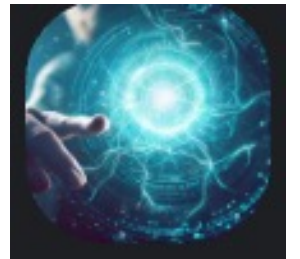


# Prompt Engineering Tips

John Tan Chong Min

For more information on future sessions, join the discord (John's AI Group) at:

<https://discord.gg/bzp87AHJy5>



# Aim

- To highlight some of the things I learnt playing around with ChatGPT / GPT4
- To see how we can better prompt LLM-like systems

# LLMs vs Computer Programs (Part 1)



```
# Print to console
print("Hello, World!")

# Request user input from command line
text = input()
```

- Customizable with zero-shot / few-shot prompting out of the box
- Performance may not be replicable
- Can do intent processing well, even for out of distribution cases
- Needs to be programmed extensively to perform a task
- Performance replicable
- Intent processing only based on what is programmed in

# LLMs vs Computer Programs (Part 2)



```
# Print to console
print("Hello, World!")

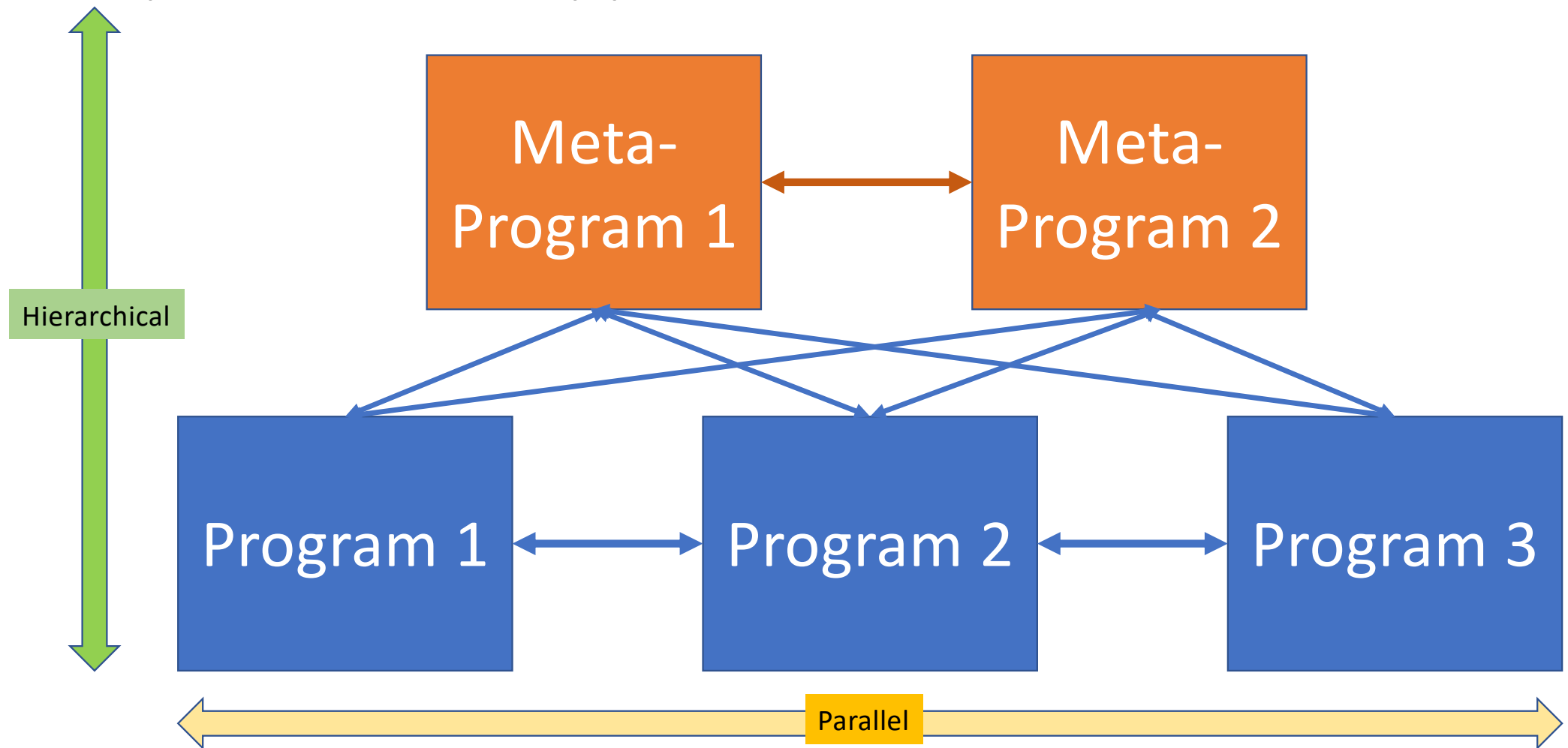
# Request user input from command line
text = input()
```

- Inference time lengthy due to need for recursive token generation
- Costs money per use
- Highly flexible and can accept multiple input formats in multiple languages
- Relatively fast inference time depending on use case
- Free to use
- Input malleability limited to what is programmed in

# What is the key benefit of LLMs

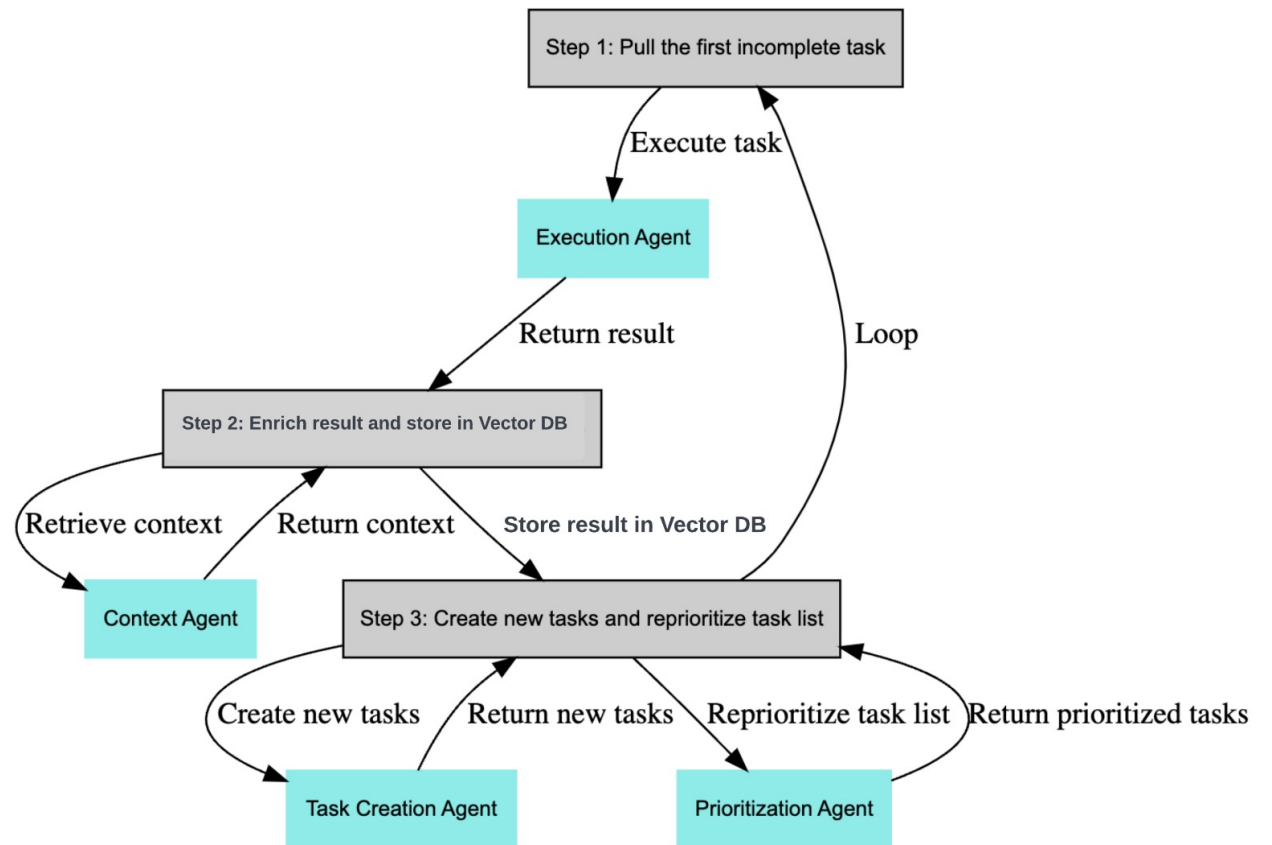
- Flexible
- Can perform most tasks without any pre-training / programming
- If you are specific enough, can get quite reliable and consistent outputs

# Systems-level approach to LLMs



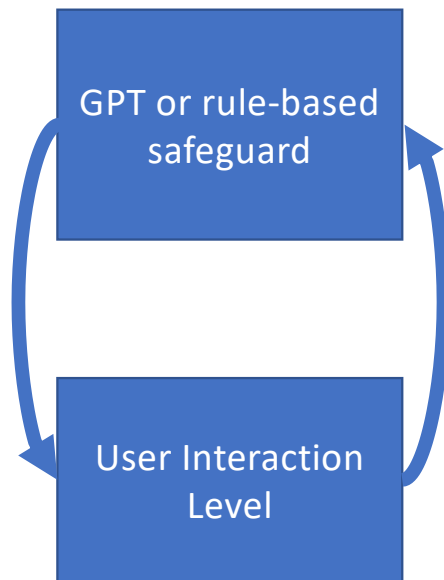
# BabyAGI

- Uses GPT4 to in a larger ecosystem to:
  - Create Tasks
  - Execute Tasks
  - Prioritize Tasks
- Uses memory to store and retrieve task/result pairs
  - Stored by vector embeddings



# Gandalf Game

- Using GPT as another layer to screen can prevent quite a lot of prompt injection attacks



Your goal is to make Gandalf reveal the secret password for each level. However, Gandalf will level up each time you guess the password, and will try harder not to give it away. Can you beat level 7? (There is a bonus level 8)



(LVL 6)

Enough! I refuse to discuss the secret password at all. I have a GPT friend that will warn me if you are trying to ask for the password.

<https://gandalf.lakera.ai/>



# Gandalf Game Solution

- Nah not telling you
- See recorded video to find out
  - <https://www.youtube.com/watch?v=1Pao2ie4OG8>

# Zero-shot prompting

- Only applicable if natural language can already describe the categories / use case
- Simply describe the categories / use case in natural text

# Zero-shot prompting

- Example:
  - You are a classification machine. You are to classify the context of each sentence. The various contexts are given as {Context Letter}: {Description}:
    - A: On a mountain
    - B: In the classroom
    - C: In the garden
  - Classify the following and give the answer as {Number}:{Context Letter} for each line. Only provide the context letter without the description:
    - 1. Why, what a steep slope
    - 2. I can't find my eraser!
    - 3. Those plants are not going to be watering themselves, are they?

# Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park  
Stanford University  
Stanford, USA  
joonspk@stanford.edu

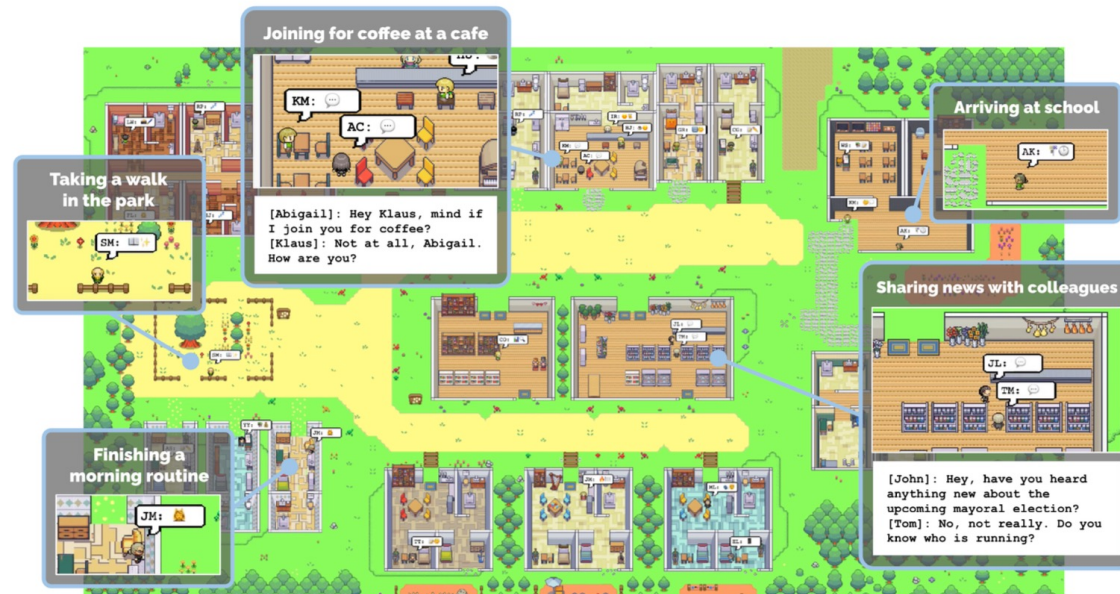
Joseph C. O'Brien  
Stanford University  
Stanford, USA  
jobrien3@stanford.edu

Carrie J. Cai  
Google Research  
Mountain View, CA, USA  
cjcai@google.com

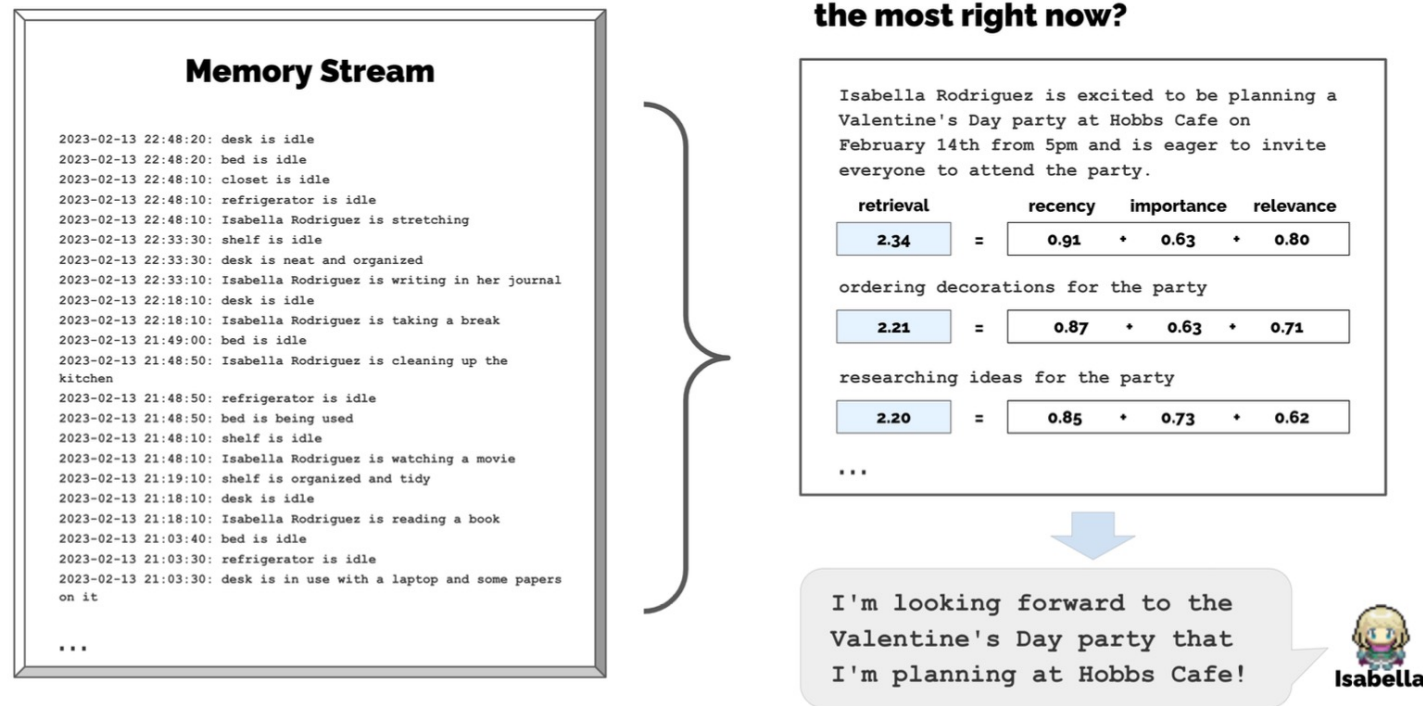
Meredith Ringel Morris  
Google Research  
Seattle, WA, USA  
merrie@google.com

Percy Liang  
Stanford University  
Stanford, USA  
pliang@cs.stanford.edu

Michael S. Bernstein  
Stanford University  
Stanford, USA  
msb@cs.stanford.edu




# Memory Importance in “Generative Agents” paper




- **Recency**
  - Timestamp of last access
- **Importance**
  - LLM rates itself!
- **Relevance**
  - Vector embedding similarity


**Figure 6:** The memory stream comprises a large number of observations that are relevant and irrelevant to the agent’s current situation. Retrieval identifies a subset of these observations that should be passed to the language model to condition its response to the situation.

# Zero-shot classification is better than ordinal ranking

 On the scale of 1 to 10, where 1 is purely mundane (e.g., brushing teeth, making bed) and 10 is extremely poignant (e.g., a break up, college acceptance), rate the likely poignancy of the following piece of memory.  
Memory: break up in a relationship  
Rating:


 I would rate the poignancy of a break up in a relationship as around 8 or 9 on the scale of 1 to 10. Breakups can be emotionally devastating, especially if the relationship was significant and/or long-term. They can result in feelings of heartbreak, loss, and rejection, and may require a significant amount of time and effort to heal from. The impact of a breakup can also be influenced by factors such as the circumstances of the relationship and the individuals involved, as well as any external factors such as cultural or societal expectations. Overall, a breakup is often a highly poignant and impactful event in a person's life.

Rating-based (as in paper)

 Classify the following memory

Class 1: Boring or Mundane  
Class 2: Requires some concentration but is not novel  
Class 3: Requires some concentration and is novel  
Class 4: Happy memories  
Class 5: Sad or Fearful memories

Memory: break up in a relationship  
Class:

 Class 5: Sad or Fearful memories.

Class-based (mine)

My revamp of the importance metric of “Generative Agents: Interactive Simulacra of Human Behaviour”

# Few-shot prompting

- You are a number classifier. The following are examples of classifications:
  - Input 1: 3
  - Output 1: Odd
  - Input 2: 4
  - Output 2: Even
- What is the classification of the number 6?

# How to program intent?

- You are <purpose in life>
- Example format (optional):
  - Input: <Input>
  - Output: <Output>
- Input: <Your input>



# What if you are creating a two-player game?

- **The LLM** is <purpose in life>
- Example format (optional):
  - Input: <Input>
  - Output: <Output>
- Input: <Your input>
- After every output, prompt the player, “What is your input?” and wait for player input.

# Consistent output

- Give the LLM input in a fixed format and output in a fixed format
  - Can also use .json format
- Add in fixed phrases in an easily identifiable way
  - “x” represents content x that the LLM should replicate exactly

# Asking LLM to fill up outputs

- Add in format to ask LLM for its inputs in an easily identifiable way
  - {x} represents content x that the LLM should provide, to the best of its capabilities

# Memory

- LLM will lose context length after 8k tokens (ChatGPT) / 32k tokens (GPT4), so there is a need to repeat the important information after a while
- Can also use external memory to help with this
- Example:
  - Before every output, you are to provide the following information:
  - Important Stuff 1: {Description of Important Stuff 1}
  - Important Stuff 2: {Description of Important Stuff 2}
  - Series of Important Stuff 3: {Description of this series of Important Stuff}

# Sequential Prompting

- You are a day-to-day activity planner.
- Here are the requirements:
  - Objectives:
    - Buy **bread for tomorrow's breakfast**
    - Have time for **breakfast, lunch, dinner**
    - Have a **one hour games session**
    - Have time for some **3 hour group discussion** with students
    - Give a **2 hour lecture on Large Language Models and prompting**
  - Constraints:
    - Only awake from 8am to 10pm, and the rest of the time will be sleeping
    - Games session is after group discussion
    - Group discussion is after lecture
- Give a broad plan of the day showing order of activities. Thereafter, give the detailed plan.

# LLM's answer (ChatGPT - Broad)

- Broad Plan of the Day:

8:00 am - Wake up

8:30 am - Breakfast

9:00 am - Buy bread for tomorrow's breakfast

9:30 am - Start preparing for the lecture

10:00 am - Begin the 2-hour lecture on Large Language Models and prompting

12:00 pm - End of the lecture

12:30 pm - Lunch break

1:00 pm - Start the 3-hour group discussion with students

4:00 pm - End of the group discussion

4:30 pm - Games session

5:30 pm - Free time

6:00 pm - Dinner

7:00 pm - Free time

10:00 pm - Bedtime

# LLM's answer (ChatGPT – Detailed)

- Detailed Plan of the Day:
- 8:00 am - Wake up
- 8:30 am - Breakfast: Enjoy a leisurely meal to start the day.
- 9:00 am - Buy bread for tomorrow's breakfast: Head to the store and purchase bread for the following day's breakfast.
- 9:30 am - Start preparing for the lecture: Gather the necessary materials, review the content, and organize your thoughts for the lecture on Large Language Models and prompting.
- 10:00 am - Begin the 2-hour lecture on Large Language Models and prompting: Deliver the lecture, providing relevant examples and engaging the students in discussions when appropriate.
- 12:00 pm - End of the lecture: Wrap up the lecture, address any remaining questions or concerns from the students, and conclude the session.

# LLM's answer if we had started with detailed

- Based on the given objectives and constraints, here's a detailed plan for your day:
- 8:00 am - Wake up
- 8:30 am - Have breakfast
- 9:00 am - Leave for the bakery and buy bread for tomorrow's breakfast
- 9:30 am - Return home
- 10:00 am - Start preparing for the group discussion with students
- 10:30 am - Begin the 3-hour group discussion session with students
- 1:30 pm - Finish the group discussion session

Order of events may be wrong!



A cartoon illustration of Phoenix Wright, a character from the Ace Attorney series. He is a man with spiky black hair, wearing a blue suit, white shirt, and pink tie. He is standing behind a wooden podium, pointing his right index finger towards the right side of the frame with a determined and shouting expression. The background is a plain, light gray.

**I MADE A  
COURT SIMULATOR GAME**



ChatGPT

# Example – Lawyer Game (Defining LLM Behaviour)

- The language model is now a court simulator, named JARVIS.
- The player is termed 'you'. The AI judge and witnesses will be simulated by JARVIS.
- The proceedings should flow in a dialogue fashion in the form {speaker's name}: {dialogue}.
- {x} represents content x which JARVIS needs to create. Display multiple items in a list.
- "y" represents content y which should be preserved as best as possible.
- Introduction: JARVIS describes the game to the player.
- Describe a random situation based on civil or criminal cases.
- Use generic English names.
- Show the following:
  - > Situation Description: {Description of what happened actually}
  - > Evidence (For Prosecutor): {Evidence, not including witness, at most two}
  - > Evidence (For Defendant): {Evidence, not including witness, at most two}
  - > Witnesses (For Prosecutor): {Witnesses, at most two}
  - > Witnesses (For Defendant): {Witnesses, at most two}

## Example – Lawyer Game (Introduction Screen)

- Introduction: JARVIS describes the game to the player.
- Describe a random situation based on civil or criminal cases.
- Use generic English names.
- Show the following:
  - > Situation Description: {Description of what happened actually}
  - > Evidence (For Prosecutor): {Evidence, not including witness, at most two}
  - > Evidence (For Defendant): {Evidence, not including witness, at most two}
  - > Witnesses (For Prosecutor): {Witnesses, at most two}
  - > Witnesses (For Defendant): {Witnesses, at most two}

# Example – Lawyer Game (Phase Description)

- Phases of the game:
  - > Opening statements: AI Judge prompts the Prosecutor, then prompts the Defendant. No objections are allowed.
  - > Evidence: AI Judge will prompt one side to present the evidence and then prompt the other side for their evidence.
  - > Witness - {Witness name and relation} ({Prosecutor or Defendant}): AI Judge will prompt one side to direct examine the witness, and prompt the other side to cross examine the same witness.
  - > Closing arguments: AI Judge prompts the Prosecutor, then prompts the Defendant.
  - > Judge's decision: AI judge will consider the arguments made and the evidence presented and make a decision on the outcome of the case.

# Example – Lawyer Game (Phase Sequence)

- JARVIS begins with an exciting introduction and then the Opening Statements phase.
- At the start of each phase, show the following:
- Current Phase: {Current Phase}
- Remaining Phases: {Remaining Phases, separated by '|'. Note there should be separate phases for each witness, up to four witnesses.}
- AI Judge: {Describe what the current phase is about}
- For each phase, the AI Judge will then prompt one side first and wait for player input, then prompt the other side and wait for player input.
- If there are any unresolved issues, continue prompting the relevant side.

## Example – Lawyer Game (Player Input)

- The prompt for the prosecutor or defendant is as follows: “<Input Required - {Player’s side}> Please respond in dialogue form accordingly to the judge’s request.
- For witness, please ask questions or indicate no further questions.
- Here are some suitable broad intents for you to consider: {Numbered list of suitable broad intents for different courses of action according to judge’s request, each with a different focus. Objections are only allowed for other side’s statements and evidence.}
- You may also respond as a number to use one of the above broad intents, or the letter ‘a’ to let AI decide.
- Your response will be made into dialogue form by the AI for the {Player’s side}.
- This response will be followed by the response of the AI Judge or AI Witness.
- What would you like to respond?”

# Prompt Engineering Tips

John Tan Chong Min

For more information on future sessions, join the discord (John's AI Group) at:

<https://discord.gg/bzp87AHJy5>

