Performed by Roman Piznak Test Task: Building a Multimodal RAG System

Task Overview

Your task is to design and implement a **Multimodal Retrieval-Augmented Generation (RAG)** System that retrieves relevant news articles from *The Batch*, incorporating both textual and visual data. The system should allow users to input queries, answer questions, retrieve relevant articles, and display any associated media (such as images). The focus is on developing a practical solution that integrates different types of data efficiently and presents the results in a user-friendly format.

Requirements

You are required to build an end-to-end RAG system with the following features and outputs:

1 Data Ingestion and Preprocessing

a. Retrieve news articles and their associated media from *The Batch* and prepare the data for use in your RAG system.

2 Multimodal Database Creation

a. Store and index text and image data to enable efficient retrieval based on user queries.

3 Query System for Retrieval

a. Implement a user interface where users can input queries and receive answers and relevant articles, including text and associated images.

4 Multimodal Integration

- a. Integrate textual and visual data to improve the quality and relevance of the retrieval results.
- b. It is recommended to use GPTs, Claude, or Gemini LLM models.

System Evaluation

a. Evaluate the performance of your RAG system and explain how well it retrieves relevant content based on user queries.

6 Simple UI for Testing and Demo

- a. Create a simple user interface for testing and showcasing your RAG system. You can use Streamlit or a similar tool to build a clean, interactive UI that allows users to input queries and view the retrieved articles and images.
- b. **Requirement**: The UI should allow users to input a query, trigger the retrieval process, display the retrieved news articles alongside any relevant images. The UI should be simple and intuitive to use.

7Optional Deployment

a. Optionally, the system can be deployed on a cloud platform for scalability and accessibility.

Final Submission

Documentation: Provide a report detailing: your approach, the tools and models you selected, the reasoning behind your decisions

Codebase for Submission: You must submit the complete codebase for your RAG system, ensuring it can be easily set up and tested.

Requirements:

- The codebase should be organized and include clear documentation: A README file with instructions for setting up the system, installing dependencies, running the application
- If a specific environment or libraries are required, provide: environment setup instructions, or requirements.txt file (or equivalent for your chosen platform)
- The codebase should include all relevant scripts, such as data ingestion, data processing, user Interface (UI), any additional components (e.g. evaluation scripts, utilities), the project should be structured for easy review and testing by others.

Demo: Include a video with a demo showcasing how users interact with your system, using the simple UI to input queries and view the results.

Solution overview

In this project, I implemented a Multimodal Retrieval-Augmented Generation (RAG) System that retrieves relevant news articles from <u>The Batch</u>, combining both textual and visual data. The system allows users to input natural language queries through a simple **Streamlit**-based user interface, performs semantic retrieval using **FAISS** and **HuggingFace** embeddings, and displays the retrieved articles along with associated images. To further enhance the relevance and quality of generated responses, the system integrates **ChatGPT-40**, which is used to generate answers and summaries based on the retrieved context. The solution also includes an evaluation pipeline to measure retrieval performance (Hit@k, MRR) and is designed with scalability and potential cloud deployment in mind.

1 Data Ingestion and Preprocessing

a. Retrieve news articles and their associated media from *The Batch* and prepare the data for use in your RAG system.



Picture 1 - Categories of articles

Important: 'Weekly Issues' is not a separate category. It's just a set of all articles grouped by week.

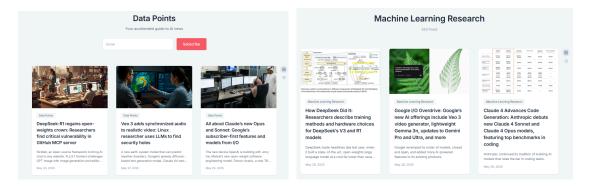
If we will start to explore categories, we will find out, that their endpoints have the same form:



General form is:

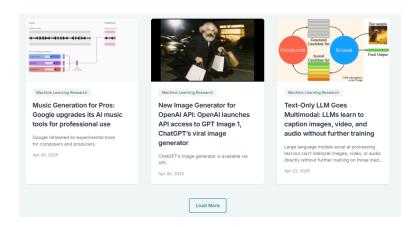
https://www.deeplearning.ai/the-batch/tag/{some_tag}

That is, we already understand which pages we need to scrap. Let's take a closer look at the structure of these pages.



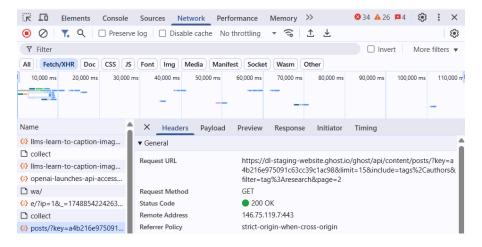
Picture 3 - Pages of different categories

Each page has limited amount of articles, to see more there is a separate button:



Picture 4 - Page with "Load More" button

If we will open "Network" in DevTools and try to trigger that button, we will see the GET request:



Picture 5 - Network with GET request

So, we can call this GET request manually in code and it will return us the articles. Instead of parsing the full HTML pages, I chose to use this clean and structured API (used by the site's frontend itself). This approach ensures stability, easier maintenance, and consistent access to well-structured article data.

Information for the article consists of:

```
"text": "some text"

"metadata": {

    "title": "some title",

    "url": "https://some//url",

    "published_at": "2025-05-28",

    "tags": [

        "Letters",

        "Tech & Society",

        "May 28, 2025"

],

"feature_image": "https://link//to_the_image"}
```

Storing each article as "text" + "metadata" provides a clean separation of content for downstream processing:

- "text" used for embedding and semantic search in the RAG pipeline
- "metadata" used in the UI (title, URL, published date, associated image)

We store all the data in json format. That is how it looks like in result:

Picture 6 - Example data

Important: Some articles have audio transcription that will be scrapped like this:

Loading the Elevenlabs Text to Speech AudioNative Player...

So, I just deleted this text in the start of the articles.

2 Multimodal Database Creation

a. Store and index text and image data to enable efficient retrieval based on user queries.

At this stage, my goal was to create an efficient retrieval database that allows the system to quickly and accurately return the most relevant articles in response to user queries. For this purpose, I chose to use FAISS as the core vector store technology. FAISS is a fast and highly optimized library for similarity search in large-scale dense vector spaces. It is widely used in production RAG systems and supports incremental updates and persistence, which is important for maintaining an up-to-date database of articles.

To represent articles semantically, I used HuggingFaceEmbeddings with the all-MiniLM-L6-v2 model. This model offers an excellent balance between embedding quality and speed. It produces compact embeddings

suitable for semantic search, and it is lightweight enough to run on standard hardware.

Each article is processed by combining its "title" and "text" fields into a single text block, which is then split into overlapping chunks using a CharacterTextSplitter, with such parameters:

CHUNK_SIZE=1000 CHUNK_OVERLAP=200

This combination was optimal due to the articles sizes (each \sim 3-4 k).

This is a common best practice for RAG pipelines — it allows the system to retrieve the most relevant part of an article, rather than entire long documents. In addition to the text embeddings, I store key metadata for each chunk: the article URL, title, and associated image link. This enables the system to display not just the retrieved text, but also the relevant visual content, which is essential for multimodal integration in the user interface.

A key feature of my implementation is the ability to incrementally update the vectorstore. The system keeps track of which articles have already been indexed and processes only new content when run again. This design makes the database scalable and efficient to maintain over time.

3 Query System for Retrieval

a. Implement a user interface where users can input queries and receive answers and relevant articles, including text and associated images.

For the user interface, I implemented a simple **Streamlit**-based web app. It allows the user to enter a natural language query and view retrieved articles with both text and associated images.

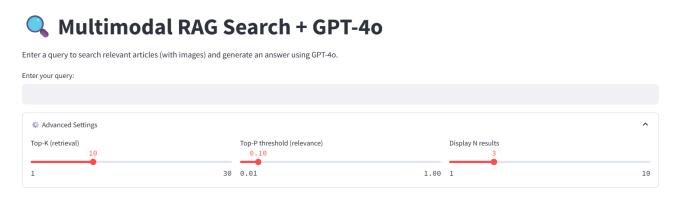
The system performs semantic search over the FAISS vectorstore, using all-MiniLM-L6-v2 embeddings. To improve result ranking, I integrated a CrossEncoder reranker (ms-marco-MiniLM-L-6-v2), which ensures that the most relevant results are shown first.



Picture 7 - Home page

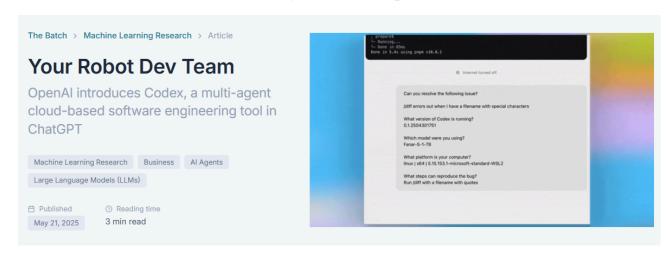
To make the system more flexible, I added an "Advanced Settings" panel in the UI, where users can adjust key parameters:

- Top-K number of initial candidates retrieved from FAISS
- Top-P threshold minimum relevance score required for display
- Number of displayed results how many top results to show



Picture 8 - Advanced settings

Let's demonstrate work of the system. Let's open some article:



Picture 9 - Article's title and images

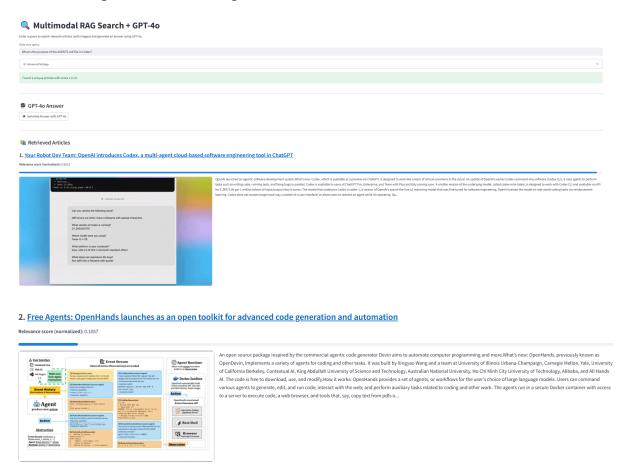
Let's highlight a certain paragraph, for example:

"A file called AGENTS.md can modify agent behavior (like a README.md file, but for agents instead of humans). This file can specify how and when an agent makes pull requests, provide guidelines for coding style, or list tests to verify generated code."

Let's formulate a question that does not refer to this paragraph in the same words, but is relevant in general:

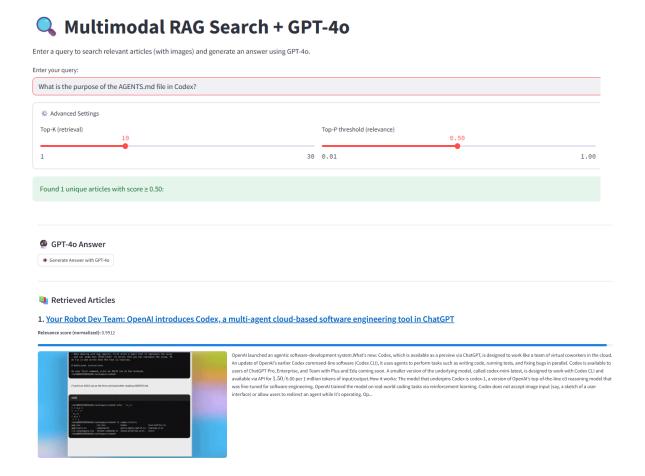
"What is the purpose of the AGENTS.md file in Codex?"

Let's perform retrieving:



Picture 10 - Retrieving results

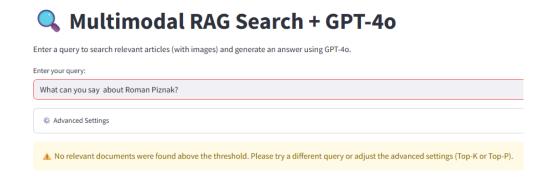
As we can see, our related article is on the first place, but there is also another less related article. We can filter these articles by changing relevance score (we made 0.5, before was 0.1):



Picture 10 - Retrieving results

Now we see that there is only one relevant article.

Cases where there are no relevant documents are also processed:



Picture 11 - No relevant articles case

4 Multimodal Integration

a. Integrate textual and visual data to improve the quality and relevance of the retrieval results.

In my system, multimodal integration is implemented during the construction of the input prompt for the LLM. For each retrieved relevant document from the FAISS vectorstore:

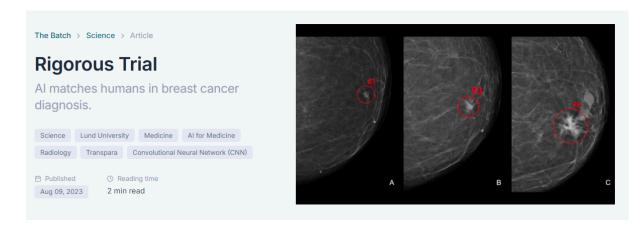
- The article text is included as part of the GPT-40 prompt.
- If the document contains an associated image, the image is also included in the prompt in the form of an image_url object.

As a result, the LLM generates answers based on both textual and visual information, which improves the quality, accuracy, and relevance of the response.

b. It is recommended to use GPTs, Claude, or Gemini LLM models.

In this implementation, the system uses **GPT-40**, a multimodal version of GPT that natively supports text + image inputs.

Let's demonstrate the work. First of all, I gave ChatGPT article text:



Picture 12 - Title of the article and some info

Then I asked for creating semi-hard query, here are results:



How can Al help improve breast cancer diagnosis in clinical practice?

Context:

A randomized, controlled trial conducted at Lund University in Sweden tested whether Al could safely assist radiologists in diagnosing breast cancer from mammograms. The experimental group used an Al system, Transpara, to score mammograms and mark potential tumor locations, which were then reviewed by radiologists. The study showed that Al-assisted screening matched the cancer detection rate of traditional double-reading by two radiologists while reducing radiologist workload by over 40%, without increasing false positives. The results suggest that Al could help doctors evaluate more cases faster, which is especially valuable given the global shortage of radiologists. However, the system still requires further study and validation across more diverse patient populations before widespread clinical adoption.

Answer:

Al can improve breast cancer diagnosis by helping radiologists process more cases efficiently without compromising accuracy. It can highlight potential cancerous areas on mammograms, assist in prioritizing high-risk cases, and reduce the workload on medical staff. This enables faster screenings and could lead to earlier detection, which improves treatment outcomes. However, additional validation is needed to ensure Al systems perform reliably across different clinical settings and patient groups.

Then I tried to do it with my system:



Multimodal RAG Search + GPT-40

Enter a query to search relevant articles (with images) and generate an answer using GPT-4o.

Enter your query:

How can AI help improve breast cancer diagnosis in clinical practice?

Advanced Settings

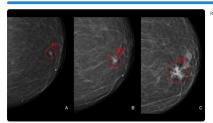
Found 3 unique articles with score ≥ 0.10:

Picture 13 - Retrieving results



1. Rigorous Trial: AI matches humans in breast cancer diagnosis.

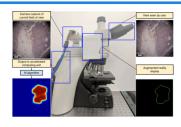
elevance score (normalized): 0.9192



Rigorous Trial: AI matches humans in breast cancer diagnosis....

2. Seeing Cancer

Relevance score (normalized): 0.7891



Microscopes outfitted with Al-driven augmented reality could improve the accuracy of cancer diagnoses. What's happened: Google Health developed an attachment for analog microscopes that outlines signs of breast and prostate cancer in real time. How it works: A computer vision system spots cancer in a cell side, while augmented-reality tech superimposes the Al's prediction over the slide at around 27 frames per second. The developers combined the Inception V3 image classifier with a fully convolutional neural network, which allowed the system to recognize tumorous patterns much faster A camera captures a head-on view of the slide and projects it, overlaid with the Al prediction, into the microscope eypiece. Behind the news: Pathologists use microscopes to measure tumor size relative to nearby lymph nodes and to count the number of cells nearing or undergoing mitosis. That information tells them how aggressively a pathent's cancer is spreading. Why it matters: Interpreting cell sides is su...

3. Standards for Testing Medical AI: Consort-AI and Spirit-AI for clinical trial quality

Relevance score (normalized): 0.373



New guidelines for reporting on experiments with medical AI aim to ensure that such research is transparent, rigorous, and reliable.What's new: Spirit-AI and Consort-AI are complementary protocols designed to improve the quality of clinical trials for AI-based interventions. How it works: The guidelines are intended to address concerns of doctors, regulators, and funders of technologies such as the Google tumor detector shown above. Spirit-AI allos for clinical trials to beserve established best practices in medice. For example, it asks that researchers clearly explain the AI's intended use, the version of the algorithm to be used, where its input data comes from, and how the model would contribute to doctors' decisions. Consort-AI aims to ensure that such studies are reported clearly. Its provisions largely mirror those of Spirit-AI.Both sets of recommendations were developed by over 100 stakeholders around the world. Researchers at University of Birmingham and University Hospitals Bir...

Picture 14 - Retrieved articles

We can see that our relevant article is on the first place, Let's generate and answer based on the retrieved documents:



- 1. Accuracy and Speed: Al systems, such as those tested in rigorous trials, can match human performance in diagnosing breast cancer, potentially increasing the accuracy and speed of diagnoses
- 2. Augmented Reality Microscopes: Al-driven augmented reality can enhance traditional microscopes, allowing pathologists to see Al predictions overlaid on cell slides in real time. This can improve the accuracy of identifying cancerous cells and assessing tumor spread.
- 3. Standardized Testing and Reporting: New guidelines like Spirit-Al and Consort-Al ensure that Al tools are tested and reported with transparency and rigor. This helps integrate Al into clinical practice reliably, ensuring that Al tools are used effectively and safely

These advancements can lead to more precise and timely diagnoses, ultimately improving patient outcomes in breast cancer treatment

Picture 15 - Answer, generated based on the context

If we analyze the answer, we can say that it is very close to real answer, but there are some additions.

5 System Evaluation

a. Evaluate the performance of your RAG system and explain how well it retrieves relevant content based on user queries.

To evaluate the quality of the built system, I decided to create a test dataset. To do this, I used ChatGPT, giving it a prompt:

Hello! You are a smart system to help create a dataset for the RAG quality assessment.

Your task: based only on the given context: (Paste context here)

Create some semi-hard queries for this article. Return the results in the format:

{generated query}: (Paste url here)

This was done for 20 different articles, creating a dataset of 100 instances. This is how it looks like:



Picture 16 - Dataset view

To evaluate the quality of the retrieval component of the system, I used standard information retrieval metrics, which are well-suited for RAG evaluation:

- **Hit@K** This metric measures whether the correct document (ground truth URL) appears among the top K retrieved results. I evaluated Hit@1, Hit@3, Hit@5, and Hit@10 to see how well the system surfaces the relevant document at different cutoff levels.
- Mean Reciprocal Rank (MRR) This metric captures not just whether the correct document was retrieved, but also how high it was ranked. It assigns higher scores when the correct document appears closer to the top of the list, thus giving a more nuanced picture of retrieval quality.

I evaluated the system in two configurations:

- 1. Pure FAISS retrieval This served as a baseline to assess how well the embedding model (all-MiniLM-L6-v2) + FAISS index performed on its own.
- 2. FAISS retrieval + CrossEncoder reranker To measure the improvement from applying a neural reranker (cross-encoder/ms-marco-MiniLM-L-6-v2), which can refine the ranking of candidate documents with deeper semantic understanding.

I got these results:



Picture 17 - Achieved results

The evaluation shows that the system performs quite well overall. Even the base FAISS retrieval provides solid results (Hit@1 above 60%, MRR above 0.7), and adding the CrossEncoder reranker significantly improves top-1 accuracy and overall ranking quality (MRR close to 0.8). This confirms that using a neural reranker is effective — it helps promote the most relevant documents to the top of the results list, optimal k is 3.

6 Simple UI for Testing and Demo

As you have already seen, this point has been fully implemented.

7Optional Deployment

I ran out of tokens, so there was no point in performing the deployment