Xử lý File Excel Không Cấu Trúc với LangChain

Tình huống thực tế

Hãy tưởng tượng bạn được giao nhiệm vụ phân tích một **file Excel khổng lồ** chứa đầy các **phản hồi của khách hàng**.

File này có:

- Nhiều sheet,
- Nhiều dòng hữu ích,
- Nhưng cũng rất nhiều dòng rác hoặc trống.

Làm sao bạn **lọc ra thông tin quan trọng** từ một "rừng dữ liệu" như vậy?

Đây chính là lúc bạn cần đến **LangChain** – một thư viện mạnh mẽ giúp xử lý dữ liệu không có cấu trúc.

Mục tiêu của bài học:

Sau bài này, bạn sẽ biết cách:

1. Tải file Excel vào LangChain
2. Chia nhỏ dữ liệu để xử lý tốt hơn
3. Tạo **embeddings** từ dữ liệu để phục vụ cho phân tích, tìm kiếm hoặc sinh nội dung

Bước 1: Tải dữ liệu Excel

Chúng ta dùng **UnstructuredExcelLoader** trong module `langchain_community`.

Cú pháp cơ bản:

```
from langchain_community.document_loaders import UnstructuredExcelLoader


loader = UnstructuredExcelLoader("reviews.xlsx", mode="elements")
docs = loader.load()
```

`mode="elements"` giúp chia nhỏ từng phần trong file Excel.

- Nếu file của bạn cực kỳ có cấu trúc (ví dụ: bảng biểu đều đặn), bạn có thể thử `mode="table"`.
- Dữ liệu giống `pandas.read_csv()` – chỉ là cách tiếp cận "LangChain-style".

Bước 2: Chia nhỏ nội dung (Chunking)

Lý do: Các mô hình ngôn ngữ có giới hạn **số lượng token**. Nếu bạn đưa vào quá nhiều chữ, nó sẽ bị cắt mất nội dung.

Giải pháp: Dùng **RecursiveCharacterTextSplitter**

```
from langchain.text_splitter import RecursiveCharacterTextSplitter


splitter = RecursiveCharacterTextSplitter(chunk_size=2000, chunk_overlap=200)
chunks = splitter.split_documents(docs)
```

- **chunk_size**: độ dài mỗi đoạn văn bản.
- **chunk_overlap**: phần nội dung lặp lại giữa các đoạn (để giữ ngữ cảnh liền mạch).

Ví dụ:

- Nếu bạn đặt `chunk_size=2000` và `chunk_overlap=200`, mỗi đoạn mới sẽ giữ lại 200 ký tự cuối của đoạn trước.

Gợi ý:

- Nếu dữ liệu của bạn là đánh giá ngắn (reviews), bạn có thể giảm xuống `chunk_size=300`, `chunk_overlap=50` cho tiết kiệm tài nguyên.

Bước 3: Tạo Embeddings (biểu diễn ngữ nghĩa)

Dùng `OpenAIEmbeddings` từ LangChain để biến các đoạn văn bản thành vector ngữ nghĩa.

```
from langchain_openai import OpenAIEmbeddings


embeddings = OpenAIEmbeddings(model="text-embedding-3-large")
```

Mô hình `text-embedding-3-large` là phiên bản tốt nhất hiện tại của OpenAI.

Lưu ý:

- **API Key** cần được bảo mật bằng biến môi trường hoặc file `secrets`.
- Nếu bạn muốn tiết kiệm chi phí, có thể dùng model embedding nhẹ hơn hoặc tạo **embeddings tùy chỉnh** (sẽ học sau).

Tổng kết

Chúng ta đã học:

1. Cách nạp dữ liệu Excel vào LangChain
2. Cách chia nhỏ văn bản để xử lý hiệu quả
3. Cách tạo embeddings để chuẩn bị cho các tác vụ phân tích AI

Từ khóa quan trọng:

- `chunk_size` và `chunk_overlap` là 2 yếu tố then chốt ảnh hưởng đến hiệu quả và độ chính xác.

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #1
Được tạo 6 tháng 5 2025 10:09:53 bởi Đỗ Ngọc Tú
Được cập nhật 6 tháng 5 2025 10:15:35 bởi Đỗ Ngọc Tú