Customise your RAG application! 🦙

How do I:

- parse my documents into smaller chunks!
- use a different vector store!
- retrieve more context when I query!
- use a different LLM!
- use a different response mode!
- stream the response back!
- create a chatbot instead of Q&A

@akshay_pachaar

Here's a minimal implementation of a Document Chat RAG application, all in just 6 lines of code!

Now let's customize it step by step ...

Swipe 👉

@akshay_pachaar

```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine()
response = query_engine.query("What did the author do growing up?")
print(response)
```

1️⃣ I want to parse my documents into smaller chunks:

Swipe 👉

@akshay_pachaar

```python
# Global settings
from llama_index.core import Settings

Settings.chunk_size = 512

# Local settings
from llama_index.core.node_parser import SentenceSplitter

index = VectorStoreIndex.from_documents(
    documents, transformations=[SentenceSplitter(chunk_size=512)]
)
```

2️⃣ I want to use a different vector store:

First, you can install the vector store you want to use. For example, to use Chroma as the vector store, you can install it using pip.

`!pip install llama-index-vector-stores-chroma`

Then, you can use it in your code:

Swipe 👉

@akshay_pachaar

```python
import chromadb
from llama_index.vector_stores.chroma import ChromaVectorStore
from llama_index.core import StorageContext

chroma_client = chromadb.PersistentClient()
chroma_collection = chroma_client.create_collection("quickstart")
vector_store = ChromaVectorStore(chroma_collection=chroma_collection)
storage_context = StorageContext.from_defaults(vector_store=vector_store)
```

StorageContext defines the storage backend for where the documents, embeddings, and indexes are stored.

```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(
    documents, storage_context=storage_context
)
query_engine = index.as_query_engine()
response = query_engine.query("What did the author do growing up?")
print(response)
```

3️⃣ I want to retrieve more context when I query:

`as_query_engine()` builds a default `retriever` and `query engine` on top of the index.

You can configure them with keyword arguments to return top 5 similar documents instead of the default 2.

Swipe 👉

@akshay_pachaar

```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine(similarity_top_k=5)
response = query_engine.query("What did the author do growing up?")
print(response)
```

4️⃣ I want to use a different LLM:

Thanks to Ollama & it's integration with LlamaIndex, using custom local LLMs is a cake walk!

Swipe 👉

@akshay_pachaar

```python
# Global settings
from llama_index.core import Settings
from llama_index.llms.ollama import Ollama

Settings.llm = Ollama(model="mistral", request_timeout=60.0)

# Local settings
index.as_query_engine(llm=Ollama(model="mistral", request_timeout=60.0))
```

5️⃣ I want to use a different response mode:

There are multiple response modes available:

- refine
- compact
- tree_summarize
- accumulate
- and you can find more info in the LlamaIndex docs

It's really simple to use it:

Swipe 👉

@akshay_pachaar

```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine(response_mode="tree_summarize")
response = query_engine.query("What did the author do growing up?")
print(response)
```

6️⃣ I want to stream the response back:

When you do this, you don't have to wait for the entire response to be generated by the LLM. Just stream the tokens as outputted by the LLM!

This makes Chatbot applications really cool!

Swipe 👉

@akshay_pachaar

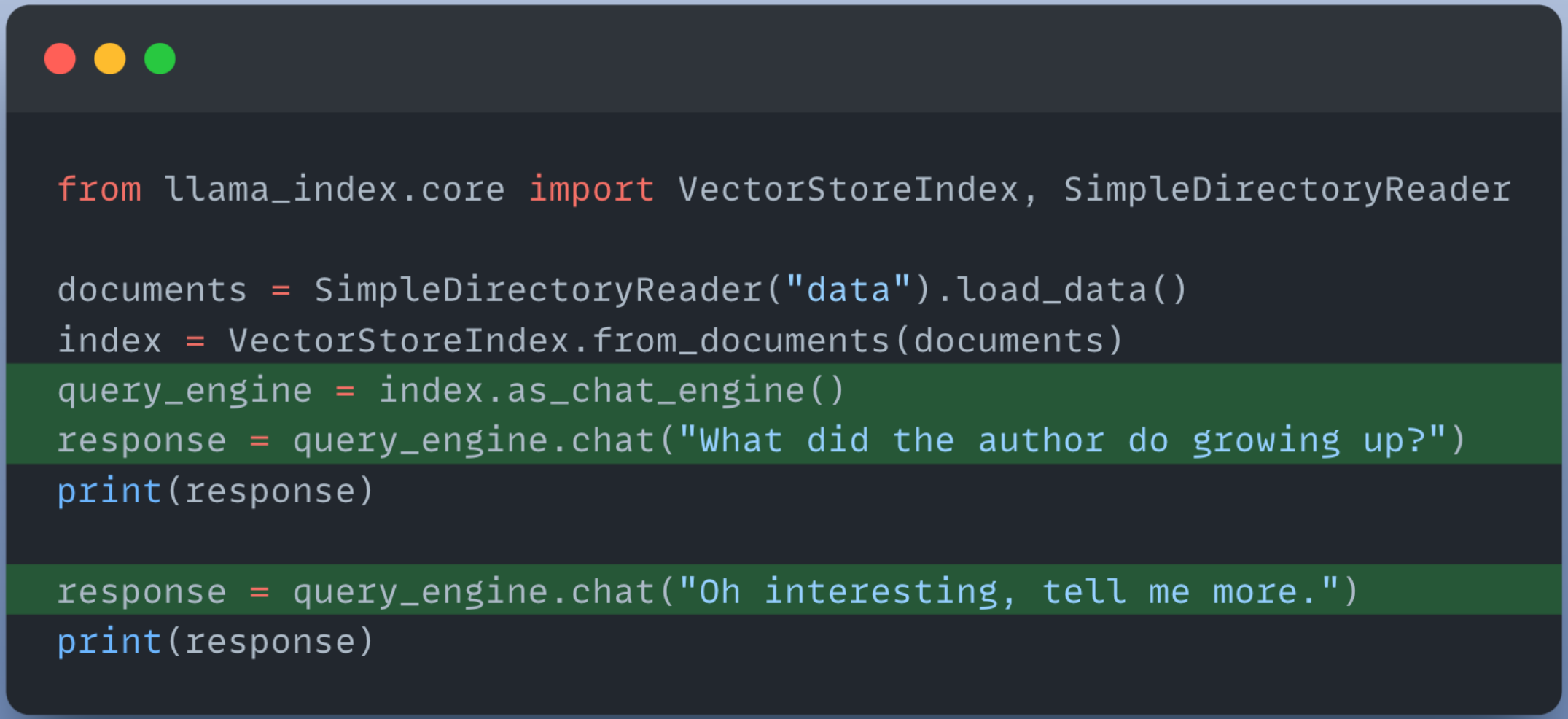
```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine(streaming=True)
response = query_engine.query("What did the author do growing up?")
response.print_response_stream()
```

7️⃣ I want a chatbot instead of Q&A

Swipe 👉

@akshay_pachaar



```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_chat_engine()
response = query_engine.chat("What did the author do growing up?")
print(response)

response = query_engine.chat("Oh interesting, tell me more.")
print(response)
```

That's a wrap!

If you interested in:

- Python 🐍
- Data Science 📈
- Machine Learning 🤖
- MLOps 🛠️
- NLP 🗣️
- Computer Vision 📽️
- LLMs 🧠

Follow me on LinkedIn ✔️
Everyday, I share tutorials on above topics!

Cheers!! 🙂

@akshay_pachaar