TextChunks

1. Extracting text from pdf

~~~
```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
~~~

- `import fitz`: Imports the PyMuPDF library (often imported as `fitz`). Install it if you haven't already: `pip install pymupdf`. This library is used for working with PDF files.
- `def extract_text_from_pdf(pdf_path):`: Defines a function that takes the PDF file path as input.
- `doc = fitz.open(pdf_path)`: Opens the specified PDF file.
- `text = "\n".join([page.get_text() for page in doc])`: Extracts the text content from each page of the PDF and joins it into a single string. The `\n` ensures that page breaks are preserved as newline characters.
- `return text`: Returns the complete extracted text.

2. Text Chunking

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

def split_text(text, chunk_size=100, chunk_overlap=0):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size, chunk_overlap=chunk_overlap
    )
    return text_splitter.split_text(text)
```

- **`from langchain.text_splitter import RecursiveCharacterTextSplitter`**: Imports the `RecursiveCharacterTextSplitter` from the `langchain` library. Install it: `pip install langchain`. This is a powerful tool for splitting text into chunks, and it attempts to respect sentence and paragraph boundaries.
- **`def split_text(text, chunk_size=100, chunk_overlap=0):`**: Defines a function to split the input text into chunks.
    - `chunk_size`: The desired number of characters in each chunk.
    - `chunk_overlap`: The number of overlapping characters between consecutive chunks. This helps maintain context across chunks.
- **`text_splitter = RecursiveCharacterTextSplitter(...)`**: Creates an instance of the `RecursiveCharacterTextSplitter` with the specified `chunk_size` and `chunk_overlap`.
- **`return text_splitter.split_text(text)`**: Splits the input `text` into chunks and returns a list of strings, where each string is a chunk.

3. Example Usage and Output

```
pdf_text = extract_text_from_pdf("/Users/vinod/Desktop/mike/sample.pdf")
chunks = split_text(pdf_text)

print(f"Total chunks: {len(chunks)}")
print(chunks[:2])  # Print first two chunks
```

- **`pdf_text = extract_text_from_pdf(...)`**: Calls the `extract_text_from_pdf` function to extract the text from your PDF. **Remember to replace `"/Users/vinod/Desktop/mike/sample.pdf"` with the actual path to your PDF file.**
- **`chunks = split_text(pdf_text)`**: Calls the `split_text` function to split the extracted text into chunks.
- **`print(f"Total chunks: {len(chunks)}")`**: Prints the total number of chunks created.
- **`print(chunks[:2])`**: Prints the first two chunks. This is a good way to inspect the output and make sure the chunking is working as expected.

This code provides a robust way to extract and chunk PDF text. You can adjust the `chunk_size` and `chunk_overlap` parameters to fine-tune the chunking process for your specific needs. The `RecursiveCharacterTextSplitter` is generally a better choice than simply splitting by fixed lengths, as it tries to keep sentences and paragraphs together within chunks.