Database Project (Fall 2024)

Homework #10 (50pts, Due date: Dec 10)

Student ID: _____2018310773____

Student Name: _____임승재____

Instruction: In this homework, we provide a jupyter notebook file (DBP_Homework10.ipynb). You should follow the instructions in these documents. Only the provided code's '*EDIT HERE*' sections must be edited. You must *DELETE* your *GROQ_API_KEY* before submitting.

Submission Guide: Submit two files as follows:
-DBP_Homwork10_StudentID.zip
- DBP_Homwork10_StudentID.ipynb
- DBP_Homwork10_StudentID.pdf

1. [20pts] You want to provide a Question Answering service based on Retrieval-Augmented Generation (RAG) for a given webpage. To raise the satisfaction of users, you have to do prompt engineering. Prompt engineering is to make an optimal prompt for a given task.

So, you should implement the RAG pipeline as given parameters.

Parameters
Splitter: RecursiveCharacterTextSplitter Chunk size for splitter: 800 Chunk overlap size for splitter: 100 Vector store: Chromadb Embedding for vector store: HuggingFaceEmbeddings (model_name: intfloat/multilingual-e5-small) Retriever: Same as vector store Retriever search type: similarity Retriever search kwargs k: 6 LLM: {model: lamma-3.1-8b-instant, api_key: }

Next, you should do prompt engineering for the given query string *"What is Task Decomposition?"*. **You should apply three prompts to the RAG pipeline: two prompts from Langchain hub (https://smith.langchain.com/hub) and one prompt on your own.**

a. Langchian hub - rlm/rag-prompt

[Answer]
Enter your code and result here. You must show your result (captured image or string).

Code:

```
# a.   Langchian hub - rlm/rag-prompt
# ============== EDIT HERE ==================
prompt = hub.pull("rlm/rag-prompt")
# ==========================================
print("## Prompt template ##")
print("Input variables: ", prompt.input_variables)
print("Template: ", prompt.messages[0].prompt.template)
print()

# ============== EDIT HERE ==================
rag_chain = (
      {"context": retriever | format_docs,
       "question": RunnablePassthrough()}
      | prompt
      | llm
      | StrOutputParser()
)
# ==========================================

print("## Generated string ##")
for chunk in rag_chain.stream(query):
      print(chunk, end="", flush=True)
```

Result:

```
/usr/local/lib/python3.10/dist-packages/langsmith/client.py:261:
LangSmithMissingAPIKeyWarning: API key must be provided when using hosted
LangSmith API
    warnings.warn(
## Prompt template ##
Input variables:    ['context', 'question']
Template:    You are an assistant for question-answering tasks. Use the following pieces of
retrieved context to answer the question. If you don't know the answer, just say that you don't
know. Use three sentences maximum and keep the answer concise.
Question: {question}
Context: {context}
Answer:

## Generated string ##
Task Decomposition is the process of breaking down a complicated task into smaller,
manageable subgoals. This is achieved by instructing the model to "think step by step" or by
using techniques like Chain of Thought (CoT) and Tree of Thoughts. It enables efficient
handling of complex tasks and improves the quality of final results.
```

b. Langchain hub – gregkamradt/test-question-making

[Answer]
Enter your code and result here. You must show your result (captured image or string).

[5 pts]
Code:
```
# b.   Langchain hub – gregkamradt/test-question-making
# ============== EDIT HERE ===================
prompt = hub.pull("gregkamradt/test-question-making")
# ===========================================
print("## Prompt template ##")
print("Input variables: ", prompt.input_variables)
print("Template: ", prompt.template)
print()

# ============== EDIT HERE ==================
rag_chain = (
    {"context": retriever | format_docs,
     "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
# ===========================================

print("## Generated string ##")
for chunk in rag_chain.stream(query):
    print(chunk, end="", flush=True)
```

Result:
```
/usr/local/lib/python3.10/dist-packages/langsmith/client.py:261:
LangSmithMissingAPIKeyWarning: API key must be provided when using hosted
LangSmith API
  warnings.warn(
## Prompt template ##
Input variables:   ['context', 'question']
Template:   Your goal is to create a well crafted set of answers for a test for a specific
question.
Your answers will be used on a test to access a students knowledge. You will be given the
question and your goal is to follow the output format below w/ guidelines

Answer Choice Guidelines
Position the correct option so that it appears approximately the same number of times in each
possible position for a quiz.
Answer choices should be written clearly and similarly to each other in content, length, and
grammar; avoid giving clues through the use of faulty grammatical construction.
Make all distractors plausible; they should be common misconceptions that learners may
have.
```

In answer choices, avoid “all of the above” and “none of the above," which can lead to artificially higher levels of performance.
In answer choices, avoid references to answer choices by letter (ie: “Both A and B”), as our answers are randomized
When numeric options are used, the options should be listed in numeric order and in a single format (i.e., as terms or ranges).
Rationale Guidelines
All rationales should begin with “Correct.” or “Incorrect.”
All answer options (including correct answer(s) and distractor(s)) must have their own rationale.
Rationales should be unique for each answer option when appropriate. Rationales for distractors should ideally point out a learner’s error in understanding and provide context to help them go back and figure out where they went wrong.
Rationales should not refer to the answer by letter (ie, “option A is incorrect because…”) because answer options will be randomized in our system.
Rationales for Distractors should not give away the correct answer to the question.
Formative Quiz questions (which occur after each module), should include a sentence at the end of each rationale that points the learner back to the relevant video to review the information. e.g., “See “Why Data Governance.”
Summative Quiz questions (which occur at the end of the course), should include a sentence at the end of each rationale that points the learner back to the relevant module to review the information. E.g., “See Module 1: What is Data Governance?”. They should also include the module’s learning objective at the end of each rationale.

Example of a Quiz Question Submission
Below are examples for each component of a multiple-choice question item.

Stem Example:
A company is storing an access key (access key ID and secret access key) in a text file on a custom AMI. The company uses the access key to access DynamoDB tables from instances created from the AMI. The security team has mandated a more secure solution. Which solution will meet the security team’s mandate?

Answer Choices (Distractors A-C and Correct Answer D) Example:
A. Put the access key in an S3 bucket, and retrieve the access key on boot from the instance.
B. Pass the access key to the instances through instance user data.
C. Obtain the access key from a key server launched in a private subnet.
D. Create an IAM role with permissions to access the table, and launch all instances with the new role. (correct)

Rationale Example:

```
[Formative] Incorrect. Data governance is not something specific to big data technologies.
See “Why Data Governance.”

[Summative] Incorrect. It is not relevant to clarify the size of the big data team. Learning
Objective: Apply Hadoop and use new tools to manage and control without compromising the
platform’s basic value. Review Module 1: “What is Data Governance?”

{context}
Question: {question}
YOUR ANSWER:

## Generated string ##
Stem:
Task decomposition is a crucial component of planning in autonomous agent systems. It
involves breaking down complex tasks into smaller, manageable subgoals to enable efficient
handling of complex tasks. What is task decomposition?

Answer Choices:
A. Task decomposition is a process of generating multiple possible solutions to a problem,
without considering the feasibility of each solution.
B. Task decomposition is a technique used to transform big tasks into multiple manageable
tasks by utilizing more test-time computation.
C. Task decomposition is a method of planning that involves creating a tree structure of
possible actions and their consequences.
D. Task decomposition is a process of breaking down complex tasks into smaller, manageable
subgoals, enabling efficient handling of complex tasks.

Rationale:
[Formative] Correct. Task decomposition is a crucial component of planning in autonomous
agent systems, enabling efficient handling of complex tasks. See "Task Decomposition" in the
provided text.
[Summative] Correct. Task decomposition is a process of breaking down complex tasks into
smaller, manageable subgoals, enabling efficient handling of complex tasks. Learning
Objective: Explain the role of task decomposition in planning. Review Module 1: "Planning
and Task Decomposition."
```

c. Feel free to create a new prompt to help your QA service.

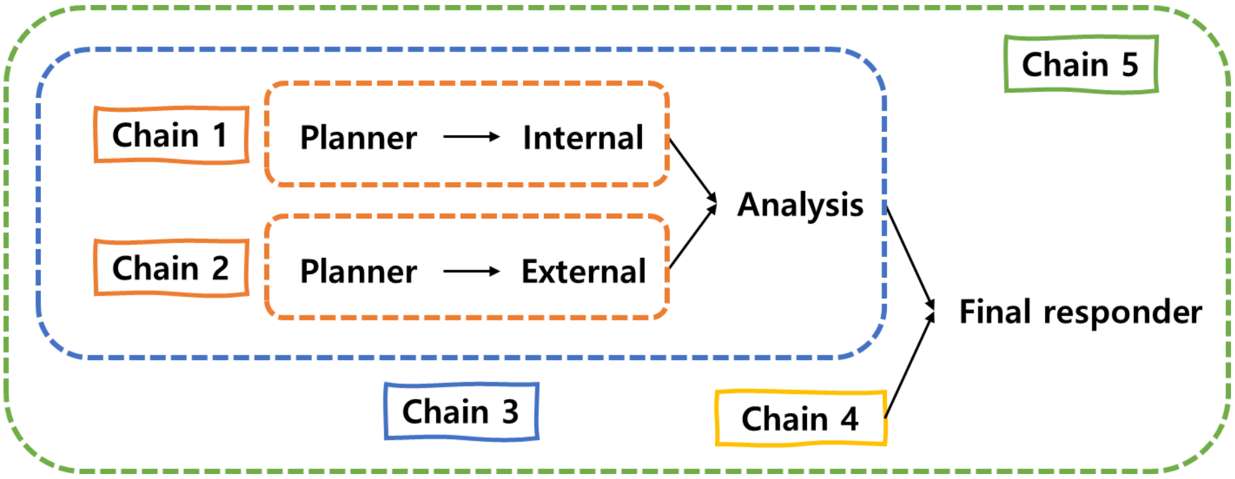
[Answer]
Enter your code and result here. You must show your result (captured image or string).

Code:
```
# c.   Feel free to create a new prompt to improve your QA service.
# ============== EDIT HERE ===================
from langchain.prompts import PromptTemplate
prompt = PromptTemplate(
    input_variables=["context","question"],
    template="You are an expert in computer science concepts. Using the following
context:\n{context}\nAnswer the question concisely: {question}"
)
# ============================================
print("## Prompt template ##")
print("Input variables: ", prompt.input_variables)
print("Template: ", prompt.template)
print()

# ============== EDIT HERE ===================
rag_chain = (
    {"context": retriever | format_docs,
     "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
# ============================================

print("## Generated string ##")
for chunk in rag_chain.stream(query):
    print(chunk, end="", flush=True)
```

Result:
```
## Prompt template ##
Input variables:    ['context', 'question']
Template:    You are an expert in computer science concepts. Using the following
context:
{context}
Answer the question concisely: {question}

## Generated string ##
Task Decomposition is the process of breaking down a complex task into smaller,
manageable subgoals or steps. This enables efficient handling of complex tasks by
transforming them into multiple simpler tasks. It involves analyzing the task, identifying
its components, and creating a plan to achieve the desired outcome. Task Decomposition
can be achieved through various methods, including:
```

1. Chain of Thought (CoT): A prompting technique that instructs the model to "think step by step" to decompose hard tasks into smaller and simpler steps.
2. Tree of Thoughts: An extension of CoT that explores multiple reasoning possibilities at each step, creating a tree structure.
3. Task-specific instructions: Using instructions specific to the task, such as "Write a story outline" for writing a novel.
4. Human inputs: Using human inputs to guide the task decomposition process.

2. [30pts] Implement a RAG multiple chain pipeline. The purpose of this pipeline is to find information about stakeholders for a given project. **The pipeline contains five chains.** The details of the chains are shown in the below figure.



The implementation parameters are same as question 1 in `text_splitter`, `vectorstore`, `retriever`, `llm` variables. The prompts for `planner`, `internal` and `external` are in the given ipynb file. You can use `from_tempate()` method for them. But you have to use `from_messages()` method for `analysis` and `final_responder`. The specific messages are shown in the table below.

analysis	("system", "Generate a stakeholder analysis map for a given project."), ("human", "The details of a given project is following: {result3}"), ("ai", "Internal stakeholders:\n{result1}\n\nExternal stakeholders:\n{result2}")
final_responder	("system", "Generate a final response given the information."), ("ai", "{a_response}"), ("human", "Common pitfalls:\n{a_pitfalls}")

Read the given ipynb file carefully and write all your codes and results here.
[Answer]
Enter your code and result here. You must show your result (captured image or string).

Code:
```
# Load the document
loader = WebBaseLoader(web_paths=("https://asana.com/resources/project-stakeholder",))
raw_documents = loader.load()

# Same with question 1
# ============== EDIT HERE ==================
# Split the document into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=800, chunk_overlap=100)
splits = text_splitter.split_documents(raw_documents)

# Embed each chunks and load it into vector store
vectorstore = Chroma.from_documents(
    documents = splits,
    embedding=HuggingFaceEmbeddings(
        model_name="intfloat/multilingual-e5-small",
        model_kwargs={'device':'cuda'})
    )

# Retrieve
retriever = vectorstore.as_retriever(search_type = "similarity", search_kwargs=dict(k=6))

# Generate
llm = ChatGroq(
    model = "llama-3.1-8b-instant",
    temperature=0,
    api_key=GROQ_API_KEY,
)
# ==========================================

def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)

planner_prompt_template = "I want to do the project management for {input}"
planner_prompt = ChatPromptTemplate.from_template(planner_prompt_template)

# Planner
planner = (
    {"input": itemgetter("input")}
    # ============== EDIT HERE ==================
    | planner_prompt
    | llm
    | StrOutputParser()
    # ==========================================
)

internal_prompt_template = """Answer the question based only on the following context:
{context}

Question: {question}

Answer in the following: {base}
"""
internal_prompt = ChatPromptTemplate.from_template(internal_prompt_template)

internal = (
    {"question": itemgetter("question1"),
     "context": itemgetter("question1") | retriever | format_docs,
     "base": itemgetter("base_response")}
    | internal_prompt
    | llm
    | StrOutputParser()
)

external_prompt_template = """Answer the question based only on the following context:
{context}

Question: {question}

Answer in the following: {base}
"""
external_prompt = ChatPromptTemplate.from_template(internal_prompt_template)

# ============== EDIT HERE ==================
# External, it is similar to `internal`
# Use `question`, `retriever`, `format_docs` for `context`
external = (
    {"question": itemgetter("question2"),
     "context": itemgetter("question2") | retriever | format_docs,
     "base": itemgetter("base_response")}
    | external_prompt
    | llm
    | StrOutputParser()
)
# ==========================================

# Chain 1
chain1 = (
    {"base_response": planner,
     "question1": itemgetter("question1")}
    | internal
)

# You can check that you implement `chain1`, `internal` correctly using this cell
chain1.invoke({"input": "scrum", "question1": "List the internal stakeholders."})

# ============== EDIT HERE ==================
# Chain 2
chain2 = (
    {"base_response": planner,
     "question2": itemgetter("question2")}
    | external
)
# ==========================================

# You can check that you implement `chain2`, `external` correctly using this cell
chain2.invoke({"input": "scrum", "question2": "List the external stakeholders."})

# You can check that you implement `planner` correctly using this cell
planner.invoke({"input": "scrum"})

from langchain.prompts import (
    AIMessagePromptTemplate,
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate
)
analysis = (
    ChatPromptTemplate.from_messages(
        messages=[
            # ============== EDIT HERE ==================
            SystemMessagePromptTemplate.from_template("Generate a stakeholder analysis map for a given
project."),
            HumanMessagePromptTemplate.from_template("The details of a given project is following:
{result3}"),
            AIMessagePromptTemplate.from_template("Internal stakeholders:\n{result1}\n\nExternal
stakeholders:\n{result2}")
            # ==========================================
        ]
    )
    | llm
    | StrOutputParser()
)

# ============== EDIT HERE ==================
# Chain 3
# For `result3`, use the following line:
# "result3": itemgetter("question3") | retriever | format_docs

chain3 = (
    {
        "base_response": planner,
        "question1": itemgetter("question1"),
        "question2": itemgetter("question2"),
        "question3": itemgetter("question3"),
        "result1": chain1,
        "result2": chain2,
        "result3": itemgetter("question3") | retriever | format_docs
    }
    | analysis
)

# ==========================================

# You can check that you implement `chain3`, `analysis` correctly using this cell
chain3.invoke({
    "input": "scrum",
    "question1": "List the internal stakeholders.",
    "question2": "List the external stakeholders.",
    "question3": "How to do a stockholder analysis?"
    })

final_responder = (
    ChatPromptTemplate.from_messages(
        messages=[
            # ============== EDIT HERE ==================
            SystemMessagePromptTemplate.from_template("Generate a final response given the information."),
            AIMessagePromptTemplate.from_template("{a_response}"),
            HumanMessagePromptTemplate.from_template("Common pitfalls:\n{a_pitfalls}"),
            # ==========================================
        ]
    )
    | llm
    | StrOutputParser()
)

# Chain 4 (Don't modify)
chain4 = (
    {"question": itemgetter("question4"),
     "context": itemgetter("question4") | retriever | format_docs}
    | ChatPromptTemplate.from_template("{context} {question}")
    | llm
    | StrOutputParser()
)

chain4.invoke({"question4": "List the common stakeholder mapping pitfalls."})

# ============== EDIT HERE ==================
# Chain 5
chain5 = (
    {"base_response": planner,
     "question1": itemgetter("question1"),
     "question2": itemgetter("question2"),
     "question3": itemgetter("question3"),
     "question4": itemgetter("question4"),
     "a_response": chain3,
     "a_pitfalls": chain4}
    | final_responder
)
# ==========================================

# You can check that you implement `chain5`, `final_responder` correctly using this cell
chain5.invoke({
    "input": "scrum",
    "question1": "List the internal stakeholders.",
    "question2": "List the external stakeholders.",
    "question3": "How to do a stockholder analysis map?",
    "question4": "List the common stakeholder mapping pitfalls."
    })
```

Results:
Chain1.invoke:

```
Based on the provided context, here are the internal stakeholders:

1. Project manager
2. Project team members
3. Project portfolio manager and/or program manager
4. Project sponsor (if you have one)
5. Executive leaders
6. Other cross-functional internal teams
```

Chain2.invoke:

```
The external stakeholders are:

1. Customers
2. Contractors
3. Subcontractors
4. Users
5. Investors
```

```
6. Suppliers

Chain3.invoke:

7. Partners
8. Regulators
9. Competitors
10. Community members

Power/Interest Matrix:
To categorize stakeholders based on their power and interest, we can use the following matrix:

| Power | Interest | Stakeholder Type |
| --- | --- | --- |
| High | High | Critical (e.g., project sponsor, executive leaders) |
| High | Low | Influencers (e.g., regulators, investors) |
| Low | High | Interested (e.g., customers, users) |
| Low | Low | Unconcerned (e.g., competitors, community members) |

Stakeholder Analysis Map:
Here's a sample stakeholder analysis map:

**Critical Stakeholders**

* Project sponsor
* Executive leaders
* Project manager
* Project team members

**Influencers**

* Regulators
* Investors
* Partners

**Interested Stakeholders**

* Customers
* Users
* Contractors
* Subcontractors

**Unconcerned Stakeholders**

* Competitors
* Community members

**Stakeholder Engagement Strategies**

* Critical stakeholders: Regular updates, progress reports, and meetings
* Influencers: Informal communication, occasional updates, and meetings
* Interested stakeholders: Regular updates, progress reports, and occasional meetings
* Unconcerned stakeholders: Minimal communication, occasional updates

**Stakeholder Expectations**

* Critical stakeholders: Project deliverables, timelines, and budget
```

* Influencers: Project impact, outcomes, and potential risks
* Interested stakeholders: Project updates, progress, and potential benefits
* Unconcerned stakeholders: Minimal expectations

This stakeholder analysis map provides a starting point for managing stakeholder expectations and engagement throughout the project lifecycle.

Chain4.invoke:
The text discusses the importance of stakeholder analysis in project management and provides steps to create an effective stakeholder analysis map. Here are the common pitfalls to avoid:

1. **Lack of stakeholder boundaries**: Overeager project stakeholders can cause scope creep. Solution: Implement a change control process.
2. **Bringing stakeholders in too late**: Starting the stakeholder analysis after the project kickoff. Solution: Create a stakeholder analysis map up front.
3. **Not identifying all relevant stakeholders**: Failing to account for both internal and external stakeholders. Solution: Ask yourself who cares about the project, who will it impact, who can influence it, and who can approve or reject it.
4. **Not understanding stakeholder needs**: Failing to listen to and understand the needs and perspectives of stakeholders. Solution: Prioritize communication and buy-in efforts based on stakeholder influence and interest levels.

To create an effective stakeholder analysis map, follow these four steps:

1. **Identify all relevant stakeholders**: Ask yourself who cares about the project, who will it impact, who can influence it, and who can approve or reject it.
2. **Analyze stakeholder influence-interest levels**: Use a stakeholder analysis matrix to categorize stakeholders based on their level of influence and interest in the project.
3. **Understand stakeholder needs**: Listen to and understand the needs and perspectives of stakeholders.
4. **Prioritize communication and engagement**: Prioritize communication and buy-in efforts based on stakeholder influence and interest levels.

Additionally, consider using stakeholder mapping techniques such as:

* Stakeholder mapping: Identifying stakeholders and the impact they might have on the project based on two key aspects: stakeholder impact and stakeholder interest.
* Stakeholder analysis matrix: Categorizing stakeholders based on their level of influence and interest in the project.
* RACI chart: Tracking who everyone is, why they matter, and what their impact on the project will be.

Chain5.invoke:
Common Pitfalls to Avoid in Stakeholder Analysis

Effective stakeholder analysis is crucial in project management to ensure that all stakeholders are engaged, informed, and aligned with the project's objectives. However, there are common pitfalls to avoid when conducting stakeholder analysis. Here are some of the most common mistakes and their solutions:

1. **Lack of stakeholder boundaries**: Overeager project stakeholders can cause scope creep.
   - **Solution**: Implement a change control process to manage stakeholder requests and ensure that the project scope remains focused.
2. **Bringing stakeholders in too late**: Starting the stakeholder analysis after the project kickoff.
   - **Solution**: Create a stakeholder analysis map up front to ensure that all stakeholders are identified and engaged from the beginning of the project.
3. **Not identifying all relevant stakeholders**: Failing to account for both internal and external stakeholders.
   - **Solution**: Ask yourself who cares about the project, who will it impact, who can influence it, and who can approve or reject it. This will help you identify all relevant stakeholders.

4. **Not understanding stakeholder needs**: Failing to listen to and understand the needs and perspectives of stakeholders.
   - **Solution**: Prioritize communication and buy-in efforts based on stakeholder influence and interest levels.

Creating an Effective Stakeholder Analysis Map

To create an effective stakeholder analysis map, follow these four steps:

1. **Identify all relevant stakeholders**: Ask yourself who cares about the project, who will it impact, who can influence it, and who can approve or reject it.
2. **Analyze stakeholder influence-interest levels**: Use a stakeholder analysis matrix to categorize stakeholders based on their level of influence and interest in the project.
3. **Understand stakeholder needs**: Listen to and understand the needs and perspectives of stakeholders.
4. **Prioritize communication and engagement**: Prioritize communication and buy-in efforts based on stakeholder influence and interest levels.

Stakeholder Mapping Techniques

Consider using stakeholder mapping techniques such as:

* **Stakeholder mapping**: Identifying stakeholders and the impact they might have on the project based on two key aspects: stakeholder impact and stakeholder interest.
* **Stakeholder analysis matrix**: Categorizing stakeholders based on their level of influence and interest in the project.
* **RACI chart**: Tracking who everyone is, why they matter, and what their impact on the project will be.

By avoiding common pitfalls and using effective stakeholder analysis techniques, you can ensure that your project is successful and that all stakeholders are engaged and aligned with the project's objectives.