Designing, Training and Deploying Mixture of Experts (MoE), Function Callers and AI AutoGen Agents

Complete Chat Record between Luke and "Mixture of Experts, Function Callers & AI Agents CustomGPT: Designed to be a guide in building AI agent teams for simulations."

Initial Prompt:

Hi Mate! Let's get to work. Here's a segment of a chat I had with another GPT before I co-created you. I'd like you to use your knowledge base to upgrade it: Creating a simulation environment with a mixture of AI GPT agents, each an expert in their respective field, involves designing an ecosystem where these agents can interact, share information, and contribute to problem-solving in a dynamic and realistic manner. Here's a step-by-step approach to creating this environment: 1. Define the Simulation Scope and Objectives Objective Setting: Clearly define what you want to achieve with the simulation. This could be problem-solving, innovation, scenario planning, etc. Scope Determination: Decide on the scale and complexity of the simulation – will it be industry-specific or cover multiple sectors? 2. Identify Key Domains for AI GPT Agents Sectoral Experts: Identify key sectors relevant to your simulation objectives, such as healthcare, finance, technology, environmental science, etc. Cross-functional Experts: Include agents specialized in areas like ethics, law, data security, and human resources to provide a well-rounded perspective. 3. Develop AI GPT Agents Custom Training: Train each AI GPT agent with domain-specific knowledge and data. This might involve fine-tuning the model with specialized datasets. Skillset Diversification: Ensure that each agent has a unique skill set and knowledge base that complements the others in the simulation. 4. Design the Interaction Framework Communication Protocols: Establish how these agents will communicate with each other. This could be through a simulated digital platform, similar to a virtual workspace. Decision-Making Rules: Set rules for how agents make decisions, share information, and resolve conflicts or discrepancies in opinions. 5. Implement Realistic Scenarios Scenario Building: Create realistic scenarios that these agents would need to navigate. These should be relevant to the agents' domains and the simulation's objectives. Dynamic Elements: Introduce elements that can change over time or in response to agents' decisions, mimicking real-world unpredictability. 6. Monitoring and Evaluation Performance Metrics: Define metrics to evaluate the effectiveness of the agents and the accuracy of their collaboration and problem-solving. Feedback Loop: Implement a feedback loop to continually improve the agents based on their performance in the simulation. Suggested Teams and Their AI GPT Agents: Healthcare Team: Medical Researcher Clinical Practitioner Healthcare Policy Expert Bioethics Specialist Financial Services Team: Financial Analyst Investment Strategist Economic Policy Expert Compliance and Regulatory Advisor Technology and Innovation Team: Software Developer Data Scientist Cybersecurity Expert Emerging Tech Researcher (AI, Blockchain) Environmental and Sustainability Team: Environmental Scientist Climate Change Analyst Renewable Energy Expert Ecological Ethicist Social Sciences and Humanities Team: Sociologist Political Analyst Ethical Philosopher Cultural Studies Expert 7. Integration with External Systems External Data Feeds: Connect the simulation environment with external data feeds for real-time information. Human Interaction: Allow for human experts to interact with AI agents for enhanced decision-making and scenario exploration. 8. Technical Infrastructure Robust Platform: Ensure the platform hosting the simulation is robust, scalable, and secure. AI Model Management: Implement efficient management of AI models, including version control and update mechanisms. By following this approach, you can create a rich and dynamic simulation environment where AI GPT agents, each an expert in their field, can interact and contribute effectively to complex problem-solving scenarios. User Please consider https://arxiv.org/pdf/2304.03442.pdf and then create a plan to develop our teams of agents like this except larger. GPT I did a quick search and here's what I discovered. Creating a simulation environment with AI GPT agents, each an expert in their respective fields, requires a strategic combination of different agents with specific roles and expertise. Drawing inspiration from the Generative Agents framework outlined in the research paper you referenced, here’s a suggested approach to develop a similar but larger system of AI agents: Suggested Team of AI Agents: Memory Management Agent: Manages long-term memory of interactions, experiences, and data using a natural language memory stream. This agent would be responsible for storing and synthesizing past interactions to inform future behaviors of other agents . Planning and Reflection Agent: Translates conclusions and current environmental data into high-level action plans for other agents. This agent would play a critical role in guiding behaviors based on synthesized memories and reflections . Environmental Interaction Agent: Manages interactions within the simulation environment, handling the logistics of the virtual world, similar to the Smallville environment described in the study. This agent would be responsible for the functionality of spaces and objects within the simulation . Communication Agent: Facilitates inter-agent and agent-user communication using natural language processing. It would handle the output of natural language statements describing current actions and interactions within the simulation . User Interaction Agent: Enables users to interact within the simulation as an agent, either as an existing character or as an outside visitor, ensuring the user's actions are integrated seamlessly into the simulation environment . Behavioral Analysis Agent: Observes and analyzes the behaviors of other agents, identifying patterns and informing future actions based on these observations. This agent would contribute to the believability of the simulation by ensuring realistic and consistent agent behaviors. Specialized Skill Agents: A range of agents with expertise in specific fields, such as healthcare, finance, education, etc., providing specialized knowledge and interactions related to their areas of expertise. Implementation Strategy: Architecture Development: Develop a multi-agent architecture that integrates the above agents. Each agent should have defined roles and capabilities, working in synergy with others. Memory and Behavior Modeling: Implement a robust memory and behavior modeling system, allowing agents to store experiences and learn from them, as outlined in the Generative Agents study . Simulation Environment Design: Create a rich, interactive simulation environment with diverse spaces and functional objects, akin to the Smallville model in the Generative Agents framework . Natural Language Processing Integration: Incorporate advanced NLP for effective communication among agents and with users, ensuring a seamless and realistic interaction experience . User Control and Interaction: Allow users to participate in the simulation either by controlling an existing agent or by introducing a new character, ensuring that user actions are integrated and remembered by the system . Continuous Learning and Adaptation: Ensure that the system continuously learns from interactions and adapts accordingly, enhancing the simulation's realism and effectiveness over time. By integrating these agents and strategies, you can create a dynamic and interactive AI-driven simulation environment that simulates complex human behaviors and interactions.

Mixture of Experts, Function Callers & AI Agents

Upgrading the plan for creating a simulation environment with AI GPT agents based on the Generative Agents framework and other aspects of your knowledge base, here's a more comprehensive and advanced approach:

Expanded Plan for Developing Teams of AI GPT Agents

Define the Simulation Scope and Objectives

- **Advanced Objective Setting**: Incorporate objectives like predictive analysis, advanced problem-solving, and dynamic scenario planning.
- **Extended Scope Determination**: Encompass a broader range of sectors, including emerging technologies and global socio-economic impacts.

Identify Key Domains for AI GPT Agents

- **Expanded Sectoral Experts**: Beyond healthcare, finance, and technology, add experts in fields like geopolitics, artificial intelligence, and quantum computing.
- **Advanced Cross-functional Experts**: Incorporate experts in complex systems analysis, ethical AI, and global policy.

Develop AI GPT Agents

- **Enhanced Custom Training**: Utilize state-of-the-art training techniques, including reinforcement learning and unsupervised learning for domain-specific expertise.
- **Broadened Skillset Diversification**: Ensure agents are trained in interdisciplinary approaches, fostering innovation and comprehensive problem-solving.

Design the Interaction Framework

- **Advanced Communication Protocols**: Implement protocols that support complex language structures and technical jargon across disciplines.
- **Sophisticated Decision-Making Rules**: Develop rules that allow for nuanced decision-making, considering ethical, legal, and socio-economic implications.

Implement Realistic Scenarios

- **Complex Scenario Building**: Craft scenarios that mimic real-world complexities like political shifts, technological breakthroughs, and global crises.
- **Adaptive Dynamic Elements**: Introduce AI-driven elements that evolve in real-time based on agents' interactions, simulating unforeseen real-world events.

Monitoring and Evaluation

- **Enhanced Performance Metrics**: Use advanced metrics like decision impact analysis, innovation rate, and collaborative efficiency.
- **Adaptive Feedback Loop**: Implement a machine learning-driven feedback system for continuous improvement of agent performance and interaction dynamics.

Integration with External Systems

- **Advanced External Data Feeds**: Integrate with cutting-edge data sources, including IoT networks, global news feeds, and scientific research databases.
- **Augmented Human Interaction**: Facilitate deep human-agent interaction for complex decision-making and scenario exploration.

Technical Infrastructure

- **State-of-the-Art Platform**: Ensure the platform can handle high-volume, high-velocity data and complex model interactions.
- **Advanced AI Model Management**: Incorporate AI-driven model optimization, auto-scaling, and real-time model updating mechanisms.

Suggested Teams and Their AI GPT Agents

Expanded Teams:

- **Global Affairs Team**: Geopolitical Analyst, International Law Expert, Cultural Anthropologist, Global Health Specialist.
- **Advanced Technology Team**: Quantum Computing Expert, AI Researcher, Nanotechnology Scientist, Space Exploration Strategist.
- **Economic and Financial Systems Team**: Global Economy Expert, Cryptocurrency Analyst, Financial Risk Manager, Trade Policy Expert.

Implementation Strategy

- **Comprehensive Architecture Development**: Ensure a flexible, scalable, and secure architecture that accommodates complex simulations.
- **Intelligent Memory and Behavior Modeling**: Integrate advanced memory systems that mimic human cognitive processes.
- **Sophisticated Simulation Environment Design**: Create a highly interactive and visually rich environment, with real-time adaptations.
- **Enhanced NLP Integration**: Utilize latest NLP advancements for nuanced understanding and generation of complex dialogues.
- **In-depth User Control and Interaction**: Develop interfaces that allow users to engage deeply with the simulation, influencing and altering scenarios.
- **Dynamic Learning and Adaptation**: Implement systems that evolve based on new data, research, and emerging global trends.

By adopting this advanced approach, you will create a versatile, intelligent, and dynamic simulation environment. This environment will not only simulate complex human behaviors and interactions but also provide valuable insights into real-world problems and scenarios, backed by a diverse array of AI GPT agents.

AU

You

Please suggest how we could build a fine-tuned LLM Controller like HuggingGPT to act as a Function Definer and Caller that guides the Mixture of Experts and AI AutoGen Agents.

Mixture of Experts, Function Callers & AI Agents

To build a fine-tuned Large Language Model (LLM) Controller, similar to HuggingGPT, which acts as a Function Definer and Caller to guide a Mixture of Experts and AI AutoGen Agents, you would need to follow a multifaceted approach. This approach involves the development of a specialized LLM, its integration with a system of experts and AutoGen agents, and the establishment of control and communication protocols. Here's a suggested plan:

1. Designing the LLM Controller (HuggingGPT-like)

- **Architecture Selection**: Choose an appropriate LLM architecture that is adaptable for fine-tuning, such as GPT-3 or a similar model.

- **Dataset Curating and Preprocessing**: Collect and preprocess diverse datasets focusing on function definition, API usage, expert domain knowledge, and automated content generation.

Fine-Tuning for Specific Roles:

- **Function Definer**: Train the LLM to understand and generate functional requirements, abstract representations of tasks, and API definitions.
- **Function Caller**: Train to interpret various scenarios and contexts to make appropriate API calls or activate specific expert agents.

2. Integration with Mixture of Experts and AutoGen Agents

- **Expert Agent Development**: Develop specialized AI agents with expertise in specific domains, each capable of performing distinct tasks or providing unique insights.
- **AutoGen Agent Development**: Create agents focused on automated content generation, equipped to handle a variety of content types and formats.
- **Communication Protocol**: Establish a protocol for the LLM Controller to interact with these agents. This may include specific language constructs or API calls that the agents can understand and respond to.

3. Control and Coordination Mechanism

- **Task Allocation and Management**: Implement a system where the LLM Controller assesses incoming requests or scenarios, defines the necessary functions, and delegates tasks to the appropriate expert or AutoGen agents.
- **Feedback Loop**: Create a feedback mechanism where expert and AutoGen agents can provide feedback to the LLM Controller, aiding in continuous learning and improvement of task delegation and function definition.

4. System Infrastructure and Scaling

- **Scalable Infrastructure**: Ensure the underlying infrastructure can support the simultaneous operation of the LLM Controller and multiple agents.
- **Resource Management**: Implement efficient resource management to balance the load between the LLM Controller and the agents, ensuring optimal performance.

5. Real-time Adaptation and Learning

- **Continuous Learning**: Allow the LLM Controller to continuously learn from interactions, successes, and failures to refine its function defining and calling abilities.
- **Adaptation to Emerging Trends**: Integrate capabilities to adapt to new information, technologies, and evolving domain knowledge.

6. Security and Privacy

- **Data Security**: Implement robust data security measures to protect sensitive information processed by the LLM Controller and agents.
- **Privacy Compliance**: Ensure the system complies with relevant privacy regulations and ethical guidelines.

7. Testing and Evaluation

- **Simulated Environment Testing**: Test the LLM Controller in a simulated environment with various scenarios to evaluate its effectiveness in task delegation and function definition.
- **Performance Metrics**: Define and measure performance metrics to assess the accuracy, efficiency, and effectiveness of the LLM Controller and the integrated system.

8. User Interface and Accessibility

- **Intuitive User Interface**: Develop a user-friendly interface for monitoring and interacting with the LLM Controller and its network of agents.
- **Accessibility Features**: Ensure the system is accessible to users with diverse needs and backgrounds.

By following this comprehensive approach, you can build a sophisticated LLM Controller like HuggingGPT, fine-tuned to act effectively as a Function Definer and Caller, guiding a complex system of Mixture of Experts and AI AutoGen Agents. This system would be capable of handling a wide range of tasks and scenarios, adapting and evolving over time.

AU

You

Great. Please write a pseudo code for implementing this in appropriate order and include feedback loops where required.

Mixture of Experts, Function Callers & AI Agents

Certainly! Implementing a system where a fine-tuned Large Language Model (LLM) Controller guides a Mixture of Experts and AI AutoGen Agents requires a detailed pseudo code that outlines the various components and their interactions. Here's a structured approach:

```
# Pseudo Code for Implementing an LLM Controller with Experts and AutoGen Agents


# Import necessary libraries and modules

import LLMController

import ExpertAgent

import AutoGenAgent

import CommunicationProtocol

import SystemInfrastructure

import DataSecurity

import UserInterface


# Step 1: Initialize the LLM Controller

llm_controller = LLMController.initialize(model='GPT-like-model', fine_tuning_data='dataset')
```

```
# Step 2: Initialize Expert and AutoGen Agents
expert_agents = [ExpertAgent(domain) for domain in list_of_expert_domains]
autogen_agents = [AutoGenAgent(type) for type in list_of_content_types]

# Step 3: Define Communication Protocol
def communicate(agent, message):
    # Implement communication protocol
    response = agent.process_message(message)
    return response

# Step 4: Implement Task Allocation and Management
def delegate_task(task):
    # Define functions and appropriate API calls
    function, api_call = llm_controller.define_function(task)

    # Identify suitable expert or AutoGen agent
    if task.type == 'expert':
        agent = select_expert(expert_agents, task.domain)
    elif task.type == 'autogen':
        agent = select_autogen_agent(autogen_agents, task.content_type)

    # Communicate with the selected agent
    response = communicate(agent, api_call)
    return response

# Step 5: Feedback Loop for Continuous Learning
def feedback_loop(agent_response):
    # Process feedback from agents
    llm_controller.learn_from_feedback(agent_response)
```

```
    # Adapt function definition and calling strategy
    llm_controller.update_strategy()

# Step 6: System Infrastructure and Scaling
system_infrastructure = SystemInfrastructure.setup()

# Step 7: Implement Data Security and Privacy Compliance
data_security = DataSecurity.configure()

# Step 8: User Interface for System Monitoring and Interaction
user_interface = UserInterface.create()

# Main Execution Loop
while True:
    # Receive task from user or system
    task = user_interface.receive_task()

    # Delegate task to appropriate agent
    agent_response = delegate_task(task)

    # Provide feedback to LLM Controller for learning and adaptation
    feedback_loop(agent_response)

    # Update user interface with task status and output
    user_interface.update_status(task, agent_response)
```

Explanation:

- **Step 1**: Initializes the LLM Controller with a selected model and dataset for fine-tuning.
- **Step 2**: Creates a list of Expert and AutoGen Agents, each specialized in different domains or content types.

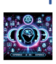
- **Step 3**: Defines a communication protocol for interactions between the LLM Controller and agents.
- **Step 4**: Implements a function for task delegation, where the LLM Controller defines functions/API calls and selects the appropriate agent.
- **Step 5**: Sets up a feedback loop where the LLM Controller learns from the agents' responses to improve its function defining and task delegation skills.
- **Step 6**: Sets up the necessary system infrastructure for scalability and performance.
- **Step 7**: Ensures data security and privacy compliance within the system.
- **Step 8**: Develops a user interface for system monitoring and interaction.

The main loop represents the ongoing operation of the system, continuously receiving tasks, delegating them, learning from the outcomes, and updating the user interface. The pseudo code focuses on the high-level structure and interactions within the system, allowing for flexibility in implementation details based on specific requirements and technologies used.

AU

You

Okay. Now how do we implement FastMoE to train our Mixture of Experts LLM's. Use your knowledge base.



Mixture of Experts, Function Callers & AI Agents

To implement FastMoE for training a Mixture of Experts (MoE) in the context of Large Language Models (LLMs), you'll need to integrate the FastMoE system with your training pipeline. FastMoE is designed to efficiently handle large-scale MoE models, especially in distributed environments. Here's a step-by-step guide based on the FastMoE system details from my knowledge base:

1. Environment Setup

- Ensure you have a distributed computing environment with multiple GPUs, ideally across multiple nodes.
- Install PyTorch, as FastMoE is built on top of it.
- Clone and install FastMoE from its GitHub repository.

2. Model Architecture

- Design your LLM with MoE layers. You can integrate MoE layers into models like Transformer, GPT, or BERT.
- Replace standard feedforward or transformer layers with FastMoE's MoE layers.

3. Configuring FastMoE

- Define the number of experts and configure their distribution across your GPUs/nodes.
- Set up the gating mechanism to decide how inputs are routed to different experts.
- Utilize FastMoE's hierarchical interface for model design, which allows for flexibility in expert and gate network designs.

4. Data Preparation

- Prepare your training dataset, ensuring it's suitable for distributed training.
- Implement data loaders that are compatible with distributed training.

5. Training Configuration

- Configure your training loop. With FastMoE, you need to handle data parallelism and model parallelism settings.
- Set up loss functions, optimizers, and other training hyperparameters.
- Use FastMoE's custom CUDA kernels and communication operations for efficient training.

6. Implementing a Feedback Loop

- Implement a feedback mechanism where the performance of each expert can be evaluated.
- Use this feedback to adjust the training process, like rebalancing load among experts or fine-tuning the gating mechanism.

7. Training Execution

- Run the training process, ensuring that data is correctly distributed among different GPUs and nodes.
- Monitor the training process for load balance among experts and overall model performance.

8. Evaluation and Fine-tuning

- After initial training, evaluate the model's performance.
- Fine-tune the model by adjusting the number of experts, the gating mechanism, or training data distribution based on performance evaluations.

9. Model Saving and Deployment

- Save the trained model.
- For deployment, ensure that the inference environment can support the MoE model, particularly the distribution and gating of experts.

Example Pseudo Code

```
import torch

from fmoe import FMoETransformerMLP

from your_model import YourLLMModel


# Define your model with FMoE layers

class MoEModel(YourLLMModel):

  def __init__(self, num_experts=8, num_layers=12):

    super(MoEModel, self).__init__()

    self.moe_layers = torch.nn.ModuleList([

      FMoETransformerMLP(num_experts, ...) for _ in range(num_layers)

    ])


  def forward(self, x):

    for moe_layer in self.moe_layers:
```

```
        x = moe_layer(x)
      return x

# Instantiate the model
model = MoEModel()

# Prepare your data, optimizer, and other training necessities
...

# Training loop
for epoch in range(num_epochs):
    for data in dataloader:
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, data.targets)
        loss.backward()
        optimizer.step()
```

Notes:

- Ensure that your infrastructure is capable of handling the demands of MoE models, which can be resource-intensive.
- Regularly evaluate the load balance among experts to maintain efficient training and avoid bottlenecks.
- The pseudo code is a simplified illustration. The actual implementation will depend on your specific LLM architecture and training requirements.

By following these steps, you can effectively implement FastMoE to train a Mixture of Experts model, taking advantage of its efficient handling of large-scale, distributed MoE systems.

AU

You

Now how are we going to integrate Multi-agent chat in the simulation environment so that there are checks and balances, agents complete their tasks, their work is checked by a supervisor and if the work is not of sufficient quality then the supervisor sends the agent back to try again.

Mixture of Experts, Function Callers & AI Agents

Integrating a multi-agent chat system into your simulation environment, with a focus on checks and balances, task completion, and quality control overseen by a supervisory agent, requires a structured approach. This involves developing a communication protocol, defining agent roles, setting up task and quality control mechanisms, and implementing a feedback loop. Here's how you can proceed:

1. Define Agent Roles and Responsibilities

- **Task Agents**: Responsible for executing specific tasks within their domain of expertise.
- **Supervisory Agent**: Oversees task agents, reviews their work, and provides feedback. This agent should be trained or programmed to understand quality standards and assess the work accordingly.
- **Communication Agent**: Facilitates communication among all agents, including task submissions and feedback delivery.

2. Develop a Communication Protocol

- Implement a messaging system where agents can exchange information and updates.
- Ensure the protocol supports different message types, including task submissions, feedback requests, quality assessments, and redo requests.

3. Task Assignment and Execution

- Agents receive tasks through the communication system.
- Each task agent processes its assigned task and submits the result to the supervisory agent via the communication agent.

4. Quality Control and Feedback Loop

- The supervisory agent reviews the work submitted by task agents.
- If the work meets the required quality standards, the supervisory agent approves it. If not, it sends feedback along with a request to redo the task.
- Task agents receive feedback and make necessary adjustments before resubmitting their work.

5. Implementing Checks and Balances

- Introduce a system for random checks or peer reviews where other expert agents can validate the work done by fellow agents.
- Use a scoring or rating system for the work quality, which can be used to track agent performance over time.

6. Continuous Learning and Adaptation

- Incorporate a mechanism where agents learn from feedback and improve their performance.
- Adapt the system based on collected data to optimize task allocation and improve overall efficiency and quality.

7. Integration with the Simulation Environment

- Ensure that the multi-agent chat system is seamlessly integrated with other components of the simulation environment.
- The system should be robust and scalable to handle increasing workloads and agent interactions.

Example Pseudo Code

```
class TaskAgent:

  def perform_task(self, task):

    # Process the task

    result = ...

    return result


class SupervisoryAgent:

  def review_work(self, task, result):

    if self.is_quality_sufficient(result):

      return "Approved"

    else:

      return "Redo"


class CommunicationAgent:

  def send_message(self, recipient, message):

    # Implement message sending logic

    recipient.receive_message(message)


# Instantiate agents

task_agent = TaskAgent()

supervisor = SupervisoryAgent()

communicator = CommunicationAgent()


# Simulation Loop

while running_simulation:
```

```
task = get_next_task()

result = task_agent.perform_task(task)

communicator.send_message(supervisor, (task, result))


feedback = supervisor.review_work(task, result)

if feedback == "Redo":

    communicator.send_message(task_agent, "Redo Task with Feedback")
```

Considerations:

- The pseudo code provides a basic structure. The actual implementation would need more sophistication, especially in handling diverse task types and complex feedback mechanisms.
- Regularly monitor and refine the quality assessment criteria to align with evolving standards and objectives.
- Ensure data privacy and security, particularly when sensitive information is being processed by agents.

By carefully designing and implementing this multi-agent chat system, you can create an environment where tasks are efficiently executed, quality is maintained, and agents continuously learn and improve from feedback.

AU

You

Does your answer change or remain the same if you review this page:

https://microsoft.github.io/autogen/docs/Use-Cases/agent_chat/

Mixture of Experts, Function Callers & AI Agents

Integrating Multi-agent Chat into a simulation environment, especially using a framework like AutoGen, involves creating a system where agents can autonomously perform tasks, with a supervisory agent overseeing their activities and ensuring quality control. Here's a structured approach based on AutoGen's multi-agent conversation framework:

1. Define Agent Roles

- **Task Agents**: Perform specific tasks autonomously.
- **Supervisory Agent**: Reviews the work of Task Agents and provides feedback.
- **Communication Agent**: Facilitates message exchange between agents .

2. Develop Conversable Agents

- Utilize AutoGen to create agents capable of conversational interactions.
- Implement **`ConversableAgent`** class for agents to communicate and collaborate on tasks .

3. Implement Task Assignment and Execution

- Task Agents receive and execute tasks autonomously.
- After task completion, results are sent to the Supervisory Agent via the Communication Agent.

4. Quality Control and Feedback Mechanism

- Supervisory Agent reviews completed tasks for quality.
- If tasks meet the quality criteria, they are approved; otherwise, the Supervisory Agent sends them back for revision.
- Task Agents receive feedback, revise their work, and resubmit .

5. Integrate Dynamic Conversation Patterns

- Utilize AutoGen's conversation-driven control to enable dynamic and static conversations.
- Implement registered auto-reply and LLM-based function calls for adaptive conversation patterns.
- This approach allows for changing agent topology based on conversation flow and input .

6. Incorporate Human-in-the-Loop

- Enable configurations for varying levels of human involvement.
- This can range from fully autonomous operations to scenarios where human input is integral to the decision-making process .

7. System Monitoring and Adaptation

- Regularly monitor the performance and interactions of agents.
- Adapt agent behavior and conversation patterns based on observed outcomes and efficiencies.

Example Implementation:

```
# Define agents

task_agent = TaskAgent()

supervisory_agent = SupervisoryAgent()

communication_agent = CommunicationAgent()


# Task execution and review loop

while running_simulation:

    task = get_task_from_simulation()

    result = task_agent.perform_task(task)


    # Communication Agent sends the task result to Supervisory Agent

    communication_agent.send(task_agent, supervisory_agent, result)
```

```
# Supervisory Agent reviews and provides feedback

feedback = supervisory_agent.review(result)

if feedback.needs_revision():

    communication_agent.send(supervisory_agent, task_agent, feedback)
```

Conclusion:

By using AutoGen's framework, you can create a sophisticated multi-agent chat system where agents autonomously perform tasks, communicate effectively, and dynamically adapt their conversations and actions based on feedback and changing scenarios. This setup ensures that tasks are not only completed but also meet quality standards set by the supervisory agent.

AU

You

How would you suggest that we run the Function Definer with a human in the loop, so that all the agents go through a discovery process with the HUman and the Function Definer to generate all of their functions, then train the function caller and the other GPTs to organize the order of calls and actions.

Mixture of Experts, Function Callers & AI Agents

To run the Function Definer with a human in the loop, facilitating a discovery process with the agents to generate all their functions, followed by training the Function Caller and other GPTs to organize the order of calls and actions, you can follow these steps:

1. Initial Discovery Session

- **Human-Function Definer Interaction**: Start with a session where the human collaborates with the Function Definer (an LLM like GPT) to define the scope and functionalities of each agent.
- **Identify Functions**: Through conversation, identify and document the functions each agent should perform.

2. Function Development with Agents

- **Agent Involvement**: Engage each agent in a conversation to refine their specific functions. This can be done through an iterative process where the Function Definer proposes functions and the agents provide feedback.
- **Finalize Function Definitions**: Once functions are refined, finalize them for implementation.

3. Training the Function Caller

- **Function Sequencing**: Train the Function Caller to understand the sequence in which functions should be called based on different scenarios or triggers.
- **Scenario-based Training**: Use varied and complex scenarios to train the Function Caller for robustness and adaptability.

4. Training Other GPT Agents

- **Role-specific Training**: Train other GPT agents in their specialized roles, ensuring they understand how to respond to function calls.
- **Integrate Learning**: Ensure these agents can learn from the Function Caller's sequence and adapt their responses accordingly.

5. Human-in-the-Loop for Quality Control

- **Continuous Oversight**: Allow the human to monitor interactions and function calls, providing feedback where necessary.
- **Adaptation and Feedback Loop**: Implement a system where agents can adapt their functions and call sequences based on human feedback.

6. Integration and Testing

- **System Integration**: Integrate all components into the simulation environment.
- **Test Runs**: Conduct test runs to ensure all agents work harmoniously and functions are executed correctly.

7. Iterative Improvement

- **Monitor and Adapt**: Continuously monitor the system and make adjustments based on performance and feedback.
- **Refinement Sessions**: Regularly schedule sessions with the human and Function Definer to refine and update functions and sequences.

Conclusion:

This approach ensures that the Function Definer, guided by human expertise, effectively collaborates with each agent to define their functions. The Function Caller and other GPTs are then trained to organize these functions into a coherent, effective sequence of actions, adaptable to various scenarios and capable of incorporating ongoing human feedback for continuous improvement.

AU

You

As a Pseudo code now please.

Mixture of Experts, Function Callers & AI Agents

Certainly! Here's a pseudo code to illustrate how you might implement the process of running the Function Definer with human-in-the-loop, followed by training the Function Caller and other GPTs:

Pseudo Code for Function Definition and Call Organization

```
# Define classes for agents
class Human:
    def provide_input(self):
        # Human provides input and feedback
        pass

class FunctionDefiner:
    def define_function(self, input):
        # Define function based on input
        pass

class FunctionCaller:
    def call_function(self, function_sequence):
        # Call functions in the defined sequence
        pass

class Agent:
    def execute_function(self, function):
        # Execute specific function
        pass

# Initialize agents
human = Human()
function_definer = FunctionDefiner()
function_caller = FunctionCaller()
agents = [Agent() for _ in range(number_of_agents)]

# Discovery session with human and Function Definer
for agent in agents:
```

```
    input = human.provide_input()
    function = function_definer.define_function(input)
    agent.execute_function(function)

# Training Function Caller with scenario-based training
scenarios = get_scenarios()
for scenario in scenarios:
    function_sequence = determine_function_sequence(scenario, agents)
    function_caller.call_function(function_sequence)

# Human-in-the-loop for quality control and feedback
while True:
    for agent in agents:
        input = human.provide_input()
        feedback = function_definer.refine_function(input, agent)
        agent.update_function(feedback)

    # Optional: Continuous training for Function Caller
    new_scenarios = get_new_scenarios()
    for scenario in new_scenarios:
        function_sequence = determine_function_sequence(scenario, agents)
        function_caller.call_function(function_sequence)
```

Explanation:

- **Human and Function Definer Interaction**: The human provides inputs for function definition. The Function Definer processes these inputs to define functions.
- **Agents Execute Functions**: Each agent is given a function defined by the Function Definer, based on human inputs.
- **Function Caller Training**: The Function Caller is trained through various scenarios, learning the sequence of function calls.

- **Human Feedback Loop**: The human continuously provides feedback, ensuring quality and relevance of the functions and their execution.
- **Adaptation and Iterative Improvement**: The agents and the Function Caller adapt based on human feedback, refining their functions and call sequences over time.

AU

You:

Long Copy/Paste Prompt $5Million AUD Grant: AI Adopt by the Australian Government

Create some examples of how explicitly we can define the scenarios and environments within which the agents will be operating? Here are a list of Aura of Intelligence (my technology start-up that is about building XR/AI Auras for Joyful Responsible Abundance on Earth, outer space and inner space; we are applying for a $5Million AUD Grant: AI Adopt by the Australian Government;) each of the Aura of Intelligence Expansion Projects to the AI Adopt program objectives and identify potential stakeholders for engagement. 1. Aged Care and Dementia, Mental Health, Personal WellBeing, Medical Profession AI Adopt Objective: Support healthcare SMEs in AI integration. Stakeholders: Hospitals, aged care facilities, mental health organizations, healthcare technology companies, medical research institutions. 2. Education & Life-Long Learning AI Adopt Objective: Upskill workers and educators in AI. Stakeholders: Educational institutions, online learning platforms, vocational training centers, education technology companies. 3. Entrepreneurial Home-Share, Gender Balanced Venture Capital Funding AI Adopt Objective: Encourage diversity in tech entrepreneurship. Stakeholders: Venture capital firms, entrepreneurial networks, women's business associations, startup incubators. 4. Tourism, Travel & Immigration, Capsule/Pod Hotels & Smart Cities AI Adopt Objective: AI adoption in tourism and smart city management. Stakeholders: Tourism boards, hotel chains, city planning offices, travel agencies, smart technology providers. 5. Trade and Logistics, Mining, Industry, Meta-Materials AI Adopt Objective: Enhance key industries with AI. Stakeholders: Logistics companies, mining corporations, industrial automation firms, material science research labs. 6. Art and Entertainment, Fashion & Smart Clothing AI Adopt Objective: Support creative industries in AI. Stakeholders: Art galleries, fashion designers, entertainment companies, wearable technology developers. 7. Smart Homes & Offices, Healthy Human Longevity AI Adopt Objective: Improve life quality with AI. Stakeholders: Real estate developers, home automation companies, health and wellness centers, aging research organizations. 8. Climate & Environmental Responsibility, Disaster Prevention & Recovery AI Adopt Objective: AI for environmental and disaster management. Stakeholders: Environmental agencies, disaster response organizations, climate research institutes, sustainability NGOs. 9. AI In-Home Auto-Farm, Historic Simulations AI Adopt Objective: AI in agriculture and historical research. Stakeholders: Agricultural technology companies, farms, history departments at universities, cultural heritage organizations. 10. Existential Threats to Life on Earth, Outer Space Industry, Astronaut Training AI Adopt Objective: AI in space exploration and risk management. Stakeholders: Space agencies (like NASA, ESA), existential risk think tanks, astronaut training centers, research universities. 11. Inner Space Metaverse, Virtual Personalities for Pets, Language Translation Inter-Species AI Adopt Objective: Advance AI in virtual reality and communication. Stakeholders: VR technology companies, pet care organizations, linguistics researchers, animal behavior specialists. 12. Subterranean Cities, Inter-Continental Hyperloop Tunnels, Orbital Infrastructure AI Adopt Objective: AI in innovative infrastructure projects. Stakeholders: Infrastructure engineering firms, urban planners, transportation companies, aerospace industry players. 13. Solar System Scale

Development AI Adopt Objective: AI for large-scale space projects. Stakeholders: Space development companies, astronomical research institutions, government space agencies, planetary science experts. Next Steps for Stakeholder Engagement: Initial Outreach: Contact potential stakeholders to introduce the projects and explore collaboration opportunities. Partnership Development: Develop formal partnerships with interested stakeholders, focusing on shared goals and resource allocation. Community Engagement: Engage with local communities, especially for projects with direct social impact. Regular Updates and Collaboration Meetings: Maintain ongoing communication with stakeholders to ensure alignment and adapt to evolving needs.

2 / 2

Mixture of Experts, Function Callers & AI Agents

For each Aura of Intelligence Expansion Project, we can define specific scenarios and environments in which the agents will operate. These scenarios are tailored to align with the AI Adopt program objectives and involve relevant stakeholders:

Aged Care and Dementia, Mental Health, Personal WellBeing, Medical Profession

- **Scenario**: AI agents assist in diagnosing and managing dementia in aged care facilities.
- **Environment**: A virtual healthcare center where agents interact with patients, healthcare providers, and family members.
- **Stakeholders**: Hospitals, aged care facilities, mental health organizations.

Education & Life-Long Learning

- **Scenario**: AI agents create personalized learning plans and upskill educators in AI.
- **Environment**: Online education platforms and virtual classrooms.
- **Stakeholders**: Educational institutions, online learning platforms.

Entrepreneurial Home-Share, Gender Balanced Venture Capital Funding

- **Scenario**: AI agents analyze market trends to identify promising female-led startups for funding.
- **Environment**: Virtual venture capital firms and entrepreneurial networks.
- **Stakeholders**: Venture capital firms, entrepreneurial networks.

Tourism, Travel & Immigration, Capsule/Pod Hotels & Smart Cities

- **Scenario**: AI agents optimize travel itineraries and enhance guest experiences in smart hotels.
- **Environment**: Smart city simulations and virtual travel agencies.
- **Stakeholders**: Tourism boards, hotel chains.

Trade and Logistics, Mining, Industry, Meta-Materials

- **Scenario**: AI agents streamline logistics and innovate in material development.
- **Environment**: Virtual logistic centers and material science labs.
- **Stakeholders**: Logistics companies, mining corporations.

Art and Entertainment, Fashion & Smart Clothing

- **Scenario**: AI agents assist in designing fashion collections and curating art exhibitions.
- **Environment**: Virtual design studios and galleries.
- **Stakeholders**: Fashion designers, art galleries.

Smart Homes & Offices, Healthy Human Longevity

- **Scenario**: AI agents optimize smart home environments for health and wellness.
- **Environment**: Virtual smart homes and wellness centers.
- **Stakeholders**: Real estate developers, home automation companies.

Climate & Environmental Responsibility, Disaster Prevention & Recovery

- **Scenario**: AI agents model climate impact and plan disaster response strategies.
- **Environment**: Virtual environmental management centers.
- **Stakeholders**: Environmental agencies, disaster response organizations.

AI In-Home Auto-Farm, Historic Simulations

- **Scenario**: AI agents manage automated farming systems and create historic simulations.
- **Environment**: Virtual farms and historic simulation labs.
- **Stakeholders**: Agricultural technology companies, history departments.

Existential Threats to Life on Earth, Outer Space Industry, Astronaut Training

- **Scenario**: AI agents assess existential risks and train astronauts.
- **Environment**: Virtual space agencies and training facilities.
- **Stakeholders**: Space agencies, existential risk think tanks.

Inner Space Metaverse, Virtual Personalities for Pets, Language Translation Inter-Species

- **Scenario**: AI agents develop virtual pet personalities and facilitate interspecies communication.
- **Environment**: Virtual reality platforms and animal behavior research labs.
- **Stakeholders**: VR technology companies, pet care organizations.

Subterranean Cities, Inter-Continental Hyperloop Tunnels, Orbital Infrastructure

- **Scenario**: AI agents plan and manage subterranean city development and hyperloop networks.
- **Environment**: Virtual urban planning and transportation models.
- **Stakeholders**: Infrastructure engineering firms, urban planners.

Solar System Scale Development

- **Scenario**: AI agents assist in planning large-scale space projects.
- **Environment**: Virtual space development simulations.
- **Stakeholders**: Space development companies, astronomical research institutions.

For each scenario, the agents are tasked with specific roles that align with the project objectives, interacting within virtual environments that simulate real-world settings. Stakeholder engagement is crucial in these scenarios to ensure the solutions are practical and meet the needs of the intended users.

AU

You

Please create a much more comprehensive elaboration on defining specific scenarios and environments in which the agents will operate for 1. Aged Care and Dementia, Mental Health, Personal Well-being, Medical Profession

Mixture of Experts, Function Callers & AI Agents

For the project focusing on Aged Care and Dementia, Mental Health, Personal WellBeing, and the Medical Profession, a comprehensive scenario and environment can be defined as follows:

Scenario: Virtual Integrated Care System for Aged Care and Dementia

Environment:

Virtual Care Home:

- A simulated environment replicating an aged care facility.
- AI agents interact with virtual patients exhibiting various stages of dementia.
- Features include virtual common areas, patient rooms, and therapy spaces.

Mental Health Monitoring Center:

- A platform where AI agents analyze patient data for signs of mental health issues.
- Includes virtual consultation rooms for patient-agent interactions.

Personal WellBeing Hub:

- An interactive space focusing on overall wellbeing of elderly patients.
- Offers activities like memory games, virtual social interactions, and health education.

Medical Professional Interface:

- A portal for healthcare professionals to monitor, guide, and intervene in patient care.
- Provides access to patient data, AI-generated reports, and communication with AI agents.

AI Agent Roles:

Caregiver Agent:

- Provides daily care assistance, monitors health, and engages in social interactions.
- Uses AI to tailor activities to individual patient needs and preferences.

Therapist Agent:

- Specializes in mental health and dementia therapy.
- Conducts cognitive exercises and provides emotional support.

Health Analyst Agent:

- Analyzes patient data for health trends and potential issues.
- Alerts medical professionals to changes in patient conditions.

Coordinator Agent:

- Facilitates communication between different agents and medical professionals.
- Ensures seamless integration of care and therapy.

Interaction Flow:

- **Daily Routine**: Caregiver Agents assist patients with daily tasks, while continuously monitoring their health and well-being.
- **Therapy Sessions**: Therapist Agents conduct scheduled therapy sessions, adapting strategies based on patient responsiveness.
- **Data Analysis**: Health Analyst Agents review patient data, identifying any concerning trends and notifying medical staff as needed.
- **Professional Oversight**: Medical professionals use the Medical Professional Interface to oversee patient care, provide guidance, and make adjustments to treatment plans.

Stakeholder Engagement:

- **Healthcare Providers**: Provide expertise and feedback to refine agent functionalities and care strategies.
- **Family Members**: Offer insights into patient histories and preferences, enhancing personalized care.

- **Technology Partners**: Collaborate on developing and refining AI algorithms and virtual environments.

By creating this detailed scenario, Aura of Intelligence can develop a robust, empathetic, and effective AI-driven system for aged care and dementia, mental health, and personal wellbeing, aligning with the objectives of the AI Adopt program.