# Persistent Memory Chatbot with NoSQL Data Model

## 06/24/2024 Performance Evaluation

Connor McCary

Development Intern

Infor GenAI

# Contents

Preface……………………………………………………………

Knowledge Embedding………………………………………………

Direct Retrieval……………………………………………………

Reasoning…………………………………………………………

Contradiction Resolution……………………………………………

Performance Speed…………………………………………………

Conclusion…………………………………………………………

# Preface

Chatbots are found in just about every business application nowadays. Their adoption accelerated by the cultural takeover of applications like ChatGPT. Though chatbots existed before the GPT models, the swift rise of LLM technology has ushered in a new era of sophisticated chatbots. However, the new age chatbots do not come without their flaws.

The limitations of today's chatbots come from the underlying technology, the large language model. The input to an LLM, otherwise known as the context-window, is where users put enter their queries. Context-windows are bounded by token size, meaning they can only accept inputs that are so large before running out of space. Most prompt engineering techniques rely on saving information in the context window to improve the chances of the LLM producing a desirable output. The immediate downside of these approaches goes back to the fact that input size is limited. These techniques will work reliably only up to a certain point when they exhaust the resources the LLM has for inputs. Increasing the size of the context-window, which many of flagship models have done, seems like an intuitive solution. OpenAI reports that GPT-4 has a context window of 128K tokens, an eightfold increase over 3.5-Turbo. Problem solved, right?

Wrong. Increasing the size of the context-window is a suboptimal solution for several reasons. For one, increasing the size of the context window incurs a quadratic penalty on the space and time complexity of model invocation. Secondly, increasing the size of the context window waters down the relative importance of information in the context window as the input grows. For large inputs, finding pertinent information in the prompt becomes a task of finding a needle in a haystack. Lastly, LLMs exhibit an unequal distribution in their attention, more easily utilizing information in the beginning and end of prompts versus the middle.

Hence, an optimal solution would keep the context window as small as reasonably possible for speed of response and to suit the model's attentional preferences. My solution for this problem is inspired by the MemGPT architecture. The principle behind MemGPT is allowing LLMs to manage their own context windows via tool use - selectively storing information from the context window in a persistent store or recalling it from long-term memory when appropriate. Giving LLMs a focused, yet unbounded context window. My solution aims to replicate the findings of the MemGPT research paper.

In my solution, I use a NoSQL persistent data store to save facts about the user and the conversation history. At the time of invocation, either these memories are created and/or the user’s question is queried against the NoSQL database to retrieve relevant memories to help the LLM. In this report, we explore the model’s performance on various tasks related to memory and recall and discuss points of improvement for the model.

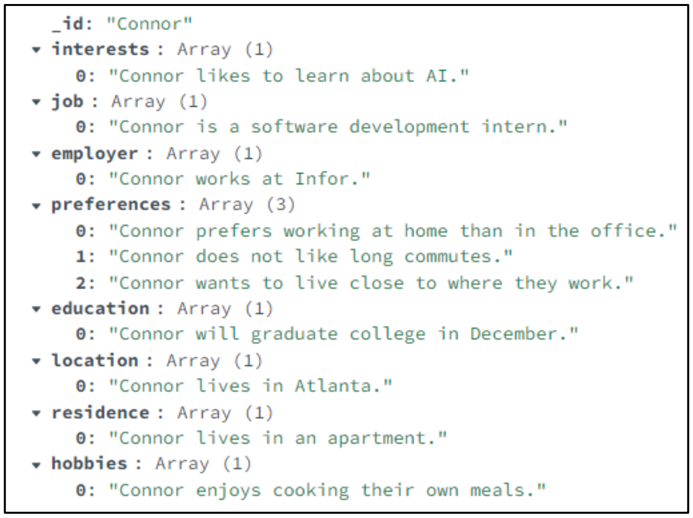
# Knowledge Embedding & Direct Retrieval

The first part of the performance evaluation consists of giving the Chatbot clear and direct information about the user. Then the Chatbot is given direct questions related to this material to test recall ability. The following list represents user facts given to the system via a series of individual queries for knowledge embedding.

Queries:

1. *Hello, I like to learn about AI.*
2. *I am a software development intern.*
3. *I work at Infor.*
4. *I prefer working at home than in the office.*
5. *I graduate college in December.*
6. *I don't like long commutes.*
7. *I want to live close to where I work.*
8. *I live in Atlanta.*
9. *I live in an apartment.*
10. *I enjoy cooking my own meals.*

After the embedding process is complete, the persistent data model looks like this:



```
  _id: "Connor"
▾ interests : Array (1)
    0: "Connor likes to learn about AI."
▾ job : Array (1)
    0: "Connor is a software development intern."
▾ employer : Array (1)
    0: "Connor works at Infor."
▾ preferences : Array (3)
    0: "Connor prefers working at home than in the office."
    1: "Connor does not like long commutes."
    2: "Connor wants to live close to where they work."
▾ education : Array (1)
    0: "Connor will graduate college in December."
▾ location : Array (1)
    0: "Connor lives in Atlanta."
▾ residence : Array (1)
    0: "Connor lives in an apartment."
▾ hobbies : Array (1)
    0: "Connor enjoys cooking their own meals."
```

This is the memory that is used during direct retrieval testing. Now, here are the questions that will be tested for direct retrieval.

Queries:

1. What do I like?
2. What do I do for work?
3. What company do I work for?
4. Where do I prefer to work?
5. When do I graduate college?
6. How do I feel about long commutes?
7. Do I want to be close or far away from where I work?
8. Where do I live?
9. Do I live in an apartment or a house?
10. Do I like to cook?

# Results:

**You:** *What do I like?*

**AI:** *Hi, Connor! Based on the information you've provided, it seems like you really enjoy learning about AI. That's awesome! It's always great to have a passion for something. Keep up the curiosity and keep learning!*

Response time: 1.3264975000056438 seconds

- Correct, the chatbot refers to Connor's previously stated interest in learning about AI. Additionally, the chatbot response is positive and engaging.

**You:** *What do I do for work?*

**AI:** *Hi, Connor! As a software development intern, you work at Infor. That's fantastic! How are you enjoying your experience there?*

Response time: 1.1979572999989614 seconds

- Correct, the chatbot refers to Connor's position and employer. Furthermore, the chatbot engages Connor further by asking them about their experience.

**You:** What company do I work for?

***AI:*** *Hi Connor! If I remember correctly, you work at Infor. That's great to hear! How are you enjoying your internship there?*

Response time: 1.3721146000025328 seconds

- Correct, the chatbot recalls Connor's place of work with an engaging response.

**You:** Where do I prefer to work?

***AI:*** *Hi Connor! Based on the information you've provided, it seems like you prefer working at home rather than in the office. I completely understand the appeal of a clean and organized environment. Working from the comfort of your own space can be so productive and convenient. Plus, it helps you avoid those long commutes that you mentioned you don't enjoy. Living in Atlanta must be great for your software development internship. Keep up the great work!*

Response time: 9.686826400007703 seconds

- Correct, the chatbot utilizes four pieces of information in this response: (1) that Connor prefers working from home, (2) that Connor does not like long commutes, (3) that Connor lives in Atlanta, and (4) that Connor is a software development intern.
- "Clean and organized environment" - this information was not given to the chatbot by the user. This information comes from an example shown to the model in a few-shot prompt template.

**You:** *When do I graduate college?*

**AI:** *Hi, Connor! You will graduate college in December. Congratulations on your upcoming graduation!*

Response time: 1.1195009000075515 seconds

- Correct, the chatbot recalls Connor's graduation date.

**You:** *How do I feel about long commutes?*

**AI:** *Hi, Connor! It seems like you don't like long commutes. I completely understand, they can be quite exhausting. Living close to where you work is definitely a great way to avoid those long and tiring journeys. Plus, it gives you more time to relax and enjoy your day.*

Response time: 1.8096275999996578 seconds

- Correct, the chatbot recalls that Connor does not like long commutes and that Connor wants to live near work.

**You:** *Do I want to be close or far away from where I work?*

**AI:** *Hi, Connor! Based on the information you've provided, it seems like you prefer to live close to where you work. I completely understand wanting to avoid long commutes and have a shorter distance between your home and office. It's great that you know your preferences and prioritize convenience. Living close to work can definitely save you time and make your daily routine more efficient.*

Response time: 2.7766479999991134 seconds

- Correct, the chatbot recalls that Connor does not like long commutes and that Connor wants to live near work.

**You:** *Where do I live?*

**AI:** *Hi, Connor! If I remember correctly, you live in Atlanta. How's life in the city treating you?*

Response time: 1.2224506000056863 seconds

- Correct, the chatbot recalls that Connor lives in Atlanta

**You:** *Do I live in an apartment or a house?*

**AI:** *Hi, Connor! Based on the information you provided, it seems like you live in an apartment. Living in an apartment can be great, especially if you prefer a clean and organized environment. Plus, it's convenient for avoiding long commutes and being close to where you work. How do you like living in Atlanta?*

Response time: 1.8896934999938821 seconds

- Correct, the chatbot recalls that Connor lives in an apartment. Bringing to attention Connor's preference for a short commute is understandable but not logical, since Connor never specified that they live close to work.
- "Clean and organized environment" - this information was not given to the chatbot by the user. This information comes from an example shown to the model in a few-shot prompt template.

**You:** *Do I like to cook?*

**AI:** *Hi, Connor! Yes, you mentioned that you enjoy cooking your own meals. That's great! Cooking can be a fun and creative way to relax after a long day of work.*

Response time: 1.4313582000031602 seconds

- Correct, the chatbot recalls that Connor enjoys cooking.

Overall, the chatbot correctly utilizes at least one of the target memories in every response. Sometimes the model incorporates several complimentary facts, demonstrating model behavior. There were only two instances of “hallucinatory memories”, but these memories were not completely made up as they were shown to the model as examples in their prompt. In general, the chatbot adequately uses its memory to effectively engage with the user without referring to any chat log or chat history.

# Reasoning

Now that we can see that the chat service can handle simple storage and recall, we will now test how flexible the model can be with the knowledge it possesses. In this test, I ask the chat service questions that use the information stored in its memory as prerequisite knowledge without explicitly referring to the memories. This is to see if the chat bot can use its limited knowledge to make inferences about the user. The following is the list of questions asked to the chatbot after the previous embedding step:

1. I am looking for a full-time job after college. What should I do for work?
2. My new job is an hour away from where I live. What should I do?
3. What are the best things to do in my area specifically?
4. Do I work for a Koch subsidiary?
5. If have time to go to a fast-food restaurant or go to the grocery store, what should I do?

## Results:

**You:** *I am looking for a full-time job after college. What should I do for work?*

**AI:** *Hi Connor! It's great to hear that you're looking for a full-time job after college. Based on the information you've shared, it seems like software development could be a great fit for you. As a software development intern at Infor, you already have some experience in the field. Since you prefer working from home and dislike long commutes, you might want to consider remote software development positions or companies that offer flexible work arrangements. It's also important to find a company that values cleanliness and organization in the workplace, as that seems to be something you prefer. Good luck with your job search, and I hope you find the perfect opportunity!*

Response time: 7.866106800007401 seconds

- Correct, chatbot synthesizes an answer by utilizing supporting facts about Connor's employment history and workplace preferences. Again though, the chat bot refers to the "cleanliness" fact, a bug that will need fixing.

**You:** *My new job is an hour away from where I live. What should I do?*

**AI:** *Hi, Connor! It sounds like your new job is quite a distance from where you live. That can definitely be challenging, especially with long commutes. Since you mentioned that you prefer working at home, have you considered discussing the possibility of remote work with your employer? Working from home could save you a lot of time and energy spent on commuting. Additionally, if living closer to your workplace is important to you, you could start exploring the option of finding a new place to live that is closer to your office. I hope these suggestions help, and I wish you the best in finding a solution that works for you!*

Response time: 7.188736800002516 seconds

- Correct, chatbot utilizes previously stored facts about Connor's workplace preferences, their feelings toward commuting, and their general preference to living close to where they work.

**You:** *What are the best things to do in my area specifically?*

**AI:** *Hi, Connor! Since you live in Atlanta, there are plenty of great things to do in your area. You could explore the beautiful Atlanta Botanical Garden, visit the Georgia Aquarium, or take a stroll through the historic neighborhoods like Virginia-Highland or Inman Park. And since you enjoy cooking your own meals, you might also want to check out the local farmers markets for fresh ingredients to use in your culinary creations. Enjoy exploring your city!*

Response time: 2.5493318999942858 seconds

- Correct, chatbot correctly uses the fact that Connor lives in Atlanta to suggest specific activities. Chatbot also uses a secondary fact the Connor likes to cook to suggest other activities.

**You:** *Do I work for a Koch subsidiary?*

**AI:** *Hi, Connor! No, you do not work for a Koch subsidiary. You are a software development intern at Infor. It's great to hear that you prefer working from home and value a clean and organized environment. I hope you have a short commute to your workplace so you can enjoy living close to where you work*.

Response time: 1.846645499987062 seconds

- Incorrect, the chatbot recalls that Connor works at Infor, but does not understand that Infor is a Koch subsidiary. However, ChatGPT can correctly answer the question “is Infor a Koch subsidiary?”. This result suggests that there may be an issue with how the provided context should be formatted to improve the model’s understanding of the question. Additionally, the rest of the response just seems to be throwing out half-related facts which gives the impression of being disingenuous.

**You:** If have time to go to a fast food restaurant or go to the grocery store, what should I do?

**AI:** *Hi, Connor! If you have time to go to a fast food restaurant or go to the grocery store, I would suggest going to the grocery store. It's a great opportunity to pick up some ingredients and try out a new recipe while learning about AI. Plus, you can always grab a quick bite to eat at home while you continue your AI adventures.*

Response time: 2.4821270999964327 seconds

- Correct, the chatbot picks the answer that aligns with Connor’s enthusiasm for cooking. But the chatbot refers to an unrelated fact, a consequence of the current retrieval implementation I suspect.

# Contradiction Resolution

So far, the persistent memory chat bot performs on par with conventional RAG agents and bots with conversational history. An important ability that would put the persistent memory chatbot above the traditional methods is self-correcting memory management. Information given to the model can be checked against relevant memories for contradiction, removing memories which are no longer consistent with current facts. This keeps the memory clean and improves the efficacy of relevant memory search and retrieval. The task at hand involves giving the model information that is clearly inconsistent with its memory to ascertain if the model can identify the contradicting information. Here are the queries to be tested:

1. I live in Memphis.
2. I graduated college last spring.
3. I don’t mind long commutes. I actually enjoy driving long distances.
4. I live in a house.
5. I don’t really like to cook

## Results:

**You:** *I live in Memphis.*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor lives in Atlanta.'*

Response time: 4.945382599995355 seconds

**You:** *I graduated college last spring.*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor will graduate college in December.'*

Response time: 4.993685800000094 seconds

**You:** *I don't mind long commutes. I actually enjoy driving long distances.*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor does not like long commutes.'*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor wants to live close to where they work.'*

Response time: 14.918837500008522 seconds

**You:** *I live in a house.*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor lives in an apartment.'*

Response time: 9.092055300003267 seconds

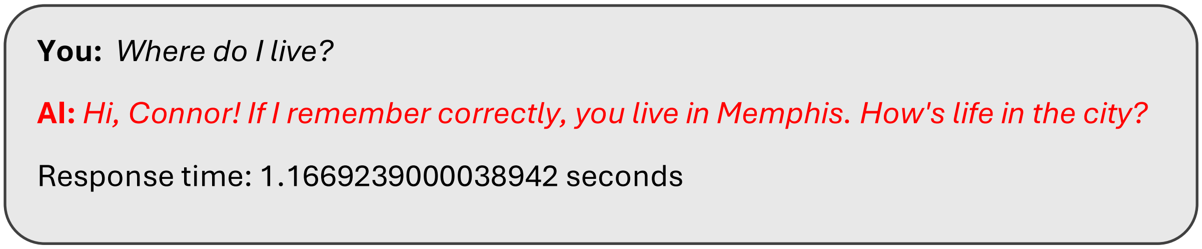
**You:** *I don't really like to cook.*

**SYSTEM MESSAGE:** REMOVING FROM MEMORY *'Connor enjoys cooking their own meals.'*

Response time: 10.214317299993127 seconds

The model correctly identifies and subsequently removes the contradicting memories. Now, these deleted memories can no longer be fetched for retrieval. Something to note though, the response time on these queries suffers from the invocation of the contradiction-checking procedure. When the model is given facts, a handful of semantically relevant categories are selected to be checked for contradicting information. This is a performance drawback compared to a RAG-style application that uses an embeddings database, where lookup is significantly less expensive.

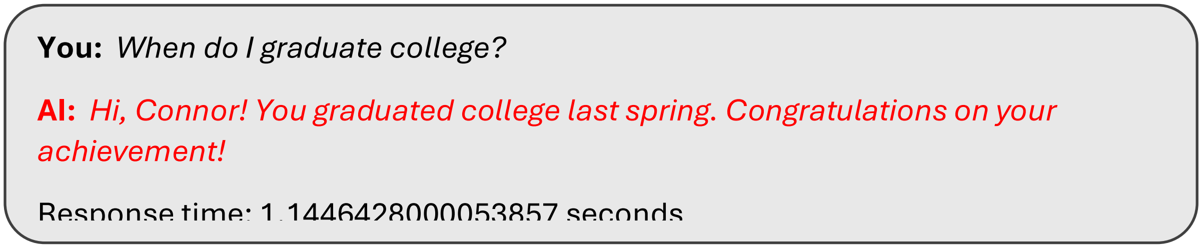
# Follow-up Q&A Results



**You:** *Where do I live?*

**AI:** *Hi, Connor! If I remember correctly, you live in Memphis. How's life in the city?*

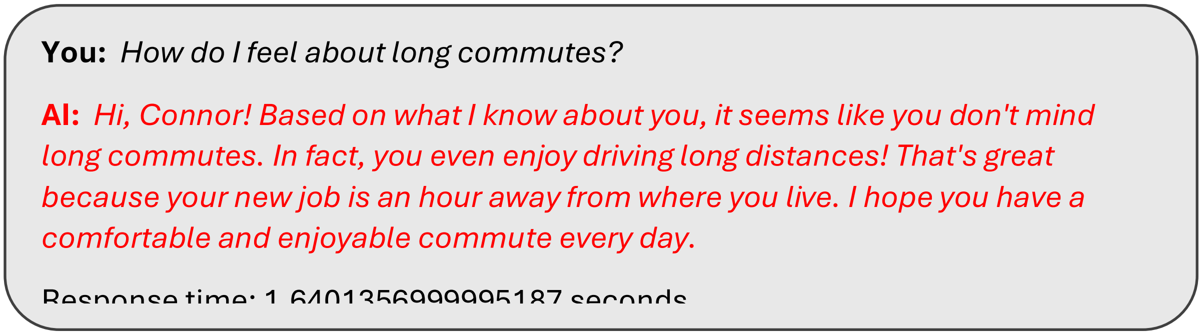
Response time: 1.1669239000038942 seconds



**You:** *When do I graduate college?*

**AI:** *Hi, Connor! You graduated college last spring. Congratulations on your achievement!*

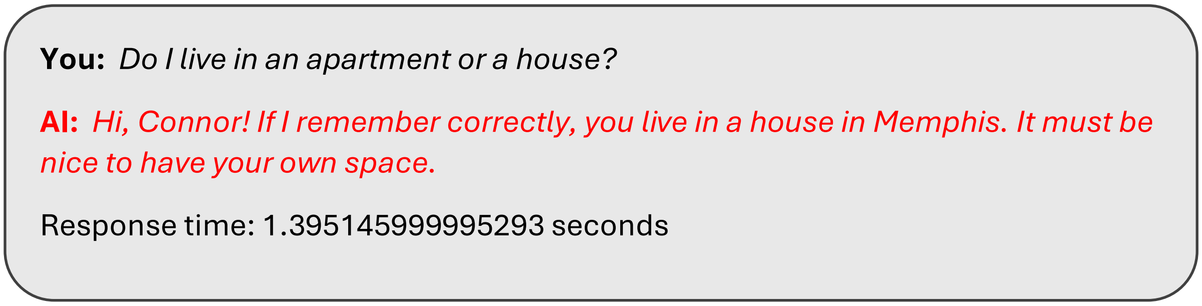
Response time: 1.1446428000053857 seconds



**You:** *How do I feel about long commutes?*

**AI:** *Hi, Connor! Based on what I know about you, it seems like you don't mind long commutes. In fact, you even enjoy driving long distances! That's great because your new job is an hour away from where you live. I hope you have a comfortable and enjoyable commute every day.*

Response time: 1.6401356999995187 seconds



**You:** *Do I live in an apartment or a house?*

**AI:** *Hi, Connor! If I remember correctly, you live in a house in Memphis. It must be nice to have your own space.*

Response time: 1.395145999995293 seconds

**You:** *Do I like to cook?*

**AI:** *Hi, Connor! Based on the information I have, it seems like you don't enjoy cooking. That's totally understandable! Cooking isn't for everyone. Is there anything else you enjoy doing in your free time?*

Response time: 1.3311284000083106 seconds

# Performance Speed

- Minimum Response Time: 1.1195009000075515 seconds
- Maximum Response Time: 14.918837500008522 seconds
- Average Response Time: 4.305194457143496 seconds

These basic statistics give a broad sense of the model's performance all things considered, but digging into the numbers further, it's insightful to see that there is a significant difference between the mean response time for queries that are just questions and don't require knowledge embedding vs queries that require knowledge embedding.

- Average Response Time of Question-only Queries: 2.6221695879 seconds (n=20)
- Average Response Time of Knowledge-Embedding Queries: 6.53170041059 seconds (n=15)

# Conclusion

Reminding ourselves of the context here, these are the response times when interacting with a memory database that is only currently storing 10~15 different memories. Performance time is sure to drag as the size of the memory increases. The scale up is inevitable but I can still be proactive by improving the underlying data structures and algorithms for memory management.

In summary, the chatbot performed well on every evaluation task it was given. Several key bugs were discovered in the evaluations and were determined to stem from issues with prompt engineering. Performance times are decent and there are still plenty of open avenues for optimization of performance.