RAG-Based Machine Learning QA System

Ahmed Mostafa Ismael Omar Gamal Abdel Aziz
Ahmed Ehab
Supervisor: Eng. Rashed Abu Zaid

May 5, 2025

1 Introduction

This project implements a Retrieval-Augmented Generation (RAG) system for answering questions about machine learning. Our system combines web scraping capabilities with state-of-the-art NLP models to provide accurate, context-aware answers. The system automatically collects content from authoritative sources, processes it into searchable knowledge chunks, and uses neural retrieval-generation mechanisms to answer user queries. Developed as a university project, this implementation demonstrates practical applications of modern NLP techniques while remaining fully open-source and locally executable.

2 System Architecture

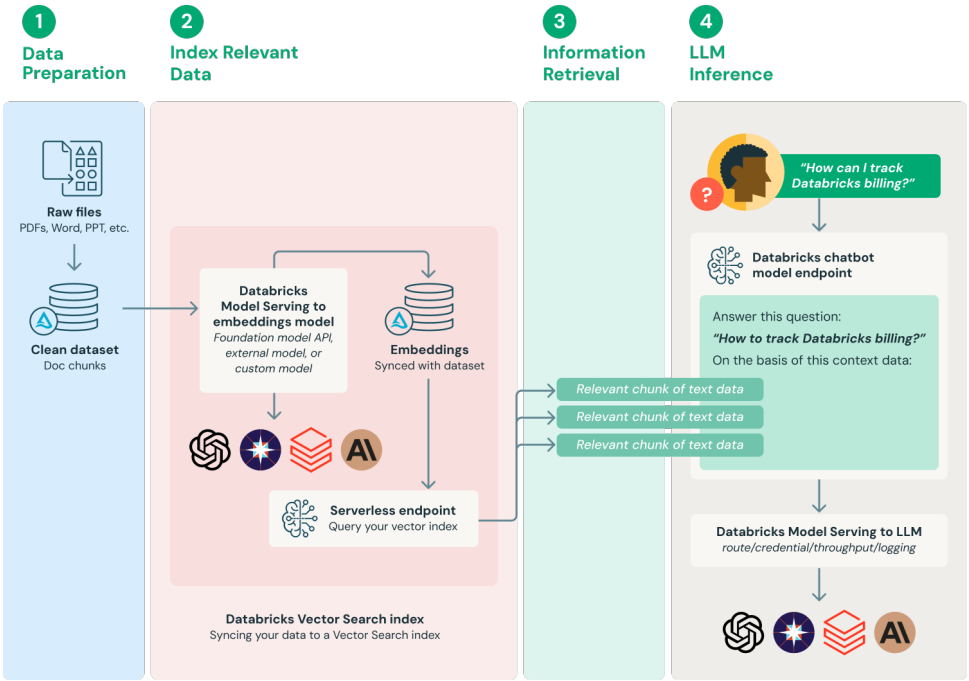

Figure 1: System Architecture Diagram

The system comprises four main components:

2.1 Web Scraper

- Automated content collection from specified URLs
- CSS-based content extraction with noise removal
- Text cleaning and normalization

- Minimum word threshold (30,000 words)

2.2 Text Processor

- Sentence tokenization using NLTK
- Dynamic chunking (200 words/chunk)
- Context-aware segmentation

2.3 Retrieval Engine

- FAISS vector database
- Sentence-BERT embeddings (all-MiniLM-L6-v2)
- Nearest neighbor search

2.4 Generation Model

- Zephyr-7B- model
- Instruction-tuned for QA
- Context-aware prompting

3 Tools & Technologies

- **Python**: Core programming language
- **BeautifulSoup/Requests**: Web scraping
- **NLTK**: Text processing
- **Sentence-Transformers**: Embeddings
- **FAISS**: Vector search
- **HuggingFace Transformers**: Zephyr-7B model
- **Gradio**: Web interface
- **PyTorch**: Neural network backend

4 What is RAG?

Retrieval-Augmented Generation (RAG) is a Generative AI (GenAI) architecture that augments a Large Language Model (LLM) with fresh, trusted data retrieved from authoritative internal knowledge bases and enterprise systems, to generate more informed and reliable responses.The acronym "RAG" is attributed to the 2020 publication, "Retrieval-Augmented Generation for Knowledge-Intensive Tasks", submitted by Facebook AI Research (now Meta AI). The paper describes RAG as "a general-purpose fine-tuning recipe" because it's meant to connect any LLM with any internal or external knowledge source.As its name suggests, retrieval-augmented generation inserts a data retrieval component into the response generation process to enhance the relevance and reliability of the answers.The retrieval model accesses, selects, and prioritizes the most relevant documents and data from the appropriate sources based on the user's query, transforms it into an enriched, contextual prompt, and invokes the LLM via its API. The LLM responds with an accurate and coherent response to the user. This is LLM AI learning in action.A helpful RAG analogy is a stock trader using the combination of publicly available historical financial information and live market feeds.Stock traders in a firm make investment decisions based on analyzing financial markets, industry, and company performance. However, to generate the most profitable trades, they access live market feeds and internal stock recommendations provided by their firm.In this example, the publicly available financial data used is the LLM, and the live data feeds and firm's internal stock recommendations represent the internal sources accessed by the RAG. Retrieval-Augmented Generation (RAG) combines information retrieval with text generation:

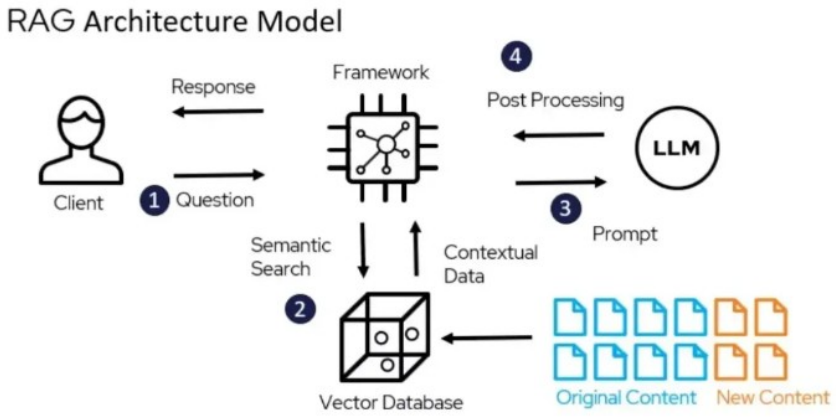

Figure 2: RAG Architecture (Source: HuggingFace)

- **Retrieval Phase**: Finds relevant context from knowledge base
- **Generation Phase**: Produces answer using retrieved context

Advantages over standard transformers:

- Accesses up-to-date external knowledge
- Reduces hallucination
- Improves answer grounding

5 Model Architecture

5.1 Embedding Model

- all-MiniLM-L6-v2 (384-dim embeddings)
- 6-layer Transformer
- 22M parameters
- Optimized for semantic search

6 Model Generation

Model Card for Zephyr 7B β

Zephyr is a series of language models that are trained to act as helpful assistants. **Zephyr-7B-β** is the second model in the series, and is a fine-tuned version of `mistralai/Mistral-7B-v0.1` that was trained on a mix of publicly available, synthetic datasets using *Direct Preference Optimization (DPO)*. We found that removing the in-built alignment of these datasets boosted performance on MT Bench and made the model more helpful. However, this also means that the model is likely to generate problematic text when prompted to do so. For more details, refer to the technical report.

Model Description

- **Model type**: A 7B parameter GPT-like model fine-tuned on a mix of publicly available, synthetic datasets.
- **Language(s)**: Primarily English
- **License**: MIT
- **Fine-tuned from model**: `mistralai/Mistral-7B-v0.1`

Model Sources

- **Repository**: `https://github.com/huggingface/alignment-handbook`
- **Demo**: `https://huggingface.co/spaces/HuggingFaceH4/zephyr-chat`
- **Chatbot Arena**: `http://arena.lmsys.org`

Performance

At the time of release, Zephyr-7B-β was the highest ranked 7B chat model on the MT-Bench and AlpacaEval benchmarks:

Model	Size	Alignment	MT-Bench (score)	AlpacaEval (win rate %)
StableLM-Tuned-α	7B	dSFT	2.75	-
MPT-Chat	7B	dSFT	5.42	-
Xwin-LMv0.1	7B	dPPO	6.19	87.83
Mistral-Instructv0.1	7B	-	6.84	-
Zephyr-7B-α	7B	dDPO	6.88	-
Zephyr-7B-β	7B	dDPO	**7.34**	**90.60**
Falcon-Instruct	40B	dSFT	5.17	45.71
Guanaco	65B	SFT	6.41	71.80
Llama2-Chat	70B	RLHF	6.86	92.66

Table 1: Performance comparison of Zephyr-7B-β with other models

7 RAG vs Traditional Transformers

Aspect	RAG	Vanilla Transformer
Knowledge Source	External documents	Pretrained parameters
Information Freshness	Dynamic updates possible	Fixed at training time
Hallucination Risk	Reduced via grounding	Higher
Compute Requirements	Moderate	Very High

8 Improvement Roadmap

1. Implement recursive URL discovery for web scraping
2. Add semantic chunking with overlap
3. Experiment with hybrid (vector + keyword) search
4. Fine-tune embedding model on domain data
5. Implement query expansion
6. Add citation references to answers
7. Develop evaluation metrics (BLEU, ROUGE)