

Flan-T5 với RAG

Dưới đây là một **bài thực hành chi tiết** giúp bạn xây dựng hệ thống **RAG (Retrieval-Augmented Generation)** sử dụng mô hình **Flan-T5** để trả lời câu hỏi dựa trên tài liệu PDF. Hệ thống này sẽ:

- **Trích xuất nội dung** từ tệp PDF.
- **Chia nhỏ văn bản** thành các đoạn (chunk) và tạo **vector embedding** cho từng đoạn.
- **Lưu trữ các vector** trong cơ sở dữ liệu FAISS để truy vấn nhanh chóng.
- **Truy xuất các đoạn văn bản liên quan** đến câu hỏi người dùng.
- **Sử dụng Flan-T5** để sinh câu trả lời dựa trên ngữ cảnh truy xuất được.

Môi trường và thư viện cần thiết

Trước tiên, hãy cài đặt các thư viện cần thiết:

```
pip install transformers sentence-transformers faiss-cpu pypdf
```

1. Trích xuất văn bản từ PDF

```
from pypdf import PdfReader

def extract_text_from_pdf(pdf_path):
    reader = PdfReader(pdf_path)
    text = ""
    for page in reader.pages:
        text += page.extract_text()
    return text
```

2. Chia văn bản thành các đoạn nhỏ (chunk)

```
def split_text(text, chunk_size=500, overlap=50):
    chunks = []
    start = 0
    while start < len(text):
        end = start + chunk_size
        chunks.append(text[start:end])
        start = end - overlap
```

```
start += chunk_size - overlap
return chunks
```

3. Tạo vector embedding cho từng đoạn văn bản

```
from sentence_transformers import SentenceTransformer

def create_embeddings(chunks):
    model = SentenceTransformer('all-MiniLM-L6-v2')
    embeddings = model.encode(chunks)
    return embeddings, model
```

4. Lưu trữ embeddings trong FAISS5. Truy xuất các đoạn văn bản liên quan đến câu hỏi

```
def retrieve_relevant_chunks(question, model, index, chunks, top_k=3):
    question_embedding = model.encode([question])
    distances, indices = index.search(np.array(question_embedding), top_k)
    retrieved_chunks = [chunks[i] for i in indices[0]]
    return retrieved_chunks
```

6. Sinh câu trả lời bằng Flan-T5

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

def generate_answer(question, context, model_name="google/flan-t5-base"):
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

    prompt = f"Context: {context}\n\nQuestion: {question}\n\nAnswer:"
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True)
    outputs = model.generate(**inputs, max_new_tokens=100)
    answer = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return answer
```

7. Kết hợp tất cả thành một pipeline

```
def rag_pipeline(pdf_path, question):
    # Bước 1: Trích xuất và xử lý văn bản
    text = extract_text_from_pdf(pdf_path)
    chunks = split_text(text)
```

```
# Bước 2: Tạo embeddings và index
embeddings, embed_model = create_embeddings(chunks)
index = create_faiss_index(np.array(embeddings))

# Bước 3: Truy xuất các đoạn liên quan
relevant_chunks = retrieve_relevant_chunks(question, embed_model, index, chunks)
context = " ".join(relevant_chunks)

# Bước 4: Sinh câu trả lời
answer = generate_answer(question, context)

return answer
```

Ví dụ sử dụng

```
pdf_path = "duong-luoi-bo.pdf"
question = "Tranh chấp ở Biển Đông bắt đầu từ khi nào?"
answer = rag_pipeline(pdf_path, question)
print("Câu trả lời:", answer)
```

Gợi ý nâng cao

- **Tăng hiệu suất:** Sử dụng `faiss-gpu` nếu bạn có GPU.
- **Cải thiện chất lượng câu trả lời:** Fine-tune Flan-T5 trên tập dữ liệu của bạn.
- **Giao diện người dùng:** Tích hợp với Gradio hoặc Streamlit để tạo giao diện thân thiện.

Phiên bản #2

Được tạo 7 tháng 5 2025 10:00:24 bởi Đỗ Ngọc Tú

Được cập nhật 7 tháng 5 2025 10:06:33 bởi Đỗ Ngọc Tú