มหาวิทยาลัยธรรมศาสตร์
THAMMASAT UNIVERSITY

[CN230-W14-MiniProject]

Retrieval-Augmented Generation (RAG)

รายงานผลลัพธ์จากการ Query / การค้นหาเชิงความหมาย

จัดทำโดย

6610742436 นายปวริศ แฝงจันดา

6610742246 นายณัฐพัชร์ รีศิริวัฒนกุล

6610742261 นายวุฒิภัทร ผมขุนทด

นำเสนอ

ผศ.ดร.อัครวุฒิ ตาคม

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา

วพ.230 ระบบฐานข้อมูล

(CN230 Database Systems)

ภาคการศึกษา 1/2568

หลักสูตรวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธรรมศาสตร์

1. Create RAG Agent with FAISS

1.1 Implement RAG Agent

```python
from langchain.agents import create_agent
tools = [retrieve_context]
prompt =( """
You are a helpful assistant that answers questions using a retrieval tool.

You MUST:
1. Call `retrieve_context` with the user's query.
2. Wait for the tool result.
3. After receiving the tool result, generate a FINAL ANSWER for the user.
4. DO NOT stop after calling the tool — you must produce a final answer.

If the context does not contain the answer, reply:
"I don't have enough information to answer that question."
""")
agent = create_agent(model, tools, system_prompt=prompt)
```

```python
query = "tell me the name of player who play Role as a Top lane in PSG Esports Team."

for event in agent.stream({"messages": [{"role": "user", "content": query}]}, stream_mode="values"):
    event["messages"][-1].pretty_print()
```

```
================================ Human Message =================================

tell me the name of player who play Role as a Top lane in PSG Esports Team.
================================== Ai Message ==================================
Tool Calls:
  retrieve_context (dc552f45-8cd9-48a2-babe-0270a8908c7e)
 Call ID: dc552f45-8cd9-48a2-babe-0270a8908c7e
  Args:
    query: PSG Esports Team Top lane player
================================= Tool Message =================================
Name: retrieve_context

PSG Esports Top Lane 4Savage Thailand Main

PSG Esports Roamer Acapae Thailand Main

PSG Esports Middle Fakeplayz Thailand Main

PSG Esports Bottom Lane Deboom Thailand Main
================================== Ai Message ==================================

The top lane player for PSG Esports is 4Savage.
```

1.2 Implement RAG Chain

```python
from langchain.agents.middleware import dynamic_prompt, ModelRequest

@dynamic_prompt
def prompt_with_context(request: ModelRequest) -> str:
    last_query = request.state["messages"][-1].text
    retrieved_docs = vector_store.similarity_search(last_query)
    docs_content = "\n\n".join(doc.page_content for doc in retrieved_docs)
    system_message = f"""You are a helpful assistant that can answer questions based on the following context:

{docs_content}

Answer the user's question based only on the provided context. If you cannot answer the question based on the context, say "I don't have enough information to answer that question."
"""
    return system_message

rag_chain_faiss = create_agent(model, tools=[], middleware=[prompt_with_context])
```

```python
query = "Who is a player of Buriram United Esports."

for step in rag_chain_faiss.stream({"messages": [{"role": "user", "content": query}]}, stream_mode="values"):
    step["messages"][-1].pretty_print()
```

```
================================ Human Message =================================

Who is a player of Buriram United Esports.
================================== Ai Message ==================================

Deawwy, NuNu, PogPog, and Kanashi are players of Buriram United Esports.
```

2. Rag pattern 2 graph db using neo4j document

2.1 Implement RAG Agent

```
from langchain.agents import create_agent
neo4j_tools = [neo4j_retrieve_context]
prompt = ("""
You are a helpful assistant that answers questions using a retrieval tool.

You MUST:
1. Call `retrieve_context` with the user's query.
2. Wait for the tool result.
3. After receiving the tool result, generate a FINAL ANSWER for the user.
4. DO NOT stop after calling the tool — you must produce a final answer.

If the context does not contain the answer, reply:
"I don't have enough information to answer that question."
""")
neo4j_agent = create_agent(model, neo4j_tools, system_prompt=prompt)
```

```
query_neo4j = "Who is a player of PSG Esports?."

for event in neo4j_agent.stream({"messages": [{"role": "user", "content": query_neo4j}]}, stream_mode="values"):
    event["messages"][-1].pretty_print()
```

```
================================ Human Message =================================

Who is a player of PSG Esports?.
================================== Ai Message ==================================
Tool Calls:
  neo4j_retrieve_context (240b1846-6b72-40b6-b327-cfe7c07d087e)
 Call ID: 240b1846-6b72-40b6-b327-cfe7c07d087e
  Args:
    query: PSG Esports players
================================= Tool Message =================================
Name: neo4j_retrieve_context

Source: {'role': 'Middle', 'nationality': 'Thailand', 'team': 'PSG Esports', 'player': 'Fakeplayz', 'status': 'Main'}
Content: PSG Esports Middle Fakeplayz Thailand Main

Source: {'role': 'Roamer', 'nationality': 'Thailand', 'team': 'PSG Esports', 'player': 'Acapae', 'status': 'Main'}
Content: PSG Esports Roamer Acapae Thailand Main

Source: {'role': 'Jungler', 'nationality': 'Thailand', 'team': 'PSG Esports', 'player': 'Bonus', 'status': 'Main'}
Content: PSG Esports Jungler Bonus Thailand Main

Source: {'role': 'Roamer', 'nationality': 'Thailand', 'team': 'King of Gamers Club', 'player': 'PJ', 'status': 'Main'}
Content: King of Gamers Club Roamer PJ Thailand Main
================================== Ai Message ==================================

PSG Esports has the following players: Fakeplayz (Middle), Acapae (Roamer), and Bonus (Jungler). All of them are from Thailand and are main players.
```

2.2 Neo4J Implement RAG Chain

```
from langchain.agents.middleware import dynamic_prompt, ModelRequest

@dynamic_prompt
def prompt_with_neo4j_context(request: ModelRequest) -> str:
    last_query = request.state["messages"][-1].text
    retrieved_docs = db.similarity_search_with_score(last_query, k=4)
    docs_content = "\n\n".join(doc[0].page_content for doc in retrieved_docs)
    system_message = f"""You are a helpful assistant that can answer questions based on the following context:

{docs_content}

Answer the user's question based only on the provided context. If you cannot answer the question based on the context, say "I don't have enough information to answer that question."
"""
    return system_message

neo4j_chain = create_agent(model, tools=[], middleware=[prompt_with_neo4j_context])
```

```
query_neo4j = "Who is a player played as a Top Lane in TALON Team."

for step in neo4j_chain.stream({"messages": [{"role": "user", "content": query_neo4j}]}, stream_mode="values"):
    step["messages"][-1].pretty_print()
```

```
================================ Human Message =================================

Who is a player played as a Top Lane in TALON Team.
================================== Ai Message ==================================

NTNz and BASKUNG are players who played as Top Lane in TALON.
```

3. Neo4j with triple data using cypher query

```
from langchain_neo4j import Neo4jGraph, GraphCypherQAChain
import os

graph = Neo4jGraph(
    url=os.environ["NEO4J_URI"],
    username=os.environ["NEO4J_USERNAME"],
    password=os.environ["NEO4J_PASSWORD"]
)

chain = GraphCypherQAChain.from_llm(
    model, graph=graph, verbose=True, allow_dangerous_requests=True
)

chain.run("Who is a player of Bacon Time?")
```

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (p:Node)-[:Main]->(t:Team {name: 'Bacon Time'}) RETURN p.name UNION MATCH (p:Node)-[:King_of_Gamers_Club___Loan]->(t:Team {name: 'Bacon Time'}) RETURN p.name UNION MATCH (p:Node)-[:DNP]->(t:Team {name: 'Bacon Time'},
Full Context:
[{'p.name': 'MarkKy'}, {'p.name': 'Erez'}, {'p.name': 'Kimsensei'}, {'p.name': 'Opar'}, {'p.name': 'TaoX'}, {'p.name': 'Myra'}]

> Finished chain.
'MarkKy, Erez, Kimsensei, Opar, TaoX, Myra is a player of Bacon Time.'
```