# RAG4Thesis (Author-Aware RAG for Multi-Source Academic QA)

Ting An Teng

## 1 Introduction

When working with multiple senior theses in an academic lab, students often face the challenge of efficiently finding and comparing key experimental details across different documents. These theses are typically confidential and not publicly accessible, making it difficult to use online tools or external APIs for information retrieval. Moreover, the information is often scattered and phrased differently by each author, which makes direct comparison even harder.

To address this, a local author-aware question-answering system based on Retrieval-Augmented Generation (RAG) was built in this project. The system processes a set of theses stored locally, retrieves relevant information for a given question from each author's document independently, and generates author-specific answers using a local large language model (LLM). The core design ensures that all outputs are based strictly on the original thesis content, with clear attribution to the source and author.

To improve retrieval accuracy, the system integrates a pointwise reranking method, where the top 10 candidate passages are first selected using vector similarity, then scored individually by an LLM to select the most relevant top 3 passages for each author. This method is compared against a baseline that directly selects passages using cosine similarity alone. Both modes are evaluated using a small, lab-specific question-answer dataset.

## 2 Problem Definition and Algorithm

### 2.1 Task Definition

*Problem statement:*

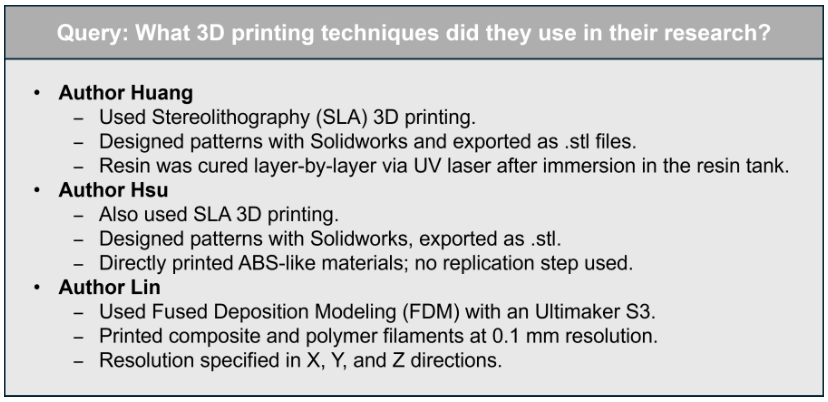
This task is to build an RAG system to answer user queries about lab theses by retrieving and generating relevant answers strictly based on the uploaded thesis document.

*Input:*

The input is simply a natural language question from the user (e.g., "What 3D printing techniques did they use in their research?").

*Output:*

The output is a set of answers, each associated with its source thesis and author, extracted from the uploaded documents.



**Query: What 3D printing techniques did they use in their research?**

- **Author Huang**
  - Used Stereolithography (SLA) 3D printing.
  - Designed patterns with Solidworks and exported as .stl files.
  - Resin was cured layer-by-layer via UV laser after immersion in the resin tank.
- **Author Hsu**
  - Also used SLA 3D printing.
  - Designed patterns with Solidworks, exported as .stl.
  - Directly printed ABS-like materials; no replication step used.
- **Author Lin**
  - Used Fused Deposition Modeling (FDM) with an Ultimaker S3.
  - Printed composite and polymer filaments at 0.1 mm resolution.
  - Resolution specified in X, Y, and Z directions.

**Figure 1.** Example output showing author-specific answers generated by the system for a given query.

### 2.2 Algorithm Definition

This project used a Retrieval-Augmented Generation (RAG) framework with additional reranking methods to enable a local LLM to answer domain-specific questions. The Mistral NeMo model via the Ollama framework was the local LLM used to generate answers for this project. RAG combines information retrieval with language generation. The process begins by encoding the user's question into a vector and retrieving relevant passages from a knowledge base. These passages are then combined with the original query and fed into an LLM, generating a final answer based on the retrieved information and its own knowledge. This approach improves the accuracy and reliability of LLM's outputs. My project comprises three main code components: embedding, retrieval, and retrieval with a pointwise reranking method.

The overall workflow of this RAG system is illustrated below:

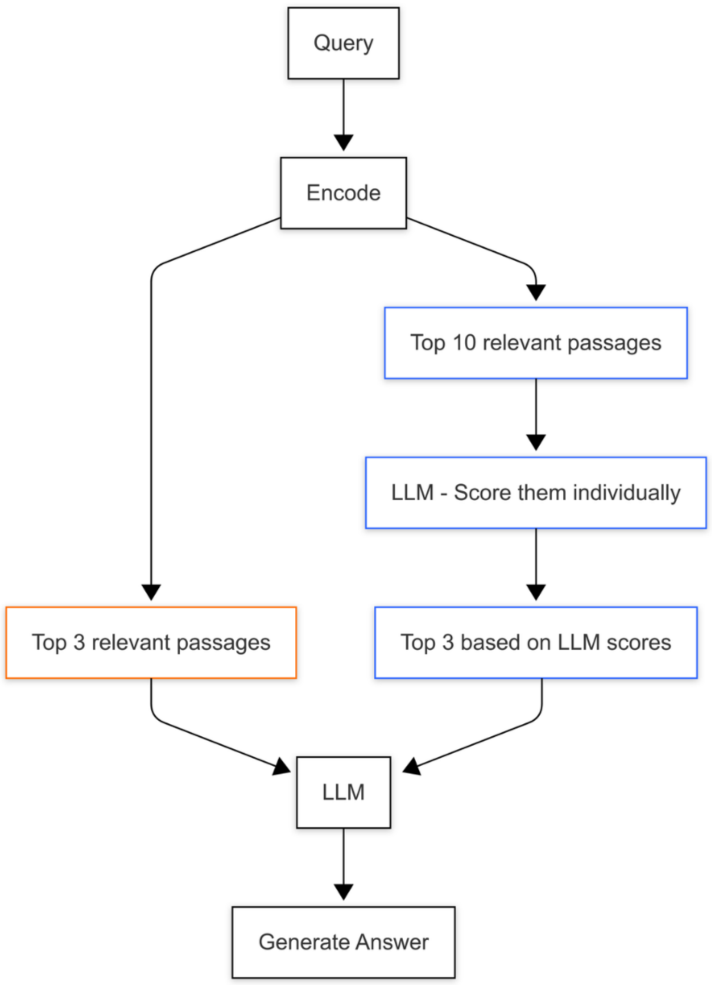


**Figure 2.** System workflow for answer generation. The left (orange) branch shows the baseline method, which selects the top 3 passages based on cosine similarity. The right (blue) branch shows the pointwise reranking method, which first retrieves the top 10 passages, then uses an LLM to score each passage individually and selects the top 3 based on these scores. Both branches feed into the final LLM to generate the answer.

***1.*** *Document Embedding and Vector Storage*

a. For each thesis (grouped by author), split the text into chunks.

b. Encode each chunk into embedding vectors using a pre-trained model from HuggingFace.

c. Store all embedding vectors and corresponding metadata (authors in this case) in a vector database.

***2.*** *Query Retrieval & Answer Generation*

a. When a user submits a question, encode it into a query vector.

b. For each author, compute the cosine similarity between the query vector and all their text chunks.

c. Select each author's top three most relevant passages based on cosine similarity.

d. For each author, combine their top passages with the original question, forming a fixed input format.

e. Feed the constructed input to a local LLM (e.g., Mistral NeMo) to generate answers for each author.

f. Output all answers, each labeled by its author.

*3. Query Retrieval with Pointwise Reranking Method*

a. When a user submits a question, encode it into a query vector.

b. For each author, retrieve the top 10 candidate passages based on cosine similarity with the query vector.

c. Use an LLM to score each candidate passage individually based on how well it answers the query (pointwise relevance scoring).

d. Select the top three passages with the highest LLM scores for each author.

e. Combine the top passages into context grouped by author, then feed the combined context and original query into a local LLM (e.g., Mistral NeMo) to generate answers.

f. Output all answers, each labeled by its corresponding author.

**Example:**

To illustrate the retrieval and reranking process more clearly, **Figure 3** shows how the system handles a single question using baseline and pointwise modes. For each author, relevant thesis passages are retrieved and ranked independently. The top 3 passages are selected based on cosine similarity in the baseline mode. In the pointwise mode, the system retrieves 10 candidate passages per author and uses Mistral NeMo to assign a relevance score to each passage. Based on these scores, the top 3 passages are then selected and passed to the LLM for final answer generation. This figure also highlights the author-aware structure of the system, where each answer is generated separately for each source.

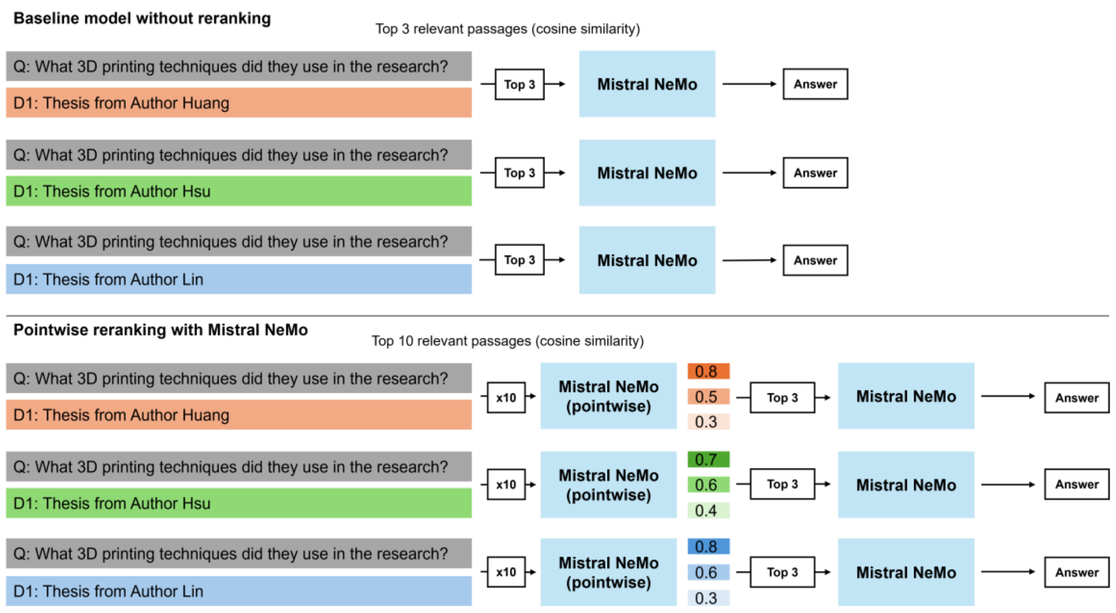

Baseline model without reranking
Top 3 relevant passages (cosine similarity)
Q: What 3D printing techniques did they use in the research?
D1: Thesis from Author Huang
D1: Thesis from Author Hsu
D1: Thesis from Author Lin
Top 3
Mistral NeMo
Answer
Pointwise reranking with Mistral NeMo
Top 10 relevant passages (cosine similarity)
x10
Mistral NeMo (pointwise)
0.8
0.5
0.3
0.7
0.6
0.4
0.8
0.6
0.3

**Figure 3.** Example of the author-based QA process in both baseline and pointwise reranking modes.

# 3 Experimental Evaluation

## 3.1 Methodology

The hypothesis is that pointwise reranking will improve the relevance and accuracy of the generated answers compared to the baseline method.

To evaluate the effectiveness of pointwise reranking in my RAG system, I compared two modes:

1. **Baseline with author filter:**
    a. The top 3 relevant passages are directly retrieved from the vector database for each author.
    b. No additional reranking is applied.
2. **Pointwise with author filter:**
    a. For each author, the top 10 candidate passages are first retrieved using vector similarity (cosine similarity) as a coarse filter.
    b. Then, a language model is used to score each candidate passage individually based on how well it answers the query (pointwise reranking).
    c. The top 3 passages with the highest relevance scores are selected and passed to the LLM to generate the final answer.

For both modes, the selected passages are combined with the original question and provided to the LLM to generate an answer for each author.

Evaluation is based on how well the generated answers match or cover the key points of reference answers from my QA list. The QA list was created from fundamental senior theses in the target domain, ensuring that the test data represents actual user queries and relevant knowledge sources. There are five question-and-answer pairs in the QA list. Additionally, only three senior theses were included in this evaluation. The evaluation was performed manually because only a limited number of QA pairs are available.

For each mode, we manually assess whether the generated answer sufficiently covers the key points of the reference answer. Scores are assigned as follows:

- **1**: The answer covers the key points and provides accurate and complete information.
- **0.5**: The answer is partially correct, covering some key points but lacking completeness.
- **0**: The answer is inaccurate, incomplete, or fails to provide relevant information when it should be available in the thesis.

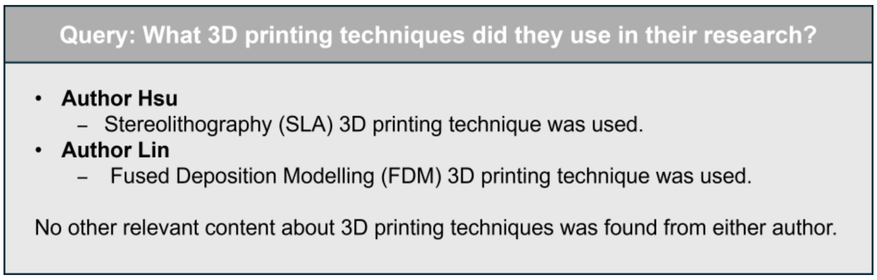
Variables in this project:

- **Independent Variable**: The retrieval modes (Baseline with author filter or Pointwise with author filter)
- **Dependent Variable**: The quality of generated answers is determined by how well the answers match the reference answers in relevance and accuracy.

## 3.2 Results

### Author-specific Retrieval

Initially, the system retrieved relevant passages from all theses without distinguishing between authors. However, this often led to poor answers. As shown in **Figure 4**, content from author Huang was not retrieved even though it was relevant to the query, because the signal from other authors overwhelmed the retrieval process. As the number of files increases, this problem becomes more severe, since unrelated but similar content from other authors can interfere with retrieval. To address this problem, an author-aware retrieval step was added. During the retrieval step, passages are filtered by author, a function natively supported by the Chroma vector database through LangChain's retriever interface (**Figure 6**). It filters chunks based on metadata (e.g., author name) stored during the embedding phase; only chunks belonging to each author are considered when generating their respective answers. The embedding stage remains the same, where all thesis chunks are stored along with the author's metadata. **Figure 5** presents the author-specific retrieval results in much more accurate and complete answers, showing that the improvement in relevance and completeness occurs when the number of files increases.

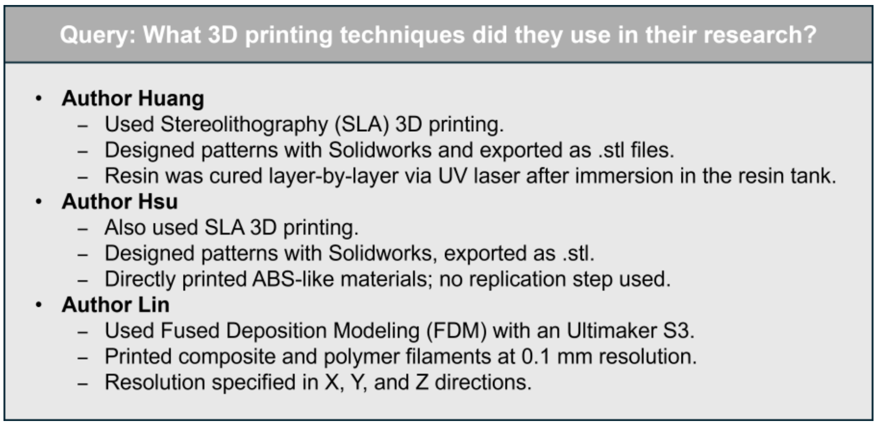


**Query: What 3D printing techniques did they use in their research?**

- **Author Hsu**
  - Stereolithography (SLA) 3D printing technique was used.
- **Author Lin**
  - Fused Deposition Modelling (FDM) 3D printing technique was used.

No other relevant content about 3D printing techniques was found from either author.

**Figure 4.** Answer generated without author-specific retrieval. Relevant content from Author Huang was not retrieved, resulting in an incomplete answer.



**Query: What 3D printing techniques did they use in their research?**

- **Author Huang**
  - Used Stereolithography (SLA) 3D printing.
  - Designed patterns with Solidworks and exported as .stl files.
  - Resin was cured layer-by-layer via UV laser after immersion in the resin tank.
- **Author Hsu**
  - Also used SLA 3D printing.
  - Designed patterns with Solidworks, exported as .stl.
  - Directly printed ABS-like materials; no replication step used.
- **Author Lin**
  - Used Fused Deposition Modeling (FDM) with an Ultimaker S3.
  - Printed composite and polymer filaments at 0.1 mm resolution.
  - Resolution specified in X, Y, and Z directions.

**Figure 5.** Answer generated with author-specific retrieval. The system successfully retrieved passages from the correct source, leading to a complete and accurate response.

```
# retrieval without author filtering
retriever = vectorstore.as_retriever(
    search_kwargs={"k": 3}
)
docs = retriever.get_relevant_documents(question)
```

```
# retrieval with author filtering
retriever = vectorstore.as_retriever(
    search_kwargs={"k": 10, "filter": {"author": author}}
)
docs = retriever.get_relevant_documents(question)
```

**Figure 6.** The top code block shows retrieval without author filtering—passages are selected from all documents regardless of source. The bottom block adds a filter, which ensures that passages are only retrieved from the specified author.

## Retrieval with pointwise reranking method

The evaluation results are summarized in **Figure 7**, which shows the average answer quality scores for three authors under the baseline and pointwise retrieval modes. Among all authors, the pointwise reranking method consistently achieved higher average scores than the baseline. Author Huang's average score increased from 0.6 (baseline) to 0.8 (pointwise). As for Author Hsu's score, it increased from 0.5 (baseline) to 0.7 (pointwise). For Author Lin, the score improved from 0.7 (baseline) to 0.9 (pointwise). For Author Hsu, the improvement was 40%, demonstrating the most significant increase among all authors. These results indicate that pointwise reranking leads to more accurate and complete answers compared to the baseline retrieval method, even under a limited number of sources.

Furthermore, the average answer scores for each query under both modes are shown in **Figure 8**, while **Table 1** shows the question list. From the results, both modes answered query 3 completely correctly. This is likely because the question was straightforward and directly corresponded to a specific section in the thesis, where temperature and reaction time were explicitly stated. Interestingly, for Author Lin, who did not perform PEGMA treatment in her research, the correct answer should be "No relevant content found regarding PEGMA treatment." Both retrieval modes correctly identified this. As for query 5, which is also related to PEGMA treatment, the pointwise mode provided a more accurate and aware answer for Author Lin, correctly stating that she did not perform or mention any long-term durability tests for PEGMA treatment. At the same time, the baseline method failed to recognize and provide these details.

For query 1, both modes struggled with this question, resulting in low average scores. This was likely because the relevant information was scattered throughout the text and described differently by each author. For example, Author Hsu experimented with a wide range of working distances, making it more difficult for the model to extract a concise and complete answer.

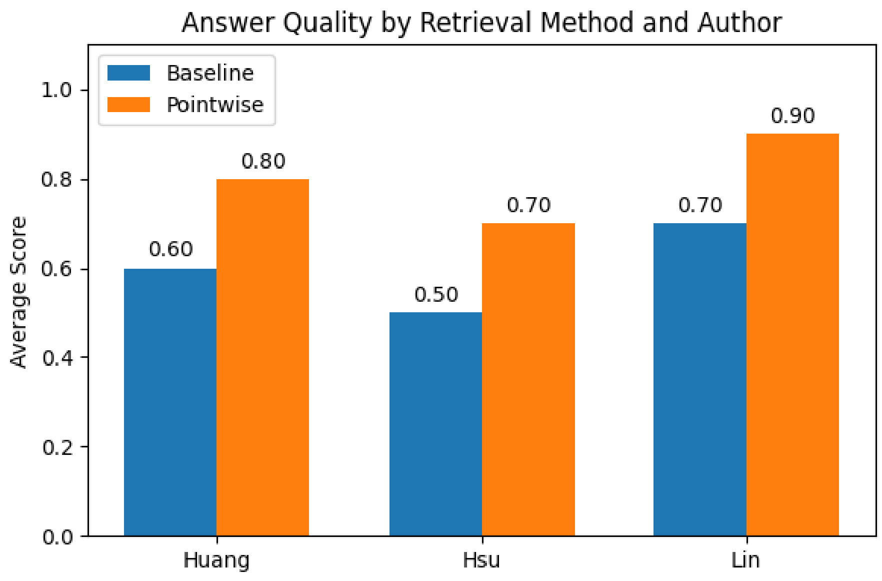


**Figure 7.** Average answer quality for each author under two different retrieval methods. Pointwise reranking leads to consistently higher scores.

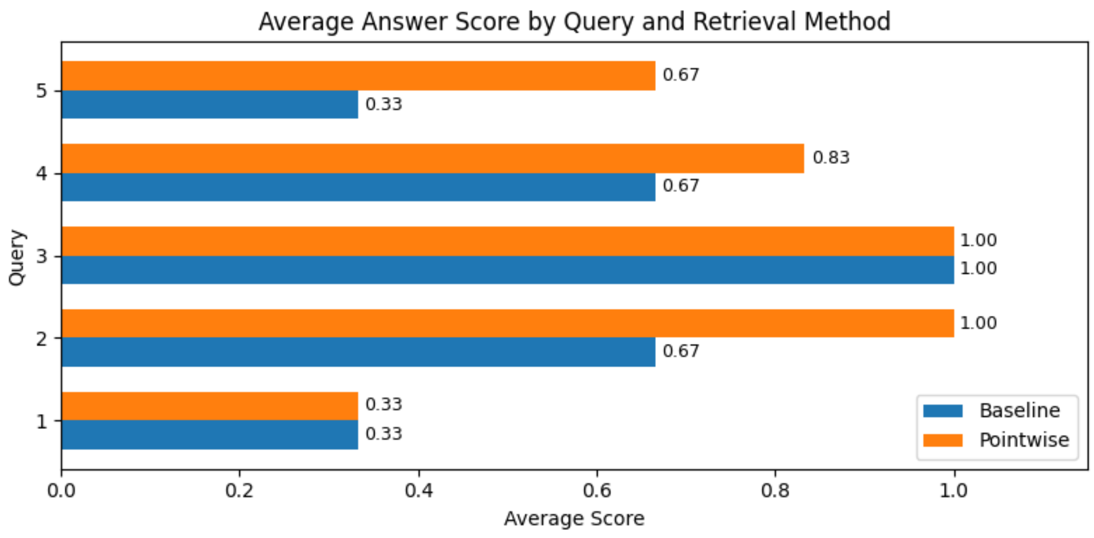


**Figure 8.** Average answer score for each query under baseline and pointwise retrieval methods. Pointwise reranking generally improves or maintains answer quality, with notable improvements in queries 2, 4, and 5.

| | Query |
|---|---|
| **1** | What are the working distances for the APP plasma in their research? |
| **2** | What 3D printing techniques did they use in their research? |
| **3** | What are the temperature and the reaction time they used for PEGMA treatment in their research? |
| **4** | What are the main creatures that inspired the research? |
| **5** | What measurements did they use to confirm the long-term durability of PEGMA treatment or grafting? |

**Table 1.** Question list used for evaluation.

## 3.3 Discussion

The experimental results support the hypothesis that pointwise reranking improves the quality of generated answers. All three authors showed higher average answer scores under the pointwise retrieval mode than the baseline. Compared to the baseline system, which directly selects the top 3 passages based on cosine similarity, the pointwise method retrieves more candidate passages (top 10). It uses a language model to score each passage based on how well it answers the query. This advantage is evident in questions that are more complex or subtle. For example, in query 5 about PEGMA treatment durability, the pointwise system correctly identified that Author Lin did not mention it, while the baseline gave incomplete or misleading answers. This shows that LLM-based reranking helps filter out irrelevant content and improves accuracy. However, both methods failed to give the correct answers for questions like Query 1, where information is scattered or not clearly stated. This suggests that a better reranking or LLM model is needed when answers are not explicitly mentioned. In conclusion, adding a simple pointwise reranking makes the RAG system more capable of choosing relevant content, but it comes with additional computational cost due to extra LLM calls.

## 4 Related Work

This research **(Nogueira et al., 2020)** proposed a pointwise reranking method using the T5 sequence-to-sequence model. Given a query and a document, the model is trained to predict a relevance label ("true" or "false"), and at inference, the probability of "true" is used as the relevance score. Each query–document pair is processed independently, and final rankings are based on these scores.

This work is closely related to this project, which also applies a pointwise strategy using an LLM to score each candidate passage individually based on relevance. While their work focuses on improving whole passage ranking, our project applies a similar pointwise idea in a more author-aware setup. Specifically, we group all passages by author and apply reranking within each author's subset, allowing us to generate answers by each author with traceable sources. This design is especially suited for academic use cases where content must be attributed to specific authors.

## 5 Future Work

### 5.1 Larger and more diverse dataset

At this stage, the system is built using only three senior theses. To make it more valuable and reliable for academic purposes, future work should include more theses from different authors to improve the coverage and robustness of the system.

### 5.2 Improved reranking model or architecture

The current LLM-based pointwise reranking improves the results, but the overall accuracy remains limited. Future improvements could explore other reranking strategies, such as pairwise **(Song et al., 2025)** or listwise **(Yoon et al., 2024)** methods, and experiment with more advanced embedding or language models to enhance performance.

### 5.3 More rigorous and scalable evaluation

The current evaluation relies on manual scoring with a small QA set. In the future, more solid and scalable evaluation metrics like nDCG@k or a benchmark such as BEIR **(Thakur et al., 2021)** can be used to better measure how relevant and well-ranked the answers are.

## 6 Conclusion

This project presents an RAG QA system tailored for academic use, with two main contributions. First, it adopts an author-aware retrieval strategy, where relevant passages are retrieved and reranked independently for each author. This enables the system to generate answers from specific sources. Second, it integrated a pointwise reranking method using a local LLM, allowing for improved accuracy in a zero-shot setting without additional fine-tuning. Finally, the system runs locally using Ollama and Mistral NeMo, which are essential for handling confidential academic documents such as master's theses without relying on cloud APIs. In summary, this integrated design enhances the reliability and accuracy of the author-based QA process, making it a helpful tool for students who need to extract information from multiple theses.

## Bibliography

Nogueira, R., Jiang, Z., Pradeep, R., & Lin, J. (2020, November). Document Ranking with a Pretrained Sequence-to-Sequence Model. In T. Cohn, Y. He, & Y. Liu, *Findings of the Association for Computational Linguistics: EMNLP 2020* Online.

Song, H., Choi, J., & Kim, M. (2025). Ext2Gen: Alignment through Unified Extraction and Generation for Robust Retrieval-Augmented Generation. arXiv:2503.04789. Retrieved February 01, 2025, from https://ui.adsabs.harvard.edu/abs/2025arXiv250304789S

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663. Retrieved April 01, 2021, from https://ui.adsabs.harvard.edu/abs/2021arXiv210408663T

Yoon, S., Kim, J., & Hwang, S.-w. (2024). Analyzing the Effectiveness of Listwise Reranking with Positional Invariance on Temporal Generalizability. arXiv:2407.06716. Retrieved July 01, 2024, from https://ui.adsabs.harvard.edu/abs/2024arXiv240706716Y