LlamaIndex Documentation

LlamaIndex (formerly known as GPT Index) is a powerful Python library that lets you connect your **custom data** (like PDF files, documents, Notion, databases) to **Large Language Models (LLMs)** such as GPT-4. It's ideal for building **question-answering systems** and **retrieval-augmented generation (RAG)** pipelines.

Installation

You can install LlamaIndex using pip:

```
pip install llama-index
```

If you also want to work with PDFs, you'll need:

```
pip install llama-index[all]
```

Step 1: Load Your Data

LlamaIndex provides various loaders for different data sources:

Load from Text Files (Local Folder)

```
from llama_index import SimpleDirectoryReader

documents = SimpleDirectoryReader('data').load_data()
```

Ensure your `data/` folder contains `.txt` or `.md` files.

Load from PDFs

```
from llama_index.readers.file import PDFReader
```

```
reader = PDFReader()
documents = reader.load_data(file_path='sample.pdf')
```

Load from Web Pages

```
from llama_index.readers.web import SimpleWebPageReader

urls = ["https://example.com"]
documents = SimpleWebPageReader().load_data(urls)
```

Step 2: Index the Data

You can create a vector index from your loaded documents:

```
from llama_index import VectorStoreIndex

index = VectorStoreIndex.from_documents(documents)
```

This will:

- Split large text into chunks
- Embed them using OpenAI Embeddings (by default)
- Store in memory or a persistent vector store

Step 3: Ask Questions

Turn the index into a query engine and start chatting:

```
query_engine = index.as_query_engine()
response = query_engine.query("What is this document about?")
print(response)
```

Configuration

Set Your API Key

LlamaIndex uses OpenAI by default. You must set the API key:

```
export OPENAI_API_KEY=your_key_here
```

Or in Python:

```
import os
os.environ["OPENAI_API_KEY"] = "your_key_here"
```

Customize Chunking and Embeddings

```
from llama_index import ServiceContext
from llama_index.text_splitter import SentenceSplitter
from llama_index.embeddings import OpenAIEmbedding

splitter = SentenceSplitter(chunk_size=512)
service_context = ServiceContext.from_defaults(embed_model=OpenAIEmbedding(), text_splitter=splitter)

index = VectorStoreIndex.from_documents(documents, service_context=service_context)
```

Persist and Reload Index

```
# Save to disk
index.storage_context.persist(persist_dir="storage")

# Load back later
from llama_index import StorageContext, load_index_from_storage

storage_context = StorageContext.from_defaults(persist_dir="storage")
index = load_index_from_storage(storage_context)
```

Advanced Features

RAG with LangChain

You can plug LlamaIndex into LangChain as a retriever:

```
from langchain.chains import RetrievalQA
from langchain.llms import OpenAI

retriever = index.as_retriever()
qa_chain = RetrievalQA.from_chain_type(llm=OpenAI(), retriever=retriever)
qa_chain.run("Summarize this file")
```

Other Input Types

- Notion: `NotionPageReader`
- Google Docs: `GmailReader`
- CSVs: `PandasCSVReader`
- YouTube: `YouTubeTranscriptReader`