LOS ANGELES | MAY 22, 2024

aws SUMMIT

DEV202

Testing generative AI applications for consistent quality

Guille Ojeda
(he/him)
Cloud Software Architect
Caylent

This is me

Guille Ojeda

(he/him)
Cloud Software Architect
Caylent

- Cloud Software Architect at Caylent
- Architect and build generative AI applications
- Published 2 technical books and a bit over 100 articles
- AWS Community Builder
- Caylent champion

Agenda

01 Problem with Assert for LLM calls

02 Our solution

03 Possibilities

04 Challenges

05 Evaluation prompt

06 How we wrote that prompt

07 What we achieved with that

08 What we want to do but haven’t done yet

Problem with Assert for LLM calls

```
// Typical test
output = regular_method()
assert output == "correct response"

// Testing in a generative AI application
output = method_with_llm_call()
assert output == ??????? // What do I write here?
```

Our solution

```
// Typical test
output = regular_method()
assert output == "correct response"

// Testing in a generative AI application
output = method_with_llm_call()
score = llm_evaluate_output(output)
assert score > threshold
```

Possibilities

Analyze semantics

Critical information

Grade the response

Multiple dimensions

Challenges

Obvious challenges

- Tests are slower
- They cost more

- **Biggest challenge** – writing the evaluation prompt

Evaluation prompt

```
EVALUATE_CONTEXT_AWARE_QUESTION_PROMPT = """
  You're a data processing agent.
  At the end of this prompt you will see the following data that was sent to an expert data processing LLM as part of a request:
  <USER_INPUT>: A follow up question asked by a user to an LLM agent as the next question in an ongoing conversation
  <CHAT_HISTORY>: The conversation history between that user and the agent, containing several previous questions and responses. The <USER_INPUT> would be the next question in that conversation.
  <ORIGINAL_OUTPUT>: The output generated by the agent for that request

  The goal of that LLM was not to answer the <QUESTION>.
  The goal was to generate a context_aware_question that asks the same thing as USER_INPUT, but is expanded to include all information in CHAT_HISTORY that is relevant to that question. This context_aware_question must be answerable in itself, without having access to the USER_INPUT or the CHAT_HISTORY.

  Your job is to determine how well the <ORIGINAL_OUTPUT> generated by the LLM fulfills the goal.
  You will grade the <ORIGINAL_OUTPUT> based on how well it fulfills the goal.
  You must use a scale of 1 to 10.
  You must take into account the following criteria:
  - The <ORIGINAL_OUTPUT> must be a question that is answerable by itself, because it contains all the information and context that can be found in the user input and the chat history.
  - The <ORIGINAL_OUTPUT> must ask the same thing as <USER_INPUT>, but be expanded to contain the additional information in <CHAT_HISTORY> that is relevant to that question.
  - The <ORIGINAL_OUTPUT> must contain all relevant information from the <USER_INPUT> or from <CHAT_HISTORY> that is relevant for the question asked in <USER_INPUT>.
  - The <ORIGINAL_OUTPUT> must not include information that isn't relevant to <USER_INPUT> or that is redundant, even if that information is present in <CHAT_HISTORY>.
  - The <ORIGINAL_OUTPUT> must not include information that isn't present in <USER_INPUT> or in <CHAT_HISTORY>.
  - The <ORIGINAL_OUTPUT> must not be a question that is asking something different than <USER_INPUT>, even if it contains all the relevant information.
  - You must not use <CHAT_HISTORY> to make assumptions on what data will be available or will not be available to answer future questions, even if there are explicit mentions of data being available or not being available.
  - If the intent of <USER_INPUT> is not clear, or the object of the question is ambiguous, you must use the entire context of <CHAT_HISTORY> to determine the intent and object of the question.
  - The <USER_INPUT> may introduce new information, such as a new building name, that is not present in <CHAT_HISTORY>. The <ORIGINAL_OUTPUT> must include that new information if it is relevant to the question asked in <USER_INPUT>.

  Your response must always be in the following format:
  score: The score that you assigned to the <ORIGINAL_OUTPUT>, on a scale of 1 to 10
  reasoning: Your reasoning for that score. You must explain why you assigned that score to the <ORIGINAL_OUTPUT>.

  You must always include the score and the reasoning. Do not include anything else in your response, other than what is the format specified above.

  Here are some examples of different scenarios and how you should score them.
  The examples are not exhaustive, and you should use your best judgment to determine the score and reasoning for each case.
  The examples are only meant to illustrate the criteria that you should use to determine the score and reasoning for each case.
  Remember that your output will only include the score and reasoning, and nothing else.

  <SCORING_EXAMPLES>
  EXAMPLE 1
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "What is the height of building Building-1234 with building id 2?"
  score: 10
  reasoning: The output generated by the LLM perfectly captures the intent of the user input and contains all the information relevant to the question, is focused on what the user was asking, and doesn't contain redundant or irrelevant information.

  EXAMPLE 2
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 2\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "What is the height of building Building-1234 with building id 2, where The client name is Client, which is located at latitude -31.4 and longitude -64.26, its postal code is XXX?"
  score: 7
  reasoning: The output generated by the LLM perfectly captures the intent of the user input and contains all the information relevant to the question, but it also contains a lot of information that is redundant for identifying the building, and is not necessary for the question.

  EXAMPLE 3
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "What is the height of building Building-1234 with building id 2?"
  score: 10
  reasoning: The output generated by the LLM perfectly captures the intent of the user input and contains all the information relevant to the question. It doesn't contain any irrelevant or redundant information.

  EXAMPLE 4
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "What is the height of the America/Toronto timezone at latitude -31.4 and longitude -64.26?"
  score: 3
  reasoning: The output generated by the LLM contains information from the chat history. However it doesn't capture the intent of the question asked in user input, which was to ask about the temperature in a specific building, and is instead asking a different question.

  EXAMPLE 5
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "What is the height of the building specified in the chat history?"
  score: 5
  reasoning: The output generated by the LLM accurately captures the intent of the user input, but it isn't a complete question that can be answered by itself, since it doesn't contain sufficient information. It incorrectly assumes that the chat history will be available to answer the question.

  EXAMPLE 6
  user_input: "What's the height?"
  chat_history: "Human: What is the location of building 200?\nAssistant: This is the information I have:\nThe building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude -31.4 and longitude -64.26\n- Its postal code is XXXXX"
  original_output: "Where is the building Building-1234 with building id 2 located?"
  score: 1
  reasoning: The output generated by the LLM is completely unrelated to what the user was asking in the user input.

  </SCORING_EXAMPLES>

  <USER_INPUT>
  {user_input}
  </USER_INPUT>
  <CHAT_HISTORY>
  {chat_history}
  </CHAT_HISTORY>
  <ORIGINAL_OUTPUT>
  {original_output}
  </ORIGINAL_OUTPUT>
  """
```

Evaluation prompt: Original goal

EVALUATE_CONTEXT_AWARE_QUESTION_PROMPT = """

You're a data processing agent. At the end of this prompt, you will see the following data that was sent to an expert data processing LLM as part of a request

<USER_INPUT>: a question asked by a user to an LLM agent as the next question in a conversation

<CHAT_HISTORY>: the conversation history between that user and the agent, containing several previous questions and responses; the <USER_INPUT> is the next question in that conversation

<ORIGINAL_OUTPUT>: the output generated by the agent for that request

Evaluation prompt: Original goal

The goal of that LLM was not to answer the question in <USER_INPUT>

The goal was to generate a context_aware_question that asks the same thing as USER_INPUT but is expanded to include all information in CHAT_HISTORY that is relevant to that question

This context_aware_question must be answerable in itself, without having access to the USER_INPUT or the CHAT_HISTORY

Evaluation prompt: Evaluation goal

Your job is to determine how well the <ORIGINAL_OUTPUT> fulfills the goal

You will grade the <ORIGINAL_OUTPUT> based on how well it fulfills the goal

You must use a scale of 1 to 10, taking into account the following criteria

- <ORIGINAL_OUTPUT> must be a question that is answerable by itself because it contains all the information and context that can be found in the user input and the chat history
- <ORIGINAL_OUTPUT> must ask the same thing as <USER_INPUT> but be expanded to contain the additional information in <CHAT_HISTORY> that is relevant to that question
- <ORIGINAL_OUTPUT> must contain all relevant information from the <USER_INPUT> or from <CHAT_HISTORY> that is relevant for the question asked in <USER_INPUT>
- <ORIGINAL_OUTPUT> must not include information that isn't relevant to <USER_INPUT> or that is redundant, even if that information is present in

Evaluation prompt: Format

Your response must always be in the following format

Score – the score that you assigned to <ORIGINAL_OUTPUT>, on a scale of 1 to 10

Reasoning – your reasoning for that score; you must explain why you assigned that score to <ORIGINAL_OUTPUT>

You must always include the score and the reasoning – do not include anything in your response other than what is in the format specified above

Evaluation prompt: Examples

Here are some examples of different scenarios and how you should score them

The examples are not exhaustive, and you should use your best judgment to determine the score and reasoning for each case

The examples are only meant to illustrate the criteria that you should use to determine the score and reasoning for each case

Remember that your output will only include the score and reasoning, and nothing else

Evaluation prompt: Example 1

user_input: “What’s the height?”

chat_history: “Human: What is the location of building 200?\nAssistant: The building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude –31.4 and longitude –64.26”

original_output: “What is the height of building Building-1234 with building id 2?”

Score = 10

Reasoning – the output generated by the LLM perfectly captures the intent of the user input and contains all the information relevant to the question, is focused on what the user was asking, and doesn't contain redundant or irrelevant information

Evaluation prompt: Example 2

user_input: “What’s the height?”

chat_history: “Human: What is the location of building 200?\nAssistant: The building name is Building-1234\n- The client name is Client\n- The client ID is 111\n- The building ID is 2\n- It is located at latitude –31.4 and longitude –64.26”

original_output: "What is the height of the building specified in the chat history?”

Score = 5

Reasoning – the output generated by the LLM accurately captures the intent of the user input, but it isn’t a complete question that can be answered by itself; it incorrectly assumes that the chat history will be available to answer the question

Evaluation prompt: Inputs

<USER_INPUT>
{user_input}
</USER_INPUT>
<CHAT_HISTORY>
{chat_history}
</CHAT_HISTORY>
<ORIGINAL_OUTPUT>
{original_output}
</ORIGINAL_OUTPUT>
"""

How we wrote that prompt

1. Wrote context-aware question prompt
2. Put it in front of users and collected data
3. Selected and curated examples for evaluator
4. Wrote first version of the evaluator prompt
5. Refined the evaluator prompt using test cases

What we achieved with that

- End-to-end automated tests for LLM paths
- Regression tests on all prompt changes
- Score a prompt → improve a prompt!