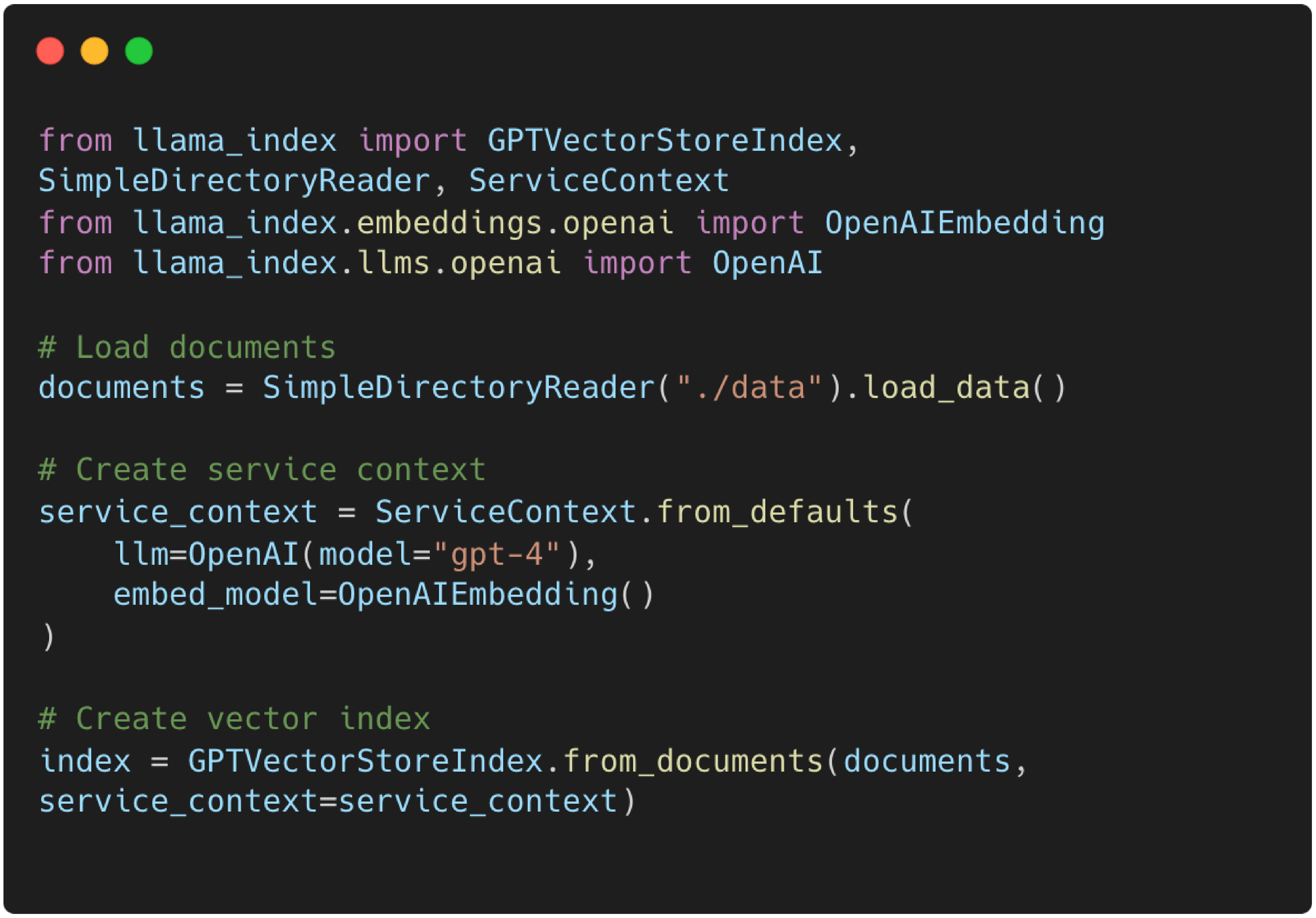
Analytics Vidhya

Mastering RAG

RAG with LlamaIndex

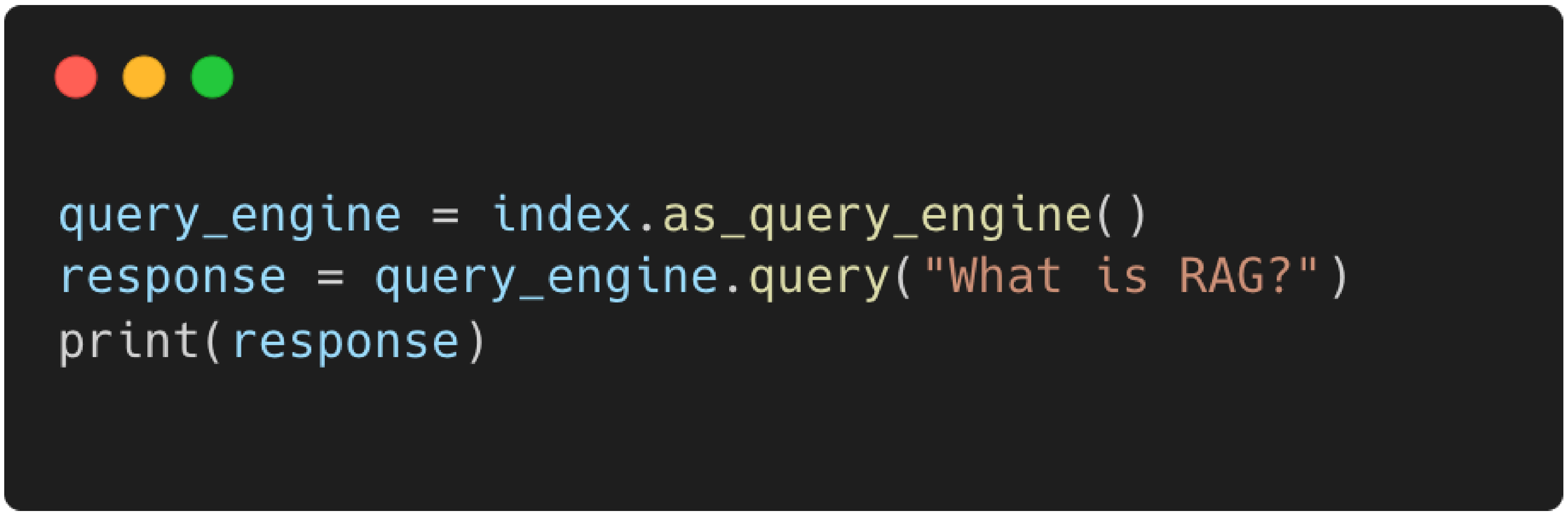


```python
from llama_index import GPTVectorStoreIndex,
SimpleDirectoryReader, ServiceContext
from llama_index.embeddings.openai import OpenAIEmbedding
from llama_index.llms.openai import OpenAI

# Load documents
documents = SimpleDirectoryReader("./data").load_data()

# Create service context
service_context = ServiceContext.from_defaults(
    llm=OpenAI(model="gpt-4"),
    embed_model=OpenAIEmbedding()
)

# Create vector index
index = GPTVectorStoreIndex.from_documents(documents,
service_context=service_context)
```



```python
query_engine = index.as_query_engine()
response = query_engine.query("What is RAG?")
print(response)
```

Why Use LlamaIndex for RAG?

LlamaIndex (formerly GPT Index) provides a modular and efficient framework for implementing RAG. It simplifies data ingestion, indexing, and querying by offering various tools for data connection and retrieval optimization.

Advantages of LlamaIndex

- Flexible Data Integration: Supports diverse data sources such as PDFs, databases, APIs, and JSON files.
- Efficient Indexing: Uses embedding-based retrieval, hierarchical indices, and metadata filtering for optimized search.
- Scalability: Enables distributed and hybrid search strategies.
- Seamless LLM Compatibility: Works with OpenAI, Hugging Face models, and custom LLM deployments.

Implementing RAG with LlamaIndex

Step 1: Install Dependencies

Ensure you have the required libraries installed:

```
pip install llama-index openai faiss-cpu
```

Step 2: Initialize the LlamaIndex Components

```
from llama_index import GPTVectorStoreIndex,
SimpleDirectoryReader, ServiceContext
from llama_index.embeddings.openai import OpenAIEmbedding
from llama_index.llms.openai import OpenAI

# Load documents
documents = SimpleDirectoryReader("./data").load_data()

# Create service context
service_context = ServiceContext.from_defaults(
    llm=OpenAI(model="gpt-4"),
    embed_model=OpenAIEmbedding()
)

# Create vector index
index = GPTVectorStoreIndex.from_documents(documents,
service_context=service_context)
```

Step 3: Querying with RAG

```python
query_engine = index.as_query_engine()
response = query_engine.query("What is RAG?")
print(response)
```

Step 4: Enhancing Retrieval with Faiss (Optional)

For large-scale indexing, we can integrate Faiss for efficient similarity search:

```python
from llama_index.vector_stores.faiss import FaissVectorStore
import faiss

# Initialize Faiss index
faiss_index = faiss.IndexFlatL2(1536)
vector_store = FaissVectorStore(faiss_index)

# Recreate the index with Faiss
index = GPTVectorStoreIndex.from_documents(
    documents, service_context=service_context, vector_store=vector_store)
```

Analytics Vidhya

Optimizing RAG with LlamaIndex

1. Hybrid Search Strategies

Combining keyword-based search (BM25) with semantic embeddings improves retrieval precision:

```
from llama_index.query_engine import RetrieverQueryEngine
from llama_index.retrievers import BM25Retriever, VectorIndexRetriever

bm25_retriever = BM25Retriever.from_defaults(documents)
vector_retriever = index.as_retriever()

hybrid_retriever = RetrieverQueryEngine(
    retrievers=[bm25_retriever, vector_retriever],
    retriever_mode="hybrid"
)
```

2. Metadata Filtering

Applying filters ensures more relevant results:

```
response = query_engine.query(
    "What is RAG?",
    filters={"category": "AI", "date": "2024"}
)
```

3. Chunking Strategies

Using sliding window techniques or hierarchical indices helps manage long documents effectively.

```
from llama_index.node_parser import SimpleNodeParser

parser = SimpleNodeParser(chunk_size=512, chunk_overlap=128)
nodes = parser.get_nodes_from_documents(documents)
```

Use Cases of RAG with LlamaIndex

- Enterprise Search: Enhancing internal knowledge bases for improved employee productivity.
- Chatbots and Virtual Assistants: Powering AI-driven customer support with real-time knowledge updates.
- Legal and Compliance Research: Querying extensive regulatory documents for fast and accurate retrieval.