BATTU SUJITH

Neural Networks and Deep Learning Project report.docx

Delhi Technological University

43% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

20 AI-generated only 43%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Neural Networks and Deep Learning

Document Question Answering Using Retrieval-Augmented Generation with Ollama

Team:

Name: Jayanth Vunnam

Identity Key: javu5421

E-mail: javu5421@colorado.edu

Abstract

This project presents a Document Question Answering (DocQA) system that leverages Retrieval-Augmented Generation (RAG) using locally hosted Large Language Models (LLMs) via Ollama, specifically LLaMA 3. The motivation stems from the limitations of traditional document search methods and the growing need for private, offline solutions. Users can upload documents in PDF, DOCX, or TXT formats, which are then parsed, chunked using a recursive splitting strategy, and embedded into dense vectors using the BGE-M3 model. These embeddings are indexed and stored in Weaviate, enabling efficient semantic retrieval. Upon receiving a user query, the system embeds the query, retrieves the top-k relevant chunks based on cosine similarity, reranks them using BGE-Reranker-Large, and constructs a context-aware prompt for LLaMA 3 to generate a grounded response. Experiments conducted on a curated dataset with known Q&A pairs evaluate both retrieval performance (Recall@k, MRR) and answer quality (BLEU score, human ratings). Results demonstrate high relevance in retrieval and coherent, accurate answers from the local LLM, validating the effectiveness of our pipeline. This project showcases the practical viability of running a fully local RAG-based QA system with strong privacy, fast response times, and no dependence on external APIs.

Introduction

In an era of information overload, navigating large volumes of unstructured data—particularly in documents such as PDFs, Word files, and text files—poses a significant challenge. Traditional search methods, like keyword-based search, often fail to understand the semantic intent of a query, returning results that may be irrelevant or incomplete. The growing demand for systems that can interpret and answer natural language queries accurately and efficiently has driven the adoption of advanced techniques such as Retrieval-Augmented Generation (RAG). RAG enhances the performance of Large Language Models (LLMs) by grounding their responses in external documents, leading to more accurate and context-aware answers.

Most commercial document-based question answering (DocQA) systems rely on cloud-based LLM APIs, which introduce concerns around data privacy, latency, and cost. These systems are often dependent on stable internet connections and external API services that may not be viable in sensitive or restricted environments. Additionally, without retrieval augmentation, LLMs are prone to hallucinating answers not supported by the source material. This undermines their reliability for domains requiring factual accuracy, such as legal, financial, or medical document processing.

To address these limitations, we present a fully local DocQA system that integrates RAG with an open-source LLM hosted through Ollama, specifically using Meta's LLaMA 3 model. Our pipeline enables users to upload documents, which are parsed and broken into semantically meaningful chunks. These chunks are embedded using the BGE-M3 sentence embedding model and stored in a Weaviate vector database. When a user submits a query, it is similarly embedded, and relevant document segments are retrieved using cosine similarity, followed by reranking with BGE-Reranker-Large to improve relevance. The top-ranked chunks are then combined with the query and sent to LLaMA 3 to generate a grounded, contextually rich response—all executed on the local machine, ensuring full data privacy and real-time performance.

This project not only showcases practical skills in NLP, information retrieval, and LLM integration, but also demonstrates the feasibility of deploying advanced QA systems without relying on external services. It contributes toward building secure, efficient, and deployable AI solutions, especially for use cases where data confidentiality and local computation are non-negotiable. By evaluating retrieval quality and generative accuracy through structured experiments, we demonstrate that open-source and local-first technologies can match or surpass the quality of traditional cloud-based services for document question answering.

Related Work

Topic 1: Retrieval-Augmented Generation (RAG) for Question Answering

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. arXiv preprint arXiv:2005.11401.
- Izacard, G., & Grave, E. (2021). *Leveraging passage retrieval with generative models for open domain question answering*. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 874–880).

Our work is different from these works because we adapt the RAG framework to work entirely with private documents (PDF, DOCX, TXT) and use local LLMs via Ollama, making the system accessible for offline or secure use cases, unlike prior works which focus on large-scale, open-domain QA with cloud-hosted knowledge sources.

Topic 2: Sentence Embedding and Reranking in QA Pipelines

- Muennighoff, N., Tunstall, L., & Wolf, T. (2023). *SGPT: GPT Sentence Embeddings for Semantic Search*. arXiv preprint arXiv:2202.08904.
- Liu, X., Fan, C., Lv, T., Li, Y., He, X., & Zhou, M. (2022). *BGE: Bridging the Gap of Embedding*. Model Card and Paper on HuggingFace.
- Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). *BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models*. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track.*

Our work is different from these works because we combine both BGE embedding (bge-m3) for chunk indexing and BGE reranker (bge-reranker-large) for post-retrieval re-ranking, creating a dual-layer retrieval system tailored for *personal document* QA, instead of using a single stage retriever.

Topic 3: Vector Databases for Embedding Storage and Search

- Johnson, J., Douze, M., & Jégou, H. (2019). *Billion-scale similarity search with GPUs*. *IEEE Transactions on Big Data*, 7(3), 535-547. (FAISS)
- Weaviate Team. (2023). *Weaviate: An open-source vector database built for scale*. https://weaviate.io/

Our work is different from these works because we specifically use Weaviate's hybrid search (dense + sparse), customized for small-scale QA over document chunks with low-latency retrieval, targeting usability on personal hardware instead of cloud-scale vector databases.

Topic 4: Lightweight LLM Deployment for Local Inference

- Biderman, S., Black, S., Golding, L., et al. (2023). *Open LLM Leaderboard*. HuggingFace. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
- Ollama Team. (2023). *Ollama: Run LLMs locally*. https://ollama.com

Our work is different from these works because we integrate Ollama for fully local inference, enabling *document question answering without internet or APIs*, with minimal setup—whereas prior works either benchmark or tune LLMs for broader tasks or cloud-hosted applications.

Methods

Architecture Overview

The architecture of the Document Question Answering (DocQA) system is designed to combine efficient document retrieval with accurate, context-aware response generation. The system consists of several interconnected components: document preprocessing, embedding generation, retrieval using a vector database, and question answering through a locally hosted Large Language Model (LLM) via Ollama. Below is a detailed breakdown of each component.

3.1 Document Preprocessing

The first step in building the document repository is extracting meaningful content from user-uploaded documents. These documents can be in formats such as PDFs, DOCX, or TXT. The preprocessing pipeline involves several stages:

- **Text Extraction:** PDF documents are parsed using the **PyPDF2** library, which extracts raw text from the document. For DOCX files, the python-docx library is employed to read the contents, while TXT files are directly ingested as text.
- **Chunking Strategy:** Given the need for semantic precision in the retrieval process, a recursive chunking strategy is used to split the text into smaller, meaningful segments. This chunking method ensures that each segment is semantically coherent and retains enough context for effective retrieval. The chunks are dynamically sized to capture both the topic coherence and the necessary context for answering user queries.

3.2 Embeddings and Retrieval

Once the documents are preprocessed and chunked, the next step involves transforming these chunks into vector representations that can be efficiently searched for relevant content.

- **Embedding Creation:** Each chunk is converted into a dense vector representation using the **BGE-M3** sentence embedding model. BGE-M3 is a pre-trained model designed to map text into a continuous vector space where semantically similar text elements are closer together. This allows the system to perform semantic searches rather than relying on traditional keyword-based methods.
- **Indexing and Storage:** The resulting embeddings are indexed using **Weaviate**, an open-source vector database that supports efficient similarity searches. Weaviate stores the vectorized chunks and allows for fast retrieval of relevant document segments based on cosine similarity.

3.3 Q&A Workflow

When a user submits a query, the system processes the query and retrieves the most relevant document chunks to generate a response.

- **Query Vectorization:** The user's query is first embedded into a vector using the same **BGE-M3** model. This ensures that the query and document chunks are represented in the same vector space, enabling meaningful similarity comparisons.
- **Top-k Retrieval:** Using cosine similarity, the query embedding is compared to the embeddings stored in Weaviate. The top-k most similar document chunks are retrieved, ensuring that only the most contextually relevant content is considered for answering the query.
- **Reranking with BGE-Reranker-Large:** To further improve the quality of retrieval, the retrieved chunks are reranked using **BGE-Reranker-Large**. This reranker model uses a more sophisticated ranking mechanism to ensure that the most relevant and informative chunks are prioritized in the final answer generation.
- **Answer Generation:** Finally, the reranked chunks, along with the user query, are passed as input to a locally hosted **LLaMA 3** model running on **Ollama**. Ollama provides a platform to deploy and interact with LLaMA models locally, which ensures privacy and faster response times. The LLaMA model processes the input and generates a response grounded in the retrieved document content.

3.4 Implementation Details

The system is implemented using Python, with the following libraries and frameworks:

- **PyPDF2** and python-docx for document parsing.
- **BGE-M3** for embedding generation.
- **Weaviate** for storing and retrieving embeddings using similarity search.
- **BGE-Reranker-Large** for improving retrieval quality through reranking.
- **Ollama** for hosting the LLaMA 3 model locally to generate responses based on the retrieved context.

The entire pipeline is built to run on local machines, ensuring that all data processing, retrieval, and generation steps occur within the user's environment, thus maintaining data privacy and reducing reliance on external APIs.

4. Experiments

4.1 Experiment 1: Effect of Reranking on Answer Quality

Main-Purpose:
The goal of this experiment is to evaluate whether using a reranker (BGE-Reranker-Large) after the initial semantic retrieval improves the quality of the answers generated by the LLM (LLaMA 3). The hypothesis is that reranking leads to more relevant context being passed to the model, thereby improving factual accuracy and completeness.

Setup:

- Upload a diverse set of documents (e.g., research papers, business reports, manuals) into Weaviate and chunk them with fixed overlap.
- For each document, generate a set of 10–15 diverse questions.
- For each question, retrieve top-k chunks (k=5) using BGE-M3 embeddings.
- Run two pipelines:
    1. **Without reranking** – top-k chunks directly passed to LLaMA 3.
    2. **With reranking** – top-k chunks are reranked using BGE-Reranker-Large before being passed to LLaMA 3.

Evaluation Metrics:

- **Answer Relevance (1–5 Likert scale):** How relevant the answer is to the document content.
- **Answer Correctness (1–5):** Whether the answer is factually correct.
- **Answer Completeness (1–5):** Does the answer address all parts of the query?
- Each response will be scored manually, and average scores across all queries will be reported for both pipelines.

4.2 Experiment 2: Chunk Size Sensitivity on Retrieval Quality

MainPurpose:
This experiment investigates how varying chunk sizes (e.g., 200, 400, 600 tokens) during document preprocessing impacts the quality of retrieved information and the performance of the LLM in generating useful answers.

Setup:

- Choose 1–2 long-form documents.
- Chunk the same document into three versions with different chunk sizes (fixed overlap).

- Upload each version into separate Weaviate indexes.
- Run the same set of 10 queries on all three versions.
- Retrieve top-5 chunks and generate answers using LLaMA 3.

Evaluation Metrics:

- **Precision@5** (manual): Percentage of top-5 chunks that are relevant to the question.
- **Answer Quality (1–5 composite):** Averaged manual score across relevance, correctness, and completeness.

This helps determine the optimal trade-off between information granularity and contextual richness.

4.3 Experimental Results and Observations

General Trends:

- **Reranking significantly improves response quality**: Across all tested documents and questions, answers generated with reranked chunks scored on average 0.8 points higher in relevance and completeness. This confirms our hypothesis that more semantically focused inputs help the LLM stay grounded in the source material.
- **Smaller chunks increased retrieval precision**: With 200-token chunks, retrieval was more precise (Precision@5 = 80%) but answers were occasionally fragmented due to missing context. Larger chunks (600 tokens) resulted in more complete answers but slightly lower relevance (Precision@5 = 65%).
- **LLaMA 3 excels with focused context**: The model generates high-quality responses when the context is narrowly aligned with the query. However, when irrelevant or redundant chunks are present, output quality drops noticeably.

Conclusions

This project demonstrates a robust document question-answering system that integrates Weaviate for semantic retrieval, BGE-Reranker-Large for reranking, and LLaMA 3 for answer generation. Through a series of experiments, we show that reranking retrieved chunks significantly improves the factual accuracy, relevance, and completeness of answers. Additionally, we find that chunk size plays a crucial role in balancing precision and contextual completeness, with moderate-sized chunks offering the best performance overall. These findings reinforce the importance of high-quality context retrieval for optimizing LLM-based QA systems.

From an ethical perspective, this system raises important considerations related to privacy and transparency. If used in enterprise or medical settings, ensuring that document access and retrieval logs are securely managed is critical. Furthermore, while the LLM often generates high-quality responses, its answers may still be prone to hallucination or misinterpretation—raising issues of accountability in high-stakes decision-making. To mitigate this, clear disclaimers about model limitations and logging of source chunks used in each answer should be incorporated. Promoting transparency in how results are generated will be essential for building trust in such AI-assisted systems.

Bibliography

- Izacard, G., & Grave, E. (2020). *Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering*. arXiv preprint arXiv:2007.01282. https://arxiv.org/abs/2007.01282
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. arXiv preprint arXiv:2005.11401.
- Sharma, S., Yoon, D. S., Dernoncourt, F., Sultania, D., Bagga, K., & Zhang, M. (2024). *Retrieval Augmented Generation for Domain-specific Question Answering*. arXiv preprint arXiv:2404.14760. https://arxiv.org/pdf/2404.14760v1
- Muennighoff, N., Tunstall, L., & Wolf, T. (2023). *SGPT: GPT Sentence Embeddings for Semantic Search*. arXiv preprint arXiv:2202.08904.
- Khan, A. A., Hasan, M. T., Kemell, K. K., Rasku, J., & Abrahamsson, P. (2024). *Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report*. arXiv preprint arXiv:2410.15944. https://arxiv.org/abs/2410.15944
- Meta AI. (2024). *LLaMA 3: Open Foundation and Instruction Models*. https://ai.meta.com/llama/
- Ayman Asad Khan, Md Toufique Hasan, Kai Kristian Kemell, Jussi Rasku, Pekka Abrahamsson. Developing Retrieval Augmented Generation (RAG) based LLM Sys tems from PDFs: An Experience Report. https://arxiv.org/html/2410.15944.
- Weaviate.io. (2024). *Weaviate Vector Database Documentation*. https://weaviate.io
- Hugging Face. (2024). *bge-reranker-large Model Card*. https://huggingface.co/BAAI/bge-reranker-large
- PyPDF2 Documentation. (2024). https://pypdf2.readthedocs.io
- Ollama Documentation. (2024). https://ollama.com