DrDroid

LlamaIndex Cheatsheet

Getting Started

pip install llama-index
Install LlamaIndex

pip install llama-index-llms-openai
Install OpenAI LLM integration

from llama_index import SimpleDirectoryReader, GPTVectorStoreIndex, LLMPredictor, ServiceContext
Import common modules

from llama_index.llms import OpenAI
Import OpenAI LLM

Document Loading

documents = SimpleDirectoryReader('data_directory/').load_data()
Load documents from a directory

documents = SimpleDirectoryReader().load_data(file_paths=['file1.pdf', 'file2.txt'])
Load specific files

from llama_index.readers.web import SimpleWebPageReader
Import web reader

documents = SimpleWebPageReader().load_data(urls=['https://example.com'])
Load data from websites

Index Creation

index = GPTVectorStoreIndex.from_documents(documents)
Create vector store index from documents

index.save_to_disk('index.json')
Save index to disk

index = GPTVectorStoreIndex.load_from_disk('index.json')
Load index from disk

from llama_index import StorageContext, load_index_from_storage
Import storage context

storage_context = StorageContext.from_defaults(persist_dir='./storage')
Create storage context

index = load_index_from_storage(storage_context)
Load index from storage

Querying

query_engine = index.as_query_engine()
Create a query engine

response = query_engine.query('your question here')
Execute the query

print(response)
Print the response

query_engine = index.as_query_engine(similarity_top_k=5)
Create query engine with top k results

Advanced Retrieval

from llama_index.retrievers import VectorIndexRetriever
Import vector retriever

retriever = VectorIndexRetriever(index=index, similarity_top_k=5)
Create a retriever

nodes = retriever.retrieve('query text')
Retrieve nodes from index

from llama_index.query_engine import RetrieverQueryEngine
Import retriever query engine

query_engine = RetrieverQueryEngine(retriever=retriever)
Create query engine from retriever

Custom LLM Configuration

llm = OpenAI(model='gpt-4', temperature=0.2)
Configure OpenAI LLM

service_context = ServiceContext.from_defaults(llm=llm)
Create service context with custom LLM

index = GPTVectorStoreIndex.from_documents(documents, service_context=service_context)
Create index with custom service context

Embedding Models

from llama_index.embeddings import OpenAIEmbedding
Import OpenAI embedding model

embed_model = OpenAIEmbedding(model='text-embedding-ada-002')
Configure embedding model

service_context = ServiceContext.from_defaults(embed_model=embed_model)
Create service context with custom embedding model

Chat Engines

chat_engine = index.as_chat_engine()
Create a chat engine

response = chat_engine.chat('your message')
Chat with the engine

response = chat_engine.chat('follow-up question')
Ask follow-up questions maintaining context