ch3 Summarizing text using Langchain

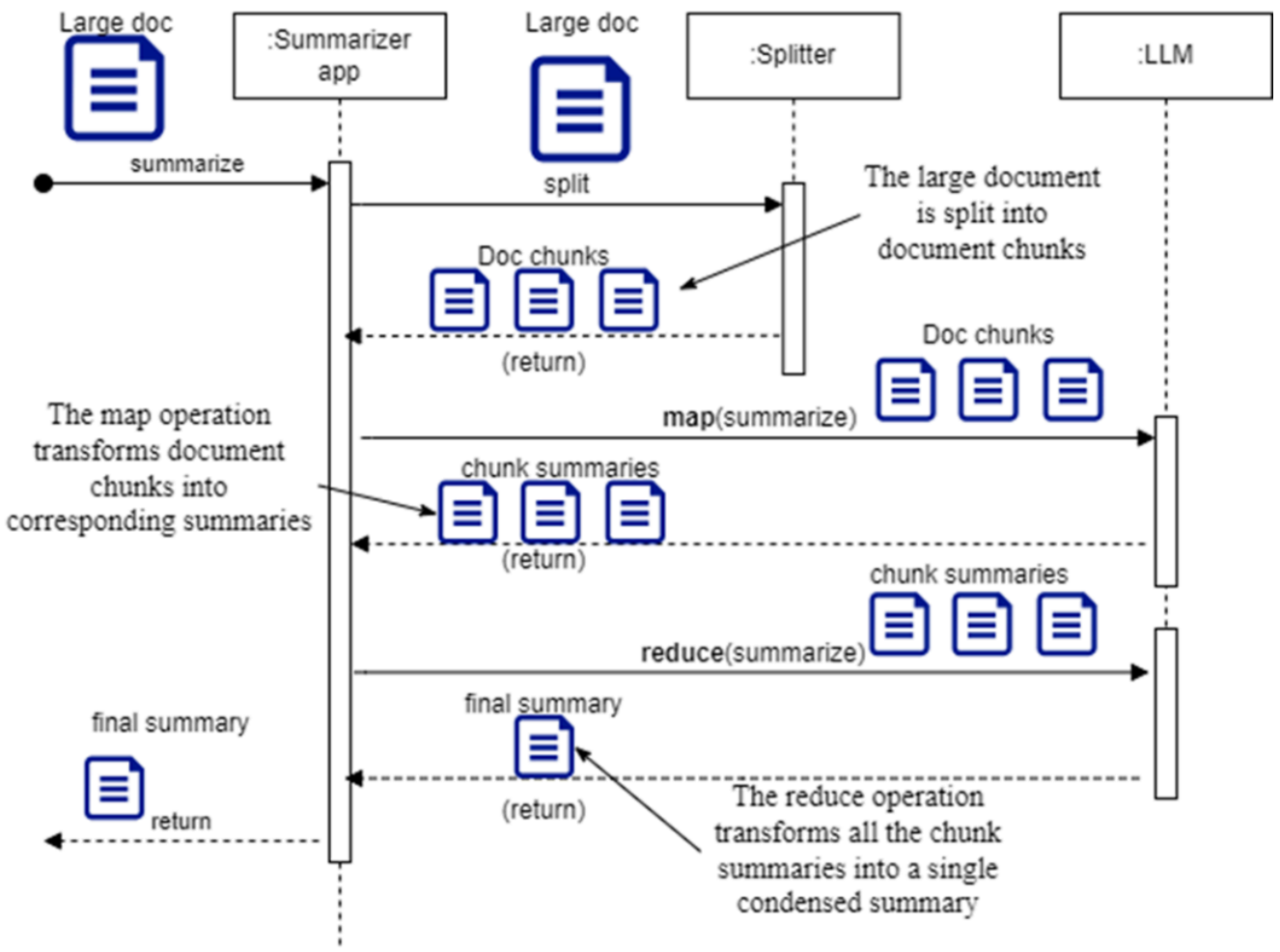
Lesson 3 Summarizing Text using Langchain

- LLM의 Context-window를 초과하는 대용량 문서 요약
- 여러 문서에 걸친 요약(Summarizing across documents)

---

1. LLM context-wondow를 초과하는 대용량 문서 요약

인기 있는 LLM들의 컨텍스트 위도우가 계속 확장되고 있는 추세이지만, 여전히 선택한 모델의 토큰 한도를 초과하는 문서를 다뤄야 할 경우가 있다. 이러한 경우에는 아래 그림에 묘사된 것처럼 Map-Reduce방식의 접근법을 사용할 수 있다.



위 그림에 묘사된 Context window를 초과하는 큰 문서를 요약하는 방법은 다음과 같다.

- 문서를 Splitter를 이용하여 더 작은 Chunk 단위로 분할하기
- 각각의 Chunk단위를 요약하기(Map)
- 모든 요약된 Chunk를 이용하여 전체 요약 작성하기(Reduce)

2. 프로그래밍을 이용한 Map-reduce Summary

2-1 Jupyter Notebook 환경설정

- Window cmd 또는 Anaconda Jupyter 터미널에서 다음 예제와 같이 폴더를 생성 → 이동 → virtual environment 생성 → Virtual Environment 활성화

```
C:\Github\building-llm-applications>md ch03 # directory 생성
C:\Github\building-llm-applications>cd ch03 # 폴더로 이동
C:\Github\building-llm-applications\ch03>python -m venv env_03 #virtual env 생성
C:\Github\building-llm-applications\ch03>.\env_ch03\Scripts\activate #activate
(env_ch03) C:\Github\building-llm-applications\ch03>
```

- 실행에 필요한 라이브러리 설치 → 노트북 실행

```
(env_ch03) C:\Github\building-llm-applications\ch03>pip install tiktoken notebook
langchain langchain-openai

(env_ch03) C:\Github\building-llm-applications\ch03>jupyter notebook
```

- 실행에 필요한 라이브러리 셋업 및 예제 파일(Moby-Dick.txt) 불러들이기

```
from langchain_openai import ChatOpenAI
from langchain_text_splitters import TokenTextSplitter
from langchain_core.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnableLambda, RunnableParallel
import getpass

with open("./Moby-Dick.txt", 'r', encoding='utf-8') as f:
    moby_dick_book = f.read()
```

- OpenAI key 입력.

```
OPENAI_API_KEY = getpass.getpass('Enter your OPENAI_API_KEY')
```

- 언어모델 준비

```
llm = ChatOpenAI(openai_api_key=OPENAI_API_KEY,model_name="gpt-4o-mini")
```

- 텍스트 Split(분할) 기능

```
# Split
text_chunks_chain = (
    RunnableLambda(lambda x:
        [
            {
                'chunk': text_chunk,
            }
            for text_chunk in
              TokenTextSplitter(chunk_size=3000, chunk_overlap=100).split_text(x)
        ]
    )
)
```

- 각 분할된 Chunk를 요약하는 Map 기능 구현(병렬 처리 'RunnableParallel')

```
# Map
summarize_chunk_prompt_template = """
Write a concise summary of the following text, and include the main details.
Text: {chunk}
"""

summarize_chunk_prompt = PromptTemplate.from_template(summarize_chunk_prompt_template)
summarize_chunk_chain = summarize_chunk_prompt | llm

summarize_map_chain = (
    RunnableParallel (
        {
            'summary': summarize_chunk_chain | StrOutputParser()
        }
```

```
    )
)
```

- 최종 요약 Reduce기능 구현

```
# Reduce
summarize_summaries_prompt_template = """
Write a coincise summary of the following text, which joins several summaries, and include the main details.
Text: {summaries}
"""

summarize_summaries_prompt = PromptTemplate.from_template(summarize_summaries_prompt_template)
summarize_reduce_chain = (
    RunnableLambda(lambda x:
        {
            'summaries': '\n'.join([i['summary'] for i in x]),
        })
    | summarize_summaries_prompt
    | llm
    | StrOutputParser()
)
```

- 마지막으로, 문서 Splitting chain, Map chain , Reduce chain 체인을 하나의 Map-Reduce chain으로 결합

```
map_reduce_chain = (
   text_chunks_chain
   | summarize_map_chain.map()
   | summarize_reduce_chain
)
```

- map_reduce_chain실행

```
summary = map_reduce_chain.invoke(moby_dick_book)
```

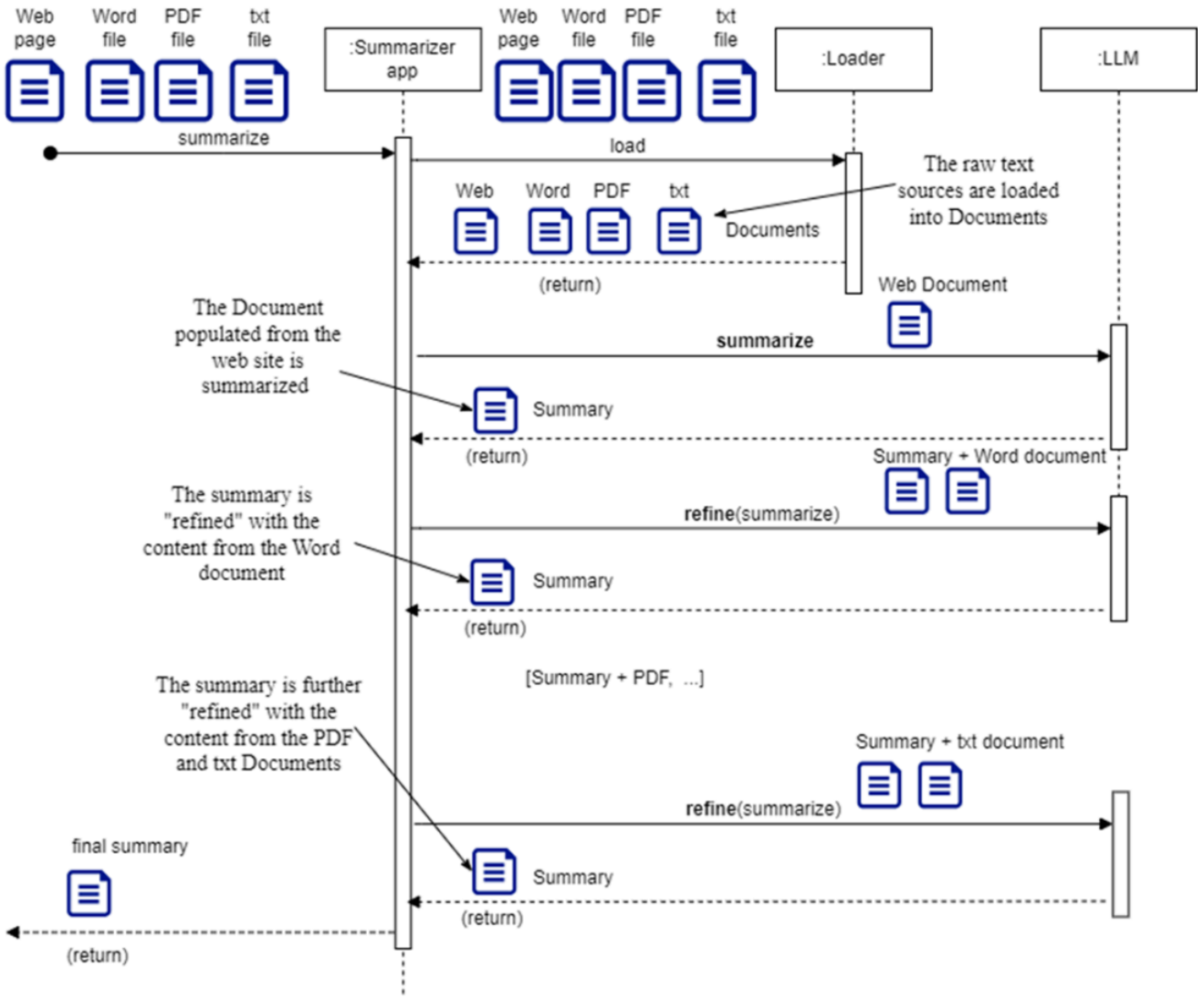
- 결과 출력

```
print(summary)
```

```
The introduction to the Project Gutenberg eBook of "Moby-Dick; or The Whale" by Herman
Melville outlines the book's availability and updates, with the first eBook release in
June 2001 and the latest in August 2021. The narrative begins with Ishmael, the
narrator, who seeks solace at sea to escape his melancholic state, showcasing the
ocean's allure compared to city life. He reflects on his reasons for joining a whaling
voyage, driven by a fascination with whales and a thirst for adventure. After arriving
in New Bedford, Ishmael faces challenges finding lodging, ultimately settling at "The
Spouter Inn," where he encounters a chaotic environment and a mysterious harpooneer
named Queequeg.
As Ishmael shares a bed with Queequeg, whom he initially fears to be a cannibal, he
gradually overcomes his apprehensions. The morning after their first night together
highlights their strange yet developing bond, as Ishmael observes Queequeg's unique
customs and politeness, emphasizing themes of fate, choice, and the allure of the
unknown in the whaling industry
```

3. 여러 문서에 걸친 요약(Summarizing across documents) - Refine(정제) 기법을 적용한 여러 문서 요약

Refine을 적용한 문서 요약 방법에서는 요약 초안을 점진적으로 다듬어가며 최종 요약을 완성한다. 각 문서는 현재 요약 초안과 함께 LLM에 전달되어 요약되고, 이 과정을 모든 문서에 반복하면 최종 요약을 완성할 수 있다. 앞서 소개한Map-reduce 방식은 처리 부담을 줄이기 위해 일부 내용 손실이 허용되는 상황에서 대량의 텍스트를 요약하는데 적합하다. 반면, Refine기법은 각 문서의 핵심 내용을 빠짐없이 담아내고자 할 때 더 효과적이다. Refine 기법의 전체적인 작동 원리는 다음 그림에 잘 요약되어 있다.



- 다양한 소스의 문서를 불러온다.
- 첫 번째 문서(web 문서)를 요약한다.
- 두 번째 단계에서는 Word 문서가 web 문서 요약본을 정제(refine)하며 요약을 이어간다.
- 이러한 과정을 모든 문서에 반복하여 최종 요약본을 완성한다.

4. 프로그래밍을 이용한 Refine기능 구현

- 필요한 라이브러리 설치

```
(env_ch03) C:\Github\Building-llm-applications\ch03>pip install wikipedia docx2txt pypdf
langchain-community
```

- Jupyter notebook 실행

```
(env_ch03) C:\Github\Building-llm-applications\ch03>jupyter notebook
```

- wikipedia에서 관련 페이지 불러오기

```
from langchain_community.document_loaders import WikipediaLoader

wikipedia_loader = WikipediaLoader(query="Paestum", load_max_docs=2)
wikipedia_docs = wikipedia_loader.load()
```

- 로컬 폴더에서 word, pdf, txt파일 로딩하기(ch03_summarization > Paestum폴더)

```
from langchain_community.document_loaders import Docx2txtLoader
from langchain_community.document_loaders import PyPDFLoader
from langchain_community.document_loaders import TextLoader
```

```
word_loader = Docx2txtLoader("Paestum/Paestum-Britannica.docx")
word_docs = word_loader.load()

pdf_loader = PyPDFLoader("Paestum/PaestumRevisited.pdf")
pdf_docs = pdf_loader.load()

txt_loader = TextLoader("Paestum/Paestum-Encyclopedia.txt")
txt_docs = txt_loader.load()
```

- 모든 문서 통합

```
all_docs = wikipedia_docs + word_docs + pdf_docs + txt_docs
```

- 문서 요약 프롬프트 작성 및 doc_summary_chain 생성

```
doc_summary_template = """Write a concise summary of the following text:
{text}
DOC SUMMARY:"""
doc_summary_prompt = PromptTemplate.from_template(doc_summary_template)

doc_summary_chain = doc_summary_prompt | llm
```

- Refine 프롬프트 작성 및 refine_chain 생성

```
refine_summary_template = """
You must produce a final summary from the current refined summary
which has been generated so far and from the content of an additional document.
This is the current refined summary generated so far: {current_refined_summary}
This is the content of the additional document: {text}
Only use the content of the additional document if it is useful,
otherwise return the current full summary as it is."""

refine_summary_prompt = PromptTemplate.from_template(refine_summary_template)

refine_chain = refine_summary_prompt | llm | StrOutputParser()
```

- Refine요약 기능 구현

```
def refine_summary(docs):
intermediate_steps = []
current_refined_summary=''

for doc in docs:
    # Step1: summarize the individual document using doc_summary_chain
    doc_summary = doc_summary_chain.invoke({"text": doc.page_content})

    # Step2 : Create a intermediate step with the current refined summary and the document summary.
    intermediate_step = \
      {"current_refined_summary": current_refined_summary,
       "text": doc_summary}
    intermediate_steps.append(intermediate_step)

    current_refined_summary = refine_chain.invoke(intermediate_step)
```

```
    return {"final_summary": current_refined_summary,
            "intermediate_steps": intermediate_steps}
```

- refine_summary기능 실행

```
full_summary = refine_summary(all_docs)
```

- 결과 출력

```
print(full_summary)
```

```
{'final_summary': "**Final Summary:**\n\nPaestum, an ancient Greek city located on the
coast of the Tyrrhenian Sea in Magna Graecia, was established around 600 BC by settlers
from Sybaris and originally named Poseidonia. The city flourished as a Greek settlement
[... SHORTENED ...] those interested in ancient Greek culture and architecture.",
'intermediate_steps': [{'current_refined_summary': '', 'text': 'Paestum ( PEST-əm, US
also PEE-stəm, Latin: ['pae stũː]) was a major ancient Greek city on the coast of the
Tyrrhenian Sea, in Magna Graecia. The ruins of Paestum are famous for their three
ancient Greek temples in the Doric order dating from about 550 to 450 BC that are in an
excellent state of preservation. The city walls and amphitheatre [... SHORTENED ...]
```

5. 핵심내용 요약

- 보통 요약의 첫 단계는 각 원본 텍스트를 LangChain의 Document 객체로 로딩하는 것이다.
- 많은 요약 작업에서는 Map-Reduce 방식이 효과적이다. 이 방식은 "맵" 단계에서 각 문서나 청크를 개별적으로 요약하고, "리듀스" 단계에서 이 요약들을 통합하여 최종 요약을 만든다.
- 여러 문서를 요약할 경우, 각 문서를 적절한 로더로 불러온 뒤 개별적으로 요약하고, 각 문서의 요약을 점진적으로 통합하며 전체 요약을 정제해 나갈 수 있다.
- Map-reduce 방식은 일부 내용 손실이 허용되는 대용량 텍스트 요약에 적합하고, Refine 기법은 각 부분의 핵심을 최종 요약에 충실히 담아낼 수 있도록 해준다.