1. API Key 발급 및 설정

From dotenv import load_dotenv

Dotenv

: 민감한 정보나 환경에 따른 설정을 .env 파일에 저장해 관리할 수 있게 해주는 도구

➔ 보안 강화, 환경별 설정 분리, 개발 편의성 증진 등의 장점이 있다

API key 사용을 위한 결제 후 받은 api key를 OPEN_AI_KEY="" 형식으로 .env파일에 입력

2. ChatOpenAI

Import logging

: LLM 호출시 발생 가능한 다양한 문제 추적 위함

From langchain_openai import ChatOpenAI

llm = ChatOpenAI(~)

: Langchain이 제공하는 ChatOpenAI기능을 불러온다

: ChatOpenAI를 기반으로 하는 객체를 생성한다

이때 ~~ 안에 들어갈 내용은 챗 모델을 다루기 위한 설계도 역할을 한다

Llm_with_logprob

LogProb : 텍스트에 대한 모델의 토큰 확률의 로그값

토큰이란? : LLM이 문장을 쪼개는데 사용하는 단위

확률이 높을수록 0에 가까운 음수값

확률이 낮을수록 0에서 멀어지는 음수값

➔ LLM은 확률이 가장 높은(logprob값이 가장 큰) 값을 성공확률이 높은 예측이라

고 생각한다.

Answer = llm.stream

스트리밍: 데이터를 한번에 다 받지 않고 조금씩 나누어 처리하는 방식

Get_openai_callback() as cb;

: api 호출에 대한 토큰 사용량, 비용 추적 위한 객체

Answer.response_metadata에 토큰 사용에 대한 정보가 담기고

이후 응답에 대해서 토큰의 존재 유무(같은 응답을 했었는지)에 따라 저장된 캐시에서 불러오는 등의 방식으로 토큰 절약

3. LCEL

: Langchain의 복잡성이라는 단점을 극복하기 위해 내부동작을 직관적으로 연결할 수 있게 만든 표현 언어

- 파이프 연산자를 사용해 연결 (데이터 연결 파이프라인)

➔ 프롬프트 템플릿의 활용

: 프롬프트(명령), 모델(llm), parser(출력) 등이 하나의 흐름으로 연결됨

Chain이 ()로 묶이는게 아닌 prompt | model | ~~ 등으로 묶임

4. Runnable

: LCEL의 구성요소들을 쉽게 만들수 있도록 일반화한 실행 가능한 단위

표준 인터페이스인 만큼 포함되는 메인 메소드들이 존재한다

- Stream, invoke, batch 등 // 동기적 비동기적 메소드들 모두 존재

● 동기, 비동기란? 작업을 실행하고 기다리는 방식의 차이
프로그램이 한 작업을 끝낼때까지 기다릴 것인가, 동시에 다른 작업을 할 것인가

For token in chain.stream(topic : ~)

Chain.invoke(topic: ~)

Chain.batch(topic : ~ , topic: ~)

: 주어진 토픽에 대한 스트림 생성 및 내용 출력

: 주제를 인자로 받아 주제에 대한 처리 수행(딕셔너리 전달)

: 여러 개의 딕셔너리를 포함하는 리스트 처리 수행

비동기 배치 시...(async batch)

Abatch, ainvoke, asteam으로 메소드 생성

Await abatch_process로 일괄적으로 프로세스가 완료될때까지 대기

Runnable은 parallel(병렬성)이라는 방법을 사용할 수 있다

Chain1 = (prompt | model | StroutParser())

Chain2 = (동일)

Combined = RunnableParallel(~~ = chain1, ~~ = chain2)

Combined.invoke사용해서 각 주제에 대해 할당된 체인 호출