🤖 Building a Q&A RAG System 🗂️

🎯 Objectives

This homework challenges you to build a solid **Retrieval-Augmented Generation (RAG)** pipeline by modifying, testing, and improving document retrieval strategies. You will experiment with **ingestion pipelines**, explore **multiple retrieval strategies**, and implement advanced **evaluation techniques**. By the end of this assignment, you will:

✅ **Optimize an ingestion pipeline** to improve retrieval quality for different documents.
✅ **Design and test a comprehensive query set** to evaluate retrieval effectiveness.
✅ **Enhance multimodal retrieval** for images and tables embedded into PDFs
✅ Comparing **multiple retrieval strategies**, including **vector-based, keyword-based (BM25), and auto-merging retrieval**.
✅ **Implement and analyze a hybrid retrieval approach**, combining **vector-based and keyword-based search** for improved accuracy.

This hands-on project will provide practical experience in **building and optimizing RAG-based AI applications**, making retrieval systems smarter and more effective.

📌 Understanding the Provided Code

The provided **HW4-RAQ.ipynb** sets up a **basic RAG pipeline** using **LlamaIndex**, handling document ingestion, retrieval, and evaluation. Below is a high-level breakdown:

📝 **Document Ingestion & Processing**

- Loads a **PDF document** and processes it through an **ingestion pipeline**.
- Splits the document into **text chunks** and generates **vector embeddings** for retrieval.

🔍 **Retrieval Methods**

- **Vector Search:** Uses embeddings for semantic similarity.
- **BM25 Retrieval:** Keyword-based matching.
- **AutoMerging Retriever:** Dynamically groups similar chunks for improved context retrieval.

📊 **Query Execution & Evaluation**

- Executes **queries** and retrieves relevant document sections.
- Evaluates retrieval accuracy using **faithfulness, relevancy, MRR, and hit rate**.

🚀 **Your Tasks:** You will extend this pipeline by experimenting with **custom document ingestion**, integrating **multimodal content (tables & images)**, and optimizing retrieval with **hybrid search techniques** on a different document (**WebMD.PDF**)

☃️Have fun building your first RAG system! ☃️

📌 Task 1: Optimizing the Ingestion Pipeline for Medical Documents [5 points]

Objective: Modify and optimize the **ingestion pipeline** to improve retrieval quality by experimenting with different processing techniques, including chunk size, overlap, and document segmentation strategies.

📖 Background

The ingestion pipeline is a critical component of **Retrieval-Augmented Generation (RAG)** systems, as it determines how documents are **processed, chunked, and stored** for retrieval. Optimizing this pipeline can significantly impact retrieval effectiveness by ensuring that information is logically structured and contextually meaningful.

For this task, you will work with the **WebMD Report on Chronic Migraine (Winter 2025)** and explore different ways to ingest it into the RAG system. The goal is to optimize **document segmentation and retrieval performance** by adjusting the ingestion pipeline settings.

🔷 Steps:

1. **Load and process the WebMD PDF** into the existing RAG pipeline.
2. **Analyze the document structure** (headings, sections, tables) and decide how to best handle different content types.
3. **Experiment with different chunking strategies**, adjusting: **Chunk size** (e.g., small vs. large chunks), **Chunk overlap** (ensuring contextual continuity).
4. **Sentence vs. paragraph-based splitting** (considering meaningful segmentation).
5. **Run retrieval queries** to test how different ingestion settings affect response quality.
6. **Compare retrieval performance** with and without ingestion optimizations.

Resources: https://docs.llamaindex.ai/en/stable/module_guides/loading/

📤 Deliverable:

- The final **Jupyter Notebook** with modifications to the ingestion pipeline.
- A **brief analysis (1-2 paragraph)** discussing the impact of your optimized ingestion strategy with the baseline. Provide screenshots of the outcome to support your argument. In your report include a chart/graph on performance (before/after) of all retrieval methods (don't worry about hybrid retrieval (Task 4) for now).

📌 Task 2: Defining a Comprehensive RAG Test Set [5 points]

Objective: Create a **diverse and challenging test set** to evaluate different retrieval methods in the RAG system, ensuring coverage of various query types.

📖 Background

A robust **test set** is essential for evaluating retrieval performance and helps assess the strengths and weaknesses of different retrieval strategies.

By designing a **carefully structured** test set, you will explore the **trade-offs between retrieval approaches** and how different types of questions impact performance.

🔷 Steps:

1. **Develop at least 10 diverse test questions** that challenge different retrieval methods. Ensure your test set includes:

    a. **Cross-sectional questions** → Require pulling information from multiple sections of the document.
    b. **Keyword-heavy questions** → Rely on specific terminology or exact phrase matches.
    c. **Semantic/contextual questions** → Depend on understanding meaning beyond exact keywords.
    d. **Structured data retrieval** → Require extracting information from structured content, such as lists or formatted text.
    e. **Image-based or multimedia-related questions** → Reference key information stored in images or figures.

2. **Run the RAG system** (retrieval, query engine, etc.) to analyze performance

📤 Deliverable:

- Submit my_questions.txt that contains your questions.
- The final **Jupyter Notebook** with any modifications made.
- A **brief analysis (1-2 paragraph)** discussing retrieval trade-offs and include a chart/graph for performance comparison on different question categories (don't worry about hybrid retrieval (Task 4) for now).

📌 Task 3: Extracting Information from Images [6 points]

Objective: Extract **information from images** (figures, scanned sections, handwritten notes) and integrate it into the RAG pipeline to improve retrieval completeness.

📖 Background

In class, we used **Tabula** to extract **structured tabular data** from PDFs and incorporate it into the ingestion pipeline. However, **some important content in a PDF is stored in images**—many documents contain **figures, charts, scanned sections, or handwritten notes** that require a different approach. Standard text-based retrieval does not process this information.

OCR (Optical Character Recognition) enables the extraction of textual content from these images, making them searchable and improving retrieval accuracy.

🔷 Steps:

1. **Use an OCR tool** (Tesseract or EasyOCR) to extract text from images in the document.
2. **Aggregate extracted text** from multiple images into a single document.
3. **Convert the extracted text** into a **Document object** for ingestion.
4. **Append this document** to the existing text-based content in the ingestion pipeline.
5. **Run the RAG system** (retrieval, query engine, etc.) to analyze performance

💡 Hints

- **Extraction Options:**
    - **Single Image Extraction**: Use **PyMuPDF** if targeting specific images embedded in the PDF.
    - **Full-Page Conversion**: Use **pdf2image** if running OCR on entire pages.
- **OCR Choices:**
    - **Tesseract OCR**: Fast and customizable, requires some configuration.
    - **EasyOCR**: Easier to use but runs slower on large documents.

📤 Deliverable:

- The final **Jupyter Notebook** with any modifications made for images-based extraction.
- A **brief analysis (1-2 paragraph)** discussing your approach, implementations, and the impact of this component.

📌 Task 4: Implementing a Hybrid Retrieval Approach [6 points]

Objective: Implement a hybrid retrieval system that combines lexical (keyword-based) and semantic (vector-based) retrieval to improve response accuracy.

📖 Background

In real-world applications, **hybrid retrieval** is a common industry practice used in **search engines, chatbots, and enterprise knowledge management systems**. Combining **semantic (vector-based) retrieval** with **lexical (BM25) retrieval** improves accuracy by leveraging both **meaning-based** and **keyword-matching** approaches.

By implementing **query fusion**, you will optimize retrieval effectiveness and enhance response quality.

🔷 Steps:

1. **Create separate BM25 and vector-based retrievers**, each with a higher top-k for better fusion.
2. **Combine retrieval scores** using a **weighted fusion approach** (e.g., 0.6 for vector, 0.4 for BM25).
3. **Use a Query Fusion Retriever** to merge results from both retrieval methods.
4. **Create a hybrid query engine** and execute retrieval queries.
5. **Evaluate hybrid retrieval performance** using metrics like **MRR, hit rate, precision, and recall**.

Resources: https://docs.llamaindex.ai/en/stable/module_guides/querying/retriever/

💡 Hints

- **Retrieval Fusion:** Use **QueryFusionRetriever** from **LlamaIndex** to merge results from both retrievers.

📤 Deliverable:

- The final **Jupyter Notebook** with implementation of the hybrid retrieval.
- A **brief analysis (1-2 paragraph)** discussing your implementation and the impact of hybrid retrieval vs. other methods. In your report, provide a nice chart that depicts a **visual summary** of the performance of all four retrieval methods.

📌 Task 5: Let's make our RAG even better! 🚀 [6 points]

🎯 **Objective:** Inspired by your observation, you will pick one aspect of the RAG pipeline and improve it in a meaningful way. This allows for **creativity and deeper engagement** with the system.

📖 **Background:**
Now that you have implemented key components of a RAG pipeline, it's time to **go beyond** and refine one aspect of your choice. This task is open-ended: you should identify a specific weakness or area of improvement and attempt to enhance it. Your goal is to **experiment with novel ideas, evaluate the impact, and justify your changes.**

🔷 **Possible directions for improvement:**

1. **Re-ranking retrieved documents**: Instead of relying on simple top-k retrieval, can you implement a second-stage ranking mechanism to improve the final retrieval quality?
2. **Chunk refinement**: Is there a better way to **dynamically** determine chunk size instead of using a fixed-size approach? Try adaptive chunking based on **document structure** (e.g., breaking at meaningful section boundaries).
3. **Multi-query expansion**: Can you **automatically generate alternative versions** of a user's question to improve retrieval coverage? Consider using LLM-based paraphrasing or query expansion techniques.
4. **Noise reduction in OCR retrieval**: If using OCR, can you **post-process extracted text** to remove artifacts (e.g., formatting errors, hallucinated characters)?
5. **Domain adaptation for embeddings**: Instead of generic sentence embeddings, can you fine-tune or use a domain-specific model (e.g., **biomedical embeddings for medical documents**)?

📤 Deliverable:

- The final **Jupyter Notebook** with your implementation of this improvement.
- A **brief analysis (1-2 paragraph)** discussing what you improved, why it matters, and the before/after performance impact.

📌 Grading Guidelines

Your submission will be evaluated based on the following:

Criteria	Weight	Expectations
Functionality	50%	The feature works correctly, integrates well, and improves the app.
Creativity & Complexity	25%	Unique ideas and technically interesting implementations will be appreciated.
Clarity & Documentation	25%	Code should be clean and well-commented, with clear comments, brief explanation, and screenshots of the output.

🌟 **Bonus:** Exceptional effort, extra polish, or an innovative feature may earn additional credit!

📤 Final Submission

You should submit three files:

✅ **Modified Code** (final version) with all changes, including:

- All modifications made for the five tasks.
- Your final implementation of the RAG pipeline.
- **Alternative submission:** You may also submit a **Google Colab URL** instead of an .ipynb file.
  - Ensure the link is set to **"Anyone with the link can view."**
  - **Do not edit** the file after submission—if the last saved timestamp is later than the deadline, it will be considered a **late submission**.

✅ **my_questions.txt**

- A text file containing the **comprehensive test set** you created for Task 2.

✅ **PDF Report**

- A well-structured report including **all deliverables** for the five tasks.
- Observations, justifications, and screenshots of results.

Note: To ensure proper identification of your submission, please **rename your files** as follows:

- **Jupyter Notebook: HW4_YourName.ipynb** (Replace **YourName** with your actual name).
- **PDF Report: HW4_YourName.pdf** (Replace **YourName** with your actual name).