

Data Science Program

Capstone Report - Winter 2024

Our Wonderful Project

Jonathan Schild,

Paul Kelly

supervised by

Amir Jafari

Abstract

Retrieval-Augmented Generation (RAG) systems represent a cutting-edge innovation in natural language processing, combining generative language models with external information retrieval to enhance accuracy and contextual relevance. This study introduces *FolkRAG*, a proof-of-concept system designed to query public data from the American Folklife Center (AFC) at the Library of Congress, addressing challenges posed by fragmented and inconsistent metadata across archival collections. By optimizing vector store parameters and implementing advanced RAG retrieval strategies using hypothetical document embeddings and re-ranking, *FolkRAG* demonstrates the potential to improve access to cultural heritage materials, enabling natural language queries and coherent, citation-supported responses while upholding the core tenets of librarianship.

Contents

[1 Introduction 2](#_Toc184571324)

[2 Problem Statement 3](#_Toc184571325)

[3 Literature Review 3](#_Toc184571326)

[4 Methodology 4](#_Toc184571327)

[4.1 Data Aggregation 5](#_Toc184571328)

[4.2 Initial Approach 6](#_Toc184571329)

[4.3 Vector Store 7](#_Toc184571330)

[4.4 RAG Architectures 8](#_Toc184571331)

[5 Results and Conclusion 12](#_Toc184571332)

[6 Future Work 16](#_Toc184571333)

[7 Bibliography 19](#_Toc184571334)

# Introduction

A single archival collection can be, and often is, described and presented online in a variety of ways that incorporate different metadata and metadata schema housed in different locations that, while connected intellectually, may or may not be connected semantically. 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 in on LoC’s website, 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) in loc.gov/collections 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. Confused yet? Researchers (and indeed staff) often are.

# Problem Statement

It is impossible to search all of these data sources at once. 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. There is also no way to interact with these discovery systems in a way that accounts for half-remembered details, threads upon which an individual can pull for 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. 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. In short, to allow natural language query and return natural language answers with citation via Retrieval Augmented Generation (RAG) and in doing so develop a methodology to determine vector store parameters that optimize document retrieval accuracy for cultural heritage materials.

# Literature Review

RAG represents a significant advancement in natural language processing, combining large language models (LLMs) with targeted information retrieval capabilities. 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 (Veturi et al., 2024). 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.

The retrieval component can operate in either dense or sparse vector spaces, with dense embeddings capturing more nuanced semantic relationships between words and phrases. The choice of embedding model significantly impacts system performance. 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. (2023). The initial Naive RAG paradigm, introduced by Lewis et al. (2020), 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. (2022) in their work with large-scale document collections.

Advanced RAG represents a significant evolution, introducing sophisticated optimization strategies. A key innovation is Sentence Window Retrieval, which decouples embedding and synthesis processes. Yang (2023) demonstrated how this technique uses smaller text units for retrieval while maintaining broader context for generation. Another significant advancement is Hypothetical Document Embedding (HyDE), developed by Gao et al. (2022), which generates hypothetical answers using LLMs to improve retrieval accuracy through answer-to-answer comparison. Pal (2023) further validated this approach's effectiveness in reducing hallucinations.

The reranking component of Advanced RAG has seen substantial development, as detailed by Bhavsar (2023). Various approaches including cross-encoders, multi-vector models, and LLM-based rerankers have been implemented to improve retrieved document relevance. Notably, Cuconasu et al. (2024) demonstrated that random or noisy documents, when strategically positioned in the context, can sometimes improve rather than degrade LLM performance.

The latest evolution is the Modular RAG paradigm, offering enhanced adaptability through specialized components and flexible architectures (Gao et al., 2023). This approach introduces dedicated functional modules for different tasks, enables integration with fine-tuning techniques, and supports iterative and adaptive retrieval processes. Wang et al. (2024) demonstrated its effectiveness in specialized domains such as alpha mining.

Recent research has also focused on evaluation frameworks. Veturi et al. (2024) proposed comprehensive metrics including retrieval precision and answer similarity, while emphasizing the importance of human evaluation. Their work highlights the ongoing challenge of balancing automated metrics with human judgment in assessing RAG system performance.

This evolution from basic retrieval-augmented generation to sophisticated, modular systems capable of handling complex information retrieval and generation tasks demonstrates the field's rapid advancement. The continued development of evaluation frameworks and optimization techniques suggests that RAG will remain crucial in improving LLM performance and reliability.

# Methodology

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. The holdings comprise documentation of traditional culture from around the world and feature multiformat materials in both analog and digital formats. Its digital collections generally contain photographs, audio recordings of oral history interviews or folksong, correspondence, and manuscript materials such as field notes or logs for recordings or photos, as well as archival collections.

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.

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.

Digital objects on the LoC’s website are arranged into collections that often, but not always, correspond to archival collections on the *findingaids* or *catalog* subdomains. Objects are offered in a variety of audio, video, image, and document formats, and both JSON metadata and the objects themselves are publicly accessible and downloadable via the loc.gov application programming interface (API). Each digital object or file is typically represented by a single metadata record.

## Data Aggregation

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.[[1]](#footnote-1) 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. Note that although .mp4 files were acquired, due to the computationally intensive nature of the Whisper large model and our limited timeframe they were ultimately not included in the project data. We also opted to exclude .wav and .mov files from this project.

Our LoC API queries resulted in two .csv files for each collection, one named *search\_results* and the other named *file\_list*. Each request is based upon a search, and each search returns a list of resources and their descriptive metadata, returned as JSON and saved as a .csv. Each resource can contain any number of derivative files (for example, an image may be available as .jp2, .jpg, or .tif; a video may be available as .mp4 or .mov; audio may be available as .mp3 or .wav), and *search\_results* contains links to the item pages where those files are available. It is this information that is used in a second API call to the item pages to generate *file\_list,* a list of every file for every resource in the search, its type, and its location. Once these files were acquired, a metadata processor to link each file with selected fields across a collection's *file\_list*, *search\_results*, EAD XML, and MARC XML before assigning that information to each chunk in our vector store.

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* 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* 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. Each document's metadata is enhanced with file-specific information including its type (transcript, OCR, or text), original filename, and processing status. The unified metadata schema includes over 30 fields spanning administrative, descriptive, and technical metadata, ensuring that researchers can access collection context regardless of their entry point into the material. The processor recognizes and handles AFC's unique identifier patterns, accommodating both legacy and current naming conventions. It also manages special cases like English translations of audio transcripts, matching them back to their source recordings through pattern recognition.

To ensure data integrity, the processor includes validation steps that verify each metadata field's presence and format. The system implements comprehensive logging that tracks successful and failed matches, providing detailed reports on the processing outcomes. This systematic approach to metadata processing and integration creates a foundation for reliable information retrieval and generation in the RAG pipeline, while maintaining archival and bibliographic context.

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 (collection\_title, collection\_date, collection\_abstract, series\_title) and catalog records (catalog\_title, catalog\_creator, catalog\_date, catalog\_description, catalog\_subjects, catalog\_notes, catalog\_language, catalog\_genre, catalog\_contributors, catalog\_repository, catalog\_collection\_id). 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. While there is some overlap between the kinds of information stored in some of these fields, it is preserved in the interest of cross-referencing and maintaining provenance.

## Initial Approach

In the initial development stages of our RAG, 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 in our approach as we transitioned 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. Deeplake's unified storage 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 transition to Deeplake represented a crucial turning point in our development process, highlighting the importance of selecting technologies that align with the specific requirements of archival materials. Our experience underscored the significance of robust metadata handling capabilities in RAG systems designed for complex document collections, where contextual information plays a vital role in retrieval effectiveness.

## Vector Store

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, and Amazon's Titan, each offering different tradeoffs discovered during literature review.

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 (each field gets its own tensor). 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 and without overwhelming memory resources.

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. For transcript files, the chunking process includes additional preprocessing to remove timecode annotations while preserving the temporal relationship between text segments. The configurable chunk size balances the granularity of text segments against the context window limitations of their chosen language models. It also enabled the review of how different chunk sizes impacted retrieval accuracy.

The embedding generation process incorporates several safeguards to ensure data integrity. A checkpointing system tracks progress and enables recovery from failures, crucial when processing tens of thousands of documents. The system maintains detailed logs of the embedding process, tracking successful operations and documenting any anomalies or failures at both the document and batch level.

Error handling is implemented at multiple levels, from individual document processing to batch operations, with detailed logging that captures both successful operations and failures. The system includes memory optimization techniques, implementing garbage collection and Compute Unified Device Architecture (CUDA) memory cache clearing where appropriate to maintain stable performance during long processing runs. This vector store implementation creates a foundation for reliable information retrieval while preserving the complex relationships and context inherent in archival materials. The result is a system that maintains the integrity of archival description while enabling natural language interaction with the collections.

## RAG Architectures

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 (Figure 1) architecture receives the input query from the user and then embeds it using the same model that was used to construct the vector store. [something about how documents are retrieved-cosine similarity], and the most relevant *top\_k* documents are sent to the LLM to generate a response to the query using a detailed prompt made up of the query, content, metadata, and explicit instructions (Figure 2).

A diagram of a software company

Description automatically generated

Figure 1: Diagram representation of the naïve RAG system.

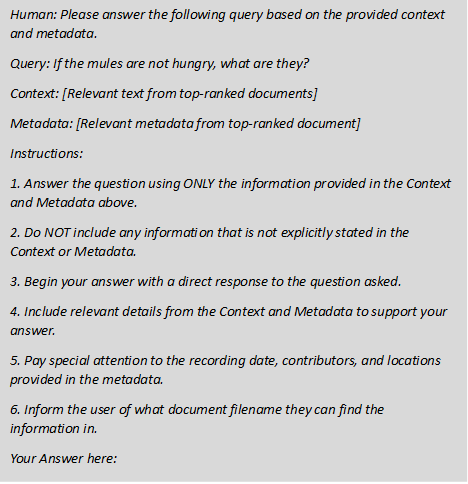


Figure 2: Example of prompt provided to the LLM for response generation.

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 (Figure 2) 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.

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.*

A close-up of a computer screen

Description automatically generated

Figure 3: Diagram representation of the RAG system implanting HyDE.

The final novel RAG system (Figure 3) implements a similar but enhanced HyDE mechanism before retrieval and incorporates a re-ranking mechanism after retrieval. 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 (Figure 4) sent to the LLM during HyDE generation are held equal, including across all systems. 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.

A screenshot of a computer

Description automatically generated

Figure 4: Diagram representation of the RAG system implanting enhanced HyDE and re-ranking.

A re-ranking mechanism is used to reassess the relevance of all of the remaining retrieved documents to the original input query. Their relevancy scores are calculated using a weighted sum of TF-IDF cosine similarity using 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 to response to the query, accordingly. 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)**

**Time Different in Days**

**Freshness Score = (0, 1** - **)**

**365**

**Number of Matching Query Keywords in Document**

**Keyword Coverage =**

**Total Unique Keywords in Query**

# Results and Conclusion

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.

A graph of different colored lines

Description automatically generated

Figure: 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.

A graph of different colored lines

Description automatically generated

Figure: 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)

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.

# Future Work

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.

The solution section covers all of your contributions (architecture, algorithms, formulas, findings).

It explains in detail each contribution, if possible, with figures/schematics.

Don't forget that a figure goes a long way towards helping your reader understand your work.

For instance, Figure 1 outlines the layers involved in a distributed certification service, and how

they articulate together. Nevertheless, a figure must always come with at least one paragraph of

explanation. The rule is that anyone should be able to understand your solution from reading

the text in this section, even if they skip the figures.



Figure 1: Architecture of our distributed certifcation service

Figure 2 is a pretty good example of a figure that is completely useless unless it is not accompanied by a textual explanation.

# Bibliography

|  |  |
| --- | --- |
| [1] | <https://www.loc.gov/research-centers/american-folklife-center/about-this-research-center/> |
| [2] | https://lccn.loc.gov/ |
| [3] | <https://dictionary.archivists.org/> |
| [4] | https://findingaids.loc.gov/exist\_collections/ead3master/ |
| [5] | https://findingaids.loc.gov |
| [6] | https://catalog.loc.gov |
| [7] | https://github.com/openai/whisper |
| [2] |  |
| [3] |  |
| [1] |  |
| [2] |  |
| [3] |  |
| [1] |  |
| [2] | https://lccn.loc.gov/ |
| [3] | <https://dictionary.archivists.org/> |
| [1] | <https://www.loc.gov/research-centers/american-folklife-center/about-this-research-center/> |
| [2] | https://lccn.loc.gov/ |
| [3] | <https://dictionary.archivists.org/> |
| [1] | <https://www.loc.gov/research-centers/american-folklife-center/about-this-research-center/> |
| [2] | https://lccn.loc.gov/ |
| [3] | <https://dictionary.archivists.org/> |

1. *All data for this project was acquired legally via public sources; access to internal LoC sources was neither requested nor required. All computing was performed on Amazon Web Services (AWS) g5.2xlarge Ubuntu Elastic Compute (EC2) instances provided by The George Washington University (GWU).* [↑](#footnote-ref-1)