DATASCI 507 Final Project Proposal
Medical Paper Recommendation Model via Retrieval-Augmented Generation

Name: Jiahao Cheng Email: jiahaoch@umich.edu UMID: 85362806

1. Overview

1.1 Background and Motivation

The rapid growth of academic publications, like in the medical and machine learning domains, has made it increasingly difficult for researchers and students to efficiently discover relevant work. Traditional keyword-based search engines often fail to capture the semantic intent of user queries, leading to incomplete or irrelevant results. This challenge is particularly pressing in fields like medical research, where timely access to accurate literature is critical.

To address this problem, I propose developing a **paper recommendation model using Retrieval-Augmented Generation (RAG)**. The model will combine deep learning-based **dense retrieval** with **language generation**, allowing users to not only receive relevant papers but also **query-aware natural language summaries** explaining why each recommendation is useful.

This project is driven by both academic curiosity and practical needs. As someone who frequently performs literature reviews, I find that filtering through irrelevant search results is often more time-consuming than discovering new insights. I aim to build a system that simulates the assistance of a smart research assistant—fetching relevant literature and providing justifications.

1.2 Data and Model

I plan to use the **MedRAG/pubmed** dataset from Hugging Face, a curated collection of PubMed papers designed specifically for retrieval-based generation tasks. Each entry includes a medical paper title, abstract, and PMID.

For modeling, the model will include three core components:

- **Bi-encoder**: I will use **`all-MiniLM-L6-v2`**, a Sentence Transformer, as a **bi-encoder** to encode queries and documents into dense vector representations.
- **Cross-Encoder**: I will apply **ms-marco-MiniLM-L6-v2**, a pre-trained MS MARCO sentence transformer model for re-ranking the top retrieved documents based on pairwise query-document input.
- **Language Generator**: I will locally deploy **`DeepSeek-R1-Distill-Llama-8B`**, using prompt engineering skills to generate concise, query-aware summaries of the top-ranked papers.

The first 2 models are sourced from **Hugging Face**, the last model is sourced from **Ollama**.

1.3 Expected Insights

Through this project, I expect to gain insights into:

- Analyzing different methods of document representation and RAG.
- The quality and interpretability of generated summaries for medical content.
- Trade-offs between model performance, efficiency, and deployment complexity

2. Prior Work

2.1 Literature review

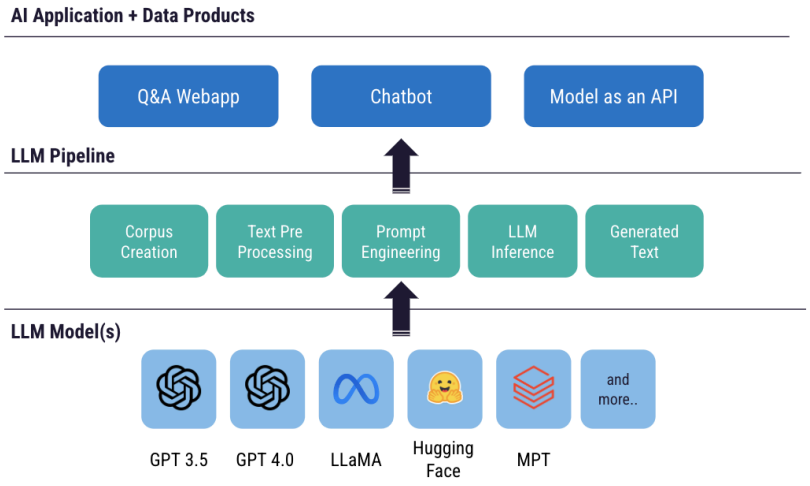
- **Large Language Model(LLM)**

Large Language Models (LLMs) are a class of foundation models trained on massive corpora of text—spanning billions of documents—using self-supervised learning. LLMs use multi-layered neural networks and an attention mechanism to learn semantic, syntactic, and contextual relationships within sequences of tokenized text.

During training, LLMs learn to predict the next token in a sequence, forming a probabilistic model of language. This enables them to generate coherent, context-aware responses across diverse domains. Fine-tuning methods such as **prompt engineering**, **reinforcement learning with human feedback (RLHF)**, and **domain-specific tuning** help mitigate issues like bias and hallucination—where a model produces plausible but incorrect information.

In real-world applications, LLMs power:

- **Text generation** (e.g., summarizing research papers or generating abstracts)
- **Conversational agents** and **virtual assistants**
- **Content summarization** for long-form documents
- **Code generation**, **sentiment analysis**, and **language translation**



- **DeepSeek-R1**

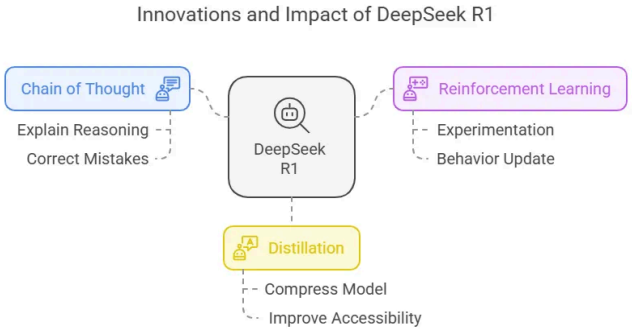
DeepSeek R1 is a recent large language model developed in China, notable not only for its performance but also for its innovative training methodology. The model introduces three core concepts that make it highly suitable for grounded, explainable, and scalable NLP systems like the one proposed in this project:

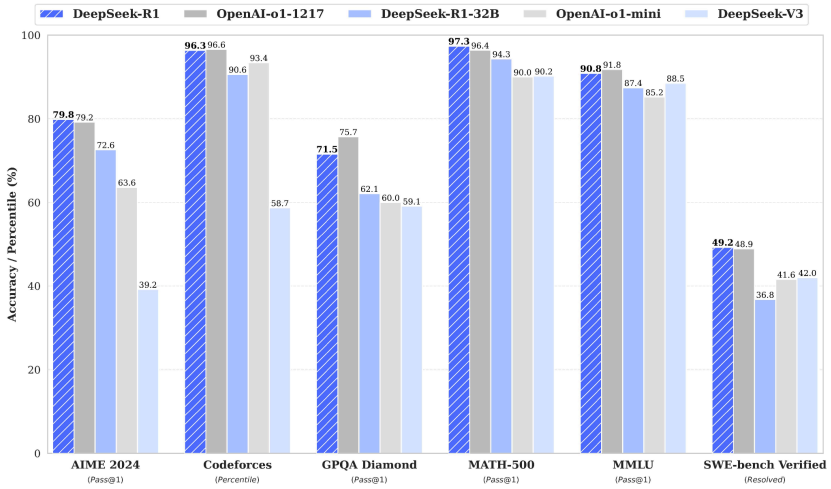
Chain of Thought (CoT): Traditional language models often provide answers without explanations, making it hard to trace mistakes. DeepSeek R1 incorporates **Chain of Thought reasoning**, which encourages the model to **explain its reasoning step-by-step**. This improves interpretability and often leads to more accurate results, even without additional fine-tuning. In the context of paper recommendation, CoT allows the model to explain *why* a paper is relevant to the user's query, making recommendations more transparent and trustworthy.

Reinforcement Learning via Group Relative Policy Optimization (GRPO): Instead of relying solely on supervised learning, DeepSeek R1 uses **reinforcement learning** to improve its output iteratively. Through **GRPO**, the model compares new responses to past attempts and reinforces those that are better. This enables **self-improvement** without requiring massive labeled datasets. This training method allows models like DeepSeek to adapt efficiently while maintaining a lower training cost.

Distillation for Accessibility: While the full DeepSeek R1 model contains over 670 billion parameters, it has been **distilled** into smaller models such as **`DeepSeek-R1-Distill-Llama-8B`**. Through this process, the large model teaches a smaller one, significantly reducing infrastructure demands while retaining much of the original performance.

In conclusion, DeepSeek R1 is important not just because of what it can do, but because of how it does it. **Chain of Thought** makes AI more transparent. **Reinforcement learning** makes it more self-improving. **Distillation** makes it more available.





- **Retrieval-Augmented Generation(RAG)**

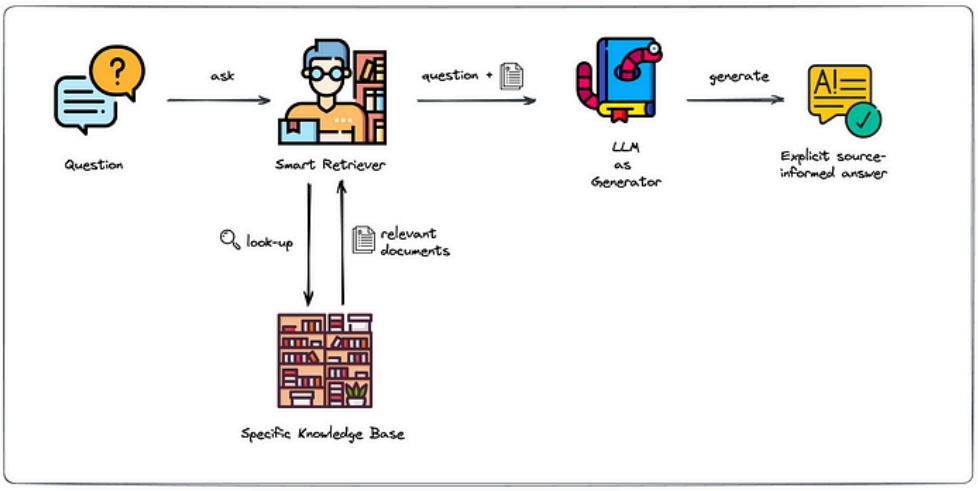
Large Language Models (LLMs), such as **GPT-4** (OpenAI), **Claude** (Anthropic), **Llama** (Meta), and **DeepSeek-R1**, face several limitations: they can be inconsistent, occasionally hallucinate facts, and rely on static knowledge encoded at training time. This creates challenges in tasks requiring up-to-date, verifiable information—especially in high-stakes domains like medical literature search.

Retrieval-Augmented Generation (RAG) offers a promising solution. Instead of relying solely on the model's internal parameters, RAG combines an external document retriever with a generator. This hybrid setup grounds the generated answers in a **retrieved, contextually relevant knowledge base**, improving factual accuracy and reducing hallucinations.

Benefits of RAG include:

- **Access to real-time, reliable sources**, ensuring responses are grounded and checkable
- **Reduction in hallucination and sensitive data leakage**, by limiting reliance on internal memory
- **Cost-effective updates**, as there's no need to continuously retrain the model for knowledge refresh
- **Transparent sourcing**, allowing users to trace generated content back to specific documents

In the context of academic paper recommendation, these benefits are essential. RAG enables both relevant retrieval and readable justification, enhancing user trust and model interpretability.



- **Bi-encoder & Cross encoder**

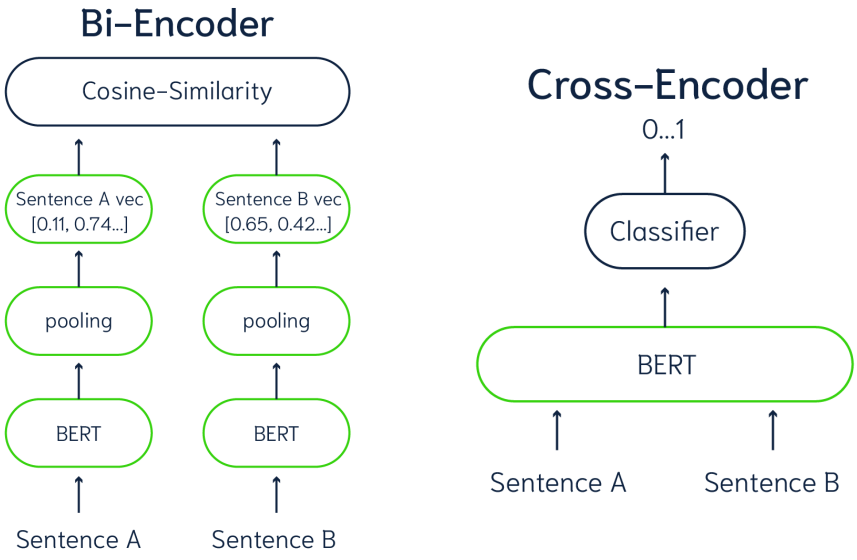
In modern natural language processing (NLP), **Bi-encoders** and **cross-encoders** are two primary architectures used for modeling the semantic relationship between a query and a document.

Cross-encoders jointly encode the query and the document by concatenating them and feeding them into a transformer model. This approach produces a single output score (e.g., relevance or similarity) by considering the full interaction between both pieces of text. While cross-encoders tend to be more accurate—especially in tasks where subtle semantic distinctions are important- they are computationally expensive. This is because every query-document pair must be processed together, making them inefficient for large-scale retrieval where millions of comparisons are needed.

In contrast, **Bi-encoders** encode the query and document **independently** into dense vector representations. The relevance score is then computed using a distance metric (such as cosine similarity) between the embeddings. Bi-encoders are significantly more scalable, enabling real-time search across large corpora. However, they generally underperform cross-encoders in tasks requiring fine-grained semantic matching.

When to use one versus the other depends on the use case.

- **Bi-encoder**: If you want to compare hundreds of thousands of sentences(Focus on efficiency)
- **Cross encoder**: If subtle differences matter(Focus on accuracy)



2.2 Potential Methods

- **Usage of Language Model - Deepseek r1 and Prompt Engineering**

In this project, I propose using **`DeepSeek-R1-Distill-Llama-8B`** as the core **language generation module** in the retrieval-augmented recommendation model. The language model will function as a **virtual research assistant**, summarizing academic papers and generating human-readable explanations for why each paper is relevant to a user's query. **`DeepSeek-R1-Distill-Llama-8B`** is a distilled version of the full DeepSeek R1 model, offering high performance while remaining computationally feasible for local deployment via **Ollama**. It is particularly well-suited for this task due to its two key features:

Reasoning Process: The **Chain of Thought(CoT)** and **Reinforcement Learning(RL)** process makes this model improve its accuracy by correcting mistakes by themselves. When summarizing a paper or justifying its relevance, the model explains its logic step by step, enhancing trust and interpretability.

Easy deployment: the distilled model is easy to be locally deployed in my own PC, leads to less cost.

Finally, using prompt engineering to modify the prompt content and format can make the generation content better.

- **Usage of Retrieval-Augmented Generation(RAG)**

To enhance the accuracy and trustworthiness of paper recommendations, this project employs the **Retrieval-Augmented Generation (RAG)** architecture. RAG combines a **retriever model** with a **language generation model**, enabling the system to generate natural language outputs that are directly grounded in external documents.

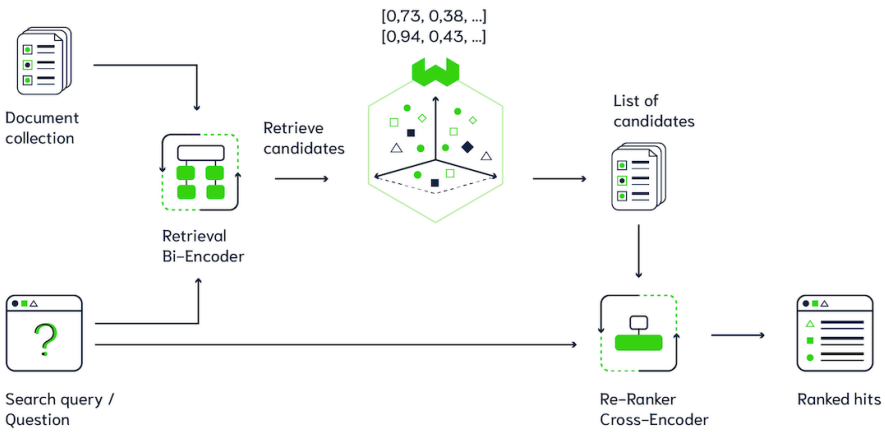
Benefits of RAG include: accessing to real-time, reliable sources, ensuring responses are grounded and checkable, reducing hallucination, cost-effectiveness, transparent-sourcing.

- **Combining Bi-Encoders and Cross-Encoders to improve the RAG pipeline**

Catching fish with the big net represents how Bi-Encoders work. Bi-encoders are fast but are not as accurate as the expensive fisherman, aka the Cross-Encoder. Cross-encoders are time-consuming, like the fisherman who would need to limit the number of fishing rounds they could do. So we can chain those two methods behind each other.

To combine cross-encoder and Bi-encoder in a RAG (Retrieval-Augmented Generation) pipeline, I can use a two-stage search pipeline that first uses a bi-encoder for fast and efficient retrieval of a list of result candidates, followed by a cross-encoder for reranking the most relevant results.

By chaining these two techniques, the system benefits from the **speed of bi-encoders** and the **accuracy of cross-encoders**. This multistage approach is especially effective for **retrieval-augmented generation**, where both **efficient large-scale search** and **precise document grounding** are critical.



3. Preliminary Results

3.1 Data Understanding

- **Data Description**

PubMed is the most widely used literature resource, containing over 36 million biomedical articles. For MedRAG, it uses a PubMed subset of 23.9 million articles with valid titles and abstracts. Since the limitation of hardware limitations, I selected the first 50k articles as my dataset.

The data consists of 5 columns: id, title, content, contents, PMID. contents are the combination of title and content. PMID is the number indicating each paper's url, like https://pubmed.ncbi.nlm.nih.gov/21/. The dataset is a pre-processed dataset, so it doesn't have a NaN value.

	id	title	content	contents	PMID
0	pubmed23n0001_0	[Biochemical studies on camomile components/II...	(--)-alpha-Bisabolol has a primary antipeptic ...	[Biochemical studies on camomile components/II...	21
1	pubmed23n0001_1	[Demonstration of tumor inhibiting properties ...	A report is given on the recent discovery of o...	[Demonstration of tumor inhibiting properties ...	22
2	pubmed23n0001_2	Effect of etafenone on total and regional myoc...	The distribution of blood flow to the subendoc...	Effect of etafenone on total and regional myoc...	23
3	pubmed23n0001_3	Influence of a new virostatic compound on the ...	The virostatic compound N,N-diethyl-4-[2-(2-ox...	Influence of a new virostatic compound on the ...	24
4	pubmed23n0001_4	Pharmacological properties of new neuroleptic ...	RMI 61 140, RMI 61 144 and RMI 61 280 are newl...	Pharmacological properties of new neuroleptic ...	25

- **Initial Insights and challenges**

In our model, word embedding is based on column contents. When a user gives an input, the Retrieval model will retrieve relevant articles based on the cosine similarity between the query and the word embedding. Then, ask the Augmented model to rerank the article list based on their relevance.
Therefore, when doing retrieval and Augmentation, the model is focusing on the column content.
Below is the first article's content. There are many irrelevant stop words, punctuations, and special signs. Erasing them in advance not only reduces the workload of afterward computing but also helps the model find the true related pattern.

```
"[Biochemical studies on camomile components/III. In vitro studies about the antipeptic activity of (--)-alpha-bisabolol (autho
r's transl)]. (--)-alpha-Bisabolol has a primary antipeptic action depending on dosage, which is not caused by an alteration of t
he pH-value. The proteolytic activity of pepsin is reduced by 50 percent through addition of bisabolol in the ratio of 1/0.5. The
antipeptic action of bisabolol only occurs in case of direct contact. In case of a previous contact with the substrate, the inhib
iting effect is lost."
```

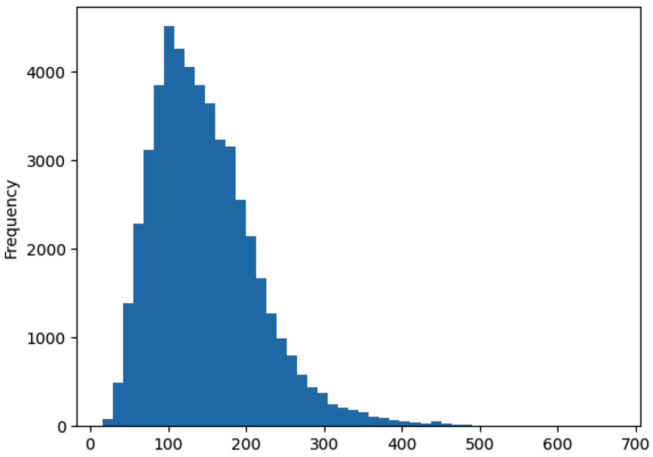
Here, I used spaCy to do Natural Language Processing. I processed each content with the following process: tokenization -> extracting alpha words -> lemmatization. Below is the result of the first contents' lemmatization(left) and extracting alpha words(right).

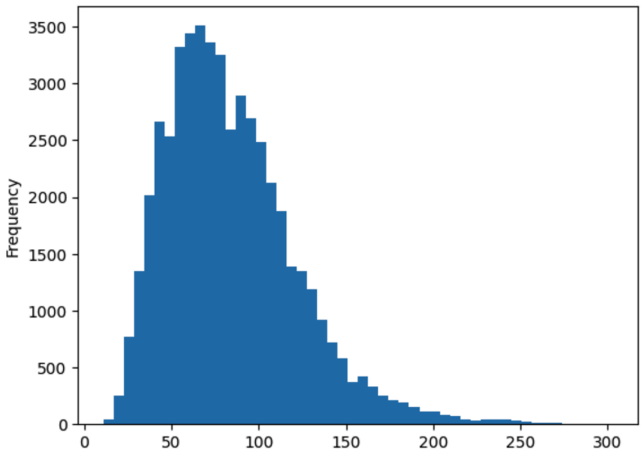
```
Biochemical : biochemical
    studies : study
 components : component
         In : in
    studies : study
        has : have
  depending : depend
         is : be
     caused : cause
         pH : ph
        The : the
         is : be
    reduced : reduce
        The : the
     occurs : occur
         In : in
 inhibiting : inhibit
         is : be
       lost : lose
```

	Text	Index	Whitespace	Is Alphanumeric?	Is Punctuation?	Is Stop Word?
0	[	0	False	False	True	False
1	Biochemical	1	False	True	False	False
2	studies	13	False	True	False	False
3	on	21	False	True	False	True
4	camomile	24	False	True	False	False
...	...	...	...	...	...	...
98	inhibiting	509	False	True	False	False
99	effect	520	False	True	False	False
100	is	527	False	True	False	True
101	lost	530	False	True	False	False
102	.	534	False	False	True	False

103 rows × 6 columns

Before cleaning, the numbers of words in each contents ranges from 16 to 674, with the mean of 148.
After cleaning, the numbers of words in each contents ranges from 11 to 303, with the mean of 84.
Below are the distributions of the number of words before(left) and after(right) cleaning:





3.2 Basic Model

The project uses three models for implementation:

- **Retrieval:** [all-MiniLM-L6-v2]

The all-MiniLM-L6-v2 model is designed to be lightweight, making it suitable for environments with limited computational resources. It requires only 25MB of storage and runs efficiently on a CPU.

Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
68.06	49.54	58.80	14200	80MB

- **Augmented:** [ms-marco-MiniLM-L6-v2]

The model can be used for Information Retrieval: Given a query, encode the query with all possible passages (e.g. retrieved with ElasticSearch). Then, sort the passages in decreasing order.

NDCG@10 (TREC DL 19)	MRR@10 (MS Marco Dev)	Docs / Sec
74.30	39.01	1800

- **Generation:** [DeepSeek-R1-Distill-Llama-8B]

3.3 Tools Specification

Tools from class	**NumPy** – Used for efficient numerical computing and handling matrix operations. **Pandas** – Applied for dataset preprocessing, statistical analysis, and data organizing. **Matplotlib** – Used for visualizing dataset distributions. **PyTorch** – Core deep learning framework for working with Hugging Face models.
Tools will explore	**Transformers (Hugging Face)** – To load pre-trained models.

	spaCy – This is for optional text preprocessing, keyword extraction, and entity recognition from paper abstracts or queries. **Ollama** – To locally run and interact with the LLM. **re (Regular Expressions)** – For text cleaning and pattern matching.

4. Project Deliverables

4.1 Expected Project Outcome

GitHub Repository:
Well-documented codebase with clear instructions on how to replicate the system.
A Jupyter notebook or Python script demonstrating the end-to-end pipeline: data preparation, retrieval, and generation.
Project Report:
A 2-page summary report in the format of a 2-page IEEE conference paper.
Interactive Demo:
A simple user interface where users can input a query and receive paper recommendations with explanations.

4.2 Sub-goals

Data Selection and Processing: Choose appropriate dataset and process the data
Model Selection and Implementation: Choose appropriate models for different purposes and integrate them together
Model Evaluation: Create test data to evaluate model performance
Report: Set up the GitHub repository and provide comprehensive documentation, write a final summary report.

5. Timeline

Weeks	Dates	Milestones & Tasks
Week 1-2	3.17 - 3.30	- Choose and explore medical paper datasets (MedRAG / PubMed) - Conduct exploratory data analysis (EDA) on titles, abstracts, etc. - Finalize project topic and review related literature - Select method: Retrieval-Augmented Generation(RAG) - Select initial models: Bi-Encoder, Cross-Encoder, DeepSeek-r1 - Write project proposal
Week 3-4	3.31 - 4.13	- Implement Bi-Encoder retrieval with vector indexing - Develop Cross-Encoder re-ranking pipeline - Locally deploy and configure DeepSeek-R1-Distill-Llama-8B with Ollama - Creating test query by using Llama3.2 - Evaluate the initial RAG pipeline with the test query - Finalize full RAG pipeline (retrieval + re-ranking + generation)
Week 5	4.14 - 4.20	- Write final report - Prepare a well-documented GitHub repository with demo