python #9

▼ 문제

- AI 기반 고객 리뷰 분석 시스템을 개발하는 팀의 일원
- 고객 리뷰 문장을 벡터로 임베딩하고, PostgreSQL의 pgvector 기능을 활용하여 비슷한 리뷰를 검색하는 기능을 구현

1단계
- SentenceTransformer로 리뷰 데이터 임베딩
- 아래 5개의 리뷰 데이터를 벡터화
  - sentence-transformers/paraphrase-MiniLM-L6-v2 모델 사용
  - reviews = [
   "배송이 빠르고 제품도 좋아요.",
   "품질이 기대 이상입니다!",
   "생각보다 배송이 오래 걸렸어요.",
   "배송은 느렸지만 포장은 안전했어요.",
   "아주 만족스러운 제품입니다."
]

2단계
- PostgreSQL 테이블 만들기
CREATE EXTENSION IF NOT EXISTS vector;
CREATE TABLE review_vectors (
  id SERIAL PRIMARY KEY,
  review TEXT,
  embedding VECTOR(384) -- 384차원 벡터 (MiniLM 모델 기준)
);

3단계
- 벡터 저장
  - Python 코드로 임베딩된 벡터를 review_vectors 테이블에 저장

4단계
- 유사도 검색 기능 만들기
  - "배송이 느렸어요" 라는 쿼리 문장에 대해 가장 유사한 리뷰 3개를 코사인 유사도로 검색하는 SQL 쿼리를 작성

쿼리와 결과 그리고 코드를 댓글로 달아주세요.

▼ 쿼리 & 코드

```
# version setting & 환경변수 설정
import torch
import transformers
import sentence_transformers
import sklearn
import numpy
import scipy

print(f"torch: {torch.__version__}")
print(f"transformers: {transformers.__version__}")
print(f"sentence-transformers: {sentence_transformers.__version__}")
print(f"scikit-learn: {sklearn.__version__}")
print(f"numpy: {numpy.__version__}")
```

```
print(f"scipy: {scipy.__version__}")

from sentence_transformers import SentenceTransformer
import numpy as np
import psycopg2

from dotenv import load_dotenv
import os

load_dotenv()  # .env 로드
```

```
torch: 2.2.2
transformers: 4.25.1
sentence-transformers: 2.2.2
scikit-learn: 1.3.2
numpy: 1.24.4
scipy: 1.10.1
```

```
# 1단계: 문장 임베딩
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
reviews = [
    "배송이 빠르고 제품도 좋아요.",
    "품질이 기대 이상입니다!",
    "생각보다 배송이 오래 걸렸어요.",
    "배송은 느렸지만 포장은 안전했어요.",
    "아주 만족스러운 제품입니다."
]
embeddings = model.encode(reviews)

# 2단계: PostgreSQL 테이블 생성
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    database="postgres",
    user="postgres",
    password=os.getenv("PG_PASSWORD"),  # 환경변수에서 불러옴
)

register_vector(conn)  # 벡터 변환 활성화
cur = conn.cursor()

# DB 초기화 (기존 테이블 삭제 후 재생성)
dim = model.get_sentence_embedding_dimension()
cur.execute("CREATE EXTENSION IF NOT EXISTS vector;")
cur.execute("DROP TABLE IF EXISTS review_vectors;")  # 테이블 완전 삭제
cur.execute(f"""
    CREATE TABLE review_vectors (
        id SERIAL PRIMARY KEY,
        review TEXT,
        embedding vector({dim})
    );
""")
conn.commit()
```

```
# 3단계: 벡터 저장
for review, emb in zip(reviews, embeddings):
    cur.execute(
        "INSERT INTO review_vectors (review, embedding) VALUES (%s, %s)",
        (review, np.array(emb, dtype=np.float32))
    )
conn.commit()

# 4단계: 유사도 검색
query = "배송이 느렸어요"
query_vec = model.encode([query])[0].astype(np.float32)

print("\n유사도 검색 결과:")
cur.execute(
    """
    SELECT review, embedding <=> %s AS cosine_distance
    FROM review_vectors
    ORDER BY embedding <=> %s
    LIMIT 3;
    """,
    (query_vec, query_vec)
)
for review, dist in cur.fetchall():
    print(f"코사인거리: {dist:.4f} | 리뷰: {review}")
```

▼ **결과**

```
유사도 검색 결과:
코사인거리: 0.0783 | 리뷰: 배송이 빠르고 제품도 좋아요.
코사인거리: 0.0990 | 리뷰: 배송은 느렸지만 포장은 안전했어요.
코사인거리: 0.1253 | 리뷰: 생각보다 배송이 오래 걸렸어요.
```