Retrieval Augmented Generation Internals

Compiled by D. Gueorguiev, 9/20/2025

# Introductory Notes

A typical RAG application has two main components:

*Indexing pipeline*: a pipeline for ingesting data from a source and indexing it. The indexing pipeline works *offline*.

*Retrieval and generation*: the actual RAG chain takes user query at run time and retrieves the relevant data from the index, then passes that to the model.

# The Semantic Search Engine

We will discuss how to build a semantic search engine using LangChain document loader, embedding model, and vector store.

# References

[1] [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Patrick Lewis et al, 2021](https://github.com/dimitarpg13/rag_architectures_and_concepts/blob/main/articles/Retrieval-Augmented_Generation_for_Knowledge-Intensive_NLP_Tasks_Lewis_2021.pdf)

[2] [Foundations of Vector Retrieval, S. Bruch, 2024](https://github.com/dimitarpg13/vector_db_intro/blob/main/articles/Foundations_of_Vector_Retrieval_Bruch_2024.pdf)

[3] [Retrieval-Augmented Generation for Large Language Models: A Survey, Y. Gao et al, 2024](https://github.com/dimitarpg13/rag_architectures_and_concepts/blob/main/articles/Retrieval-Augmented_Generation_for_Large_Language_Models-A_Survey_Gao_2024.pdf)

[4] [Build a Retrieval Augmented Generation (RAG) App: Part 1, LangChain](https://python.langchain.com/docs/tutorials/rag/),

<https://github.com/langchain-ai/langchain/blob/master/docs/docs/tutorials/rag.ipynb>

[5] Tutorial: Build a Semantic Search Engine with LangChain,

<https://python.langchain.com/docs/tutorials/retrievers/>

[6] Document loaders in LangChain: <https://python.langchain.com/docs/concepts/document_loaders/>

[7] Embedding models in LangChain: <https://python.langchain.com/docs/concepts/embedding_models/>

[8] Vector stores in LangChain: <https://python.langchain.com/docs/concepts/vectorstores/>

[9] [Build a Retrieval Augmented Generation (RAG) App: Part 2, LangChain](https://python.langchain.com/docs/tutorials/qa_chat_history/),

<https://github.com/langchain-ai/langchain/blob/master/docs/docs/tutorials/qa_chat_history.ipynb>

[] [Retrieval-Augmented Generation (RAG) from basics to advanced, Tejpal Kumawat, Medium, 2024](https://medium.com/@tejpal.abhyuday/retrieval-augmented-generation-rag-from-basics-to-advanced-a2b068fd576c)

[] [Retrieval-Augmented Generation (RAG) using LangChain, LlamaIndex, and OpenAI, Medium, Prasad Mahamulkar, 2024](https://pub.towardsai.net/introduction-to-retrieval-augmented-generation-rag-using-langchain-and-lamaindex-bd0047628e2a)

# Appendix

## Document loaders in LangChain

Interface: [BaseLoader](https://python.langchain.com/api_reference/core/document_loaders/langchain_core.document_loaders.base.BaseLoader.html)

class langchain\_core.document\_loaders.base.BaseLoader[source]

Interface for Document Loader.

Implementations should implement the lazy-loading method using generators to avoid loading all Documents into memory at once.

load is provided just for user convenience and should not be overridden.

Methods

async alazy\_load() -> AsyncIterator[Document] # A lazy loader for Document

aload(): Load data into Document objects.

lazy\_load(): A lazy loader for Documents.

load()

Load data into Document objects.

load\_and\_split([text\_splitter])

Load Documents and split into chunks.