chat_model_init_example

> 'from langchain.chat_models import init_chat_model'
>
> 'init_chat_model()' → LLM 모델을 더 간결, 범용적으로 사용 가능
>
> OpenAI GPT-4o ↔ Google Gemini 2.5 비교 실습

1. 목적

- 동일한 질문을 두 모델에 전달하여 **temperature**와 **max_output_tokens** 차이에 따른 응답 비교 수행
- `.env` 파일에 저장된 API 키를 활용하여 모델 교체를 간단하게 구현
- `init_chat_model()`을 사용하여 제공사별 import 없이 모델 초기화 가능

---

2. 환경 변수 설정

`.env` 파일 예시:

```
OPENAI_API_KEY=sk-...
GOOGLE_API_KEY=AIza...
```

---

3. 전체 코드

```python
import os
from dotenv import load_dotenv
from langchain.chat_models import init_chat_model

# 1. .env 파일 로드
load_dotenv()

# 2. 환경 변수에서 API 키 가져오기
openai_key = os.getenv("OPENAI_API_KEY")
google_key = os.getenv("GOOGLE_API_KEY")

# 3. LLM 생성 함수 정의
def get_llm(provider: str, temperature: float, max_tokens: int):
    """
    provider: 'openai' 또는 'google'
    temperature: 창의성/랜덤성 조절 값 (0.0 ~ 1.0)
    max_tokens: 한 번에 생성할 최대 토큰 수
    """
    if provider == "openai":
        # OpenAI GPT-4o 모델 초기화
        return init_chat_model(
            model="gpt-4o",                    # 모델명
```

```
            model_provider="openai",            # 제공사
            temperature=temperature,            # 창의성 조절
            max_output_tokens=max_tokens,       # 최대 토큰 수
            api_key=openai_key                  # API 키
        )

    elif provider == "google":
        # Google Gemini 2.5 Flash Lite 모델 초기화
        return init_chat_model(
            model="gemini-2.5-flash-lite",
            model_provider="google_genai",
            temperature=temperature,
            max_output_tokens=max_tokens,
            api_key=google_key
        )

    else:
        raise ValueError("지원하지 않는 provider입니다.")

# 4. 비교할 질문 정의
prompt = "인공지능이 교육 분야에 미치는 긍정적인 영향 3가지를 간단히 설명해줘."

# 5. 모델별 설정 목록
models = [
    ("openai", 0.3, 512),       # GPT-4o: 창의성 낮음, 짧은 답변
    ("google", 0.8, 1024)       # Gemini: 창의성 높음, 긴 답변
]

# 6. 실행 및 결과 출력
for provider, temp, tokens in models:
    llm = get_llm(provider, temp, tokens)       # 모델 초기화
    response = llm.invoke(prompt)               # 질문 전달
    print(f"\n=== {provider.upper()} | temp={temp} | max_tokens={tokens}
===")
    print(response.content)                     # 모델 응답 출력
```

---

4. 코드 설명

4.1 `get_llm()` 함수

- **목적**: 모델 제공사와 파라미터를 입력받아 해당 모델 객체를 반환
- **장점**:
  - 제공사별 import 불필요
  - 모델 교체 시 코드 수정 최소화
  - 파라미터 변경 용이

4.2 `temperature`와 `max_output_tokens`

- **temperature**:
  - 낮을수록 일관되고 예측 가능한 답변

    - 높을수록 창의적이고 다양한 답변
- **max_output_tokens**:
    - 응답 길이 제한
    - 값이 작으면 짧은 답변, 크면 긴 답변 가능

4.3 실행 흐름

1. `.env`에서 API 키 로드
2. `models` 리스트에 비교할 모델과 파라미터 설정
3. `for` 루프를 돌며 각 모델 초기화 → 질문 전달 → 응답 출력

5. 실행 예시 출력

```
=== OPENAI | temp=0.3 | max_tokens=512 ===
1. 맞춤형 학습 제공: 학생 개개인의 수준과 속도에 맞춘 학습 자료 제공
2. 반복 학습 지원: 필요한 개념을 여러 번 복습 가능
3. 교사 업무 경감: 채점, 자료 준비 등 자동화

=== GOOGLE | temp=0.8 | max_tokens=1024 ===
1. 개인화된 학습 경험: 학습자의 수준, 관심사, 학습 스타일에 맞춘 맞춤형 콘텐츠 제공
2. 실시간 피드백과 상호작용: AI 튜터가 즉각적인 피드백과 다양한 학습 경로 제시
3. 교육 자원 접근성 확대: 지역·경제적 제약 없이 다양한 교육 자료와 전문가 지식 활용 가능
```

---

6. 기대 효과

- **모델 교체 용이성**: `get_llm()` **호출 시** `provider`**만 변경**
- **파라미터 실험 편의성**: `temperature`와 `max_output_tokens`를 쉽게 조정 가능
- **응답 비교 분석**: 같은 질문에 대한 모델별·설정별 응답 차이 확인 가능
