and information sources. Adding more complexity, these documents are updated with new and changing information on a frequent basis, which makes keeping up-to-date on the most current information a challenging task for these individuals. In professions with a specified instructor corps, generating and maintaining instructional material on such a dynamic and vast corpus can be overwhelming and time-consuming for instructors. Knowledge tests can assist learners in encoding and retaining new knowledge, but can demand a considerable amount of time and personnel to generate and maintain. Learners are repeatedly exposed to bad information when existing knowledge tests become outdated as source information is modified or removed. In high-risk professions, such as medicine or aviation, it is imperative that learners have access to the most up-to-date corpus of documents and study materials.

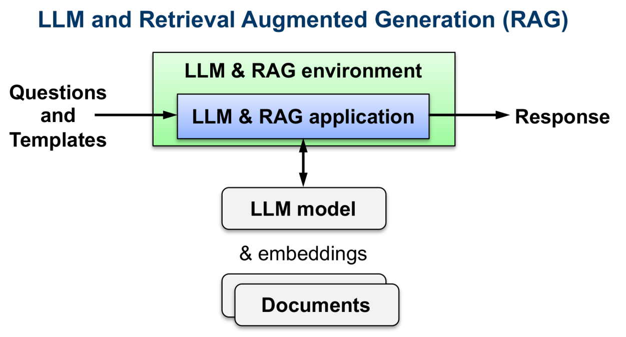

Fig. 1. Large Language Models (LLM) and Retrieval-Augmented Generation (RAG) overview

Recent development of Large Language Models (LLMs) combined with Retrieval-Augmented Generation (RAG) of documents and information not included in the LLM training data provides a framework of technology solutions to address aspects of these education challenges. Multiple documents can be embedded into one or more embedded databases or vector stores. LLM RAG can be used to query the knowledge base for specific questions; this enables rapid lookup of information across multiple large documents. LLM RAG implementation is able to perform very well on question-answering (QA), fact verification, and attribution tasks while hallucinating less than other methods [1], [2]. However, current LLM RAG capabilities fall short of fully utilizing the context of a document; LLM RAG is susceptible to what is known as the *lost-in-the-middle* challenge, where the LLM struggles to fully utilize information hidden within a long context [3], [4]. If implemented for knowledge-intensive professions with current methods, critical information may be lost or overlooked.

To evaluate LLM RAG for enhancing and facilitating education on complex, jargon-dense, closed-library documents, the Artificial Intelligence Tools for Knowledge Intensive Tasks (AIKIT) system was developed. To provide portability, AIKIT has been containerized in both Singularity [5] and Docker [6] containers and a Conda environment. AIKIT includes a Ruby on Rails web user interface. AIKIT is being released as open source at https://github.com/mit-ll/AIKIT.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

II. Methods

A. Document as a Dictionary – DaaDy

To solve the problem of incomplete text utilization for LLM RAG on large documents, **D**ocument **a**s **a** **D**ictionar**y** - **DaaDy** was developed. DaaDy is a framework in which LLM RAG can be systematically completed on each section/subsection/sentence of a document. This method takes structured documents (documents with headings, sections, and/or subsections), parses them, and stores the entire document as a series of nested dictionaries where the highest-level key is a heading/section/subsection title, and the lowest-level value is an individual sentence from the document. This is implemented with two Python tools, one for parsing a document into a DaaDy (afman_parser.py) and another to consolidate multiple dictionaries (daady_consolidator.py). Using this dictionary framework, metadata can also be stored for added functionality. The DaaDy framework allows the prompt to be queried against all sections of a document by loading each section/subsection/sentence into the retriever, individually; context length remains short enough to achieve full utilization in LLM RAG.

B. Structured Question Answer Dictionary – SQuAD

To combine LLM RAG with DaaDy, the method called **S**tructured **Qu**estion **A**nswer **D**ictionary, or **SQuAD** was developed. SQuAD is able to generate new material for knowledge tests, with each item made up of question (Q) and answer (A), along with section or paragraph reference (R), henceforth referred to as QAR. SQuAD can also be used to locate context and assess the validity of existing QARs in knowledge tests after document revisions in the knowledge base. The expedient LLM RAG assessment of current QARs and generation of new QARs on updates to the knowledge base can provide benefits to instructors and learners in knowledge-intensive professions.

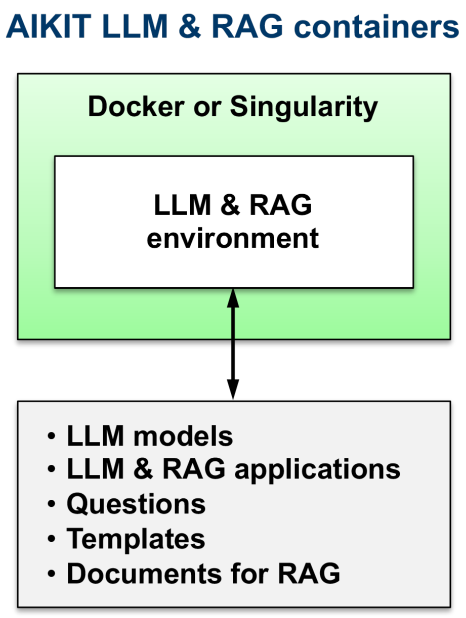

Fig. 2. Docker and Singularity containerized AIKIT

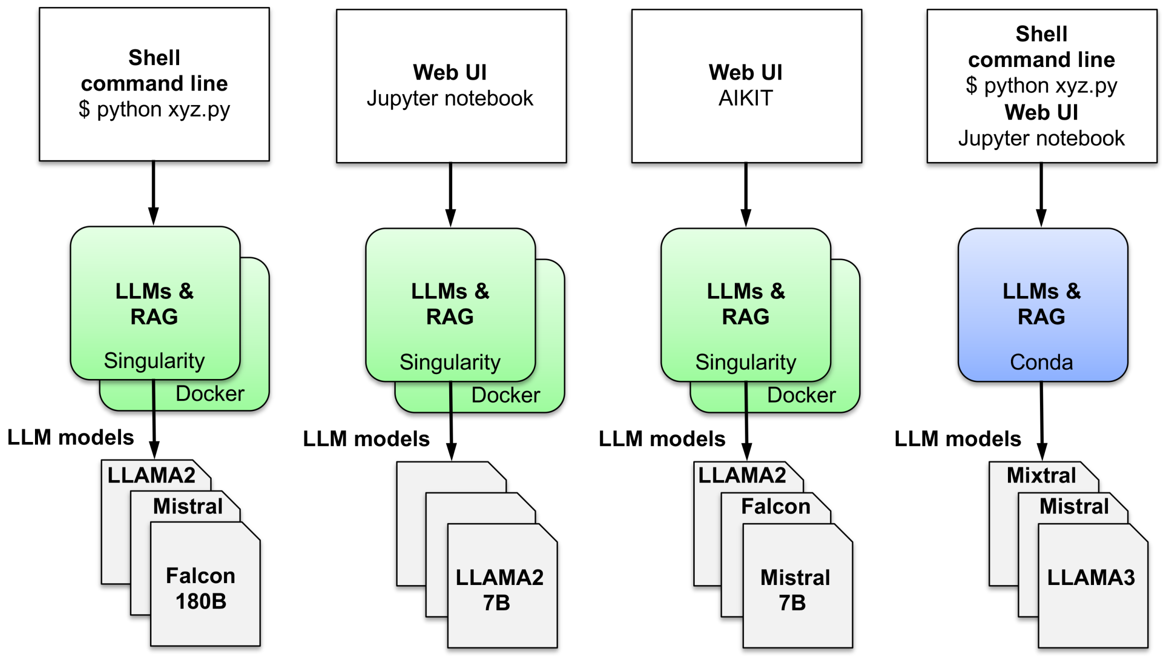

Fig. 3. AIKIT command line and web interfaces

C. Containerized AI tools for knowledge intensive tasks (AIKIT)

To easily enable hosting on multiple platforms, AIKIT was packaged into Singularity [5] and Docker [6] containers (Figure 2). AIKIT is also packaged in a Conda environment (Figure 3).

D. Large Language Models and Retrieval-Augmented Generation

AIKIT is not dependent upon any specific LLM. The LLM models Mistral-7B-Instruct-v0.2 [7], Mixtral-8x7B-Instruct-v0.1 [8] models from Mistral AI, and other models have been used with AIKIT. LLM RAG was implemented in Python [9] (v3) with LangChain [10], vector stores (embedding databases) FAISS [11] and chroma [12], and HuggingFace embeddings model sentence-transformers all-mpnet-base-v2 [13]. The LangChain PyPDFLoader [14] was used for parsing Adobe portable document format (PDF) documents. Paired Python tools were developed to create vector stores (docs_to_vs.py) and LLM RAG queries (llm_rag_query.py). These two Python tools accept JavaScript Object Notation (JSON) parameter files for input.

E. Web interface

AIKIT user interface was developed in Ruby on Rails [15] (v7.0.1) and Ruby [16] (v3.0.3). The SQLite3 database was used for development, but AIKIT will work with any Rails supported database. The AIKIT user interface invokes the Python tools docs_to_vs.py and llm_rag_query.py to create vector stores and LLM RAG queries, respectively.

F. Multi-GPU Enabled Systems

Singularity container and nvccli options were utilized to parallelize across all of the available GPUs on the hosting platform.

When running with –nvccli, by default SingularityCE will expose all GPUs on the host inside the container. This mirrors the functionality of the legacy GPU support for the most common use-case. Setting the SINGULARITY_CUDA_VISIBLE_DEVICES environment variable before running a container is still supported, to control which GPUs are used by CUDA programs that honor CUDA_VISIBLE_DEVICES.

However, more powerful GPU isolation is possible using the –contain flag and NVIDIA_VISIBLE_DEVICES environment variable. This controls which GPU devices are bound into the /dev tree in the container. For example, to pass only the first GPU into a container running on a system with multiple GPUs, one would export the following variable values as shown below to achieve this:

```
export NVIDIA_VISIBLE_DEVICES=0
export SINGULARITY_CUDA_VISIBLE_DEVICES=0
```

The Singularity contain and nvccli options were used with GNU Parallel [17]. A master shell script was created for each GPU with a text file containing the commands to run.

G. Prototyping Environment

AIKIT development and prototyping efforts were performed on both x86 and ARM-based architectures. The x86 system had two Intel Xeon Gold 6258R CPUs [18], 256GB RAM, and an NVIDIA RTX A6000 GPU. The ARM-based system had an Apple M2 known as a system on a chip which serves as both a CPU and a GPU [19], 8GB RAM, and a 256GB solid state hard drive.

H. HPC System Implementation (2-NVIDIA-V100)

The MIT Lincoln Laboratory Tx-Green system (2-NVIDIA-V100) [20] was used as the high performance computing system for our pipeline prototype development.

The GPU systems have Intel Xeon PHI 7210 64C 2.5 GHz CPU with 40 cores, 377 GB RAM, Intel Omni-Path with 2 NVIDIA Tesla V100 GPUs. LLMapReduce was used to submit jobs to the SLURM queue [21].

III. Results

A. Document as a Dictionary – DaaDy

Figure 4 shows that while the specific oscillations differ between documents and individual runs, a strong trend of decreasing context utilization is consistent across all cases during 300 attempts. In no case did the LLM RAG utilize more than 25 percent of the context when the document was longer than 18,000 characters. On average, across all 6 context bases, less than 20 percent of the context was utilized when documents were longer than 10,000 characters and less than 10 percent of the context was utilized when documents were longer than 20,000 characters; our data suggests a full-utilization maximum of between 1,000 and 2,000 characters. While research seeking to decrease the magnitude of this effect continues, instructors and learners who intend to use LLM RAG to generate training material currently lack the capability to do so effectively on long documents without losing critical information. The DaaDy framework was developed to allow question generation coverage of the document sections individually, ensure all desired content is utilized.

B. Test Questions Generation

Question, Answer, Reference (QAR) groups were generated on selected documents with LLM RAG. The goal was to comprehensively utilize the material in the selected documents from which a subset of useful, accurate, and well-phrased questions could be selected. A prompt was given for the LLM to generate a QAR for each sentence in the document which was longer than five words (see Appendix A for final prompts

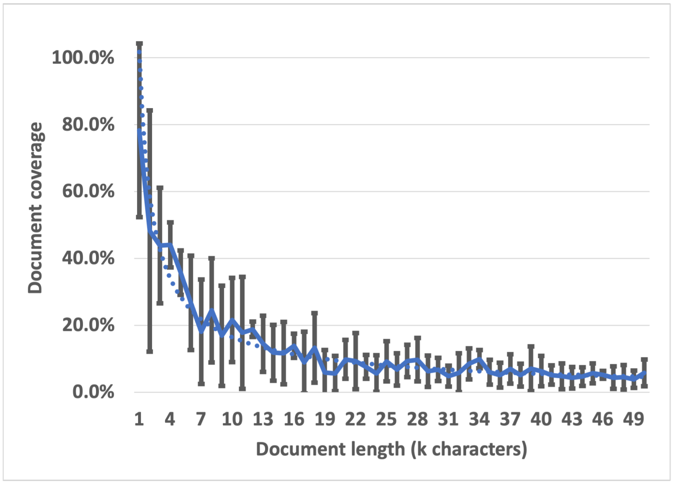

Fig. 4. Document coverage by LLM RAG generated questions

used in this research). Initially, this prompt was implemented on the document in its entirety, and a significant amount of context was unrepresented in the questions generated. It was observed that for documents less than 1,000 characters long, there was generally very high content coverage, measured by assessing the number of QARs output divided by the number of sentences in the document which were greater than five words long (a result of 1.0 was assessed as full context utilization). To study this effect further, the prompt was tested on documents of varying lengths in order to assess where information was being utilized and lost; six documents were used in total (Figure 4). The prompt was implemented and from the output, the location of each reference was derived as a percentage of the full document length. A noticeable bias of content from beginning of the document was noted (Figure 5) with 5 of 6 documents showing between 17 and 26 percent of the questions generated originating from the first 10 percent of the document (a single outlier at 9% was observed). In the 6 documents examined, QARs produced about content at 30%, 90%, and 100% of document were below the expected 10% percentage of questions (Figure 5).

To mitigate the *lost-in-the-middle* effect, DaaDy was created. DaaDy takes a document as the input and separates the document into a series of nested dictionaries containing sections, subsections, and sentences. While future users could customize the base-level of DaaDy to their needs, our testing used the sentence as the lowest level value in the dictionary. SQuAD calls the prompt separately on each desired section of the dictionary, creating a QAR for each sentence in the document. This also permits the storage of metadata about each sentence in the document, which by alleviating the LLM from the responsibility of correctly interpreting and storing data from the text, allows the user to store and retrieve sentence-level metadata with perfect recall.

Unsurprisingly, implementation of the prompt on sentence-level DaaDy data resulted in a perfect score for context utilization: for a 105,000 character-long document, 910 QARs were produced in approximately 24 minutes and 30 seconds, resulting in a per-question QAR time of 1.62 seconds on an ARM-based system. An expert in the field was asked to assess the QARs on their utility, accuracy, and phrasing. The expert was also asked to identify and categorize any anomalies in the QARs produced by LLM RAG. Out of 477 questions assessed, there were 123 questions flagged as incongruous with the text provided. There were seven main categories of anomalous QARs (see Appendix B for definitions and examples): unable to answer, repetitive QA, unnecessary justification, missing context (lists), non-sequitur, misleading QA, and acronym hallucination. For both SQuAD question generation and evaluation, significant degradation in LLM RAG performance was observed when niche acronyms were used or phrases were used outside of their normal context.

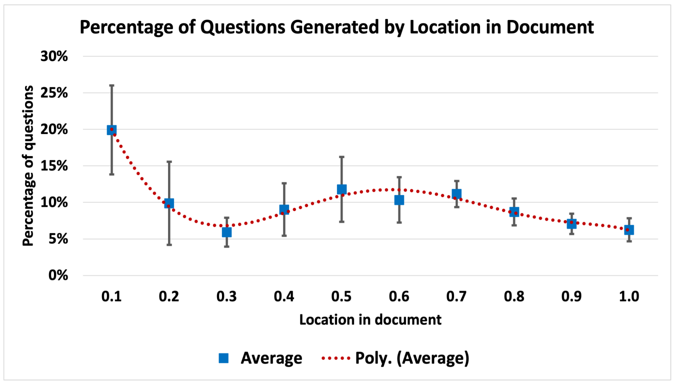

Fig. 5. Context Utilization in Varying Document Lengths

C. Test Questions Evaluation

Existing test questions based on outdated references and publications were evaluated with LLM RAG on documents via SQuAD to identify whether the question was still supported by the knowledge base, in need of revision, or if relevant content had been removed. Two question-evaluation trials were run. First, each question in the test was posed using the entire source publication as the context. Second, the same queries were made using only the localized context from the DaaDy as search context. The results of these methods were compared against an expert's assessment of the test questions. The expert compared each QAR against the current source publication and given paragraph reference from the source document. The QAR was categorized into one of three bins: 1) correct answer contained in specified reference context, 2) correct answer not contained in specified reference context, 3) question verbiage so vague that a specific, correct answer could not be reasonably determined. Once this gold standard was established, the expert graded the answers generated in each of the two trial methods and categorized each response into one of nine categories (see Appendix C for definitions and examples): false response, irrelevant response, correct response, correct absence, incorrect absence, irrelevant response, incomplete response, RAG error, and context regurgitation responses. The results of these two trials are summarized in figure 6.

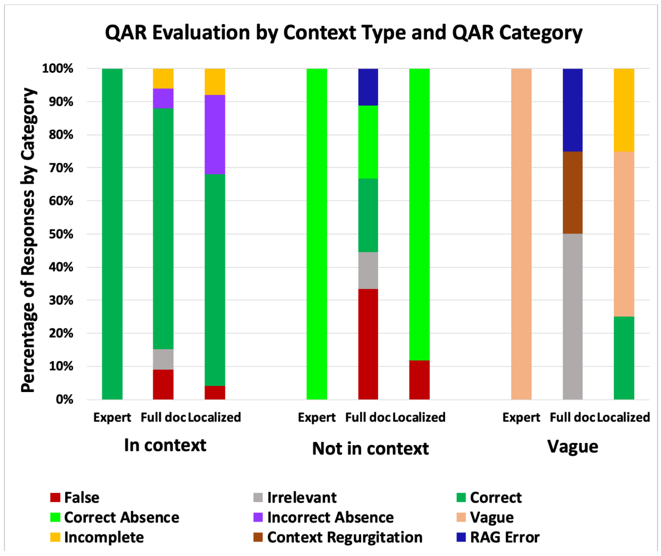

Fig. 6. Context-Based Question Evaluation versus Expert Assessment

D. AIKIT User Interface

A Ruby on Rails web interface was developed for AIKIT. The AIKIT UI includes a user interface enabling access to documents, document queries (LLM RAG), tests, and test results. LLM model queries and LangChain [10] chaining of questions is also included. The instructor interface is also included with access to test questions and answers, and evaluation of test questions.

E. Documents Query

Querying knowledge base documents is implemented in AIKIT as standard RAG embedding of documents with a LLM. Queries can be run via command line, Jupyter notebook, or AIKIT web interface (Figure 3). The AIKIT web interface database retains query results.

IV. Discussion

A. SQuAD

The DaaDy framework combined with SQuAD for QAR generation resulted in 100% content utilization in large documents, a significant improvement over current methods. As the quality of a question stems directly from the utility of the source context and the studied documents lack an accepted metric for relative or absolute sentence utility, no quantitative data was generated from this study to determine whether the question quality using DaaDy/SQuAD was superior or inferior than single-prompt LLM RAG. While quantitative observations were not produced, there were a number of relevant qualitative assessments made based on the observation of SQuAD QAR-generation. By using a single sentence as the context provided to the LLM RAG, a significant portion of context/background knowledge was removed from the LLM RAG, which may have caused at least four of the seven categories of anomalous QAR generation (unable to answer, repetitive QA, missing context-lists, non-sequitur, and possibly, misleading). Rudimentary trials (data not shown) showed that, generally, when context length was kept to less than 1000 characters, the full context was utilized for QAR generation. Thus, we hypothesize that if the SQuAD method instead of passing a sentence, passed 1000 or less characters that group together coherent sentences, paragraphs, or sections within the DaaDy, the generation of anomalous QARs would decrease while maximizing context utilization.

In the area of SQuAD QAR evaluation, three scenarios were studied. When the answer was contained in the provided context, LLM RAG of the full document performed better at QA than the localized context (72.7% vs. 64%), see Figure 6. Additionally, QA on the localized context reported incorrect absences significantly more than when queried against the full document (24% vs. 6.1%) (Figure 6). When the answer was not contained in the provided context, RAG of the full document produced significantly more false (33.3% vs. 11.8%) and irrelevant (11.1% vs. 0%) responses than querying only the localized context (Figure 6). We also observe that the full-document LLM RAG malfunctioned more than the localized-context LLM RAG, producing RAG errors (11.1%) whereas the localized-context RAG produced none (Figure 6). While the study of answering poorly-phrased questions lacks significant benefit, it is interesting to note that the full-document query produced irrelevant responses (50%), RAG errors (25%), and context regurgitation (25%) responses, while the localized-context query either accurately recognized the vagueness and reported that insufficient context was provided to answer the question (50%), provided a correct but incomplete response (25%), or stated that the answer was not contained in the context (25%). (Figure 6). From this data, we draw the conclusion that an increased quantity of background information permits higher certainty on QA when the answer is contained explicitly in the context. However, when the answer is not contained in the provided context, the presence of extraneous material produces undesirable (irrelevant and false) responses as well as text-generation malfunctions (RAG errors and context regurgitation). Using localized context in these cases produce a more desirable and transparent result.

The use of DaaDy and SQuAD creates a framework where LLM RAG behavior is more predictable and the context utilized can be known with high fidelity. Due to this increase in both transparency and predictability, we assert that LLM RAG can be implemented as a tool to improve human efficiency in knowledge-intensive professions. The importance of **expert supervision and quality assurance** cannot be understated. LLM RAG enhanced with SQuAD and DaaDy can increase efficiency and comprehensiveness are still susceptible to the aforementioned anomalies observed in text generation. Thus, it is absolutely critical that these methods be utilized with appropriate levels of supervision and a framework for quality assurance, else the enormous increase inefficiency could turn into a rapid spread of false information [22].

B. AIKIT UI

Access to LLMs currently is via graphical user interfaces or frequently by developing small Python programs. New interfaces providing LLM RAG capabilities are being rapidly developed. Getting the technical details connected properly is a barrier for many projects to easily access LLM RAG capabilities. The two Python tools docs_to_vs.py and llm_rag_query.py provide configurable access to creating LLM RAG embedded documents and querying them. The Ruby on Rails AIKIT web interface profiles configurable creation and querying of documents in LLM RAG knowledge bases. AIKIT provides web viewing and downloading of knowledge base documents. AIKIT also includes support of test-taking with feedback on test questions to instructors. LLM RAG queries and responses and test question responses for learners are retained in the AIKIT database.

C. Recommendations for Knowledge Base Management

Throughout this research there were numerous roadblocks that, if avoided, will significantly improve or simplify the process by which LLM RAG can be wielded to assist in knowledge-intensive professions. Well-structured documents can make the parsing from text to DaaDy expedient and easy. First, maintaining the master copy of each document in the corpus in a purely text form (void of headers, footers, page numbers, and other formatting characters) will significantly ease the burden on coding and debugging automatic parsers. Using word-processing software that encodes the document structure in text form that can be parsed using regular expressions [23] will simplify the process by which the knowledge can be accessed using LLM RAG. Finally, for professions that generate and maintain QARs, avoiding the following will allow straightforward usage of LLM RAGs for test evaluation: 1) avoid asking vague or open-ended questions, 2) avoid using different verbiage in the question than in the context (e.g. “night” versus “between sunset and sunrise”), 3) avoid referencing the publication title in the question unless that data is included in the prompt.

V. Future Work

The results of this research showed that while there is currently an upper limit to the length of context that can be fully utilized effectively by LLM RAG, there is also a minimum length at which the context is so isolated that its utility decreases to the point of difficulty and inconvenience for the user. In future iterations of SQuAD, research should be pursued to determine the optimal context length and chunk size to maximize effective context utilization. Once these parameters are defined, LLM RAG can be optimized for question generation and evaluation. Improvements to LLM RAG should provide sentence context metadata aligned with the document’s structure.

The prototype for the AIKIT UI, due to its fully offline implementation, has the potential to transition to secure systems. The ability to use AI in querying and updating a vast knowledge base while keeping one’s data and documents secure has enormous potential in many fields with highly-restrictive security requirements.

VI. Conclusion

While the capability of LLMs to produce human-like, accurate, and attributable responses has improved significantly in recent years, LLM RAG utilization of text in long documents is an area in need of improvements; these deficiencies render LLM RAG unsuitable as a tool for professions which require accountable and full utilization of the profession’s knowledge base. The document organization framework, DaaDy, and the querying method, SQuAD, presented in this paper significantly improve the utilization rate of LLM RAG over long documents and provide transparency for QA tasks. By utilizing SQuAD and DaaDy, human expertise and intuition can be enhanced by expedient context-querying and content generation.

Additionally, the AIKIT prototype is a fully-containerized, offline solution which can be easily deployed on laptops, workstations, high-performance computing (HPC) clusters, and cloud solutions. AIKIT can thus provide easy-to-use LLM RAG to a wide audience. AIKIT runs on any platform - from a system on a chip (SOC) [19] to HPC or cloud infrastructure. AIKIT is being released as open source at https://github.com/mit-ll/AIKIT. Please contact the authors with questions, requests, or feedback.

Acknowledgment

This research was facilitated by the Department of the Air Force Artificial Intelligence Accelerator at Massachusetts Institute of Technology (MIT) and MIT/Lincoln Laboratory. The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center teams for providing the HPC resources that were utilized to generate the research results reported within this paper. The authors would also like to acknowledge Jason Williams from MIT Lincoln Laboratory for providing graphic artist support.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” 2021.

[2] K. Wu, E. Wu, A. Cassasola, A. Zhang, K. Wei, T. Nguyen, S. Riantawan, P. S. Riantawan, D. E. Ho, and J. Zou, “How well do llms cite relevant medical references? an evaluation framework and analyses,” 2024.

[3] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, “Lost in the middle: How language models use long contexts,” vol. 12, pp. 157–173, _eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00638/2336043/tacl_a_00638.pdf. [Online]. Available: https://doi.org/10.1162/tacl_a_00638

[4] P. Xu, W. Ping, X. Wu, L. McAfee, C. Zhu, Z. Liu, S. Subramanian, E. Bakhturina, M. Shoeybi, and B. Catanzaro, “Retrieval meets long context large language models,” 2024.

[5] G. M. Kurtzer, V. Sochat, and M. W. Bauer, “Singularity: Scientific containers for mobility of compute.” [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0177459

[6] “Docker: lightweight linux containers for consistent development and deployment.” [Online]. Available: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

[7] "Mistral-7b-instruct-v0.2, 2024." [Online]. Available: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

[8] "Mixtral-8x7b-instruct-v0.1." [Online]. Available: https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

[9] "Python programming language." [Online]. Available: https://www.python.org

[10] "Langchain." [Online]. Available: https://python.langchain.com/v0.1/docs/get_started/introduction

[11] "Faiss ai." [Online]. Available: https://faiss.ai/index.html

[12] "chroma." [Online]. Available: https://www.trychroma.com

[13] "Huggingface all-mpnet-base-v2." [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[14] "Langchain pypdfloader." [Online]. Available: https://api.python.langchain.com/en/latest/document_loaders/langchain_community.document_loaders.pdf.PyPDFLoader.html

[15] "Ruby on rails." [Online]. Available: https://rubyonrails.org

[16] "Ruby programming language." [Online]. Available: https://www.ruby-lang.org/en/

[17] "Gnu parallel." [Online]. Available: https://www.gnu.org/software/parallel/

[18] "Intel xeon gold 6258r processor." [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/199350/intel-xeon-gold-6258r-processor-38-5m-cache-2-70-ghz/specifications.html

[19] "Apple m2." [Online]. Available: https://en.wikipedia.org/wiki/Apple_M2

[20] "Tx-green on top 500 list." [Online]. Available: https://www.top500.org/system/178939/

[21] A. Reuther, C. Byun, W. Arcand, D. Bestor, B. Bergeron, M. Hubbell, M. Jones, P. Michaleas, A. Prout, A. Rosa, and J. Kepner, "Scalable system scheduling for HPC and big data," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 76–92, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731517301983

[22] R. Fernando, "Module 1: Setting the stage," Online Lecture, 2022.

[23] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[24] "Mistral basic rag documentation." [Online]. Available: ttps://docs.mistral.ai/guides/rag/

APPENDIX A
PROMPTS FOR SQUAD GENERATION AND EVALUATION

Prompt for SQuAD QAR Evaluation [24]:

```
prompt_input = """Answer the following question as concisely as possible
based only on the provided context. If you are not sure of the answer,
your answer should be "Insufficient context provided."
Keep your answer short and to the point. Avoid using the words "the context"
in your answer.

<context>
{context}
</context>

Question: {input}"""
```

Prompt for SQuAD QAR Generation:
The input prompt used for this research was built off of previous research from Wu et al. [2]

```
prompt_input = """Answer the following question based only on the provided context:

<context>
{context}
</context>

Question: {input}""")

input = """You are given context that contains information. Generate a
question and answer pair that tests the reader's understanding of the
information contained within the prompt. Each question should provide
sufficient context so that the question does not require the context to be
fully understood. Do not reference the word 'context' in the question or
answer. The question should have a straightforward answer that can be easily
verified with the context. In addition to the question, provide the answer that
is succinctly taken from the prompt.
Provide the question and answer in the following format:
Question: Provide the question here.
Answer: Provide the Answer here. Make the answer as concise as possible."""
```

Appendix B
SQuAD Generation - Anomalous QAR Category Definitions and Examples

1) Unable to Answer - the QAR produced included a question that the context did not answer, to which it correctly answered that the answer was not available in the context provided. Example:
   - Question: "Under what circumstances are runway operations automatically suspended?"
   - Answer: "Runway operations are automatically suspended when not specified in the context. The context does not provide information on the specific reasons for suspension."
2) Repetitive QA - the QAR produced included a question and answer pair where the answer was a part of the question, resulting in either a confusing question or an obvious answer. Example:
   - Question: "In what situation might you need to evacuate an airport runway in an emergency?"
   - Answer: "In an emergency runway evacuation."
3) Unnecessary Justification - the QAR produced included a, generally obvious, justification after the answer despite the fact that it was prompted to avoid doing so and that the context was very short. Example:
   - Question: "Who or what must approve cable certifications?"
   - Answer: "The 20 OG/CC must approve cable certifications. Answer derived from the prompt: 'Cable certifications must be approved by the 20 OG/CC.'"
   - Context: "Cable certifications must be approved by the 20 OG/CC."
4) Missing Context (Lists) - the LLM RAG was prompted on each sentence, resulting in the complete loss of context when the sentence was a part of a list of items or actions. When sub-items in a list were taken out of context, the result was inaccurate and confusing information which could be seriously detrimental to the learner if not caught. Example:
   - Question: "Under what circumstances are pop patterns allowed in this situation?"
   - Answer: "Pop patterns are allowed only if required for syllabus sorties."
   - Localized Context: "No pop patterns unless required for syllabus sorties."
   - List context:
     - "2.4.2. [Bird Watch Condition] Moderate — Low approaches and chase aircraft limited to events that could not be accomplished at other airfields and are required for training or checkrides. No formation approaches."
     - "2.4.2.1. Low Altitude Structure. 1,000 feet AGL minimum, except as required for weapons delivery (defined as within 10 NM of target). If low altitude currency is required for contingency operations or training requirements, minimize time in low altitude structure below 1,000 feet AGL."
     - "2.4.2.2. Poinsett Range. 1,000 feet AGL minimum, except as required for weapons delivery (defined as base position for the curve pattern). **No pop patterns unless required for syllabus sorties.**"
   - Explanation: The question uses the verbiage "in this situation" rather than "during a Bird Watch Moderate condition" which lacks the critical connection between cause and effect for this context.
5) Non-Sequitur - when context was very short and non-descriptive, LLM RAG generated what the expert categorized as accurate but useless QAR. Example:
   - Question: "In what domain does the term 'Airfield Lighting Systems' belong?"
   - Answer: "Airfield Lighting Systems belong to the domain of aviation or airfield infrastructure."
6) Misleading - in an effort to be concise, LLM RAG would produce a QAR that was misleading. Example:
   - Question: "On which Mondays are both runways open for use?"
   - Answer: "Both runways are open on Mondays that are not the 1st, 2nd, 3rd, or 4th Monday of the month."
   - Context: "Runway 04L/22R will close every 1st/3rd Monday and Runway 04R/22L will close every 2nd/4th Monday of the month from 0700L to 1100L for preventative maintenance (PM)."
   - Explanation: The QAR produced indicates that both runways are closed on the 1st, 2nd, 3rd, and 4th Monday of the month; in reality, only one runway is closed between 0700L and 1100L.
7) Acronym Hallucination - in many cases where an acronym was used, LLM RAG hallucinated the full title and provided the incorrect entity in the question and/or the answer. Example:
   - Acronym: SOF
   - Correct Entity: Supervisor of Flying
   - Hallucinated Entity: Senior Operations Force commander

Appendix C
SQuAD Evaluation - Response Category Definitions

1) False Response - the response provided was an incorrect answer to the question.
2) Irrelevant Response - the response provided was not relevant to the question asked.
3) Correct Response - the response provided was assessed as correct by the expert.
4) Correct Absence - the response accurately reported that the answer to the question was not contained within the provided context.
5) Incorrect Absence (false negative) - the response inaccurately reported that the answer to the question was not contained within the provided context, even though it was.
6) Vague Response - the response accurately reported that there was no specific answer to the questio asked.
7) Incomplete response - the response provided was true but was missing some critical information from the same context. Example:
8) RAG Error - the response provided was obviously cut off mid word, acronym, or sentence.
9) Content Regurgitation - the response provided was an excerpt or series of excerpts copied exactly from the context.