

Personalized ChatBot Service

- SLACK for ME -


김지수 박소망찬 박수영 송지우 안보은

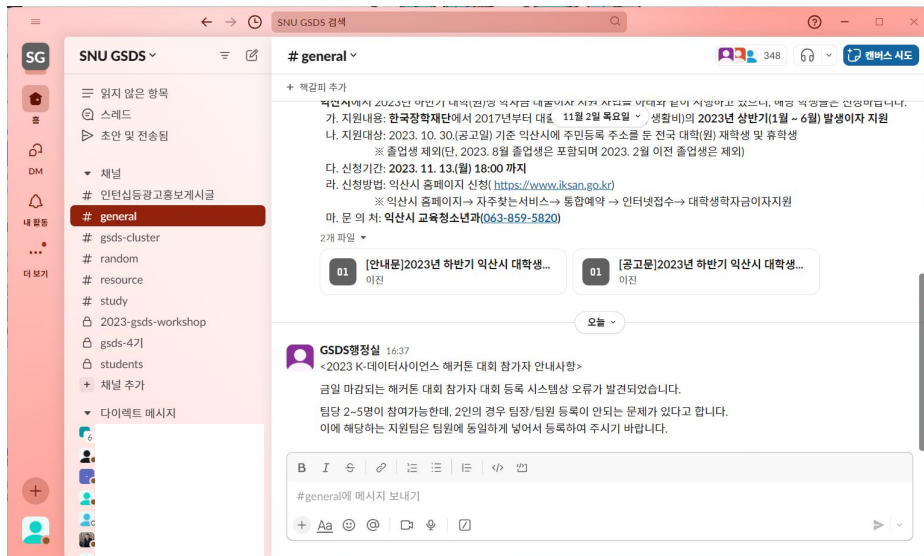


Contents

- Problem Definition
- Research goals, key idea, used methodology
- Experimental data set and analysis
- Related work and novelty
- Future direction

Problem definition

 **Slack** : professional and organizational communications, also as a community platform



- **Increase in Information Sharing:**

- With the rapid surge in information within Slack channels, users are facing difficulties in finding the information they need.

- **Decreased Efficiency and Time Wastage:**

- The time spent by users in the process of finding necessary information has increased, leading to a reduction in overall work efficiency.

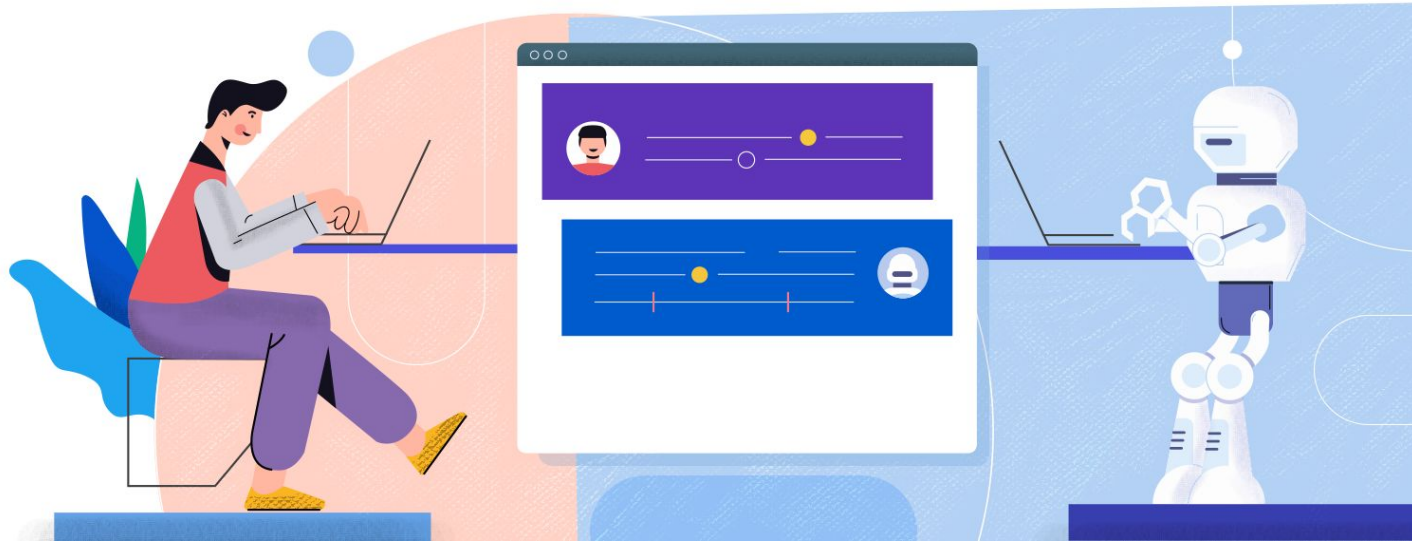
- **Limitations of In-Channel Search:**

- The built-in search feature within Slack channels proves effective when users know specific keywords, but it has limitations in real-time searches for everyday conversations and specific topics.

- **Challenges in Conversation-Centric Communication:**

- Most conversations quickly fade from memory over time, posing challenges for important information retention.

Research goals, key idea, used methodology



LangChain

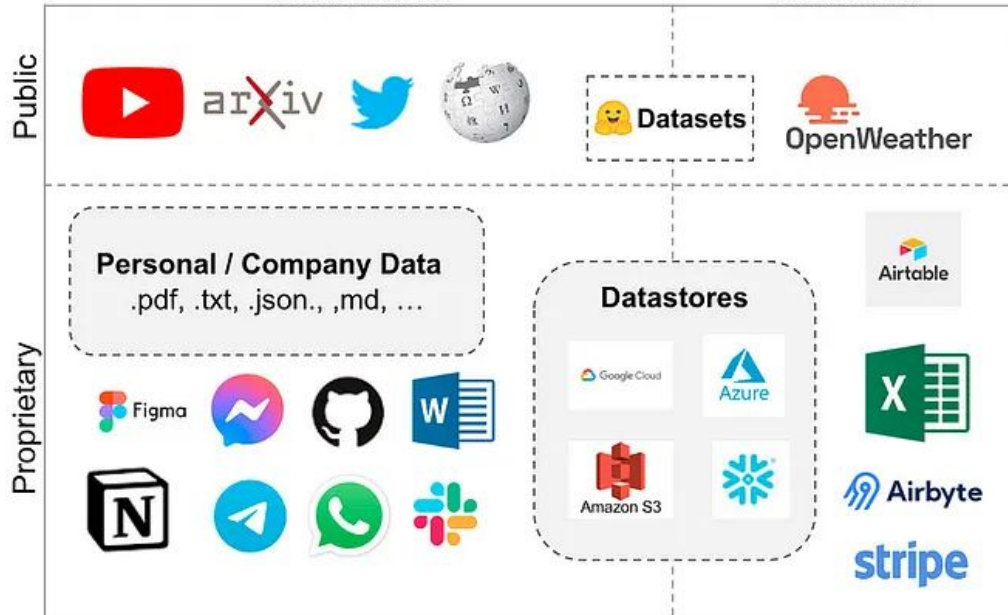
LangChain Data Ecosystem



Data Connectors (> 120 Integrations)

Unstructured

Structured



Vector Storage (> 35 Integrations)

Transformations



Embeddings (> 25 Integrations)



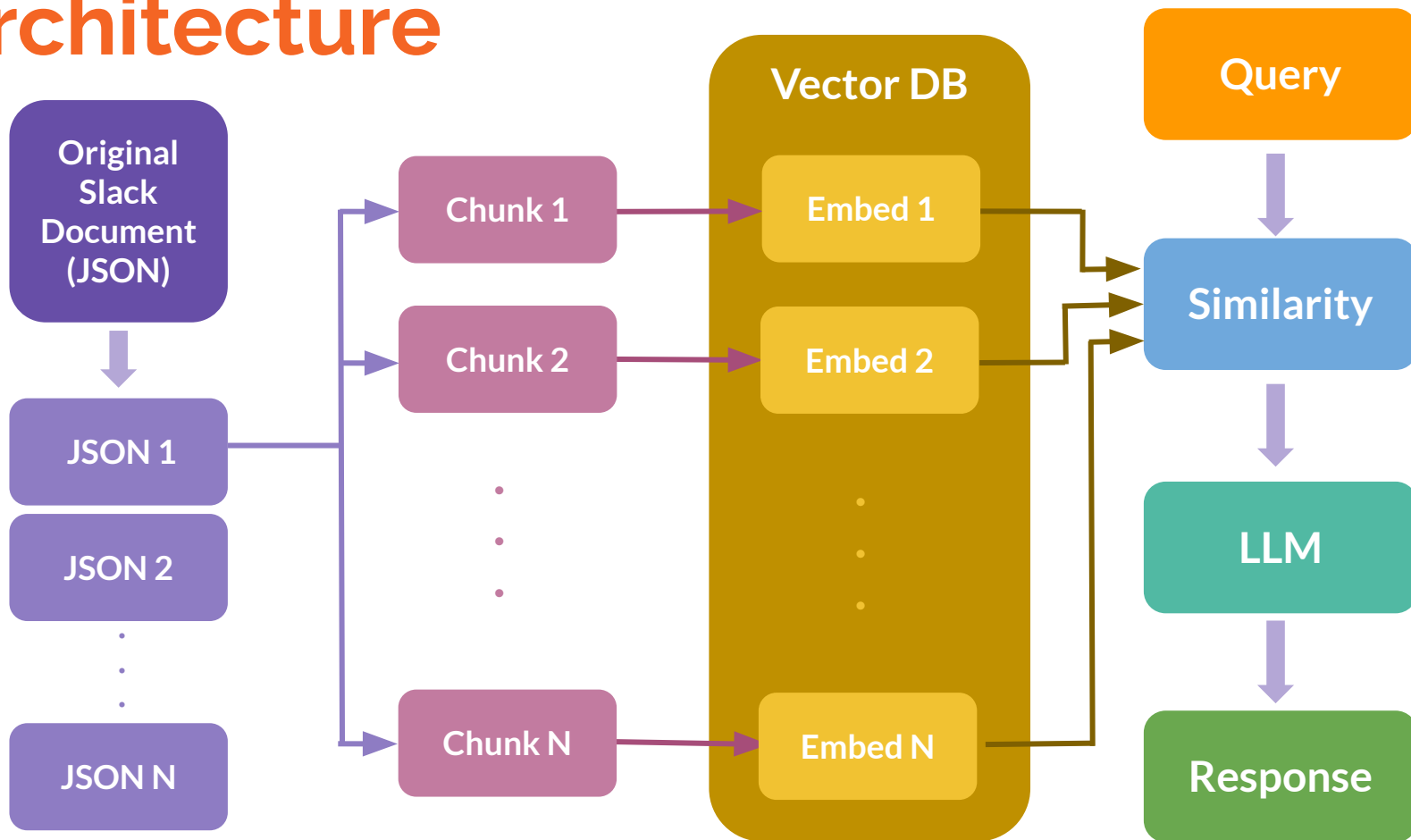
- **Main Goals:**

- **Utilizing RAG:** Implementing a chatbot powered by RAG model
- **Keyword-Based Search:** Retrieving information based on user-entered search terms
- **Similarity-Based Search:** Retrieving Slack content with high similarity

- **Key Features:**

- **Keyword Input:** Users can input search terms for desired information
- **Similarity Search:** Locating information in Slack with high similarity
- **Memory Optimization:** Efficient information retrieval based on similarity without memorizing entire Slack content

Architecture

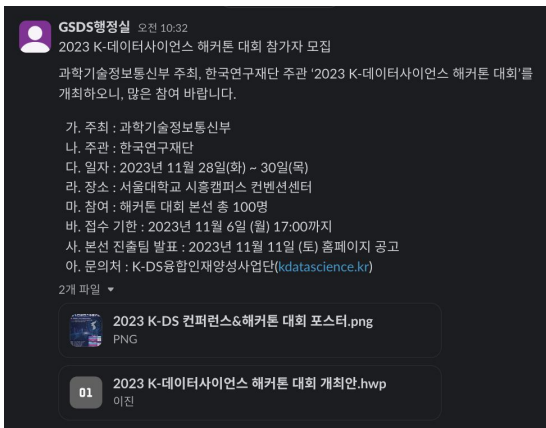


Experimental dataset and analysis

Data crawling

Tool: Slack Dumper

JSON files were generated by creating a slack export of the Slack workspace



Inside JSON files

```
[
  {
    "client_msg_id": "d5eb093a-3717-4f41-bbe9-61d72f9c25d4",
    "type": "message",
    "user": "U01QNJ7PWRX",
    "text": "2023 K-데이터사이언스 해커톤 대회 참가자 모집\n\n과학기술정보통신부 주최, 한국연구재단 주관 '2023 K-데이터사이언스 해커톤 대회'를 개최하오니, 많은 참여 바랍니다.\n\n가. 주최 : 과학기술정보통신부\n나. 주관 : 한국연구재단\n다. 일자 : 2023년 11월 28일(화) ~ 30일(목)\n라. 장소 : 서울대학교 시흥캠퍼스 컨벤션센터\n마. 참여 : 해커톤 대회 본선 총 100명\n바. 접수 기한 : 2023년 11월 6일 (월) 17:00까지\n사. 본선 진출팀 발표 : 2023년 11월 11일 (토) 홈페이지 공고\n아. 문의처 : K-DS융합인재양성사업단(kdatascience.kr)\n\n2개 파일
2023 K-DS 컨퍼런스&해커톤 대회 포스터.png
PNG
01 2023 K-데이터사이언스 해커톤 대회 개최안.hwp
이진
```

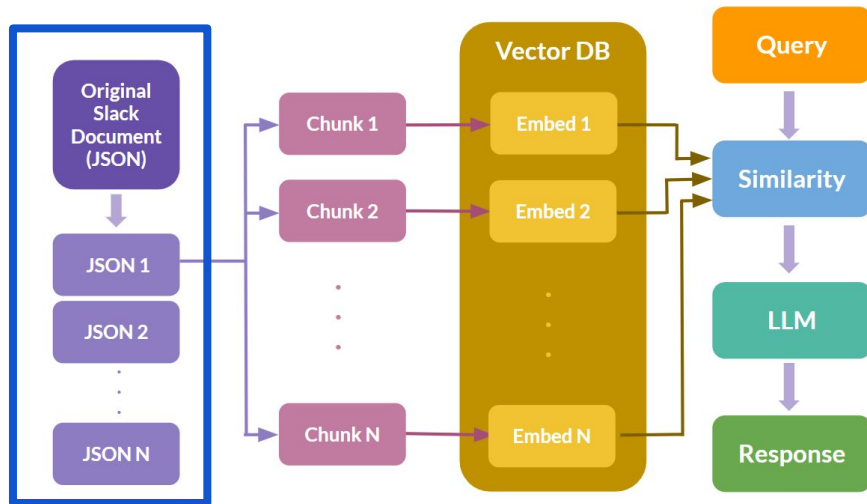
Split of the original Document file

Structure of Slack JSON File:

- A single JSON file generated in Slack comprises multiple texts, encompassing both the main body and comments.

Enhancing Search Accuracy through Separated JSON file for each text:

- To improve search accuracy, texts in a JSON file were separated and stored as an individual JSON file.
- This approach allows for a more granular search, enabling precision in retrieving specific information from the content.

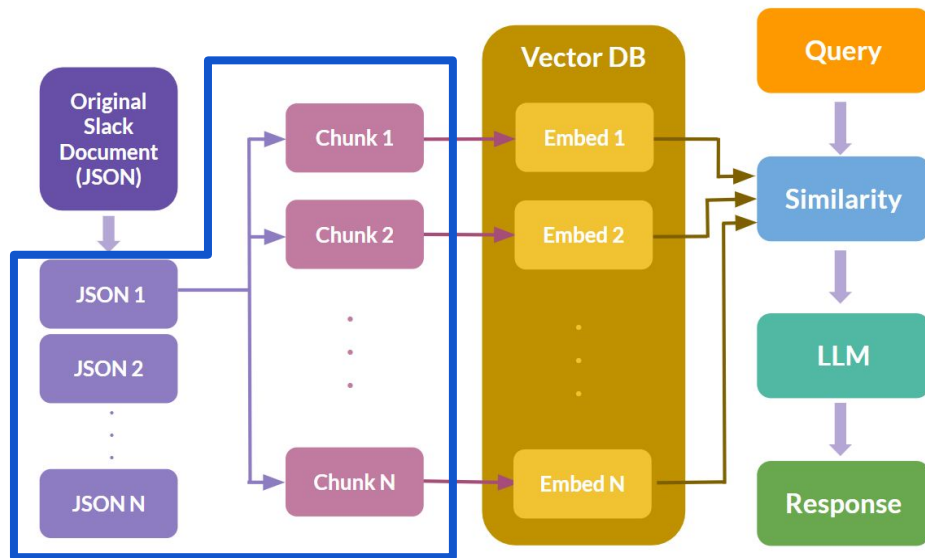


Text Splitting

Various TextSplitter

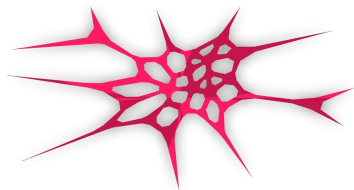
- CharacterTextSplitter
- RecursiveCharacterTextSplitter
- TokenTextSplitter

Chunk Size = 1000



Vector Database: Faiss

FAISS
Scalable Search With Facebook AI



> Entering new RetrievalQA chain...

> Finished chain.

page_content에서는 2021 지식재산 정보 활용 창업 경진대회와 관련된 포스터 게시 요청과 홈페이지 학생소개 게시 사항 조사에 대한 내용이 포함되어 있습니다.

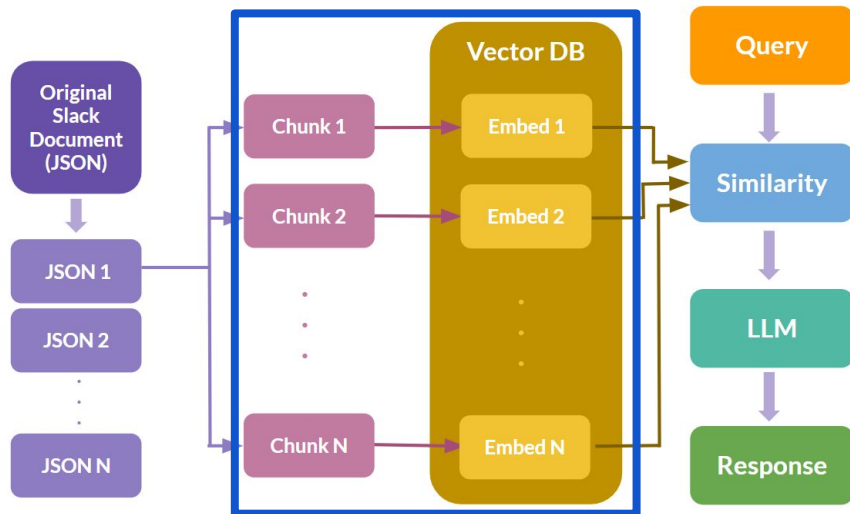
1. 2021 지식재산 정보 활용 창업 경진대회 관련 포스터 게시 요청:

- 보낸 사람: GSDS행정실
- 출처: general 1
- 보낸 날짜: 2021-03-18

2. 홈페이지 학생소개 게시사항 조사:

- 보낸 사람: 27_김진웅
- 출처: students 2021
- 보낸 날짜: 2021-03-22
- 내용: 학생들은 본인 소개항목을 작성하여 홈페이지에 게시해야 합니다. 링크를 통해 작성할 수 있으며, 수정권한이 있으므로 대외보안에 유의해야 합니다. 기한은 03.24일(수요일)이며, 기존에 올라간 내용을 업데이트하고 싶은 경우 5가지 항목 전부를 작성해야 합니다. 이미 작성된 항목에 대해서도 수정이 필요한지 확인해야 합니다.

자세한 대회 관련 공지사항은 해당 포스터 게시 요청과 홈페이지 학생소개 게시사항 조사 내용을 참고하시면 됩니다.



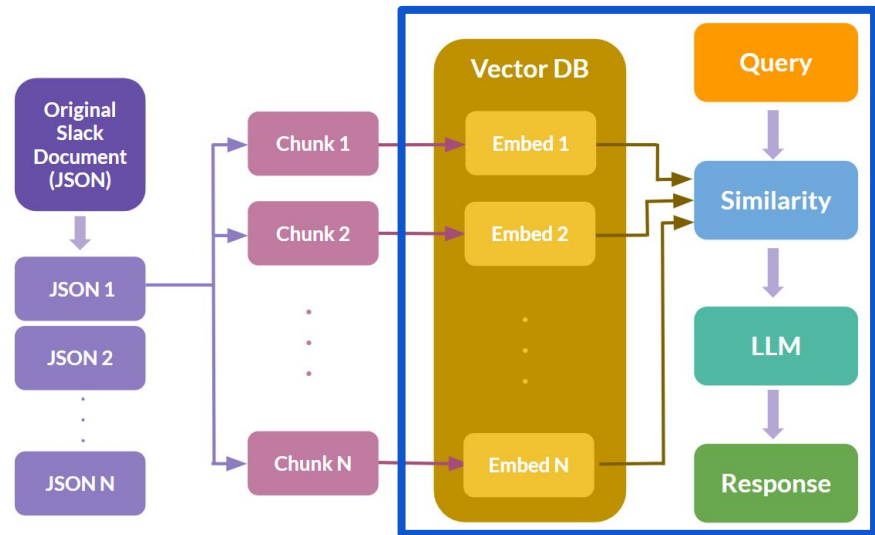
Search and data Retrieval: LangChain

```
template = """
{query} page_content에서 필요한 정보를 찾으세요.
metadata 정보를 가지고 보낸 사람과 출처를 이용하세요.
관련 정보를 자세하게 알려주세요.
"""

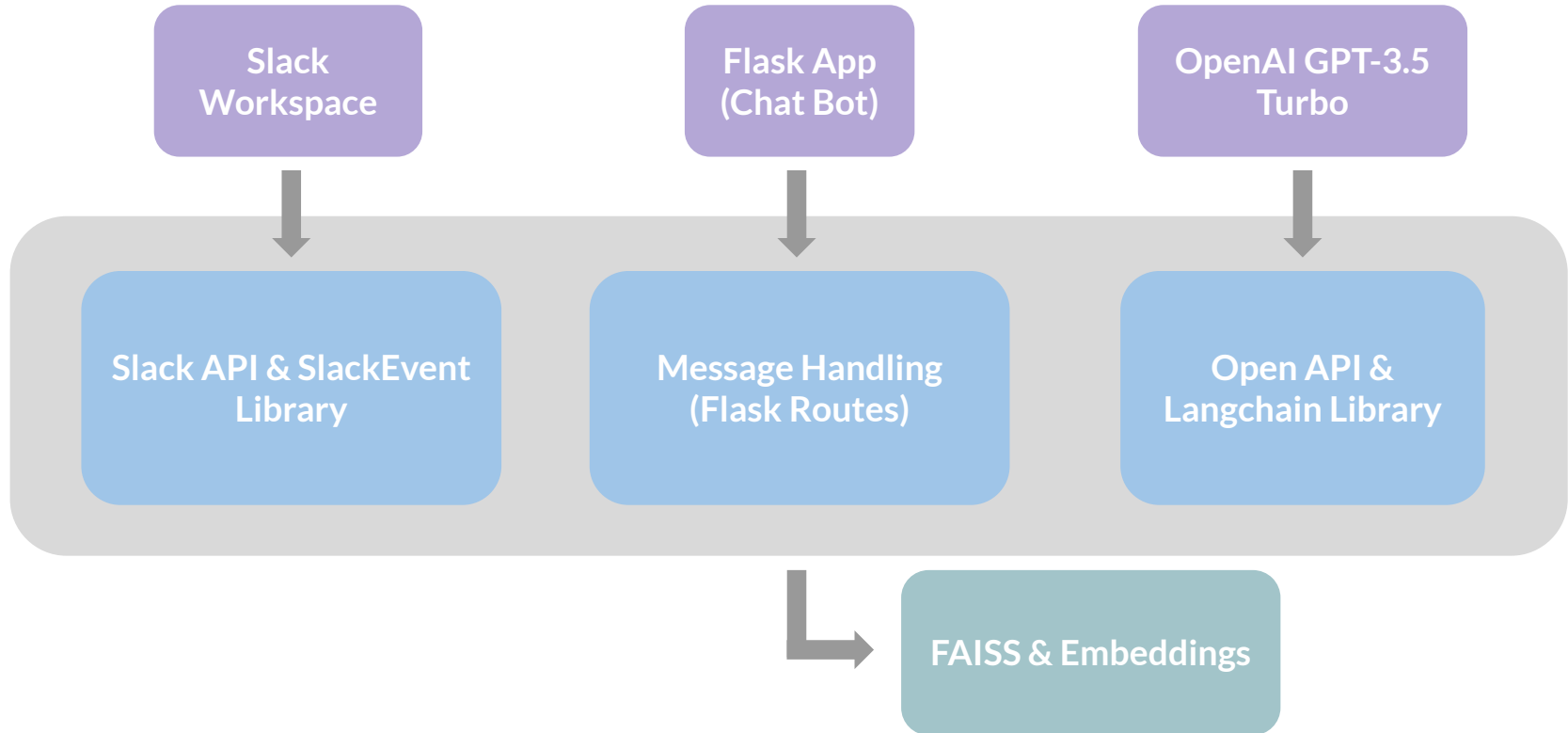
prompt = PromptTemplate(
    input_variables=["query"],
    template = template,
)

chatbot_chain = RetrievalQA.from_chain_type(
    llm = ChatOpenAI(
        temperature = 0.3, model_name = 'gpt-3.5-turbo', max_tokens = 2000
    ),
    retriever = db.as_retriever(search_kwargs={"k" : 2}),
    verbose=True,
    chain_type = "stuff",
    chain_type_kwargs={
        'document_prompt': PromptTemplate(
            input_variables=["page_content", "sender_name", "source", "Date"],
            template="내용:\n{page_content}\n보낸 사람:{sender_name}\n출처:{source}\n보낸 날짜:{date}"
        ),
    },
)

# print(chatbot_chain.run(prompt.format(query = "대학교 관련 공지사항이 무엇이 있나요?")))
result = chatbot_chain.run(prompt.format(query = "대학 관련 공지사항이 무엇이 있나요?"))
print(result)
```



User Interface: In channel Chatbot



User Interface: In channel Chatbot

- Chatbot Configuration and Template Generation
- Slack Event Message Handler Definition
- Upon Receiving a Slack Message
- Flask App Execution

```
SLACK_TOKEN="████████████████████████████████████████████████████████████████████████████████"
SIGNING_SECRET="████████████████████████████████████████████████████████████████████████████████"

app = Flask(__name__)
slack_event_adapter = SlackEventAdapter(SIGNING_SECRET, '/slack/events', app)

client = slack.WebClient(token=SLACK_TOKEN)
os.environ["OPENAI_API_KEY"] = "████████████████████████████████████████████████████████████████████████████████"
index_base_dir = './faiss_index'

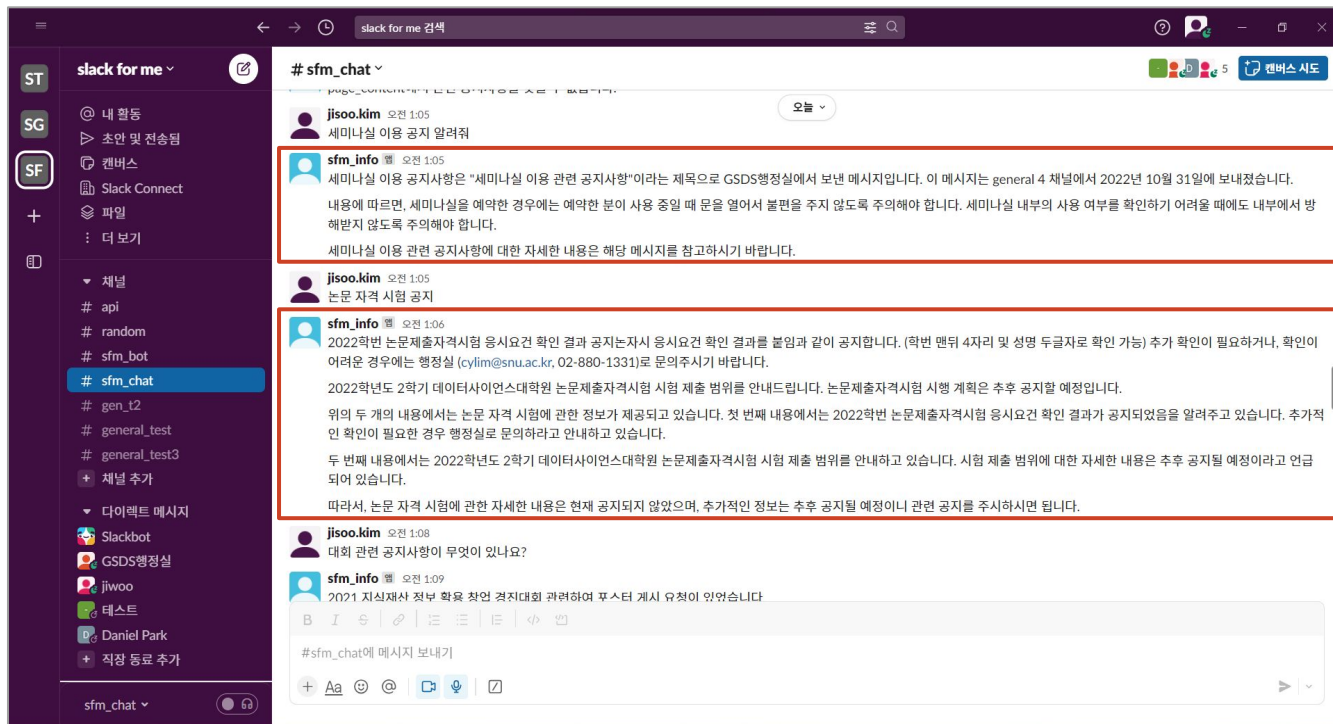
embeddings = OpenAIEmbeddings(model = 'text-embedding-ada-002' )
db = FAISS.load_local(index_base_dir, embeddings)
```

```
@slack_event_adapter.on('message')
def message(payload):
    print(payload)
    event = payload.get('event', {})
    channel_id = event.get('channel')
    user_id = event.get('user')
    text = event.get('text')

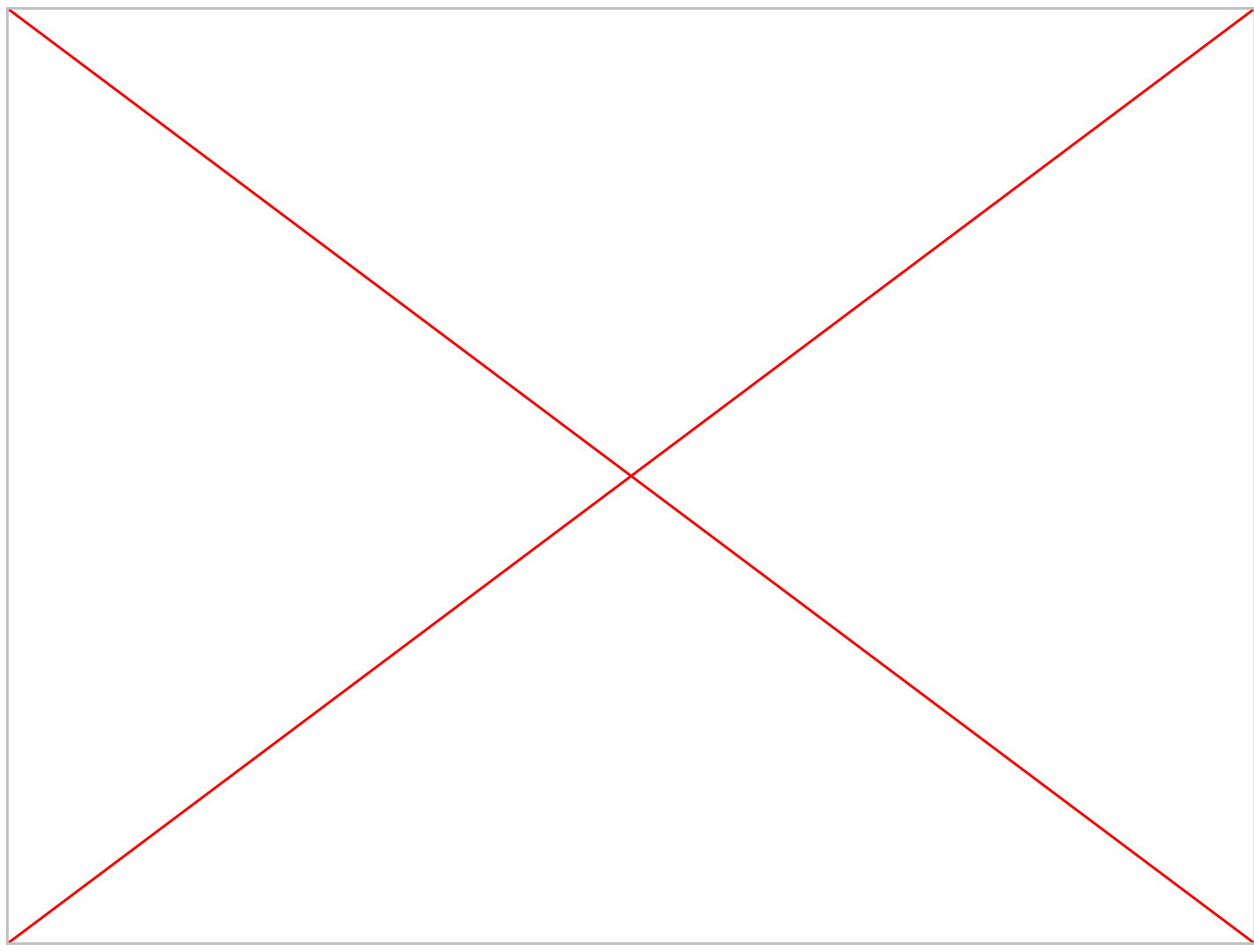
    # Check if the user has asked a new question and it is not from the bot itself
    if text and text != answer_dict.get(user_id) and event.get("subtype") is None and user_id != "U06AP1Y9RU1" and 'bot_id' not in event:

        answer_dict[user_id] = text # Update the last question for the user
        print('ans')
        # Rest of your logic for processing the question and generating an answer
        answer = chatbot_chain.run(prompt.format(query=text))
        print("!!",user_id,text)
        print(">>",answer)
        client.chat_postMessage(channel=channel_id, text=answer)
    else:
        print("Ignoring subsequent messages from the same user or bot.")
```


User Interface: In channel Chatbot



Demo



Related work:

- The Langchain official site presents various possible uses of Langchain in chatbot services.

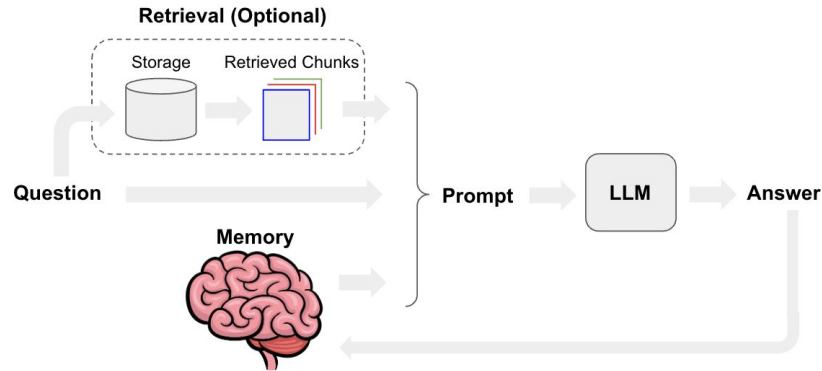


image from - https://python.langchain.com/docs/use_cases/chatbots

Novelty:

- The First attempt to create a GSDS slack chatbot, which increases utilization in relation to the actual demand of GSDS graduate students
- Data processing pipeline which allows the LLM to access metadata such as post authors

Future direction

limitations to be resolved:

- Problems in data processing
- Similarity search setting (due to token limitation)
- Temperature fine tuning
- Incorrect information
 - hallucination

Role by team member

김지수	Chatbot interface develop
박수영	Data crawling
박소망찬	Develop retrieval
송지우	Data preprocessing, Develop retrieval
안보은	Prompt engineering

**Thank you
for your attention!**

Any Questions?