FAISS

Created	@February 14, 2025 12:51 PM
Tags	Personal

PDF Text Processing and Embedding Storage with FAISS

This code demonstrates how to extract text from a PDF, split it into chunks, generate embeddings for each chunk using Sentence Transformers, and then store these embeddings in a FAISS index for efficient similarity search.

1. PDF Text Extraction

~~~
```python
import fitz  # PyMuPDF

def extract_text_from_pdf(pdf_path):
    doc = fitz.open(pdf_path)
    text = "\n".join([page.get_text() for page in doc])
    return text

pdf_text = extract_text_from_pdf("/Users/vinod/Desktop/mike/sample.pdf")
~~~

2. Text Chunking

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

def split_text(text, chunk_size=500, chunk_overlap=50):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size, chunk_overlap=chunk_overlap
    )
    return text_splitter.split_text(text)
```

```
chunks = split_text(pdf_text)
print(f"Total chunks: {len(chunks)}")
```

3. Embedding Generation

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2")
chunk_embeddings = model.encode(chunks)

# Print first 5 embeddings (optional)
for i, embedding in enumerate(chunk_embeddings[:5]):
    print(f"Embedding {i+1}: {embedding}\n")

print(f"Generated {len(chunk_embeddings)} embeddings!")
```

- `from sentence_transformers import SentenceTransformer`: Imports the Sentence Transformer library. Install it: `pip install sentence-transformers`.
- `model = ...`: Loads a pre-trained Sentence Transformer model. `all-MiniLM-L6-v2` is a good balance of speed and performance.
- `chunk_embeddings = model.encode(chunks)`: Generates embeddings for each text chunk.
- `for i, embedding in enumerate(chunk_embeddings[:5]):`: (Optional) Prints the first 5 embeddings for inspection. Remove this if you don't need to see the embeddings.
- `print(...)`: Prints the number of embeddings generated.

4. FAISS Indexing

```
import faiss
import numpy as np
```

```python
dimension = chunk_embeddings.shape[1]  # 384 for MiniLM
index = faiss.IndexFlatL2(dimension)  # L2 = Euclidean distance

# Convert embeddings to FAISS format
faiss_data = np.array(chunk_embeddings, dtype=np.float32)
index.add(faiss_data)

print("Embeddings stored in FAISS index!")
```

- **`import faiss`**: Imports the FAISS library. Install it: `pip install faiss-cpu` (for CPU) or `pip install faiss-gpu` (if you have a compatible GPU).
- **`import numpy as np`**: Imports NumPy for numerical operations.
- **`dimension = chunk_embeddings.shape[1]`**: Gets the dimensionality of the embeddings (384 for `all-MiniLM-L6-v2`).
- **`index = faiss.IndexFlatL2(dimension)`**: Creates a FAISS index. `IndexFlatL2` is a simple and efficient index for storing and searching L2-normalized vectors (embeddings).
- **`faiss_data = np.array(chunk_embeddings, dtype=np.float32)`**: Converts the embeddings to a NumPy array of `float32` data type, which is required by FAISS.
- **`index.add(faiss_data)`**: Adds the embeddings to the FAISS index.
- **`print(...)`**: Confirms that the embeddings have been added to the index.