

# Bài Thực Hành LongRAG: Truy Vấn Thông Minh Trên Tài Liệu Dài

## Mục tiêu

- Tải và xử lý văn bản dài (PDF/text)
- Chia thành các đoạn dài (long chunks)
- Tạo vector embeddings và lưu vào FAISS
- Truy xuất các đoạn liên quan từ câu hỏi người dùng
- Gửi vào GPT-4 (hoặc tương đương) để sinh câu trả lời chính xác

## Công cụ cần cài đặt

```
pip install langchain openai faiss-cpu tiktoken pypdf
```

## 1. Chuẩn bị văn bản dài (ví dụ: tài liệu PDF)

Giả sử bạn có tệp `bao_cao_tai_chinh.pdf`

```
from langchain.document_loaders import PyPDFLoader

loader = PyPDFLoader("bao_cao_tai_chinh.pdf")
documents = loader.load()
print(f"Số trang tài liệu: {len(documents)}")
```

## 2. Chia nhỏ tài liệu thành đoạn DÀI (Long chunks)

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

splitter = RecursiveCharacterTextSplitter(
    chunk_size=2000, # chunk dài
    chunk_overlap=200, # cho ngữ cảnh tốt hơn
```

```
)  
chunks = splitter.split_documents(documents)  
print(f"Tổng số đoạn sau khi chia: {len(chunks)}")
```

### 3. Tạo Embedding và lưu trữ FAISS

```
from langchain.vectorstores import FAISS  
from langchain.embeddings import OpenAIEmbeddings  
  
embeddings = OpenAIEmbeddings() # Hoặc HuggingFaceEmbeddings nếu bạn muốn miễn phí  
db = FAISS.from_documents(chunks, embeddings)
```

Bạn có thể lưu lại vector store để dùng sau:

```
db.save_local("longrag_index")
```

### 4. Truy xuất thông tin từ câu hỏi

```
query = "Phân tích các rủi ro tài chính được đề cập trong phần 4 của báo cáo?"  
  
docs = db.similarity_search(query, k=5) # Lấy 5 đoạn dài nhất gần nhất  
for i, doc in enumerate(docs):  
    print(f"--- Đoạn {i+1} ---\n{doc.page_content[:500]}\n")
```

### 5. Gửi vào GPT để sinh câu trả lời

```
from langchain.chat_models import ChatOpenAI  
from langchain.chains import RetrievalQA  
  
llm = ChatOpenAI(model_name="gpt-4", temperature=0)  
  
qa = RetrievalQA.from_chain_type(  
    llm=llm,  
    retriever=db.as_retriever(search_kwargs={"k": 5}),  
    return_source_documents=True  
)  
  
result = qa({"query": query})  
print("\n Câu trả lời:\n", result["result"])
```

## 6. Giải thích hoạt động của LongRAG

Bước	Mô tả
1. Tải tài liệu dài	Đọc toàn bộ file PDF
2. Chia đoạn lớn	Cắt đoạn 2000 tokens để giữ được ngữ cảnh sâu
3. Gán vector	Mỗi đoạn → một vector
4. Tìm kiếm vector	Truy xuất đoạn dài nhất, liên quan nhất
5. Gửi cho GPT	Dựa vào các đoạn để trả lời chính xác

## Kết luận

- **LongRAG** là kỹ thuật hiệu quả khi bạn cần **xử lý tài liệu dài và phức tạp** như báo cáo, tài liệu y tế, nghiên cứu khoa học.
- Ưu điểm: giữ được ngữ cảnh dài, độ chính xác cao
- Bạn có thể kết hợp LongRAG với giao diện chatbot để triển khai vào doanh nghiệp

Tác giả: **Đỗ Ngọc Tú**  
Công Ty Phần Mềm **VHTSoft**