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

---

6. `MultiVectorRetriever`

5) `LLM 모델로 재시도`

- **`로컬 LLM 모델로 재시도`**
  - `API` 비용 걱정 X
  - **`일관된 품질`** → 동일 모델 사용
  - **`한국어 지원`**: 한국어 지원 특화 모델 사용

- 한국어 지원 `LLM` 모델
  - 추천 모델 1: `Qwen2.5`
    - 약 `7b`
    - 품질: `한국어`, `영어` 모두 우수 / `Chat-GPT-3.5` 수준
    - 속도: 약 `20~30` 토큰/초 *(로컬이라 약간 느릴 수 있으나 무제한)*

```
ollama pull qwen2.5:7b
```

  - 추천 모델 2: `Gemma 3` (구글, 140개 언어 지원)
    - 품질: `140개 언어` 지원 / `Chat-GPT-3.5` 수준
    - 속도: 약 `15~25` 토큰/초

```
ollama pull gemma2:9b
```

  - 추천 모델 3: `DeepSeek-R1` (코딩, 추론 강력)

```
ollama pull deepseek-r1:8b
```

  - 추천 모델 4: **`Llama 3.3`** (메타, 범용)

```
ollama pull llama3.3:8b
```

- 📊 **로컬 LLM 모델 성능 비교표**

모델	한국어	영어	보안	속도	추천도
EXAONE 3.5	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	**최강 추천**
Llama-3-Ko	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	차선책
Bllossom	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐	교육용
DeepSeek	⭐⭐⭐	⭐⭐⭐⭐⭐	⚠️	⭐⭐⭐⭐	보안 우려

- `한국어 로컬 LLM 모델`
  - **a.** **`EXAONE 3.5`** *(LG AI Research)*

```
# Ollama에서 설치
```

```
ollama pull exaone3.5:7.8b

# HuggingFace
LGAI-EXAONE/EXAONE-3.5-7.8B-Instruct
```

- 장점:
  - `LG AI Research` 개발
  - `한국어 최고 성능`: *벤치마크 1위*
  - `MeCab` 기반 토크나이저: 한국어 교착어 특성 완벽 반영
  - `실사용 검증`: `LG` 계열사 실무 적용 중
  - `오픈소스`: `HuggingFace` 공개
- 성능:
  - 한국어: ★★★★★
  - 영어: ★★★★
  - 코딩: ★★★★
  - 추론: ★★★★★

- **b.** **`Llama-3-Open-Ko-8B`** *(beomi)*

```
# Ollama에서 설치
ollama pull beomi/llama-3-open-ko-8b

# HuggingFace
beomi/Llama-3-Open-Ko-8B
beomi/Llama-3-Open-Ko-8B-Instruct-preview
```

- 장점:
  - `Meta Llama 3 기반`: 세계 최고 모델 파인 튜닝
  - `beomi(이준범님) 개발`: *한국 LLM 커뮤니티 신뢰도 1위*
  - `지속적 업데이트`: 활발한 개발
  - `범용성`: 다양한 태스크 우수
- 성능:
  - 한국어: ★★★★★ *(최상급)*
  - 영어: ★★★★★ *(`Llama 3 성능`)*
  - 코딩: ★★★★ *(우수)*
  - 추론: ★★★★ *(우수)*

- **c.** **`Korean-Bllossom-8B`** *(서울과기대)*

```
# Ollama에서 설치
ollama pull korean-bllossom-8b

# HuggingFace
MLP-KTLim/llama-3-Korean-Bllossom-8B
```

- 장점:
  - `100GB+ 한국어 풀 튜닝`: 대용량 한국어 한습
  - `서울과기대 슈퍼컴퓨팅`: *공신력 있는 기관*
  - `한국어 특화`: 이중언어 모델
  - `교육/연구용 최적`: 학습 목적 완벽
- 성능:
  - 한국어: ★★★★★ *(최상급)*
  - 영어: ★★★★★ *(`Llama 3 성능`)*
  - 코딩: ★★★ *(양호)*
  - 추론: ★★★★ *(우수)*

---

2) `로컬 LLM 설치`

- `EXAONE 3.5` 선택
  - `보안 안전성`
    - `LG AI Research` : 대기업 공식 개발
    - `국내 기업` : 한국 법률/보안 기준 준수
    - `실사용 검증` : `LG` 계열사 실무 적용
    - `DeepSeek 우려 해소` : 보안 우려 해소

- `한국어 성능 우수`
  - `MeCab 토크나이저` : 교착어 특성 완벽
  - `한국어 벤치마크` : 세계 최고 성능
  - `한국어 코퍼스` : 대량 학습
  - `실제 테스트` : 커뮤니티 인정

- `MultiVectorRetriever 최적`
  - `요약 생성` : 뛰어난 이해력
  - `가설 쿼리` : 창의적 질문 생성
  - `일관성` : 동일 모델 사용
  - `속도` : `7.8b`로 빠름

- 설치 가이드
  - 1단계: `Ollma` **설치**

```
# Windows/Mac
https://ollama.com/download

# 설치 확인
ollama --version
```

- 2단계: `EXAONE 3.5` **다운로드**

```
# 최강 한국어 모델!
ollama pull exaone3.5:7.8b

# 다운로드 확인
ollama list
```

- 3단계: `LangChain` **통합**

```
# 랭체인 통합하기
from langchain_ollama import OllamaLLM

# EXAONE 3.5 초기화
llm = OllamaLLM(
    model="exaone3.5:7.8b",
    temperature=0
)

# 요약 생성 체인
summary_chain = (
    {"doc": lambda x: x.page_content}
    | ChatPromptTemplate.from_template(
        "다음 문서를 3-4문장으로 요약하세요:\n\n{doc}"
    )
    | llm
    | StrOutputParser()
```

```
)
```

- 사전 `VS Code` 터미널에 설치할 것

```
pip install langchain-ollama
```

```
from langchain_ollama import OllamaLLM

# Ollama 전역 서버에 연결!
llm_lama = OllamaLLM(model="exaone3.5:7.8b")

# 테스트
response = llm_lama.invoke("안녕하세요!")
print(response)                                          # 12.1s
```

- 셀 출력 (*`12.1s`*)

```
안녕하세요! LG AI Research에서 개발된 EXAONE입니다. 어떻게 도와드릴까요? 어떤 도움이 필요하신지 말씀해 주세요. 😊
```

---

3) `설정`

```
# API 키를 환경변수로 관리하기 위한 설정 파일
from dotenv import load_dotenv

# API 키 정보 로드
load_dotenv()                                # True
```

```
from langsmith import Client
from langsmith import traceable

import os

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_V2')
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANGCHAIN_API_KEY') else "설정되지 않음" # API 키 값은 직접 출력하지 않음

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API_KEY') and langchain_project:
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{langchain_tracing_v2}')")
    print(f"✅ LangSmith 프로젝트: '{langchain_project}'")
    print(f"✅ LangSmith API Key: {langchain_api_key_status}")
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.")
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다. 다음을 확인하세요:")
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있지 않습니다 (현재: '{langchain_tracing_v2}').")
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
✅ LangSmith 프로젝트: 'LangChain-prantice'
✅ LangSmith API Key: 설정됨
  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

- **`전처리 과정`**
  - 텍스트 파일에서 데이터 로드 → 로드된 문서들을 지정된 크기로 분할
  - 분할된 문서들 = 추구 벡터화 및 검색 등의 작업에 사용 가능

```
from langchain_community.document_loaders import PyMuPDFLoader
```

```
loader = PyMuPDFLoader("../10_Retriever/data/SPRI_AI_Brief_2023년12월호_F.pdf")
docs = loader.load()                                    # 0.3s
```

- **`docs`** 변수 = 데이터로부터 로드한 원본 도큐먼트

```
print(len(docs))                                        # 23
```

```
print(docs[5].page_content[:500])
```

- 6번째 페이지 일부 출력해보기

```
1. 정책/법제
2. 기업/산업
3. 기술/연구
4. 인력/교육
영국 AI 안전성 정상회의에 참가한 28개국, AI 위험에 공동 대응 선언
n 영국 블레츨리 파크에서 개최된 AI 안전성 정상회의에 참가한 28개국들이 AI 안전 보장을
위한 협력 방안을 담은 블레츨리 선언을 발표
n 첨단 AI를 개발하는 국가와 기업들은 AI 시스템에 대한 안전 테스트 계획에 합의했으며,
영국의 AI 안전 연구소가 전 세계 국가와 협력해 테스트를 주도할 예정
KEY Contents
£ AI 안전성 정상회의 참가국들, 블레츨리 선언 통해 AI 안전 보장을 위한 협력에 합의
n 2023년 11월 1~2일 영국 블레츨리 파크에서 열린 AI 안전성 정상회의(AI Safety Summit)에
참가한 28개국 대표들이 AI 위험 관리를 위한 '블레츨리 선언'을 발표
•선언은 AI 안전 보장을 위해 국가, 국제기구, 기업, 시민사회, 학계를 포함한 모든 이해관계자의 협력이
중요하다고 강조했으며,
```

---

4) `Chunk + 원본 문서 검색`

- 대용량 정보 검색 시: `더 작은 단위`로 정보를 `임베딩`하는 것이 유용할 수 있음
  - **`MultiVectorRetriever`** → 문서를 `여러 벡터`로 `저장`하고 `관리` 가능
  - **`docstore`** = 원본 문서 저장
  - **`vectoresotre`** = *`임베딩된 문서 저장`*
    - 문서룰 더 작은 단위로 나눠 더 정확한 검색이 가능
    - 원본 문서의 내용도 조회 가능

```
# 자식 청크를 인덱싱하는 데 사용할 벡터 저장소
import uuid
from langchain.storage import InMemoryStore
from langchain_chroma import Chroma
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain.retrievers.multi_vector import MultiVectorRetriever

# 기존 임베딩 모델: sentence-transformers/all-mpnet-base-v2 (768차원)
# 업그레이드된 임베딩 모델 사용하기: 다국어 고성능 모델 (1024차원)
multilingual_embeddings = HuggingFaceEmbeddings(
    model_name="sentence-transformers/paraphrase-multilingual-mpnet-base-v2",  # 1024차원
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
)

print("🎯 고성능 임베딩 모델 준비 완료!")

vectorstore = Chroma(
    collection_name="small_bigger_chunks",
    embedding_function=multilingual_embeddings,
)

# 부모 문서의 저장소 계층
store = InMemoryStore()

id_key = "doc_id"

# 검색기 (시작 시 비어 있음)
retriever = MultiVectorRetriever(
    vectorstore=vectorstore,
    byte_store=store,
    id_key=id_key,
```

```
)

# 문서 ID 생성하기
doc_ids = [str(uuid.uuid4()) for _ in docs]

# 두개의 생성된 id 확인하기
doc_ids                                                                    # 5.6s
```

- 🎯 고성능 임베딩 모델 준비 완료! (`5.6s`)

```
['f379a1c3-87d2-4c15-b993-1530cc12abff',
'309a6564-de98-4e33-a956-fabf9dd6a127',
'c743a429-9f88-4fcd-877d-70a9ec1fbd56',
'701310ba-f32e-4d15-8793-cb08a735774a',
'07f0b926-2472-46d5-b389-00329af04c66',
'c641e3e6-fae2-4eb1-8371-a49cf7e1a48c',
'4f1ebe8e-1426-407b-a798-064fc52d32c3',
'bde03cd1-9d66-49c2-a935-53c7cffe5f89',
'e31b5bf0-73d8-4329-8900-2489a37b94c2',
'7602af6f-a4af-4095-b98a-3ae351c7670a',
'ed729bf3-f9a6-49ca-b931-123c095f43d7',
'bfa9b89d-0cdd-4dd8-8a17-2fc5db340b0e',
'a7d79032-3b81-4687-bde0-6134d2dded1f',
'77fd3a60-7ef7-48c5-b8a7-e7696fc7dba4',
'3fa67910-842a-429a-b56c-180c611dd3d4',
'0cced66f-d690-4a81-8d79-39b35c05fa85',
'63de1be1-dba0-43c8-80fc-f98a8d2e2527',
'15999e87-d074-46f3-9fec-2d9cab88c294',
'4fa52a8d-b1fb-450c-90eb-f05f8522782a',
'0fff2d32-8584-4565-982c-b354878dfad8',
'9d919616-dc81-4750-be0a-d81bca4618eb',
'6a8e0a8f-8431-4b7a-9d7d-6b07a809fd65',
'50b81f53-cb6c-46bc-9875-b0a211f86115']
```

- **`문서 계층 분할기 생성`**
  - **`parent_text_splitter`**: 큰 청크로 분할하기 위한 객체
  - **`chile_text_splitter`**: 더 작은 청크로 분할하기 위한 객체

```
# RecursiveCharacterTextSplitter 객체 생성하기
parent_text_splitter = RecursiveCharacterTextSplitter(chunk_size=600)

# 더 작은 청크를 생성하는 데 사용할 분할기
child_text_splitter = RecursiveCharacterTextSplitter(chunk_size=200)        # 부모보다 작게 설정
```

- 더 큰 `Chunk` 인 `Parent` 문서 생성하기

```
parent_docs = []

for i, doc in enumerate(docs):
    _id = doc_ids[i]                                                # 현재 문서의 ID 가져오기
    parent_doc = parent_text_splitter.split_documents([doc])    # 현재 문서를 하위 문서로 분할

    for _doc in parent_doc:
        _doc.metadata[id_key] = _id                                 # metadata에 문서 ID 를 저장
    parent_docs.extend(parent_doc)
```

- **`parent_docs`** 에 기입된 **`doc_id`** 확인하기

```
# 생성된 Parent 문서의 메타데이터 확인하기

parent_docs[0].metadata
```

- `parent_docs`의 `doc_id` 확인하기

```
{'producer': 'Hancom PDF 1.3.0.542',
'creator': 'Hwp 2018 10.0.0.13462',
'creationdate': '2023-12-08T13:28:38+09:00',
'source': '../10 Retriever/data/SPRI AI Brief 2023년12월호_F.pdf',
'file_path': '../10 Retriever/data/SPRI AI Brief 2023년12월호_F.pdf',
'total_pages': 23,
```

```
'format': 'PDF 1.4',
'title': '',
'author': 'dj',
'subject': '',
'keywords': '',
'moddate': '2023-12-08T13:28:38+09:00',
'trapped': '',
'modDate': "D:20231208132838+09'00'",
'creationDate': "D:20231208132838+09'00'",
'page': 0,
'doc_id': 'f379a1c3-87d2-4c15-b993-1530cc12abff'}           # ✓ doc_id 확인하기
```

- 상대적으로 더 작은 **`Chunk`** 인 **`Child`** 문서 생성하기

```
child_docs = []

for i, doc in enumerate(docs):
    _id = doc_ids[i]                                                # 현재 문서의 ID 가져오기
    child_doc = child_text_splitter.split_documents([doc])      # 현재 문서를 하위 문서로 분할

    for _doc in child_doc:
        _doc.metadata[id_key] = _id                                 # metadata에 문서 ID 를 저장
    child_docs.extend(child_doc)
```

- **`child_docs`** 에 기입된 **`doc_id`** 확인하기

```
# 생성된 Child 문서의 메타데이터 확인하기

child_docs[0].metadata
```

- `child_docs`의 `doc_id` 확인하기

```
{'producer': 'Hancom PDF 1.3.0.542',
'creator': 'Hwp 2018 10.0.0.13462',
'creationdate': '2023-12-08T13:28:38+09:00',
'source': '../10_Retriever/data/SPRI_AI_Brief_2023년12월호_F.pdf',
'file_path': '../10_Retriever/data/SPRI_AI_Brief_2023년12월호_F.pdf',
'total_pages': 23,
'format': 'PDF 1.4',
'title': '',
'author': 'dj',
'subject': '',
'keywords': '',
'moddate': '2023-12-08T13:28:38+09:00',
'trapped': '',
'modDate': "D:20231208132838+09'00'",
'creationDate': "D:20231208132838+09'00'",
'page': 0,
'doc_id': 'f379a1c3-87d2-4c15-b993-1530cc12abff'}           # ✓ doc_id 확인하기
```

- 각각 분할된 청크의 수 확인하기

```
print(f"분할된 parent_docs의 개수: {len(parent_docs)}")
print(f"분할된 child_docs의 개수: {len(child_docs)}")
```

- 각 분할된 청크의 수 확인하기

```
분할된 parent_docs의 개수: 73
분할된 child_docs의 개수: 440
```

- 벡터저장소에 새롭게 생성한 작게 쪼개진 하위문서 집합 추가하기
- 다음: 상위 문서는 생성한 `UUID`와 맵핑하여 **`docstore`** 에 추가하기
  - **`mset()`** 메서드 → 문서 `ID`, 문서 내용 = **`key-value`** 쌍으로 문서 저장소에 저장

```
# 벡터 저장소에 parent + child 문서를 추가
retriever.vectorstore.add_documents(parent_docs)
retriever.vectorstore.add_documents(child_docs)
```

```
# docstore 에 원본 문서를 저장
retriever.docstore.mset(list(zip(doc_ids, docs)))                    # 22.3s
```

- 유사도 검색 수행하기
  - 가장 유사도가 높은 첫 번째 문서 조각 출력하기
  - **`retriever.vectorstore.similarity_search`** 메서드 → `child` + `parent` 문서 `chunk` 내에서 검색을 수행

```
# vectorstore의 유사도 검색 수행하기
relevant_chunks = retriever.vectorstore.similarity_search(
    "삼성전자가 만든 생성형 AI 의 이름은?"
)

# 출력하기
print(f"검색된 문서의 개수: {len(relevant_chunks)}")                    # 0.1s
```

- 검색된 문서의 개수: 4 *(`0.1s`)*

```
for chunk in relevant_chunks:
    print(chunk.page_content, end="\n\n")
    print(">" * 100, end="\n\n")
```

- 셀 출력

```
☞ 출처 : 삼성전자, ‘삼성 AI 포럼’서 자체 개발 생성형 AI ‘삼성 가우스’ 공개, 2023.11.08.
삼성전자, ‘삼성 개발자 콘퍼런스 코리아 2023’ 개최, 2023.11.14.
TechRepublic, Samsung Gauss: Samsung Research Reveals Generative AI, 2023.11.08.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 ‘삼성 AI 포럼 2023’ 행사에서 자체 개발한 생성 AI 모델
‘삼성 가우스’를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에
단계적으로 탑재할 계획
n 삼성 가우스는 △텍스트를 생성하는 언어모델 △코드를 생성하는 코드 모델 △이미지를 생성하는
이미지 모델의 3개 모델로 구성

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

- **`retriever.invoke()`** 메서드 → 쿼리 실행 (원본 문서의 전체 내용 검색함)

```
```

```
relevant_docs = retriever.invoke("삼성전자가 만든 생성형 AI 의 이름은?")

print(f"검색된 문서의 개수: {len(relevant_docs)}", end="\n\n")
print("=" * 100, end="\n\n")
print(relevant_docs[0].page_content)
```

- 셀 출력

```
검색된 문서의 개수: 1


====================================================================================================

SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 ‘삼성 AI 포럼 2023’ 행사에서 자체 개발한 생성 AI 모델
‘삼성 가우스’를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에
단계적으로 탑재할 계획
n 삼성 가우스는 △텍스트를 생성하는 언어모델 △코드를 생성하는 코드 모델 △이미지를 생성하는
이미지 모델의 3개 모델로 구성
•언어 모델은 클라우드와 온디바이스 대상 다양한 모델로 구성되며, 메일 작성, 문서 요약, 번역 업무의
처리를 지원
•코드 모델 기반의 AI 코딩 어시스턴트 ‘코드아이(code.i)’는 대화형 인터페이스로 서비스를 제공하며
사내 소프트웨어 개발에 최적화
•이미지 모델은 창의적인 이미지를 생성하고 기존 이미지를 원하는 대로 바꿀 수 있도록 지원하며
저해상도 이미지의 고해상도 전환도 지원
n IT 전문지 테크리퍼블릭(TechRepublic)은 온디바이스 AI가 주요 기술 트렌드로 부상했다며,
2024년부터 가우스를 탑재한 삼성 스마트폰이 메타의 라마(Llama)2를 탑재한 퀄컴 기기 및 구글
어시스턴트를 적용한 구글 픽셀(Pixel)과 경쟁할 것으로 예상
☞ 출처 : 삼성전자, ‘삼성 AI 포럼’서 자체 개발 생성형 AI ‘삼성 가우스’ 공개, 2023.11.08.
삼성전자, ‘삼성 개발자 콘퍼런스 코리아 2023’ 개최, 2023.11.14.
TechRepublic, Samsung Gauss: Samsung Research Reveals Generative AI, 2023.11.08.
```

- **MMR** 지원
  - `retriever`가 벡터 데이터베이스에서 기본적으로 수행하는 검색 유형: *유사도 검색*
  - `LangChain Vector Store`: **`MMR`** (`Max Marginal Relevance`) 검색도 지원 → **`search_type`** 속성 설정
    - `retriever` 객체의 `search_type` 속성 = **`SearchType.mmr`** 로 설정하기
    - 검색 시 `MMR` (`Maximal Marginal Relevance`) 알고리즘 사용하도록 지정하는 것

```
from langchain.retrievers.multi_vector import SearchType

# 검색 유형을 MMR(Maximal Marginal Relevance)로 설정하기
retriever.search_type = SearchType.mmr

# 관련 문서 전체 검색하기
print(retriever.invoke("삼성전자가 만든 생성형 AI 의 이름은?")[0].page_content)             # 0.1s
```

- `MMR`로 검색해보기 (0.1s)

```
SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
```

n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 '삼성 AI 포럼 2023' 행사에서 자체 개발한 생성 AI 모델
'삼성 가우스'를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에
단계적으로 탑재할 계획
n 삼성 가우스는 △텍스트를 생성하는 언어모델 △코드를 생성하는 코드 모델 △이미지를 생성하는
이미지 모델의 3개 모델로 구성
•언어 모델은 클라우드와 온디바이스 대상 다양한 모델로 구성되며, 메일 작성, 문서 요약, 번역 업무의
처리를 지원
•코드 모델 기반의 AI 코딩 어시스턴트 '코드아이(code.i)'는 대화형 인터페이스로 서비스를 제공하며
사내 소프트웨어 개발에 최적화
•이미지 모델은 창의적인 이미지를 생성하고 기존 이미지를 원하는 대로 바꿀 수 있도록 지원하며
저해상도 이미지의 고해상도 전환도 지원
n IT 전문지 테크리퍼블릭(TechRepublic)은 온디바이스 AI가 주요 기술 트렌드로 부상했다며,
2024년부터 가우스를 탑재한 삼성 스마트폰이 메타의 라마(Llama)2를 탑재한 퀄컴 기기 및 구글
어시스턴트를 적용한 구글 픽셀(Pixel)과 경쟁할 것으로 예상
☞ 출처 : 삼성전자, '삼성 AI 포럼'서 자체 개발 생성형 AI '삼성 가우스' 공개, 2023.11.08.
삼성전자, '삼성 개발자 콘퍼런스 코리아 2023' 개최, 2023.11.14.
TechRepublic, Samsung Gauss: Samsung Research Reveals Generative AI, 2023.11.08.

```
from langchain.retrievers.multi_vector import SearchType

# 검색 유형을 similarity_score_threshold로 설정
retriever.search_type = SearchType.similarity_score_threshold
retriever.search_kwargs = {"score_threshold": 0.3}

# 관련 문서 전체를 검색
print(retriever.invoke("삼성전자가 만든 생성형 AI 의 이름은?")[0].page_content)
```

- `similarity_score_threshold = 0.3`으로 설정

SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI '삼성 가우스' 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 '삼성 가우스'를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 '삼성 AI 포럼 2023' 행사에서 자체 개발한 생성 AI 모델
'삼성 가우스'를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에
단계적으로 탑재할 계획
n 삼성 가우스는 △텍스트를 생성하는 언어모델 △코드를 생성하는 코드 모델 △이미지를 생성하는
이미지 모델의 3개 모델로 구성
•언어 모델은 클라우드와 온디바이스 대상 다양한 모델로 구성되며, 메일 작성, 문서 요약, 번역 업무의
처리를 지원
•코드 모델 기반의 AI 코딩 어시스턴트 '코드아이(code.i)'는 대화형 인터페이스로 서비스를 제공하며
사내 소프트웨어 개발에 최적화
•이미지 모델은 창의적인 이미지를 생성하고 기존 이미지를 원하는 대로 바꿀 수 있도록 지원하며
저해상도 이미지의 고해상도 전환도 지원
n IT 전문지 테크리퍼블릭(TechRepublic)은 온디바이스 AI가 주요 기술 트렌드로 부상했다며,
2024년부터 가우스를 탑재한 삼성 스마트폰이 메타의 라마(Llama)2를 탑재한 퀄컴 기기 및 구글
어시스턴트를 적용한 구글 픽셀(Pixel)과 경쟁할 것으로 예상
☞ 출처 : 삼성전자, '삼성 AI 포럼'서 자체 개발 생성형 AI '삼성 가우스' 공개, 2023.11.08.
삼성전자, '삼성 개발자 콘퍼런스 코리아 2023' 개최, 2023.11.14.
TechRepublic, Samsung Gauss: Samsung Research Reveals Generative AI, 2023.11.08.

```
from langchain.retrievers.multi_vector import SearchType

# 검색 유형을 similarity로 설정, k값을 1로 설정
retriever.search_type = SearchType.similarity
retriever.search_kwargs = {"k": 1}

# 관련 문서 전체 검색하기
print(len(retriever.invoke("삼성전자가 만든 생성형 AI 의 이름은?")))                # 1
```

---

5) 요약본 (summary)을 벡터저장소에 저장

- 요약의 이점
  - 종종 `chunk`의 내용을 `보다 정확하게 추출` 가능 → 더 나은 검색 결과를 얻을 수 있음
  - 요약을 생성하는 방법, 임베딩하는 방법 알아보기

```
# PDF 파일을 로드하고 텍스트를 분할하기 위한 라이브러리 임포트
from langchain_community.document_loaders import PyMuPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter

# PDF 파일 로더 초기화
loader = PyMuPDFLoader("../10_Retriever/data/SPRI_AI_Brief_2023년12월호_F.pdf")

# 텍스트 분할
text_splitter = RecursiveCharacterTextSplitter(chunk_size=600, chunk_overlap=50)

# PDF 파일 로드 및 텍스트 분할 실행
split_docs = loader.load_and_split(text_splitter)

# 분할된 문서의 개수 출력
print(f"분할된 문서의 개수: {len(split_docs)}")                                  # 0.1s
```

- 분할된 문서의 개수: 61

```
from langchain_core.documents import Document
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_ollama import OllamaLLM

llm_lama2 = OllamaLLM(temperature=0, model="exaone3.5:7.8b")

summary_chain = (
    {"doc": lambda x: x.page_content}
    # 문서 요약을 위한 프롬프트 템플릿 생성
    | ChatPromptTemplate.from_messages(
        [
            ("system", "You are an expert in summarizing documents in Korean."),
            (
                "user",
                "Summarize the following documents in 3 sentences in bullet points format.\n\n{doc}",
            ),
        ]
    )
    # Ollama 한국어 모델을 사용하여 요약 생성
    | llm_lama2
    | StrOutputParser()
)
```

- **`chain.batch`** 메서드 사용 → **`docs`** 리스트의 문서들을 일괄 요약

- 교재 내용:
  - **`max_concurrency`** = **`10`** → 최대 10개의 문서를 동시에 처리할 수 있도록 설정

```
# 문서 배치 처리
summaries = summary_chain.batch(split_docs, {"max_concurrency": 10})
# Ollama server에 문서 10개를 동시에(병렬로) 처리해달라는 요청
# m1의 CPU, GPU, RAM 등의 메모리 자원을 동원해 최대한 동시에 처리하려고 함
```

-
  - 30분이 넘도록 완료되지 않음 → 강제 종료

- **배치 처리와 동시성** (`max_concurrency`)
  - **`max_concurrency`** 조절하기 → **`1`** 로 조절해서 진행해보고, **시간 체크해보기** 로 결정

설정	장점	단점	예상 결과 (M1, 7.8B 모델)
10 (현재)	이론상 가장 빠름	과도한 자원 경합으로 인해 오히려 비효율적이고 불안정함.	23분 이상 소요 및 지연 발생 (현재 경험 중)
5 (제안)	10보다는 자원 경합이 덜함	여전히 자원 경합이 발생하여, 1개씩 처리할 때보다 총 시간이 더 길어질 가능성이 높음.	10보다는 빠르겠지만, 최적의 순차 처리보다 느릴 수 있음.
1 (권장)	가장 안정적이고 효율적이며 자원 경합이 없음.	이론상 병렬 처리의 이점을 포기함.	가장 예측 가능한 속도 (예: 20분 내외)로 완료 가능.

```
import time

OPTIMIZED_CONCURRENCY = 1
print(f"Ollama 모델 로드 중: exaone3.5:7.8b (최적 동시성: {OPTIMIZED_CONCURRENCY})")

# 배치 처리 실행 (최적화 적용)
start_time = time.time()
print("\n▶️ 요약 배치 처리를 시작합니다...")

# max_concurrency=1로 설정하여 M1 Mac의 자원 과부하를 막고 효율적인 순차 처리를 유도하기
summaries = summary_chain.batch(split_docs, {"max_concurrency": OPTIMIZED_CONCURRENCY})

end_time = time.time()
elapsed_time = end_time - start_time
```

- 배치 처리 시작 *(`34m 6.1s`)*

```
Ollama 모델 로드 중: exaone3.5:7.8b (최적 동시성: 1)

▶️ 요약 배치 처리를 시작합니다...
```

```
# 결과 출력
print("\n--- 결과 ---")
print(f"✅ 요약 작업이 완료되었습니다!")
print(f"요약된 문서 개수는: {len(summaries)}개입니다.")
print(f"총 소요 시간: {elapsed_time:.2f}초 ({elapsed_time / 60:.2f}분)")
```

- 셀 출력

```
--- 결과 ---
✅ 요약 작업이 완료되었습니다!
요약된 문서 개수는: 61개입니다.
총 소요 시간: 2046.19초 (34.10분)
```

- 요약된 내용 출력 → 결과 확인하기

```
# 원본 문서의 내용 출력하기
print(split_docs[33].page_content, end="\n\n")


# 요약 출력하기
print("[요약]")
print(summaries[33])
```

- 결과 출력해보기

```
SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 ‘삼성 AI 포럼 2023’ 행사에서 자체 개발한 생성 AI 모델
‘삼성 가우스’를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
```

```
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에

[요약]
- **삼성전자**는 **'삼성 가우스'**라는 자체 개발 생성 AI 모델을 공개, 언어, 코드, 이미지 세 가지 분야를 아우르는 온디바이스 작동형 모델로 구성된
- **삼성 가우스**는 2023년 11월 '삼성 AI 포럼 2023'에서 처음 공개되었으며, 가우스의 이름을 따서 다양한 상황에 최적화된 크기의 모델 선택이 가능
- 이 모델은 사용자 정보 유출 위험 없이 안전하게 학습되었으며, 삼성전자는 이를 통해 다양한 제품에 단계적으로 적용할 계획임.
```

- **`Chroma`** 벡터 저장소 초기화 → 자식 청크(`child chunks`) 인덱싱 = 이 때 **`임베딩 함수`** 사용하기
  - 문서 `ID`를 나타내는 키 = **`doc_id`**

```
import uuid

# 요약 정보를 저장할 벡터 저장소 생성하기
summary_vectorstore = Chroma(
    collection_name="summaries",
    embedding_function=multilingual_embeddings,
)

# 부모 문서를 저장할 저장소 생성하기
store = InMemoryStore()

# 문서 ID를 저장할 키 이름 지정하기
id_key = "doc_id"

# 검색기 초기화하기
retriever = MultiVectorRetriever(               # 시작 시 비어 있음
    vectorstore=summary_vectorstore,            # 벡터 저장소
    byte_store=store,                           # 바이트 저장소
    id_key=id_key,                              # 문서 ID 키
)

# 문서 ID 생성하기
doc_ids = [str(uuid.uuid4()) for _ in split_docs]
```

- 요약된 문서, 메타데이터 *(여기에서는 생성한 요약본에 대한 `Document ID`)* 저장

```
summary_docs = [
    # Document 객체 생성하기
    Document(page_content=s, metadata={id_key: doc_ids[i]})         # 요약된 내용=페이지 콘텐츠, 문서 ID=메타데이터
    for i, s in enumerate(summaries)
]
```

- 일치해야 함: **`요약본의 문서의 개수`** = **`원본 문서의 개수`**

```
# 요약본의 문서의 개수
len(summary_docs)                                                   # 61
```

- **`retriever.vectorstore.add_documents(summary_docs)`** → **`summary_docs`** = 벡터 저장소에 추가
- **`retirever.docstore.mset(list(zip(doc_ids, docs)))`** → **`doc_ids`**, **`docs`** 매핑 → 문서 저장소에 저장

```
# 요약된 문서를 벡터 저장소에 추가하기
retriever.vectorstore.add_documents(
    summary_docs
)

# 문서 ID와 문서를 매핑하여 문서 저장소에 저장하기
retriever.docstore.mset(list(zip(doc_ids, split_docs)))             # 3.8s
```

- **`vectorstor`** 객체의 **`similarity_search`** 메서드 → 유사도 검색 수행하기

```
# 유사도 검색 수행하기

result_docs = summary_vectorstore.similarity_search(
    "삼성전자가 만든 생성형 AI 의 이름은?"
)                                                                   # 0.1s
```

```
# 1개의 결과 문서 출력해보기
print(result_docs[0].page_content)
```

- 1개의 결과를 문서로 출력해보기

```
- **삼성전자**는 **'삼성 가우스'**라는 자체 개발 생성 AI 모델을 공개, 언어, 코드, 이미지 세 가지 분야를 아우르는 온디바이스 작동형 모델로 구성된
- **삼성 가우스**는 2023년 11월 '삼성 AI 포럼 2023'에서 처음 공개되었으며, 가우스의 이름을 따서 다양한 상황에 최적화된 크기의 모델 선택이 가능
- 이 모델은 사용자 정보 유출 위험 없이 안전하게 학습되었으며, 삼성전자는 이를 통해 다양한 제품에 단계적으로 적용할 계획임.
```

- `retriever` 객체의 `invoke()` → 질문과 관련된 문서 검색하기

```
# 관련된 문서를 검색하여 가져오기
retrieved_docs = retriever.invoke("삼성전자가 만든 생성형 AI 의 이름은?")
print(retrieved_docs[0].page_content)
```

- 관련된 문서 검색해서 가져오기

```
SPRi AI Brief |
2023-12월호
10
삼성전자, 자체 개발 생성 AI ‘삼성 가우스’ 공개
n 삼성전자가 온디바이스에서 작동 가능하며 언어, 코드, 이미지의 3개 모델로 구성된 자체 개발 생성
AI 모델 ‘삼성 가우스’를 공개
n 삼성전자는 삼성 가우스를 다양한 제품에 단계적으로 탑재할 계획으로, 온디바이스 작동이 가능한
삼성 가우스는 외부로 사용자 정보가 유출될 위험이 없다는 장점을 보유
KEY Contents
£ 언어, 코드, 이미지의 3개 모델로 구성된 삼성 가우스, 온디바이스 작동 지원
n 삼성전자가 2023년 11월 8일 열린 ‘삼성 AI 포럼 2023’ 행사에서 자체 개발한 생성 AI 모델
‘삼성 가우스’를 최초 공개
•정규분포 이론을 정립한 천재 수학자 가우스(Gauss)의 이름을 본뜬 삼성 가우스는 다양한 상황에
최적화된 크기의 모델 선택이 가능
•삼성 가우스는 라이선스나 개인정보를 침해하지 않는 안전한 데이터를 통해 학습되었으며,
온디바이스에서 작동하도록 설계되어 외부로 사용자의 정보가 유출되지 않는 장점을 보유
•삼성전자는 삼성 가우스를 활용한 온디바이스 AI 기술도 소개했으며, 생성 AI 모델을 다양한 제품에
```

---

6) `가설 쿼리 (Hypothetical Queries)`를 활용해 문서 내용 탐색

- `LLM` = `특정 문서에 대해 가정할 수 있는 질문 목록 생성하는데 사용 가능`
  - 생성된 가설 질문 = `embedding` → 문서의 내용을 더욱 깊이 있게 탐색, 이해 가능
    - 문서의 주요 주제와 개념을 파악하는데 도움
    - 독자들이 뭇너 내용에 대해 더 많은 궁금증을 갖도록 유도 가능

- 교재 내용
  - `Function Calling` → 가설 질문 생성하는 예제
  - `ChatPromptTemplate` → 주어진 문서를 기반으로 `3개의 가상 질문을 생성하는 프롬프트 템플릿을 정의`
    - `function`, `function_call` → 가상 질문 생성 함수 호출
    - `JsonKeyOutputFunctionParser` → 생성된 가상 질문 `파싱` → `questions` `key`에 해당하는 값 추출

- `OllamaLLM()`
  - `기본 LLM` (`Legacy LLM`) 클래스
  - 단순한 `text-in`, `text-out` 응답을 처리
- `function` 인수의 역할
  - `funcions`, `funcion_call` 인수
  - = 원래 `OpenAI` 의 `Chat Model`과 같이 함수 호출을 명시적으로 지원하는 모델에 전달되어야 함
  - = 즉, `ChatOpenAI` 모델에 전달되어야 함
- `문제점`
  - `OllamaLLM` = 내부적으로 `ollama.Client().generate()` 메서드 호출 ≠ `functions`, `function_call`
  - 함수 호출과 유사한 `JSON` 출력을 얻기 위해서는 `Ollama` `X` → `ChatOllama` 사용, `with_structued_output` 메서드 사용
  - `Pydantic` 모델 사용: 원하는 출력 형식을 `명시적으로 정의`

```
# 1. 필요한 라이브러리 임포트 및 Pydantic 모델 정의
from pydantic import BaseModel, Field
from langchain_core.prompts import ChatPromptTemplate
from langchain_ollama import ChatOllama                                    # OllamaLLM 대신 ChatOllama 사용
# JsonKeyOutputFunctionsParser는 더 이상 필요하지 않음

# 2. 원하는 출력 구조를 정의하는 Pydantic 모델
class HypotheticalQuestions(BaseModel):
    """Generate hypothetical questions based on the document content."""
    questions: list[str] = Field(
        description="A list of exactly 3 hypothetical questions that the document could answer, written in Korean."
    )

# 3. ChatOllama 모델 초기화
# ChatOllama는 구조화된 출력을 더 잘 지원함
llm_chat = ChatOllama(model="exaone3.5:7.8b", temperature=0)            # max_retries는 ChatOllama의 인수 X → 제거

# 4. 구조화된 출력을 위한 체인 생성
hypothetical_query_chain = (
    {"doc": lambda x: x.page_content}

    # 아래 문서를 사용하여 답변할 수 있는 가상의 질문을 정확히 3개 생성하도록 요청하는 프롬프트
    | ChatPromptTemplate.from_template(
        "Generate a list of exactly 3 hypothetical questions that the below document could be used to answer. "
        "Potential users are those interested in the AI industry. Create questions that they would be interested in. "
        "Output should be written in Korean:\n\n{doc}"
    )

    # 5. with_structured_output을 사용하여 출력 형식을 Pydantic 모델로 강제하기
    # 'functions' 및 'JsonKeyOutputFunctionsParser' 대체
    | llm_chat.with_structured_output(HypotheticalQuestions)
)
```

- 문서에 대한 답변 출력하기
  - 출력 = 생성한 `3개의 가설 쿼리(Hypothetical Queries)`

```
# 주어진 문서에 대해 체인 실행하기
try:
    # split_docs[33]이 Document 객체이므로 .page_content 접근 대신 통째로 invoke에 전달하여
    # {"doc": ...} 형식으로 매핑되도록 합니다.
    result = hypothetical_query_chain.invoke(split_docs[33])
    print("✅ 가상 질문 생성 성공:")
    print(result.questions)

except Exception as e:
    print(f"❌ 오류 발생: {e}")
    print("Ollama 서버가 실행 중인지 확인하고, 모델을 다운로드했는지 확인하세요.")
```

- 생성한 3개의 가설 쿼리 출력하기 *`(35.6s)`*

```
✅ 가상 질문 생성 성공:
["삼성전자의 '삼성 가우스'가 온디바이스 환경에서 어떻게 사용자 데이터 보안을 강화하고 있는지 구체적으로 설명해 주실 수 있나요?", '삼성 가우스의 세
```

- **`chain.batch()`** → **`split_docs`** 데이터에 대해서 동시에 여러 개의 요청 처리하기

```
# 문서 목록에 대한 가설 질문에 대한 배치 생성해보기

import time

OPTIMIZED_CONCURRENCY = 1
print(f"Ollama 모델 로드 중: exaone3.5:7.8b (최적 동시성: {OPTIMIZED_CONCURRENCY})")

# 배치 처리 실행 (최적화 적용)
start_time = time.time()
print("\n▶️ 요약 배치 처리를 시작합니다...")

# max_concurrency=1로 설정하여 M1 Mac의 자원 과부하를 막고 효율적인 순차 처리를 유도하기
hypothetical_questions = hypothetical_query_chain.batch(
    split_docs, {"max_concurrency": 10}
)

end_time = time.time()
elapsed_time = end_time - start_time

# 결과 출력
print("\n--- 결과 ---")
print(f"✅ 요약 작업이 완료되었습니다!")
```

```
print(f"요약된 문서 개수는: {len(hypothetical_questions)}개입니다.")
print(f"총 소요 시간: {elapsed_time:.2f}초 ({elapsed_time / 60:.2f}분)")
```

- 가설 쿼리 벡터 저장소 저장하기 *(30m 0.6s)*

```
Ollama 모델 로드 중: exaone3.5:7.8b (최적 동시성: 1)

▶️ 요약 배치 처리를 시작합니다...

--- 결과 ---
✅ 요약 작업이 완료되었습니다!
요약된 문서 개수는: 61개입니다.
총 소요 시간: 1800.63초 (30.01분)
```

```
hypothetical_questions[33]
```

- 다른 문서의 3개의 가설 쿼리 출력해보기

```
HypotheticalQuestions(questions=["삼성전자의 '삼성 가우스'가 온디바이스 환경에서 어떻게 사용자 데이터 보안을 강화하고 있는지 구체적으로 설
```

- 생성한 가설 쿼리 = 벡터 저장소에 저장하는 과정
  - *이전에 진행했던 방식과 동일*

```
# 자식 청크를 인덱싱하는 데 사용할 벡터 저장소
hypothetical_vectorstore = Chroma(
    collection_name="hypo-questions", embedding_function=multilingual_embeddings
)

# 부모 문서의 저장소 계층
store = InMemoryStore()

id_key = "doc_id"

# 검색기 (시작 시 비어 있음)
retriever = MultiVectorRetriever(
    vectorstore=hypothetical_vectorstore,
    byte_store=store,
    id_key=id_key,
)

doc_ids = [str(uuid.uuid4()) for _ in split_docs]                    # 문서 ID 생성
```

- **`question_docs`** 리스트 = `메타데이터 (문서 ID)` 추가

```
question_docs = []

# hypothetical_questions 저장
for i, question_list in enumerate(hypothetical_questions):

    question_docs.extend(
        # 질문 리스트의 각 질문에 대해 Document 객체를 생성하기
        [Document(page_content=s, metadata={id_key: doc_ids[i]}) for s in question_list.questions]
        # 메타데이터 = 해당 질문의 문서 ID 포함
    )
```

- 가설 쿼리 = 문서에 추가
- 원본 문서 = **`docstore`** 에 추가

```
# hypothetical_questions 문서를 벡터 저장소에 추가하기
retriever.vectorstore.add_documents(question_docs)

# 문서 ID와 문서를 매핑하여 문서 저장소에 저장하기
retriever.docstore.mset(list(zip(doc_ids, split_docs)))                    # 4.3s
```

- **`vectorstore`** 객체의 **`similarity_search`** 메서드 → 유사도 검색 수행

```

```

```
# 유사한 문서를 벡터 저장소에서 검색하기
result_docs = hypothetical_vectorstore.similarity_search(
    "삼성전자가 만든 생성형 AI 의 이름은?"
```

- 유사도 검색 결과 = 생성한 가설 쿼리만 추가해둔 상태 → 생성한 가설 쿼리 중 유사도가 가장 높은 문서를 반환함

```
# 유사도 검색 결과 출력하기

for doc in result_docs:
    print(doc.page_content)
    print(doc.metadata)
```

- 유사도 높은 문서 반환하기

```
삼성전자가 개최한 '삼성 개발자 콘퍼런스 코리아 2023'에서 공개된 삼성의 생성형 AI 기술, 특히 Samsung Gauss에 대한 주요 업데이트 내용은 무엇인가
{'doc_id': 'eeff618f-d0ea-4434-9bce-7512bcb8da7a'}
Samsung Gauss와 같은 삼성의 생성형 AI 기술이 미래의 AI 산업 트렌드에 어떤 영향을 미칠 것으로 예상되나요?
{'doc_id': 'eeff618f-d0ea-4434-9bce-7512bcb8da7a'}
삼성 가우스의 다목적 AI 모델 구성은 어떻게 기존 스마트폰 AI 기능을 혁신적으로 발전시킬 수 있을까요?

삼성 가우스의 온디바이스 AI 기술이 2024년 이후 스마트폰 시장에서 메타의 라마2나 구글 어시스턴트와 비교해 어떤 차별점을 제공할 것으로 예상됩니까?

코드아이(code.i)와 같은 대화형 AI 코딩 어시스턴트 기능이 기업의 소프트웨어 개발 효율성에 어떤 구체적인 영향을 미칠 수 있을까요?
{'doc_id': 'bcc6be37-dd5b-427a-9ba3-f61bc187dc61'}
삼성전자의 개발자 콘퍼런스에서 다룬 AI 기술 발전 동향을 바탕으로, 기업들이 AI 혁신을 위해 어떤 전략을 수립해야 할까요?
{'doc_id': 'eeff618f-d0ea-4434-9bce-7512bcb8da7a'}
```

- **`retriever`** 객체의 **`invoke()`** 메서드 → 뭐리와 관련된 문서 검색하기

```
# 관련된 문서를 검색하여 가져오기
retrieved_docs = retriever.invoke(result_docs[1].page_content)


# 검색된 문서 출력하기
for doc in retrieved_docs:
    print(doc.page_content)
```

- 쿼리와 관련된 문석 검색하기 *(`0.1s`)*

```
삼성전자, ‘삼성 개발자 콘퍼런스 코리아 2023’ 개최, 2023.11.14.
TechRepublic, Samsung Gauss: Samsung Research Reveals Generative AI, 2023.11.08.
단계적으로 탑재할 계획
n 삼성 가우스는 △텍스트를 생성하는 언어모델 △코드를 생성하는 코드 모델 △이미지를 생성하는
이미지 모델의 3개 모델로 구성
•언어 모델은 클라우드와 온디바이스 대상 다양한 모델로 구성되며, 메일 작성, 문서 요약, 번역 업무의
처리를 지원
•코드 모델 기반의 AI 코딩 어시스턴트 ‘코드아이(code.i)’는 대화형 인터페이스로 서비스를 제공하며
사내 소프트웨어 개발에 최적화
•이미지 모델은 창의적인 이미지를 생성하고 기존 이미지를 원하는 대로 바꿀 수 있도록 지원하며
저해상도 이미지의 고해상도 전환도 지원
n IT 전문지 테크리퍼블릭(TechRepublic)은 온디바이스 AI가 주요 기술 트렌드로 부상했다며,
2024년부터 가우스를 탑재한 삼성 스마트폰이 메타의 라마(Llama)2를 탑재한 퀄컴 기기 및 구글
어시스턴트를 적용한 구글 픽셀(Pixel)과 경쟁할 것으로 예상
☞ 출처 : 삼성전자, ‘삼성 AI 포럼’서 자체 개발 생성형 AI ‘삼성 가우스’ 공개, 2023.11.08.
삼성전자, ‘삼성 개발자 콘퍼런스 코리아 2023’ 개최, 2023.11.14.
```

- next: ***셀프 쿼리 검색기 (SelfQueryRetriever)***
