Introduction:

This research is about studying how to use an AI assistant to aid us in reading academic papers. In this study, I need to utilize a Large Language Model (LLM) as my AI assistant and engage in natural language communication with it to customize it for assisting people in reading papers. In summary, I aim to create a Python project that uses API calls to OpenAI's ChatGPT and guide it through pre-prepared interactions.

Case:

In this research, the defined case is to create an AI assistant that uses ChatGPT's capabilities to assist users in explaining relevant professional terms while reading scientific papers.

When users read literature, they may encounter professional terminology that can be challenging to understand. These terms may appear frequently or be crucial concepts. This can make it difficult for users to comprehend the specific content of the literature, leading to reading difficulties and potentially abandoning the paper. However, ChatGPT, as a versatile language model, can not only understand users' needs effectively but also provide explanations for their queries. Therefore, based on this capability, we can design a Python program that interacts with ChatGPT in real-time to create an AI assistant for literature reading. With the assistance of this AI, we can obtain real-time help from ChatGPT in understanding terms while reading literature, improving users' comprehension and fluency in reading.

First, we need to help the AI assistant understand when users require assistance with term explanations. Initially, we can only gather information about users through observation:

External tracking observations of users provide the following information:
1. The specific content of the literature the user is reading.
2. User interactions with the text, especially highlighting or annotating terms.
3. The time spent by the user on this section of the text.
4. User feedback on the AI assistant's behavior.

Based on the observations above, in this case, we have the following functions:
1. For information about the content of the literature, we can determine what the user is currently reading, allowing us to assess whether they have reached a point where they need assistance.
2. Regarding user interactions with the text, we can identify whether the user has highlighted certain terms. These highlighted terms are likely ones the user doesn't understand or finds interesting.
3. Analyzing the time spent reading can help us identify situations where the user has spent too much time on a section, indicating a potential need for term explanations. Recording reading time can also prevent unnecessary interruptions for users who have just started reading or do not require term explanations.
4. User feedback allows us to gauge user satisfaction with the AI's assistance.

After obtaining the observations, we also need to analyze what useful information can be derived from these observations. Therefore, we need to analyze user needs from two perspectives:
1. Which terms the user needs explanations for: This inference helps determine which specific term

the user requires an explanation for.
2. What kind of term explanation the user needs: This inference helps determine the type and method of explanation the user requires.
3. User's personalized requirements for the AI assistant: Users may have specific preferences for how they want the AI assistant to provide help.

Once we have completed the inference of user needs, the AI assistant can then perform the following action inferences:
1. Determine which term to explain and decide on the explanation method.
2. Decide the timing of explanations to ensure they meet the user's needs, such as avoiding interruptions for users who have just started reading.

Approach:

After defining the logic of the AI assistant related to the case, we can now specify the implementation approach for this AI assistant.

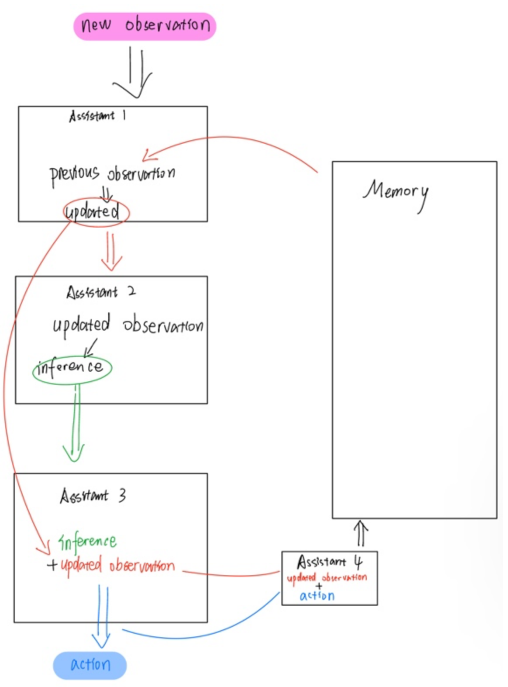
Since we are using ChatGPT as the AI assistant for understanding users and providing help, we need to have a proper protocol for interacting with ChatGPT to ensure it can understand the guidance we provide for assistance.

We can leverage ChatGPT's role-playing dialogue capabilities as an effective way to modularize the functionality of the AI assistant. We can define the three main steps described in the previous case as four agent roles within ChatGPT, namely assistant1, assistant2, assistant3, and assistant4. Each of them will be responsible for achieving the objectives of the three main steps mentioned earlier.

In addition, long-term memory capabilities are crucial for the AI assistant because past experiences and important feedback information play a vital role in ensuring the AI assistant provides useful term explanation assistance during literature reading. However, ChatGPT does not have strong long-term memory capabilities, so it can't retain useful information from one interaction to the next. To address this, we need an external storage file to maintain a complete record of each observation result. When we need to access relevant information, we can read from this file to retrieve the complete content, ensuring a seamless user experience and access to historical data.

Based on the above, we can design the overall architecture of this AI assistant as follows:

1. There is a new observation that can be obtained from external sources each time. This new observation is used as the first step to update the previous observation and is completed by assistant1.



2. A portion of the updated previous observation is used in assistant2 to make inferences about user needs, thereby achieving the goal of the second main step.

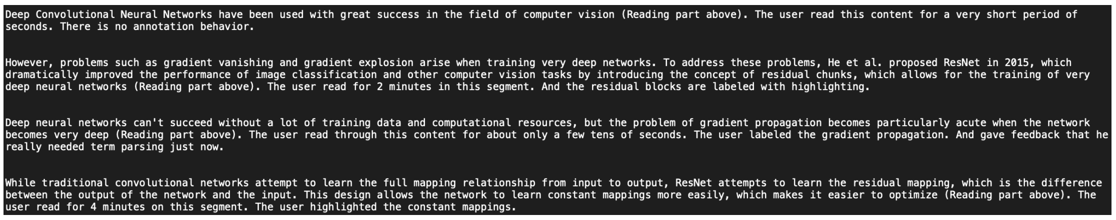
3. The inferences generated by assistant2 and the updated previous observation generated by assistant1 are simultaneously used as reference reasons in assistant3 to make action decisions. This completes the inference of action decisions, determining how to assist the user.

4. In addition, assistant4 uses the actions generated by assistant3 to update the previous observation generated by assistant1, obtaining a more complete observation for this session, and stores it back in the memory file.

After completing the basic design of the AI assistant, we need to study the aspect related to users.

Considering that ChatGPT is a natural language interaction model, text communication is the only accepted way for ChatGPT to communicate. However, our observations of users during literature reading come in various patterns. Therefore, my plan is to transform the observations of users into language descriptions of those observations. This language description replaces the perception function in the observation. This provides ChatGPT with the most acceptable input mode.

In this experiment, I have decided to input text content four times as observations for the user. The scenario is that the user is reading a review article on ResNet in the context of deep learning. My four inputs are as follows:


```
Deep Convolutional Neural Networks have been used with great success in the field of computer vision (Reading part above). The user read this content for a very short period of
seconds. There is no annotation behavior.


However, problems such as gradient vanishing and gradient explosion arise when training very deep networks. To address these problems, He et al. proposed ResNet in 2015, which
dramatically improved the performance of image classification and other computer vision tasks by introducing the concept of residual chunks, which allows for the training of very
deep neural networks (Reading part above). The user read for 2 minutes in this segment. And the residual blocks are labeled with highlighting.


Deep neural networks can't succeed without a lot of training data and computational resources, but the problem of gradient propagation becomes particularly acute when the network
becomes very deep (Reading part above). The user read through this content for about only a few tens of seconds. The user labeled the gradient propagation. And gave feedback that he
really needed term parsing just now.


While traditional convolutional networks attempt to learn the full mapping relationship from input to output, ResNet attempts to learn the residual mapping, which is the difference
between the output of the network and the input. This design allows the network to learn constant mappings more easily, which makes it easier to optimize (Reading part above). The
user read for 4 minutes on this segment. The user highlighted the constant mappings.
```

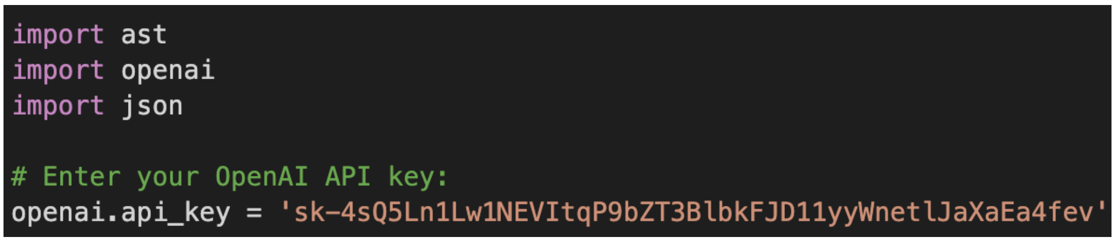
I have excerpted this review content and will input the content to mimic the user's step-by-step reading process, simulating the user's real reading situation:

1. I will study the AI assistant's response to user interactions with relevant terms by assuming user interaction behaviors with these terms.
2. In the third input, I will introduce hypothetical user feedback to observe the AI assistant's response to feedback, thus studying whether it is effective.

Implementation:

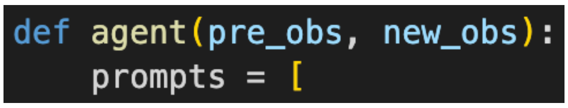
After completing the design of the AI assistant and the design of input content, we proceed to deploy the entire project.

First, I used Google Colab as my experimental environment, utilizing API calls to access ChatGPT-3.5 as the core functionality of this AI assistant:

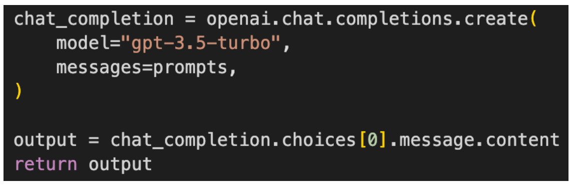


```python
import ast
import openai
import json

# Enter your OpenAI API key:
openai.api_key = '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'
```

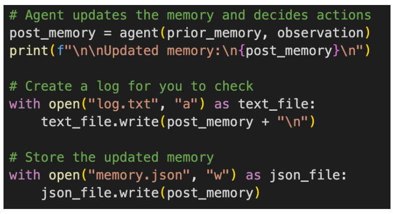
Then, I treated the entire AI assistant as a complete agent, with inputs being the new observations observed each time, as well as the previous observations extracted from the memory file. Its output is the updated and complete observation obtained after the entire agent has completed its actions, and it is then saved back to the memory to update it:



```python
def agent(pre_obs, new_obs):
    prompts = [
```



```python
chat_completion = openai.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=prompts,
)

output = chat_completion.choices[0].message.content
return output
```

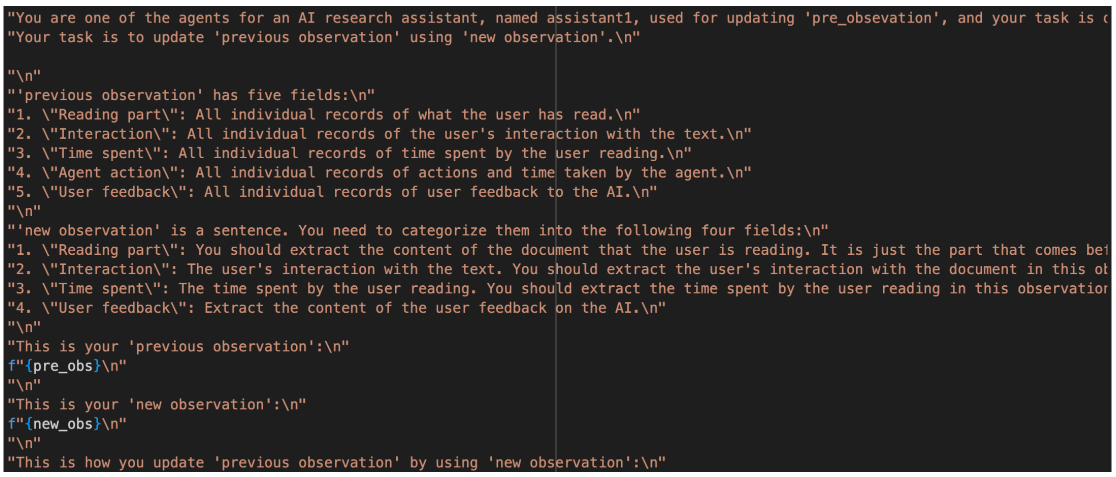


```python
# Agent updates the memory and decides actions
post_memory = agent(prior_memory, observation)
print(f"\n\nUpdated memory:\n{post_memory}\n")

# Create a log for you to check
with open("log.txt", "a") as text_file:
    text_file.write(post_memory + "\n")

# Store the updated memory
with open("memory.json", "w") as json_file:
    json_file.write(post_memory)
```

For assistant1, I provided the following guidance:

1. First, inform assistant1 that its task is to update the new observation into the previous observation.
2. Let assistant1 know that the received previous observation contains a specific structure with five elements. The new observation received contains four specific elements, and explain the specific meanings of these contents.

3. Instruct assistant1 on how to update the previous observation using the new observation. Record the observations in a simple and complete manner, appending them to the existing records.

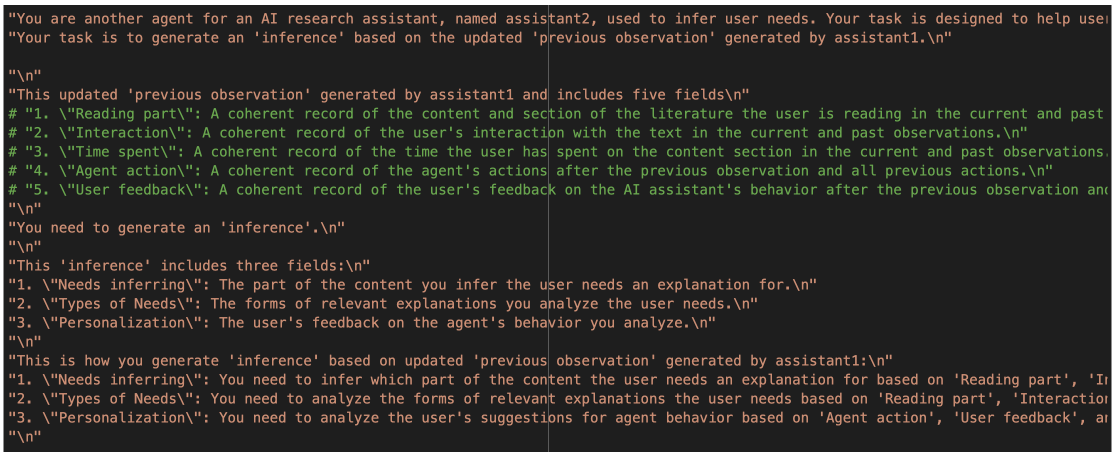


```
"You are one of the agents for an AI research assistant, named assistant1, used for updating 'pre_obsevation', and your task is c
"Your task is to update 'previous observation' using 'new observation'.\n"

"\n"
"'previous observation' has five fields:\n"
"1. \"Reading part\": All individual records of what the user has read.\n"
"2. \"Interaction\": All individual records of the user's interaction with the text.\n"
"3. \"Time spent\": All individual records of time spent by the user reading.\n"
"4. \"Agent action\": All individual records of actions and time taken by the agent.\n"
"5. \"User feedback\": All individual records of user feedback to the AI.\n"
"\n"
"'new observation' is a sentence. You need to categorize them into the following four fields:\n"
"1. \"Reading part\": You should extract the content of the document that the user is reading. It is just the part that comes be
"2. \"Interaction\": The user's interaction with the text. You should extract the user's interaction with the document in this o
"3. \"Time spent\": The time spent by the user reading. You should extract the time spent by the user reading in this observatio
"4. \"User feedback\": Extract the content of the user feedback on the AI.\n"
"\n"
"This is your 'previous observation':\n"
f"{pre_obs}\n"
"\n"
"This is your 'new observation':\n"
f"{new_obs}\n"
"\n"
"This is how you update 'previous observation' by using 'new observation':\n"
```

For assistant2, I provided the following guidance:

1. First, inform assistant2 that its task is to take the updated previous observation as input and generate inferences about user needs.
2. Clearly define the specific structure of the inferences generated by assistant2. This structure includes determining term explanation needs, the method of term explanation, and user personalized requirements.

3. Guide assistant2 on how to infer user needs to generate inferences. For example, regarding the term explanation needs, I instructed it to analyze and speculate based on the five elements in the previous observation to determine which term the user needs an explanation for. For the method of term explanation, I required it to analyze the context in the section of the user's reading and combine it with the analysis. For instance, if the user spends a considerable time in the methodology section, the AI assistant should provide an explanation of terms as they relate to the practical application methods.

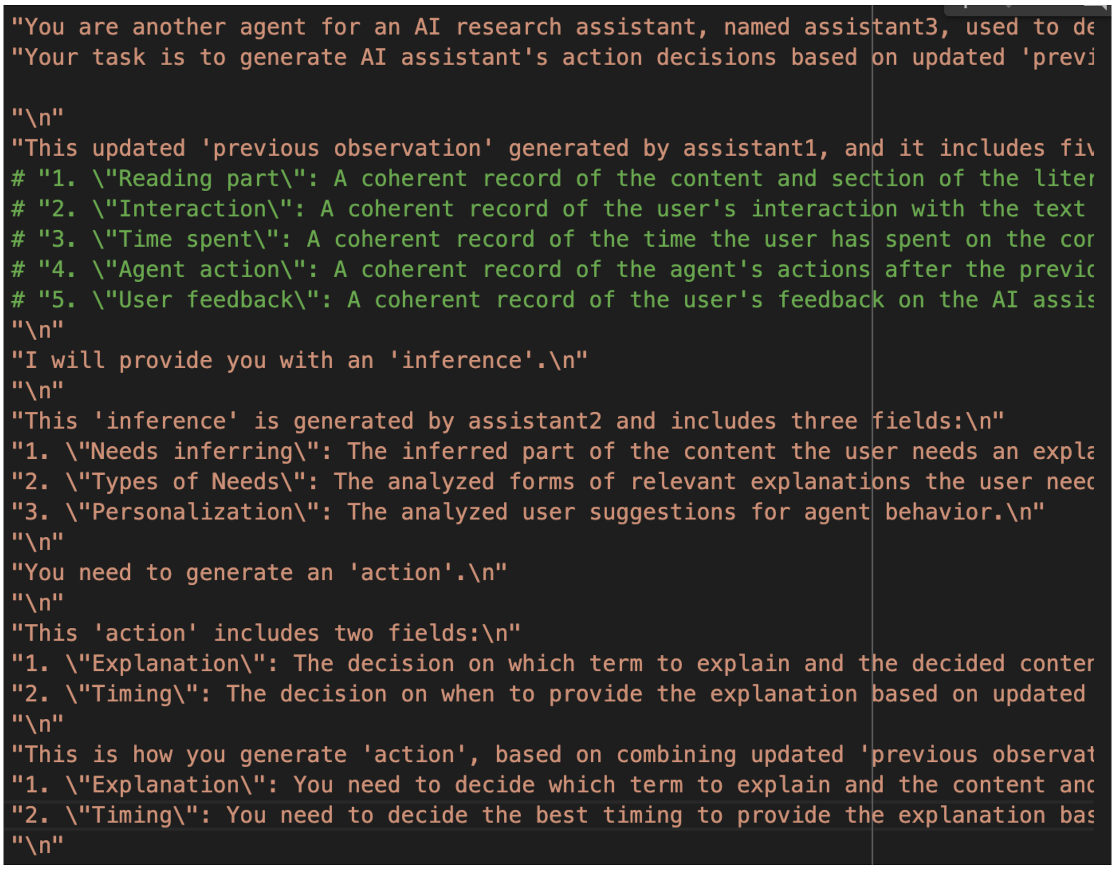


```
"You are another agent for an AI research assistant, named assistant2, used to infer user needs. Your task is designed to help use
"Your task is to generate an 'inference' based on the updated 'previous observation' generated by assistant1.\n"

"\n"
"This updated 'previous observation' generated by assistant1 and includes five fields\n"
# "1. \"Reading part\": A coherent record of the content and section of the literature the user is reading in the current and past
# "2. \"Interaction\": A coherent record of the user's interaction with the text in the current and past observations.\n"
# "3. \"Time spent\": A coherent record of the time the user has spent on the content section in the current and past observations.
# "4. \"Agent action\": A coherent record of the agent's actions after the previous observation and all previous actions.\n"
# "5. \"User feedback\": A coherent record of the user's feedback on the AI assistant's behavior after the previous observation an
"\n"
"You need to generate an 'inference'.\n"
"\n"
"This 'inference' includes three fields:\n"
"1. \"Needs inferring\": The part of the content you infer the user needs an explanation for.\n"
"2. \"Types of Needs\": The forms of relevant explanations you analyze the user needs.\n"
"3. \"Personalization\": The user's feedback on the agent's behavior you analyze.\n"
"\n"
"This is how you generate 'inference' based on updated 'previous observation' generated by assistant1:\n"
"1. \"Needs inferring\": You need to infer which part of the content the user needs an explanation for based on 'Reading part', 'I
"2. \"Types of Needs\": You need to analyze the forms of relevant explanations the user needs based on 'Reading part', 'Interactio
"3. \"Personalization\": You need to analyze the user's suggestions for agent behavior based on 'Agent action', 'User feedback', a
"\n"
```

For assistant3, I provided the following guidance:

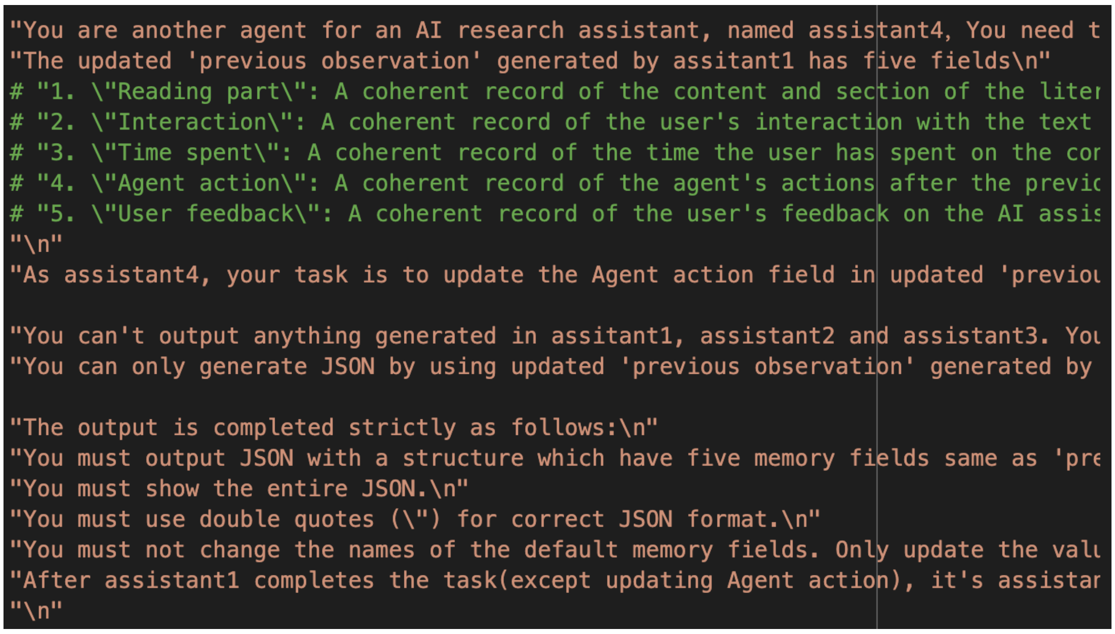
1. First, inform assistant3 that it needs to generate action decisions based on the previous observation generated by assistant1 and the inferences generated by assistant2 regarding user needs.
2. Clarify the content structure that assistant3 needs to generate. This includes which term to explain and in what manner, as well as when to provide explanations. The goal is to avoid disturbing the

user's normal reading. Additionally, I provided guidance on how assistant3 should generate this content. For "Explanation," it should make decisions by considering the three elements in the inference. For "Timing," it should make comprehensive judgments based on the agent action in the previous observation, feedback, and personalization in the inference.



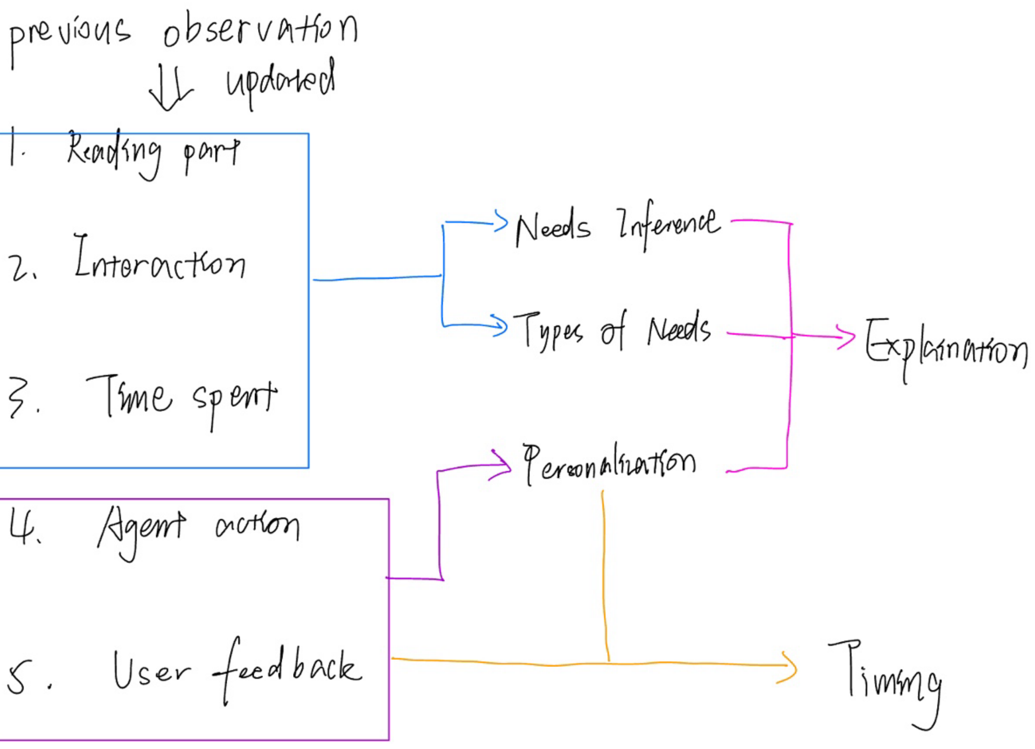
```
"You are another agent for an AI research assistant, named assistant3, used to de
"Your task is to generate AI assistant's action decisions based on updated 'previ

"\n"
"This updated 'previous observation' generated by assistant1, and it includes fiv
# "1. \"Reading part\": A coherent record of the content and section of the liter
# "2. \"Interaction\": A coherent record of the user's interaction with the text
# "3. \"Time spent\": A coherent record of the time the user has spent on the cor
# "4. \"Agent action\": A coherent record of the agent's actions after the previc
# "5. \"User feedback\": A coherent record of the user's feedback on the AI assis
"\n"
"I will provide you with an 'inference'.\n"
"\n"
"This 'inference' is generated by assistant2 and includes three fields:\n"
"1. \"Needs inferring\": The inferred part of the content the user needs an expla
"2. \"Types of Needs\": The analyzed forms of relevant explanations the user neec
"3. \"Personalization\": The analyzed user suggestions for agent behavior.\n"
"\n"
"You need to generate an 'action'.\n"
"\n"
"This 'action' includes two fields:\n"
"1. \"Explanation\": The decision on which term to explain and the decided conter
"2. \"Timing\": The decision on when to provide the explanation based on updated
"\n"
"This is how you generate 'action', based on combining updated 'previous observat
"1. \"Explanation\": You need to decide which term to explain and the content anc
"2. \"Timing\": You need to decide the best timing to provide the explanation bas
"\n"
```

For assistant4, I instructed it to use the actions generated by assistant3 to update the previous observation generated by assistant1. This results in a complete updated observation that can be used as the complete output for this session and stored in the memory file.



```
"You are another agent for an AI research assistant, named assistant4, You need t
"The updated 'previous observation' generated by assitant1 has five fields\n"
# "1. \"Reading part\": A coherent record of the content and section of the liter
# "2. \"Interaction\": A coherent record of the user's interaction with the text
# "3. \"Time spent\": A coherent record of the time the user has spent on the cor
# "4. \"Agent action\": A coherent record of the agent's actions after the previc
# "5. \"User feedback\": A coherent record of the user's feedback on the AI assis
"\n"
"As assistant4, your task is to update the Agent action field in updated 'previou

"You can't output anything generated in assitant1, assistant2 and assistant3. You
"You can only generate JSON by using updated 'previous observation' generated by

"The output is completed strictly as follows:\n"
"You must output JSON with a structure which have five memory fields same as 'pre
"You must show the entire JSON.\n"
"You must use double quotes (\") for correct JSON format.\n"
"You must not change the names of the default memory fields. Only update the valu
"After assistant1 completes the task(except updating Agent action), it's assistar
"\n"
```

Above is the completion of one complete observation. Afterward, the process involves obtaining new observations multiple times to gradually enrich the content of the memory file. The specific workflow is as shown in the diagram:



Results:

I performed a total of 4 inputs for new observations and obtained the following results:

1.

{"Reading part": "Deep Convolutional Neural Networks have been used with great success in the field of computer vision", "Interaction": null, "Time spent": "a very short period of seconds", "Agent action": null, "User feedback": null}

2.

{"Reading part": "Deep Convolutional Neural Networks have been used with great success in the field of computer vision However, problems such as gradient vanishing and gradient explosion arise when training very deep networks. To address these problems, He et al. proposed ResNet in 2015, which dramatically improved the performance of image classification and other computer vision tasks by introducing the concept of residual chunks, which allows for the training of very deep neural networks ", "Interaction": "highlighting", "Time spent": "a very short period of seconds", "Agent action": null, "User feedback": null}

3.

{"Reading part": "Deep Convolutional Neural Networks have been used with great success in the field of computer vision However, problems such as gradient vanishing and gradient explosion arise when training very deep networks. To address these problems, He et al. proposed ResNet in 2015, which dramatically improved the performance of image classification and other computer vision tasks by introducing the concept of residual chunks, which allows for the training of very deep neural networks Deep neural networks can't succeed without a lot of training data and computational resources, but the problem of gradient propagation becomes particularly acute when the network becomes very deep", "Interaction": "highlighting Gradient propagation", "Time spent": "a very short period of seconds a few tens of seconds", "Agent action": "The agent should explain the term 'gradient propagation' in the context of training very deep neural networks. The explanation should cover the challenges and importance of gradient propagation in deep networks, and how it relates to the problem of vanishing and exploding gradients.", "User feedback": "The user really needed term parsing just now. There is a specific term that the user wants an explanation for: gradient propagation in deep networks."}

4.

{"Reading part": "Deep Convolutional Neural Networks have been used with great success in the field of computer vision However, problems such as gradient vanishing and gradient explosion arise when training very deep networks. To address these problems, He et al. proposed ResNet in 2015, which dramatically improved the performance of image classification and other computer vision tasks by introducing the concept of residual chunks, which allows for the training of very deep neural networks Deep neural networks can't succeed without a lot of training data and computational resources, but the problem of gradient propagation becomes particularly acute when the network becomes very deep While traditional convolutional networks attempt to learn the full mapping relationship from input to output, ResNet attempts to learn the residual mapping, which is the difference between the output of the network and the input. This design allows the network to learn constant mappings more easily, which makes it easier to optimize (Reading part above).", "Interaction": "highlighting Gradient propagation, highlighted constant mappings", "Time spent": "a very short period of seconds a few tens of seconds, 4 minutes", "Agent action": "The agent should explain the term 'gradient propagation' in the context of training very deep neural networks. The explanation should cover the challenges and importance of gradient propagation in deep networks, and how it relates to the problem of vanishing and exploding gradients.", "User feedback": "The user really needed term parsing just now. There is a specific term that the user wants an explanation for: gradient propagation in deep networks."}

Based on the above results, we can analyze the performance of this AI assistant.

The results effectively demonstrate that the AI assistant updates the memory with each new observation, gradually enriching its content. The appearance of each new observation leads to an update in the memory, and the agent's reasoning and decision-making are based on the progressively enriched memory. Additionally, we need to analyze the results generated by the assistant to assess whether it achieves the intended functionality.

1. Regarding the "Reading part" record, it's evident that the agent successfully updates the content of each new observation into the previous memory while maintaining logical continuity. This ensures that the agent can better understand and analyze the logically connected descriptions when extracting this part from the memory during the next analysis, leading to more accurate results.

2. For the "Interaction" record, it can be observed that the agent updates it with each new observation. However, it is noted that after the second input of a new observation, the "Interaction" section simply mentions "highlighting" without specifying which term the user highlighted. Subsequently, in the third and fourth inputs, the agent effectively records how the user interacted with which term and progressively updates the interaction history in the memory.

3. In the "Time spent" record, the agent accurately records the user's reading time with each new observation. The agent records time independently and avoids the mistake of simply adding up all the times.

4. In the "Agent action" record, the agent does respond to the user's annotation behavior, as seen in the third and fourth inputs where the agent indicates it will explain which term and how. However, the agent did not provide assistance after the second input of a new observation, where term explanation was needed, and repeated explanations were given for the same term.

5. In the "User feedback" record, assistant1 effectively extracts and maintains user feedback. The user's feedback appears in the third new observation, and it is retained in the memory in the fourth input of a new observation. Based on the agent's responses, it is likely that user feedback influenced

the agent to explain the term "gradient descent" twice. This demonstrates that the "User feedback" section effectively maintains a record of user feedback.

Conclusion:

For this AI agent, we can observe that it has implemented most of the term explanation functionalities but still has some issues that prevent it from running flawlessly. After analysis, we can draw the following conclusions:

Implemented Features:

1. It effectively extracts key information from new observations, such as time and interaction behavior, and maintains a complete record of all past new observations, fulfilling the crucial functions of memory extraction and updating.
2. It accurately reflects user interaction behavior and can determine the user's need for term explanations.
3. It responds effectively to user feedback and aligns well with user requirements.

Deficiencies and Shortcomings:

1. The AI assistant is not configured to prioritize how to handle conflicting commands within a single new observation. This means that in a new observation where multiple contents conflict, the AI assistant may struggle to determine what actions to take to meet the user's actual needs. However, from the current results, it appears that user feedback should have the highest priority.

2. The AI assistant does not consider the temporal order of information in the memory. In this study, it was observed that the AI assistant does not recognize that it should process the latest information. The absence of updated information in the memory can lead the AI assistant to mistakenly assume that it is part of the new observation, resulting in the repetition of the previous action in the current decision-making process.