

Maximizing video Data Retrieval Efficiency: Leveraging Lang chain Integration of RAG with Cohere Reranker for Enhance Performance

MUKKU SUMANTH

B-TECH CSE AMITY UNIVERSITY CHHATTISGARH, INDIA

Abstract - This paper addresses cutting-edge artificial intelligence and natural language processing algorithms for extracting and organizing material from videos. Audio from videos is extracted, texts are transcribed, and PDFs are generated. The text is split using PyPDFLoader and Lang Chain's Recursive Character Text Splitter. FAISS and embeddings provide fast similarity searches, while Cohere Reranker uses LLMs to boost search result relevancy. The "command-r" technique in Cohere simplifies the creation of question-and-answer apps. Markdown and qa.invoke are used in the last stage to generate acceptable responses. The results show how well these techniques improve query resolution and multimodal information retrieval.

Key Words: LLM, NLP, Cohere API, RAG

1. INTRODUCTION

Language is vital for connecting with technology, communicating, and expressing oneself. The growing demand for robots to do sophisticated language tasks—such as translation, summarization, information retrieval, conversational exchanges, and so on—has resulted in the creation of generic models. Language models have lately made great breakthroughs, owing mostly to the availability of large-scale training data, increased computing power, and transformers. These developments have enabled the development of LLMs capable of performing a wide range of tasks like human performance, ushering in a drastic change. Modern artificial intelligence systems, known as big language models, can now interpret and create text in a cohesive manner, as well as generalize to a range of tasks (Gao et al.,2023).

1.1 Background

At the 2017 NeurIPS conference, Google researchers revealed the transformer concept in their seminal paper, "Attention Is All You Need." The major goal of this study was to improve the 2014 Seq2seq technology, namely by using the attention mechanism developed by Bahdanau et al. When BERT was originally introduced in 2018, it quickly achieved "ubiquity." BERT is an encoder-only kind, whereas the original transformer has both encoder and decoder blocks (Topal et al.,2021).

Aidan Gomez, Ivan Zhang, and Nick Frosst founded Cohere in 2019. The firm maintains offices in Palo Alto and London, in addition to its Toronto and San Francisco headquarters. Cohere Inc. While decoder-only GPT-1 was released in 2018, GPT-2 was released in 2019 and sparked a lot of attention since OpenAI initially believed it was too powerful to be made available to the public owing to worries about harmful use. GPT-3 went a step farther in 2020, and as of 2024, it is only available via API; there is no way to download and execute the model locally. However, the consumer-facing browser-based ChatGPT, which launched in 2022 and garnered some internet and media hype, piqued the public's interest. The 2023 GPT-4 was dubbed a "holy grail" because to its multimodal capabilities and better accuracy. OpenAI did not publish the high-level architecture or the number of GPT-4 parameters.

Most of the time, competitor language models have caught up with the GPT series—at least in terms of parameter count.

Since 2022, source-available models have been increasingly widespread; originally, this was especially true for BLOOM and LLaMA, although both have restrictions in terms of application area. The Mistral 7B and Mixtral 8x7b models from Mistral AI are available under the more permissive Apache license. According to the LMSYS Chatbot Arena Leaderboard as of January 2024, Mixtral 8x7b is the most powerful open LLM; it outperforms GPT-3.5 but falls short of GPT-4 (Topal et al.,2021).

One of the most powerful AI models to date appears to be Google Gemini Ultra. Gemini 1.5 Pro, on the other hand, competes with the greatest open models, such as Mixtral 8x22B and Llama 3 70B, while trailing just slightly behind the best proprietary models, such as GPT-4o and Claude 3 Opus. Next, Gemini 1.5 Flash surpasses less powerful proprietary versions, such as GPT-3.5 Turbo and Claude 3 Haiku, by a slight margin.

Despite their effectiveness, large language models (LLMs) still have limitations, especially for domain-specific or knowledge-intensive occupations. One disadvantage is that LLMs may produce "hallucinations" when processing queries requiring information that is current or beyond their training data. Retrieval-Augmented Generation (RAG) enhances LLMs by

utilizing semantic similarity computing to extract relevant document chunks from an external knowledge base to overcome hurdles. RAG effectively reduces the risk of providing factually incorrect information by using outside knowledge. The introduction of RAG into LLMs has resulted in widespread deployment, cementing its status as a key technology in the evolution of chatbots and increasing LLMs' suitability for practical applications (Naveed et al.,2023).

1.2 Research Problem

The purpose of this Paper is to develop a RAG application using the Cohere API and the Lang Chain Framework. We first upload a Video URLs. Then, using the RAG Application, we can ask questions and have the application deliver pertinent answers.

1.3 Objectives

Examine a range of presently published papers and develop a new application for more accurate distribution. Checking for limits in rag and resolving them by using advanced techniques to overcome them. Given that it was an LLM, it is critical to develop more advanced and exact ways of chunking and vector storing.

1.4 Overview

We started by explaining the RAG and the history of LLMs, and then we went over previous research on RAG and LLMs in the literature review, which was useful for my study. The techniques section contains simple illustrations that thoroughly show the core application. This project's implanting rag application made use of the Lang chain framework and the Cohere API.

2. LITERATURE REVIEW

The current models of sequence transduction rely on complex recurrent or convolutional neural networks, which include an encoder and a decoder. Furthermore, top-performing models employ an attention mechanism to connect the encoder and decoder. (Vaswani et al.,2017) propose a novel, simple network architecture dubbed the Transformer, which fully eliminates repetition and convolutions in Favor of attention mechanisms. Experiments with two machine translation tasks show that these models are more parallelizable, train significantly quicker, and have superior quality. On the WMT 2014 English to German translation challenge, their model scores 28.4 BLEU, more than 2 BLEU

higher than the current top results, including ensembles. After 3.5 days of training on eight GPUs, our model achieved a new single-model state-of-the-art BLEU score of 41.8 on the WMT 2014 English-to-French translation challenge, at a fraction of the training expenses of the best models in the literature. (Vaswani et al.,2017) demonstrate the Transformer's generalizability to other tasks by successfully applying it to English constituency parsing with both large and small amounts of training data (Vaswani et al.,2017). Large Language Models (LLMs) have extraordinary capabilities, yet they confront challenges such as delusions, outdated knowledge, and opaque, untraceable mental processes. Retrieval-Augmented Generation (RAG), which uses information from other databases, has emerged as a viable option. This increases the generation's trustworthiness and accuracy, particularly for activities requiring a high level of expertise. It also enables the continual integration and refreshment of domain-specific data. RAG synergistically blends large, dynamic datasets from external sources with LLMs' intrinsic expertise. This comprehensive review of research examines the evolution of RAG paradigms, such as the modular RAG, the advanced RAG, and the naive RAG. It extensively investigates the retrieval, generation, and augmentation methods that make up the tripartite foundation of RAG frameworks. The essay gives a complete understanding of the advances in RAG systems by emphasizing the cutting-edge technology included in each of these critical components. This paper includes the most recent assessment framework and benchmark. This essay finishes by describing the current challenges and offering future areas for research and development (Gao et al.,2023).

Recent research has demonstrated that large language models, or LLMs, may do astonishing feats in a variety of fields, including natural language processing. Because of LLMs' success, a substantial amount of study has been conducted in this field. These studies look at a variety of topics, including robotics, datasets, benchmarking, efficiency, multi-modal LLMs, architectural innovations, enhanced training methodologies, context length improvements, fine-tuning, and more. The rapid growth of methodology and the regular breakthroughs in LLM research make it incredibly difficult to comprehend the big picture of these changes.

Given the rapidly developing body of research on LLMs, it is vital that the scientific community has access to a concise yet comprehensive account of the most recent advances in this field. Their paper provides

an overview of studies on a wide range of LLM-related issues. In this self-contained, comprehensive study of LLMs, we examine advanced topics at the cutting edge of LLM research as well as relevant underlying notions. In addition to giving a comprehensive overview, the purpose of this review article is to serve as a fast, all-inclusive resource for researchers and professionals seeking information from in-depth assessments of prior research to advance the area of LLM studies (Naveed et al.,2023)

Large Language Model (LLM) outputs must contain external information, which requires retrieval-augmented generation (RAG). Although the volume of research on RAG is growing, there is a lack of thorough experimental comparisons because most of the work focuses on systematic reviews and comparisons of modern state-of-the-art (SoTA) techniques to their historical equivalents. Our research begins to bridge this gap by assessing the effects of various RAG techniques on retrieval precision and response similarity. We found that LLM reranking and hypothetical document embedding (HyDE) significantly enhance retrieval precision. Multi-query approaches performed badly, with neither Maximal Marginal Relevance (MMR) nor Cohere Rerank showing any obvious improvement over a baseline Naive RAG system. Despite contradictory results on answer similarity, sentence window retrieval proved to be the most efficient for retrieval precision. The study attests to the Document Summary Index's potential as a retrieval tool. All research materials are publicly available for further study through our GitHub repository, ARAGOG. We welcome the community to participate in our exploratory study on RAG systems (Eibich et al.,2024).

It has been proven that huge pre-trained language models may preserve factual information in their parameters and, when modified for future NLP tasks, offer cutting-edge results. Their limited access to and precise manipulation of knowledge causes them to perform worse than task-specific designs on knowledge-intensive activities. Furthermore, providing proof for their decisions and updating their knowledge of the outside world remain unresolved research concerns. So far, academics have only looked at trained models for extractive downstream tasks that use a differentiable access mechanism to explicit non-parametric memory. They look at a general-purpose fine-tuning technique for retrieval-augmented generation (RAG) They provide

RAG models in which a pre-trained neural retriever uses a Wikipedia dense vector index as the non-parametric memory and a pre-trained seq2seq model as the parametric memory. They compared two RAG formulations: one that could use numerous passages for each token and another that depended on the same retrieved passages across the whole output sequence. They outperform task-specific retrieve-and-extract architectures and parametric seq2seq models to establish the state-of-the-art on three open domain QA tasks, as well as refine and assess our models on much knowledge-intensive NLP tasks. Researchers discovered that RAG models create more accurate, diverse, and specific language for language production tasks than a cutting-edge parametric-only seq2seq baseline (Lewis et al.,2020)

3.SYSTEM DESIGN

This section describes each phase that we included in the RAG application. The graphic below depicts the entire process flow, from uploading video files to data retrieval, step by step.

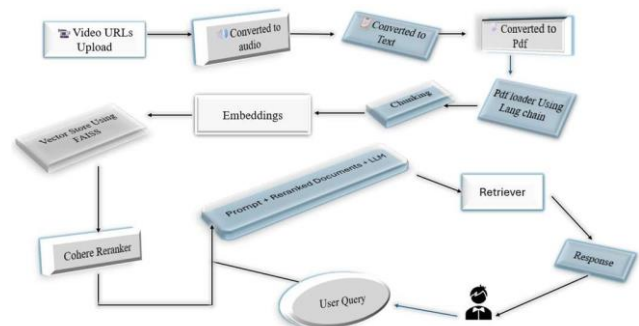



Figure 1:- System Design of Rag Application

3.1 Structuring from videos

The first step is to submit the video URLs. There is no restriction on the number of videos you may post from here. The videos are then saved in the backend for usage in the application's later phases. After saving the video files in the backend, we extracted the audio files from the given video files and saved these audio tracks in the backend for the Rag application's next stage. Following that, text is generated from audio files and saved to the backend. This textual data is then converted to PDF format and fed into the Rag program.

3.2 Pdf Loader Using Lang chain

Adobe developed the Portable Document Format (PDF),

also known as ISO 32000, in 1992 as a mechanism to display documents—including text formatting and images—in a fashion that is independent of operating systems, hardware, or application software (“PDF |  LangChain”).

```
from  
langchain_community.document_loaders  
import PyPDFLoader  
  
loader = PyPDFLoader("example.pdf")  
pages = loader.load_and_split()
```

The code above allows you to easily upload PDF files, and the results are given in an orderly format with documents, pages, and page content for each page with some meta data.

3.3 Chunking

The Recursive Character Text Splitter is an important tool in the Lang Chain toolset for breaking down large texts into consumable, semantically coherent chunks. Because this method retains the text's contextual integrity, it is particularly recommended for early text processing. It operates by recursively partitioning text based on a set of characters that the user defines. This maintains comparable text parts close to one another, preserving the semantic relationship between them.

User-Defined Characters: The user defines a list of characters to be inserted into the splitter. These characters serve as markers for where the text should be separated.

Recursive Splitting: Until the text segments are considered palatable or meet the user's criteria, the operation will continue to divide them into smaller and smaller parts.

Preserving Context: By striving to group relevant text sections together, this strategy best preserves the narrative flow and allows for more skilled processing or analysis.

3.4 Embeddings

Embeddings allow us to describe the meaning of a text as a collection of numbers. This is useful because the text may be utilized for similarity, clustering, classification, and other purposes once it is in this format and compared to other texts. To identify whether two texts are discussing the same themes, we may calculate a similarity score for each embedding using a simple comparison technique. The project collects embeddings for comparable phrases with a high

similarity score for the next phase and discontinues transmitting unrelated phrase embeddings to LLMs via Cohere Rerankers (“Embeddings”).

3.5 FAISS Vector Store

We can quickly locate embeddings of comparable audiovisual content using a library called FAISS (Facebook AI Similarity Search). It provides more scalable similarity search functions and addresses the limitations of traditional query search engines designed for hash-based searches.

FAISS enables us to search multimedia resources in ways that traditional database engines (SQL) cannot or will not allow. It includes memory-speed-accuracy trade off-optimized nearest-neighbour search algorithms for datasets with millions to billions of entries. FAISS aims to deliver cutting-edge functionality at all operating points (“FAISS”).

3.6 Cohere Reranker

Cohere, a leading producer of AI-powered solutions, focuses on machine learning and natural language processing (NLP). The Rerank API endpoint is at the heart of Cohere's services, which employ Large Language Models (LLMs) to improve search results by rearranging documents and doing semantic analysis. This technique contextualizes the meaning of user searches, resulting in more accurate and relevant results than traditional keyword-based search engines.

It would be inefficient to calculate the relevance score for millions of sites, much less a query. Because of this, you usually must combine this with a pre-filtering first-stage retrieval system so that you are presented with a set number of documents to handle. You can utilize lexical search (with Elasticsearch, OpenSearch, Solr, etc.) or embedding-based semantic search (“Say Goodbye to Irrelevant Search Results: Cohere Rerank Is Here”).

3.7 Cohere LLM

Lang Chain, a powerful Python framework, makes it easy to build applications that leverage large language models (LLMs) like BERT, GPT-3, and others. It provides a versatile and modular framework for creating applications that may employ LLMs to do several tasks, including text generation and question answering. In contrast, Cohere is an AI company that provides robust language models through their API. In this example, we will utilize the "command-r" model from Cohere, which

is a retrieval-augmented language model designed specifically for question-answering applications (Kumar,2024).

3.8 QA Retrieval

Question-answering systems (QA systems), often known as Q/A information retrieval, are a type of technology that focuses on understanding the substance of a user's question and extracting relevant knowledge or information from a textual database to deliver concise and correct responses. This QA retrieval consists of an LLM, a chain type, and a first-stage retriever. These components work together to form a chain type, from which relevant data is retrieved.

3.9 Response

Finally, after the relevant document for the input query was retrieved, we utilized the qa.invoke function to help generate organized output. We also used Python's markdown function to generate tidy and structured output.

4. RESULTS AND DISCUSSION

This section discusses the project's outputs or outcomes for each term. We thoroughly detailed each step and its associated results.

4.1 PDF loader

The output from the PDF loader is displayed below. It contains Documents have several more properties that are preserved as metadata, in addition to a page_content property that defines the content to be searched and translated into vectors. The metadata indicates which fields in the original content should be stored as document metadata.

```
[Document(page_content='Transcribed Video Text\n GPT or generative pre-trained transformer is a large language model or an LLM that can generate human-like text. I've been using GPT in its various forms for years. In this video, we are going to number one, ask what is an LLM. Number two, we are going to describe how they work. And then number three, we're going to ask what are the business applications of LLMs. Let's start with number one, what is a large language model? Well, a large language model is an instance of something else called a foundation model. Now, foundation models are pre-trained on large amounts of unlabeled and self-supervised data, meaning the model learns from patterns in the data in a way that produces generalizable and adaptable output. And large language models are instances of foundation models applied specifically to text and text-like things. I'm talking about things like code. Now, large language models are trained on large data sets of text, such as books, articles, and conversations. And look, when we say large, these models can be tens of gigabytes in size and trained on enormous amounts of text data. We're talking potentially petabytes of data here. Now, to put that into perspective, a text file that is, let's say, one gigabyte in size, that can store about 178 million words. A lot of words just in one GB. And how many gigabytes are in a petabyte? Well, it's about one million. Yeah, that's truly a lot of tech. Now, LLMs are also among the biggest models when it comes to parameter count. A parameter is a value the model can change independently as it learns. And the more parameters the model has, the more complex it can be. GPT-3, for example, is pre-trained on a corpus of actually 45 terabytes of data. And it uses 175 billion ML parameters. All right, so how do they work? Well, we can think of it like this. LLM equals three things: data, architecture. And lastly, we can think of it as training. Those three things are really the components of an LLM. Now, we've already discussed the enormous amounts of text data that goes into these things. As for the architecture, this is a neural network. And for GPT, that is a transformer. And the transformer architecture enables the model to handle sequences of data like sentences or lines of code. And transformers have designed to understand the context of each word and a sentence by considering it in relation to every other word. This allows the model to build a comprehensive page 1', metadata={'source': 'C:/Users/LENOVO/Downloads/check/transcribed_text.pdf', 'page': 0}),
```

Figure 2:-PDF upload Lang chain result.

4.2 Retrieving data after Cohere Reranker

First, we given Query as shown in below and based on this query we used “ContextualCompressionRetriever “for retrieving documents. By using cohere reranker

model="rerank-english-v3.0" we got top 3 documents using similarity search.

```
compressed_docs = compression_retriever.invoke(
    "what is the salary of data scientist and tell about data scientist work?"
)
pretty_print_docs(compressed_docs)

Document 1:

Well, if you're wondering, there are various roles offered to a data scientist. Like data analyst, machine learning engineer, deep learning engineer, data engineer, and of course, data scientist. The median base salaries of a data scientist can range from $95,000 to $165,000. So that was

Document 2:

The median base salaries of a data scientist can range from $95,000 to $165,000. So that was about the data science. Are you ready to be a data scientist? If yes, then start today. The world of data needs you. That's all from my side today. Thank you for watching. Comment below the next

Document 3:

Transcribed Video Text
te about data science because we will tell you how does it really work under the hood. Emma is a data scientist. Let's see how a day in a life goes while she's working on a data science project. Well,
```

Figure 3:-Retrieved Result After Using Cohere Reranker.

4.3 Final Response

We eventually communicated with the application's end. Below figure shows final answer to the input query. The outcome was the correct response to the input query, and it looked to be more accurate.

```
In [17]: query = "what is the salary of data scientist and tell about data scientist work?"
response = qa.invoke({"query": query})
result = response["result"]

In [18]: result

Out[18]: 'The median base salary of a data scientist ranges from $95,000 to $165,000. Data scientists work on analyzing data and translating it to inform decision-making. Their roles include data analysis, machine learning, and deep learning roles.'

In [19]: from IPython.display import Markdown as md
md(result)

Out[19]: 'The median base salary of a data scientist ranges from $95,000 to $165,000. Data scientists work on analyzing data and translating it to inform decision-making. Their roles include data analysis, machine learning, and deep learning roles.'
```

Figure 4:- Final Output for given input Query.

5. FUTURE WORK

The GPT-4o model combines natural language processing with image recognition, allowing it to understand and respond to text and visual inputs.

Using a proprietary matching algorithm and the RAG technique, we extend the GPT-4o model's capabilities by exploring our knowledge base for things that complement the stated traits. Our algorithm takes color and style coherence into account to provide relevant suggestions to consumers. We intend to use this notebook to illustrate how these technologies may be utilized to develop a clothing recommendation system. There are various advantages to employing GPT-4o and RAG (Retrieval-Augmented Generation) together: Contextual Understanding: GPT-4o understands the things, situations, and behaviours depicted in the incoming pictures. This allows for more specific and relevant advice or information to be delivered in a range of disciplines, including education, gastronomy, and interior design (“How to Combine GPT4 with Vision with RAG to Create a Clothing Matchmaker App | OpenAI Cookbook”)

RAG incorporates a retrieval component that retrieves a wide corpus of material from a range of

fields using GPT-4's creative abilities. This suggests that the system may provide suggestions or give insights based on a wide range of facts, from historical details to scientific hypotheses.

Customization: The strategy makes it straightforward to tailor various programs to the needs or preferences of a certain user. The system may be tailored to provide personalized experiences, such as generating suggestions based on a user's preferred artwork or providing teaching materials depending on a student's degree of comprehension (“How to Combine GPT4 with Vision with RAG to Create a Clothing Matchmaker App | OpenAI Cookbook”).

6. CONCLUSION

Retrieval-Augmented Generation (RAG) enhances LLMs by retrieving relevant document chunks from an external knowledge base using semantic similarity computing. In our project, we start by uploading one or more video URLs. Next, we extract structural text information. The next step is to divide it into parts. Finally, we embed data using the Cohere embeddings approach. Finally, we utilize the FAISS Vector Store to save vector data. Finally, to increase performance, we employ the Cohere Reranker. We used the Cohere API for everything, even LLM. Eventually, we received relevant data for our input query.

7. REFERENCES

Eibich, Matouš, et al. “ARAGOG: Advanced RAG Output Grading.” *ArXiv.org*, 1 Apr. 2024, arxiv.org/abs/2404.01037. Accessed 22 May 2024.

“Embeddings.” *Cohere AI*, docs.cohere.com/docs/embeddings. Accessed 23 May 2024.

“FAISS.” *Ai.meta.com*, ai.meta.com/tools/faiss/.

Gao, Yunfan, et al. “Retrieval-Augmented Generation for Large Language Models: A Survey.” *ArXiv.org*, 18 Dec. 2023,

[arxiv.org/abs/2312.10997#:~:text=Retrieval%20Augmented%20Generation%20\(RAG\)](https://arxiv.org/abs/2312.10997#:~:text=Retrieval%20Augmented%20Generation%20(RAG)).

Kumar, Lalit. “Building a Retrieval Augmented Generation (RAG) Application with LangChain and Cohere.” *Medium*, 12 May 2024, medium.com/@lalit.k.pal/building-a-retrieval-augmented-generation-rag-application-with-langchain-and-cohere-13ad54c9a361. Accessed 23 May 2024.

Lewis, Patrick, et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” *ArXiv (Cornell University)*, 22 May 2020.

Naveed, Humza, et al. “A Comprehensive Overview of Large Language Models.” *ArXiv.org*, 18 Aug. 2023, arxiv.org/abs/2307.06435.

“How to Combine GPT4 with Vision with RAG to Create a Clothing Matchmaker App | OpenAI Cookbook.” *Cookbook.openai.com*, cookbook.openai.com/examples/how_to_combine_gpt4o_with_rag_outfit_assistant. Accessed 24 May 2024.

“PDF | 📄 LangChain.” *Python.langchain.com*, python.langchain.com/v0.1/docs/modules/data_connection/document_loaders/pdf/. Accessed 23 May 2024.

“Say Goodbye to Irrelevant Search Results: Cohere Rerank Is Here.” *Cohere*, cohere.com/blog/rerank. Accessed 23 May 2024.

Vaswani, Ashish, et al. "Attention Is All You Need."

ArXiv.org, 12 June 2017,

arxiv.org/abs/1706.03762.

Topal, M. Onat, et al. "Exploring Transformers in

Natural Language Generation: GPT, BERT, and

XLNet." *ArXiv:2102.08036 [Cs]*, 16 Feb.

2021, arxiv.org/abs/2102.08036.