LangChain 외부 데이터 연결

- LangChain: 문서, 데이터베이스, API 등 다양한 외부 소스와 쉽게 연결할 수 있습니다.
- ChatGPT API: 외부 데이터 통합을 위해서는 추가적인 개발이 필요합니다.

라이브러리 설치

```
pip install openai
pip install langchain
pip install langchain-openai
pip install langchain_community
pip install langchain-core
pip install faiss-cpu
pip install chardet
```

FAISS 설치

- CPU 버전 설치

```
pip install faiss-cpu
```

- GPU 버전 설치

```
pip install faiss-gpu
```

In [3]:
```
import langchain
import langchain_community
import faiss
import openai

print("langchain 버전 : ", langchain.__version__)
print("langchain_community 버전 : ", langchain_community.__version__)
print("faiss-cpu 버전 : ", faiss.__version__)
print("openai 버전 : ", openai.__version__)
```

```
langchain 버전 :  0.3.7
langchain_community 버전 :  0.3.5
faiss-cpu 버전 :  1.9.0
openai 버전 :  1.54.1
```

01. ChatGPT API를 활용한 외부 데이터 연결

- 문서 로딩을 직접 구현이 필요.
- 문서 내용과 질문을 결합하여 프롬프트 생성이 필요.
- 토큰 제한으로 인해 긴 문서 처리가 어려울 수 있다.

In [5]:
```
import os
from getpass import getpass
```

```
import chardet

# API 키를 안전하게 입력받음
# API_KEY = getpass.getpass("OpenAI API 키를 입력하세요: ")
os.environ["OPENAI_API_KEY"] = getpass("OpenAI API 키를 입력하세요: ")
```

In [6]:
```
from openai import OpenAI
```

In [7]:
```
# 문서 내용 로드 (여러 인코딩 시도)
def load_document(file_path):
    encodings = ['utf-8', 'ISO-8859-1', 'Windows-1252']
    for encoding in encodings:
        try:
            with open(file_path, 'r', encoding=encoding) as file:
                return file.read()
        except UnicodeDecodeError:
            continue
    raise ValueError("파일을 읽을 수 없습니다. 지원되는 인코딩이 없습니다.")

document_content = load_document('document.txt')
client = OpenAI(api_key=os.environ["OPENAI_API_KEY"])

# 질문-답변
def ask_question(document, question):
    prompt = f"다음 문서를 바탕으로 질문에 답하세요:\n\n{document}\n\n질문: {ques

    # completions: 단일 문자열 프롬프트
    # 각 요청이 독립적이며, 이전 맥락을 자동으로 유지하지 않음.
    # completions: 역할 지정 기능이 없음.
    next = client.completions.create(
        model="gpt-3.5-turbo-instruct",
        prompt=prompt,
        max_tokens=100,
        n=3
    )

    ai_message = next.choices[0].text
    return ai_message

query = "문서의 주요 내용은 무엇인가요?"
response = ask_question(document_content, query)
print(response)
```

```
인공지능의 발전과 미래에 대한 내용입니다. 인공지능의 의미, 주요 분야, 응용 분야, 윤리적, 사회적 문제, 미래 전망 등을 다루고 있습니다.
```

02. LangChain을 활용한 외부 데이터 통합 예제

- 외부 텍스트 파일을 쉽게 로드 가능.
- 문서의 벡터 인덱스를 자동으로 생성.
- 생성된 인덱스를 기반으로 질문에 답변.

심플 초기 버전

- [자동화] VectorstoreIndexCreator를 사용하여 여러 단계를 자동으로 처리

- [자동화] 문서 로드, 벡터화, 저장소 생성, 검색기 생성 등 대부분의 작업을 내부적으로 처리.
- [단점] 고급 사용자 정의 워크플로우에는 적합하지 않음.
- [단점] 영구적으로 데이터를 저장하고 있지 않음.

In [8]:
```
from langchain.document_loaders import TextLoader       # 텍스트 파일을 로드 하는
from langchain.indexes import VectorstoreIndexCreator   # 문서의 벡터 인덱스를 생
from langchain_openai import OpenAIEmbeddings
from langchain.llms import OpenAI                       # OpenAI의 언어 모델을 사
```

In [9]:
```
# 문서 로드
loader = TextLoader('document.txt')

# 인덱스 생성
# VectorstoreIndexCreator를 사용하여 로드된 문서의 벡터 인덱스를 생성
# 이 과정에서 문서는 벡터로 변환되어 저장
embeddings = OpenAIEmbeddings()   # 벡터 임베딩을 처리하기 위한 클래스
index = VectorstoreIndexCreator(embedding=embeddings).from_loaders([loader])

# OpenAI LLM 생성
llm = OpenAI(temperature=0, max_tokens=150)

# 질문-답변
query = "문서의 주요 내용은 무엇인가요?"
response = index.query(query, llm=llm)
print(response)
```

```
c:\Users\daniel_wj\anaconda3\envs\langchain1\Lib\site-packages\langchain\indexes
\vectorstore.py:128: UserWarning: Using InMemoryVectorStore as the default vector
store.This memory store won't persist data. You should explicitlyspecify a vector
store when using VectorstoreIndexCreator
  warnings.warn(
C:\Users\daniel_wj\AppData\Local\Temp\ipykernel_23960\4002558161.py:11: LangChain
DeprecationWarning: The class `OpenAI` was deprecated in LangChain 0.0.10 and wil
l be removed in 1.0. An updated version of the class exists in the :class:`~langc
hain-openai package and should be used instead. To use it run `pip install -U :cl
ass:`~langchain-openai` and import as `from :class:`~langchain_openai import Open
AI``.
  llm = OpenAI(temperature=0, max_tokens=150)
```

문서의 주요 내용은 인공지능(AI)의 발전과 응용 분야, 그리고 AI 발전의 윤리적, 사회적 문제에 대한 논의입니다.

- /usr/local/lib/python3.10/dist-packages/langchain/indexes/vectorstore.py:128: UserWarning: Using InMemoryVectorStore as the default vectorstore.This memory store won't persist data. You should explicitlyspecify a vectorstore when using VectorstoreIndexCreator warnings.warn(
- 데이터를 영구적으로 저장하고 있지 않다는 메시지
  - 해결 - FAISS와 같은 벡터 저장소를 사용

03. FAISS 이용 개선 버전

In [10]:
```
# 필요한 모듈 임포트
from langchain.vectorstores import FAISS
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

```python
from langchain.document_loaders import TextLoader
from langchain_openai import OpenAIEmbeddings, ChatOpenAI

# 문서 로드
loader = TextLoader('document.txt')
documents = loader.load()

# 텍스트 분할
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=20
splits = text_splitter.split_documents(documents)

# 벡터 저장소 생성
embeddings = OpenAIEmbeddings()
vectorstore = FAISS.from_documents(splits, embeddings)  # FAISS를 벡터 저장소로

# OpenAI LLM 생성
llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0, max_tokens=150 )

# 질문-답변
query = "문서의 주요 내용은 무엇인가요?"
retriever = vectorstore.as_retriever()  # FAISS를 검색기로 설정

# 검색된 문서를 기반으로 응답 생성
retrieved_docs = retriever.invoke(query)  # 문자열 형태의 query 전달
context = " ".join([doc.page_content for doc in retrieved_docs])

# LLM 호출
prompt = f"다음 문서를 바탕으로 질문에 답변해주세요:\n\n{context}\n\n질문: {query
response = llm.invoke(prompt)  # 문자열 형태의 prompt 전달

# 결과 출력
print(response)
```

```
content='문서의 주요 내용은 인공지능(AI)의 발전과 그 응용 분야, 그리고 AI 기술의 발전에 따른 윤리적 및 사회적 문제를 다루고 있습니다. AI는 금융, 의료, 자율주행 자동차, 개인화된 추천 시스템, 스마트홈 기술, 기후 변화 예측 등 다양한 산업에서 혁신적인 변화를 일으키고 있으며, 머신러닝, 딥러닝, 자연어 처리, 컴퓨터 비전 등의 기술이 핵심 역할을 하고 있습니다. \n\n또한, AI의 발전은 개인정보 보호, 일자리 대체, AI의 편향성, 책임 있는 AI 개발과 사용 등 여러 윤리적, 사회적 문제를 동반하고 있으며' additional_kwargs={'refusal': None} response_metadata={'token_usage': {'completion_tokens': 150, 'prompt_tokens': 1612, 'total_tokens': 1762, 'completion_tokens_details': {'accepted_prediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 0}}, 'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_0ba0d124f1', 'finish_reason': 'length', 'logprobs': None} id='run-1a6856f0-9195-47dd-83e1-3237df1e2efe-0' usage_metadata={'input_tokens': 1612, 'output_tokens': 150, 'total_tokens': 1762, 'input_token_details': {'audio': 0, 'cache_read': 0}, 'output_token_details': {'audio': 0, 'reasoning': 0}}
```

코드의 한계

- 확장성 부족: 복잡한 워크플로우(예: 체인 연결) 구현이 어려움.
- 프롬프트 유연성 제한: 정적 프롬프트 생성.

04. 최종 구현 코드 - (주석 Simple)

- LangChain의 체인 기능(create_retrieval_chain)을 활용하여 검색과 답변 생성을 연결.
- 검색, 문서 결합, GPT 호출 등의 단계를 체인으로 자동화.

In [11]:
```python
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import create_retrieval_chain
from langchain.chains.combine_documents import create_stuff_documents_chain

from langchain_openai import OpenAIEmbeddings, ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate

from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import TextLoader

# 문서 로드
# TextLoader를 사용하여 document.txt 파일에서 텍스트 데이터를 읽어옵니다.
loader = TextLoader('document.txt')

# 문서 전체 내용을 documents 변수에 저장
documents = loader.load()

# 문서 분할
# chunk_size: 각 청크의 최대 문자 수
# chunk_overlap: 연속된 청크 간의 중복되는 문자 수 (문맥 유지를 위해)
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200
)
splits = text_splitter.split_documents(documents)

# 벡터 저장소 생성
embeddings = OpenAIEmbeddings()
vector_store = FAISS.from_documents(splits, embeddings)

# 검색기 생성
retriever = vector_store.as_retriever()

# LLM 설정
llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0, max_tokens=150 )

# 동적 프롬프트 생성을 위한 템플릿 정의.
# 프롬프트 템플릿에서 정의된 변수(context와 question)는 체인이 실행될 때 반드시 제
prompt = ChatPromptTemplate.from_template("""
다음 문서를 바탕으로 질문에 답변해주세요.
문서: {context}
질문: {input}
""")

# 문서 결합 체인 생성
document_chain = create_stuff_documents_chain(llm, prompt)

# 검색 체인 생성
retrieval_chain = create_retrieval_chain(retriever, document_chain)

# 질문-답변
query = "문서의 주요 내용은 무엇인가요?"
```

```
# retrieval_chain.invoke에 필요한 변수 제공
# 방법 1: input 키 사용
response = retrieval_chain.invoke({
    "input": query
})

print(response["answer"])
```

문서의 주요 내용은 인공지능(AI)의 발전과 그 응용 분야, 그리고 AI 발전에 따른 윤리적 및 사회적 문제를 다루고 있습니다.

1. **AI의 역할과 응용 분야**: AI는 금융, 의료, 자율주행 자동차, 개인화된 추천 시스템, 스마트홈 기술, 기후 변화 예측 등 다양한 산업에서 혁신적인 변화를 일으키고 있으며, 머신러닝, 딥러닝, 자연어 처리, 컴퓨터 비전 등의 기술이 활용되고 있습니다.

2. **AI 발전의 윤리적, 사회적 문제**: AI의 발전은 개인정보 보호, 일자리 대체, AI의 편향성, 책임

좀 더 자세한 설명

In [12]:
```
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import create_retrieval_chain
from langchain.chains.combine_documents import create_stuff_documents_chain

from langchain_openai import OpenAIEmbeddings, ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate

from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import TextLoader

# 문서 로드
# TextLoader를 사용하여 document.txt 파일에서 텍스트 데이터를 읽어옵니다.
loader = TextLoader('document.txt')

# 문서 전체 내용을 documents 변수에 저장
documents = loader.load()

# 문서 분할
# chunk_size: 각 청크의 최대 문자 수
# chunk_overlap: 연속된 청크 간의 중복되는 문자 수 (문맥 유지를 위해)

# 긴 문서를 처리하기 위해 1,000자 크기의 청크로 분할
# chunk_overlap=200: 각 청크 사이에 200자를 겹치게 하여 문맥을 유지합니다.
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200
)

# 청크화된 텍스트를 splits 변수에 저장
splits = text_splitter.split_documents(documents)

# 벡터 저장소 생성
# OpenAIEmbeddings: 텍스트를 벡터로 변환하는 임베딩 모델을 사용
# FAISS: 변환된 벡터를 저장하고 유사도 검색을 수행하는 벡터 저장소 라이브러리
embeddings = OpenAIEmbeddings()

# vector_store에 텍스트 청크의 벡터가 저장
vector_store = FAISS.from_documents(splits, embeddings)
```

```python
# 검색기 생성
# FAISS 저장소에서 텍스트를 검색할 수 있는 검색기를 생성
retriever = vector_store.as_retriever()

# LLM 설정
llm = ChatOpenAI(
    model_name="gpt-4o-mini", temperature=0,  max_tokens=150
)

# 프롬프트 템플릿 생성
# 동적 프롬프트 생성을 위한 템플릿 정의.
#   - 프롬프트 템플릿에서 정의된 변수(context와 question)는 체인이 실행될 때 반드
# GPT 모델에 전달할 프롬프트 형식을 정의
# {context}: 검색된 문서 내용이 삽입됩니다.
# {input}: 사용자의 질문이 삽입됩니다.
prompt = ChatPromptTemplate.from_template("""
다음 문서를 바탕으로 질문에 답변해주세요.

문서: {context}

질문: {input}
""")

# 문서 결합 체인 생성
# 검색된 문서를 결합하고 GPT 모델에 전달하기 위한 체인을 생성
document_chain = create_stuff_documents_chain(llm, prompt)

# 검색 체인 생성
# 검색기와 문서 결합 체인을 연결하여 검색 및 질문-답변 기능을 통합한 체인을 생성
retrieval_chain = create_retrieval_chain(retriever, document_chain)

# 질문-답변
query = "문서의 주요 내용은 무엇인가요?"

# retrieval_chain.invoke에 필요한 변수 제공
# retriever가 유사한 문서를 검색해 context로 전달
# document_chain이 context와 query를 사용해 GPT 모델의 응답을 생성
response = retrieval_chain.invoke({
    "input": query
})

print(response["answer"])
```

문서의 주요 내용은 인공지능(AI)의 발전과 그 응용 분야, 그리고 AI 기술의 발전에 따른 윤리적 및 사회적 문제를 다루고 있습니다.

1. **AI의 역할과 응용 분야**: AI는 금융, 의료, 자율주행 자동차, 개인화된 추천 시스템, 스마트홈 기술, 기후 변화 예측 등 다양한 산업에서 혁신적인 변화를 일으키고 있으며, 알고리즘 트레이딩, 사기 탐지, 신용 평가 등 금융 리스크 관리에서도 중요한 역할을 하고 있습니다.

2. **AI 발전의 윤리적, 사회적 문제**: AI의 발전은 개인정보 보호, 일자리 대체, AI의