



# [관동대] RAG 챗봇 만들기

## Step1: VS code로 각 모듈 구현해보기

개요

모듈 구현

Loader

Splitter

Storage

Retriever

Generator

Chaining

Streamlit 실행

## Step2: 프롬프트 엔지니어링 (Prompt Engineering)

모델에 페르소나 및 지시사항 입력

모델의 페르소나 및 조건 변경 실습

원하는 답변을 얻기위한 질문 규칙

문서 기반 질문 답변 실습

## Step3: 챗봇 꾸미기 및 고도화

Streamlit 코드를 통해 UI를 변경 실습

챗봇의 인사말 변경

챗봇 타이틀 변경

특정 입력을 통한 요약하기

프론트엔드 꾸미기

## Step4: 본인만의 챗봇 만들어보기

Step1 체크리스트

Step2 체크리스트

Step3 체크리스트

Step4 체크리스트 (optional)

## Step1: VS code로 각 모듈 구현해보기

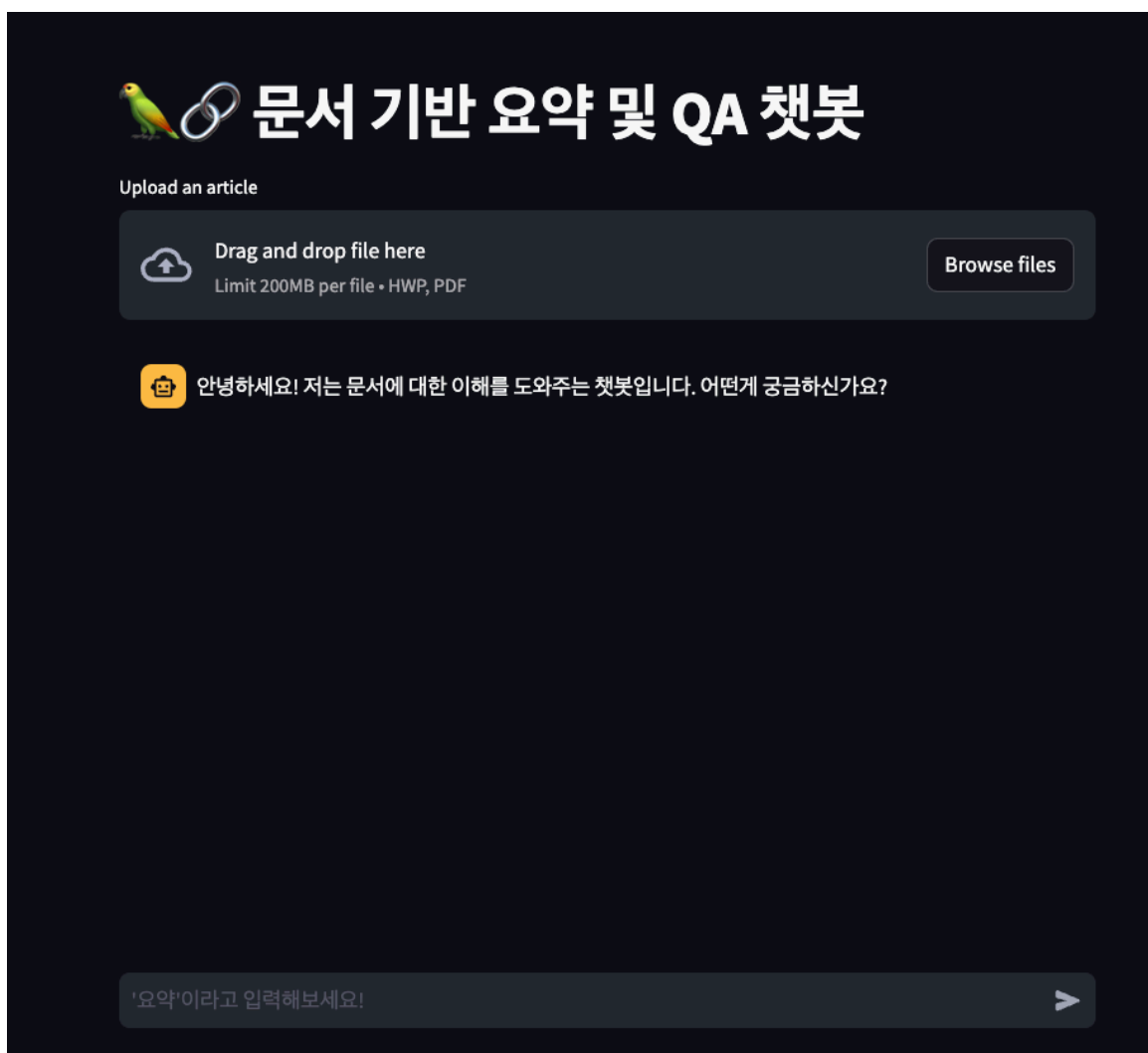
임베딩? 벡터? 데이터베이스? LLM?

이 복잡한 모듈을 어떻게 다 구현해..?

걱정 마세요. 저희에게는 LangChain이 있습니다.

## 개요

- 사람이 살아가는데 필수적인 3가지 요소인 의식주, 그중에 주(집)은 최근 1인 가구가 많아지는데 반해 자주 바뀌는 부동산 관련 법으로 인해 많은 사람들이 혼란스러워 하고, 잘 알지 못하면 손해를 보는 경우가 많습니다.
- LangChain을 이용하여 **대한민국 법제처의 가장 최신 주택임대차보호법 공식 문서**를 학습시키고,
  - (엄밀히 말하면 train은 아니고 retrieval, 즉 오픈북 시험이라고 생각하시면 좋습니다)
- 궁금한 내용에 대해 바로 답변을 받아보는 문서 QA 봇을 만들어봅시다.



## OpenAI API Key Setting

- 소중한 api key를 Python code 안에 Hard coding하면 안된다.

- 따라서 `getpass` 함수를 사용하여 코드를 실행한 후 OpenAI API Key를 입력 칸에 복붙한다.


```
# @title set API key
import os
import getpass

# Get the Upstage API key using getpass
if "OPENAI_API_KEY" not in os.environ or not os.environ["OPENAI_API_KEY"]:
    os.environ["OPENAI_API_KEY"] = getpass.getpass("Enter your OPENAI API

print("API key has been set successfully.")
```

자료 출처: 아래의 링크에서 PDF 형식으로 파일을 다운받습니다.

주택임대차보호법 | 국가법령정보센터 | 법령 > 본문

 [https://www.law.go.kr/LSW/LsiJoLinkP.do?lsNm=주택임대차보호법&paras=1&docType=&languageT](https://www.law.go.kr/LSW/LsiJoLinkP.do?lsNm=주택임대차보호법&paras=1&docType=&languageType=KO#)  
ype=KO#

## 실습 목표

- 위 PDF 파일을 벡터 데이터베이스에 저장하고,
- 우리의 LLM이 이를 바탕으로 우리의 질문에 답변해주는 것.

**LangChain의 주요 개념 : LLM 및 여러 source들을 Chaining 한다.**

- 아까 배웠던
  - loader
  - splitter
  - storage
  - retriever
  - generator(LLM)
- 이 5가지 모듈을 LangChain에서 불러오고, **Chain**으로 연결해주기.

## DocQA 파이프라인 함수

- **input** : file, query, callback(optional, stream으로 답변 출력하기 위한 callback)
- **output** : response

## 모듈 구현

아까 배웠던 모듈들의 input, output을 생각하며 따라해봅시다.

starter.py에서 실습

### Loader

```
# loader
raw_text = get_pdf_text(uploaded_file)
```

### Splitter

- **청크 크기(Chunk Size)**
  - 청크가 포함할 수 있는 최대 문자 수
- **청크 중복(Chunk Overlap)**
  - 인접한 두 청크 간에 겹칠 수 있는 문자 수

```
# splitter
text_splitter = CharacterTextSplitter(
    separator = "\n\n",
    chunk_size = 1000,
    chunk_overlap = 200,
)
all_splits = text_splitter.create_documents([raw_text])

print("총 " + str(len(all_splits)) + "개의 passage")
```

### Storage

```
# storage
vectorstore = FAISS.from_documents(documents=all_splits, embedding=Oper
```

## Retriever

- top-k 개수를 다양하게 실험해보자.

```
# retriever
docs_list = vectorstore.similarity_search(query_text, k=3)
docs = ""
for i, doc in enumerate(docs_list):
    docs += f"문서{i+1}':{doc.page_content}\n"
```

## Generator

- "gpt-4o-mini로 생성해보자.
- rag\_prompt를 다양하게 바꿔서 최적의 prompt를 만들어보자.

```
# generator
llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0, streaming=True)
```

## Chaining

- 각 모듈의 input-output을 고려하여 chaining

```
# chaining
rag_prompt = [
    SystemMessage(
        content="너는 문서에 대해 질의응답을 하는 '문서봇'이야. 주어진 문서를 참고하여
    ),
    HumanMessage(
        content=f"질문:{query_text}\n\n{docs}"
    ),
]

response = llm(rag_prompt)
```

## Streamlit 실행

```
streamlit run starter.py
```

## Step2: 프롬프트 엔지니어링 (Prompt Engineering)

### 모델에 페르소나 및 지시사항 입력

```
# prompt formatting
rag_prompt = [
    SystemMessage(
        content="너는 문서에 대해 질의응답을 하는 '문서봇'이야. 주어진 문서를 참고하
    ),
    HumanMessage(
        content=f"질문:{query_text}\n\n{docs}"
    ),
]
```

- SystemMessage란?
  - 모델에게 페르소나 및 답변의 조건을 정의하는 부분
  - 모델에게 다양한 페르소나 및 조건을 입력해봅시다!
    - 페르소나 프롬프트 예시

"너는 문서에 대해 질의응답을 하는 '문서봇'이야. 주어진 문서를 참고하여 시

- 조건 정의 프롬프트 예시

"너는 문서에 대해 질의응답을 하는 '문서봇'야. 주어진 문서를 참고하여 사용

### 모델의 페르소나 및 조건 변경 실습

내가 원하는 형태로 챗봇의 페르소나와 조건을 변경해보세요!

# 원하는 답변을 얻기위한 질문 규칙

AI에게 원하는 답을 잘 얻기 위해서는 질문하는 방식, 즉 '프롬프트'를 잘 작성해야 한다.

딱 2가지만 기억하자!

## 1. 구체적으로 질문하기

- **의미:** 원하는 답변을 정확하게 받으려면, 질문을 구체적으로 해야 한다. 해매하거나 넓은 범위의 질문은 기대하는 답변을 얻기 어려울 수 있다.
- **예시:** "동물에 관한 정보"보다 "아프리카 사바나의 대표 동물이 뭐야?"라고 구체적으로 물어보는 것이 좋다.

## 2. 다양한 질문을 시도하고 개선하기 (피드백)

- **의미:** 처음 질문이 완벽한 답을 주지 않을 수도 있다. 여러 가지 방식으로 질문을 바꿔보며, 가장 좋은 답변을 얻는 질문 방식을 찾아야 한다.
- **예시:** "사과의 영양성분은?"이라는 질문에 만족스러운 답변을 받지 못했다면, "사과의 주요 영양소와 그 효능은 무엇인가?"로 질문을 바꿔보세요.

## 문서 기반 질문 답변 실습

위 프롬프트 엔지니어링 가이드라인을 따라 원하는 정보를 얻어보자.

원하는 정보를 얻지 못했을 때 어떻게 프롬프트를 어떻게 바꿔야 할지 생각해보자.

### ▼ 질문 실습 예제

- Q1. 임대차가 끝난 후 보증금이 반환되지 아니한 경우에는 어떻게 해야 할까?
  - 정답: 임차인은 임차주택의 소재지를 관할하는 지방법원 · 지방법원지원 또는 시 · 군 법원에 임차권등기명령을 신청할 수 있다.
- Q2. 확정일자를 갖추지 않은 임차인은 어떤 문제가 생기니?
  - 정답: 민사집행법에 따른 경매 또는 국세징수법에 따른 공매를 할 때에 임차주택(대지를 포함한다)의 환가대금에서 후순위권리자나 그 밖의 채권자보다 우선하여 보증금을 변제받을 권리가 없어진다.
- Q3. 확정일자를 받으려면 어떻게 해야 하니?
  - 정답: 확정일자는 주택 소재지의 읍 · 면사무소, 동 주민센터 또는 시(특별시 · 광역시 · 특별자치시는 제외하고, 특별자치도는 포함한다) · 군 · 구(자치구를 말한다)의 출장소, 지방법원 및 그 지원과 등기소 또는 「공증인법」에 따른 공증인(이하 이 조에서 "확정일자부여기관"이라 한다)이 부여한다.

- Q4. 임차인에게 별다른 계약 갱신에 대한 통지를 하지 않으면 어떻게 되니?
  - 정답: 그 기간이 끝난 때에 전 임대차와 동일한 조건으로 다시 임대차한 것으로 본다.
- Q5. 임차인은 계약을 갱신하고 싶지 않으면 임대인에게 갱신거절 통지를 언제까지 해야하니?
  - 정답: 임대차 기간이 끝나기 6개월 전부터 2개월 전까지 기간에 청구
- Q6. 임대인이 계약갱신 요구를 거절할 수 있니?
  - 정답: 제6조에도 불구하고 임대인은 임차인이 제6조제1항 전단의 기간 이내에 계약갱신을 요구할 경우 정당한 사유 없이 거절하지 못한다.
- Q7. 임차인은 계약 갱신요구를 몇 번까지 할 수 있니?
  - 정답: 임차인은 제1항에 따른 계약갱신요구권을 1회에 한하여 행사할 수 있다. 이 경우 갱신되는 임대차의 존속기간은 2년으로 본다.

## Step3: 챗봇 꾸미기 및 고도화

### Streamlit 코드를 통해 UI를 변경 실습

#### 챗봇의 인사말 변경

```
# chatbot greetings
if "messages" not in st.session_state:
    st.session_state["messages"] = [
        ChatMessage(
            role="assistant", content="안녕하세요! 저는 문서에 대한 이해를 도와주는 챗봇입니다."
        )
    ]
```

#### 챗봇 타이틀 변경

```
# page title
st.set_page_config(page_title='🦜🔗 문서 기반 요약 및 QA 챗봇')
st.title('🦜🔗 문서 기반 요약 및 QA 챗봇')
```

#### 특정 입력을 통한 요약하기



```
# 요약이란 문장을 받았을때 generate_summarize 코드 실행
if prompt == "요약":
    response = generate_summarize(st.session_state['raw_text'], stream_handler=st.session_state["messages"]).append(
        ChatMessage(role="assistant", content=response)
    )
```

```
def generate_summarize(raw_text, callback):

    # generator
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0, streaming=True)

    # prompt formatting
    rag_prompt = [
        SystemMessage(
            content="다음 나올 문서를 'Notion style'로 요약해줘. 중요한 내용만."
        ),
        HumanMessage(
            content=raw_text
        ),
    ]

    response = llm(rag_prompt)
    return response.content
```

## 프론트엔드 꾸미기

이전 파일 업로더를

```
uploaded_file = st.file_uploader('Upload an document', type=['pdf'])
```

아래 코드와 같이 사이드바로 옮길 수 있어요 😊

```
st.markdown("""
<style>
.main { background-color: #f5f5f5; }
.sidebar .sidebar-content { background-color: #f0f0f0; }
```

```
.stButton>button { background-color: #4CAF50; color: white; }
</style>
"", unsafe_allow_html=True)

# 파일 업로드
st.sidebar.header('📄 파일 업로드')
uploaded_file = st.sidebar.file_uploader('문서를 업로드하세요', type=['hwp', 'pdf'])

# Reset 버튼 사이드바에 추가
st.sidebar.header("⚙️ 세션 관리")
if st.sidebar.button("Reset Session", key="reset", help="모든 세션을 초기화합니다"):
    st.experimental_rerun()
```

## Step4: 본인만의 챗봇 만들어보기

특정 도메인의 문서가 들어왔을때, 잘 대답할 수 있도록 step1~3에서 배웠던 내용을 적용하여 구축해보세요!

### Step1 체크리스트

- `splitter` 를 이용하여 어느 조건으로 문서를 분할할 것인가?
- `retriever` 에서 문서를 몇개 가져올 것인가?
- `generator` 에서 어떤 모델을 사용하여 *temperature*를 어느정도 수준으로 맞춰야 하는가?

### Step2 체크리스트

- `SystemMessage` 를 변경하여,
  - 내가 구축하고자 하는 챗봇의 페르소나는 어떤 것이 어울릴까?
  - RAG를 이용해 찾은 문서가 적절하지 않을때, 어떻게 대처해야 할까?

### Step3 체크리스트

- (옵션)Streamlit의 기능을 이용해 나만의 챗봇을 꾸며보세요!
- 특정 입력이 들어갔을때, 답변하고자 하는 코드를 작성해보세요!

### Step4 체크리스트 (optional)

- 4주차에 배운 Langchain 모듈을 챗봇에 붙여보세요!