

RAG 기반 LLM 서비스 구현

Streamlit 기반 웹 서비스 구현하기

충북대학교 의과대학 박 승



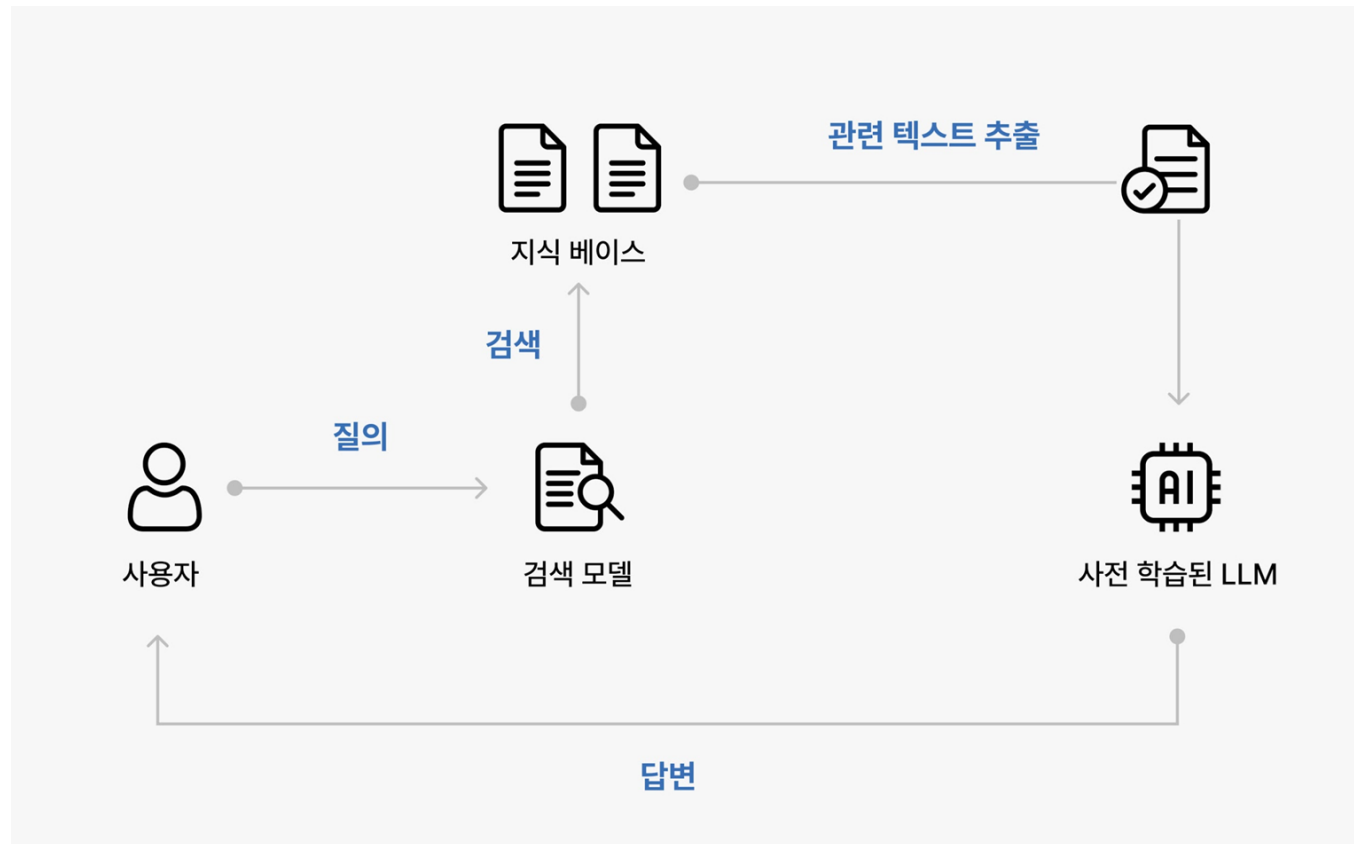
충북대학교
CHUNGBUK NATIONAL UNIVERSITY

학습 목표

1. 검색 증강 생성 기법에 대해 이해한다.
2. 검색 증강 생성 기법을 활용한 다양한 웹 서비스를 개발한다.

검색 증강 생성 (Retrieval Augmented Generation)

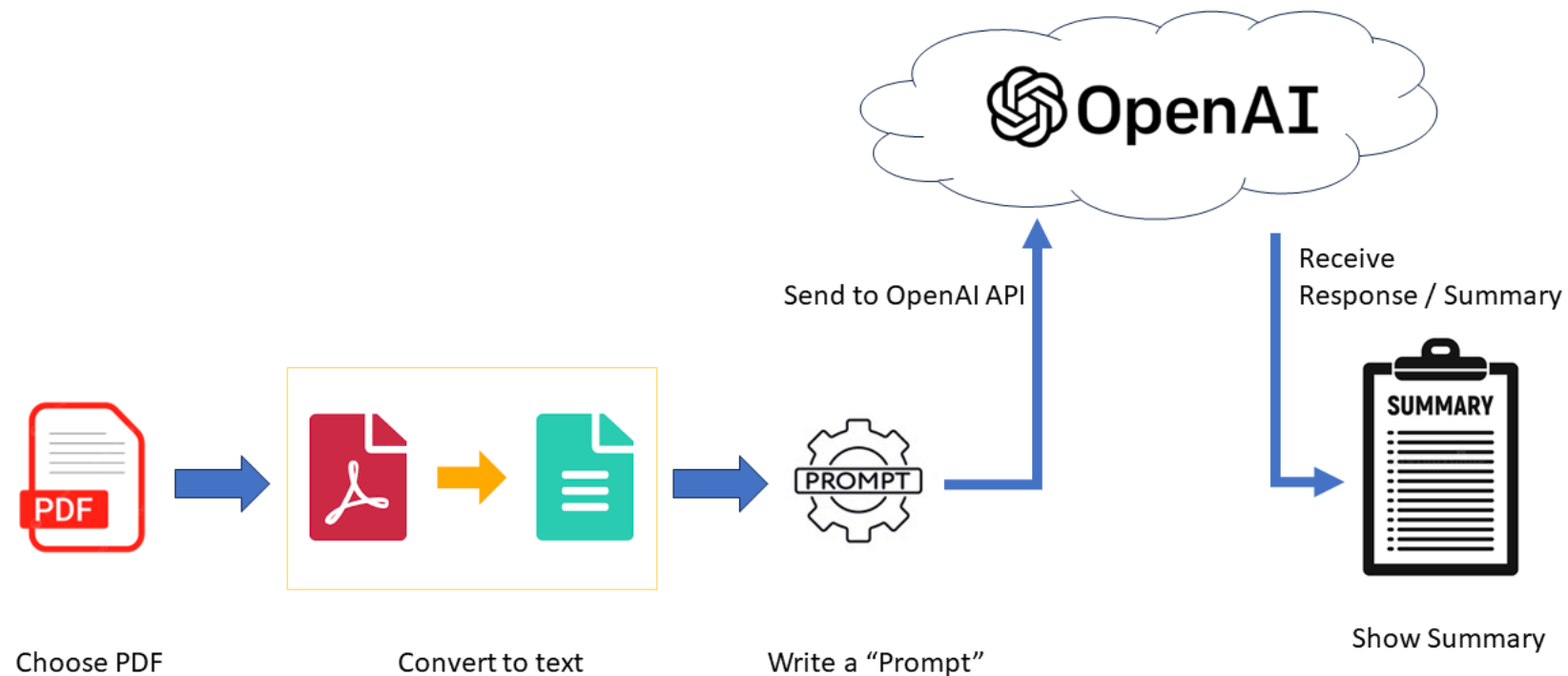
- 대규모 언어 모델(LLM)의 출력을 향상시키기 위해 외부 지식 베이스를 참조하는 기술



검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 1) PDF 요약 서비스

- 메인 프레임워크

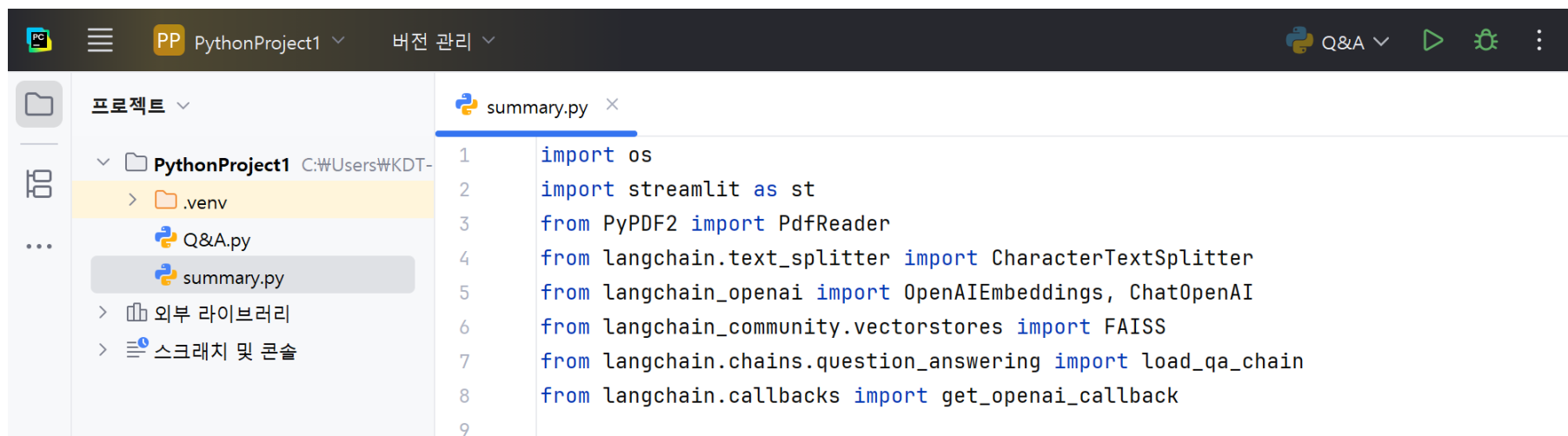


- 설치: `pip install streamlit PyPDF2 langchain langchain-openai langchain-community openai faiss-cpu`

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 1) PDF 요약 서비스

- 1. PyCharm 실행 및 summary.py 파일 생성



```
1 import os
2 import streamlit as st
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import CharacterTextSplitter
5 from langchain_openai import OpenAIEmbeddings, ChatOpenAI
6 from langchain_community.vectorstores import FAISS
7 from langchain.chains.question_answering import load_qa_chain
8 from langchain.callbacks import get_openai_callback
9
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 1) PDF 요약 서비스

- 2. 텍스트 분할, 임베딩, DB 저장 파트 구현

```
10  # OpenAI API 키 설정
11  os.environ["OPENAI_API_KEY"] = "sk-proj-0V0VDnhzv6-1R2n-10D1_voC0oi_AlFWZpcb-ItyRvJaf"
12
13  # 텍스트를 분할하여 임베딩 후 FAISS 벡터 DB 생성
14  def process_text(text): 1개의 사용 위치
15      text_splitter = CharacterTextSplitter(
16          separator="\n",
17          chunk_size=1000,
18          chunk_overlap=200,
19          length_function=len
20      )
21      chunks = text_splitter.split_text(text)
22      embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")
23      documents = FAISS.from_texts(chunks, embeddings)
24      return documents
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 1) PDF 요약 서비스

■ 3. Streamlit 기반 웹 어플리케이션 구현

```

26 # Streamlit 메인 애플리케이션
27 def main(): 1개의 사용 위치
28     st.title("GPT-4 기반 PDF 요약기")
29     st.divider()
30
31     # PDF 파일 업로드
32     pdf = st.file_uploader('PDF 파일을 업로드하세요.', type='pdf')
33
34     if pdf:
35         # PDF 텍스트 추출
36         pdf_reader = PdfReader(pdf)
37         text = ""
38         for page in pdf_reader.pages:
39             text += page.extract_text()
40
41         # 텍스트 처리 후 벡터 DB 생성
42         documents = process_text(text)
43
44         # 요약 쿼리
45         query = "업로드된 PDF 파일의 내용을 약 3~5 문장으로 요약해주세요."
46
47         if query:
48             # 벡터 유사도 기반 문서 검색
49             docs = documents.similarity_search(query, k=3)

```

```

51
52 # GPT-4 모델로 질의응답 체인 설정
53 llm = ChatOpenAI(model="gpt-4", temperature=0.1)
54 chain = load_qa_chain(llm, chain_type='stuff')
55
56 # 요약 생성 및 API 사용 비용 출력
57 with get_openai_callback() as cost:
58     response = chain.run(input_documents=docs, question=query)
59     print(cost)
60
61 st.subheader("요약 결과:")
62 st.write(response)
63
64 # 애플리케이션 실행
65 if __name__ == '__main__':
66     main()

```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 1) PDF 요약 서비스

- 4. PyCharm 터미널 내 streamlit run summary.py 를 통해 실행

```
터미널  로컬  ×  Command Prompt  ×  +  ▾  
Microsoft Windows [Version 10.0.26100.4202]  
(c) Microsoft Corporation. All rights reserved.  
  
(kdtcb_learn) C:\Users\KDT-19\PycharmProjects\PythonProject1>streamlit run summary.py
```

PythonProject1 > summary.py

GPT-4 기반 PDF 요약기

PDF 파일을 업로드하세요.



Drag and drop file here

Limit 200MB per file • PDF

Browse files



의료바이오_서비스개발_07_고급_프롬프트_공학.pdf 1.6MB



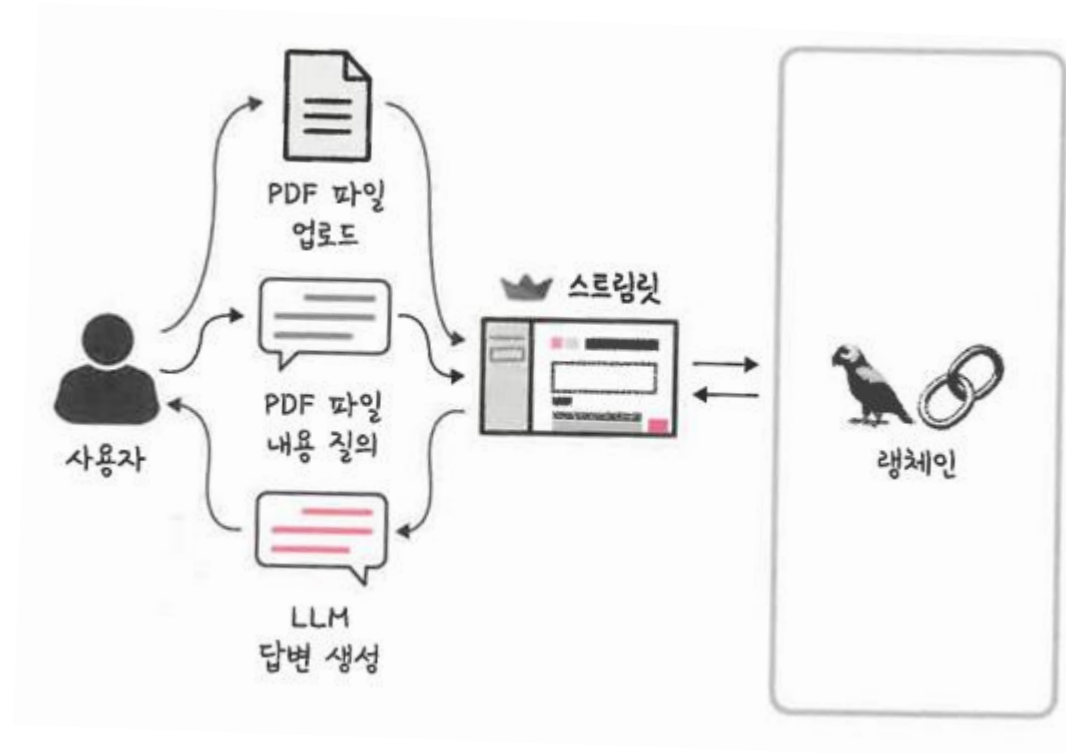
요약 결과:

이 PDF 파일은 고급 프롬프트 공학에 대한 내용을 다루고 있습니다. 이는 자연어 처리 시스템에서 원하는 출력 결과를 얻기 위해 입력 프롬프트를 설계하고 최적화하는 기술입니다. 제로샷, 퓨샷 학습, Chain of Thought (CoT) 프롬프트, 자기 일관성, 그리고 Tree of Thoughts (ToT) 프롬프트 등의 다양한 기법들이 소개되어 있습니다. 이러한 기법들은 AI 모델의 성능을 극대화하고 사용자가 의도한 결과를 정확하게 도출하는 데 중요합니다.

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 2) PDF 질의응답 서비스

■ 메인 프레임워크

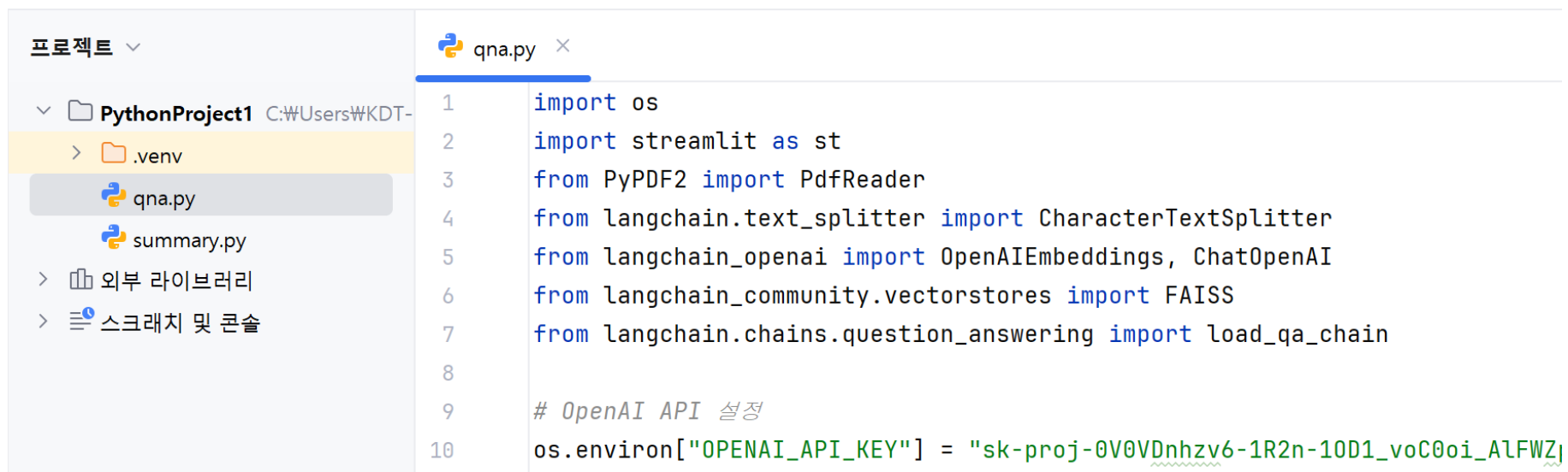


■ 설치: pip install streamlit PyPDF2 langchain langchain-openai langchain-community openai faiss-cpu

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 2) PDF 질의응답 서비스

- 1. PyCharm 실행 및 qna.py 파일 생성



The screenshot shows the PyCharm IDE interface. On the left, the '프로젝트' (Project) view displays the file structure of 'PythonProject1'. The '.venv' folder is expanded, showing 'qna.py' and 'summary.py'. The main editor window displays the code for 'qna.py'.

```

1  import os
2  import streamlit as st
3  from PyPDF2 import PdfReader
4  from langchain.text_splitter import CharacterTextSplitter
5  from langchain_openai import OpenAIEmbeddings, ChatOpenAI
6  from langchain_community.vectorstores import FAISS
7  from langchain.chains.question_answering import load_qa_chain
8
9  # OpenAI API 설정
10 os.environ["OPENAI_API_KEY"] = "sk-proj-0V0VDnhzv6-1R2n-10D1_voC0oi_A1FWZj
  
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 2) PDF 질의응답 서비스

▪ 2. Streamlit 기반 웹 서비스 구현

```

12 # Streamlit 제목 설정
13 st.title("PDF 기반 GPT-4 질의응답")
14
15 # PDF 파일 업로드
16 pdf = st.file_uploader("PDF 파일을 업로드하세요.", type='pdf')
17
18 if pdf:
19     # PDF에서 텍스트 추출
20     reader = PdfReader(pdf)
21     text = ""
22     for page in reader.pages:
23         text += page.extract_text()
24
25     # 텍스트를 작은 단위(chunk)로 분할
26     splitter = CharacterTextSplitter(
27         separator="\n",
28         chunk_size=1000,
29         chunk_overlap=200,
30         length_function=len
31     )
32     chunks = splitter.split_text(text)
33
34     # 임베딩을 이용한 벡터 저장소 생성
35     embeddings = OpenAIEmbeddings()
36     vector_store = FAISS.from_texts(chunks, embeddings)
  
```

```

34 # 임베딩을 이용한 벡터 저장소 생성
35 embeddings = OpenAIEmbeddings()
36 vector_store = FAISS.from_texts(chunks, embeddings)
37
38 # 사용자 질의 입력
39 query = st.text_input("PDF 내용에 대해 질문하세요:")
40
41 if query:
42     # 유사도 검색으로 관련 chunk 추출
43     docs = vector_store.similarity_search(query, k=3)
44
45     # GPT-4 모델로 질의응답 체인 로드
46     llm = ChatOpenAI(model_name="gpt-4", temperature=0)
47     qa_chain = load_qa_chain(llm, chain_type="stuff")
48
49     # 질의에 대한 답변 생성
50     response = qa_chain.run(input_documents=docs, question=query)
51
52     # 결과 표시
53     st.subheader("답변 결과")
54     st.write(response)
  
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 2) PDF 질의응답 서비스

- 3. PyCharm 터미널 내 streamlit run qna.py 를 통해 실행

```

터미널  로컬 ×  Command Prompt ×  +  ▾
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

(kdtcb_learn) C:\Users\KDT-19\PycharmProjects\PythonProject1>streamlit run qna.py
  
```

PDF 기반 GPT-4 질의응답

PDF 파일을 업로드하세요.



Drag and drop file here

Limit 200MB per file • PDF

Browse files



의료바이오_서비스개발_07_고급_프롬프트_공학.pdf 1.6MB



PDF 내용에 대해 질문하세요:

고급 프롬프트 공학 기술들에 대해 설명해줘.

답변 결과

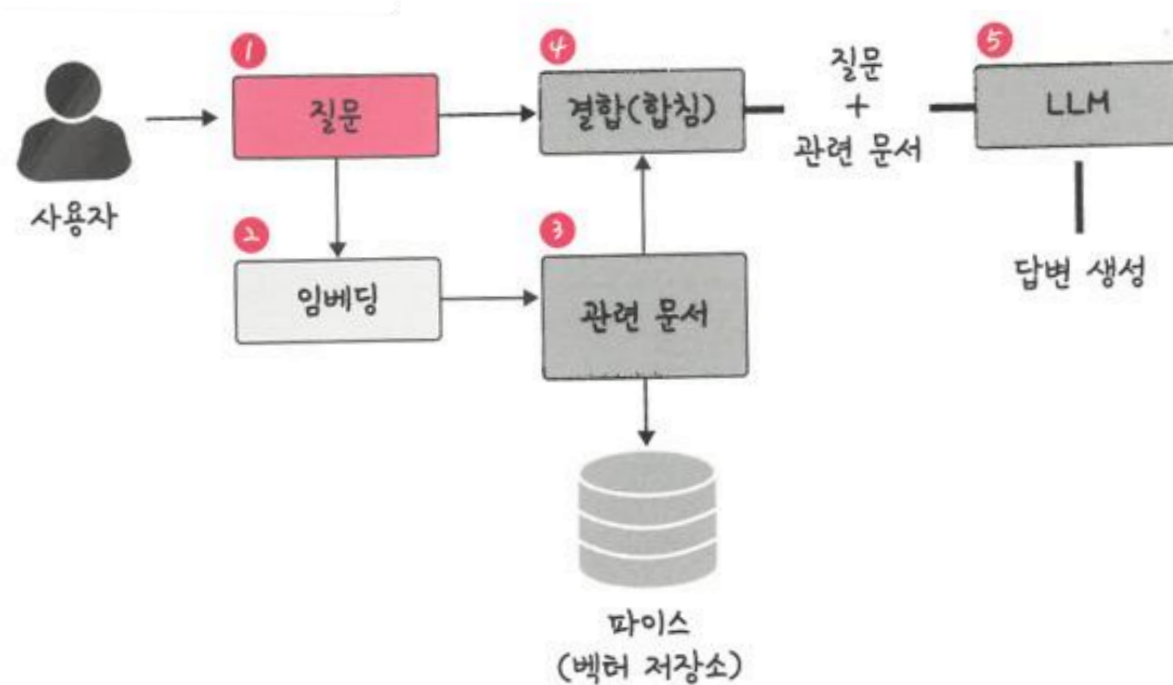
고급 프롬프트 공학에는 다음과 같은 기술들이 포함됩니다:

1. 제로샷(zero-shot) 프롬프트: 어떠한 데모나 예제를 제공하지 않고 과업 지침을 직접 LLM(Language Model)에게 전달하는 방식입니다.
2. 퓨샷(few-shot) 학습: 명시적인 지침 없이 작업과 관련된 몇 가지 입력-출력 예제만을 LLM에게 제공하는 방식입니다. 이를 통해 답변의 품질과 정확도를 크게 개선할 수 있습니다.
3. Chain of Thought (CoT) 프롬프트: 중간 추론 단계를 포함한 응답을 점두사로 추가하여 모델이 추론할 수 있는 능력을 부여하는 방식입니다.
4. 자기일관성(Self-consistency): 질문에 대해 여러 후보 답변을 생성한 후, 가장 일관된 또는 가장 빈번한 답변을 최종 출력으로 선택하는 방식입니다.
5. Tree of Thoughts (ToT) 프롬프트: 문제를 해결하기 위해 다양한 사고 경로(branch)를 탐색하는 방식입니다. 이를 통해 문제 해결 성능을 개선할 수 있습니다.

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 3) PDF 기반 대화형 서비스

■ 메인 프레임워크

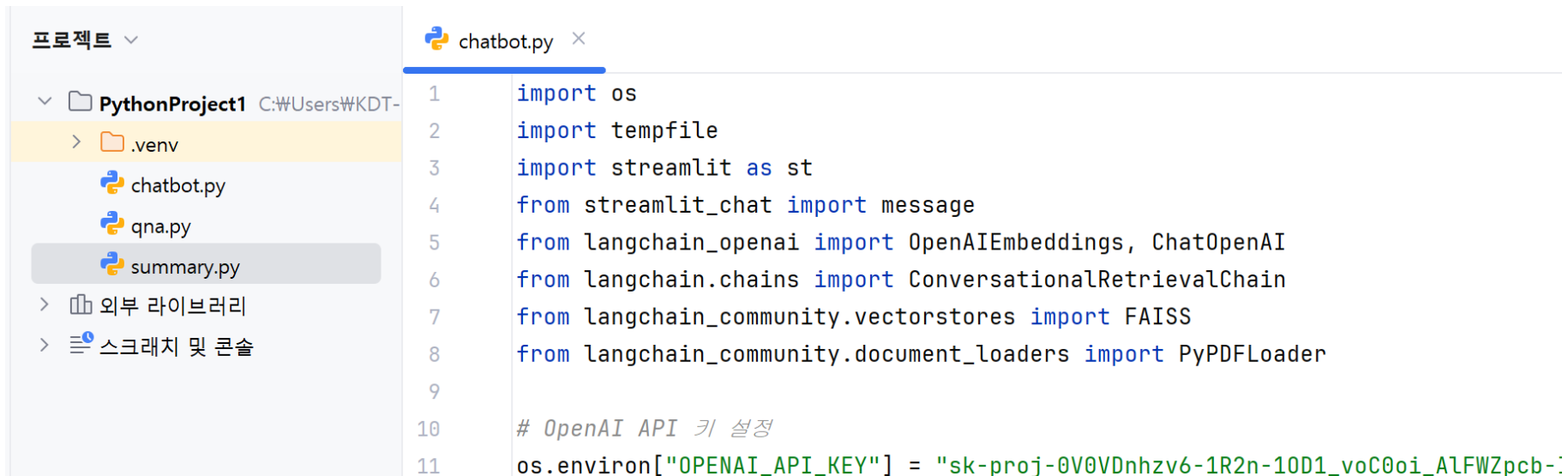


- 설치: `pip install streamlit streamlit-chat langchain langchain-openai langchain-community openai PyPDF2 faiss-cpu`

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 3) PDF 기반 대화형 서비스

- 1. PyCharm 실행 및 chatbot.py 파일 생성



```

1  import os
2  import tempfile
3  import streamlit as st
4  from streamlit_chat import message
5  from langchain_openai import OpenAIEmbeddings, ChatOpenAI
6  from langchain.chains import ConversationalRetrievalChain
7  from langchain_community.vectorstores import FAISS
8  from langchain_community.document_loaders import PyPDFLoader
9
10 # OpenAI API 키 설정
11 os.environ["OPENAI_API_KEY"] = "sk-proj-0V0VDnhzv6-1R2n-10D1_voC0oi_AlFWZpcb-:"
  
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 3) PDF 기반 대화형 서비스

▪ 2. Streamlit 기반 웹 서비스 구현

```

13 # 사이드바에서 PDF 파일 업로드
14 uploaded_file = st.sidebar.file_uploader("PDF 파일을 업로드하세요.", type="pdf")
15
16 if uploaded_file:
17     # 임시파일로 PDF 저장 후 로딩
18     with tempfile.NamedTemporaryFile(delete=False) as tmp_file:
19         tmp_file.write(uploaded_file.getvalue())
20         tmp_file_path = tmp_file.name
21
22     # PDF에서 문서 로드
23     loader = PyPDFLoader(tmp_file_path)
24     data = loader.load()
25
26     # 문서 임베딩 및 벡터 DB 생성
27     embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")
28     vectors = FAISS.from_documents(data, embeddings)
29
30     # GPT-4 기반 대화형 검색 체인 설정
31     chain = ConversationalRetrievalChain.from_llm(
32         llm=ChatOpenAI(temperature=0.0, model_name='gpt-4'),
33         retriever=vectors.as_retriever()
34 )

```

```

36 # 대화 처리 함수
37 def conversational_chat(query): 1개의 사용 위치
38     result = chain({
39         "question": query,
40         "chat_history": st.session_state['history']
41     })
42     st.session_state['history'].append((query, result["answer"]))
43     return result["answer"]
44
45 # 세션 상태 초기화
46 if 'history' not in st.session_state:
47     st.session_state['history'] = []
48
49 if 'generated' not in st.session_state:
50     st.session_state['generated'] = [f"{uploaded_file.name}에 관해 질문주세요."]
51
52 if 'past' not in st.session_state:
53     st.session_state['past'] = ["안녕하세요!"]
54
55 # 사용자 입력 컨테이너
56 response_container = st.container()
57 container = st.container()

```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 3) PDF 기반 대화형 서비스

▪ 2. Streamlit 기반 웹 서비스 구현

```
59 with container:
60     with st.form(key='Conv_Question', clear_on_submit=True):
61         user_input = st.text_input(
62             "Query:",
63             placeholder="PDF 파일에 대해 질문하세요. :)",
64             key='input'
65         )
66         submit_button = st.form_submit_button(label='Send')
67
68     if submit_button and user_input:
69         output = conversational_chat(user_input)
70         st.session_state['past'].append(user_input)
71         st.session_state['generated'].append(output)
72
73     # 채팅 기록 출력
74     if st.session_state['generated']:
75         with response_container:
76             for i in range(len(st.session_state['generated'])):
77                 message(st.session_state["past"][i], is_user=True, key=f'{i}_user', avatar_style="fun-emoji", seed="Nala")
78                 message(st.session_state["generated"][i], key=str(i), avatar_style="bottts", seed="Fluffy")
79
```


검색 증강 생성 (Retrieval Augmented Generation)

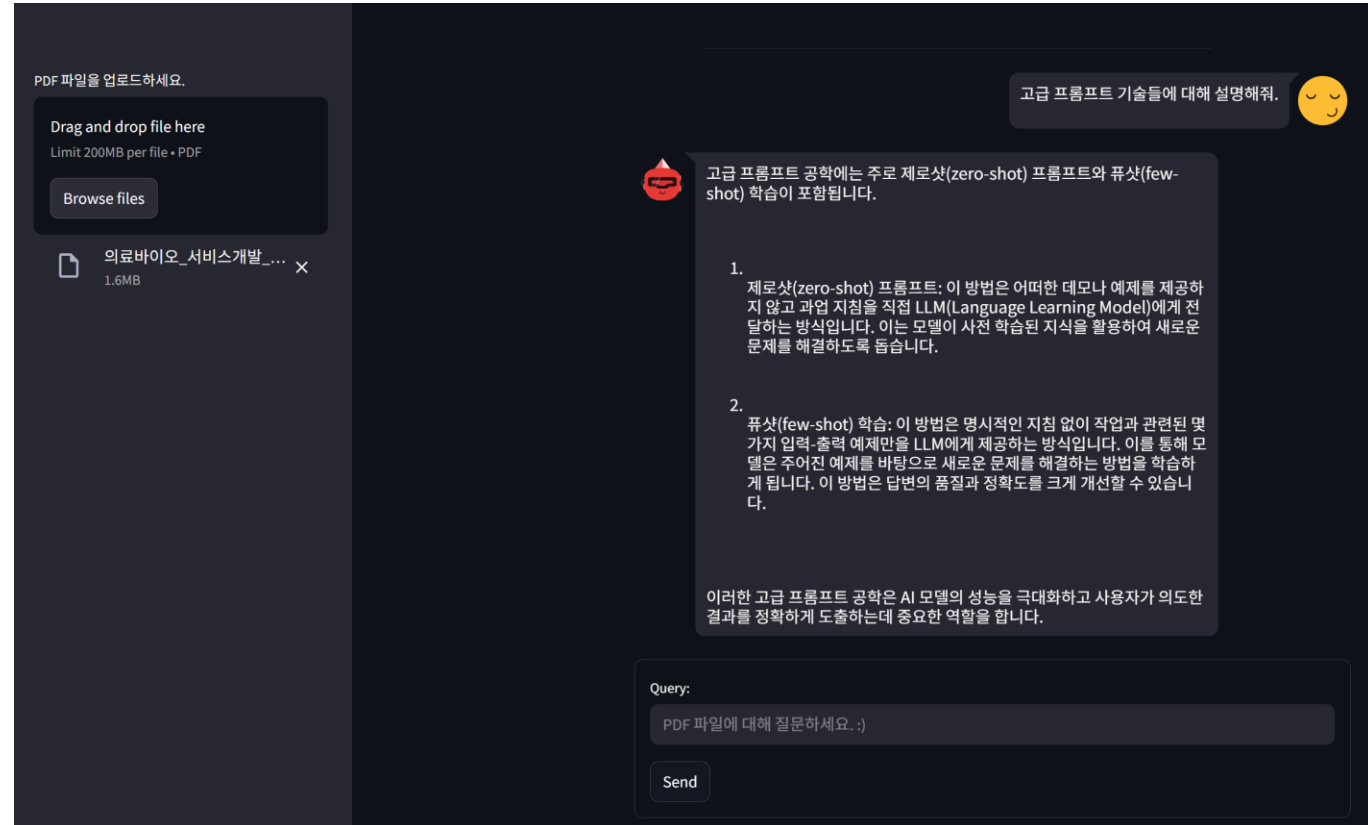
■ 실습 3) PDF 기반 대화형 서비스

- 3. PyCharm 터미널 내 streamlit run chatbot.py 를 통해 실행

```

터미널  로컬  ×  Command Prompt  ×  +  ▾
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

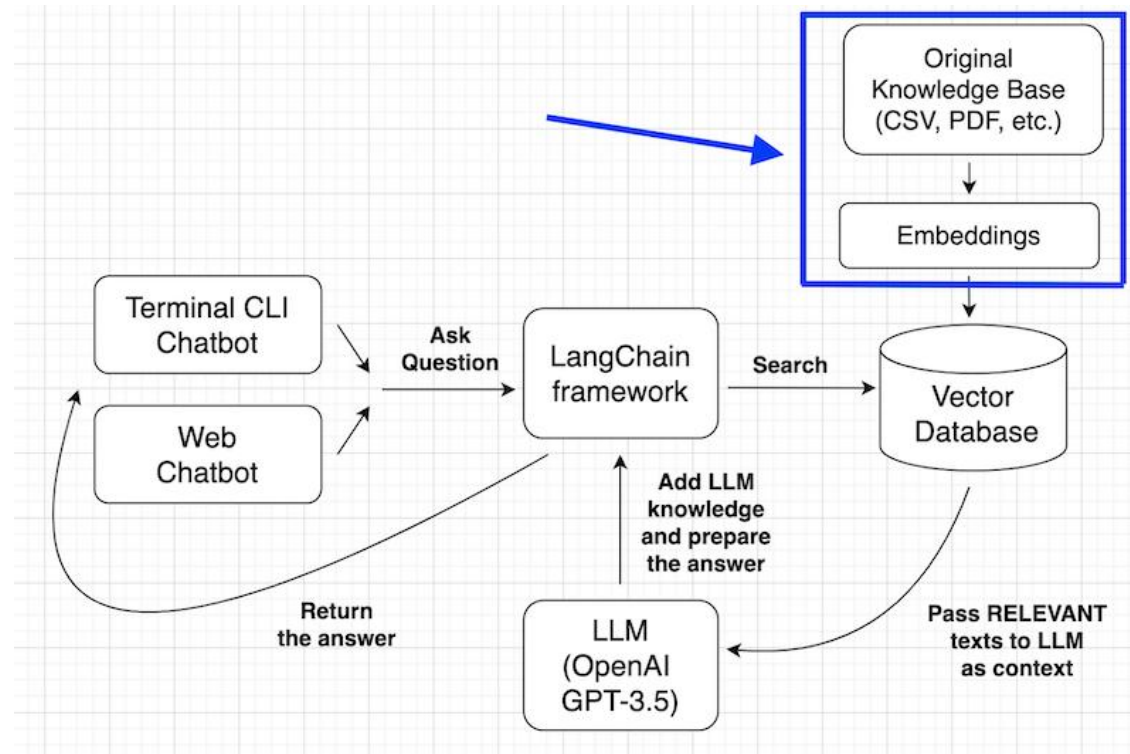
(kdtcb_learn) C:\Users\KDT-19\PycharmProjects\PythonProject1>streamlit run chatbot.py
  
```



검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 4) CSV 파일 분석 서비스

■ 메인 프레임워크

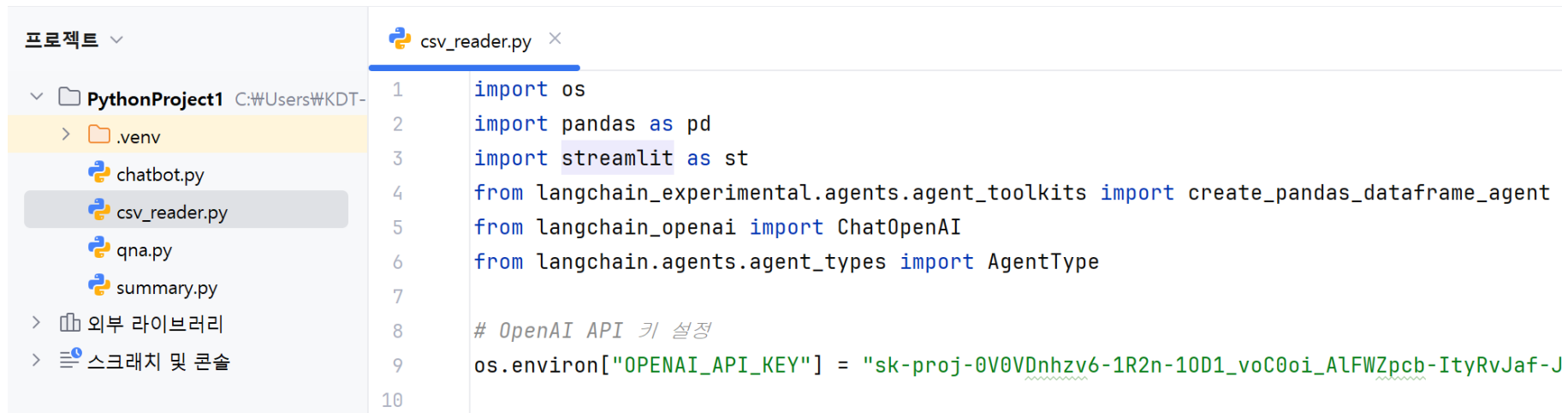


■ 설치: pip install streamlit pandas langchain langchain-openai langchain-experimental openai

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 4) CSV 파일 분석 서비스

- 1. PyCharm 실행 및 csv_reader.py 파일 생성



The screenshot shows the PyCharm IDE interface. On the left, the '프로젝트' (Project) view displays the file structure of 'PythonProject1'. The files listed are chatbot.py, csv_reader.py (highlighted), qna.py, and summary.py. On the right, the editor window shows the code for csv_reader.py. The code includes imports for os, pandas (as pd), and streamlit (as st). It also imports create_pandas_dataframe_agent from langchain_experimental.agents.agent_toolkits, ChatOpenAI from langchain_openai, and AgentType from langchain.agents.agent_types. A comment indicates the OpenAI API key setting, followed by the assignment of the API key to the environment variable OPENAI_API_KEY.

```

1  import os
2  import pandas as pd
3  import streamlit as st
4  from langchain_experimental.agents.agent_toolkits import create_pandas_dataframe_agent
5  from langchain_openai import ChatOpenAI
6  from langchain.agents.agent_types import AgentType
7
8  # OpenAI API 키 설정
9  os.environ["OPENAI_API_KEY"] = "sk-proj-0V0VDnhzv6-1R2n-10D1_voC0oi_AlFWZpcb-ItyRvJaf-J"
10
  
```

검색 증강 생성 (Retrieval Augmented Generation)

■ 실습 4) CSV 파일 분석 서비스

```

11 # Streamlit 앱 제목 설정
12 st.title("CSV 데이터 분석 웹 서비스")
13
14 # CSV 파일 업로드
15 uploaded_file = st.file_uploader("CSV 파일을 업로드하세요", type="csv")
16
17 if uploaded_file:
18     # CSV 파일을 데이터프레임으로 읽기
19     df = pd.read_csv(uploaded_file)
20     st.write("### 업로드된 데이터 미리보기:")
21     st.dataframe(df.head())
22
23 # LangChain 데이터프레임 분석 에이전트 생성
24 agent = create_pandas_dataframe_agent(
25     ChatOpenAI(temperature=0, model='gpt-4o'),
26     df,
27     verbose=False,
28     agent_type=AgentType.OPENAI_FUNCTIONS,
29     allow_dangerous_code=True
30 )
  
```

```

32 # 사용자가 데이터 분석 질문 입력
33 st.write("### 데이터 분석 질문 입력")
34 user_query = st.text_input("질문 입력", placeholder="질문을 입력하세요.")
35
36 # 분석 수행 및 결과 출력
37 if st.button("분석 시작"):
38     if user_query:
39         with st.spinner("분석 중..."):
40             result = agent.run(user_query)
41             st.write("### 분석 결과:")
42             st.write(result)
  
```

검색 증강 생성 (Retrieval Augmented Generation)

실습 4) CSV 파일 분석 서비스

```
터미널 로컬 × Command Prompt × + ∨
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

(kdtcb_learn) C:\Users\KDT-19\PycharmProjects\PythonProject1>streamlit run csv_reader.py
```

CSV 데이터 분석 웹 서비스

CSV 파일을 업로드하세요



Drag and drop file here

Limit 200MB per file • CSV

Browse files



diabetes.csv 23.3KB



업로드된 데이터 미리보기:



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc	Ag
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

데이터 분석 질문 입력

질문 입력

Glucose가 100 이상인 케이스는 얼마나 돼?

분석 시작

분석 결과:

Glucose가 100 이상인 케이스는 571건입니다.

감사합니다

Q&A



충북대학교
CHUNGBUK NATIONAL UNIVERSITY