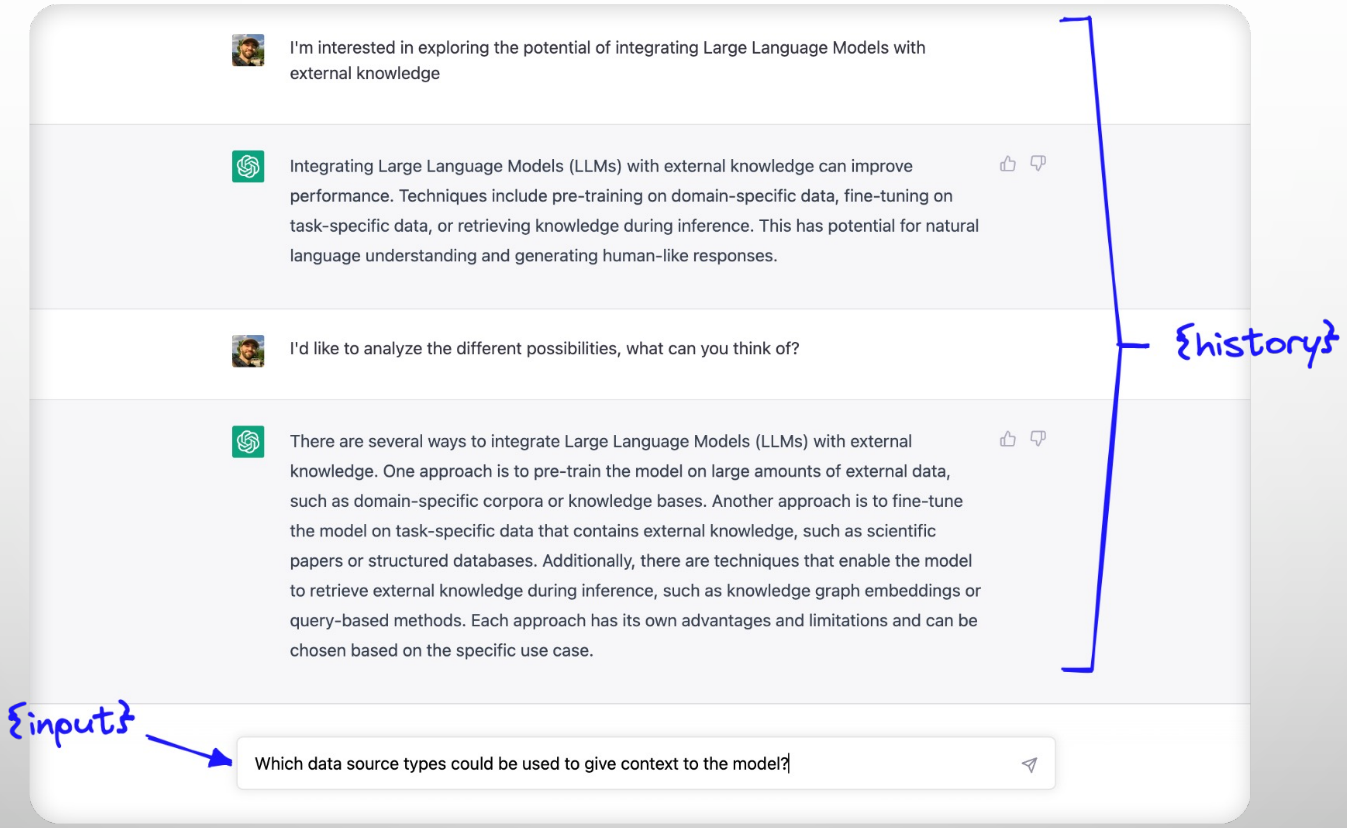
# IMPROVING
# SHORT TERM MEMORY

## Topics

- What's the issue?
- Three approaches / techniques
    - Conversation Buffer
    - Conversation Summary
    - A combination of both
- What about Function results?

## HISTORY & INPUTS


I'm interested in exploring the potential of integrating Large Language Models with external knowledge

Integrating Large Language Models (LLMs) with external knowledge can improve performance. Techniques include pre-training on domain-specific data, fine-tuning on task-specific data, or retrieving knowledge during inference. This has potential for natural language understanding and generating human-like responses.

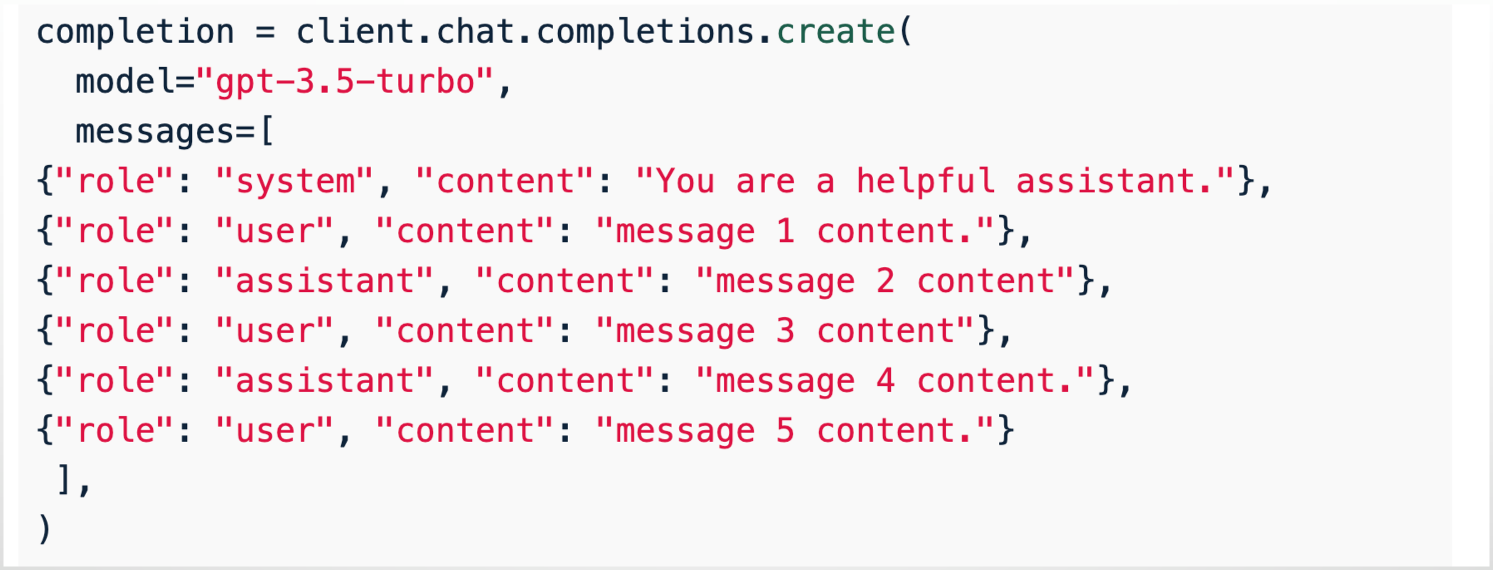
I'd like to analyze the different possibilities, what can you think of?

There are several ways to integrate Large Language Models (LLMs) with external knowledge. One approach is to pre-train the model on large amounts of external data, such as domain-specific corpora or knowledge bases. Another approach is to fine-tune the model on task-specific data that contains external knowledge, such as scientific papers or structured databases. Additionally, there are techniques that enable the model to retrieve external knowledge during inference, such as knowledge graph embeddings or query-based methods. Each approach has its own advantages and limitations and can be chosen based on the specific use case.

Which data source types could be used to give context to the model?

https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/

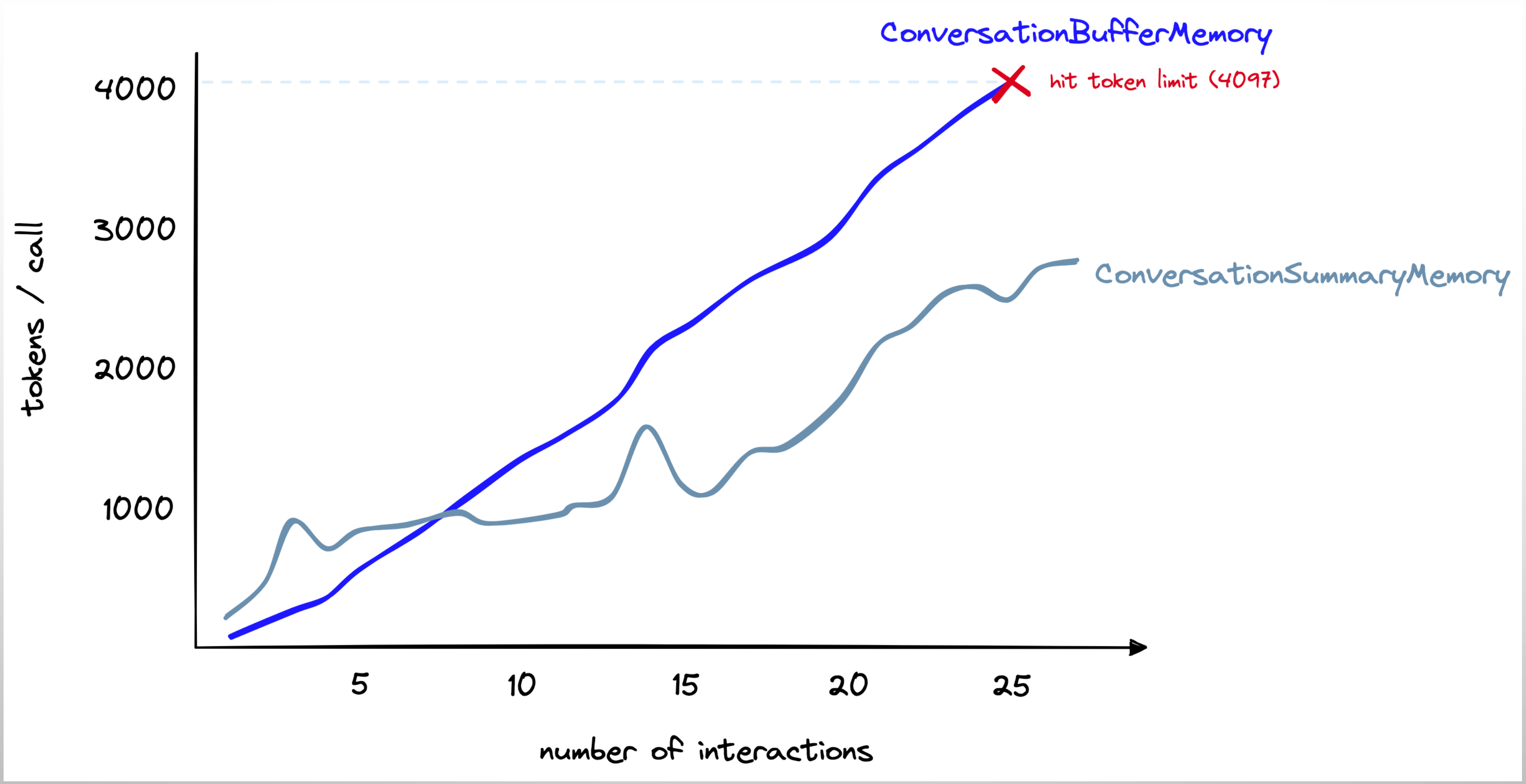
## REMEMBER:
## HOW TO IMPLEMENT A CONVERSATION BUFFER



```
completion = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": "message 1 content."},
{"role": "assistant", "content": "message 2 content"},
{"role": "user", "content": "message 3 content"},
{"role": "assistant", "content": "message 4 content."},
{"role": "user", "content": "message 5 content."}
 ],
)
```

https://community.openai.com/t/how-to-pass-conversation-history-back-to-the-api/697083

# WHAT HAPPENS IN A LONG CONVERSATION?

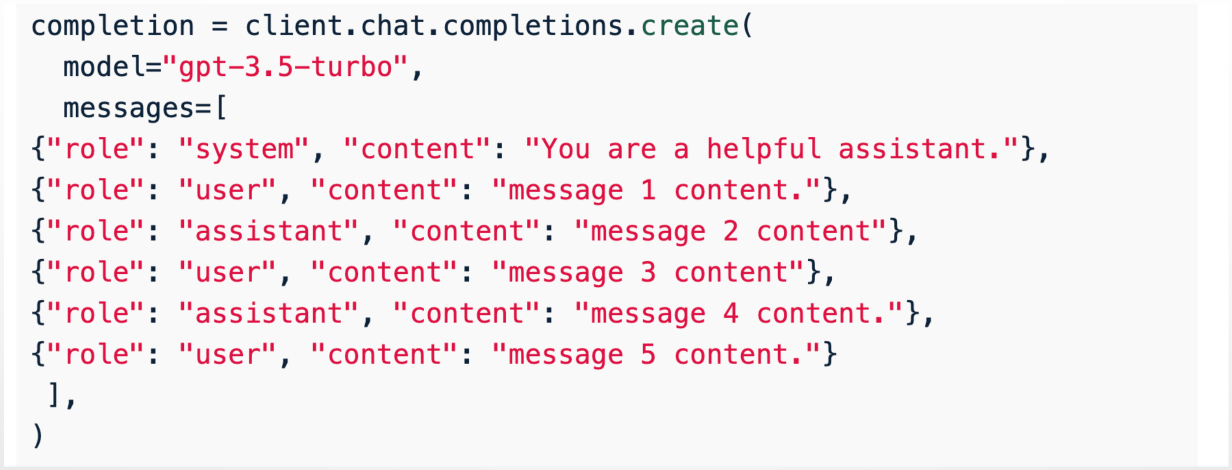


https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/

## QUESTION?

With input context growing – how important is it to have different short term memory techniques (vs just passing the entire conversation)

## WHAT HAPPENS IN A LONG CONVERSATION?



```
completion = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
{"role": "system", "content": "You are a helpful assistant."},
{"role": "user", "content": "message 1 content."},
{"role": "assistant", "content": "message 2 content"},
{"role": "user", "content": "message 3 content"},
{"role": "assistant", "content": "message 4 content."},
{"role": "user", "content": "message 5 content."}
 ],
)
```

**‘messages’ can become too long:**

- There is a limited (but growing) context window
- However, cost/latency is also an issue:
  → longer context windows cost more to process
  → longer context windows take longer to process

https://community.openai.com/t/how-to-pass-conversation-history-back-to-the-api/697083

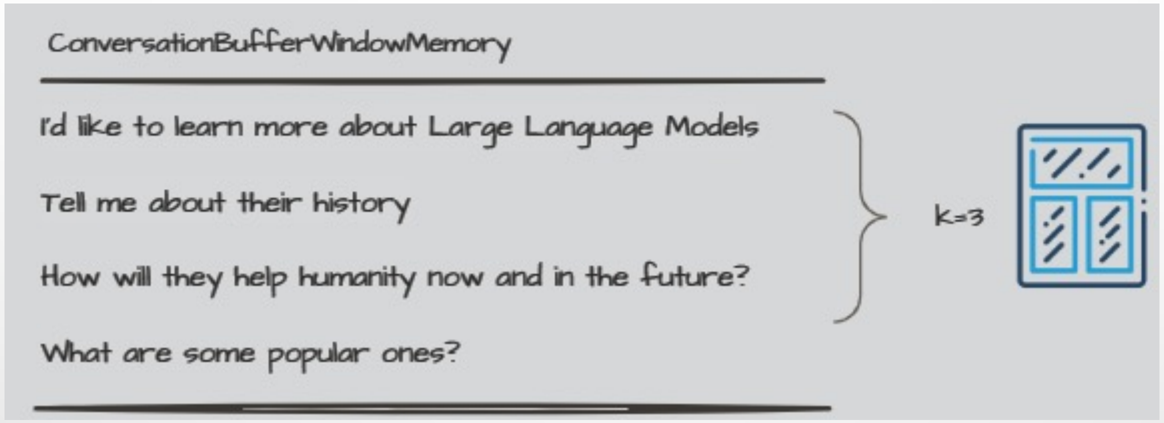
## THREE TECHNIQUES FOR SHORT TERM MEMORY

**Conversation Buffer**

**Conversation Summary**

**Conversation Buffer & Summary**

## CONVERSATION BUFFER


ConversationBufferWindowMemory
I'd like to learn more about Large Language Models
Tell me about their history
How will they help humanity now and in the future?
What are some popular ones?
k=3

**Buffering**: *Pass in the the last N messages*

The chatbot doesn't have memory of the older messages but has complete recent context.

https://medium.com/@michael.j.hamilton/conversational-memory-with-langchain-82c25e23ec60

## LAB 3B

**Create a streaming chatbot with a conversation buffer**

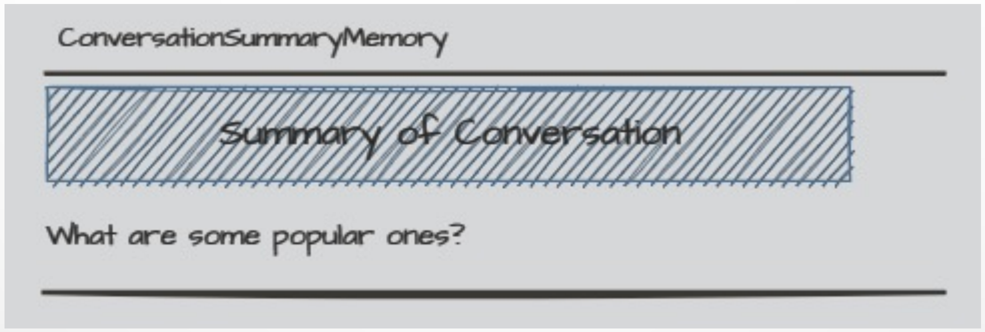
- Keep only the last 2 messages from the user (and the response of those messages from OpenAI).

- This will act as a conversation buffer –
- NOTE: typically, one would have 5 to 10 (at least) messages

**If you have time, create a ‘token-based’ buffer’**

- Calculate the total number of tokens passed to the LLM for each request
- Change the message ‘buffer’ to pass to the LLM at most ‘max_tokens’ (where ‘max_tokens’ is defined in the application)

## CONVERSATION SUMMARY

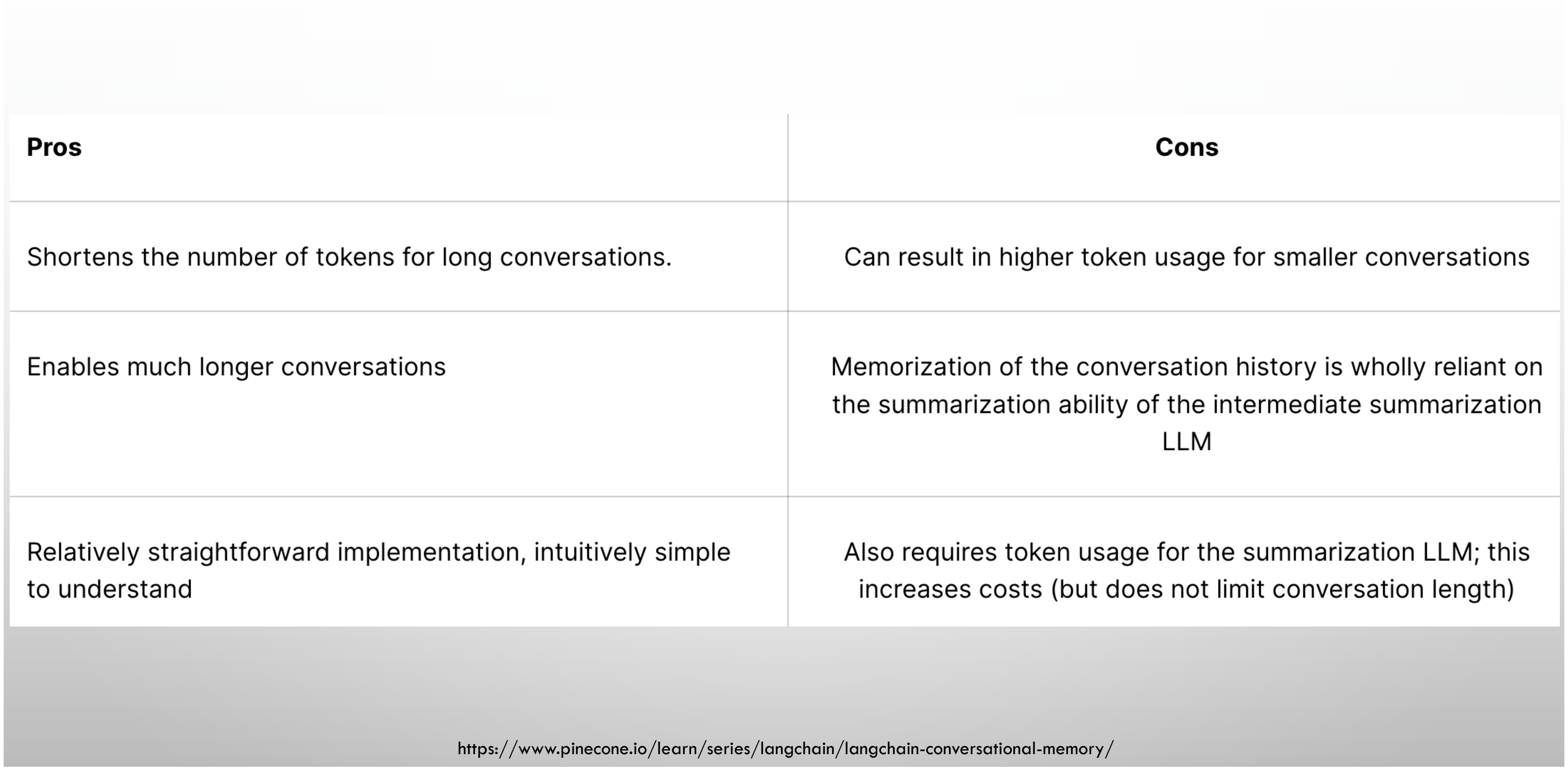


**Summarization**: *Recursively summarize the conversation every time the conversation exceeds some threshold of conversations/messages.*

Retains context from the whole conversation but may lose some details

https://medium.com/@michael.j.hamilton/conversational-memory-with-langchain-82c25e23ec60

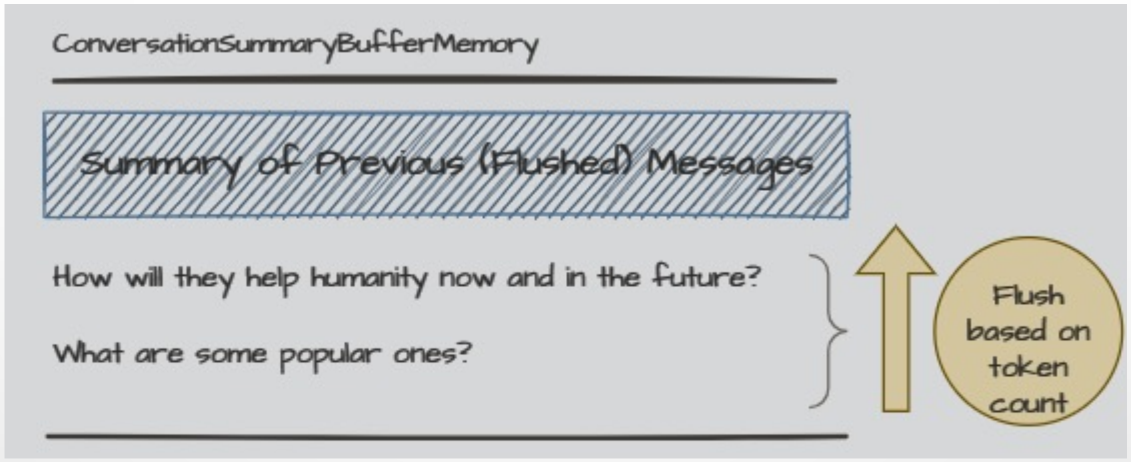
# PROS AND CONS OF CONVERSATION SUMMARY MEMORY



| **Pros** | **Cons** |
|---|---|
| Shortens the number of tokens for long conversations. | Can result in higher token usage for smaller conversations |
| Enables much longer conversations | Memorization of the conversation history is wholly reliant on the summarization ability of the intermediate summarization LLM |
| Relatively straightforward implementation, intuitively simple to understand | Also requires token usage for the summarization LLM; this increases costs (but does not limit conversation length) |

https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/

## CONVERSATION BUFFER & SUMMARY


ConversationSummaryBufferMemory
Summary of Previous (Flushed) Messages
How will they help humanity now and in the future?
What are some popular ones?
Flush based on token count

**Buffer & Summarize**: *Summarize the conversation up to Nth message and also pass in last N messages.*

The best of both worlds -- but has a larger number of tokens in the request.

https://medium.com/@michael.j.hamilton/conversational-memory-with-langchain-82c25e23ec60