
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

11. `폴백` *(fallback)* `모델 지정`

1) `폴백` *(fallback)*

- `LLM` 애플리케이션의 문제: 다양한 오류 및 실패
  - `LLM API 문제`, `모델 출력 품질 저하`, `다른 통합 관련 이슈` 등
- **`fallback`** 기능
  - `LLM` 문제 처리 및 격리에 활용 가능
  - **`LLM` 수준 외 `전체 실행 가능한 수준에 적용할 수 있음`**

2) **`LLM API Error에 대처 방법`**

- `fallback` 사용의 가장 일반적인 사례 중 하나 = **`LLM API 오류 처리`**
- **`LLM API 요청 실패 원인`**
  - `API` 다운
  - `속도 제한에 도달`
  - 기타 등등
- **`중요`**
  - **기본적으로 많은 LLM wrapper (*래퍼*) = 오류 포착 및 재시도**
  - `fallback` 사용 시 → **`기본 동작 해제 후 사용`해야 함**
    - *그렇지 않으면 첫 번째 래퍼가 계속 재시도 → 실패 발생하지 않음*
- 사전에 `VS Code` 터미널에 설치되어 있는지 확인하기

```
pip install -qU langchain langchain-openai
```

✦

3) `RateLimitError` 발생 경우에 대한 모의 테스트

- `환경설정`

- `LLM` 설정
  - ① `gpt-4o-mini`
  - ② `gemini-2.5-flash-lite`

```
import getpass
import os

# 2nd OpenAI API key사용해 OpenAI 모델 초기화
OPEN_API_KEY = os.getenv("OPENAI_API_KEY2")

if not os.environ.get("OPENAI_API_KEY"):
    os.environ["OPENAI_API_KEY"] = getpass.getpass("Enter API key for OpenAI: ")

from langchain_openai import ChatOpenAI

model = ChatOpenAI(model="gpt-4o-mini", temperature=0)               # 4.4s
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

from dotenv import load_dotenv
import os

# LLM 초기화
# API 키 확인
if not os.getenv("GOOGLE_API_KEY"):
    os.environ["GOOGLE_API_KEY"] = input("Enter your Google API key: ")

# LLM 생성하기
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,
)
```

- `gemini-2.5-flash-lite` - (`0.7s`)

```
E0000 00:00:1760059917.561864 2692561 alts_credentials.cc:93] ALTS creds ignore
```

① `rate limit만 설정`

```
from unittest.mock import patch

import httpx
from openai import RateLimitError

request = httpx.Request("GET", "/")                                    # GET 요청 생성

# 200 상태 코드와 함께 응답을 생성하기
response = httpx.Response(
    200, request=request
)

# 에러메시지 출력하기
# "rate limit" 메시지와 응답 및 빈 본문을 포함하는 RateLimitError를 생성합니다.
error = RateLimitError("rate limit", response=response, body="")
```

- `openai_llm` 변수 = `OpenAI` 객체 생성
- `max_retries` 매개변수 `=` `0`
  - *`API 호출비용 제한 등으로 인한 재시도 방지`*
- `with_fallbacks` 메서드 사용 → `gemini`를 `fallback` `LLM`으로 설정 → `llm` 변수에 할당하기

```
# OpenAI의 ChatOpenAI 모델을 사용해 openai_llm 객체 생성하기
openai_llm = ChatOpenAI(max_retries=0)          # 속도 제한 등으로 인한 재시도 방지하기

# Google의 gemini 모델을 사용하여 gemini_llm 객체 생성하기
gemini_llm = gemini_lc

# openai_llm을 기본으로 사용 → 실패 시 gemini_llm을 대체로 사용하도록 설정
llm = openai_llm.with_fallbacks([gemini_llm])
```

```
# OpenAI LLM → 오류 발생시키기

with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
        # 질문 OpenAI LLM에 전달하기
        print(openai_llm.invoke("Why did the chicken cross the road?"))
                                # "닭이 길을 건넌 이유는 무엇일까요?"
    except RateLimitError:
        # 오류 발생 → 오류 메시지 출력
        print("에러 발생")
```

- `OpenAI 오류 발생시키기` - (`0.2s`)

```
에러 발생
```

② `rate limit(API 호출비용) 제한 오류 → 대체 모델 사용하기`

- `OpenAI API`의 `rate limit` *(비용 제한)* → 오류 발생 → 동작 테스트
  - `OpenAI`의 `GPT` 모델 시도 → `ERROR` 발생 → `fallback`모델인 `gemini-2.5-flash-lite`이 대신 추론 수행
  - **`with_fallback()` = 대체 모델 설정 → 대체 모델이 성공적 수행 시 `RateLimitError` 발생** ❌

```
# OpenAI API 호출 시 에러가 발생하는 경우 gemini로 대체하는 코드

with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
        # 질문을 언어 모델에 전달 → 응답 출력하기
        print(llm.invoke("대한민국의 수도는 어디야?"))              # fallback을 적용한 l

    except RateLimitError:
        # RateLimitError 발생 → "에러 발생" 출력하기
        print("에러 발생")
```

- `fallback`이 적용된 `llm` 사용됨 - (`1.0s`)

```
content='대한민국의 수도는 **서울**입니다.' additional_kwargs={} response_metadata={'pr
```

③ `llm.with_fallbacks()` 설정한 모델 = **일반 `runnable`모델과 동일하게 작동**

```
from langchain_core.prompts import ChatPromptTemplate

# 프롬프트 생성
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "질문에 짧고 간결하게 답변해 주세요.",                          # 시스템 역할 설
        ),
        ("human", "{country} 의 수도는 어디입니까?"),                       # 사용자 질문 템
    ]
)
```

```
# chain 생성
chain = prompt | llm  # 프롬프트와 언어 모델을 연결하여 체인 생성
# chain = prompt | ChatOpenAI() # 이 코드를 실행하면 "오류 발생" 문구가 출력됨
```

```
with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
```

```
        print(chain.invoke({"country": "대한민국"}))                    # 체인을 호출하여 결
    except RateLimitError:                                              # API 비용 제한 오
        print("오류 발생")
```

- **`llm.with_fallback()` → 일반 `runnable` 모델에서 작동** - (`1.2s`)

```
content='서울' additional_kwargs={} response_metadata={'prompt_feedback': {'block
```

---

4) `처리해야 할 오류를 구체적으로 명시한 경우`

- **불필요한 `fallback` 호출을 줄이고, 오류 처리의 효율성을 높일 수 있음**
  - 특정 오류를 처리하기 위해 `fallback`이 호출되는 시점을 더 명확하게 지정 가능 → `fallback` 메커니즘이 동작하는 상황을 보다 세밀하게 제어 가능
  - *예시: `특정 예외 클래스`, `오류 코드` 지정 → 해당 오류가 발생했을 때만 `fallback` 로직 실행되도록 설정*

- **`exceptions_to_handle`** 파라미터

목적	예외 유형 (OpenAI 기준)	설명
속도 제한 (`Rate Limiting`)	RateLimitError	API 호출 횟수가 너무 많아 일시적으로 차단되었을 때, 다른 API로
`API 키 오류`	AuthenticationError	API 키가 잘못되었거나 만료되었을 때, 백업 키를 사용하는 다른 L
연결 / 서버 오류	APIConnectionError, InternalServerError	네트워크 불안정이나 LLM 제공사 서버의 일시적인 문제로 연결이

① 교재와 동일 - `exceptions_to_handle=(KeyboardInterrupt,)`

```
# openai_llm을 기본으로 사용 → 실패 시 gemini_llm을 대체로 사용하도록 설정
llm2 = openai_llm.with_fallbacks(
    [gemini_llm],                                   # 대체 LLM = gemini
    exceptions_to_handle=(KeyboardInterrupt,),      # 예외 처리 대상 지정
)
```

```
# chain 생성하기

chain2 = prompt | llm2
```

```
with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
        # 체인 호출 → 결과 출력해보기
        print(chain2.invoke({"country": "대한민국"}))
    except RateLimitError:
```

```
        # RateLimitError 예외 발생 시 "오류 발생" 출력하도록 설정
        print("오류 발생")
```

- **`exceptions_to_handle=(KeyboardInterrupt,)`** - (`0.0s`)

  ```
  오류 발생
  ```

- - `exceptions_to_handle` 파라미터에서 `KeyboardInterrupt` 예외가 발생할 경우에만 `fallback`이 구동되도록 설정했기 때문
  - `KeyboardInterrupt`를 제외한 모든 예외에서는 `fallback`이 발생하지 않음

② API 호출 강제 발생시키기 - `exceptions_to_handle=(APIConnectionError)`

```
from openai import RateLimitError, APIConnectionError
from unittest.mock import patch                 # 오류를 강제로 발생시키기 위해 임포트

# 예시용 더미 오류 핸들러 함수
def error(*args, **kwargs):
    # 이 함수가 호출 시 → APIConnectionError를 강제로 발생시킴
    raise APIConnectionError("네트워크 연결이 일시적으로 불안정합니다. (Mock Error)")
```

```
# 프롬프트 생성
prompt = ChatPromptTemplate.from_template("너는 전문 여행가이드야. {country}에 대해 한 문
```

```
# openai_llm을 기본으로 사용 → 실패 시 gemini_llm을 대체로 사용하도록 설정
llm3 = openai_llm.with_fallbacks(
    [gemini_llm],                                                    # 대체 LLM =
    exceptions_to_handle=(RateLimitError, APIConnectionError),       # 예외 처리 디
)
```

```
# chain 생성하기 - 프롬프트와 LLM을 연결하기

chain3 = prompt | llm3
```

```
print("--- Fallback (대체 실행) 테스트 시작 ---")
print("OpenAI API 호출이 강제로 APIConnectionError를 발생시킵니다.")
print("APIConnectionError는 exceptions_to_handle에 포함되어 있으므로, gemini LLM으로 자동

# 강제로 APIConnectionError를 발생시켜 Fallback을 유도
with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
        # 체인 호출 → APIConnectionError 발생 → Fallback 작동 = gemini의 응답 출력
        print(chain3.invoke({"country": "대한민국"}))
        result = chain3.invoke({"country": "대한민국"})
        print(f"\n✅ Fallback 성공! 최종 응답 모델: {result.response_metadata.get('mo
        print(f"응답 내용: {result.content}")
    except Exception as e:
```

```
        # 만약 설정된 예외 외의 다른 예외 발생시 출력
        print(f"\n❌ 예상치 못한 오류 발생: {type(e).__name__} - {e}")
```

- **② API 호출 강제 발생시키기 / `exceptions_to_handle=(APIConnectionError)`** - (`5.9s`)

```
--- Fallback (대체 실행) 테스트 시작 ---
OpenAI API 호출이 강제로 APIConnectionError를 발생시킵니다.
APIConnectionError는 exceptions_to_handle에 포함되어 있으므로, gemini LLM으로 자동 전환됩
```

```
AIMessage(content='저는 대한민국을 "역동적인 문화와 첨단 기술이 조화를 이루며 끊임없이 발전하는 매
```

```
✅ Fallback 성공! 최종 응답 모델: gemini-2.5-flash-lite
응답 내용: 저는 대한민국을 "역동적인 문화와 첨단 기술이 조화를 이루며 끊임없이 발전하는 매력적인 나라"라
```

③ API 키 오류 강제 발생시키기 - `exceptions_to_handle=(AuthenticationError)`

```
# AuthenticationError 추가
from openai import RateLimitError, APIConnectionError, AuthenticationError
from unittest.mock import patch
```

```
# 예시용 더미 오류 핸들러 함수
def error(*args, **kwargs):
    # 이 함수가 호출 시 → AuthenticationError를 강제로 발생시킴
    raise AuthenticationError("제공된 API 키가 유효하지 않거나 만료되었습니다. (Mock Error)"
```

```
# 프롬프트 생성
prompt = ChatPromptTemplate.from_template("너는 전문 여행가이드야. {country}에 대해 한 문
```

```
# openai_llm을 기본으로 사용 → 실패 시 gemini_llm을 대체로 사용하도록 설정

llm4 = openai_llm.with_fallbacks(
    [gemini_llm],                                                    # 대체 LLM =
    # 예외 처리 대상 지정에 추가하기
    exceptions_to_handle=(RateLimitError, APIConnectionError, AuthenticationError
)
```

```
# chain 생성하기 - 프롬프트와 LLM을 연결하기

chain4 = prompt | llm4
```

```
print("--- Fallback (대체 실행) 테스트 시작 ---")
print("OpenAI API 호출이 강제로 APIConnectionError를 발생시킵니다.")
print("APIConnectionError는 exceptions_to_handle에 포함되어 있으므로, gemini LLM으로 자동
```

```
# 강제로 APIConnectionError를 발생시켜 Fallback을 유도
with patch("openai.resources.chat.completions.Completions.create", side_effect=er
    try:
        # 체인 호출 → APIConnectionError 발생 → Fallback 작동 = gemini의 응답 출력
        print(chain4.invoke({"country": "대한민국"}))
        result = chain4.invoke({"country": "대한민국"})
        print(f"\n✅ Fallback 성공! 최종 응답 모델: {result.response_metadata.get('mo
        print(f"응답 내용: {result.content}")
    except Exception as e:
        # 만약 설정된 예외 외의 다른 예외 발생시 출력
        print(f"\n❌ 예상치 못한 오류 발생: {type(e).__name__} - {e}")
```

- **③ API 키 오류 강제 발생시키기 / `exceptions_to_handle=(AuthenticationError)`** - (`1.8s`)

```
--- Fallback (대체 실행) 테스트 시작 ---
OpenAI API 호출이 강제로 APIConnectionError를 발생시킵니다.
APIConnectionError는 exceptions_to_handle에 포함되어 있으므로, gemini LLM으로 자동 전환됩
```

```
AIMessage(content='저는 대한민국을 "역동적인 문화와 첨단 기술이 조화를 이루며 끊임없이 발전하는 매
```

```
✅ Fallback 성공! 최종 응답 모델: gemini-2.5-flash-lite
응답 내용: 저는 대한민국을 "역동적인 문화와 첨단 기술이 조화를 이루며 끊임없이 발전하는 매력적인 나라"라
```

---

5) `fallback에 여러 모델을 순차적으로 지정`

- `fallback` 모델 = **`여러 개의 모델 지정 가능`** → **`순차적으로 시도`**

```
from langchain.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser


# 프롬프트 생성
prompt_template = (
    "질문에 짧고 간결하게 답변해 주세요.\n\nQuestion:\n{question}\n\nAnswer:"
)

prompt = PromptTemplate.from_template(prompt_template)
```

- **2개의 chain 생성 - `오류를 발생하는 chain`, `정상적인 chain`**

```
# 쉽게 오류를 발생시킬 수 있도록 잘못된 모델 이름을 사용 → 체인 생성하기
chat_model = ChatOpenAI(model_name="gpt-fake")
```

```
bad_chain = prompt | chat_model
```

```
# fallback 체인 생성하기

fallback_chain1 = prompt | ChatOpenAI(model="gpt-3.6-turbo")        # 오류
fallback_chain2 = prompt | gemini_llm                               # 정상
fallback_chain3 = prompt | ChatOpenAI(model="gpt-4o-mini")          # 정상
```

```
# 두 개의 체인 결합 → 최종 체인 생성하기
chain = bad_chain.with_fallbacks(
    [fallback_chain1, fallback_chain2, fallback_chain3])
```

```
# 최종 생성된 체인 호출 → 입력값 전달하기

chain.invoke({"question": "대한민국의 수도는 어디야?"})
```

- **`fallback에 여러 모델을 순차적으로 시도`**
  - **`fallback2`** / **`gemini-2.5-flash-lite`** - (`1.6s`)

```
AIMessage(content='서울', additional_kwargs={}, response_metadata={'prompt_f
```

*

- **`fallback3`** / **`gpt-4o-mini`** - (`5.9s`)

```
AIMessage(content='서울입니다.', additional_kwargs={'refusal': None}, response_met
```

- next: ***`CH14. 체인 (Chains)`***
