Build RAG applications: Get Started

Cheatsheet: Build RAG Apps with LlamaIndex

LlamaIndex, like LangChain, is a framework for LLM-powered context augmentation. Typical use cases include chatbots, Retrieval-Augmented Generation (RAG) applications, and various types of AI assistants.

- LlamaIndex's Document class wraps or stores entire documents from the document store. In addition to storing the document's text, the Document class can store embeddings, metadata about the document such as when the document was created or the directory it came from, and relationships to other documents.
- A wide range of documents can be loaded using the `SimpleDocumentReader` loader, which can load individual files or entire directories. Moreover, additional loaders and connectors are available at LlamaHub.ai.
- Documents are further chunked into `Nodes` using document splitters. A LlamaIndex `Node` are similar in structure to a LlamaIndex `Document` and also stores metadata, relationships, and embeddings.
- LlamaIndex provides a wide variety of text splitters such as the `SentenceSplitter` which recursively splits documents on characters from a list while ensuring every chunk is no greater than a maximum token length. Moreover, in addition to providing native text splitters, LlamaIndex also provides a `LangChainNodeParser` class which is capable of wrapping and using any LangChain text splitter.
- In LlamaIndex, embeddings are generated and stored in one step. This is typically done using LlamaIndex's `VectorStoreIndex` class. The `VectorStoreIndex` class, by default, stores nodes in-memory, but can also be used to wrap external vector databases such as Chroma DB, FAISS, or Milvus.
- Embedding a user's prompt and retrieving relevant chunks typically occurs in one step in LlamaIndex. This is done using a retriever that is created from the `VectorStoreIndex` instance by calling the `as_retriever()` method. Using a retriever generated from the vector store index ensures that the same embedding model was used to embed the nodes is also used to embed the user's prompt.
- Additional advanced retrievers are offered by LlamaIndex.
- Prompt augmentation as well as querying the LLM to obtain a response occur in the same step in LlamaIndex.
    - If relevant nodes are already retrieved, this can be done by using a "response synthesizer", which takes the user's original prompts and retrieved nodes as inputs, and produces the LLM's response as output.
    - Alternatively, the one does not have to store the retrieved nodes as an intermediary step. In that case, a "query engine," created from a vector store index instance using the `as_query_engine()` method, takes the user's original prompt as input and outputs the LLM's response.
    - LlamaIndex's prompt augmentation is controlled by customizable prompt templates. Prompt templates are predefined texts that have placeholders for the user's original prompt and the retrieved text chunks.
    - LlamaIndex excels due to its simplicity, ease of development, and the presence of powerful native tools. On the other hand, LangChain, to which LlamaIndex is frequently compared, has more external integration, a more modular design, and easier customization of its various components.

Authors

Wojciech "Victor" Fulmyk