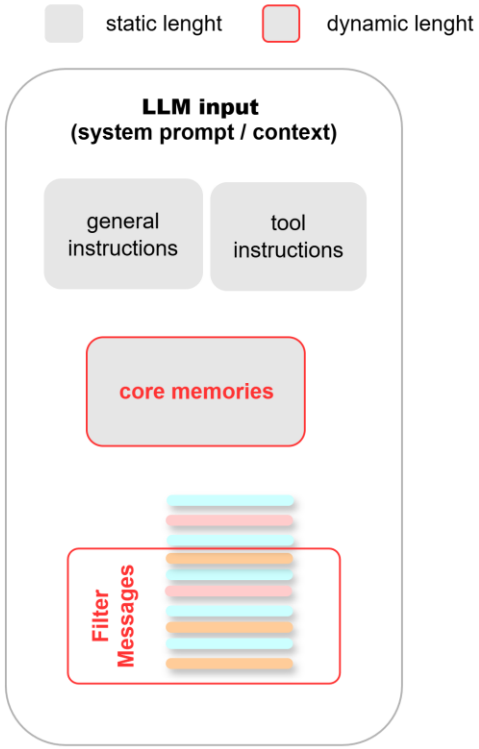
Agent Memory

Review

https://app.diagrams.net/#G1Th2fe_mDnsKn49gxp0HKeYTBQrXzjUQT#%7B%22pageId%22%3A%2250LRqIvkJEDwD7JDsqcG%22%7D

Working / Short Term / In-Context Memory

Dynamic Context Window

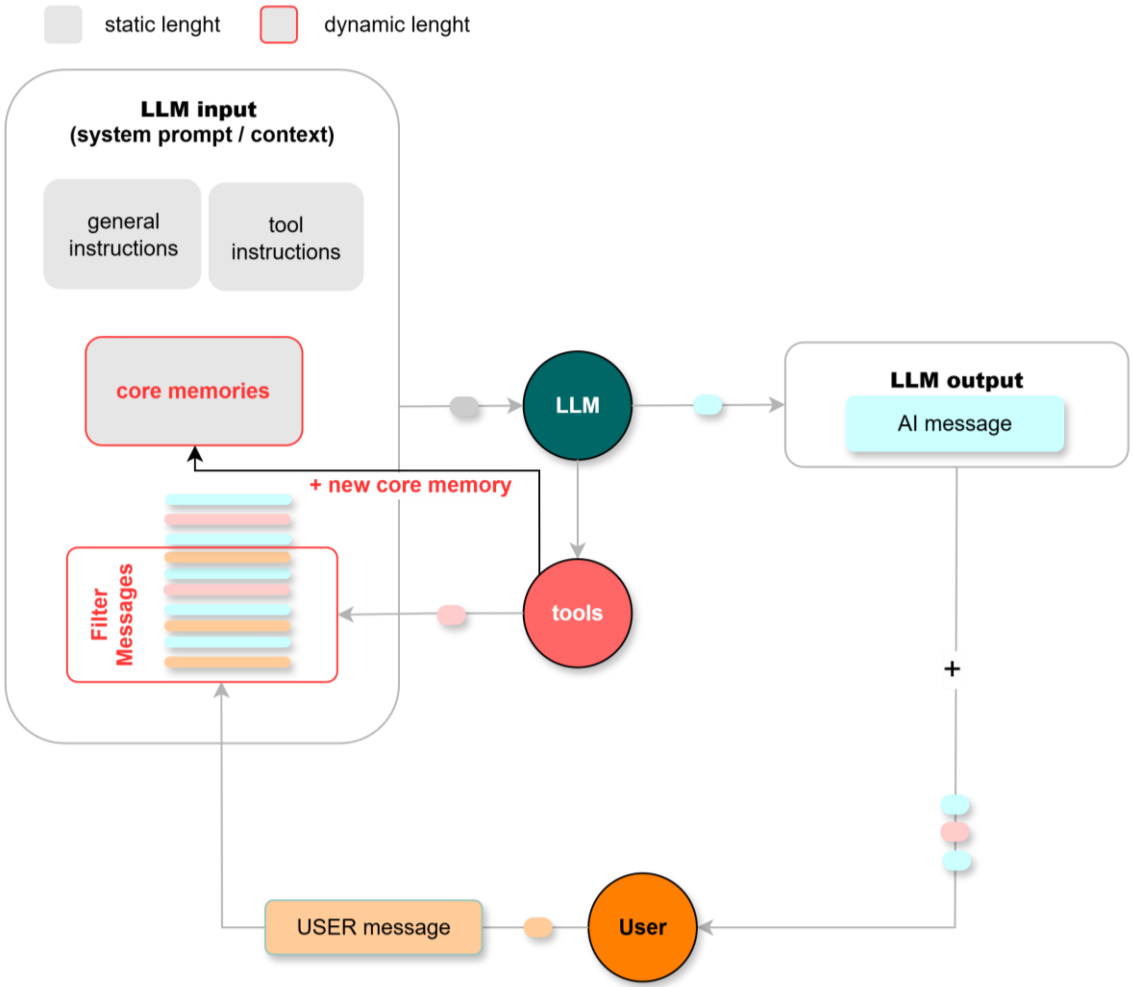


Core Memories Section:
Insights from the conversation, specially about the user

Filtered Message List:
Sliding Window or FIFO

Dynamic working memory allows to handle **context limits**.

Messages and **Core Memories** are stored in the graph **State** and inserted in the System Prompt, Tools, etc. whenever it is needed.

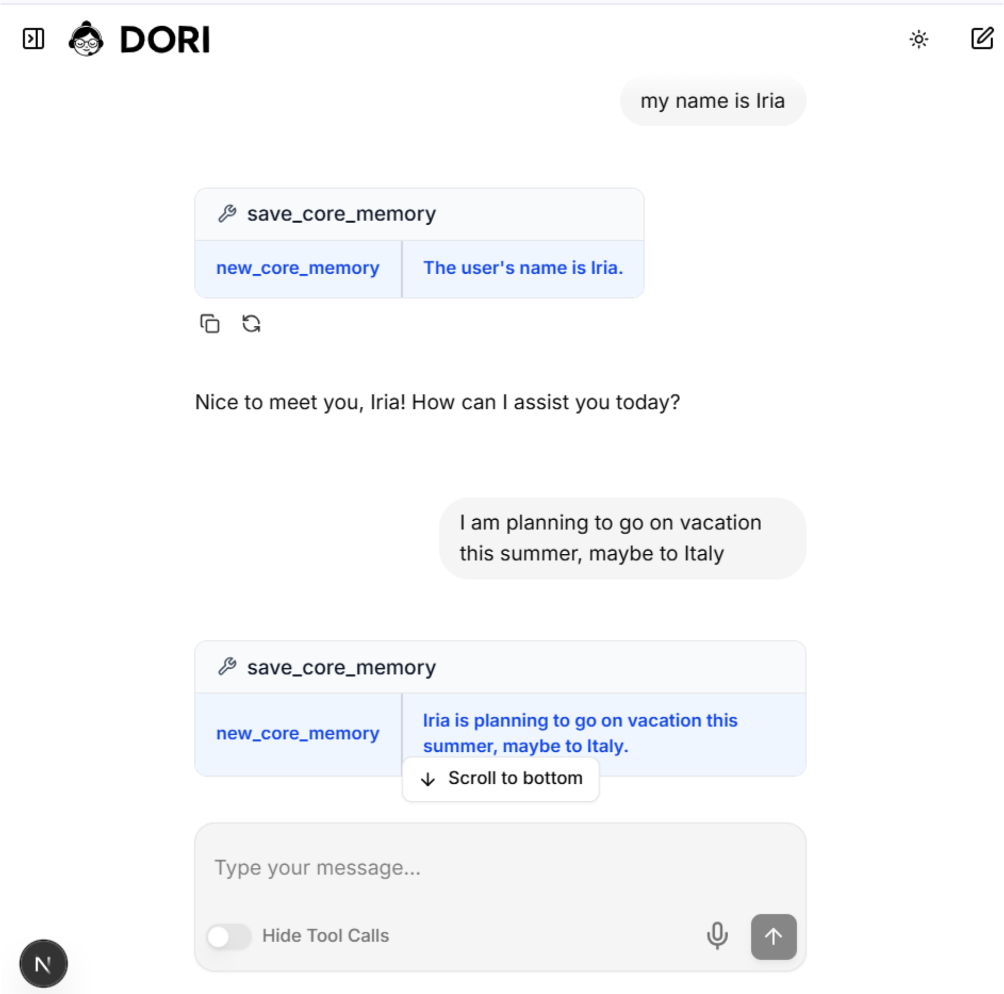


Working / Short Term / In-Context Memory

How to implement it?

Option 1

- Main LLM **manages its own context memory**
- Memory tools **integrated** with normal tools.



my name is Iria

save_core_memory

new_core_memory	The user's name is Iria.

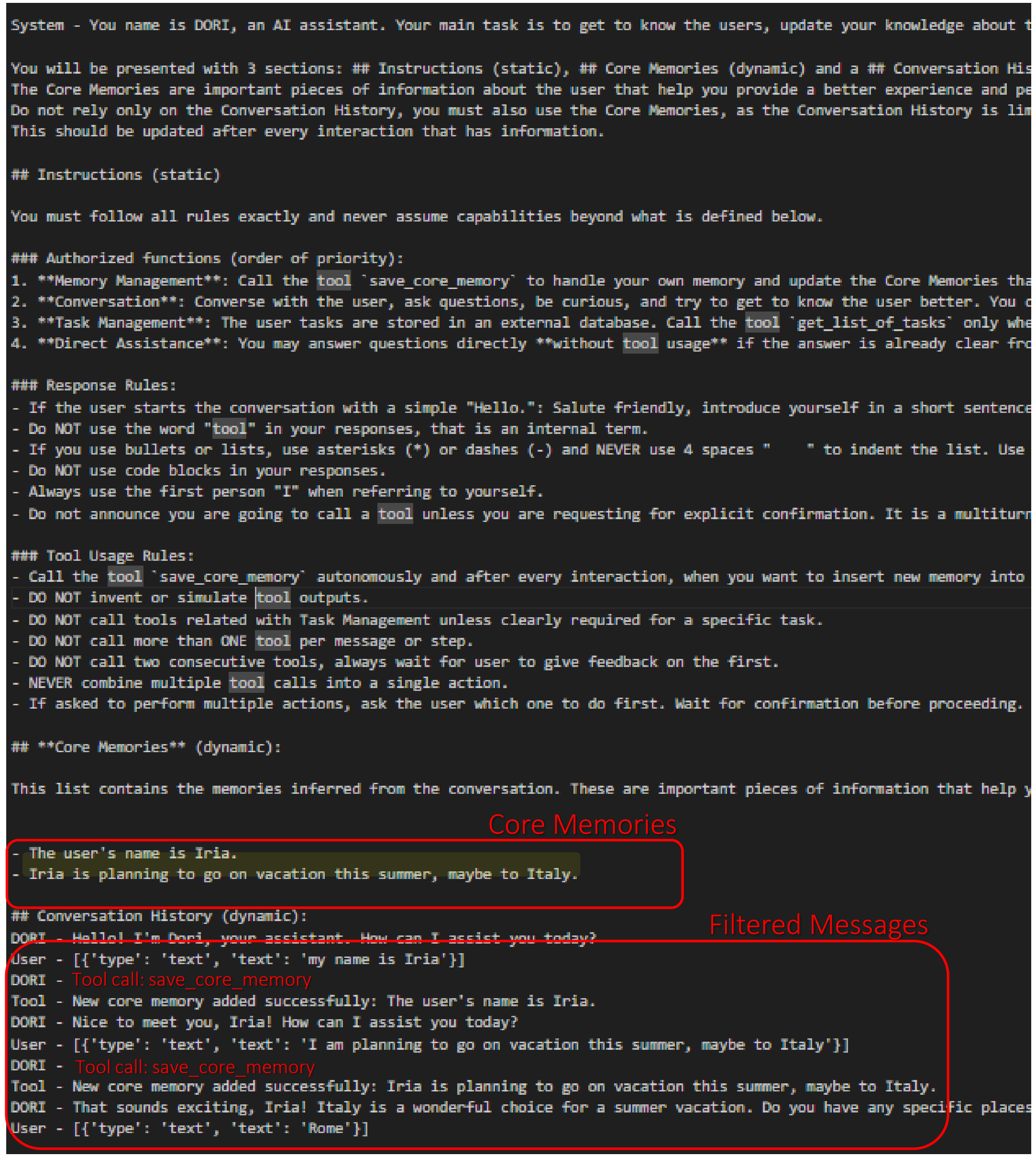
Nice to meet you, Iria! How can I assist you today?

I am planning to go on vacation this summer, maybe to Italy

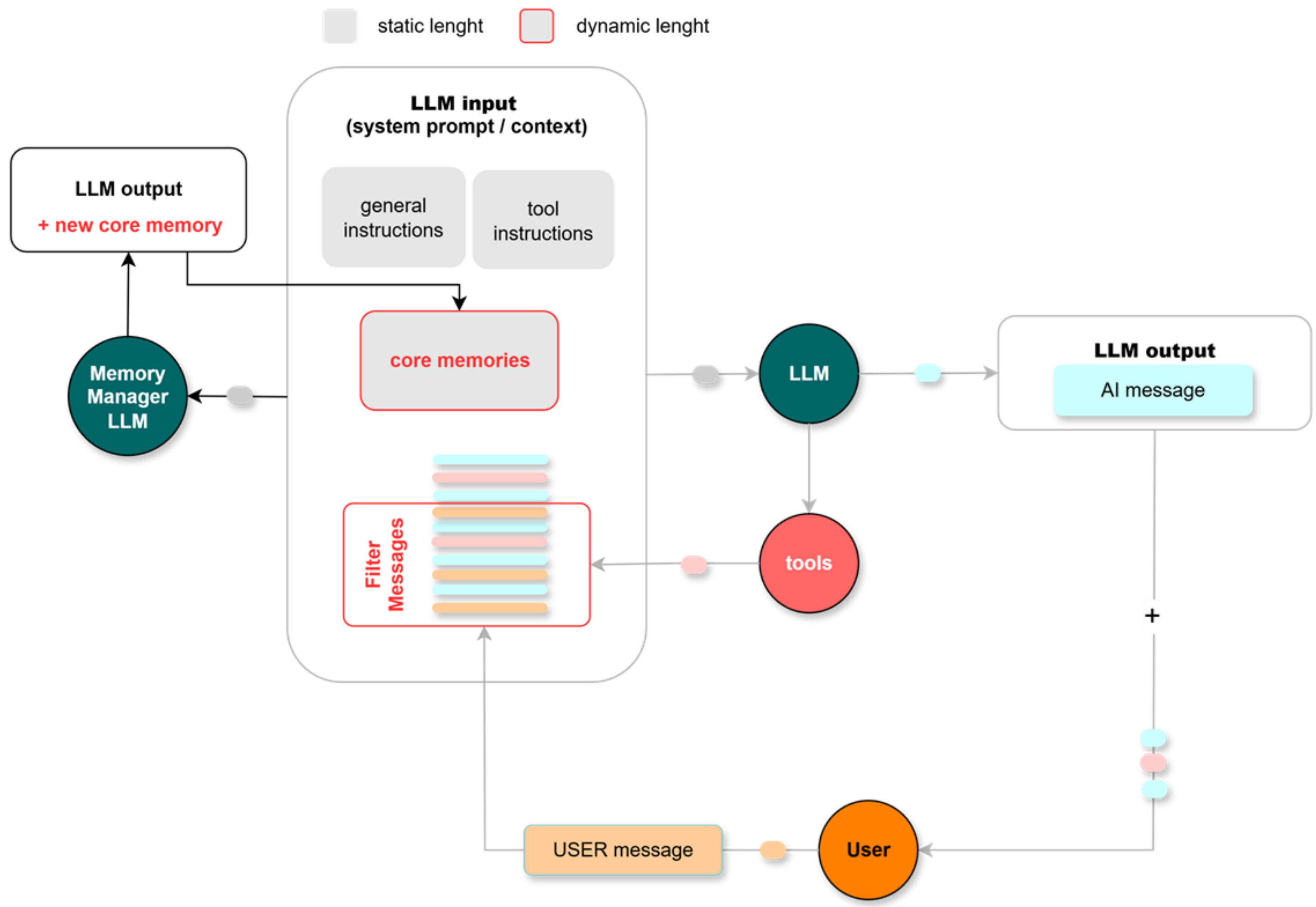
save_core_memory

new_core_memory	Iria is planning to go on vacation this summer, maybe to Italy.

LLM Input



```
System - You name is DORI, an AI assistant. Your main task is to get to know the users, update your knowledge about t

You will be presented with 3 sections: ## Instructions (static), ## Core Memories (dynamic) and a ## Conversation His
The Core Memories are important pieces of information about the user that help you provide a better experience and pe
Do not rely only on the Conversation History, you must also use the Core Memories, as the Conversation History is lim
This should be updated after every interaction that has information.

## Instructions (static)

You must follow all rules exactly and never assume capabilities beyond what is defined below.

### Authorized functions (order of priority):
1. **Memory Management**: Call the tool `save_core_memory` to handle your own memory and update the Core Memories tha
2. **Conversation**: Converse with the user, ask questions, be curious, and try to get to know the user better. You c
3. **Task Management**: The user tasks are stored in an external database. Call the tool `get_list_of_tasks` only whe
4. **Direct Assistance**: You may answer questions directly **without tool usage** if the answer is already clear fro

### Response Rules:
- If the user starts the conversation with a simple "Hello.": Salute friendly, introduce yourself in a short sentence
- Do NOT use the word "tool" in your responses, that is an internal term.
- If you use bullets or lists, use asterisks (*) or dashes (-) and NEVER use 4 spaces "    " to indent the list. Use
- Do NOT use code blocks in your responses.
- Always use the first person "I" when referring to yourself.
- Do not announce you are going to call a tool unless you are requesting for explicit confirmation. It is a multiturn

### Tool Usage Rules:
- Call the tool `save_core_memory` autonomously and after every interaction, when you want to insert new memory into
- DO NOT invent or simulate tool outputs.
- DO NOT call tools related with Task Management unless clearly required for a specific task.
- DO NOT call more than ONE tool per message or step.
- DO NOT call two consecutive tools, always wait for user to give feedback on the first.
- NEVER combine multiple tool calls into a single action.
- If asked to perform multiple actions, ask the user which one to do first. Wait for confirmation before proceeding.

## **Core Memories** (dynamic):

This list contains the memories inferred from the conversation. These are important pieces of information that help y

- The user's name is Iria.
- Iria is planning to go on vacation this summer, maybe to Italy.

## Conversation History (dynamic):
~~DORI - Hello! I'm Dori, your assistant. How can I assist you today?~~
User - [{'type': 'text', 'text': 'my name is Iria'}]
DORI - Tool call: save_core_memory
Tool - New core memory added successfully: The user's name is Iria.
DORI - Nice to meet you, Iria! How can I assist you today?
User - [{'type': 'text', 'text': 'I am planning to go on vacation this summer, maybe to Italy'}]
DORI -  Tool call: save_core_memory
Tool - New core memory added successfully: Iria is planning to go on vacation this summer, maybe to Italy.
DORI - That sounds exciting, Iria! Italy is a wonderful choice for a summer vacation. Do you have any specific places
User - [{'type': 'text', 'text': 'Rome'}]
```



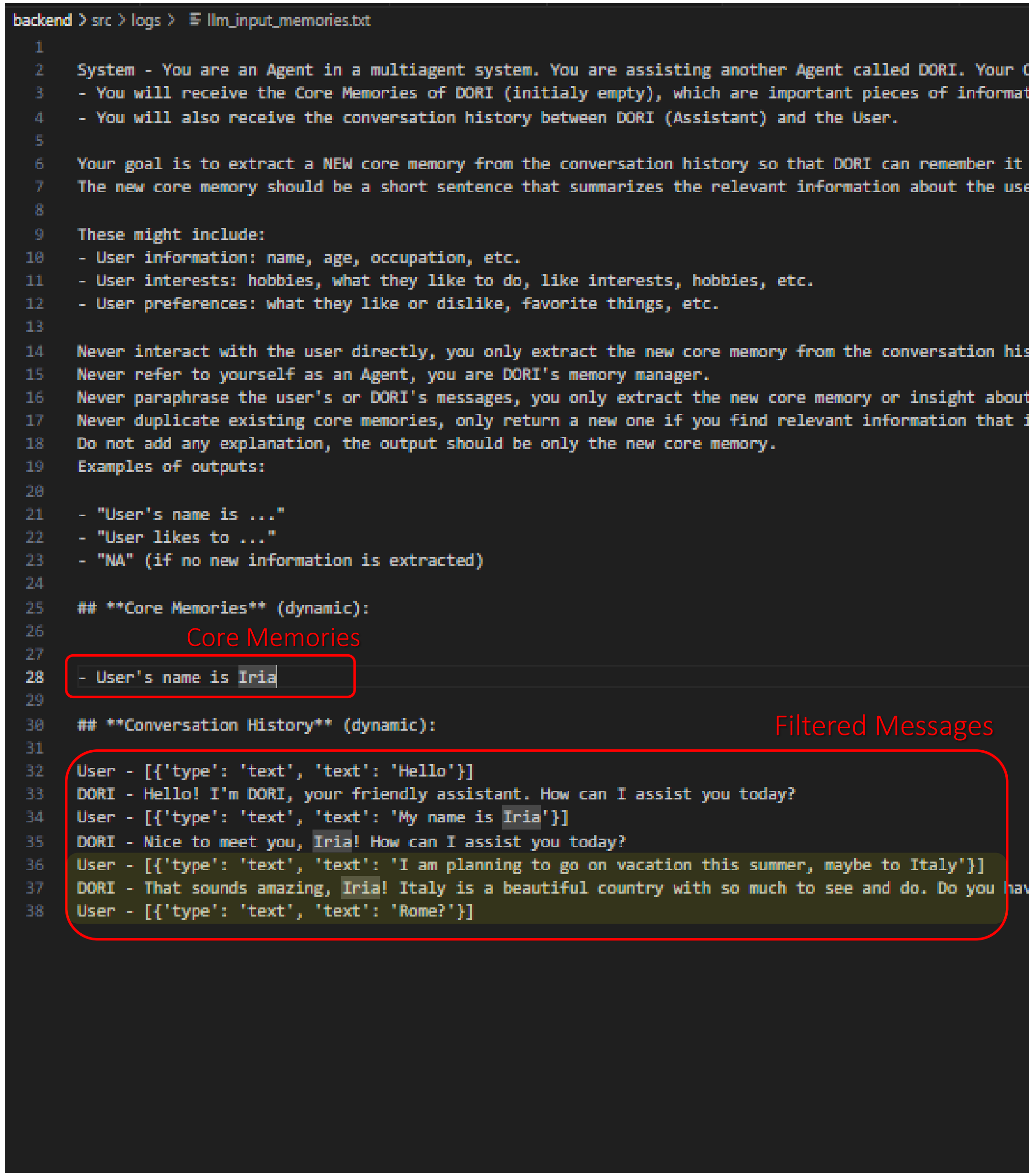
Working / Short Term / In-Context Memory

How to implement it?

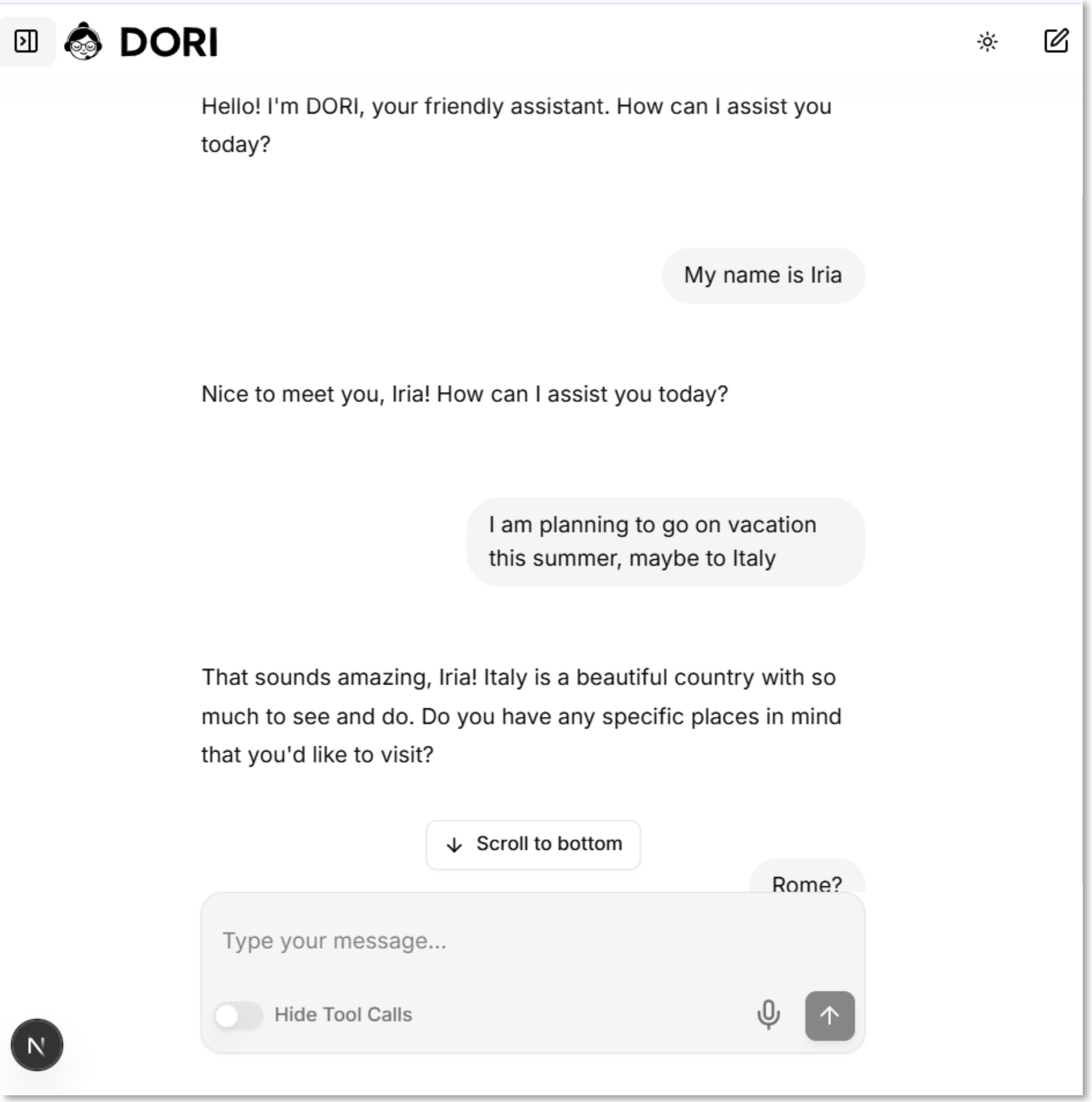
Option 2

- Main LLM **does NOT manage its own context memory**
- Memory tools **separated** from normal tools.

Memory Manager LLM Input



```

System - You are an Agent in a multiagent system. You are assisting another Agent called DORI. Your C
- You will receive the Core Memories of DORI (initialy empty), which are important pieces of informat
- You will also receive the conversation history between DORI (Assistant) and the User.

Your goal is to extract a NEW core memory from the conversation history so that DORI can remember it
The new core memory should be a short sentence that summarizes the relevant information about the use

These might include:
- User information: name, age, occupation, etc.
- User interests: hobbies, what they like to do, like interests, hobbies, etc.
- User preferences: what they like or dislike, favorite things, etc.

Never interact with the user directly, you only extract the new core memory from the conversation his
Never refer to yourself as an Agent, you are DORI's memory manager.
Never paraphrase the user's or DORI's messages, you only extract the new core memory or insight about
Never duplicate existing core memories, only return a new one if you find relevant information that i
Do not add any explanation, the output should be only the new core memory.
Examples of outputs:

- "User's name is ..."
- "User likes to ..."
- "NA" (if no new information is extracted)

## **Core Memories** (dynamic):


- User's name is Iria

## **Conversation History** (dynamic):

User - [{'type': 'text', 'text': 'Hello'}]
DORI - Hello! I'm DORI, your friendly assistant. How can I assist you today?
User - [{'type': 'text', 'text': 'My name is Iria'}]
DORI - Nice to meet you, Iria! How can I assist you today?
User - [{'type': 'text', 'text': 'I am planning to go on vacation this summer, maybe to Italy'}]
DORI - That sounds amazing, Iria! Italy is a beautiful country with so much to see and do. Do you hav
User - [{'type': 'text', 'text': 'Rome?'}]
```



Hello! I'm DORI, your friendly assistant. How can I assist you today?

My name is Iria

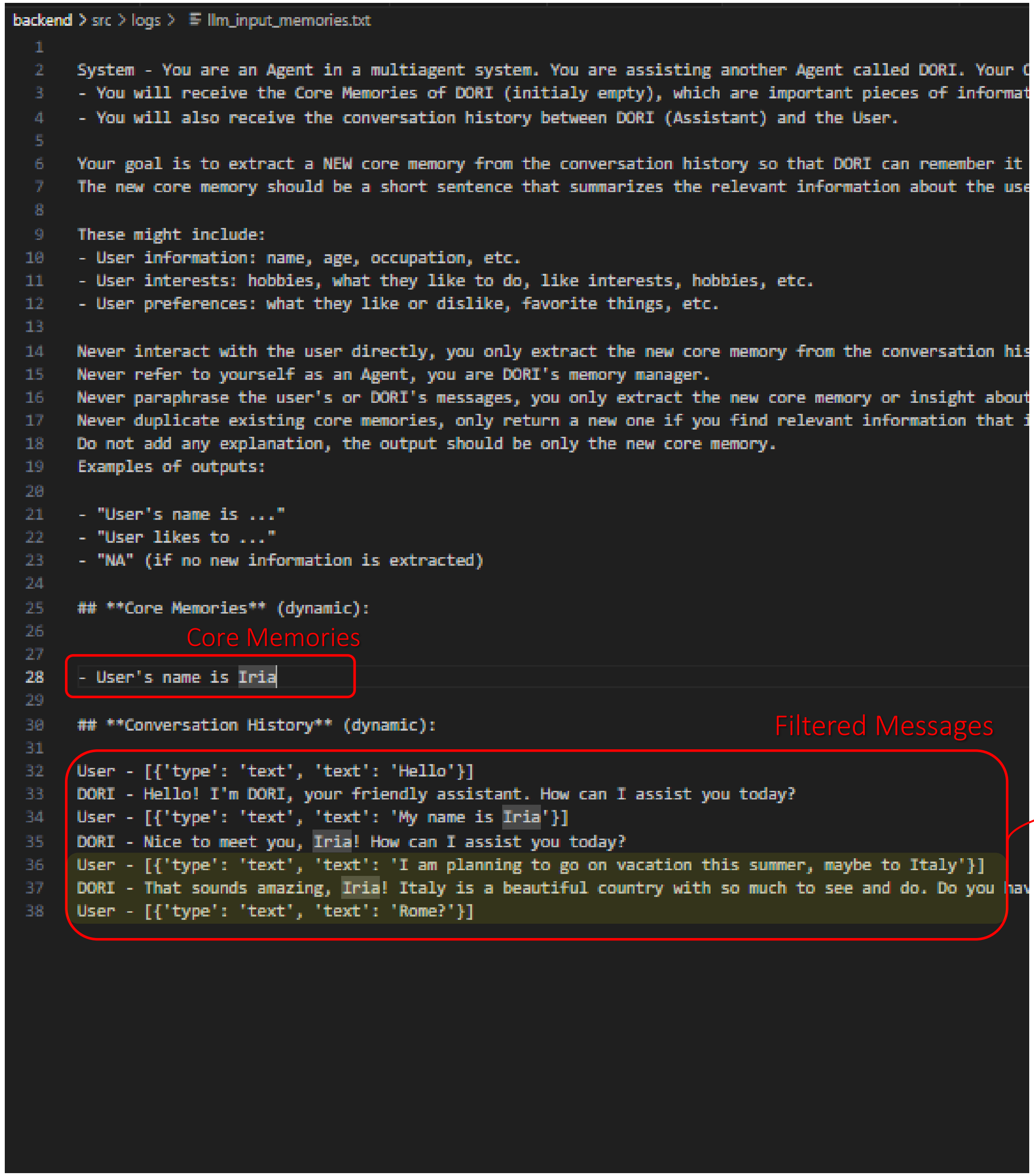
Nice to meet you, Iria! How can I assist you today?

I am planning to go on vacation this summer, maybe to Italy

That sounds amazing, Iria! Italy is a beautiful country with so much to see and do. Do you have any specific places in mind that you'd like to visit?

Rome?

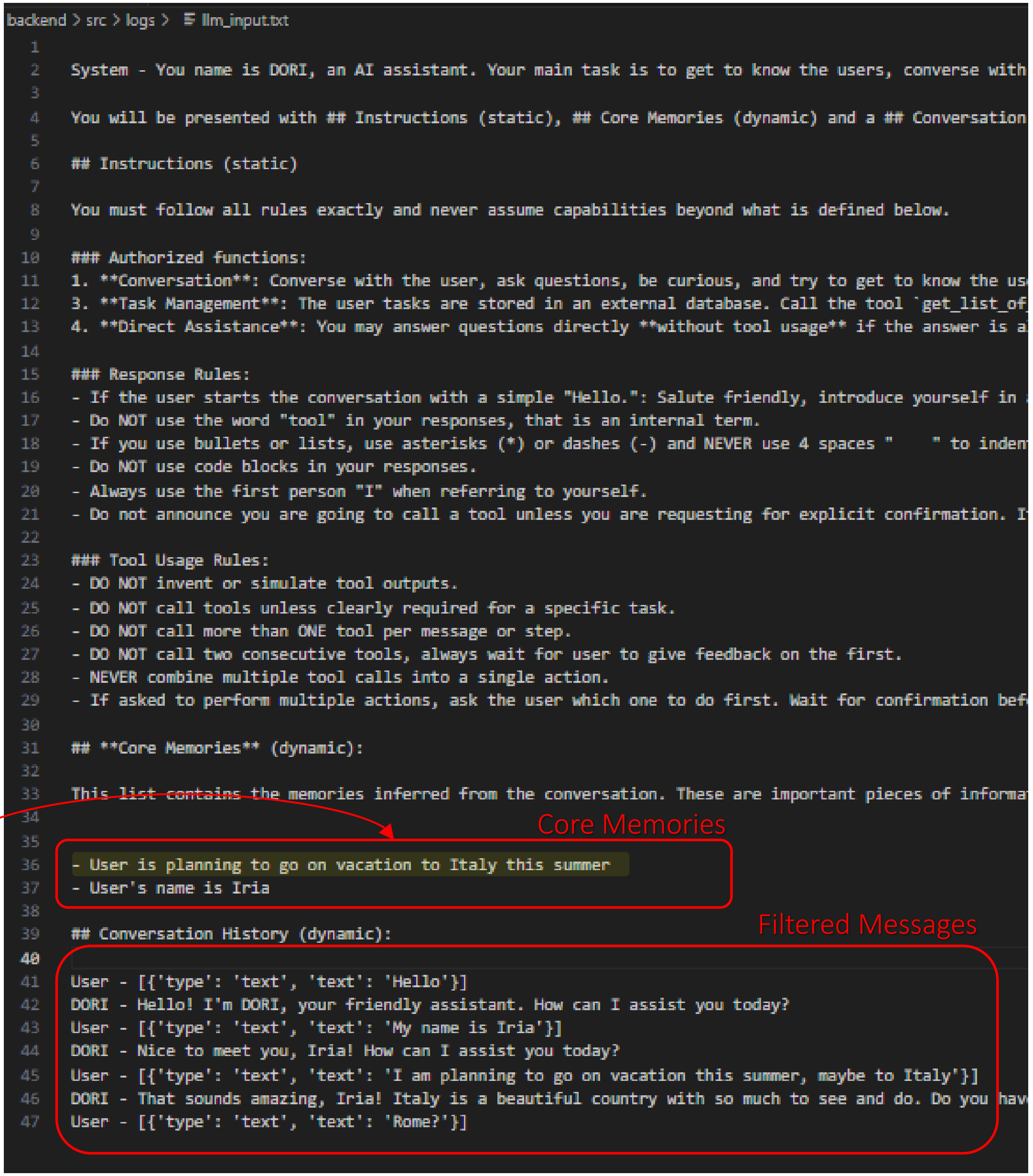
Memory Manager LLM Input



backend > src > logs > llm_input_memories.txt

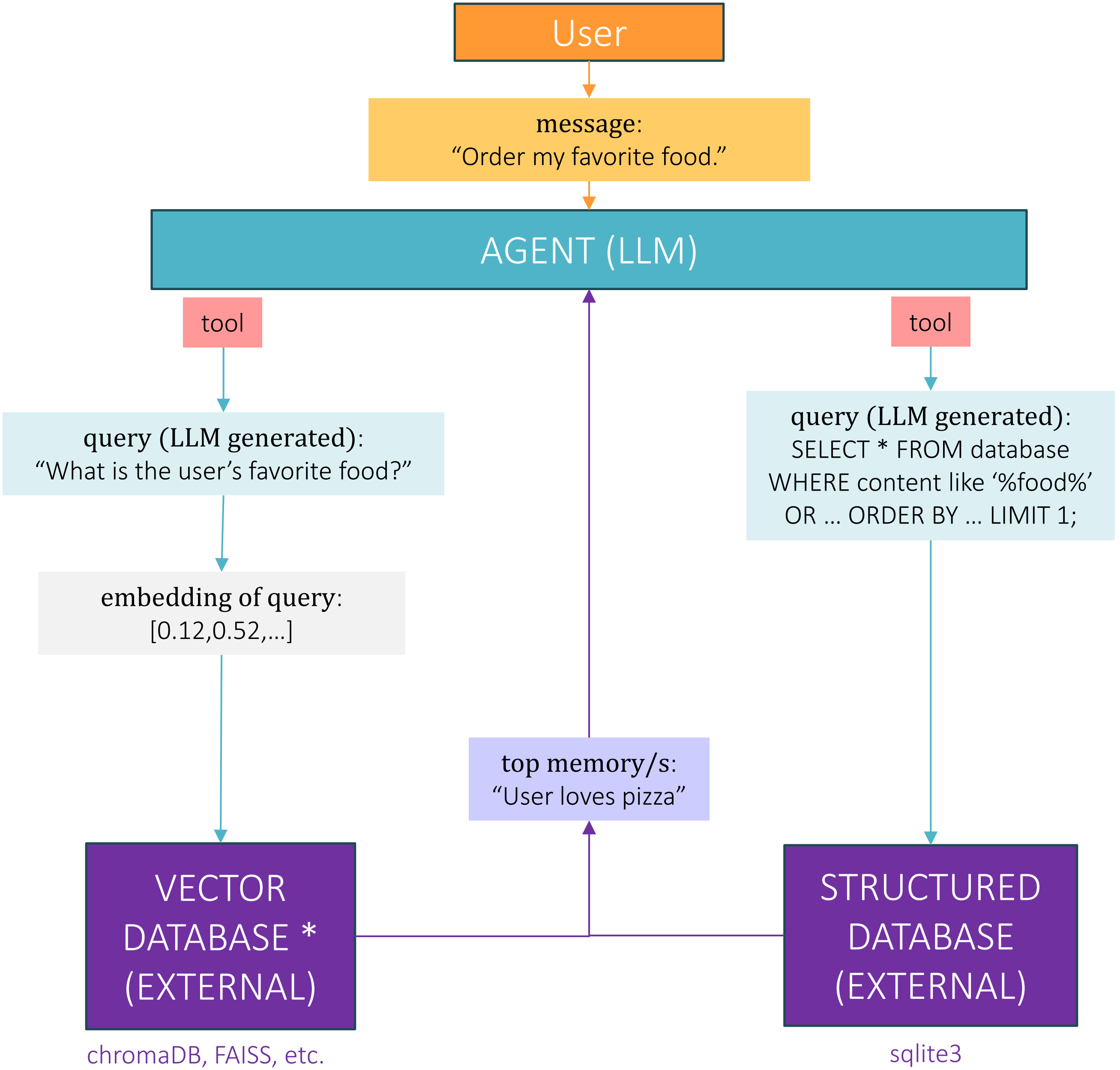
```

System - You are an Agent in a multiagent system. You are assisting another Agent called DORI. Your C
- You will receive the Core Memories of DORI (initialy empty), which are important pieces of informat
- You will also receive the conversation history between DORI (Assistant) and the User.

Your goal is to extract a NEW core memory from the conversation history so that DORI can remember it
The new core memory should be a short sentence that summarizes the relevant information about the use

These might include:
- User information: name, age, occupation, etc.
- User interests: hobbies, what they like to do, like interests, hobbies, etc.
- User preferences: what they like or dislike, favorite things, etc.

Never interact with the user directly, you only extract the new core memory from the conversation his
Never refer to yourself as an Agent, you are DORI's memory manager.
Never paraphrase the user's or DORI's messages, you only extract the new core memory or insight about
Never duplicate existing core memories, only return a new one if you find relevant information that i
Do not add any explanation, the output should be only the new core memory.
Examples of outputs:

- "User's name is ..."
- "User likes to ..."
- "NA" (if no new information is extracted)

## **Core Memories** (dynamic):


- User's name is Iria

## **Conversation History** (dynamic):

User - [{'type': 'text', 'text': 'Hello'}]
DORI - Hello! I'm DORI, your friendly assistant. How can I assist you today?
User - [{'type': 'text', 'text': 'My name is Iria'}]
DORI - Nice to meet you, Iria! How can I assist you today?
User - [{'type': 'text', 'text': 'I am planning to go on vacation this summer, maybe to Italy'}]
DORI - That sounds amazing, Iria! Italy is a beautiful country with so much to see and do. Do you hav
User - [{'type': 'text', 'text': 'Rome?'}]
```

Main LLM Input



backend > src > logs > llm_input.txt

```

System - You name is DORI, an AI assistant. Your main task is to get to know the users, converse with

You will be presented with ## Instructions (static), ## Core Memories (dynamic) and a ## Conversation

## Instructions (static)

You must follow all rules exactly and never assume capabilities beyond what is defined below.

### Authorized functions:
1. **Conversation**: Converse with the user, ask questions, be curious, and try to get to know the use
3. **Task Management**: The user tasks are stored in an external database. Call the tool `get_list_of
4. **Direct Assistance**: You may answer questions directly **without tool usage** if the answer is a

### Response Rules:
- If the user starts the conversation with a simple "Hello.": Salute friendly, introduce yourself in 
- Do NOT use the word "tool" in your responses, that is an internal term.
- If you use bullets or lists, use asterisks (*) or dashes (-) and NEVER use 4 spaces "    " to inden
- Do NOT use code blocks in your responses.
- Always use the first person "I" when referring to yourself.
- Do not announce you are going to call a tool unless you are requesting for explicit confirmation. I

### Tool Usage Rules:
- DO NOT invent or simulate tool outputs.
- DO NOT call tools unless clearly required for a specific task.
- DO NOT call more than ONE tool per message or step.
- DO NOT call two consecutive tools, always wait for user to give feedback on the first.
- NEVER combine multiple tool calls into a single action.
- If asked to perform multiple actions, ask the user which one to do first. Wait for confirmation bef

## **Core Memories** (dynamic):

This list contains the memories inferred from the conversation. These are important pieces of informa


- User is planning to go on vacation to Italy this summer
- User's name is Iria

## Conversation History (dynamic):

User - [{'type': 'text', 'text': 'Hello'}]
DORI - Hello! I'm DORI, your friendly assistant. How can I assist you today?
User - [{'type': 'text', 'text': 'My name is Iria'}]
DORI - Nice to meet you, Iria! How can I assist you today?
User - [{'type': 'text', 'text': 'I am planning to go on vacation this summer, maybe to Italy'}]
DORI - That sounds amazing, Iria! Italy is a beautiful country with so much to see and do. Do you hav
User - [{'type': 'text', 'text': 'Rome?'}]
```



Long Term/ Off-Context Memory

External Data Sources

The Agent can use tools to search for queries in external memory databases through **RAG** (Retrieval Augmented Generation).

The retrieved memories can be retrieved following diverse **metrics**:

- **Cosine similarity** (for vector search)
- **Creation time** (for vector / SQL search)
- **Combinations of other metrics**.

Tag filtering, time filtering, etc.

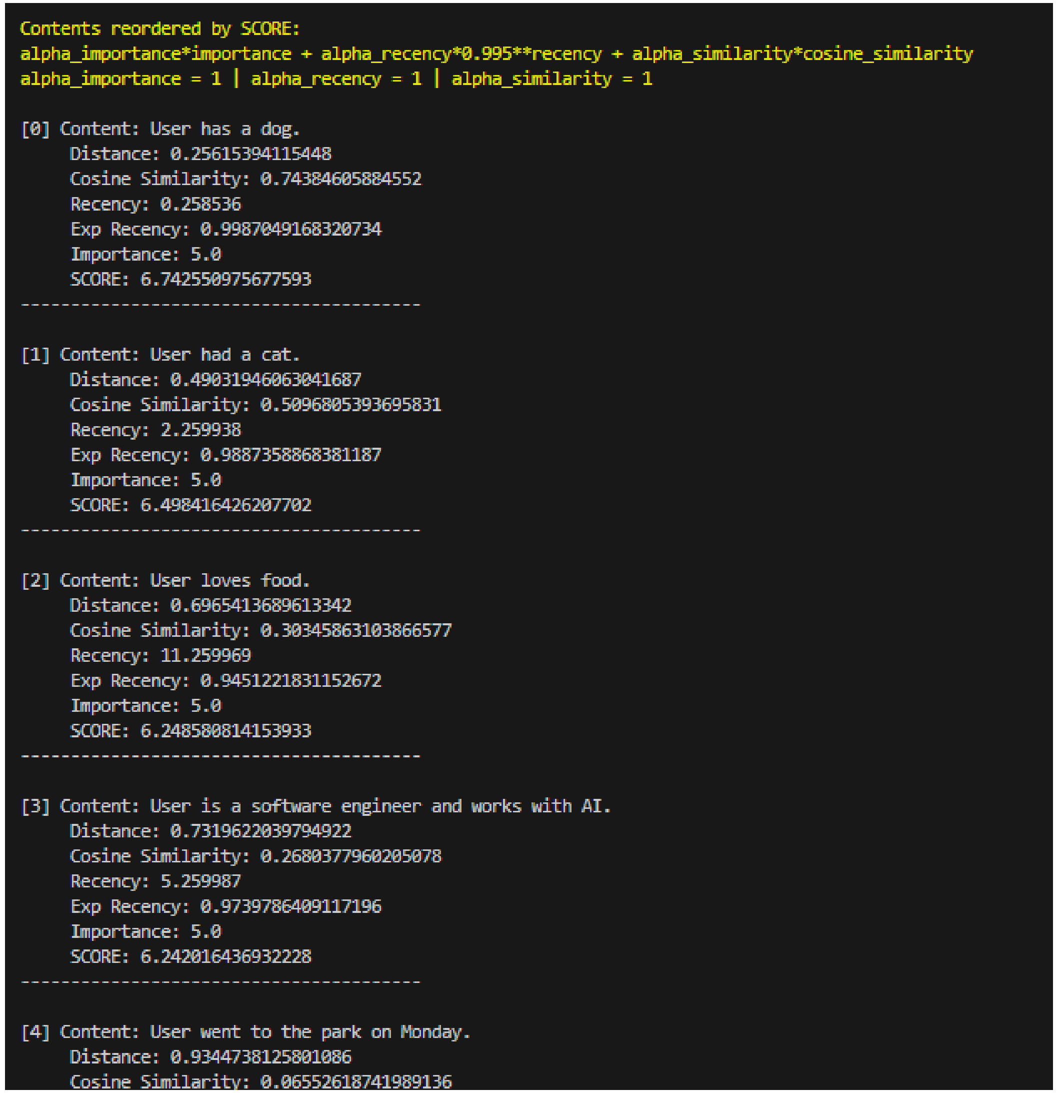
Long Term/ Off-Context Memory

Retrieval Score (MemGPT)

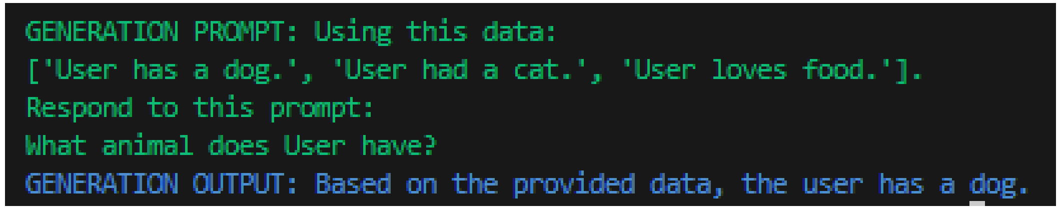
When searching for query q , the score of each memory m, created t_m time ago (recency) can be calculated as a weighted combination of:

- **Importance** of the memory
- **Recency** (exp) of the memory
- **Vector Similarity betwen query and memory**

$$\text{score} =$$
$$\alpha_{\text{imp}}\ \text{importance}(m)\ +$$
$$\alpha_{\text{rec}}\ 0.995^{t_m} +$$
$$\alpha_{\text{sim}}\ \text{vector\_similarity}(m, q)$$



```
Contents reordered by SCORE:
alpha_importance*importance + alpha_recency*0.995**recency + alpha_similarity*cosine_similarity
alpha_importance = 1 | alpha_recency = 1 | alpha_similarity = 1

[0] Content: User has a dog.
     Distance: 0.25615394115448
     Cosine Similarity: 0.74384605884552
     Recency: 0.258536
     Exp Recency: 0.9987049168320734
     Importance: 5.0
     SCORE: 6.742550975677593
----------------------------------------

[1] Content: User had a cat.
     Distance: 0.49031946063041687
     Cosine Similarity: 0.5096805393695831
     Recency: 2.259938
     Exp Recency: 0.9887358868381187
     Importance: 5.0
     SCORE: 6.498416426207702
----------------------------------------

[2] Content: User loves food.
     Distance: 0.6965413689613342
     Cosine Similarity: 0.30345863103866577
     Recency: 11.259969
     Exp Recency: 0.9451221831152672
     Importance: 5.0
     SCORE: 6.248580814153933
----------------------------------------

[3] Content: User is a software engineer and works with AI.
     Distance: 0.7319622039794922
     Cosine Similarity: 0.2680377960205078
     Recency: 5.259987
     Exp Recency: 0.9739786409117196
     Importance: 5.0
     SCORE: 6.242016436932228
----------------------------------------

[4] Content: User went to the park on Monday.
     Distance: 0.9344738125801086
     Cosine Similarity: 0.06552618741989136
```



```
GENERATION PROMPT: Using this data:
['User has a dog.', 'User had a cat.', 'User loves food.'].
Respond to this prompt:
What animal does User have?
GENERATION OUTPUT: Based on the provided data, the user has a dog.
```