SI 630 Final Project Report - Enhancing UMSI Academic Information Retrieval with a RAG-Based Question-Answering System

Winter 2025

Sue Yixin Wang

Abstract

This paper presents the development and evaluation of a Retrieval-Augmented Generation (RAG) question-answering system designed specifically for University of Michigan School of Information (UMSI) academic content. The system combines a curated dataset of UMSI-specific materials with a modular pipeline consisting of document retrieval, reranking, and prompt-based answer generation components implemented through the Haystack framework. Through systematic experimentation with various configurations—including sparse (BM25) and dense embedding retrievers, different reranking models, and zero-shot versus few-shot prompting strategies—the research demonstrates how each component affects answer quality. Evaluation using both traditional metrics (F1, recall) and semantic measures (BLEURT) reveals that embedding-based retrievers with metadata enrichment produce the most semantically accurate responses, while simpler prompt designs often outperform more complex configurations. The work contributes a domain-specific academic RAG pipeline, empirical insights into prompt engineering and document processing for academic question answering, and identifies promising directions for future work including dynamic few-shot sampling, hybrid retriever ensembles, and more robust evaluation frameworks. This research provides foundational knowledge for improving information access in specialized academic domains.

1 Introduction (5 points)

Recent advancements in Natural Language Processing (NLP) have enabled powerful question-answering (QA) systems that retrieve information with increasing accuracy and efficiency. However, many existing QA systems struggle when applied to specialized domains—such as academic or career-related queries—where users need concise, well-contextualized, and reliably sourced answers. In this project, I set out to build an end-to-end QA system using the Retrieval-Augmented Generation (RAG) framework, implemented through Haystack, to generate structured and accurate responses about the University of Michigan School of Information (UMSI).

Unlike traditional keyword-based search tools or generic language models, my approach integrates curated academic content with a multi-stage retrieval pipeline. I constructed a comprehensive dataset by manually collecting and annotating UMSI-specific materials, including course offerings, program requirements, historical background, and etc. My system consists of three key components: a retriever to identify relevant documents, a reranker to improve the precision of results, and a prompter to generate informative and well-structured answers using instruction-tuned large language models. I also explored zero-shot and few-shot prompting strategies, along with multiple data processing techniques to improve both retrieval and generation quality.

One of the core insights I gained through this work is that RAG systems can be improved from multiple angles—such as experimenting with different methods for document chunking, embedding models, retrievers, rerankers, and prompters. However, these components interact in complex ways, and their performance gains must be tested empirically rather than assumed. I iteratively refined each module based on evaluation results, reinforcing the importance of systematic experimentation and tuning in building effective QA systems.

The main contributions of this project are: (1) a curated and annotated dataset of UMSI academic content, (2) a tailored RAG-based QA pipeline for academic use cases, (3) empirical insights into prompt engineering and document processing, and (4) an exploration of how each component in a

RAG pipeline—from chunking to reranking—can be iteratively adjusted and evaluated to maximize overall system performance. Through this work, I demonstrate how combining structured domain knowledge with adaptive generation techniques can significantly enhance user experience in academic information retrieval.

2 Data (10 points)

2.1 Data Collection and Sources

For this project, I created a custom dataset focused on the University of Michigan School of Information (UMSI), targeting academic information that prospective or current students frequently seek. The dataset was compiled from publicly available official web pages, including content about programs, courses, admissions, general information, and class schedules. The primary URLs used in the data collection process were:

UMSI Programs:

- https://www.si.umich.edu/programs

UMSI General Information:

- https://www.si.umich.edu/about-umsi

UMSI Courses:

- https://www.si.umich.edu/programs/courses

U-M Schedule of Classes:

- https://ro.umich.edu/calendars/schedule-classes

  I collected 25 documents in total, consisting of 15 text files and 10 PDFs. Each document was assigned metadata, including its title and a category label. The metadata was consolidated into a centralized `metadata.json` file to support indexing, querying, and relevance mapping.

2.2 Proprocessing and Dataset Construction

To prepare the data for QA system development, I cleaned the text from each source document, removed formatting noise from PDFs, and organized the content into meaningful sections using manual chunking. I used semantic chunking based on paragraph structure and headings to ensure contextual integrity across sections. These cleaned chunks were later used for embedding and retrieval within the QA pipeline. The documents were categorized into several themes based on content type. A summary of the document distribution is shown below:

Category	Count	Percentage
MSI program	6	24%
About UMSI	5	20%
Course Schedules	4	16%
BSI Program	3	12%
Academic Information	1	4%
Other	6	24%
Total	25	100%

Table 1: Placeholder caption

2.3 QA Pair Generation and Annotation

I generated an initial set of question-answer (QA) pairs by uploading the 25 source documents into ChatGPT-4o as a project and prompting it to generate QA pairs based on the document content. From this auto-generated pool, I manually selected and annotated 50 questions each from the training and development splits using Doccano, a text annotation platform. During manual annotation, I revised or corrected both the questions and answers for clarity, factuality, and alignment with the document context.

The final dataset includes 570 QA pairs, divided into 400 for training (70%), 85 for development (15%), and 85 for testing (15%). The dataset includes a wide range of question types, ensuring linguistic and topical diversity. The average question length across splits ranged from 14.6 to 15.9 words, while the average answer length ranged from 13.5 words (train) to 19.3 words (test). Table 2 summarizes question types across the full dataset:

Question Type	Percentage
How	20.6%
Is/Are	18.9%
Do/Does	17.6%
What	16.5%
Other	26.4%
Total	100%

Table 2: Question types

2.4 Quality Estimation and Relevance Scoring

To estimate the reliability of QA pairs and test answer consistency, I computed **inter-annotator agreement (IAA)** between model-generated answers and my human-annotated versions. The evaluation metrics included F1 score, Jaccard similarity, BLEU and ROUGE. Since relevance was evaluated on a continuous basis rather than categorical labels, I treated these metrics as proxies for soft relevance scores. Overall, the evaluation results reveal a moderate level of alignment between model-generated responses and the human-annotated reference answers. The average F1 scores, hovering around 0.49, indicate that roughly half of the tokens in the generated answers overlap with those in the ground truth, suggesting that the system is generally capturing relevant content. This is a promising outcome given the open-ended nature of academic question-answering, where multiple valid answers may exist and exact lexical matches are often not expected. Similarly, the Jaccard similarity scores (0.33–0.34) and token overlap values ($\tilde{0}.48$) reinforce the notion that the generated answers frequently contain core concepts and keywords that match human expectations, even if phrased differently. This is particularly relevant in educational domains, where paraphrasing, lists, and summary-style formulations are common and semantically rich expressions may differ on the surface level. The BLEU and ROUGE metrics provide additional perspective on the linguistic quality of generated responses. High unigram overlap (BLEU-1 around 0.45, ROUGE-1 above 0.54) suggests strong alignment on individual words or short phrases. However, lower scores on higher-order n-grams ndicate that the system is less successful at producing extended sequences that mirror human-like sentence structure or flow. This may point to opportunities for improvement in fluency and coherence, especially for longer or multi-part answers. From a practical standpoint, these findings suggest that while the QA system is not yet delivering perfect or complete answers, it is often generating useful and partially correct responses that can reduce the effort required to find relevant academic information. Even incomplete answers that point users toward the correct concept—such as identifying a program's core strengths or summarizing key application criteria—can be more helpful than conventional search interfaces, which typically return lengthy or fragmented results requiring manual review.

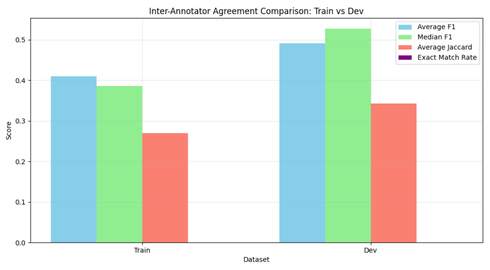


Figure 1: Inter-annotator Agreement Comparison Graph

Below are the evaluation results for the 100-question samples from the training and development sets:

Metric	Dev	Train
Average F1	0.491	0.488
Median F1	0.528	0.508
Average Jacccard	0.343	0.335
BLEU-1	0.457	0.452
ROUGE-1	0.546	0.543
ROUGE-L	0.469	0.441

Table 3: Inter-annotator agreement in train and dev datasets

3 Related Work (10 point)

3.1 Instruction Finetuning for Large Language Models

Chung et al. (2022) explored instruction finetuning as a method to improve large language models' ability to generalize to unseen tasks. Their work scaled the number of finetuning tasks to 1.8K and introduced Flan-PaLM, achieving state-of-the-art performance in zero-shot and few-shot prompting scenarios. In my project, I adopted this principle at a smaller, domain-specific scale by implementing zero-shot and few-shot prompting with instruction-tuned

LLMs to generate and refine question-answer pairs. Rather than training from scratch, I used these instruction-following capabilities within a retrieval-augmented generation (RAG) pipeline. This allowed me to adapt large models to specialized academic content—particularly UMSI program and course details—without requiring thousands of fine-tuning tasks.

3.2 Document-Level LSTMs for Question Answering

LSTMs have been widely used for question-answering tasks, particularly in document-level NLP applications. Wang et al. (2019) introduced GLUE, a multi-task benchmark for natural language understanding (NLU), which evaluated deep learning models on various NLP tasks, including question-answering and sentiment analysis. Their research demonstrated that multi-task learning and transfer learning approaches improve cross-task knowledge transfer. However, traditional LSTM-based models struggle with long-context retrieval and generating contextually relevant responses. My project moved away from LSTMs due to their limited ability to handle long-range dependencies and complex document structures. Instead, I designed a pipeline that leverages transformer-based models for both retrieval and generation. Using Haystack's architecture, I implemented document chunking and dense retrieval to enable long-context comprehension, a key shortcoming of traditional LSTM models. This helped my system generate contextually appropriate answers from multi-page PDFs and structured metadata files.

3.3 BERT-Based Question Answering Models

Devlin et al. (2019) proposed BERT (Bidirectional Encoder Representations from Transformers), which revolutionized pre-trained deep bidirectional models for NLP tasks. BERT's architecture allows contextual embeddings from both left and right context, making it effective for extractive question-answering tasks. However, BERT does not perform retrieval-based augmentation, limiting its ability to answer questions beyond its pretraining corpus. My project extended beyond BERT's limitations by integrating retrievers and rerankers that locate relevant documents from a curated academic dataset before feeding them to a generator (ChatGPT or similar models). This hybrid setup ensures that the final answers are not only contextual but also grounded in up-to-date and domain-specific information. In my implementation (fixbm25.py, models.py), I experimented with different retrievers (BM25, dense) and combined them with prompt-tuned generation to simulate retrieval-enhanced BERT-like behavior, but in a generative QA format.

3.4 HayStack

Haystack, created by deepset-ai, is a user-friendly toolkit designed to help developers easily build and deploy NLP applications, especially question-answering systems. Its main goal is to simplify working with powerful NLP models by providing modular building blocks that can be combined to quickly prototype and test different setups. Haystack allows users to experiment with different ways to retrieve and process documents, and it supports various evaluation metrics like precision, recall, and accuracy to see what combinations work best. By making it simple and flexible, Haystack helps people without extensive NLP backgrounds build sophisticated applications more easily. Haystack was the core framework I used to build the retrieval-augmented pipeline, as seen in main.py and fixbm25.py. I leveraged Haystack's modularity to implement: a retriever (BM25 and dense embedding variants), a reranker to prioritize documents based on semantic relevance, a prompt-based generator for answer synthesis. Haystack also supported the indexing and querying of my manually curated UMSI dataset (25 files, as organized in metadata.json). Its flexibility allowed me to test different configurations of chunking strategies, retrievers, and rerankers to assess the impact on answer quality. Moreover, it enabled easy integration with external LLM APIs for generation, turning Haystack into a powerful experimentation platform for my RAG setup.

3.5 The Retrieval-Augmented Generation (RAG) Approach

Lewis et al. (2020) proposed the Retrieval-Augmented Generation (RAG) model, which combines a traditional neural language model with an external database of information—kind of like giving the model a Wikipedia cheat sheet. The main idea behind RAG is to help language models provide answers that are more accurate, detailed, and factually correct, especially when dealing with topics that need deep knowledge. The authors tested two versions of the model: RAG-Sequence, which uses the same source of information throughout the answer, and RAG-Token, which can choose different information for each word it generates. They evaluated these models on different tasks like answering trivia questions, generating detailed answers, creating Jeopardy-style questions, and verifying facts. The results were impressive—RAG outperformed regular language models and even set new records on some popular NLP benchmarks. One of the coolest things they showed was that the RAG model could easily update its knowledge by changing the external database, keeping the answers fresh and relevant. My project directly operationalized the RAG framework proposed by Lewis et al. (2020). Instead of pretraining a custom RAG model, I implemented a modular variant using Haystack: a retriever retrieved academic documents; a reranker reordered them based on relevance; and a generative model was prompted with the top-ranked documents to produce answers. I tested various chunking and document preprocessing techniques (main.py), showing that chunk granularity and retrieval depth significantly impact answer quality. This framework ensured that generated responses could dynamically reflect new or updated information in the curated UMSI dataset—an advantage RAG offers over static LLMs.

3.6 BLEURT: Learning Robust Metrics for Text Generation

The paper *"BLEURT: Learning Robust Metrics for Text Generation"* presents BLEURT as a learned evaluation metric designed to closely align with human judgments when assessing the quality of machine-generated text. Unlike traditional metrics such as BLEU and ROUGE—which often fail to capture semantic subtleties and perform poorly on paraphrased or contextually rich outputs—BLEURT leverages fine-tuned BERT-based models trained on human-annotated data and millions of synthetic examples to improve generalization. This enables it to outperform existing metrics, particularly in low-resource or out-of-distribution evaluation scenarios, as demonstrated on benchmarks like the WMT Metrics Shared Task and WebNLG. In my project, I adopted BLEURT as a complementary evaluation tool alongside traditional metrics like F1, especially to assess the quality of long-form, generative responses. Given the open-ended and varied nature of academic question-answering, BLEURT offered a more semantically sensitive and human-aligned measure of answer quality, helping me better understand the comparative strengths of each method implemented in the RAG pipeline.

3.7 BibTex Style References

@articlechung2022scaling, title=Scaling instruction-finetuned language models, author=Chung, Hyung Won and Hou, Le and Longpre, Shayne and Zoph, Barret and Tay, Yi and others, journal=arXiv preprint arXiv:2210.11416, year=2022

@inproceedingswang2019glue, title=GLUE: A multi-task benchmark and analysis platform for natural language understanding, author=Wang, Alex and Singh, Amanpreet and Michael, Julian and Hill, Felix and Levy, Omer and Bowman, Samuel R., booktitle=International Conference on Learning Representations (ICLR), year=2019

@articledevlin2019bert, title=BERT: Pre-training of deep bidirectional transformers for language understanding, author=Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, journal=arXiv preprint arXiv:1810.04805, year=2019

@articlestadelmann2019haystack, title=Haystack: the end-to-end NLP framework for pragmatic builders, author=Stadelmann, Thomas and Soni, Tanay and Lee, Sebastian, journal=arXiv preprint arXiv:1910.10073, year=2019

@inproceedingslewis2020retrieval, title=Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, author=Lewis, Patrick and Perez, Ethan and Piktus, Aleksandra and Petroni, Fabio and Karpukhin, Vladimir and Goyal, Naman and Küttler, Heinrich and Lewis, Mike and Yih, Wen-tau and Rocktäschel, Tim and Riedel, Sebastian and Kiela, Douwe, booktitle=Advances in Neural Information Processing Systems (NeurIPS), year=2020

@articlesellam2020bleurt, title=BLEURT: Learning Robust Metrics for Text Generation, author=Sellam, Thibault and Das, Dipanjan and Parikh, Ankur P., journal=arXiv preprint arXiv:2004.04696, year=2020

4 Methods (50 points)

4.1 Pipeline Structure

To build a robust question-answering (QA) system tailored for academic and career-related inquiries at the University of Michigan School of Information (UMSI), I designed and implemented a modular Retrieval-Augmented Generation (RAG) pipeline using the Haystack framework. Each version of the pipeline consisted of three main stages: (1) a document retriever to surface relevant content, (2) a reranker to prioritize the retrieved chunks, and (3) a prompt-based generator for answer synthesis. All documents were indexed in an InMemoryDocumentStore, enabling rapid prototyping without reliance on external databases. Documents were manually curated, cleaned, and semantically chunked, ensuring meaningful units for retrieval and downstream generation.

4.2 Baseline Model: Sparse Retrieval with BM 25

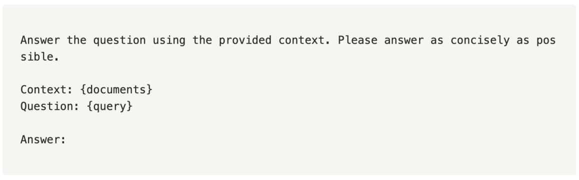
As a baseline, I implemented a classic retrieval configuration using BM25Retriever from Haystack. BM25 was selected due to its effectiveness on keyword-overlapping queries, ease of use, and interpretability. The reranking step was handled using a MiniLM-based cross-encoder, specifically cross-encoder/ms-marco-MiniLM-L-12-v2, chosen for its efficiency and performance on sentence-level relevance tasks. For generation, I employed the LaMini-Flan-T5-783M model via PromptNode from Haystack's prompt module, which supports instruction tuning and prompt customization. This setup served as a foundational benchmark for evaluating retrieval effectiveness without embedding-based representations.

4.3 Embedding-Based Retrieval and Meta-Aware Variants

To address the limitations of sparse retrieval in semantically phrased queries, I implemented an alternative pipeline using dense embeddings. This variant relied on the sentence-transformers/multi-qa-mpnet-base-dot-v1 model as the retriever via Haystack's EmbeddingRetriever. The retriever indexed both the document body and (in some configurations) the title metadata using the embed meta fields argument. This allowed metadata-informed semantic retrieval, which proved helpful in navigating academic content where titles convey critical semantic cues (e.g., "MSI Capstone Projects"). To maintain consistency in reranking and generation, this variation continued to use the MiniLM reranker and LaMini-Flan-T5 generator. Embedding-based methods improved relevance on abstract or loosely phrased queries, justifying their inclusion.

4.4 Reranker Variants: MiniLM vs. DistilRoBERTa

To explore how the reranker architecture influences document prioritization and answer quality, I compared two models: the MiniLM-based reranker (cross-encoder/ms-marco-MiniLM-L-12-v2) and the DistilRoBERTa-based reranker (cross-encoder/qnli-distilroberta-base). The MiniLM reranker is lightweight and optimized for web-style QA, offering fast performance with strong baseline accuracy. In contrast, the DistilRoBERTa model, fine-tuned on natural language inference (NLI), is better suited for capturing nuanced textual entailment, which proved beneficial in more complex reasoning cases. However, its

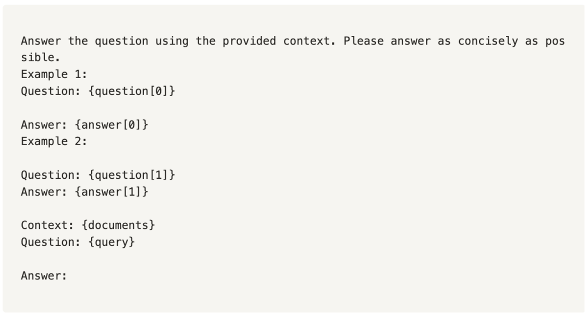


```
Answer the question using the provided context. Please answer as concisely as pos
sible.

Context: {documents}
Question: {query}

Answer:
```

Figure 2: Zero shot prompt



```
Answer the question using the provided context. Please answer as concisely as pos
sible.
Example 1:
Question: {question[0]}

Answer: {answer[0]}
Example 2:

Question: {question[1]}
Answer: {answer[1]}

Context: {documents}
Question: {query}

Answer:
```

Figure 3: Few shot prompt

increased computational cost made it slightly slower in practice. Given this trade-off, MiniLM was selected as the default reranker for most experiments, providing an effective balance between efficiency and quality.

4.5 Prompt Engineering: Zero-Shot vs. Few-Shot

The default prompting strategy employed a zero-shot instruction template that directed the generator to produce answers based solely on the retrieved context. The prompt instructed the model to ”answer the question using the provided context” and to ”respond as concisely as possible,” followed by the contextual documents, the user’s question, and an empty answer field to be completed by the model. To further enhance the quality and structure of generated responses, I implemented a few-shot prompting mechanism using the `few_shot_pipeline()` function. This method retrieved designated annotated question-answer examples from my training dataset, leveraging the BM25Retriever. The retrieved examples were embedded into the `meta['few_shot_example']` field of the prompt template, serving as contextual demonstrations for the model. Few-shot prompting was particularly effective for complex or open-ended questions, aligning with findings from Chung et al. (2022) that instruction-finetuned models benefit significantly from even minimal in-context guidance.

5 Evaluation and Results (30 points)

5.1 Evaluation Methods

To evaluate the quality of the generated answers, I adopted a hybrid evaluation strategy that combined traditional token-level metrics with semantic similarity scoring. Using evaluation.py, I computed a range of standard metrics, including F1 score, Jaccard similarity, token overlap, BLEU (1/2/4), and ROUGE (1/2/L). These metrics were useful for quantifying lexical overlap between system outputs and human-annotated reference answers. However, given the long-form, paraphrased nature of many responses, these methods often failed to capture the true semantic fidelity of the answers. For instance, the exact match rate remained close to zero, and F1 scores plateaued at around 0.49, underscoring the limitations of relying solely on surface-level comparison.

To address this gap, I incorporated BLEURT, a learned evaluation metric introduced by Sellam et al. (2020) that better aligns with human judgment. I integrated BLEURT using the scorewithbleurt.py and usebleurt.py scripts, applying the BLEURT-20 checkpoint to evaluate the similarity between generated answers and reference responses. BLEURT enabled more nuanced assessment by capturing semantic similarity, even when phrasing varied significantly. This dual-metric approach—combining traditional and learned evaluation methods—provided a comprehensive view of system performance, supporting both quantitative benchmarking and deeper qualitative insights.

5.2 Results

Among the tested configurations, the 1Shotk1 method, featuring zero-shot prompting with top-1 document retrieval, achieved the highest F1 score at 0.1596. This suggests that this setup offered the most favorable token-level alignment between generated and reference answers, balancing retrieval precision with concise, well-formed output. In terms of recall, the 5Shotk1 configuration performed best with a score of 0.1796, indicating that including five documents in the context helped cover a broader range of relevant information. However, this came at the cost of slightly lower precision, as the additional content may have introduced distractors or diluted focus.

For semantic quality, both the Embed Meta and Embed Retriever pipelines achieved the highest BLEURT score of 0.4055. This indicates that embedding-based retrieval, particularly when incorporating metadata such as document titles, enhanced the relevance and fluency of the generated responses. These approaches performed well in capturing nuanced, paraphrased content that would be undervalued by surface-level metrics.

Despite the overall modest scores in F1 and recall, several key findings emerged. First, prompting strategy significantly influenced performance. The 1Shotk1 approach consistently ranked among the top across all metrics, demonstrating that minimal prompting combined with high-quality retrieval can yield effective and contextually accurate responses. Second, embedding-based retrieval with metadata, as implemented in the Embed Meta pipeline, led to superior semantic outcomes, underscoring the value of leveraging structured document metadata in information retrieval tasks.

On the other hand, few-shot prompting with top-5 retrieval (1Shotk5) consistently underperformed across all metrics. This suggests that while adding more examples might intuitively improve performance, it can actually overwhelm the model with noise if not carefully balanced with context length and query focus. Additionally, the uniformly low F1 and zero EM scores highlight a broader trend: token-level metrics substantially underestimate quality in generative QA systems. This makes a strong case for relying more heavily on learned, semantic evaluation methods like BLEURT, which better align with human judgment.

In conclusion, the 1Shotk1 configuration emerged as the most balanced performer, excelling in both lexical and semantic evaluation.

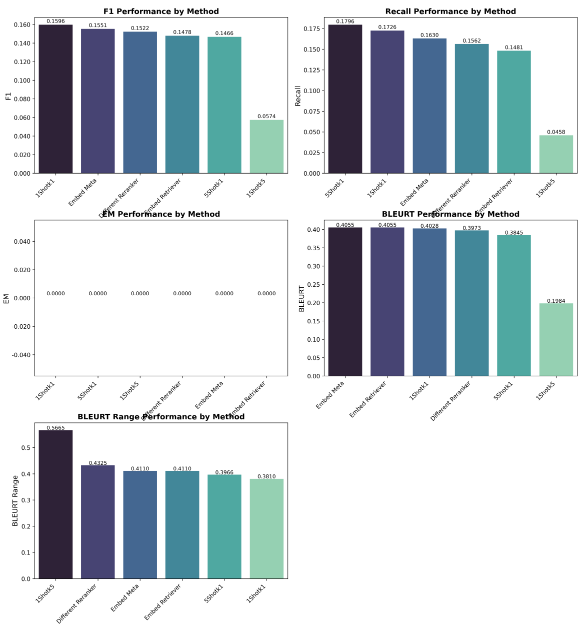


Figure 4: Metrics Comparison

The Embed Meta and Embed Retriever approaches showed particular promise in generating fluent, semantically rich answers, making them strong candidates for future iterations. These results demonstrate the measurable impact of retrieval strategy and prompt design on QA system performance and affirm the importance of using complementary metrics, especially BLEURT, to fully capture the quality of generated content in open-ended domains.

Method	BLEURT	F1	Recall
1Shotk1	0.4028	0.1596	0.1726
1Shotk5	0.1984	0.0574	0.0458
5Shotk1	0.3845	0.1466	0.1796
Embed Meta	0.4055	0.1551	0.1630
Different Reranker	0.3973	0.1522	0.1562
Embed Retriever	0.4055	0.1478	0.1481

Table 4: Quantitative Performance Overview

6 Discussion (20 points)

The goal of this project was to build a retrieval-augmented question answering system capable of generating accurate, structured, and contextually relevant answers to academic queries. While the system showed promise in its ability to synthesize long-form responses using retrieved content, sev-

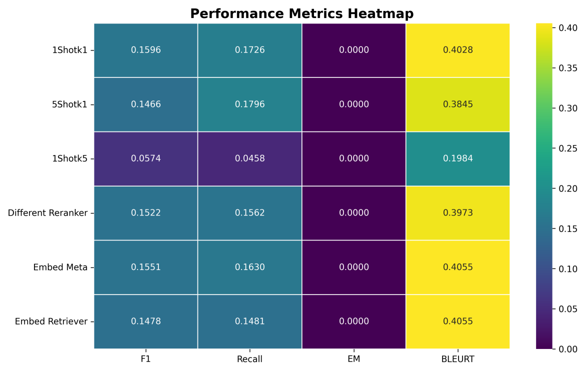


Figure 5: Metrics Heatmap

eral limitations emerged that provide insight into both the successes and the challenges of the current approach.

One key issue is the limited scope and size of the dataset. Although I carefully curated 25 UMSI-related documents and generated 635 QA pairs, this dataset may still be insufficient to fully capture the breadth and complexity of real academic queries. The restricted number of QA examples also limited the diversity and representativeness of the few-shot prompts, potentially undermining the generalization ability of the generator, especially in low-resource prompting configurations like 1Shotk5. As a result, the pipeline's performance, while functional, may not yet be robust enough for real-world deployment or user-facing academic support systems.

Another important limitation lies in the choice of prompter. Throughout the experiments, I used LaMini-Flan-T5 as a consistent, instruction-tuned model. However, recent research has shown that larger and more advanced prompt-tuned or fine-tuned language models, including FLAN-T5-XL or GPT-4, often yield significantly improved results in long-form QA and summarization tasks. Incorporating a more powerful generator, or experimenting with advanced prompt engineering techniques (e.g., chain-of-thought reasoning, self-consistency prompts), could unlock better use of retrieved content and improve both answer structure and factuality.

The evaluation strategy also presented significant challenges. While traditional token-level metrics such as F1 and Recall were used to benchmark performance, they consistently underestimated the quality of generated responses. This is especially apparent in cases where the answer was semantically correct but phrased differently from the reference. All models yielded a 0.0 Exact Match (EM) score, and F1 scores were below 0.16 even for the best method (1Shotk1). These results are not necessarily indicative of poor system performance but rather reflect the incompatibility of surface-level metrics with open-ended, paraphrased, and multi-sentence outputs. As observed in this project, BLEURT provided a more realistic evaluation of answer quality by capturing semantic similarity, with top-performing methods achieving BLEURT scores above 0.40. This highlights the need for semantic-aware evaluation frameworks when working with generative NLP systems.

Beyond the metrics, the design decisions around retrieval and reranking played a pivotal role. Embedding-based retrievers, particularly when enriched with metadata such as document titles, outperformed sparse retrievers in BLEURT evaluation. This suggests that integrating document structure and contextual fields can materially enhance the semantic richness of retrieved content. Meanwhile, reranker variation had a modest effect, reinforcing that the retrieval step itself may have more impact than reranking under the current settings.

Prompting strategy also introduced variation in results. Interestingly, the simple 1Shotk1 configuration outperformed more complex setups like 5Shotk1 and 1Shotk5. This may point to the sensitivity of large language models to prompt design, where adding too many examples or too much retrieval context can distract the model or overload the context window, resulting in degraded performance.

Taken together, the results point to several areas for future work. First, expanding the dataset with more diverse and realistic academic documents—and corresponding QA pairs—would enable more robust training, prompting, and evaluation. Second, integrating a more powerful prompter with advanced prompting techniques could yield more coherent, informative, and structured answers.

Third, adopting embedding-based reranking, or integrating hybrid retrieval techniques, could further refine content selection and improve response relevance. Finally, the evaluation framework itself should evolve to incorporate semantic similarity metrics (e.g., BERTScore, COMET) or human evaluations, given the known limitations of F1 and EM for this task. In conclusion, while the current system functions and provides reasonable outputs, there is ample opportunity to enhance its accuracy, fluency, and evaluability. The results underscore the importance of carefully aligning model architecture, prompt strategy, and evaluation with the demands of long-form, open-domain question answering.

7 Conclusion (5 points)

In this project, I designed, implemented, and evaluated a retrieval-augmented question answering (QA) system tailored to academic and career-related queries at the University of Michigan School of Information (UMSI). Leveraging the Haystack framework, I built a modular pipeline composed of document retrieval, reranking, and prompt-based answer generation. I explored multiple pipeline configurations, including sparse and dense retrievers, reranker variations, and zero-shot versus few-shot prompting, to understand how different architectural and prompting strategies influence answer quality.

To evaluate system performance, I employed both traditional metrics (F1, Recall, Exact Match) and semantic-aware metrics (BLEURT). While surface-level metrics remained low due to the open-ended, paraphrased nature of the task, BLEURT scores revealed that several pipeline configurations—especially embedding-based retrievers enriched with metadata—produced semantically rich and contextually accurate responses. The best-performing setup (1Shotk1) demonstrated the value of simplicity and precision in prompt design, while the Embed Meta approach showed promise in improving semantic alignment through structured document representation.

Through this work, I uncovered key challenges, including the difficulty of evaluating long-form generative QA using standard metrics, the importance of high-quality prompts and retrievers, and the need for a more diverse dataset to improve generalization. These findings point toward several promising directions for future work: expanding the dataset, incorporating more powerful prompters and prompt engineering techniques, exploring hybrid retrieval models, and refining the evaluation framework with embedding-based or human-in-the-loop metrics.

This project demonstrates a functional and extensible approach to domain-specific QA using retrieval-augmented generation, and serves as a foundation for further exploration in long-form academic information access.

8 Other Things We Tried (10 point)

8.1 Web Scraping and Data Preparation

One of the earliest challenges I encountered was data acquisition. I initially planned to scrape structured data from official UMSI and University of Michigan websites to automatically build a comprehensive academic dataset. However, I ran into security firewalls and anti-scraping protections, which blocked automated data collection through tools like requests, BeautifulSoup, and Selenium. After investing considerable time troubleshooting this—writing user-agent rotators, introducing delays, and trying proxies—I was ultimately unable to overcome the barriers. As a result, I switched to a manual data collection process, extracting and cleaning content by hand from 25 key documents, which was both time-consuming and labor-intensive. This step alone took multiple days of effort but was essential for building the document store used in my pipeline.

8.2 Attempts to Replace the Prompt Model

Another major area of experimentation involved improving the quality of the prompter. Based on current literature and my own observations, I believe that better generation performance could be achieved by using more powerful models than the default LaMini-Flan-T5-783M. To test this hypothesis, I attempted to substitute the default prompt node with GPT-4o-mini and LLaMA3 models. I began by trying to integrate GPT-4o-mini and LLaMA3 into the existing Haystack pipeline, keeping the retriever and reranker unchanged. However, this introduced a wide range of compatibility issues, including: (1) Tokenizer mismatches between Haystack's PromptNode and the HuggingFace tokenizer used by LLaMA3. (2) Model loading errors related to framework differences (e.g., Haystack Farm vs. Transformers). (3) API limitations for GPT-4o when used in combination with local Haystack nodes. After several unsuccessful attempts, I simplified the setup by creating standalone Python scripts that directly called the OpenAI API for GPT-4o-mini and loaded LLaMA3 through HuggingFace pipelines. While this bypassed the Haystack integration, it introduced new problems: GPT-4o-mini often returned generic or uninformative answers (e.g., "I don't know"), while LLaMA3 produced outputs that lacked coherence or relevance to the query.

I also explored running these models outside the main pipeline, intending to decouple the generator from Haystack entirely. I developed an independent generation script to test prompts in isolation, but was unable to fully debug the output formatting and token management. Despite these setbacks, these experiments deepened my understanding of model integration, and I remain confident that a more advanced generator, paired with better prompt engineering and example selection, could significantly improve answer quality.

9 What You Would Have Done Differently or Next (10 point)

There were several promising directions I identified throughout the project that, due to time constraints, I was unable to fully explore. One such idea was dynamic few-shot sampling. Rather than using static, pre-selected QA examples within prompts, I had planned to implement a retrieval-based few-shot selection mechanism. This would allow the model to dynamically select the most relevant in-context examples based on the test query, potentially leading to improved alignment with user intent and more accurate responses.

Another direction I considered was developing a hybrid retriever ensemble that combines the strengths of both sparse (BM25) and dense retrieval methods. The idea was to score and rank documents using both approaches and then aggregate or rerank results based on a combined relevance signal. This hybrid strategy could have improved coverage and recall, particularly for queries with mixed lexical and semantic characteristics. However, proper tuning of this setup proved time-intensive and was not completed within the project timeline.

I also aimed to experiment with an embedding-based reranker. While I used cross-encoder-based rerankers such as MiniLM and DistilRoBERTa, I was interested in testing whether embedding-based rerankers could provide a more scalable and efficient alternative, especially for large document sets. Lastly, I had intended to run a human evaluation to qualitatively assess the fluency, factual accuracy, and usefulness of generated answers. Given the known limitations of F1 and exact match metrics for long-form responses, a small-scale user study could have provided more meaningful insights—but it was ultimately deprioritized due to time constraints.

Despite these limitations, the project results suggest several actionable next steps for improvement. One major opportunity lies in using more advanced and flexible prompters. Integrating instruction-tuned models via APIs, such as GPT-4 or Claude, or deploying open-source models like LLaMA3 locally, could substantially enhance response quality, particularly for complex or multi-part academic questions. Additionally, further work in prompt engineering would likely yield improvements. This could include experimenting with different prompt formats, lengths, and few-shot configurations, or incorporating instructional cues to guide the model's tone, structure, and specificity. Finally, future iterations of this project should adopt more robust semantic evaluation metrics such as BERTScore or COMET, used in conjunction with BLEURT, to better capture the richness and relevance of generated content beyond token-level overlap.

References

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, et al. *Scaling Instruction-Finetuned Language Models*. arXiv preprint arXiv:2210.11416, 2022. https://arxiv.org/abs/2210.11416.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805, 2019. https://arxiv.org/abs/1810.04805.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://arxiv.org/abs/2005.11401.

Pietsch, Malte, et al. *Haystack: The End-To-End NLP Framework for Pragmatic Builders*. GitHub, deepset-ai, 2019. https://github.com/deepset-ai/haystack.

Sellam, Thibault, Dipanjan Das, and Ankur P. Parikh. *BLEURT: Learning Robust Metrics for Text Generation*. arXiv preprint arXiv:2004.04696, 2020. https://arxiv.org/abs/2004.04696.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. In *International Conference on Learning Representations (ICLR)*, 2019. https://arxiv.org/abs/1804.07461.