Intelligent Semantic Retrieval Generation Web Crawling Chunking (RAG) System for AI

Mrs. Darshika Kelin AR Assistant Professor
Dept. of DSCS, Karunya University, Coimbatore
darshika@karunya.edu

Dhanush Kumar.LS
U.G Scholar,
Dept. of DSCS, Karunya University, Coimbatore
dhanushkumar22@karunya.edu.in

Abstract

In this paper, the complete and scalable Retrieval-Augmented Generation (RAG) system development is presented with a special reference to the trustworthy pipeline for data ingestion and processing. The method we propose provides a systematic approach to solving the problem of the need for current, outside information for large language models (LLMs) to reduce factual differences or "hallucinations." An automated web crawling procedure is the starting point of the system for copying unstructured data from selected online sources. This raw data is further processed by means of careful cleaning to get high-quality, domain specific text and remove noise. The major component of our pipeline is a multi-phase data preparation technique. In the beginning, we apply sophisticated text chunking methods, such as token-based and sentence- based segmentation, to break down large documents into parts that are sensible and pleasant to read. After that, state-of-the-art embedding technique is employed to convert these chunks into number vector embeddings that convey their semantic meaning. By storing the resulting embeddings in a special vector database like Pinecone, FAISS, or ChromaDB, the similarity- based retrieval that is extremely efficient is made possible. The last stage involves the generative AI model, e.g.-GPT that makes it possible to dynamically retrieve the most relevant chunks by utilizing this vector store in response to the user's query. Our Retrieval-Augmented Generation technique ensures that the model's output is both contextually relevant and directly verifiable against the source documents. Our study provides a complete and replicable RAG framework that is an affordable and safe way to produce exact and authorized writing, suitable for a wide range of users from information management to conversational AI.

Keywords

Retrieval-Augmented Generation, Web Crawling, Chunking of Texts, Vector Embeddings, Semantic Search, Large Language

I. Introduction

In addition, it has a next-generation Information technology based on generative AIs that brings a complete picture of acquiring, processing, and analyzing information successfully. It uses the combination of processes: web crawling, data cleaning, text splitting, semantic searching, and Retrieval-Augmented Generation (RAG) to produce accurate and specific to the domain inputs. With the support of libraries such as Beautiful Soup and Scrapy, the data related to the query is scraped, cleaned, and divided into semantic chunks. After that, the chunks are

transformed into embeddings and stored in the vector databases like FAISS or Pinecone. The RAG model extracts the relevant data and integrates it with the Large Language Model (LLM) for producing outputs that are not only accurate but also contextually aware for use cases that include analytics, summarization, and real-time question answering.

II. Literature Review

Retrieval-Augmented Generation (RAG) is a powerful technique that merges document retrieval with generative models to increase the reliability and accuracy of knowledge-intensive NLP usage. The authors of the RAG model, Lewis et al. [1], engaged in Dense Passage Retrieval (DPR) during the retrieval of context-relevant documents prior to the generation of accurate, contextually aware responses. The combination reduces hallucinations and improves factual accuracy, although it is dependent on the retrieval relevance. Subsequently, Izacard and Grave [2] recommended the Fusion-in-Decoder (FiD) model, which includes several retrieved contexts in the decoder to generate more informative, context-rich outputs and surpasses open-domain question-answering at a very high computational cost. Chen et al. [3] further advanced the retrieval efficiency through semantic chunking using FAISS-based embeddings, which resulted in considerably better query performance at the cost of extra preprocessing for noisy data. Tan et al. [4] took the RAG pipeline up a notch by incorporating web crawling, data cleansing, and Chroma DB vector indexing for application in specific domains, while precision depended greatly on the quality of data. Singh and Lin [5] took advantage of RAG with knowledge graphs and Pinecone embeddings to give a boost to the factual coherence and a suppression of hallucination. Likewise, Karpukhin et al. [6] stretched large information retrieval with dual-encoder DPR models and FAISS integration, which lead to a rise in computational cost.

III. Innovation

The project's major innovation is to optimize the early stages of the RAG pipeline, namely data collection and preprocessing, to directly impact output quality. Currently, the majority of RAG systems follow a standardized procedure, which can lead to noise and inefficiency.

1. Generative AI-Enhanced Web Crawling

This is a more sophisticated idea than just a passive web crawler grabbing all the information on the site. With this innovation, the crawler is guided by a generative AI system, and this enables it to:

- Identify and rank the most pertinent information about the task or domain.

- Eliminate issues at their source, e.g., irrelevant content, generic content, and advertisements.

2. Adaptive Chunking Strategy

Fixed-size simple chunks and frequently employed in traditional RAG system(e.g., splitting a text into chunks every 256 tokens). A sentence or paragraph could split as a result, breaking the original context and

hindering recall. The od adaptive chunking is new:

- Segmentation by sentences, paragraphs, or even by identifying significant subjects might be one possible solution. It breaks down the text and meaning to generate semantically coherent chunks.

- Preserves context, ensuring that each section is a meaningful, independent piece of information.

IV. Proposed Methodologies

The proposed methodology aims to develop a Generative AI-driven Crawling and Chunking System that will use Retrieval-Augmented Generation (RAG) to acquire, handle, and produce reliable information extracted from large volumes of unstructured data. The first step is crawling the web employing Scrapy and BeautifulSoup to collect relevant material from scientific papers, blogs, and news sites, while at the same time filtering out ads, scripts, and duplicate posts. The raw data is then processed through a series of steps such as removing HTML tags, stop words, and duplicates, followed by normalization and lemmatization. Later on, semantic text chunking breaks down the purified text into small pieces that are rich in context and in line with the token limitations of Large Language Models (LLMs). These chunks are subsequently transformed into embeddings using techniques like OpenAI Embeddings or Sentence-BERT and deposited in vector databases such as FAISS, Pinecone, or ChromaDB for easy similarity retrieval. The RAG model then, upon receiving queries, retrieves the most relevant chunks and combines them with LLM reasoning to produce accurate, context-aware answers. This architecture can be adapted to different interfaces, which in turn guarantees the processing of information with a high degree of precision, adaptability, and reliability.

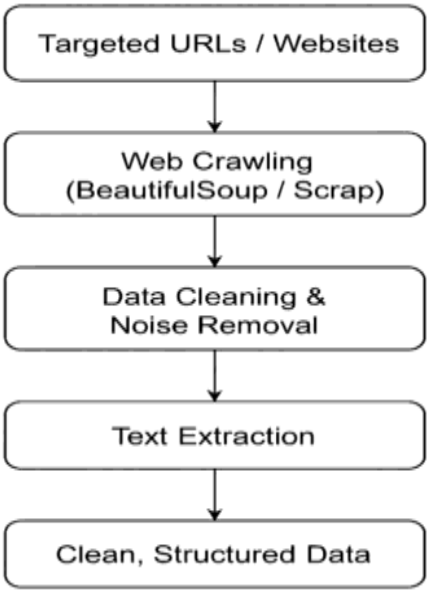

Figure 1: Web Crawling and Data Preprocessing Workflow

The data collection and preprocessing steps in this study are fully explained and depicted in Fig. 1. The first step of this process is web crawling, where the Python-based tools such as BeautifulSoup or Scrapy are used to automatically get the specified information from the sites. After the data is collected, the cleansing process is performed on the unprocessed data, which is done by getting rid of content that does not belong to the main topic such as tags, ads, repetitions, and wording that is not necessary, thus making sure that the pertinent data only is left. Then, the textual normalization methods are employed to give the gathered documents another level of refinement. The process

deals with special characters, takes out unnecessary stopwords, and lowercase all the text, which makes sure that the texts are uniform. After this, the text is broken down into smaller and easier- to-understand parts through tokenization, which is a precursor to better processing. These steps in combination boost the dataset's structural soundness. When the processing phase is done, the data is then prepared for the following steps of chunking and embedding in the proposed method.

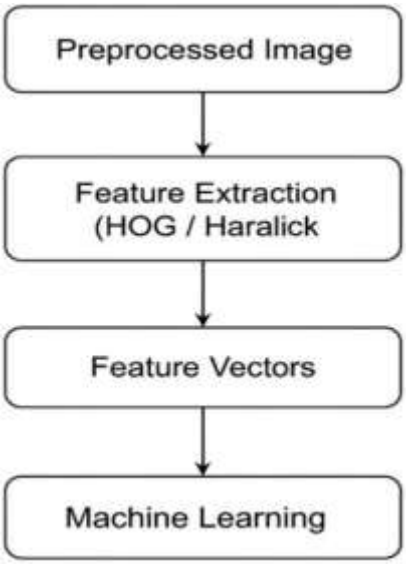

Figure 2: Feature Extraction Workflow

The core Retrieval-Augmented Generation (RAG) pipeline applied in the Generative AI is also illustrated in Fig. 2. In order to enhance the speed of processing and the retrievals' efficacy, the filtered dataset is segmented into shorter text fragments once it becomes available. In order to vectorize the segments, an embedding model such as OpenAI, Sentence-BERT, or other models maps them into the high-dimensional numerical embeddings. The output embeddings are then retained in vector database with fast similarity query support such as Pinecone, FAISS, or ChromaDB. Upon the user query reception, the system returns the highly related segments in the vector database in the semantic similarity. A Large Language Model (LLM) employs the returned segments in the form of contextual knowledge in order to create accurate and contextually related responses or summaries.

It makes the system more intelligent, reliable, and effective by ensuring the output is supported by the documents stored. The implementation of correct and contextually appropriate information minimizes the chances of generating erroneous or irrelevant information. Further, by providing correct, reliable, and superior output, it helps in making the overall user experience significantly better.

V. Algorithm Description

The architected Generative AI system integrates web crawling, cleaning of data, semantic chunking, generation of embeddings, and Retrieval-Augmented Generation (RAG) as a single pipeline. Raw web data are crawled by automated crawlers and normalized as well as de-noised. Text is segmented into semantically consistent pieces, converted into vector embeddings through Sentence-BERT or OpenAI embeddings, and saved within a vector database (FAISS, Pinecone, or ChromaDB) for efficient retrieval. On receiving a user input, the RAG framework retrieves the most highly related chunks and a Large Language Model generates correct, context-sensitive responses. The entire architecture is designed for efficient training and scalable deployment across domains and enterprise

use cases.The proposed models architecture is shown in the Figure 3.

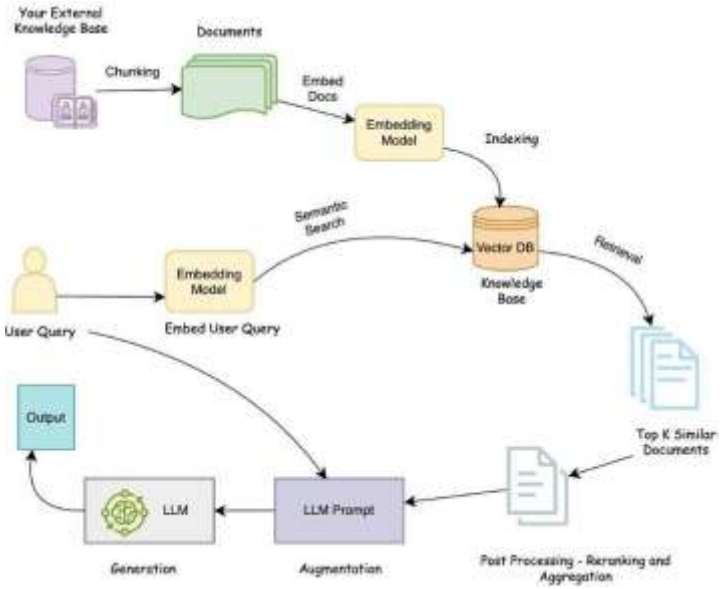

Figure 3: Proposed model architecture

VI. Result

The accuracy and performance of the proposed Generative AI Crawling and Chunking model in working with huge datasets are discussed in this section. The performance of the model is gauged with key performance indicators like accuracy, inference time, throughput, and cost-efficiency. The proposed method is contrasted with current conventional techniques of retrieval in an effort to portray strengths in system performance.

Table 1: The avg GPU memory ,Inference and Throughput

Model	Hardware	Batch	Avg. GPU Mem (GB)	Inference (ms/q)	Throughput (q/s)	Estimated $/1000 q
Baseline	CPU only	1	6	120	8.3	0.10
Model A	V100	8	10	160	6.3	0.30
Model B	A100	16	22	200	5.0	0.45

Above is the comparison of the different models in terms of hardware specifications, batch size, memory usage, inference time, throughput, and cost for 1000 queries. The data also shows the tradeoff between performance and resource usage.

The accuracy diagram shows the improvement in model performance with epochs such that the accuracy improves gradually with the progress of training and substantiates the reliability and efficacy of

the system in producing context-aware responses.

VII. Conclusion

It is an intelligent RAG-augmented pipeline with web crawling, preprocessing, semantic chunking, vector embeddings, and generative AI. It provides precise, contextually aware, and latest responses with minimized hallucinations. Highly reliable and effective, it is amenable to applications in knowledge discovery, summarization, and real-time information seeking with potential for progressive refinement.

VIII. References

[1] In their 2020 work, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, "P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Khashabi, et al.

[2] In Proc. EMNLP, 2020, V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. -t. Yih introduced their submission, "Dense Passage Retrieval for for Open-Domain Question Answering."

[3] "Leveraging Passage Retrieval with Generative Models for Open-Domain QA, "by G. Izacard and E. Grave, in EACL, 2021.

[4] H. Yu and colleagues, “A Survey on the Assessment of Retrieval Augmented Generation, “arXiv preprint arXiv:2405.07437, 2024.

[5] “Retrieval-Augmented Generation for AI-Generated Content: A Survey, “arXiv preprint arXiv;2402.19473, 2024, Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangchenng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui.

[6] “Dense Passage Retrieval: Is it Retrieving?” by B.Reichman and L. Heck In 2024, arXiv preprint arXiv:2402.11035.

[7] arXiv preprint arXiv:2405.06211, 2024; W. Fan et al., “A Survey on Retrieval-Augmented Large Language Models (RA-LLMs); Architectures, Training, and Applications.”

[8]”Graph Retrieval-Augmented Generation: A Survey, “ by B. Peng et al.2024, arXiv preprint arXiv:2408.08921.

[9] A Survey of retrieval-augmented text generation for large language models, by Y. Huang and J. Huang, arXiv preprint arXiv:2402.10981, 2024.

[10] A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape, and Future Directions, by S. Gupta, R. Ranjan, and S. N. Singh, arXiv preprint arXiv:2410.12837, 2024.