FolkRAG: A Retrieval-Augmented Generation System for Cultural Heritage Materials

Paul Kelly (*/ ✉)
Department of Data Science
George Washington University
Washington, DC
pjameskelly@gwu.edu
https://orcid.org/0000-0001-7252-2375

Jonathan Schild
Department of Data Science
George Washington University
Washington, DC
Jschild01@gwu.edu
https://orcid.org/0009-0003-7969-0445

Amir Jafari
Department of Data Science
George Washington University
Washington, DC
ajafari@gwu.edu
https://orcid.org/0000-0002-1730-4807

Abstract
Archival collections, such as those at the Library of Congress (LoC), often suffer from fragmented metadata and disconnected discovery tools, making research navigation challenging. These collections—comprising catalog records, finding aids, and digital surrogates—frequently lack uniform semantic links, forcing researchers to piece together information from disparate sources. Current discovery systems are limited in their ability to support intuitive or exploratory inquiries, requiring users to possess significant technical expertise or rely on librarian assistance. To address these challenges, this study introduces FolkRAG, a proof-of-concept system that leverages Retrieval-Augmented Generation (RAG). RAG is an advanced natural language processing framework that combines large language models with external information retrieval systems, enhancing access to archival materials through natural language queries and citation-supported responses. FolkRAG is designed to query public data from the American Folklife Center (AFC) at the LoC, optimizing vector store parameters and integrating advanced retrieval strategies, including the core principles of Sentence Window Retrieval, Hypothetical Document Embedding, and re-ranking. By bridging intellectual and semantic gaps in archival metadata, FolkRAG provides researchers with a cohesive interface for discovering catalog records, finding aids, and digital objects. This approach not only upholds key values such as reliability, transparency, and contextual accuracy but also redefines archival search as a more intuitive and exploratory experience. By showcasing how metadata integration from diverse data sources is used to construct a vector database that powers advanced RAG systems, FolkRAG demonstrates the potential to improve access to cultural heritage materials for both novice and expert users without sacrificing archival practices.

Keywords
Retrieval-Augmented Generation (RAG), Cultural Heritage, Digital Archives, Natural Language Processing, Document Retrieval, Vector Embeddings, Sentence Window Retrieval, Hypothetical Document Embedding (HyDE), Re-ranking, Metadata, Archival Access Systems, Information Retrieval, Library and Information Science, Document Chunking, Knowledge Management, Large Language Models (LLM), Digital Libraries, Archival Science, Folklife, Ethnography, Oral History

1 Introduction

A single archival collection is often described and presented online using diverse metadata and schemas stored in various locations. While intellectually connected, these may lack semantic linkage. Elements of that collection are often displayed online in the form of digital surrogates that utilize additional technical and descriptive information that, again, may or may not link back to a central document that allows a researcher to make sense of the material.

The collections at Library of Congress (LoC), and specifically the digital representations of those of the American Folklife Center (AFC) upon which this experiment centers, adhere to this. A collection is typically represented by a catalog record which may or may not link to a finding aid, if it has one; the finding aid may or may not link back to the catalog record; there may be digital versions of objects from the collection (image, text, audio, or video) online that are sometimes, but not always, linked from the catalog record or finding aid; those objects have metadata that mostly, but not uniformly, link back to the previously mentioned sources.

It is both challenging and time-consuming to search all these data sources simultaneously, as Yakel and Duff highlighted in the early years of information retrieval [1, 2]. A researcher must rely on their own facility with archival research to successfully navigate a collection, and, when necessary, on the guidance of reference librarians who sometimes know these collections intimately, but often do not. Researchers and staff alike rely on these systems of description to guide them. It can be challenging to interact with these discovery systems in a way that accommodates half-remembered details or allows for threads that one might pull to create a more natural or esoteric entry point into a subject area. One cannot ask questions without in-depth knowledge of how these systems work, which few people truly possess. To address these challenges, computational advancements like Retrieval-Augmented Generation (RAG) offer promising solutions, integrating natural language processing with targeted retrieval mechanisms. The aim of this project, therefore, is to build a system that addresses these concerns computationally without abandoning the core tenets of librarianship - service, accountability, context, and authority.

RAG represents a significant advancement in natural language processing, combining large language models (LLMs) with targeted information retrieval capabilities [3]. At their foundation, RAG systems supplement LLMs with external data sources stored in vector databases, enabling them to overcome limitations like hallucinations and static training data [4, 5]. The core architecture comprises a retriever that locates relevant information from the vector database and a generator that incorporates this information to produce accurate, contextually-rich responses [6, 7].

The retrieval component can operate in either dense or sparse vector spaces, with dense embeddings capturing more nuanced semantic relationships between words and phrases [6, 8]. The choice of embedding model significantly impacts system performance [9]. For instance, Instructor-XL offers superior understanding of nuanced language but requires substantial computational resources, while AWS Bedrock's Titan model enables faster processing at scale despite potential challenges with high-volume API calls.

The evolution of RAG systems has progressed through three distinct paradigms, as outlined by Gao et al. [10]. The initial Naive RAG paradigm, introduced by Lewis et al. [3], established the foundational "Retrieve-Read" framework. While groundbreaking, this approach faced limitations in retrieval precision and potential hallucination issues when handling irrelevant information, as demonstrated by Borgeaud et al. in their work with large-scale document collections [11].

Advanced RAG represents a significant evolution, introducing sophisticated optimization strategies [10]. One key breakthrough is Sentence Window Retrieval, which has shown that smaller chunking strategies during the construction of the RAG's vector database can improve retrieval [12]. Another significant advancement is Hypothetical Document Embedding (HyDE), introduced by Gao et al. in 2022 [13], which uses LLMs to generate hypothetical answers to input queries with the goal of improving retrieval accuracy by supplying the LLM with additional relevant context. The incorporation of re-ranking in advanced RAG has also seen substantial development [14, 15]. Various approaches including cross-encoders, multi-vector models, and LLM-based re-rankers have been implemented to improve retrieved document relevance. Notably, Cuconasu et al. [16] demonstrated that random or noisy documents, when strategically positioned in the context, can sometimes improve rather than degrade LLM performance.

This study employs RAG systems to address the limitations of fragmented metadata and disconnected discovery tools prevalent in extensive archival collections, such as those at the LoC. A vector database is designed to address the lack of standardization and the specialized nature of metadata across multiple datasets, which can hinder effective data discovery and RAG retrieval strategies [17]. By integrating advanced retrieval methodologies with language generation capabilities, the proposed system offers a cohesive framework designed to enhance the intuitiveness and efficiency of research navigation within these collections. For example, a researcher exploring American Folklife Center collections could query the system with vague recollections like 'folk songs from the 1930s about mining,' and the system would retrieve relevant digital objects, catalog records, and finding aids, presenting them in a cohesive, easily navigable format. This approach not only aligns with the core principles of librarianship—such as service, accountability, and authority—but also reimagines how computational tools can support exploratory and esoteric inquiries [12].

Through the use of advanced RAG techniques, including Sentence Window Retrieval, Hypothetical Document Embedding, and optimized re-ranking strategies, this project demonstrates the potential for bridging intellectual and semantic gaps in archival metadata. By providing researchers with contextually enriched, accurate responses drawn from diverse data sources, this system redefines archival search, ensuring that users can engage with collections regardless of their technical expertise or prior familiarity [18]. Unlike generic RAG systems, this project is designed to integrate RAG with archival metadata schemas and discovery tools, tailoring its capabilities to the unique requirements of exploratory archival research and esoteric inquiry. This work underscores the transformative role of RAG in making archival materials more accessible, interconnected, and meaningful. It highlights the potential for integrating RAG into other domains of cultural heritage and academia, paving the way for even more intuitive, interconnected, and meaningful exploration of historical and cultural records.

2 Proposed Solution

2.1 Data Collection and Metadata

Note - All computing was performed on Amazon Web Services (AWS) g5.2xlarge Ubuntu Elastic Compute (EC2) instances provided by The George Washington University (GWU).

AFC, founded in 1976 when Congress passed the American Folklife Preservation Act, has fulfilled its charge to preserve and present folklife in all its diversity, and documents and shares the many expressions of human experience to inspire, revitalize, and perpetuate living cultural traditions. It houses the Archive of Folk Culture, which was originally founded as the Archive of American Folk Song in 1928. Its holdings include photographs, audio and video recordings of oral history interviews and folksong, correspondence, and manuscript materials such as ethnographic field notes and logs in both analog and digital formats.

The catalog is typically the first place a public record of a library resource is created. It contains high level bibliographic information about an item, and allows that information, whether it pertains to a book, archival collection, or other format, to be located. The record is created by a cataloger using the Machine Readable Cataloging (MARC) format and is available as MARC XML. A bibliographic item is typically represented by a single catalog record.

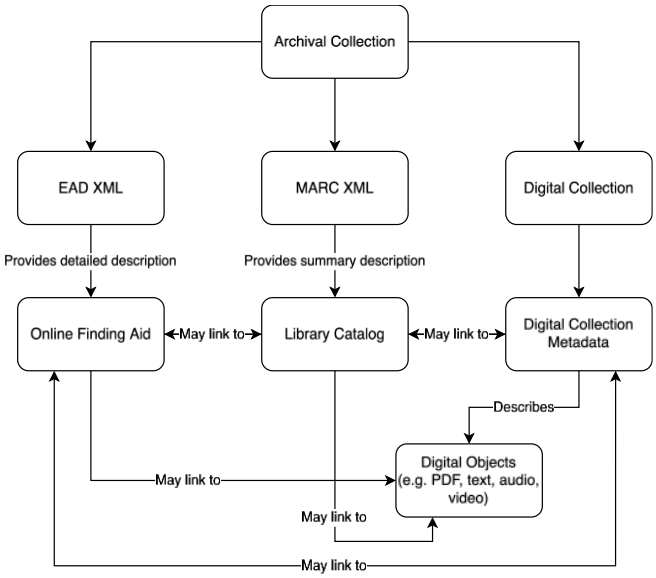

Fig. 1. Diagram of relationships between data and metadata

Archival finding aids describe, in a broad sense, what a collection contains - if physical, its container list will document each box, folder, and sometimes item; if born-digital, its container list will document whatever arrangement has been imposed on the digital materials; if both, then a combination of each approach is used. Finding aids also provide contextual information about the record creators, related materials, provenance, and how the collection has been arranged, why, and by whom. They also utilize controlled vocabularies such as subject headings to provide researchers with a general impression of the topics covered in the collection. They take the form of structured XML known as Encoded Archival Description (EAD). An archival collection will typically be represented by a single EAD, written by an archivist once a collection has been processed and is available for research.

For our project, we wrote a web scraper to obtain individual EAD XML and MARC XML records for all AFC

collections and utilized the LoC API to pull AFC digital collection JSON metadata at the file level and transform it into .csv. Since that metadata contains file locations and mime-types, we then structured and executed wget requests to download all AFC .txt, .pdf, .mp3, and .mp4 files. We extracted all .pdf text with tesseract optical character recognition (OCR) and transcribed (and, where appropriate, translated into English) all .mp3s with OpenAI's large Whisper speech recognition model. We scraped 158 EAD XML files, 158 MARC XML files, and obtained metadata for 48 digital collections. Our wget pulls followed by either OCR or Whisper transcription resulted in acquisition of 10,980 machine generated transcripts, 4,292 OCR files, and 29,778 library-created text files for a total of 45,050 documents and corresponding metadata records, representing the entirety of AFC's publicly available digital collections, as of October 2024.

Our LoC API queries resulted in two .csv files for each collection - search_results.csv and file_list.csv. Each request is based upon a search, and each search returns a list of resources and their descriptive metadata - this is returned as JSON and saved as a .csv. Each resource can contain any number of derivative files, and search_results.csv contains links to the item pages where those files are available. This information is used in a second API call to the item pages to generate file_list.csv - the list of every file for every resource in the search, its type, and its location.

The metadata processor takes a modular approach to integrating diverse archival description sources into a unified whole. It extracts and normalizes identifiers from URLs in various formats through pattern matching, generates potential matching patterns for different resource types, and implements thorough error handling and logging. For EAD and MARC, the processor parses hierarchical elements like title, date, abstract, subject headings, and series titles, and preserves the contextual information these sources provide at the collection level. The system maps related fields across schemas while maintaining source attribution, addressing the key challenge of disconnection between finding aids, catalog records, and digital objects.

The main processing function first parses the search_results.csv to create multiple mapping dictionaries that associate different forms of identifiers (base IDs, digital IDs, and resource URLs) with their corresponding metadata. It then processes the file_list.csv to establish direct file-to-identifier mappings. For each file encountered, the processor attempts multiple matching strategies: first trying an exact match through file_list mappings, then attempting matches through the various identifier dictionaries if the direct approach fails.

The final metadata schema incorporates fields for identification (original filename, file type, chunk id, total chunks, call number), descriptive information (title, date, created published, language, type), authorship and custody (contributors, creator, repository, collection, source_collection), content description (description, notes, subjects, original format, online formats), rights management (rights, access restricted), geographic coverage (locations), resource location (url), as well as specialized fields from finding aids and catalog records. Array fields like contributors, notes, and subjects are stored as JSON strings to preserve their multi-valued nature while maintaining compatibility with Deeplake's data structure.

2.2 Initial Approach and Vector Database Development

In the initial development stages, we systematically explored multiple technical approaches while addressing increasingly complex requirements for processing archival materials. Our early experiments focused on identifying optimal solutions for embedding model selection and vector storage implementation. This collection encompassed a wide range of document types, presenting distinct processing requirements and metadata relationships that needed to be preserved throughout the retrieval process.

Our initial vector storage implementation revealed significant technical challenges that shaped the evolution of our system architecture. We began with Facebook AI Similarity Search (FAISS), attracted by its reputation for fast similarity search operations and efficient GPU support. Initial performance testing showed promising results, with FAISS demonstrating high-speed processing capabilities for collections ranging from 50,000 to 100,000 documents. However, as we progressed with implementing detailed archival metadata integration, FAISS's limitations became increasingly problematic. The system struggled to efficiently manage the complex hierarchical collection information, creator attribution details, and temporal metadata that are crucial in archival contexts.

These limitations led to a pivotal shift to Deeplake as our vector storage solution. Deeplake offered a more comprehensive framework for handling both document embeddings and complex metadata structures within a unified system. While this transition initially required additional development effort to migrate our existing implementation, it ultimately provided a more robust foundation for our RAG system. Deeplake's integrated approach eliminated the need for parallel metadata management systems, simplifying our architecture while maintaining the rich contextual information necessary for effective document retrieval.

This architectural pivot proved especially valuable as our collection grew beyond 100,000 documents. Deeplake's integrated approach to handling both embeddings and metadata enabled more sophisticated query operations that could simultaneously consider semantic similarity and archival context. The system demonstrated superior capability in maintaining the complex relationships between documents, their descriptions, and their place within larger archival hierarchies, while providing consistent query performance across our expanding collection.

The vector store utilizes DeepLake and LangChain as the underlying database technology, chosen for their ability to handle the scale and complexity of AFC's digital collections while maintaining efficient similarity search capabilities and metadata-based filtering. The system offers multiple embedding model options: HuggingFace's Instructor-XL, MiniLM-L6-v2, and Amazon's Titan embedding model. Each model presents different tradeoffs - Instructor-XL excels at understanding nuanced language and context but requires more computational resources, while Titan leverages AWS Bedrock's infrastructure for potentially faster processing at scale.

The vector store initialization process creates a comprehensive tensor structure that preserves both the textual content and its associated metadata. Each document is

represented by a set of tensors including the raw text, embeddings, and all metadata fields established during processing. The system implements memory management through batch processing, with configurable batch sizes (defaulting to 100 documents) to handle the large volume of digital collection material efficiently.

Document chunking is performed using LangChain's recursive character text splitter with configurable chunk size and overlap percentage (defaulted to 15% overlap between chunks), preserving document boundaries and maintaining contextual coherence, similar to the Sentence Window Retrieval technique. For transcript files, the chunking process includes additional preprocessing to remove timecode annotations while preserving the temporal relationship between text segments.

The embedding generation process incorporates several safeguards to ensure data integrity:

- A checkpointing system tracks progress and enables recovery from failures.
- Detailed logs of the embedding process track successful operations and document anomalies.
- Error handling at multiple levels, from individual document processing to batch operations.
- Memory optimization techniques including garbage collection and CUDA memory cache clearing.

3 Model

Three architectures were developed and implemented, expanding from a naïve RAG system to incorporating advanced methods, like hypothetical document embedding strategies and re-ranking. The naïve RAG (Fig. 2) architecture receives the input query from the user and then embeds it using the same model that was used to construct the vector store. The cosine similarity then measures the semantic similarity between the query vector and stored vectors by calculating the dot product of vectors divided by the product of their magnitudes. The top_k documents are identified and re-ordered according to their cosine similarity and the most relevant top_k documents are sent to the LLM to generate a response.

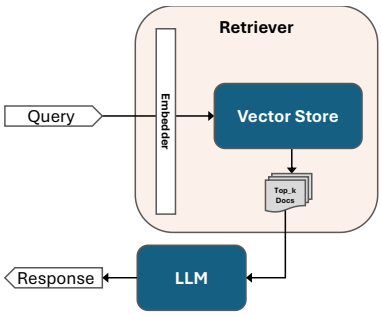

Fig. 2. Diagram of the naïve RAG system

The generator combines the retrieved documents' content into a unified context variable and employs a structured prompt template for the LLM. Included are the query and relevant details, metadata to provide to the user for reference, and explicit instructions on what to include and exclude in the output. For token length efficiency and to avoid mixing context with unmatching metadata, only the best-scoring document's metadata is used in generating a response to the user's original query.

Example prompt for our RAG system generators:

Human: Please answer the following query based on the provided context and metadata.

Query: [User Query]

Context: [Relevant text from top-ranked documents]

Metadata: [Relevant metadata from top-ranked document]

Instructions:

1. Answer the question using ONLY the information provided in the Context and Metadata above.

2. Do NOT include any information that is not explicitly stated in the Context or Metadata.

3. Begin your answer with a direct response to the question asked.

4. Include relevant details from the Context and Metadata to support your answer.

5. Pay special attention to the recording date, contributors, and locations provided in the metadata.

6. Inform the user of what document filename they can find the information in.

Two advanced RAG systems were developed to measure the degree of change retrieval accuracies between basic naïve architectures and the more advanced systems. The first (Fig. 3) incorporates a generic hypothetical document embedding mechanism ahead of the document retrieval, similar to that described during the literature review. The LLM receives the input query, generates a hypothetical response to it, combines the two into a single hypothetical document that is then embedded and sent through the same retrieval and response generation procedures. To ensure factual integrity and deter hallucinated text from being incorporated into the RAG pipeline, the temperature is suppressed to 0.7 during HyDE generation.

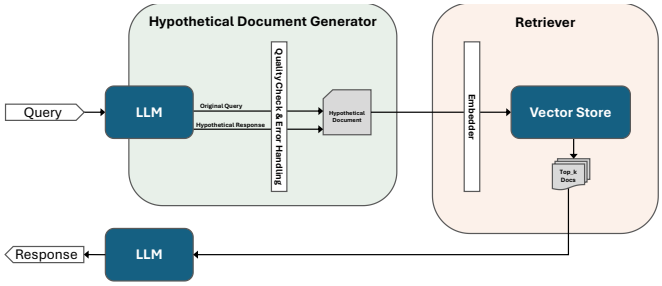

Fig. 3. Diagram of the RAG system using basic HyDE

The final novel RAG system (Fig. 4) implements a similar but enhanced HyDE mechanism before retrieval and incorporates a re-ranking mechanism after retrieval, hereafter

referred to as HyDER. Instead of a single hypothetical document, the HyDE generator uses the original input query to generate two derivative queries that are similar but different, as well as a response to each of the three queries to form three independent hypothetical documents. Progressively higher temperatures (i.e., 0.7, 0.8, 0.9) during the formation of each hypothetical document ensures a degree of variety, and the query-response prompts sent to the LLM during HyDE generation are held equal, including across all systems.

Derivative query generation prompt:

> *Rewrite this query to be slightly different but similar in meaning: {query}*

Hypothetical response generation prompt:

> *You are a document that answers this question {query}.*
>
> *Write a short, natural paragraph that directly answers this question. Include additional relevant information if possible.*

All of the generated content is passed through a cleaner to compensate for instances when the LLM does not generate adequate output, or if the output includes language not relevant to the objective. For example in some cases, the LLM refrains from forming a response out of its perceived copyright concerns. In such instances, valid output is taken from the other hypothetical documents to assure each hypothetical document is complete for onward passing through the RAG pipeline. Since retrieval is accomplished for each hypothetical document separately, three times the original top_k documents are collected. All of the results are pooled and duplicate documents are dropped since there is some presumed overlap in retrieved documents.

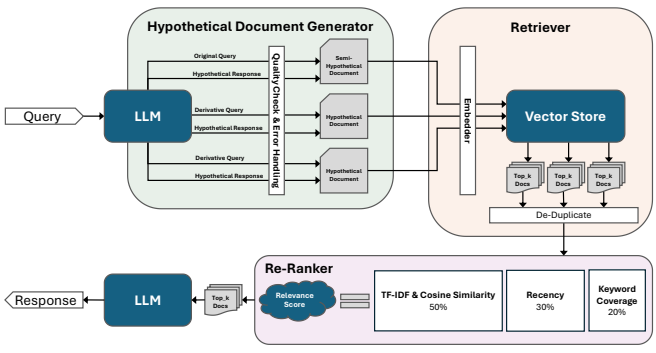

Fig. 4. Diagram of the RAG system implementing enhanced HyDE and re-ranking (HyDER)

A re-ranking mechanism is used to reassess the relevance of all the remaining retrieved documents to the original input query. Their relevancy scores are calculated using a weighted sum of the TF-IDF cosine similarity scores as calculated using the Python package sklearn, how recent the retrieved document is dated, and the query's keyword prevalence within the retrieved document. With the retrieved documents re-ordered, those exceeding the top_k value are dropped while the remainder are sent to the LLM for generating a response to the query. Complete calculations during re-ranking to derive relevance are as follows:

> *Total Score = (TF-IDF Score x 0.5) + (Freshness Score x 0.3) + (Keyword Coverage x 0.2)*
>
> *TF-IDF Score = Cosine Similarity (TF-IDF of Query, TF-IDF of Document)*
>
> *Freshness Score = (0, 1 - Time Different in Days/365)*
>
> *Keyword Coverage = Number of Matching Query Keywords in Document/Total Unique Keywords in Query.*

4 Results and Discussion

4.1 Evaluation

Initial testing employed a small data sample to evaluate different system configurations. Vector stores were built in chunks of 250, 500, 1,000, and 2,000 characters and examined using three different models. We tested both Naïve and HyDE architectures to gauge the efficacy of HyDE as a RAG strategy. Each system configuration was evaluated using 100 questions, with each question paired with the unique filename of the document containing its correct answer. Accuracy was measured based on whether the correct document appeared among the top_k filenames returned by the retriever.

The preliminary results (Fig. 5) confirmed several key findings. The MiniLM-L6-v2 embedder and Naïve architecture demonstrated inadequate performance. In contrast, the basic HyDE architecture showed significant improvements in retrieval accuracy, generally increasing it between ten and twenty percentage points across most system configurations. Notably, the vector store constructed using embedding chunk sizes of 250 characters, despite being a particularly short length of text, performed comparably to the 1,000-sized chunks for optimal vector store performance. The most promising system architectures emerged as those combining the basic HyDE generator with either Instructor-XL or Titan as the embedding model, constructing vector stores in chunk sizes of 250 or 1,000. In testing on the sample set, these configurations achieved 90% retrieval accuracy on the 100 sample test questions within a top_k of four documents.

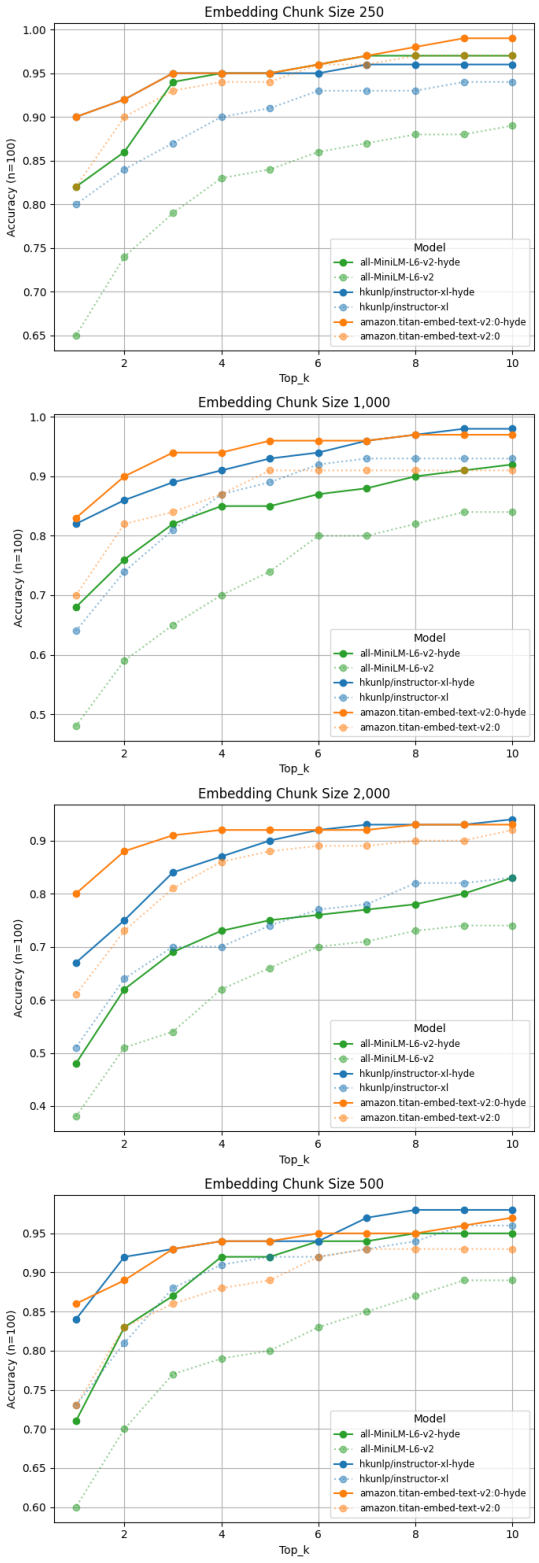

Fig. 5. Initial retrieval evaluations on testing subset

Based on these initial findings, evaluation was expanded to the entire dataset. New vector stores were constructed based on 250 and 1,000 chunking strategies which contained 1.6 million and 400,000 million chunked documents, respectively. We tested system performance using 118 new questions to gauge retrieval accuracy, and the MiniLM-L6-v2 embedder was dropped from consideration. The evaluation included comparisons of the Naïve and HyDE architectures, along with the newly introduced enhanced HyDER architecture.

While both Instructor-XL and Titan embedding models showed initial promise, we conducted detailed accuracy comparisons to determine the superior option. Under the enhanced HyDE architecture, testing with top_k values of one and three on vector stores chunked by 250-character blocks revealed significant performance differences. The Titan-generated embeddings achieved accuracies of 45% and 59% for top_k values of one and three, respectively. In comparison, the Instructor-XL generated embeddings demonstrated superior performance, achieving accuracy of 61% and 67% for the same top_k values. Based on these results, we discontinued use of the Titan embeddings and conducted all subsequent RAG system evaluations using the Instructor-XL model as the embedder.

Final evaluations (Fig. 6) revealed close performance between the HyDE and HyDER architectures, with HyDER showing a slight edge. The vector store constructed using 250-character sized chunks consistently provided better results than larger chunk sizes. Both architectures achieved 90% accuracy at or just beyond a top_k value of 15 when using 250 chunk-sized vector stores, with performance plateauing at approximately 94% after a top_k of 25.

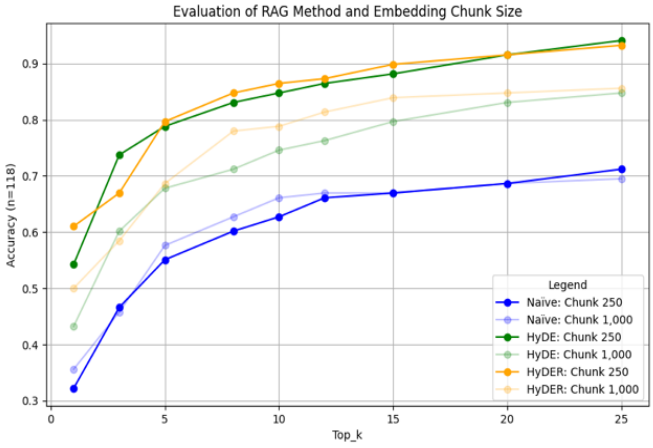

Fig. 6. Final retrieval evaluations on all data

Furthermore, a thorough examination of specific query results provided clear evidence of the impact of chunk size on retrieval accuracy. The results for the following query with the top_k value set to 1 demonstrate the superior performance of the 250-character chunk size in retrieving the correct source material:

> *"What ingredients did the Vietnamese interviewee say go into pho?"*

The 250-character chunks correctly retrieved the source document (mb_r019_01_en.txt) with a similarity score of 0.8395. The retrieved text contained from within the chunked document:

They call it pho.

That's what they call it in my country, pho.

Ph-o.

Those are noodles?

Yes.

Noodles, they put some soup in there, some beef, some vegetables, something.

That's a famous food in my country.

Do you prepare that yourself at home?

Yeah.

In contrast, the 1,000-character chunks retrieved an incorrect document (afc1987042_mb_r010_01_en.txt) with a similarity score of 0.8532. While this document discussed pho, it contained a different speaker's general description rather than the Vietnamese interviewee's direct response:

Thin noodles.

Thin noodles.

Yeah.

You could do that on a thin yellow noodle.

Then it would be called bra mee leung lat na in the Thai language.

Very popular dish throughout all of China, all of Taiwan, Korea, Malaysia, Thailand, Laos, Cambodia, Vietnam, Burma is a thing called, the Vietnamese call it pho.

[...]

It's a beef broth soup.

Bean sprouts in it.

Thin rice noodles or wheat noodles.

Some fried garlic that we fry up ahead of time and have it sitting aside and just put it into the soup.

Would have scallions, coriander, the meat of your choice or shrimp and squid.

When examining retrieval results using the same question but with a top_k value of 5, we observed a pattern of duplicate retrieval:

250 Chunked vector store using Instructor-XL:

- afc1987042_mb_r019_01_en.txt (retrieved 3 times, correct document)
- afc1987042_mb_r010_01_en.txt (retrieved 2 times, similar but incorrect document)

1,000 Chunked vector store using Instructor-XL:

- afc1987042_mb_r010_01_en.txt (retrieved 3 times, similar but incorrect document)
- afc1987042_mb_r010_r011.txt (retrieved 2 times, incorrect document)

4.2 Generation

A review of the generated content for two selected test questions demonstrated precise and coherent outputs. The system provided accurate responses to the queries along with specific references to the source documents. For instance, it correctly identified the file containing the answer and delivered a contextually accurate response using a 250-chunk vector database with a top_k value of five. For comparison, the same question and parameters performed using the 1,000-chunk vector database provided a similar contextual response, but returned an unexpected filename as the source document.

Query:

What is Captain Pearl R. Nye's favorite ballad?

HyDER Response (250):

Captain Pearl R. Nye's favorite ballad is Barbara Allen. This information can be found in the context, where Captain Nye states, "I guess Barbara Allen is my favorite of them all." The recording was made on June 27, 1937 in Akron, Ohio, as part of a field interview conducted by John A. Lomax. This information is part of a sound recording titled "Two Sailors: Sea Shanties and Canal Boat Ballads," which is one of 10 radio programs in "The Ballad Hunter" radio series. The original filename for this information is ***sr28a_en.txt****.*

HyDER Response (1,000):

*Captain Pearl R. Nye's favorite ballad is Barbara Allen. This information can be found in the file "****afc1933001_ms403_002.txt****". The context explicitly states: "I guess Barbara Allen is my favorite of them all." This quote is attributed to Captain Nye in the*

passage. Additionally, the context mentions that the Library of Congress has eighty-three versions of Barbara Allen, further emphasizing the popularity and significance of this ballad.

The metadata indicates that this information comes from the John A. Lomax and Alan Lomax papers, specifically from a manuscript titled "Adventures of a Ballad Hunter." The document was created between 1907 and 1969 in the United States, though Captain Nye's exact recording date is not specified in the given information.

4.3 Discussion

This detailed example analysis reveals several significant findings. The 250-character chunks proved more effective at pinpointing exact relevant content, even though the 1,000-character chunk retrieved text had a slightly higher similarity score. The phenomenon of the same document appearing multiple times in the results indicates that different chunks from the same source document were independently deemed relevant. While this redundancy might appear inefficient, the different chunks likely contained distinct contextual information that could be valuable for response generation. The 1,000-character chunks' tendency to retrieve thematically related but incorrect documents suggests that larger chunks may sometimes obscure specific relevant content within broader contextual information.

The close performances between the HyDE and HyDER systems suggest that the significantly more complex HyDER architecture may not be necessary to achieve optimal results. Basic HyDE mechanisms in most instances proved sufficient to drastically improve retrieval accuracies. However, our analysis of wrong retrieval samples revealed that the initial database construction or the de-duplication process might have been ineffective in some instances. The top_k documents passed to the re-ranker sometimes sourced back to the same document filename, likely because single documents were broken up into chunks while retaining their source filename.

Although filenames may have been duplicated, the differing context for each instance likely minimized any impact on retrieval accuracy. Enforcing unique documents in the top_k results could improve accuracy for top_k > 1 but might also exceed top_k limits or omit relevant context to prioritize unique filenames. An alternative solution might be generating unique IDs for each chunk to help identify them distinctly. The exact effect on accuracy metrics remains unclear and requires further investigation.

Comparison of the generated content for a small sample of test questions showed that while the differently chunked vector databases both yielded sufficient contextual responses, the larger one failed to return the correct filename. This confirmed prior research regarding Sentence Window Retrieval, the effectiveness of overlapping chunks, and that better performance on smaller chunked vector databases. An in-depth analysis is required to identify the precise cause of this difference.

5 Conclusion

FolkRAG demonstrates that RAG systems can effectively enhance access to complex archival collections while maintaining the integrity of archival description and context. The system's success in handling diverse document types and complex metadata relationships suggests promising applications for other cultural heritage institutions facing similar challenges in making their collections more accessible through natural language interaction.

The vector store constructed with 250-character chunks using the Instructor-XL embedding model provided the best balance of performance and efficiency. While both 250 and 1,000-character chunk sizes showed promise during initial testing, the smaller chunk size consistently delivered superior results when scaled to the full dataset. This finding suggests that finer granularity in text segmentation may better serve the nuanced nature of archival materials.

The implementation of comprehensive metadata processing proved crucial to the system's success. By preserving the complex relationships between documents, their descriptions, and their place within larger archival hierarchies, FolkRAG maintains the contextual integrity that is fundamental to archival research. This approach demonstrates that RAG systems can successfully bridge the gap between traditional archival description and modern natural language interaction while upholding professional standards of librarianship.

The integration of advanced RAG techniques demonstrated a significant improvement over traditional naïve systems, highlighting the transformative potential of sophisticated methodologies in information retrieval and generation. By leveraging enhanced mechanisms, such as the expansion of HyDE to generate multiple documents and the implementation of robust re-ranking strategies, the more complex techniques for advanced RAG were shown to frequently outperform their generic counterparts; i.e., of the two advanced RAG systems evaluated, the one with more complex components often yielded better results, albeit close in comparison. These findings underscore the importance of innovative approaches in maximizing the accuracy, relevance, and utility of RAG frameworks, setting a strong foundation for future advancements in the field.

The potential issue of duplicate document retrieval warrants further investigation. While our system achieved high accuracy rates, the tendency to retrieve multiple chunks from the same source document could be addressed through improved de-duplication strategies or by implementing unique document identifiers for each chunk. This refinement could potentially improve both retrieval accuracy and the diversity of sources presented to users.

This research contributes to both the technical advancement of RAG systems and the practical application of AI in cultural heritage contexts. It demonstrates that careful attention to domain-specific requirements and metadata relationships can result in systems that not only improve access but also maintain the professional standards and contextual richness essential to archival research. Future work could explore the integration of multimodal content, enhanced metadata filtering capabilities, and more sophisticated re-ranking strategies that better account for archival hierarchies and relationships between collections.

6 Statements and Declarations

6.1 Competing Interests

An author of this paper is an employee of Library of Congress. All data for this project was acquired legally via public sources; access to internal Library of Congress data was neither requested nor required. FolkRAG is not an official product of nor endorsed by Library of Congress.

6.2 Data Availability

The data that support the findings of this study will be made publicly available upon review by Library of Congress. The data are, however, available from the authors upon reasonable request.

6.3 Code Availability

Code is available from https://github.com/darkivist/FolkRAG.

References

1. Yakel, E. (2004). Encoded Archival Description: Are Finding Aids Boundary Spanners or Barriers for Users? *Journal of Archival Organization*, *2*(1–2), 63–77. https://doi.org/10.1300/J201v02n01_06
2. Duff, W.M., & Johnson, C.A. (2002). Accidentally Found on Purpose: Information-Seeking Behavior of Historians in Archives. *The Library Quarterly, 72*, 472 - 496. https://doi.org/10.1086/lq.72.4.40039793
3. Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems* 33 (2020): 9459-9474. https://doi.org/10.48550/arXiv.2005.11401
4. Veturi, Sriram, et al. "RAG based Question-Answering for Contextual Response Prediction System." *https://doi.org/10.1145/nnnnnnn.nnnnnnn*
5. Guu, Kelvin, et al. "Retrieval augmented language model pre-training." *International conference on machine learning*. PMLR, 2020. https://doi.org/10.48550/arXiv.2002.08909
6. Karpukhin, Vladimir, et al. "Dense passage retrieval for open-domain question answering." (2020). https://doi.org/10.48550/arXiv.2004.04906
7. Izacard, Gautier, and Edouard Grave. "Leveraging passage retrieval with generative models for open domain question answering." (2020). https://doi.org/10.48550/arXiv.2007.01282
8. Lin, Jimmy, Rodrigo Nogueira, and Andrew Yates. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature, 2022. https://doi.org/10.48550/arXiv.2010.06467
9. Humeau, Samuel, et al. "Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring." (2019). https://doi.org/10.48550/arXiv.1905.01969
10. Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." *https://doi.org/10.48550/arXiv.2312.10997*
11. Borgeaud, Sebastian, et al. "Improving language models by retrieving from trillions of tokens." *International conference on machine learning*. PMLR, 2022. *https://doi.org/10.48550/arXiv.2112.04426*
12. Eibich, Matouš, Shivay Nagpal, and Alexander Fred-Ojala. "ARAGOG: Advanced RAG Output Grading." (2024). https://doi.org/10.48550/arXiv.2404.01037
13. Gao, Luyu, et al. "Precise zero-shot dense retrieval without relevance labels." *https://doi.org/10.48550/arXiv.2212.10496*
14. Dong, Jialin, et al. "Don't Forget to Connect! Improving RAG with Graph-based Reranking." https://doi.org/10.48550/arXiv.2405.18414
15. Kim, Kiseung, and Jay-Yoon Lee. "RE-RAG: Improving Open-Domain QA Performance and Interpretability with Relevance Estimator in Retrieval-Augmented Generation." *https://doi.org/10.48550/arXiv.2406.05794*
16. Cuconasu, Florin, et al. "The power of noise: Redefining retrieval for rag systems." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024. https://doi.org/10.1145/3626772.3657834
17. Hayashi, Teruaki, et al. "Metadata-based Data Exploration with Retrieval-Augmented Generation for Large Language Models." https://doi.org/10.48550/arXiv.2410.04231
18. Sawarkar, Kunal, Abhilasha Mangal, and Shivam Raj Solanki. "Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers." https://doi.org/10.48550/arXiv.2404.07220