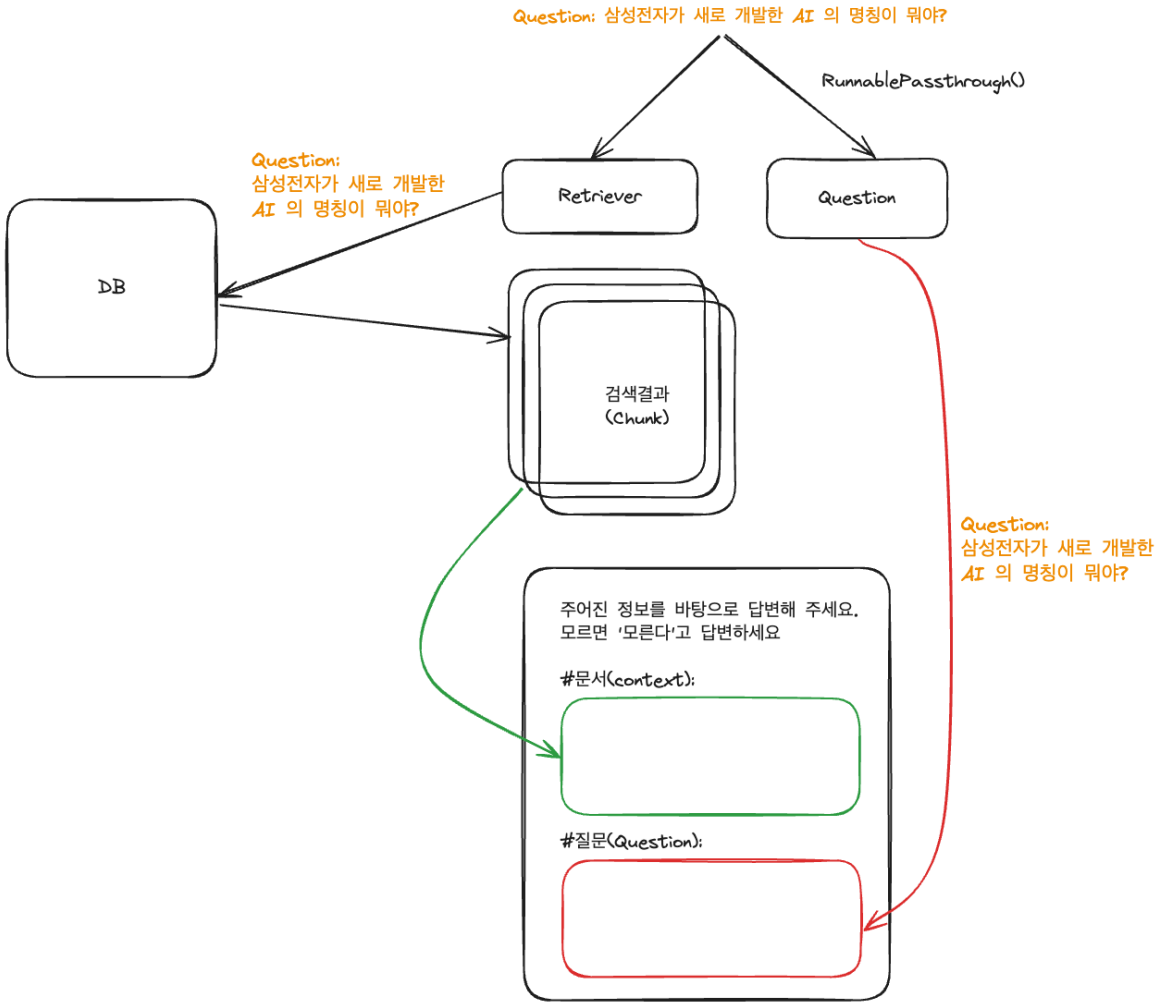
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

`CH14.` 체인 *(Chains)*

- **`Chain` 생성 단계**
  - 이전의 7단계 과정을 모두 하나로 묶어 하나의 **`RAG` 파이프라인** 으로 조립 → 완성하는 단계
  - 문서 기반 `RAG` 구조도



- **`코드`**
  - **`LCEL`** *(LangChain Expression Language)* 문법 → 이전의 7단계의 전 과정을 하나의 `Chain`으로 묶음

```
# 체인(Chain) 생성
chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
```

-
  - 완성된 `Chain`에 질의하기

```
# 체인 실행(Run Chain)
# 문서에 대한 질의를 입력 → 답변 출력하기
question = "삼성전자가 자체 개발한 AI 의 이름은?"

response = chain.invoke(question)
```

- **참고**
  - ***LCEL문법***
  - ***LangChain – Chains***

---

1. 문서 요약

1) 요약 *(Summarization)*

- 주요 개요
  - **Stuff**: **전체** 문서 **한 번에** 요약
  - **Map-Reduce**: **분할 요약** 후 ***일괄 병합***
  - **Map-Refine**: **분할 요약** 후 ***점진적인 병합***
  - **Chain of Density**: **N번** **반복** **실행** 하며, **누락**된 **entity**를 **보완**하며 **요약 개선**
  - **Clustering-Map-Refine**: 문서의 **Chunk**를 **N 개**의 **클러스터**로 나누고, **각 클러스터**에서 **중심점**에 **가까운 문서에** 대한 **요약**을 **Refine 요약**

- **대표적으로 알려진 요약 방식**
  - *요약기(`retriever`) 구축 시 중심적인 질문 =* ***`LLM`의 컨텍스트 창에 어떻게 전달할 것인가***
  - **a.** **`stuff`**
    - **단순히 모든 문서를 단일 프롬프트로 넣는 방식**
    - 가장 간단한 접근 방식
  - **b.** **`Map-reduce`**
    - 각 문서를 **`map`** 단계에서 **개별적으로 요약**
    - **`reduce`** 단계: 요약본들을 **최종 요약본** 으로 **합치는 방식**
  - **c.** **`Refine`**
    - 입력 문서를 **순회**하며 **반복적**으로 **답변**을 **업데이트**하여 **응답**을 **구성**함
    - 각 문서에 대해 **모든 비문서 입력**, **현재 문서**, **최신 중간 답변**을 **`chain`**에 전달 → **새로운 답변을 얻음**

- **환경설정**

```
# 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.
```

---

2) `Stuff`

- **`stuff documents chain`**
  - *`stuff`* = *채우다* or *채우기 위해*
  - 문서 chain 중 가장 간단한 방식
  - 문서 목록을 가져와서 **`모두 프롬프트에 삽입`** → 그 **`프롬프트`**를 **`LLM`**에 전달
  - 적합한 경우
    - 문서가 작은 경우
    - 대부분의 호출에 몇 개만 전달되는 애플리케이션에 적합

- **`데이터 로드하기`**

```
from langchain_community.document_loaders import TextLoader

# 뉴스데이터 로드
loader = TextLoader("../14_Chains/data/news.txt")
docs = loader.load()
print(f"총 글자수: {len(docs[0].page_content)}")
print("\n========== 앞부분 미리보기 =========\n")
print(docs[0].page_content[:500])
```

- 데이터 로드 - (`0.2s`)

```
총 글자수: 1708

========== 앞부분 미리보기 =========

제목:
AI2, 상업 활용까지 자유로운 '진짜' 오픈 소스 LLM '올모' 출시

내용:
앨런AI연구소(AI2)가 완전한 오픈 소스 대형언어모델(LLM) '올모(OLMo)'를 출시했다. 데이터 수집, 학습, 배포의 전 과정을 투명하게 공개한 데다 상업적
벤처비트는 1일(현지시간) 비영리 민간 AI 연구기관인 AI2가 '최초의 진정한 오픈 소스 LLM 및 프레임워크'라고 소개한 '올모'를 출시했다고 보도했다.
이에 따르면 올모는 모델 코드와 모델 가중치뿐만 아니라 훈련 코드, 훈련 데이터, 관련 툴킷 및 평가 툴킷도 제공한다. 이를 통해 모델이 어떻게 구축되었는
올모 프레임워크는 70억 매개변수의 '올모 7B' 등 4가지 변형 모델과 10억 매개변수의 '올모 1B' 모델을 제공한다. 모델들은 훈련 데이터를 생성하는 코드
```

- **한국어로 요약하는 작성 요청 프롬프트**

```
from langchain import hub

prompt = hub.pull("teddynote/summary-stuff-documents-korean")
prompt.pretty_print()
```

- 한국어 요약 요청 프롬프트 - (`3.2s`)

```
Please summarize the sentence according to the following REQUEST.
REQUEST:
1. Summarize the main points in bullet points in KOREAN.
2. Each summarized sentence must start with an emoji that fits the meaning of the each sentence.
3. Use various emojis to make the summary more interesting.
4. Translate the summary into KOREAN if it is written in ENGLISH.
5. DO NOT translate any technical terms.
6. DO NOT include any unnecessary information.

CONTEXT:
{context}

SUMMARY:"
```

```
import os
from dotenv import load_dotenv
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.callbacks import StreamingStdOutCallbackHandler

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

# LLM 초기화
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
    max_output_tokens=4096,
    streaming=True,
    callbacks=[StreamingStdOutCallbackHandler()],           # callback 설정
)
```

- `gemini-2.5-flash-lite` 모델 설정하기

```
E0000 00:00:1760073751.012350 2922705 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

```
# chain 설정
stuff_chain = create_stuff_documents_chain(gemini_lc, prompt)

# 질문 - 응답
answer = stuff_chain.invoke({"context": docs})
```

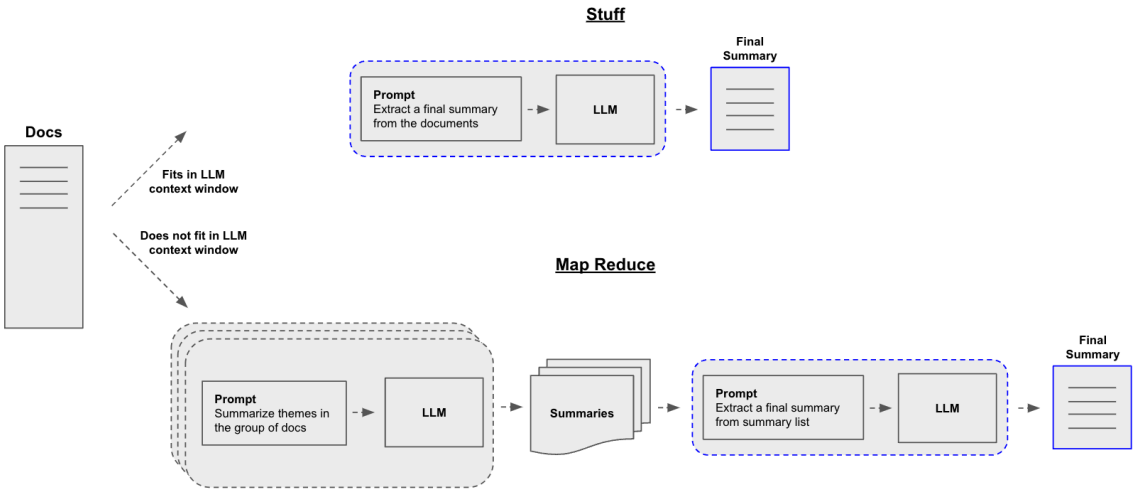
```
print(answer)
```

- **`Stuff_chain`** **응답** - (`2.3s`)

```
다음은 요청하신 내용에 따라 요약한 내용입니다.

*   ✨ 앨런AI연구소(AI2)가 완전한 오픈 소스 LLM '올모(OLMo)'를 출시했습니다.
*   🔓 데이터 수집부터 학습, 배포까지 전 과정을 투명하게 공개하고 상업적 사용까지 허용했습니다.
*   🔬 모델 코드, 가중치, 훈련 코드, 훈련 데이터, 관련 툴킷 및 평가 툴킷을 모두 제공합니다.
*   🚀 '올모 7B' 등 다양한 변형 모델과 '돌마(Dolma)' 데이터 세트를 기반으로 구축되었습니다.
*   💰 아파치 2.0 라이선스로 상업적 활용에 제한이 없습니다.
*   💡 연구자들이 LLM의 작동 원리를 과학적으로 이해하고 신뢰할 수 있는 AI 시스템을 구축하도록 지원합니다.
*   🏆 성능 면에서도 상업용 모델과 동등하거나 우수한 결과를 보여줍니다.
*   🛠️ 비영어권 언어 및 코드 생성 기능에는 일부 제약이 있습니다.
*   🌐 모든 리소스는 깃허브 및 허깅페이스에서 무료로 사용할 수 있습니다.
```

3) Map-Reduce

- **긴 문서**를 **효율적**으로 **요약**하는 기법
- **구성**
  - Map 단계: 각 chunk를 병렬로 요약
  - reduce 단계: 이 요약을 하나의 최종 요약으로 통합



  -
  - 출처
- 적합한 경우
  - **대규모 문서를 처리할 때**
  - 언어 모델의 토큰 제한을 우회해야 할 때

- **데이터 로드하기**

```
# 데이터 로드하기
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader("../14_Chains/data/SPRI_AI_Brief_2023년12월호_F.pdf")
docs = loader.load()
docs = docs[3:8]                          # 여기서 문서의 일부만 요약
print(f"총 페이지수: {len(docs)}")
```

- 총 페이지수: 5 - (1.8s)

① Map

- **각 chunk에 대한 요약을 생성하기**
  - *변경: 핵심 내용 추출로 변경하여 진행 가능*

```
from langchain import hub
from langchain_core.output_parsers import StrOutputParser

# LLM = 앞의 gemini 사용

# map prompt 다운로드
map_prompt = hub.pull("teddynote/map-prompt")

# 프롬프트 출력
map_prompt.pretty_print()
```

- **Map-prompt** - (0.2s)

```
============================== **System Message** ==============================

You are a professional main thesis extractor.
```

```
================================ **Human Message** ================================

Your task is to extract main thesis from given documents. Answer should be in same language as given document.

#Format:
- thesis 1
- thesis 2
- thesis 3
- ...

Here is a given document:
*{doc}*

Write 1~5 sentences.
#Answer:
```

- **`map_chain` 생성하기**

```
# map chain 생성
map_chain = map_prompt | gemini_lc | StrOutputParser()
```

- **`batch()` 호출 → 각 문서에 대한 요약본 생성하기**

```
# 문서에 대한 주요내용 추출
doc_summaries = map_chain.batch(docs)
```

```
print(type(doc_summaries))              # <class 'list'>
```

```
# 요약된 문서의 수 출력
len(doc_summaries)                      # 5
```

```
# 일부 문서의 요약 출력
print(doc_summaries[0])
```

- **`map_chain.batch()`** - (`2.1s`)

```
- 미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령을 발표했다.
- 이 행정명령은 AI의 안전 및 보안 기준 마련, 개인정보보호, 형평성 및 시민권 향상, 소비자 보호, 노동자 지원, 혁신 및 경쟁 촉진, 국제협력을 포함한
- 특히, 강력한 AI 시스템 개발 기업에게 안전 테스트 결과와 시스템 정보를 정부와 공유하도록 요구하며, AI 생성 콘텐츠 표시 표준을 확립한다.
- 또한, 형사사법 시스템, 주택, 보건 분야에서 AI의 무책임한 사용으로 인한 차별과 편견을 방지하기 위한 조치를 확대한다.
- AI 연구 촉진을 위한 국가AI연구자원(NAIRR)을 통해 연구 인프라를 확충하고, 외국인 전문가의 미국 내 취업 및 학업을 지원하기 위해 비자 절차를 간소
```

② Reduce

- **`map` 단계에서 진행한 `핵심 내용`들을 `하나의 최종 요약`으로 `통합`**

```
# reduce prompt 다운로드
reduce_prompt = hub.pull("teddynote/reduce-prompt")

# 프롬프트 출력
reduce_prompt.pretty_print()
```

- **`reduce_prompt`** - (`0.3s`)

```
================================ **System Message** ================================

You are a professional summarizer. You are given a list of summaries of documents and you are asked to create a s

================================ **Human Message** ================================

#Instructions:
1. Extract main points from a list of summaries of documents
2. Make final summaries in bullet points format.
```

```
3. Answer should be written in *{language}*.

#Format:
- summary 1
- summary 2
- summary 3
- ...

Here is a list of summaries of documents:
*{doc_summaries}*

#SUMMARY:
```

- **`Reduce Chain`** 생성하기

```
# reduce chain 생성
reduce_chain = reduce_prompt | gemini_lc | StrOutputParser()
```

- **`Reduce Chain`**으로 **`스트리밍 출력`**해보기

```
reduce_chain.stream({"doc_summaries": doc_summaries, "language": "Korean"})
# <generator object RunnableSequence.stream at 0x119ab3790>
```

```
import os
import time

print(f"🚀 Gemini 스트리밍 출력 시작")
print("-" * 50)

# .stream()을 호출하고, 반환되는 조각(chunk)을 for 루프를 돌면서 직접 출력하기
try:
    start_time = time.time()

    # .stream() 호출
    stream_generator = reduce_chain.stream({"doc_summaries": doc_summaries, "language": "Korean"})

    for chunk in stream_generator:
        # LLM 응답 조각(chunk)의 content 속성에 실제 텍스트 토큰이 들어 있음
        if hasattr(chunk, 'content') and chunk.content is not None:
            print(chunk.content, end="", flush=True)

    end_time = time.time()
    print("\n" + "-" * 50)
    print(f"✅ 스트리밍 완료. 총 소요 시간: {end_time - start_time:.2f}초")

except Exception as e:
    print(f"\n❌ 실행 중 오류가 발생했습니다: {type(e).__name__} - {e}")
```

- **`Reduce_chain`** **스트리밍 출력해보기** - (`1.8s`)

```
🚀 Gemini 스트리밍 출력 시작
--------------------------------------------------
- 미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령을 발표했으며, AI 안전 기준 마련, 개인정보보호, 형평성 향상, 소비자
- G7은 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했으며, 이는 AI 수명주기 전반에 걸친 위험 평가 및 완
- 28개국은 AI 안전 보장을 위한 협력 방안을 담은 블레츨리 선언을 발표했으며, 첨단 AI 개발 기업은 AI 시스템에 대한 안전 테스트 계획에 합의했습니다
- 미국 법원은 예술가들이 제기한 생성 AI 기업 대상 저작권 침해 소송을 기각했으며, 주요 이유는 저작권 미등록과 AI 생성 이미지와 특정 작품 간 유사성
- 미국 연방거래위원회(FTC)는 생성형 AI가 소비자 보호와 공정한 경쟁을 저해할 수 있다는 우려를 제기하며, 소비자 피해 및 빅테크의 시장 지배력 강화 가
--------------------------------------------------
✅ 스트리밍 완료. 총 소요 시간: 1.89초
```

③ `Chain 데코레이터`

```
from langchain_core.runnables import chain
import os
import time
from dotenv import load_dotenv
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.callbacks import StreamingStdOutCallbackHandler
```

```
@chain
def map_reduce_chain(docs):
    # API 키 확인
    if not os.getenv("GOOGLE_API_KEY"):
        os.environ["GOOGLE_API_KEY"] = input("Enter your Google API key: ")

    gemini_lc = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash-lite",
        temperature=0,                                          # temperature = 0으로 설정
        max_output_tokens=4096,
        streaming=True,
        callbacks=[StreamingStdOutCallbackHandler()],           # callback 설정
    )

    # map prompt 다운로드
    map_prompt = hub.pull("teddynote/map-prompt")

    # map chain 생성
    map_chain = map_prompt | gemini_lc | StrOutputParser()

    # 첫 번째 프롬프트, gemini, 문자열 출력 파서를 연결 → 체인 생성
    doc_summaries = map_chain.batch(docs)

    # reduce prompt 다운로드
    reduce_prompt = hub.pull("teddynote/reduce-prompt")

    reduce_llm = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash-lite",
        temperature=0,                                          # temperature = 0으로 설정
        max_output_tokens=4096,
        streaming=True,
        callbacks=[StreamingStdOutCallbackHandler()],           # callback 설정
    )

    reduce_chain = reduce_prompt | reduce_llm | StrOutputParser()

    return reduce_chain.invoke({"doc_summaries": doc_summaries, "language": "Korean"})
```

- **결과 출력해보기**

```
answer = map_reduce_chain.invoke(docs)
```

```
print(answer)
```

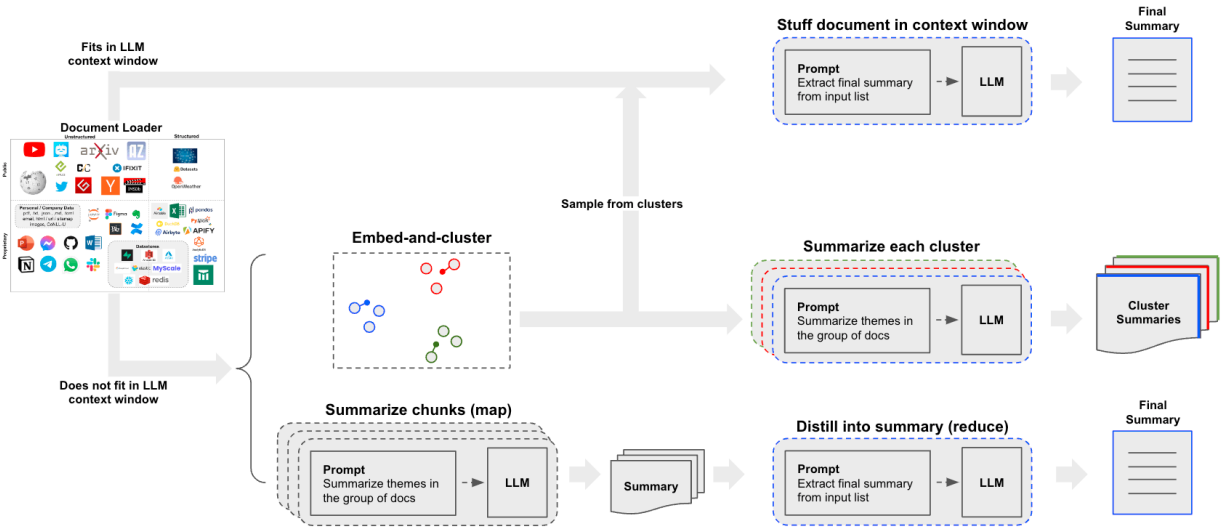
- `map_reduce_chain.invoke()` - (`4.9s`)

```
- 미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령을 발표했으며, AI 안전 기준 마련, 개인정보보호, 형평성 향상, 소비자
- G7은 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했으며, 이는 AI 수명주기 전반에 걸친 위험 평가 및 완
- 28개국은 AI 안전 보장을 위한 협력 방안을 담은 블레츨리 선언을 발표했으며, 첨단 AI 개발 기업은 AI 시스템에 대한 안전 테스트 계획에 합의했습니다
- 미국 법원은 예술가들이 제기한 생성 AI 기업 대상 저작권 침해 소송을 기각했으며, 주요 이유는 저작권 미등록과 AI 생성 이미지와 특정 작품 간 유사성
- 미국 연방거래위원회(FTC)는 생성형 AI가 소비자 보호와 공정한 경쟁을 저해할 수 있다는 우려를 제기하며, 소비자 피해 및 빅테크의 시장 지배력 강화 기
```

---

4) `Map-Refine`

- *`map-reduce`와 유사하지만 차이점이 있음*
- 작동 단계
  - **a.** **`Map`** **단계**: 문서 → **여러 작은** **`chunk`** **로 나눔** → **각** **`chunk`** **에 대한** **`개별적 요약 생성`**
  - **b.** **`Refine`** **단계**
    - **생성된 요약들을** **`순차적`** **으로 처리하며** **`최종 요약`** **을** **`점진적`** **으로** **`개선`**
    - **각 단계에서** **`이전 요약`** **과** **`새로운 chunk`** **의** **`정보`** **를** **`결합`** **하여** **`요약`** **을** **`갱신`**
  - **c.** **`반복 과정`**: 모든 `chunk`가 처리될 때까지 `refine` 단계 반복
  - **d.** **`최종 요약`**: **마지막** **`chunk`** **까지 처리한 후 얻은 요약 =** **`최종 결과`**



- [출처]

- **장점** : 문서의 **순서**를 **유지**하면서 **점진적**으로 **요약**을 **개선**할 수 있음
- **적합한 경우** : **문서의 맥락이 중요한 경우**
- **단점** : `map-reduce`에 비해 **순차적으로 처리되기 때문에 병렬화가 어려움** → 문서 처리 시 **시간**이 **더 오래 걸림**

① Map

- **`map` 단계: 각 `chunk`에 대한 요약을 생성하기**

```
from langchain import hub
from langchain_core.output_parsers import StrOutputParser
import os
from dotenv import load_dotenv
from langchain_google_genai import ChatGoogleGenerativeAI

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

# map llm 생성
map_llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
)
```

- `gemini-2.5-flash-lite` = `map_llm`

```
E0000 00:00:1760078793.706357 2922705 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

```
# map chain 생성
map_summary = hub.pull("teddynote/map-summary-prompt")

# 프롬프트 출력
map_summary.pretty_print()
```

- `map_summary 프롬프트 출력해보기` - (`0.2s`)

```
================================ **System Message** ================================

You are an expert summarizer. Your task is to summarize the following document in *{language}*.

================================ **Human Message** =================================

Extract most important main thesis from the documents, then summarize in bullet points.

#Format:
- summary 1
```

```
- summary 2
- summary 3
-...

Here is a given document:
*{documents}*

Write 1~5 sentences. Think step by step.
#Summary:
```

- **`map_chain` 생성하기**

```
# map chain 생성
map_chain = map_summary | map_llm | StrOutputParser()
```

- **첫 번째 문서에 대한 요약본 출력해보기**

```
# 첫 번째 문서의 요약 출력
print(map_chain.invoke({"documents": docs[0], "language": "Korean"}))
```

- **`map_chain.invoke({"documents": docs[0], "language": "Korean"})`** - (`1.6s`)

```
*   미국 바이든 대통령이 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다.
*   행정명령은 AI 안전 및 보안 기준 마련, 개인정보보호, 형평성 및 시민권 향상, 소비자 보호, 노동자 지원, 혁신 및 경쟁 촉진, 국제협력을 포함합니다
*   특히, 고성능 AI 시스템 개발 기업에게 안전 테스트 결과 및 시스템 정보를 정부와 공유하도록 요구하며, AI 생성 콘텐츠 표시 표준 마련을 추진합니다
*   또한, 법률, 주택, 보건 분야에서 AI로 인한 차별 및 편견 방지 조치를 확대하고, 형사사법 시스템에서의 AI 사용 모범 사례를 개발합니다.
*   AI 연구 인프라 확충, 중소기업 지원, 외국인 전문가 유치를 위한 비자 절차 간소화 등 혁신과 경쟁 촉진 방안도 포함하고 있습니다.
```

```
# 모든 문서를 입력으로 정의하기

input_doc = [{"documents": doc, "language": "Korean"} for doc in docs]
```

- 문서 `type`, `length` 확인해보기

```
# 개별 문서 문서 요약
print(type(docs))                   # <class 'list'>
len(docs)                           # 5

# 모든 문서를 입력으로 정의하기
input_doc = [{"documents": doc, "language": "Korean"} for doc in docs]
print(type(input_doc))              # <class 'list'>
len(input_doc)                      # 5
```

```
# 모든 문서에 대한 요약본 출력해보기

print(map_chain.batch(input_doc))
```

- **`map_chain.batch(input_doc)`** 결과 확인해보기 - (`2.3s`)

```
['*   미국 바이든 대통령이 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다.\n*   행정명령은 AI 안전 및 보안 기준 마련,
```

② Refine

- **`Refine` 단계: 이전의 `map` 단계에서 생성한 `chunk` 들을 `순차적` 으로 `처리` → `최종 요약` 을 `점진적` 으로 `개선`**

```
# refine prompt 다운로드
refine_prompt = hub.pull("teddynote/refine-prompt")
```

```
# 프롬프트 출력
refine_prompt.pretty_print()
```

- **`refine_prompt`** 출력해보기 - (`0.3s`)

```
================================ **System Message** ================================

You are an expert summarizer.

================================ **Human Message** =================================

Your job is to produce a final summary

We have provided an existing summary up to a certain point:
*{previous_summary}*

We have the opportunity to refine the existing summary(only if needed) with some more context below.
------------
*{current_summary}*
------------
Given the new context, refine the original summary in *{language}*.
If the context isn't useful, return the original summary.
```

```
from langchain_core.callbacks import StreamingStdOutCallbackHandler

# refine llm 생성
refine_llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
)
```

- **`refine_llm`** = `gemini-2.5-flash-lite` 생성하기

```
E0000 00:00:1760079862.796527 2922705 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

```
# refine chain 생성
refine_chain = refine_prompt | refine_llm | StrOutputParser()
```

③ **`map_reduce_chain`** 생성하기

- 지금까지의 과정을 하나의 `chain`으로 엮기

```
from langchain_core.runnables import chain
from langchain_core.callbacks import StreamingStdOutCallbackHandler
from langchain_core.output_parsers import StrOutputParser                       # 임포트는 유지
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain import hub                                                       # hub 임포트 추가

@chain
def map_refine_chain(docs):
    """
    Map-Refine 로직을 실행하고 결과를 실시간 스트리밍하며 반환하는 함수.

    입력: Document 리스트 (docs). (Invoke 시 docs를 바로 받음)
    출력: 최종 요약 텍스트
    """

    # 입력 검증
    if not isinstance(docs, list) or not docs:
        raise ValueError("입력은 Document 객체 리스트여야 하며 비어 있으면 안 됩니다.")

    ##### 1. 초기 요약 (Map 단계) #####
    # map 프롬프트 다운로드
    map_summary = hub.pull("teddynote/map-summary-prompt")

    # map llm 생성
    map_llm = ChatGoogleGenerativeAI(
        model="gemini-2.5-flash-lite",
        temperature=0,                                                  # temperature = 0으로 설정
```

```
    max_output_tokens=4096,
)

# map chain 생성
# 🚨 [오류 수정 #1] StrOutputParser() 제거
# 스트리밍 시 StrOutputParser가 'chunk'를 str로 변환하여, 뒤에서 chunk.content 접근 시 AttributeError가 발생했기 때문
map_chain = (
    map_summary
    | map_llm
    # | StrOutputParser()                                    # <-- 제거! 스트리밍 루프가 직접 content를 추출
)

# map chain 구성 및 실행
# docs[0] = Document 객체이므로 리스트로 감싸서 전달
map_generator = map_chain.stream({"documents": [docs[0]], "language": "Korean"})

# 첫 번째 문서부터 스트리밍으로 초기 요약 진행
previous_summary = ""

for chunk in map_generator:
    # chunk = AIMessageChunk 객체 → .content 속성을 갖고 있음
    if chunk and chunk.content is not None:
        print(chunk.content, end="", flush=True)
        previous_summary += chunk.content

# 2. 순차적 정제 (Refine)
# refine 프롬프트 다운로드
refine_prompt = hub.pull("teddynote/refine-prompt")

# refine llm 생성하기
refine_llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
    # callbacks, streaming 설정은 루프에서 직접 처리하므로 제거
)

# refine chain 생성하기
# Refine Chain은 StrOutputParser()를 포함 (아래 루프에서 str 객체를 받아 처리해야 함)
refine_chain = refine_prompt | refine_llm | StrOutputParser()


# 🚨 [오류 수정 #2] Refine 루프의 입력 데이터 구조 변경
# docs[1:]를 사용하여 리스트 형태로 나머지 문서를 가져와야 함
remaining_docs = docs[1:]

for i, current_doc in enumerate(remaining_docs):
    print(f"\n\n-----------------\n[단계 {i+2}/{len(remaining_docs)+1}] 정제 시작...")

    # Refine 체인을 스트리밍으로 호출
    stream_generator = refine_chain.stream({
        "previous_summary": previous_summary,
        "current_summary": [current_doc],
        "language": "Korean"
    })

    # 🚨 [핵심] 스트리밍 출력 및 다음 루프를 위한 결과 누적
    current_chunk = ""
    for chunk in stream_generator:
        # refine_chain에 StrOutputParser()가 붙어 있으므로, chunk는 순수한 str 객체로 반환됨
        # .content에 접근하지 않고 chunk 자체를 출력하기
        if isinstance(chunk, str):
            print(chunk, end="", flush=True)                # 실시간으로 출력 (중복 없음)
            current_chunk += chunk                          # 다음 루프를 위해 텍스트 누적
        # 만약을 대비한 안전 장치 (없어도 됨)
        elif hasattr(chunk, 'content') and chunk.content is not None:
            print(chunk.content, end="", flush=True)
            current_chunk += chunk.content

    # 정제된 최종 텍스트로 previous_summary 업데이트
    previous_summary = current_chunk

print("\n-----------------\n[단계 완료] 최종 요약 정제 완료!")
return previous_summary
```

- **`결과 출력해보기`**

```
refined_summary = map_refine_chain.invoke(docs)
```

- **`map_refine_chain` 실행 결과** - (`12.6s`)
  - *스트리밍 출력 확인 완료*

```
*   미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다.
*   이 행정명령은 AI 안전 및 보안 기준 마련, 개인정보보호, 형평성 및 시민권 향상, 소비자 보호, 노동자 지원, 혁신 및 경쟁 촉진, 국제협력을
*   특히, 강력한 AI 시스템 개발 기업에게는 안전 테스트 결과 및 시스템 정보를 정부와 공유하도록 요구하며, 특정 컴퓨팅 성능 이상의 클러스터에
*   또한, AI의 차별적 사용을 방지하기 위해 법률, 주택, 보건 분야에서의 조치를 확대하고, 형사사법 시스템 및 주택 임대 분야의 AI 사용 모범
*   AI 연구 촉진을 위한 국가AI연구자원(NAIRR)을 통해 연구 인프라를 확충하고, 외국인 전문가의 미국 내 취업 및 연구 활동을 지원하기 위해 비

----------
[단계 2/5] 정제 시작...

*   미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다.
*   이 행정명령은 AI 안전 및 보안 기준 마련, 개인정보보호, 형평성 및 시민권 향상, 소비자 보호, 노동자 지원, 혁신 및 경쟁 촉진, 국제협력을
*   특히, 강력한 AI 시스템 개발 기업에게는 안전 테스트 결과 및 시스템 정보를 정부와 공유하도록 요구하며, 특정 컴퓨팅 성능 이상의 클러스터에
*   또한, AI의 차별적 사용을 방지하기 위해 법률, 주택, 보건 분야에서의 조치를 확대하고, 형사사법 시스템 및 주택 임대 분야의 AI 사용 모범
*   AI 연구 촉진을 위한 국가AI연구자원(NAIRR)을 통해 연구 인프라를 확충하고, 외국인 전문가의 미국 내 취업 및 연구 활동을 지원하기 위해 비
*   **G7은 '히로시마 AI 프로세스'를 통해 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했습니다.**
*   **이 행동강령은 AI 수명주기 전반에 걸친 위험 평가 및 완화, 투명성과 책임성 보장, 정보 공유 및 이해관계자 협력, 보안 통제, 콘텐츠 인증
*   **행동강령은 기업의 자발적 채택을 권고하며, 기반 모델과 생성 AI를 포함한 첨단 AI 시스템의 위험 관리에 중점을 둡니다.**

----------
[단계 3/5] 정제 시작...
*   미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다.
*   이 행정명령은 AI 안전 및 보안 기준 마련, 개인정보보호, 형평성 및 시민권 향상, 소비자 보호, 노동자 지원, 혁신 및 경쟁 촉진, 국제협력을
*   특히, 강력한 AI 시스템 개발 기업에게는 안전 테스트 결과 및 시스템 정보를 정부와 공유하도록 요구하며, 특정 컴퓨팅 성능 이상의 클러스터에
*   또한, AI의 차별적 사용을 방지하기 위해 법률, 주택, 보건 분야에서의 조치를 확대하고, 형사사법 시스템 및 주택 임대 분야의 AI 사용 모범
*   AI 연구 촉진을 위한 국가AI연구자원(NAIRR)을 통해 연구 인프라를 확충하고, 외국인 전문가의 미국 내 취업 및 연구 활동을 지원하기 위해 비
*   **G7은 '히로시마 AI 프로세스'를 통해 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했습니다.**
*   **이 행동강령은 AI 수명주기 전반에 걸친 위험 평가 및 완화, 투명성과 책임성 보장, 정보 공유 및 이해관계자 협력, 보안 통제, 콘텐츠 인증
*   **행동강령은 기업의 자발적 채택을 권고하며, 기반 모델과 생성 AI를 포함한 첨단 AI 시스템의 위험 관리에 중점을 둡니다.**
*   **영국에서 개최된 AI 안전성 정상회의에 참가한 28개국은 AI 위험에 공동 대응하기 위한 '블레츨리 선언'을 발표했습니다.**
*   **이 선언은 AI 안전 보장을 위해 모든 이해관계자의 협력을 강조하며, 특히 최첨단 AI 시스템 개발 기업은 안전 평가 등 적절한 조치를 취할
*   **참가국들은 첨단 AI 개발 기업의 투명성 향상, 안전 테스트 도구 개발, 공공부문 역량 구축 및 과학 연구개발 분야에서 협력하기로 합의했습니
*   **영국 총리는 첨단 AI 모델에 대한 안전성 시험 계획 수립 및 테스트 수행을 주도할 영국 AI 안전 연구소 출범을 발표했으며, 국가 안보, 안전
*   **참가국들은 AI 위험과 가능성에 대한 과학적 평가 및 향후 AI 안전 연구 우선순위 제시를 위한 '과학의 현황' 보고서 작성에도 합의했습니다.
*   **한국은 영국과 6개월 뒤 AI 미니 정상회의를, 프랑스와 1년 뒤 대면 정상회의를 공동 개최할 예정입니다.**

----------
[단계 4/5] 정제 시작...
*   미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다. 이 행정명령은 AI 안전 및 보안 기준 마련,
*   **G7은 '히로시마 AI 프로세스'를 통해 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했습니다.**
*   **영국에서 개최된 AI 안전성 정상회의에 참가한 28개국은 AI 위험에 공동 대응하기 위한 '블레츨리 선언'을 발표했습니다.** 이 선언은 AI 안
*   **한국은 영국과 6개월 뒤 AI 미니 정상회의를, 프랑스와 1년 뒤 대면 정상회의를 공동 개최할 예정입니다.**
*   **미국 법원은 예술가들이 생성 AI 기업(미드저니, 스태빌리티AI, 디비언트아트)에 제기한 저작권 침해 소송을 기각했습니다.** 법원은 기각 이

----------
[단계 5/5] 정제 시작...
*   미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발 및 사용을 위한 행정명령에 서명했습니다. 이 행정명령은 AI 안전 및 보안 기준 마련,
*   **G7은 '히로시마 AI 프로세스'를 통해 첨단 AI 시스템 개발 기업을 대상으로 AI 위험 식별 및 완화를 위한 국제 행동강령에 합의했습니다.**
*   **영국에서 개최된 AI 안전성 정상회의에 참가한 28개국은 AI 위험에 공동 대응하기 위한 '블레츨리 선언'을 발표했습니다.** 이 선언은 AI 안
*   **한국은 영국과 6개월 뒤 AI 미니 정상회의를, 프랑스와 1년 뒤 대면 정상회의를 공동 개최할 예정입니다.**
*   **미국 법원은 예술가들이 생성 AI 기업(미드저니, 스태빌리티AI, 디비언트아트)에 제기한 저작권 침해 소송을 기각했습니다.** 법원은 기각 이
*   **미국 연방거래위원회(FTC)는 저작권청에 제출한 의견서에서 생성 AI로 인한 소비자 및 창작자 피해 가능성과 빅테크의 시장 지배력 강화 우려
----------
[단계 완료] 최종 요약 정제 완료!
```

---

5) `Chain of Density`

- [논문]

- **`Chain of Density`** *(CoD)* = **`gpt-4`를 사용한 `요약 생성`을 `개선`하기 위해 `개발`된 기법**
- 개요
  - 초기에 `개체`가 `적은 요약`을 `생성` → **`길이를 늘리지 않으면서` `누락된 중요 개체`들을 `반복적`으로 `통합`**
  - 일반 프롬프트보다 **`더 추상적`이고 `정보 융합`이 `뛰어나`며, `인간`이 작성한 `요약`과 `비슷한 밀도`를 가진 것으로 나타남**
- 장점

    - **점진적 개선**:
        - CoD 는 초기에 개체가 적은 간단한 요약을 생성한 후, 단계적으로 중요한 개체들을 추가하며 요약을 개선
        - **요약의 길이는 유지되면서 정보 밀도가 증가하여 읽기 쉬우면서도 정보량이 풍부한 요약이 생성됨**
    - **정보 밀도와 가독성의 균형**:
        - **요약의 정보 밀도를 조절하여 정보성과 가독성 사이의 최적 균형점을 찾음**
        - *연구 결과*: 사람들은 일반적인 GPT-4 요약보다 *더 밀도 있지만* 사람이 작성한 요약만큼 밀도가 높지 않은 *CoD 요약을 선호* 하는 것으로 나타남
    - **추상화와 정보 융합 개선**:
        - **더 추상적이고 정보 융합이 뛰어나며, 원문의 앞부분에 치우치는 경향 (`lead bias`)이 덜함**
        - **요약의 전반적인 품질과 가독성을 향상시키는 데 기여**
- 주요 파라미터
    - **`content_category`**:
        - 콘텐츠 종류
        - 기본값: `Article`
        - *예시: 기사, 동영상 녹취록, 블로그 게시물, 연구 논문*
    - **`content`**: 요약할 콘텐츠
    - **`entity_range`**:
        - 콘텐츠에서 선택하여 요약에 추가할 **엔티티의 수의 범위**
        - **기본값 = `1-3`**
    - **`max_words`**:
        - **1번 요약할 때 요약에 포함할 최대 단어**
        - **기본값 = `80`**
    - **`iterations`**:
        - **엔티티 고밀도화 라운드 수**
        - **기본값 = `3`**
        - **총 요약 = 반복 횟수 + 1**
            - `80`단어 (*기본값*)의 경우 `3`회 반복 = 이상적
            - 요약이 더 길면 `4~5`회
            - `entity_range`를 예를 들어 1~4로 변경하는 것도 도움이 될 수 있음
    - `Chain of Density` 프롬프트 사용 → 텍스트 요약을 생성하는 체인 구성
        - *Chain of Density Prompt*

---

① `CoD Chain` 생성하기

- 첫 번째 `chain` = `중간 결과`를 보여줌
- 두 번째 `chain` = `최종 요약`만을 보여줌

```
import getpass
import os
from langchain_openai import ChatOpenAI

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

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

```
import textwrap
from langchain import hub
from langchain_openai import ChatOpenAI
from langchain_core.output_parsers import SimpleJsonOutputParser

# {content}를 제외한 모든 입력에 대한 기본값 지정
cod_chain_inputs = {
    "content": lambda d: d.get("content"),
    "content_category": lambda d: d.get("content_category", "Article"),
    "entity_range": lambda d: d.get("entity_range", "1-3"),
    "max_words": lambda d: int(d.get("max_words", 80)),
```

```
    "iterations": lambda d: int(d.get("iterations", 5)),
}
```

```
# Chain of Density 프롬프트 다운로드
cod_prompt = hub.pull("teddynote/chain-of-density-prompt")

# 프롬프트 출력해보기
cod_prompt.pretty_print()
```

- **CoD 프롬프트 출력해보기** - (`0.3s`)

```
================================ **System Message** ================================

As an expert copy-writer, you will write increasingly concise, entity-dense summaries of the user provided *{cont

A Descriptive Entity is:
- Relevant: to the main story.
- Specific: descriptive yet concise (5 words or fewer).
- Faithful: present in the *{content_category}*.
- Anywhere: located anywhere in the *{content_category}*.

# Your Summarization Process
- Read through the *{content_category}* and the all the below sections to get an understanding of the task.
- Pick *{entity_range}* informative Descriptive Entities from the *{content_category}* (";" delimited, do not add
- In your output JSON list of dictionaries, write an initial summary of max *{max_words}* words containing the En
- You now have `[{"missing_entities": "...", "denser_summary": "..."}]`

Then, repeat the below 2 steps *{iterations}* times:

- Step 1. In a new dict in the same list, identify *{entity_range}* new informative Descriptive Entities from the
- Step 2. Write a new, denser summary of identical length which covers every Entity and detail from the previous

A Missing Entity is:
- An informative Descriptive Entity from the *{content_category}* as defined above.
- Novel: not in the previous summary.

# Guidelines
- The first summary should be long (max *{max_words}* yet highly non-specific, containing little information beyo
- Make every word count: re-write the previous summary to improve flow and make space for additional entities.
- Make space with fusion, compression, and removal of uninformative phrases like "the *{content_category}* discus
- The summaries should become highly dense and concise yet self-contained, e.g., easily understood without the *{
- Missing entities can appear anywhere in the new summary.
- Never drop entities from the previous summary. If space cannot be made, add fewer new entities.
- You're finished when your JSON list has 1+*{iterations}* dictionaries of increasing density.

# IMPORTANT
- Remember, to keep each summary to max *{max_words}* words.
- Never remove Entities or details. Only add more from the *{content_category}*.
- Do not discuss the *{content_category}* itself, focus on the content: informative Descriptive Entities, and det
- Remember, if you're overusing filler phrases in later summaries, or discussing the *{content_category}* itself,
- Answer with a minified JSON list of dictionaries with keys "missing_entities" and "denser_summary".
- "denser_summary" should be written in the same language as the "content".

## Example output
[{"missing_entities": "ent1;ent2", "denser_summary": "<vague initial summary with entities 'ent1','ent2'>"}, {"mi

================================ **Human Message** ================================

*{content_category}*:
*{content}*
```

- `CoD Prompt` 한국어로 읽어보기

```
================================ **시스템 메시지** ================================

전문 카피라이터로서, 당신은 사용자가 제공한 *{content_category}*에 대해 점점 더 간결하고, 정보 밀도가 높은 요약문을 작성할 것입니다.
초기 요약은 *{max_words}* 단어 이내여야 하며, *{content_category}*에서 추출한 *{entity_range}*개의 **Descriptive Entities(설명
```

Descriptive Entity의 정의:
- **Relevant(관련성)**: 주요 이야기와 관련되어야 함.
- **Specific(구체성)**: 5단어 이하로 간결하지만 구체적이어야 함.
- **Faithful(정확성)**: 반드시 *{content_category}*에 실제로 존재해야 함.
- **Anywhere(위치 자유)**: *{content_category}* 어디에서든 나타날 수 있음.

🔷 요약 작성 절차

1. *{content_category}*와 아래의 모든 섹션을 읽어 전체적인 이해를 갖습니다.
2. *{content_category}*에서 *{entity_range}*개의 유용한 Descriptive Entities를 선택합니다. (세미콜론 `;`으로 구분하고, 띄어쓰지
3. 출력은 JSON 리스트 형식으로 작성합니다.
   이때 각 객체(dict)는 다음과 같은 구조를 가집니다:

```
[{"missing_entities": "...", "denser_summary": "..."}]
```

4. 이후 아래 두 단계를 *{iterations}*번 반복합니다.

---

* 🔁 반복 절차

  * Step 1. 이전 요약에 포함되지 않은 새로운 *{entity_range}*개의 유용한 Descriptive Entities를 *{content_category}*에서 식별합

  * Step 2. 이전 요약의 모든 엔터티와 세부 정보를 유지하면서, 새롭게 식별한 Missing Entities를 포함하는 새로운 요약문을 작성합니다. 이 요약

---

* 🔸 Missing Entity의 정의
  * Descriptive Entity의 정의를 따르며, {content_category} 내에서 유용해야 함.
  * Novel(새로움): 이전 요약에 포함되지 않은 엔터티여야 함.

---

* 🧭 작성 가이드라인
  * 첫 번째 요약은 최대 {max_words} 단어로 작성하되, 정보 밀도는 낮고 다소 모호하게 작성해야 합니다.
    * (예: “이 *{content_category}*는 …에 대해 다룹니다.” 등의 표현으로 길이를 채움)
  * 이후 요약에서는 불필요한 표현을 제거하고, 새로운 엔터티를 추가하면서 점점 더 압축적이고 밀도 높은 요약으로 발전시킵니다.
  * 모든 요약은 자체적으로 이해 가능해야 하며, 원문 없이도 의미가 통해야 합니다.
  * 엔터티를 삭제하지 마세요. 공간이 부족하면 새 엔터티의 수를 줄이세요.
  * 마지막 요약까지 모든 엔터티가 누적되어야 합니다.

---

* 🧩 표현 간소화 전략
  * 불필요한 표현(예: “이 *{content_category}*는 다룹니다”)을 제거합니다.
  * 핵심 의미만 남기고 문장을 재구성합니다.
  * fusion(결합), compression(압축), **삭제(removal)**를 통해 공간을 확보합니다.
  * 너무 일반적인 문구보다 구체적인 Descriptive Entity를 우선 사용합니다.

---

* ⚙️ 출력 형식
  * 최종 출력은 압축된 JSON 리스트 형태로 작성합니다.
  * 각 객체(dict)는 다음 두 키를 반드시 포함해야 합니다:
  * "missing_entities": 세미콜론(;)으로 구분된 누락 엔터티 목록
  * "denser_summary": 동일한 길이의 (더 밀도 높은) 요약문
  * "denser_summary"는 원문의 언어(*{content}*와 동일한 언어)로 작성해야 합니다.

---

* ✅ 예시 출력

```
[
  {
    "missing_entities": "ent1;ent2",
```

```
    "denser_summary": "모호하지만 ent1, ent2를 포함한 초기 요약문"
  },
  {
    "missing_entities": "ent3",
    "denser_summary": "ent1, ent2, ent3을 포함한 더 밀도 높은 요약문"
  }
]
```

```
---

* ⚠️ 주의사항
  *    각 요약은 최대 {max_words} 단어를 초과해서는 안 됩니다.
  *    엔터티나 세부정보를 삭제하지 마세요. 오직 추가만 가능.
  *    {content_category} 자체에 대한 논의("이 글은…" 등)는 피하고, 내용 그 자체에 집중하세요.
  *    후기 요약에서 여전히 불필요한 표현이 많다면, 더 많은 구체적 엔터티를 선택해 포함하세요.

---

* 📌 요약
  *    점진적으로 정보 밀도를 높이는 요약문 작성
  *    엔터티 기반 압축 방식
  *    JSON 리스트 출력
  *    각 단계마다 누락 엔터티 추가 및 재작성
  *    모든 내용은 *{content}* 의 언어로 작성

================================ **사용자 메시지** ================================
```

```
  {content_category}:

  {content}
```

```
# Chain of Density 체인 생성
cod_chain = (
    cod_chain_inputs
    | cod_prompt
    | ChatOpenAI(temperature=0, model="gpt-4o-mini")
    | SimpleJsonOutputParser()
)
```

```
# 두 번째 체인 생성, 최종 요약만 추출 (스트리밍 불가능, 최종 결과가 필요함)

cod_final_summary_chain = cod_chain | (
    lambda output: output[-1].get(
        "denser_summary", '오류: 마지막 딕셔너리에 "denser_summary" 키가 없습니다'
    )
)
```

② 데이터 로드

```
# 데이터 로드하기
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader("../14_Chains/data/SPRI_AI_Brief_2023년12월호_F.pdf")
docs = loader.load()
print(f"총 글자수: {len(docs[0].page_content)}")
print("\n========== 앞부분 미리보기 ==========\n")
print(docs[0].page_content[:500])
```

- 앞부분 미리 일부 출력해보기 - (`1.8s`)

```
총 글자수: 10

========== 앞부분 미리보기 ==========

2023년 12월호
```

```
content = docs[1].page_content
print(f"총 글자수: {len(content)}")
print(docs[1].page_content)
```

- 요약할 데이터 확인해보기 *(교재대로 `docs[1].page_content`)*

```
총 글자수: 1961
2023년 12월호
Ⅰ. 인공지능 산업 동향 브리프 1. 정책/법제    ▹ 미국, 안전하고 신뢰할 수 있는 AI 개발과 사용에 관한 행정명령 발표  ·····················
Ⅱ. 주요 행사   ▹CES 2024·····································································
```

③ 최종 요약 추출

- **`JSON` 딕트 목록**
  - 부분 `JSON` 스트리밍하기
  - 스트리밍된 각 청크는 새로운 접미사가 추가된 동일한 `JSON` 딕트 목록
- **`\r` 캐리지 리턴 인쇄 필요**
  - 단순 연결❌ → **`다음 청크가 이전 청크를 덮어씀`** + **`반복적으로 스트리밍 추가하는 것처럼 보이게 하는 것 필요`**

```
# 결과를 저장할 빈 리스트 초기화
results: list[dict[str, str]] = []
```

```
# cod_chain을 스트리밍 모드로 실행하고 부분적인 JSON 결과를 처리

for partial_json in cod_chain.stream(
    {"content": content, "content_category": "Article"}
):
    # 각 반복마다 results를 업데이트
    results = partial_json

    # 현재 결과를 같은 줄에 출력 (캐리지 리턴을 사용하여 이전 출력을 덮어씀)
    print(results, end="\r", flush=True)
```

- **`for partial_json in cod_chain.stream`** - (`17.8s`)

```
[{'missing_entities': 'AI 안전성;G7;AI 생성 콘텐츠', 'denser_summary': '2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관
```

```
# 총 요약 수 계산
total_summaries = len(results)
print("\n")
```

```
# 각 요약을 순회하며 처리
i = 1
for cod in results:
    # 누락된 엔티티들을 추출하고 포맷팅
    added_entities = ", ".join(
        [
            ent.strip()
            for ent in cod.get(
                "missing_entities", 'ERR: "missing_entiies" key not found'
            ).split(";")
        ]
    )
    # 더 밀도 있는 요약 추출
    summary = cod.get("denser_summary", 'ERR: missing key "denser_summary"')

    # 요약 정보 출력 (번호, 총 개수, 추가된 엔티티)
    print(
        f"### CoD Summary {i}/{total_summaries}, 추가된 엔티티(entity): {added_entities}"
        + "\n"
    )
    # 요약 내용을 80자 너비로 줄바꿈하여 출력
    print(textwrap.fill(summary, width=80) + "\n")
    i += 1

print("\n============== [최종 요약] ================\n")
print(summary)
```

- **CoD 최종 요약**

```
### CoD Summary 1/5, 추가된 엔티티(entity): AI 안전성, G7, AI 생성 콘텐츠

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의
발표를 포함합니다. 미국의 AI 개발과 사용에 관한 행정명령, 저작권 소송 기각, AI 안전 기금 조성 등도 언급됩니다.

### CoD Summary 2/5, 추가된 엔티티(entity): AI 법, AI 기술자 임금, AI 연구소

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의
발표를 포함합니다. 미국의 AI 개발과 사용에 관한 행정명령, 저작권 소송 기각, AI 안전 기금 조성, AI 법 3자 협상, AI 기술자의
평균 21% 높은 임금, AI 연구소 설립 발표도 언급됩니다.

### CoD Summary 3/5, 추가된 엔티티(entity): AI 소프트웨어 매출, AI 기업, AI 모델

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의
발표를 포함합니다. 미국의 AI 개발과 사용에 관한 행정명령, 저작권 소송 기각, AI 안전 기금 조성, AI 법 3자 협상, AI 기술자의
평균 21% 높은 임금, AI 연구소 설립 발표, AI 소프트웨어 매출 2,500억 달러 돌파 전망, AI 기업과 AI 모델의 발전도
언급됩니다.

### CoD Summary 4/5, 추가된 엔티티(entity): AI 에이전트, AI 협력, AI 위험

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의
발표를 포함합니다. 미국의 AI 개발과 사용에 관한 행정명령, 저작권 소송 기각, AI 안전 기금 조성, AI 법 3자 협상, AI 기술자의
평균 21% 높은 임금, AI 연구소 설립 발표, AI 소프트웨어 매출 2,500억 달러 돌파 전망, AI 기업과 AI 모델의 발전, AI
에이전트의 패러다임 변화, AI 협력 강화 및 AI 위험에 대한 공동 대응도 언급됩니다.

### CoD Summary 5/5, 추가된 엔티티(entity): AI 투자, AI 프런티어 모델, AI 데이터 투명성

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의
발표를 포함합니다. 미국의 AI 개발과 사용에 관한 행정명령, 저작권 소송 기각, AI 안전 기금 조성, AI 법 3자 협상, AI 기술자의
평균 21% 높은 임금, AI 연구소 설립 발표, AI 소프트웨어 매출 2,500억 달러 돌파 전망, AI 기업과 AI 모델의 발전, AI
에이전트의 패러다임 변화, AI 협력 강화, AI 위험에 대한 공동 대응, AI 투자 및 AI 데이터 투명성을 위한 데이터 출처 탐색기도
언급됩니다.


============== [최종 요약] =================

2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의 발표를 포함합니다
```

```
print(summary)
```

- **CoD 최종 요약만 출력해보기**

```
2023년 12월호에서는 AI 산업 동향을 다루며, AI 안전성 관련 G7의 국제 행동강령 합의와 AI 생성 콘텐츠 표시 의무화에 대한 유튜브의 발표를 포함합니다
```

---

6) Clustering-Map-Refine

- 원 저자 및 출처: gkamradt
- 이론 연구 배경
  - map-reduce, map-refine → **모두 시간이 오래 걸리고, 비용이 많이 발생**
  - 문서를 **몇 개(N 개)의 클러스터로 나눔 → 가장 중심축에서 가까운 문서를 클러스터의 대표 문서로 인지 → 이를 map-reduce (혹은 map-refine) 방식으로 요약하는 방식을 제안**

- **데이터 로드**

```
from langchain_community.document_loaders import PyMuPDFLoader

loader = PyMuPDFLoader("../14_Chains/data/SPRI_AI_Brief_2023년12월호_F.pdf")
```

```
docs = loader.load()
len(docs)                                                    # 23
```

- **데이터 병합**
  - 텍스트 (`len(docs) # 23`)들을 하나의 문서로 합치기 *(문자수 약 28k)*
  - 목적: **page별로 구분하지 않기 위함**

```
# 하나의 Text로 모든 문서를 연결하기
texts = "\n\n".join([doc.page_content for doc in docs])
len(texts)                                    # 27977
```

- **`RecursiveCharacterTextSplitter`** → 하나의 `Text`를 여러 문서로 나누기

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=100
    )

split_docs = text_splitter.split_text(texts)
```

- **나누어진 문서의 수 확인하기**

```
# 총 문서의 수 확인하기
len(split_docs)                                  # 79
```

- **임베딩 모델 생성하기**
  - **`HuggingFace` - `BM-K/KoSimCSE-roberta` 임베딩 모델 사용**
    - 특징: `한국어 특화 임베딩`
    - 성능:
      - 한국어 문장 유사도 측정에 매우 강력함
      - 한국어 문장 유사도 측정 및 및 클러스터링에 최적
    - `HuggingFace`에서 무료 사용 가능한 **`오픈 소스`** **모델**

```
from langchain_community.embeddings import HuggingFaceEmbeddings

embeddings = HuggingFaceEmbeddings(
    model_name="BM-K/KoSimCSE-roberta",
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
)

print("✅ 임베딩 모델 (KoSimCSE-roberta) 로드 준비 완료.")
```

```
vectors = embeddings.embed_documents(split_docs)            # 11.2s
```

- **`KMeans-clustering`**
  - 총 79개의 문서 → 10개의 클러스터로 나누기

```
from sklearn.cluster import KMeans

# 클러스터 수를 선택하면 문서의 콘텐츠에 따라 조정할 수 있음
num_clusters = 10

# Perform K-means clustering
kmeans = KMeans(n_clusters=num_clusters, random_state=123).fit(vectors)
```

- **라벨링된 결과 확인하기**

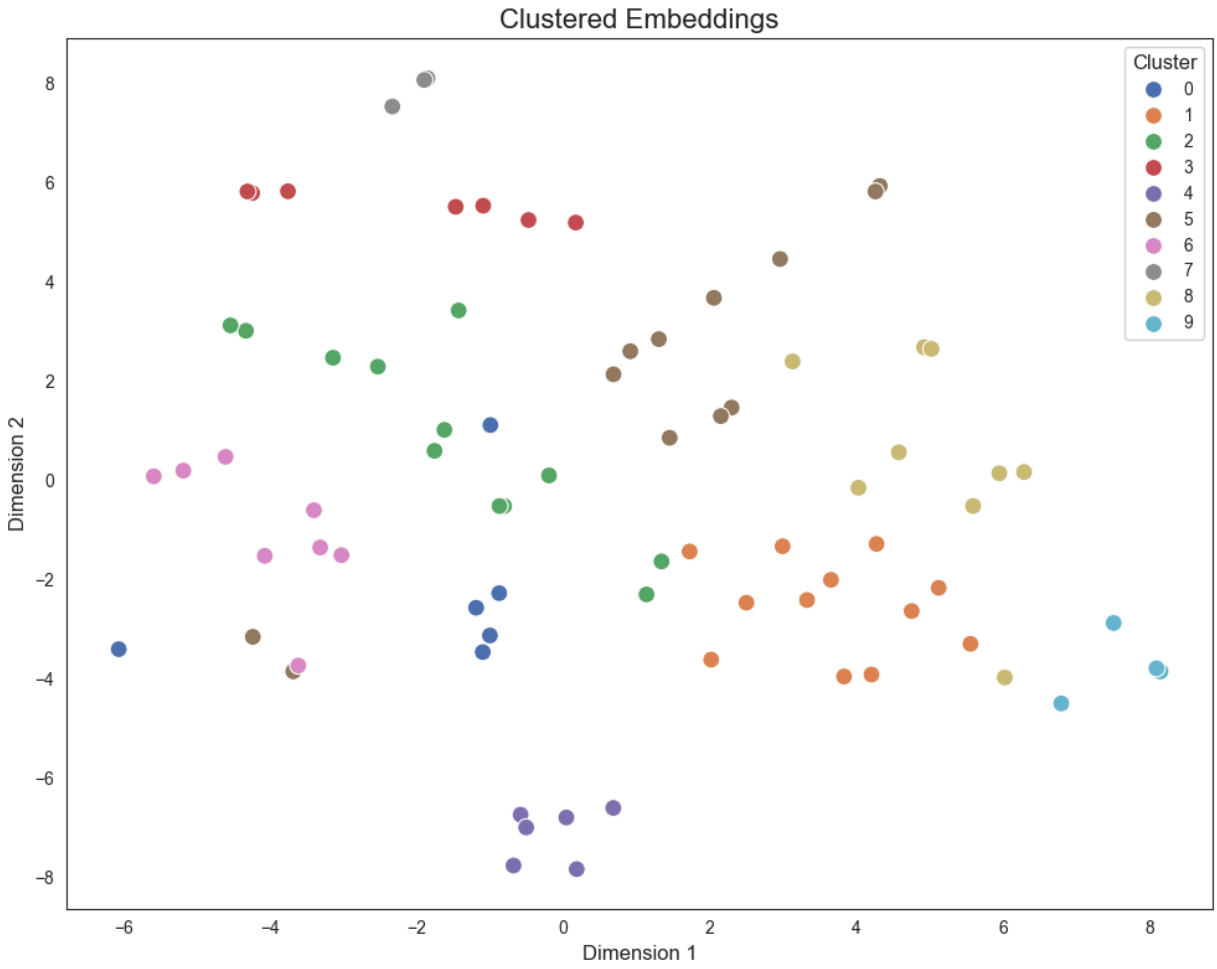
```
# 결과 확인
kmeans.labels_
```

- **`kmeans.labels_`** 결과 확인해보기 (`1.2s`)

```
array([5, 4, 4, 4, 4, 4, 4, 0, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 8, 7, 7,
    7, 5, 3, 3, 3, 3, 9, 9, 9, 9, 8, 8, 8, 8, 2, 2, 2, 2, 2, 2, 2, 5,
    2, 2, 0, 5, 5, 5, 8, 5, 5, 5, 5, 0, 0, 0, 0, 3, 3, 3, 1, 1, 1, 8,
    6, 6, 6, 6, 6, 6, 6, 5, 6, 5, 8, 8, 2], dtype=int32)
```

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 경고 제거
import warnings

warnings.filterwarnings("ignore")

# t-SNE 수행 및 2차원으로 축소
tsne = TSNE(n_components=2, random_state=42)
reduced_data_tsne = tsne.fit_transform(np.array(vectors))

# seaborn 스타일 설정
sns.set_style("white")

# 축소된 데이터 플롯
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x=reduced_data_tsne[:, 0],
    y=reduced_data_tsne[:, 1],
    hue=kmeans.labels_,
    palette="deep",
    s=100,
)

plt.xlabel("Dimension 1", fontsize=12)
plt.ylabel("Dimension 2", fontsize=12)
plt.title("Clustered Embeddings", fontsize=16)
plt.legend(title="Cluster", title_fontsize=12)

# 배경색 설정
plt.gcf().patch.set_facecolor("white")

plt.tight_layout()
plt.show()
```

- `Clustering Embeddings` 결과



- **각 cluster의 중심점에 가장 가까운 임베딩을 찾아서 저장하기**

```
import numpy as np

# 가장 가까운 점들을 저장할 빈 리스트 생성
closest_indices = []


# 클러스터 수만큼 반복
for i in range(num_clusters):

    # 해당 클러스터 중심으로부터의 거리 목록 구하기
    distances = np.linalg.norm(vectors - kmeans.cluster_centers_[i], axis=1)

    # 가장 가까운 점의 인덱스 찾기 (argmin을 사용하여 최소 거리 찾기)
    closest_index = np.argmin(distances)

    # 해당 인덱스를 가장 가까운 인덱스 리스트에 추가
    closest_indices.append(closest_index)
```

- **오름차순 정렬**
  - 문서의 요약을 순서대로 진행하기 위함

```
# 문서의 요약을 순서대로 진행하기 위하여 오름차순 정렬
selected_indices = sorted(closest_indices)
selected_indices
```

- 오름차순 정렬

```
[np.int64(2),
np.int64(17),
np.int64(21),
np.int64(25),
np.int64(29),
np.int64(35),
np.int64(36),
np.int64(51),
np.int64(55),
```

```
np.int64(68)]
```

- `Document` 객체 = 문서 생성하기
  - 10개의 선택된 문서 출력하기

```
from langchain_core.documents import Document

# 선택된 문서 출력하기
selected_docs = [Document(page_content=split_docs[doc]) for doc in selected_indices]
selected_docs
```

- 10개의 선택된 문서 출력해보기

```
[Document(metadata={}, page_content='▹ 미국 연방거래위원회, 저작권청에 소비자 보호와 경쟁 측면의 AI 의견서 제출····· 5
Document(metadata={}, page_content='•선언은 AI 안전 보장을 위해 국가, 국제기구, 기업, 시민사회, 학계를 포함한 모든 이해관계자의 협력이
Document(metadata={}, page_content='•2023년 1월 예술가 사라 앤더슨(Sarah Anderson), 캘리 맥커넌(Kelly McKernan), 칼라 \n오르
Document(metadata={}, page_content='•저작권청은 생성 AI와 관련된 저작권법과 정책 이슈를 조사하고 있으며, 폭넓은 의견 수렴을 통해 \n입법
Document(metadata={}, page_content='적용하는 계층적 접근방식에 따라 기반 모델 규제에 대한 기본적인 합의에 도달\n•그러나 11월 10일 열린
Document(metadata={}, page_content='한편, 첨단 AI 모델의 취약점이나 잠재적으로 위험한 기능 및 위험 완화 관련 정보를 공유할 수 \n있는 공
Document(metadata={}, page_content='SPRi AI Brief |  \n2023-12월호\n8\n코히어, 데이터 투명성 확보를 위한 데이터 출처 탐색기 공개
Document(metadata={}, page_content='SPRi AI Brief |  \n2023-12월호\n12\nIDC, 2027년 AI 소프트웨어 매출 2,500억 달러 돌파
Document(metadata={}, page_content='1. 정책/법제  \n2. 기업/산업 \n3. 기술/연구 \n 4. 인력/교육\n빌 게이츠, AI 에이전트로 인한 컴
Document(metadata={}, page_content='•(생태학적 타당도를 갖춘 벤치마크 사용) AGI에 대한 벤치마크는 사람들이 경제적· 사회적 또는 예술적으
```

```
from langchain_core.runnables import chain
from langchain_openai import ChatOpenAI
from langchain_core.callbacks import StreamingStdOutCallbackHandler
from langchain_core.output_parsers import StrOutputParser                    # 임포트는 유지
from langchain import hub                                                    # hub 임포트 추가

@chain
def map_refine_chain2(docs):
    """
    Map-Refine 로직을 실행하고 결과를 실시간 스트리밍하며 반환하는 함수.

    입력: Document 리스트 (docs). (Invoke 시 docs를 바로 받음)
    출력: 최종 요약 텍스트
    """

    if not isinstance(docs, list) or not docs:
        raise ValueError("입력은 Document 객체 리스트여야 하며 비어 있으면 안 됩니다.")

    # map 프롬프트 다운로드
    map_summary = hub.pull("teddynote/map-summary-prompt")

    # map llm 생성
    map_llm = ChatOpenAI(
            model_name="gpt-4o-mini",
            temperature=0,
    )

    map_chain = (
        map_summary
        | map_llm
        # | StrOutputParser()                                    # <-- 제거! 스트리밍 루프가 직접 content를 추출
    )

    # map chain 구성 및 실행
    map_generator = map_chain.stream({"documents": [docs[0]], "language": "Korean"})

    # 첫 번째 문서부터 스트리밍으로 초기 요약 진행
    previous_summary = ""

    for chunk in map_generator:
        # chunk = AIMessageChunk 객체 → .content 속성을 갖고 있음
        if chunk and chunk.content is not None:
            print(chunk.content, end="", flush=True)
            previous_summary += chunk.content

    # refine 프롬프트 다운로드
    refine_prompt = hub.pull("teddynote/refine-prompt")

    # refine llm 생성하기
    refine_llm = ChatOpenAI(
```

```
            model_name="gpt-4o-mini",
            temperature=0,
        # callbacks, streaming 설정은 루프에서 직접 처리하므로 제거
    )

    # refine chain 생성하기
    # Refine Chain은 StrOutputParser()를 포함 (아래 루프에서 str 객체를 받아 처리해야 함)
    refine_chain = refine_prompt | refine_llm | StrOutputParser()

    remaining_docs = docs[1:]

    for i, current_doc in enumerate(remaining_docs):
        print(f"\n\n-----------------\n[단계 {i+2}/{len(remaining_docs)+1}] 정제 시작...")

        # Refine 체인을 스트리밍으로 호출
        stream_generator = refine_chain.stream({
            "previous_summary": previous_summary,
            "current_summary": [current_doc],
            "language": "Korean"
        })

        # 🚨 [핵심] 스트리밍 출력 및 다음 루프를 위한 결과 누적
        current_chunk = ""
        for chunk in stream_generator:
            if isinstance(chunk, str):
                print(chunk, end="", flush=True)                # 실시간으로 출력 (중복 없음)
                current_chunk += chunk                          # 다음 루프를 위해 텍스트 누적
            # 만약을 대비한 안전 장치 (없어도 됨)
            elif hasattr(chunk, 'content') and chunk.content is not None:
                print(chunk.content, end="", flush=True)
                current_chunk += chunk.content
        previous_summary = current_chunk

    print("\n-----------------\n[단계 완료] 최종 요약 정제 완료!")
    return previous_summary
```

```
# 이전에 생성한 map_refine_chain을 사용하여 요약 생성
refined_summary2 = map_refine_chain2.invoke(selected_docs)
```

- **`map_refine_chain`을 사용한 요약 생성하기** *(스트리밍 출력 확인)* - (`2m 53.8s`)

```
- 미국 연방거래위원회가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다.
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다.
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다.
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.

-----------------
[단계 2/10] 정제 시작...
- 미국 연방거래위원회가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다.
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다.
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다.
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연

-----------------
[단계 3/10] 정제 시작...
- 미국 연방거래위원회가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다.
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다.
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다.
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로

-----------------
[단계 4/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다.
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다.
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로
```

```
------------------
[단계 5/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다.
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로

------------------
[단계 6/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다. 이 기금은 첨단 AI 모델의 취약점이나 위험 완화 관련 정보를 공유할 수 있
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기를 공개했다.
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로

------------------
[단계 7/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다. 이 기금은 첨단 AI 모델의 취약점이나 위험 완화 관련 정보를 공유할 수 있
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기(Data Provenance Explorer)를 공개했다. 이 플랫폼은 12개 기관과 협력하여 데이터셋의
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로

------------------
[단계 8/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다. 이 기금은 첨단 AI 모델의 취약점이나 위험 완화 관련 정보를 공유할 수 있
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기(Data Provenance Explorer)를 공개했다. 이 플랫폼은 12개 기관과 협력하여 데이터셋의
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로
- IDC의 예측에 따르면 AI 소프트웨어 시장은 2027년까지 2,510억 달러에 이를 것으로 보이며, 생성 AI 플랫폼과 애플리케이션은 283억 달러의 매출을

------------------
[단계 9/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다. 이 기금은 첨단 AI 모델의 취약점이나 위험 완화 관련 정보를 공유할 수 있
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기(Data Provenance Explorer)를 공개했다. 이 플랫폼은 12개 기관과 협력하여 데이터셋의
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로
- IDC의 예측에 따르면 AI 소프트웨어 시장은 2027년까지 2,510억 달러에 이를 것으로 보이며, 생성 AI 플랫폼과 애플리케이션은 283억 달러의 매출을
- 빌 게이츠는 5년 내에 일상 언어로 모든 작업을 처리할 수 있는 AI 에이전트가 보급될 것이라고 전망하며, 이는 컴퓨터 사용 방식과 소프트웨어 산업을 완

------------------
[단계 10/10] 정제 시작...
- 미국 연방거래위원회(FTC)가 저작권청에 AI 관련 소비자 보호와 경쟁에 대한 의견서를 제출했다. FTC는 생성 AI의 개발과 배포가 소비자, 근로자, 중소
- EU의 AI 법안 3자 협상이 기반모델 규제에 대한 견해 차이로 어려움을 겪고 있다. 특히, 프랑스, 독일, 이탈리아의 대표들이 기반모델에 대한 모든 유형
- 미국 프런티어 모델 포럼이 1,000만 달러 규모의 AI 안전 기금을 조성했다. 이 기금은 첨단 AI 모델의 취약점이나 위험 완화 관련 정보를 공유할 수 있
- 코히어가 데이터 투명성을 높이기 위해 데이터 출처 탐색기(Data Provenance Explorer)를 공개했다. 이 플랫폼은 12개 기관과 협력하여 데이터셋의
- 알리바바 클라우드가 최신 대형 언어 모델인 '통이치엔원 2.0'을 발표했다.
- 영국 총리 리시 수낙이 AI 안전성 정상회의를 마무리하며 첨단 AI 시스템의 안전 테스트 계획을 발표하고, 영국 AI 안전 연구소의 출범을 알렸다. 이 연
- 2023년 1월, 예술가들이 이미지 생성 AI 서비스를 개발한 3개 기업을 상대로 저작권 침해 소송을 제기했으나, 법원은 예술가들의 저작권 미등록을 이유로
- IDC의 예측에 따르면 AI 소프트웨어 시장은 2027년까지 2,510억 달러에 이를 것으로 보이며, 생성 AI 플랫폼과 애플리케이션은 283억 달러의 매출을
- 빌 게이츠는 5년 내에 일상 언어로 모든 작업을 처리할 수 있는 AI 에이전트가 보급될 것이라고 전망하며, 이는 컴퓨터 사용 방식과 소프트웨어 산업을 완
- 연구진은 AGI(인공지능 일반화)의 발전 상태를 0~5단계로 분류하고, 현재 범용 AI는 1단계 수준에 있으며, 특수 AI는 5단계까지 달성되었다고 밝혔다.
------------------
[단계 완료] 최종 요약 정제 완료!
```

```
# 최종 결과 출력
```