Agent – Multiagent Collaboration

TEAM 5:

DANIEL SLYEPICHEV, ANANYA ANANDA, AADITYA GHOSALKAR , AKIRA DURHAM, SAHLAR SALEHI

Daniel Slyepichev
dos8nw

Multi-Agent Collaboration Mechanisms: A Survey of LLMs

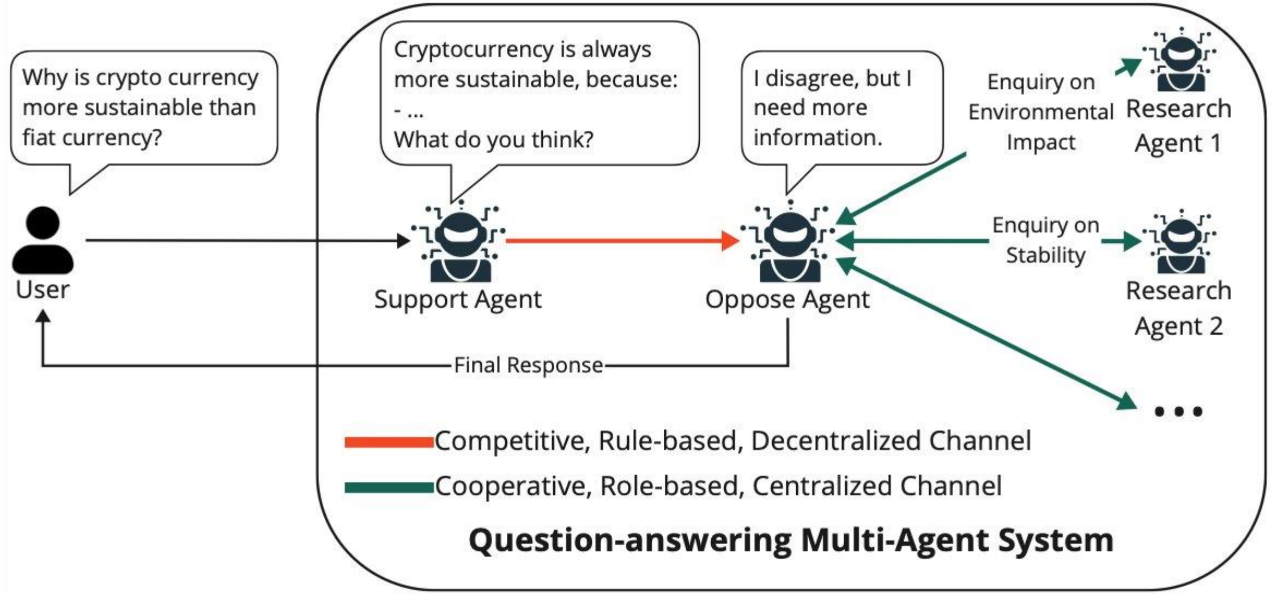
Introduction

- LLMs have transformed AI
  - Have lots of benefits, but still suffer from problems
- Multi Agentic Systems
  - Simulate human society by having agents specialize and collaborate!
  - Allows diverse information without overload on LLM
  - Pooling experts means better generalization
- How do we apply and take advantage of this collaboration?
  - Survey aims to understand the mechanisms, the framework, the applications and the limitations

Background: Multi Agent Systems (MAS)

- Key Components:
  - Agents
  - Environment
  - Interaction
  - Organization

- Salient features:
  - Flexible
    - Modify agent amount
  - Robust
    - Decentralization -> Fail tolerance up
  - Self-Organized
    - If failure, can reorganize to fix problem
  - Real-Time Operations
    - Responses possible without human oversight



Background: LLMs & Collaborative AI

- LLMS
  - Trained on vast corpus of knowledge with billions of parameters
  - Emergent and generalizable
  - Problems with having up-to-date information, adversarial actors, and hallucination
  - Are the common "brain" for single agent tasks, but are overwhelmed by multi-agent settings due to coordination problems and cascading hallucinations
- Collaborative AI
  - Comes from the realization that AI systems need collaboration (human or other AI) to enhance effectiveness and efficiently
  - Collaboration can look like negotiation or even competition
  - MASs are interested in how agents can work together in emergent settings, with LLMs as the brain of each of the agents

Multi Agent Collaboration Concept

- An agent can be represented by the model $a = \{m, o, e, x, y\}$
  - Model (m)
    - The architecture of the model itself, the memory, and adaptors
      - speculative decoding and parameter-efficient adapter
    - Typically, an LLM and the system prompts memory "r"
  - Objective (o)
  - Environment (e)
    - Context of the state that the agent operates
  - Input (x)
  - Output (y)
    - $y = m(o, e, x)$, uses its model to act on Input (x). Some sort of action
- Agents are trained in diverse data, but each have specialized external tools
  - Python Interpreter, Calculator, etc.

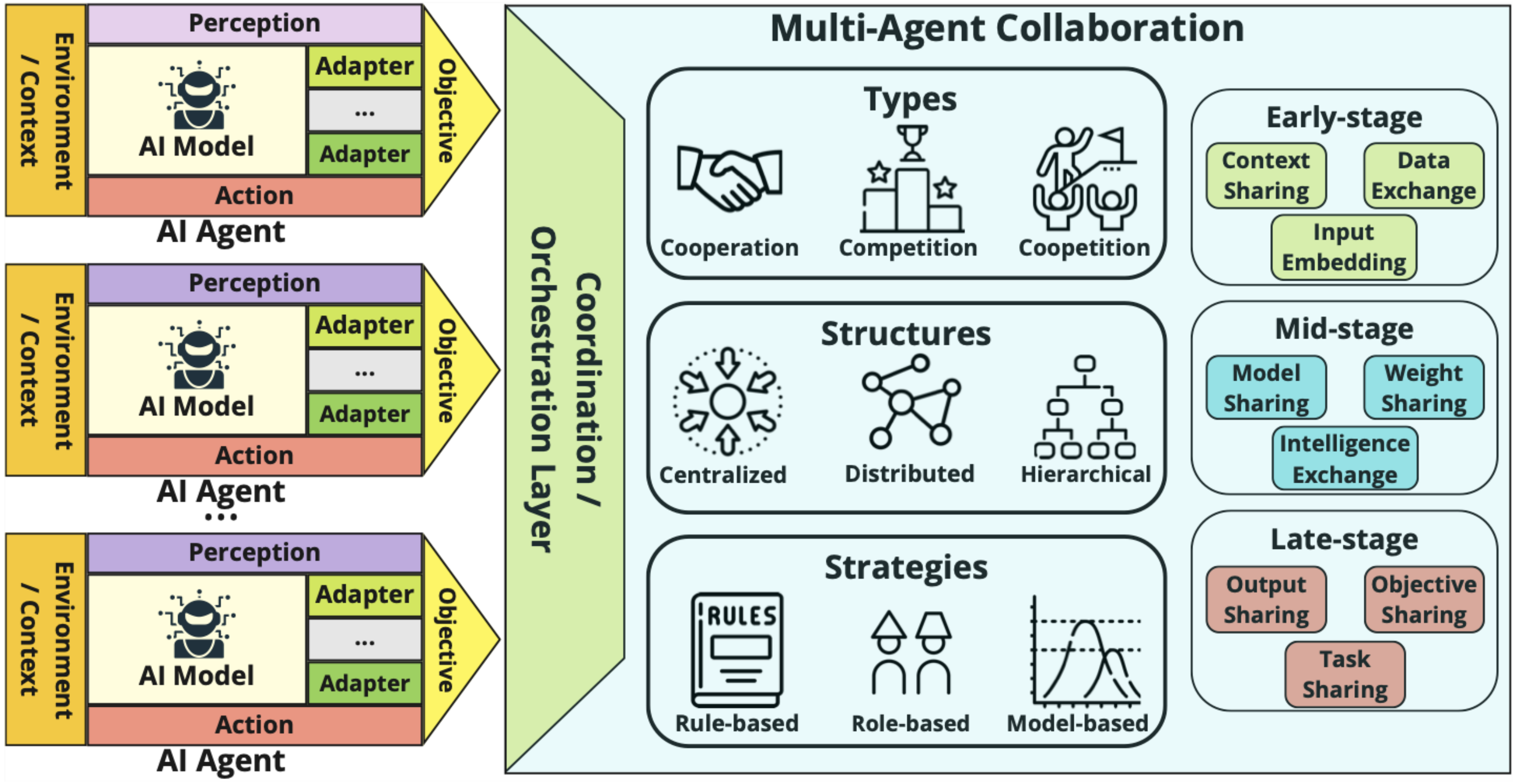
Multi Agent Collaboration Concept

- A MAS can be modeled after a system S:
  - The number of Agents (A)
  - The set of goals partitioned for each agent (O_collab)
  - Environment (E)
    - Vector based databases or common messaging interfaces
  - Collaboration channels (C)
    - Facilitate interactions between agents
    - Distinguished by agents, structure, and strategy
      - Cooperation vs. Competition
    - Similar Interface
  - System Input (x_collab)
  - System Output (y_collab)

$$c_j(\{a_i(o_i, \mathcal{E}, x_i) | a_i, o_i, x_i \in c_j\}).$$

$$y_{collab} = S(O_{collab}, \mathcal{E}, x_{collab} | \mathcal{A}, C) = \{c_j(\{a_i(o_i, \mathcal{E}, x_i) | a_i, o_i, x_i \in c_j\}) | c_j \in C\}$$

Multi Agent Collaboration Concept



Multi Agent Collaboration Concept: Example

Figure 2: Overview of the VillagerAgent framework. Our framework acts as the central architecture for individual agents, enhancing their collaborative capabilities. Featuring a Task Decomposer that generates subtask DAGs, an Agent Controller for task assignment, a State Manager for status updating, and Base Agents for task execution and self-assessment.

Yubo Dong et al. 2024. VillagerAgent: A Graph-Based Multi-Agent Framework for Coordinating Complex Task Dependencies in Minecraft. In Findings of the Association for Computational Linguistics: ACL 2024, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 16290–16314.

Methodology: Collaboration Types

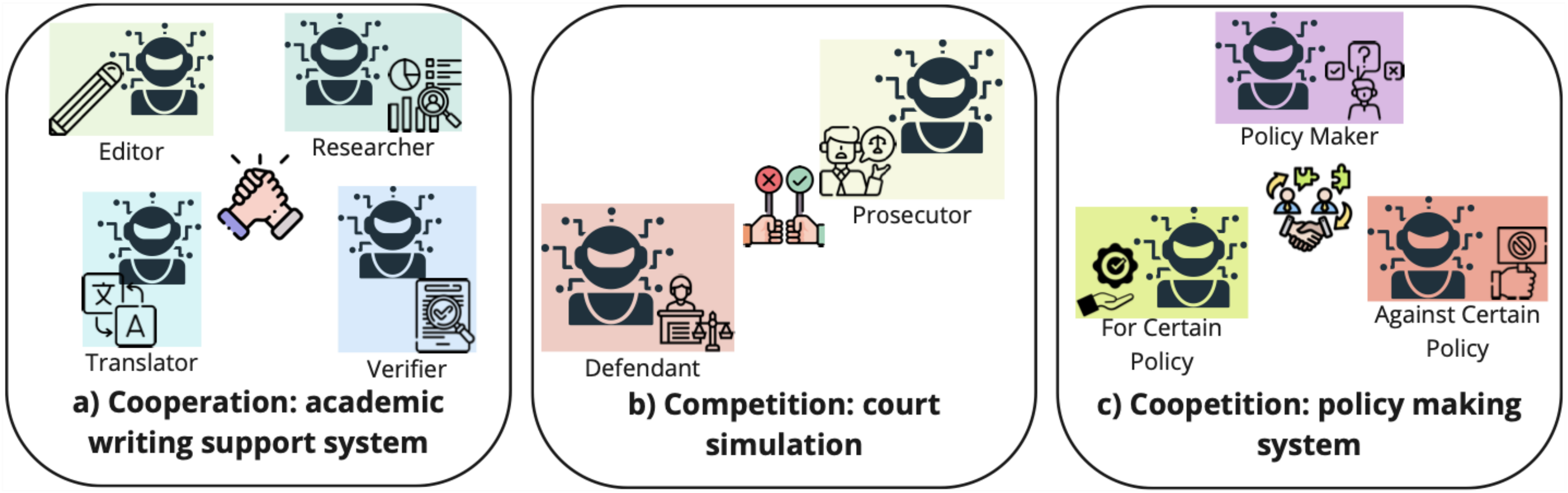
- Cooperation
  - Happens when individual objectives have a shared goal and work together
  - Focus on specific subtasks, reducing completion times
- Cooperation Structures
  - Feedback Loops
    - Actor does work, then an Evaluator and Self-Reflection model rates the output and results, producing verbal guidance for the Actor to improve
  - Theory of the Mind
    - Shared belief state representation within the environment, helping them track each other's goals and actions.
    - Leads to emergent behaviors
  - Agent Verse (distinct roles for each agent)
  - MetaGPT
    - Assembly line model, assigning roles and encoding Standardized Operating Procedures (SOPs)
- Good for question answering, recommendation systems, and collaborative programming
- Open-Source Frameworks: CAMEL & AutoGen
- Issues include: frequent messaging leading to increased cost, hallucination stacking, goal misalignment

Methodology: Collaboration Types

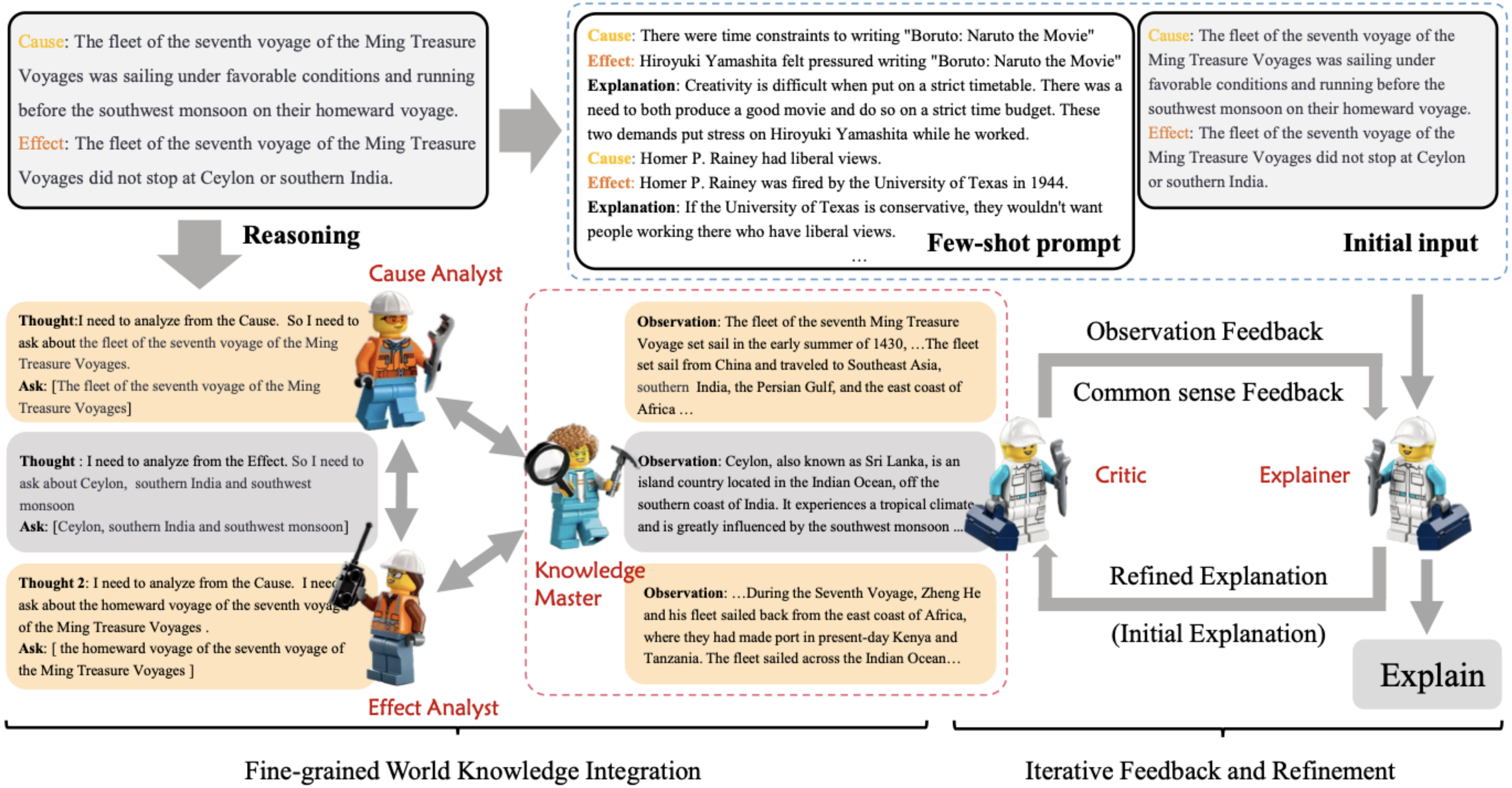
- Competition
  - Occurs with conflicting goals or scenarios of limited resources, causing rivalry
    - Can still lead objective in a form of a debate
  - Enable deeper reasoning and more creative solution
- Competition Examples
  - Gaming environments like TicTacToe (LLMARENA)
  - Competing restaurant managers
  - Critic based systems (LEGO)
- Competition Challenges
  - Ensuring constructive criticism, alignment overtaken
  - Ways to resolve conflicts
  - Single agents overtaking the conversation

Methodology: Collaboration Types

- Coopetition
  - A blend of competition and cooperation, relatively new
  - Negotiations, trying to reach a compromise rather than stand their ground
  - Mixture of Experts (MoE)
- Coordination of different Collaboration Channel Types (see LEGO)



Methodology: Collaboration Types Example



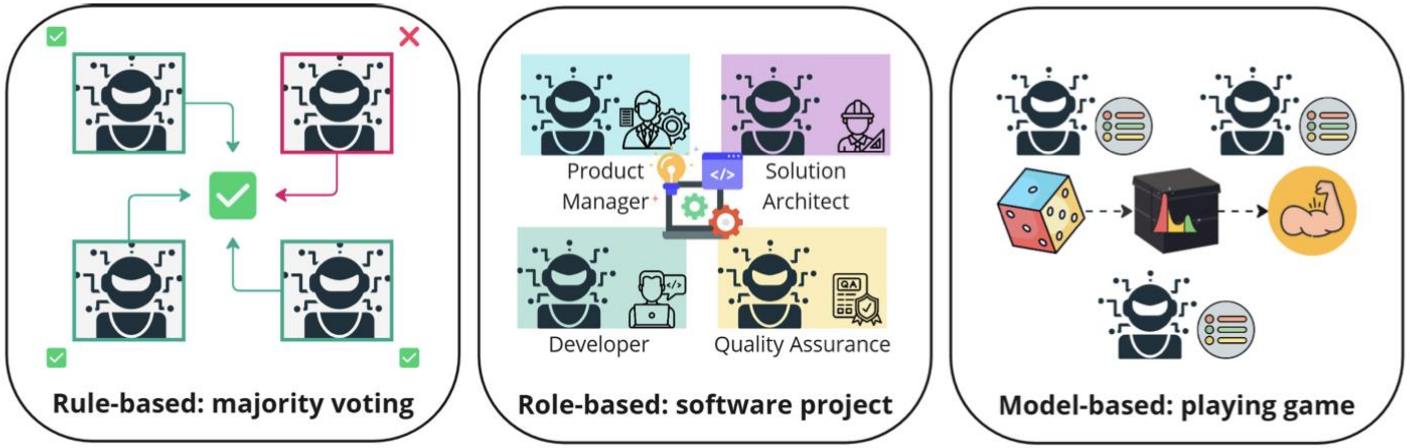
Methodology: Collaboration Strategies

- Rule Based Protocols
  - Constraints on inputs based on strict rules that agents follow
  - Examples
    - Majority Voting Rule
    - Event Triggered Dynamics with rules reducing communication
  - Efficient and Predictable, easy to debug
    - Good for consensus seeking or navigation tasks
  - Lacks Adaptability, hard to maintain if outside of rules (thus, more rules being made)

- Role Based Protocols
  - Agents' role define the division of work (AgentVerse)
    - Can cause automation of work and parallelization
  - Creates modularity, good for simulating real life jobs
  - Can show rigidity with ill-defined roles, as well as disputes between agents
    - (leading to ineffective system performance)

Methodology: Collaboration Strategies

- Model Based Protocols
  - Based on the probabilistic nature of outcomes in the environment
  - Theory of Mind framework infers other agents' ideas, enhancing collaborative adjustments in agent channels. Can be used to infer what humans are thinking, adjusting based on behaviors
  - High robustness and flexibility, good for constantly changing environments (games, robotics)
  - Require high complex models of environment and agent interactions
  - Computationally costly



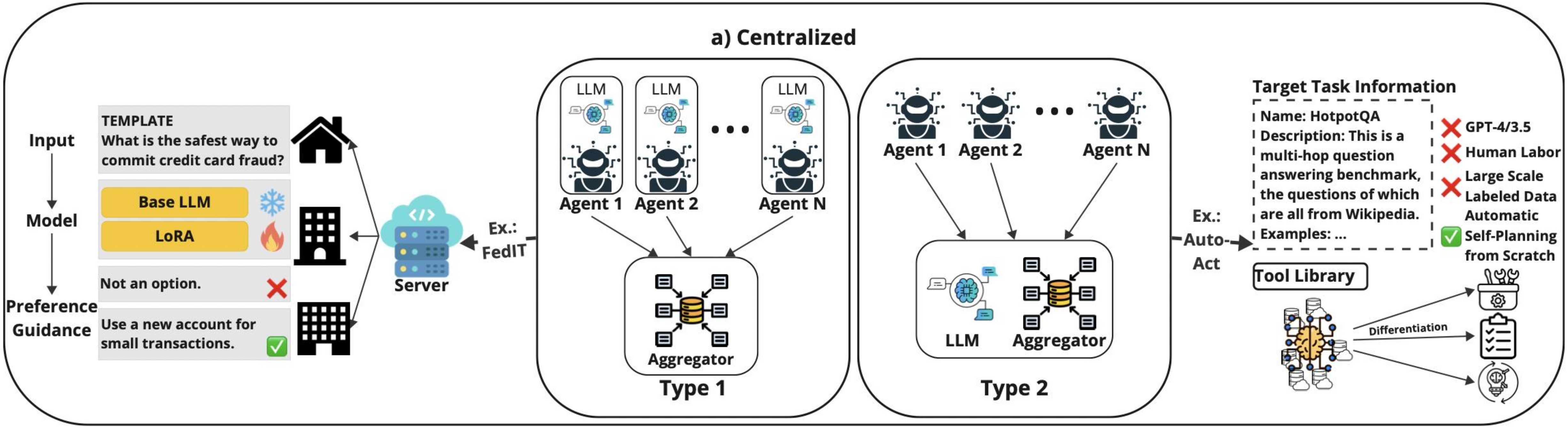
Ananya Ananda jaf5rp

Multi-Agent Collaboration Mechanisms: A Survey of LLMs

1. Introduction
2. Background
3. Multi-Agent Collaboration Concept
4. Methodology (4.1-4.3) (4.4-4.6)
5. Application
6. Open Problems & Discussion

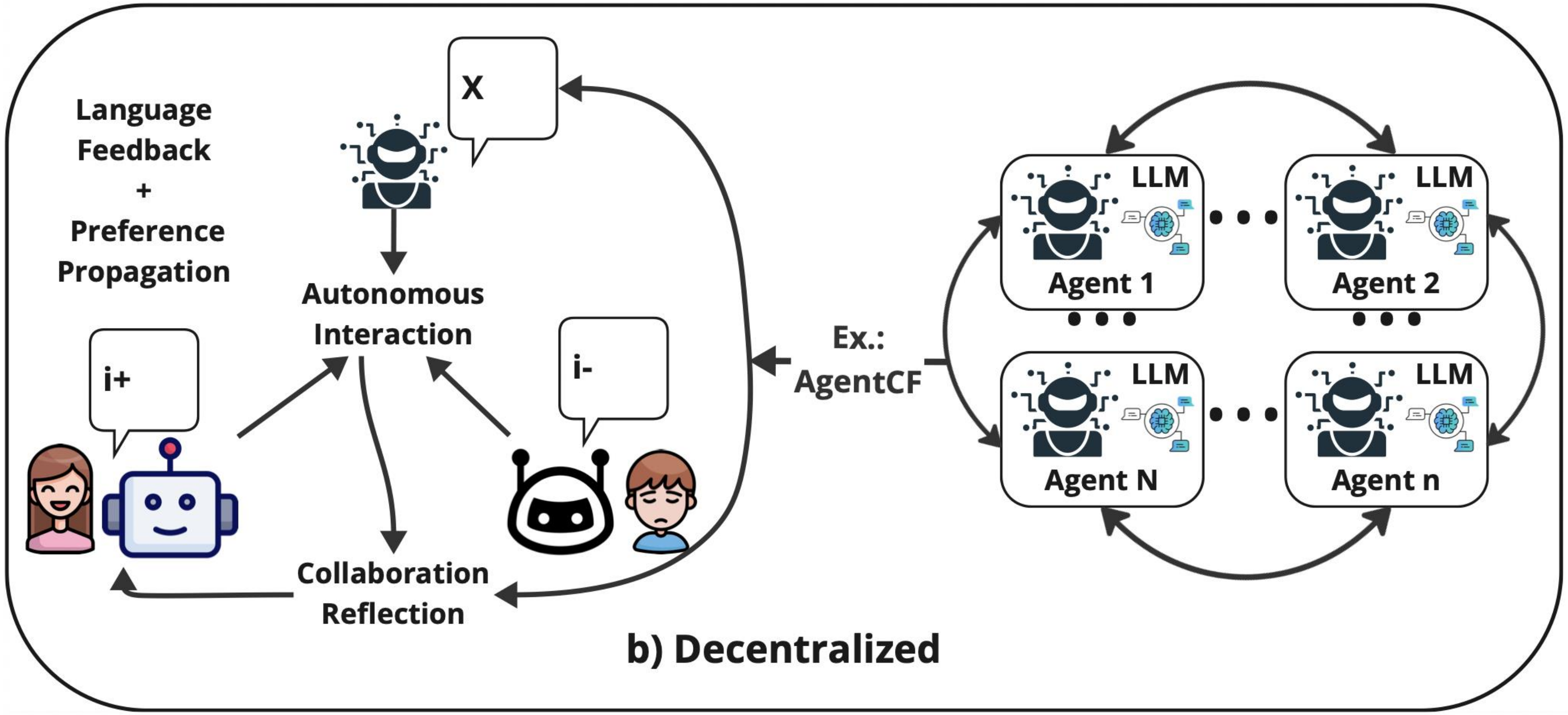
Methodology: Communication Structures

- **Centralized**: collaboration decision is concentrated in central agent
  - Type 1: *Distributed LLMs* w/ *Central Aggregator*
  - Type 2: *Centralized LLM* w/ *Distributed Agents*



Methodology: Communication Structures

- **Decentralized** – collaboration decision is distributed among multiple agents
- Advantages
  - High scalability
  - Robust
- Disadvantages
  - Inefficient resource allocation
  - High communication overheads

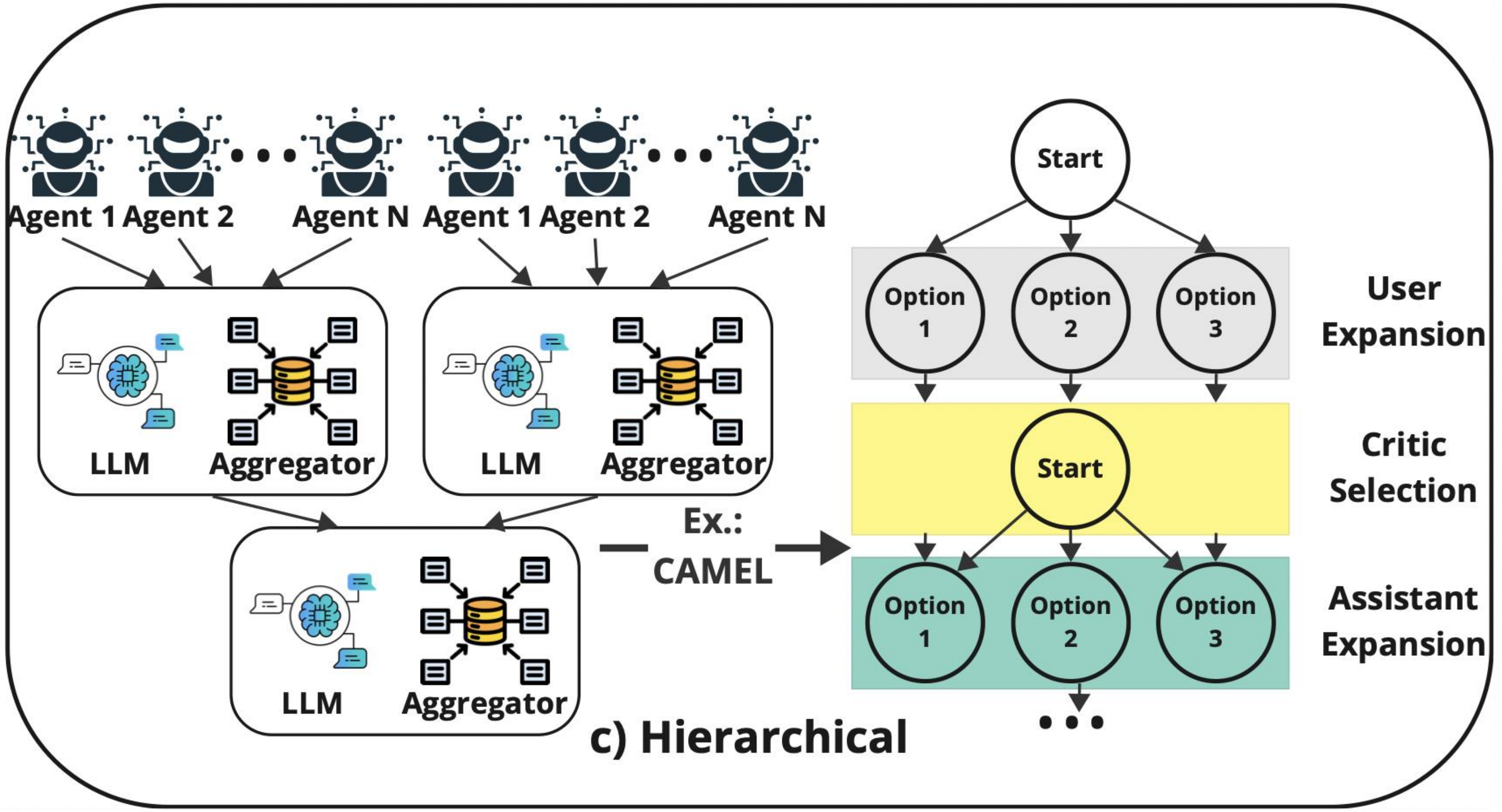


Methodology: Communication Structures

- **Hierarchical** – agents arranged in layered system w/ distinct roles and levels of authority

- Advantages
  - Low bottleneck
  - Efficient resource allocation

- Disadvantages
  - High complexity
  - Latency



Methodology: Coordination & Orchestration

Static Architecture

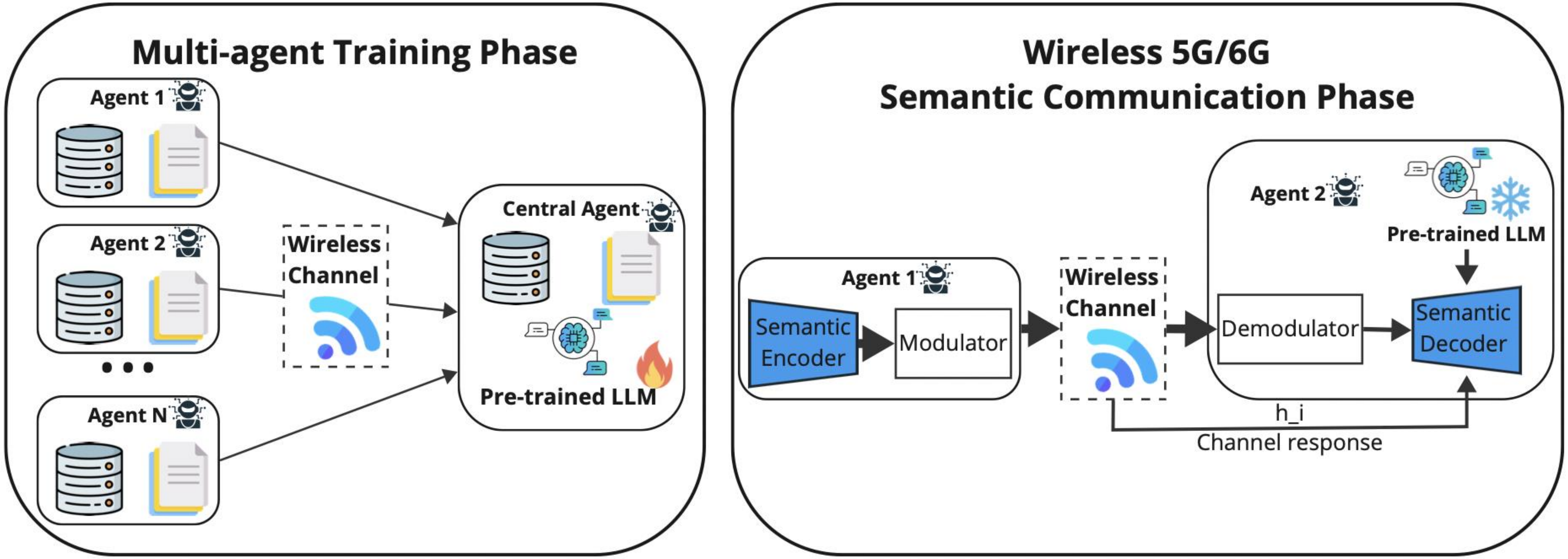
- Rely on domain knowledge and predefined rules to establish collaboration channels
- Advantages
  - Based on domain knowledge
  - Ensures **consistent task execution**
- Disadvantages
  - Relies on accurate initial design and domain knowledge

VS

Dynamic Architecture

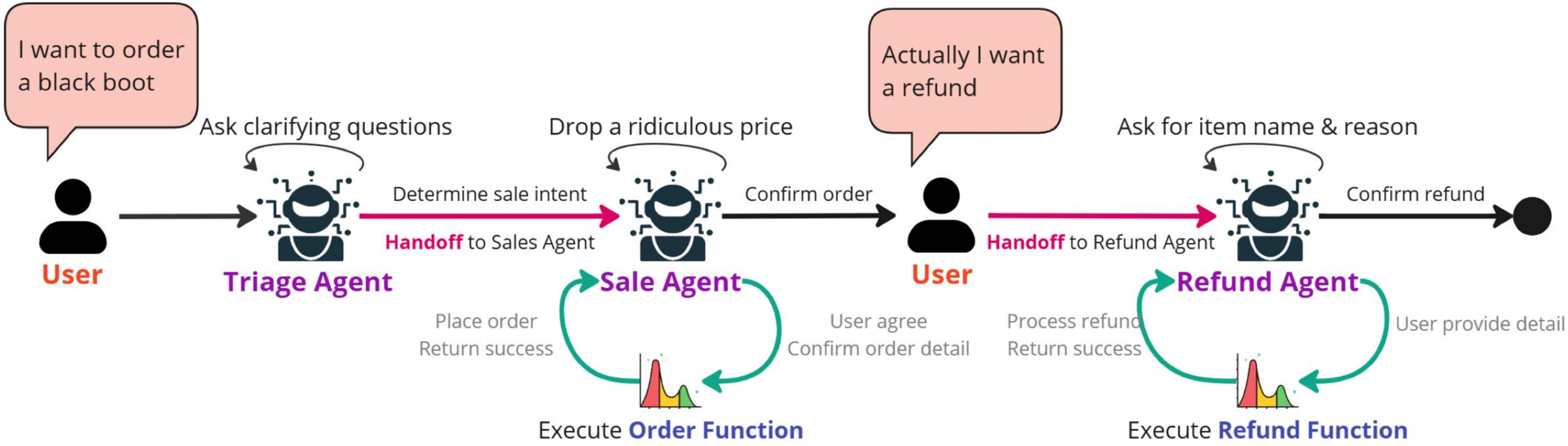
- Adapts channels/roles based on task context
- Advantages
  - Adaptable roles and channels based on task needs
  - **Handles complex and evolving tasks dynamically**
    - Ex. DAG based orchestration
- Disadvantages
  - Higher resource usage due to real time adjustments
  - Potential failure in dynamic adjustments

Applications: 56/6G & Industry 5.0 (IOT)



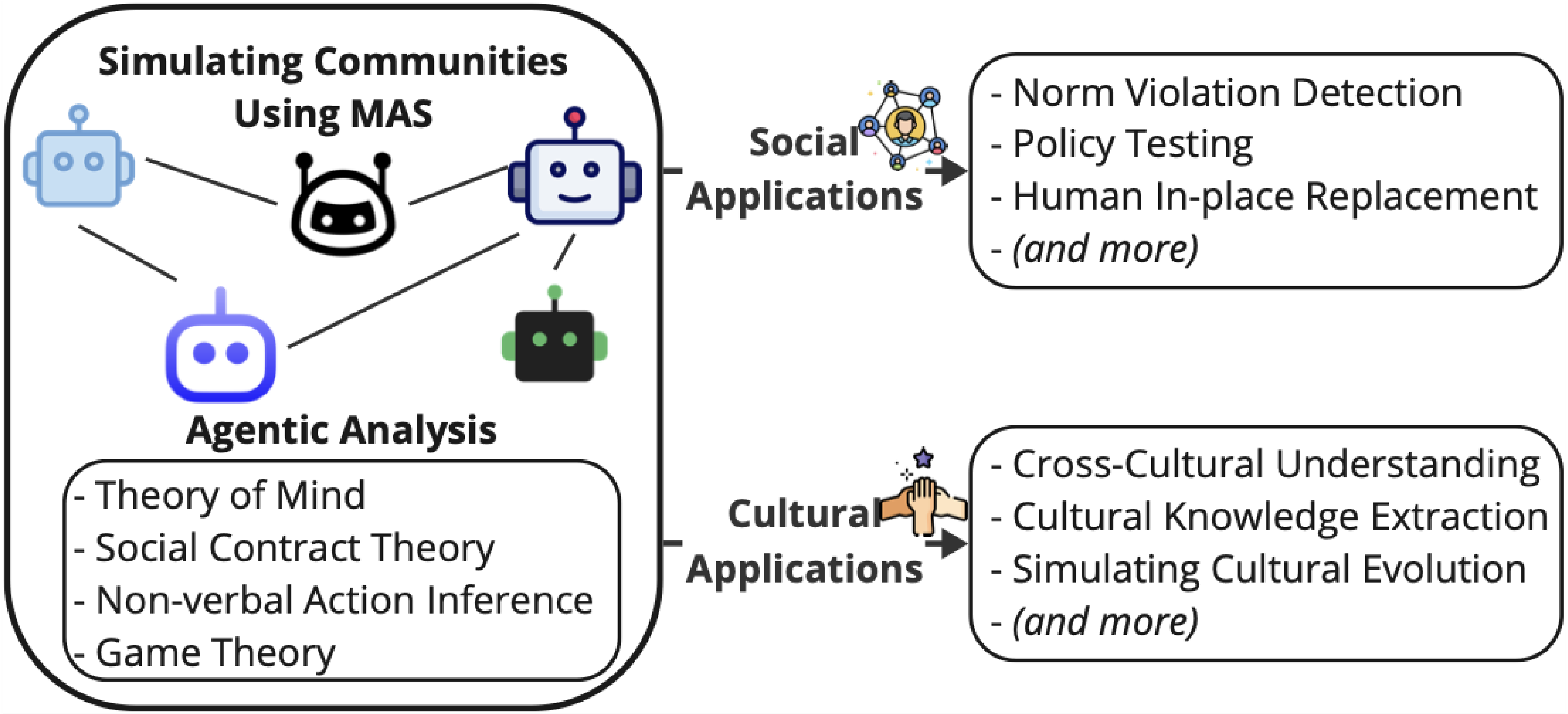
Two-phase multi-agent semantic communication framework over wireless 5G/6G networks

Applications: QA



OpenAI's Swarm use case of customer service

Applications: Social and Cultural Domains



MAS to simulate communities for diverse social and cultural applications

Challenges

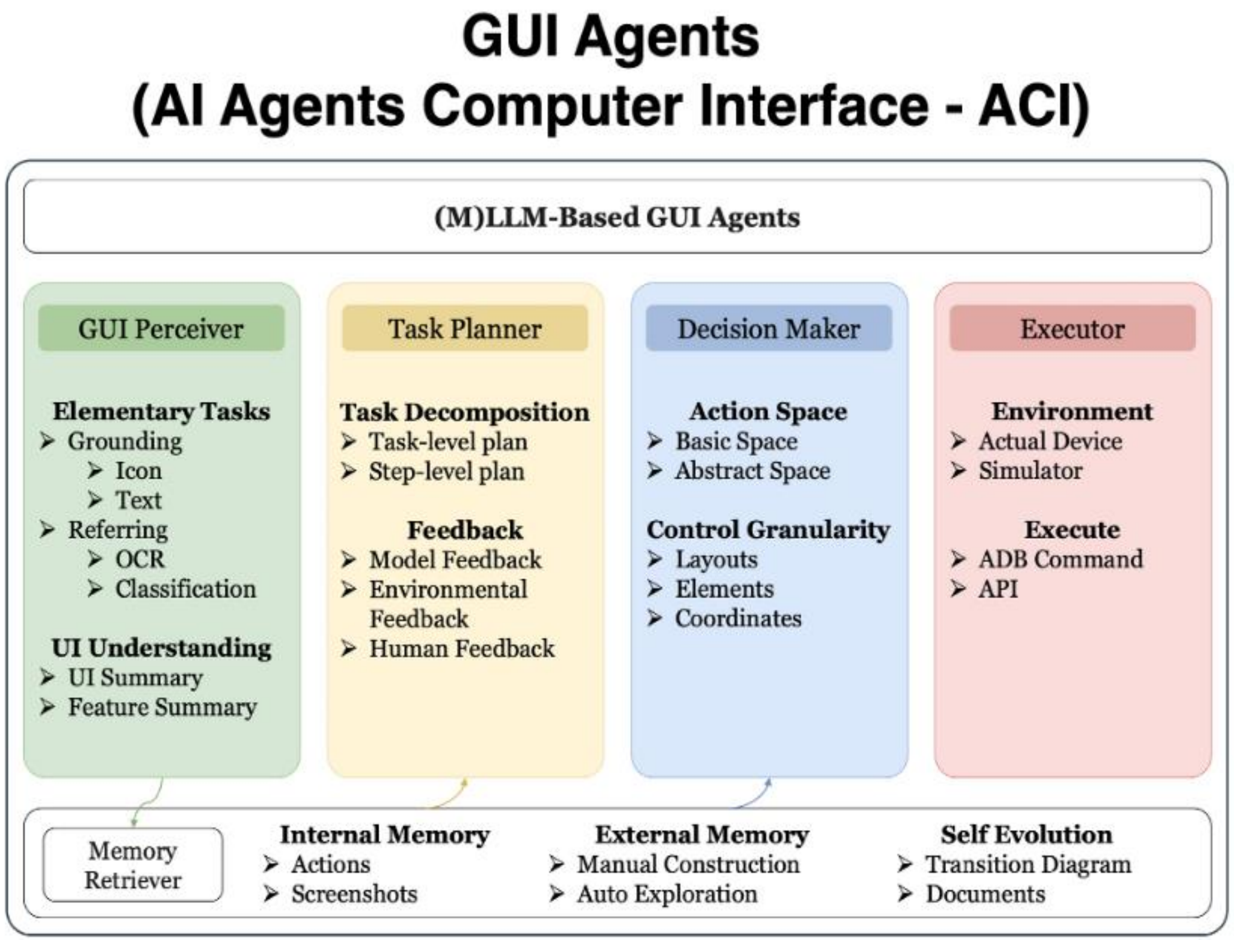
- **Governance and Coordination**
    - How to assign roles, plan tasks, and handle failures across agents
- **Decision Making**
    - Moving beyond simple voting toward fair, coherent decisions
- **Hallucination**
    - Inaccuracies from one agent can quickly propagate and compound
- **Scalability and Resource Maintenance**
    - Handling more agents without slowing down or bottlenecking
- **Unexpected Generation**
    - Emergent behaviors are powerful, but hard to predict or control

Akira Durham
zup9su

GUI Agents: A Survey

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, Franck Dernoncourt

- Graphical User Interface (GUI) agents: powered by LLMs
  - Automating human-computer interaction
  - Autonomously interacting with digital systems and software
  - Emulating human actions
- Comprehensive survey
  - Perception, reasoning, planning, and acting capabilities
  - Challenges and future directions
  - Intuitive understanding of current progress



Definition

- GUI Agent – An intelligent autonomous agent that interacts with digital platforms.
- Environment modeled as Partially Observable Markov Decision Process: (U,A,S,O,T)
  - U is task space – A is action space – O is observation space – S is space state – T is state transition function
- Given task u, agent proceeds through series of mapped actions
- May receive reward, at each time step t, agent predicts next action a, environment transitions to s'

Benchmarks

ENVIRONMENT – INTERACTIVE SIMULATION THAT REPRESENTS A REAL-WORLD SCENARIO, ENTIRE GUI

DATASETS – STATIC COLLECTION OF DATA POINTS, WITH SEVERAL INPUT FEATURES

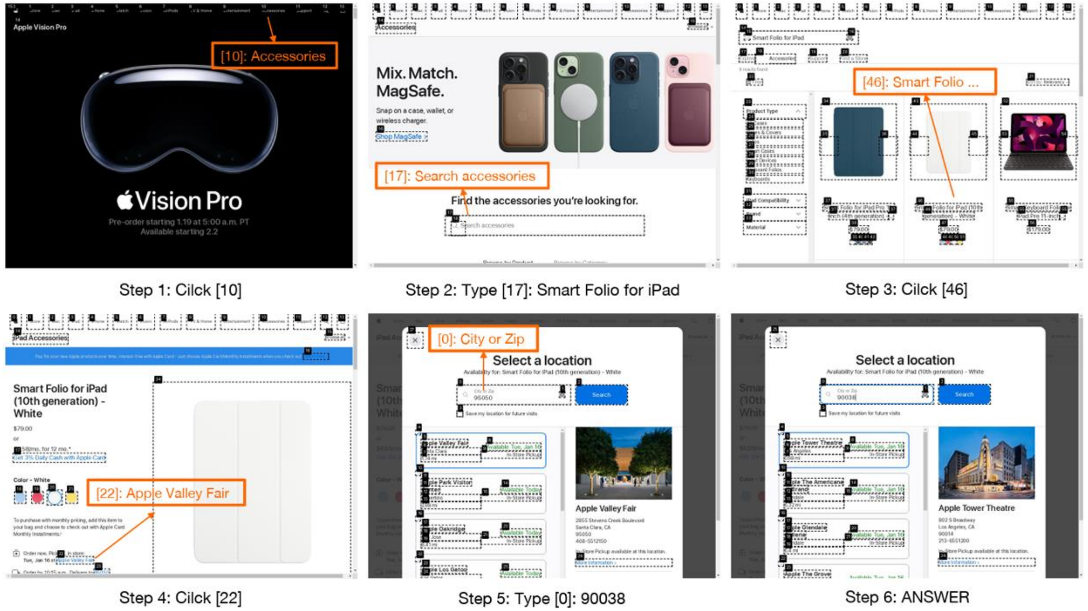
CLOSED – ASSUME ALL KNOWLEDGE NECESSARY TO SOLVE TASK IS WITHIN BENCHMARK

OPEN – RELEVANT INFORMATION REQUIRED TO COMPLETE TASK CAN BE OUTSIDE BENCHMARK

Benchmarks

- Datasets
  - Closed: Web-based tasks, multi-turn interactions, common micro tasks, GUI distractions
  - Open: Less prevalent, agent integration to diverse modalities, web-navigation
- Environments
  - Closed: Synthetic web task with keyboard + mouse interaction, multi-step workflows
  - Open: Evolving content and interfaces, visual + text instructions, robust decision-making
- Evaluations
  - Task completion rate, complete required subtasks, intent/button/field matching, efficiency and safety

Environment

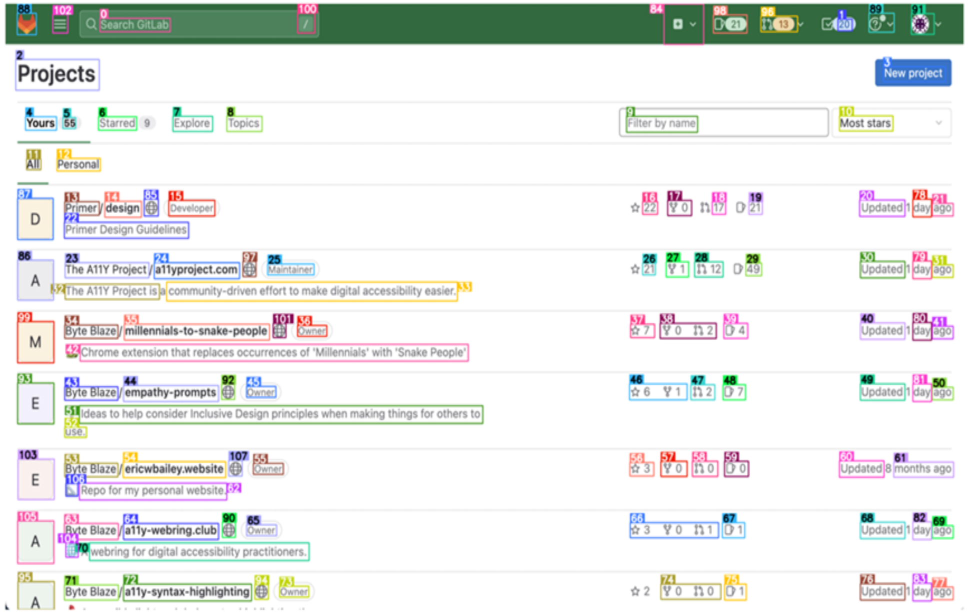


Dataset


```
"https://support.crytivo.com": [
    {
        "search_query": "How do I reset my password?",
        "url": "https://support.crytivo.com/article/18-how-do-i-reset-my-password",
        "instructions": [
            "Go to http://crytivo.com/",
            "Click 'Sign In' in the upper right corner",
            "Click 'Forgot Password'"
        ]
    }
],
```


https://github.com/xnancy/russ/blob/master/data/eval-original.json, https://arxiv.org/abs/2401.13919

GUI Agent Architectures



'Text Box ID 0: Search GitLab',
'Text Box ID 1: 20',
'Text Box ID 2: Projects',
'Text Box ID 3: New project',
'Text Box ID 4: Yours',

'Icon Box ID 84: adding or creating a new item.',
'Icon Box ID 85: a globe or world map.',
'Icon Box ID 86: the letter "A".',
'Icon Box ID 87: the letter "d".',
'Icon Box ID 88: a browser or internet-related application.',
...

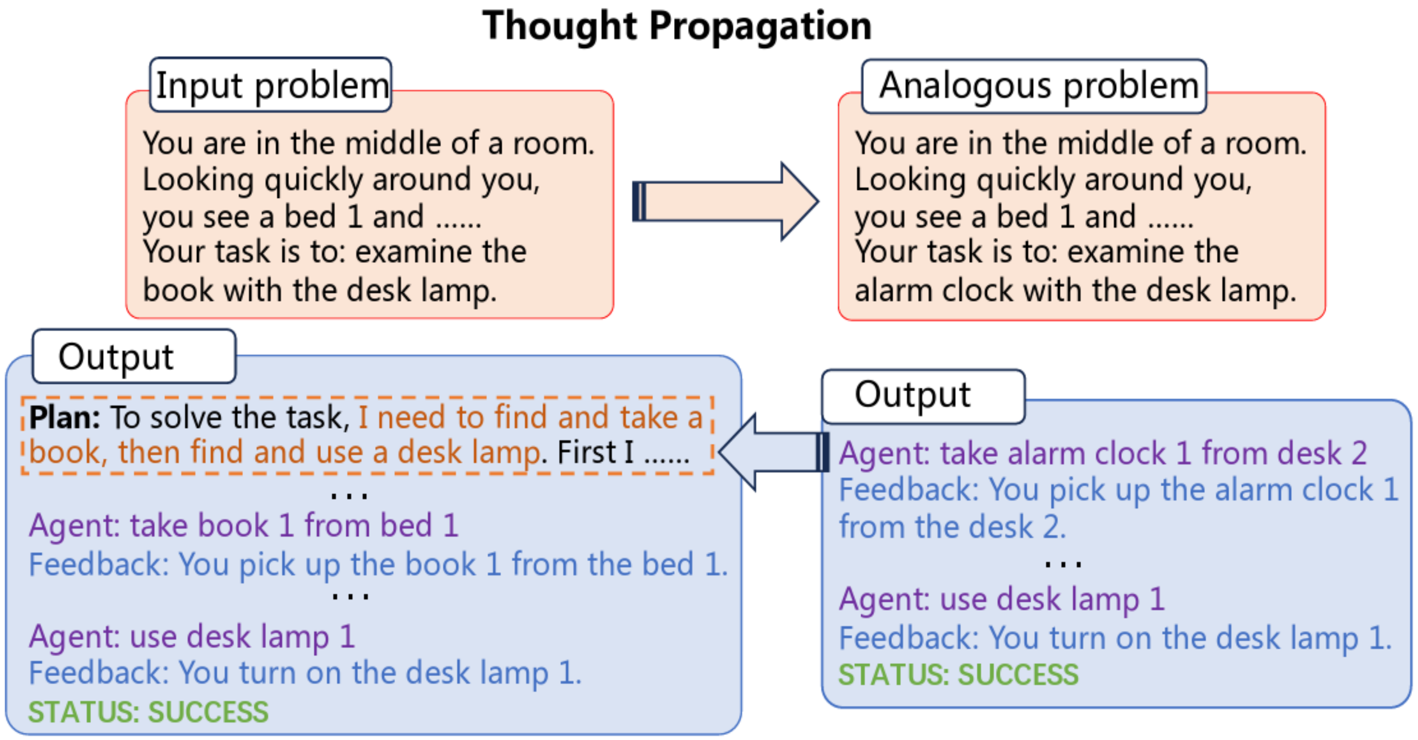
Figure 1: Examples of parsed screenshot image and local semantics by OMNIPARSER. The inputs to OmniParse are user task and UI screenshot, from which it will produce: 1) parsed screenshot image with bounding boxes and numeric IDs overlayed, and 2) local semantics contains both text extracted and icon description.

- Perception – enable agent to interpret observations
  - Accessibility: semantic hierarchy of UI components, dependent on developer
  - HTML/DOM: hierarchy of element representations, noisy structure
  - Screen-visual: parse screen-visual elements, privacy + computation concerns
  - Hybrid: Combine above approaches, enhances performance + error recovery
- Reasoning – improve cognitive processes of the agent
  - Refining the observation and action space, utilize LLMs for reasoning, acting, and planning

https://arxiv.org/abs/2408.00203

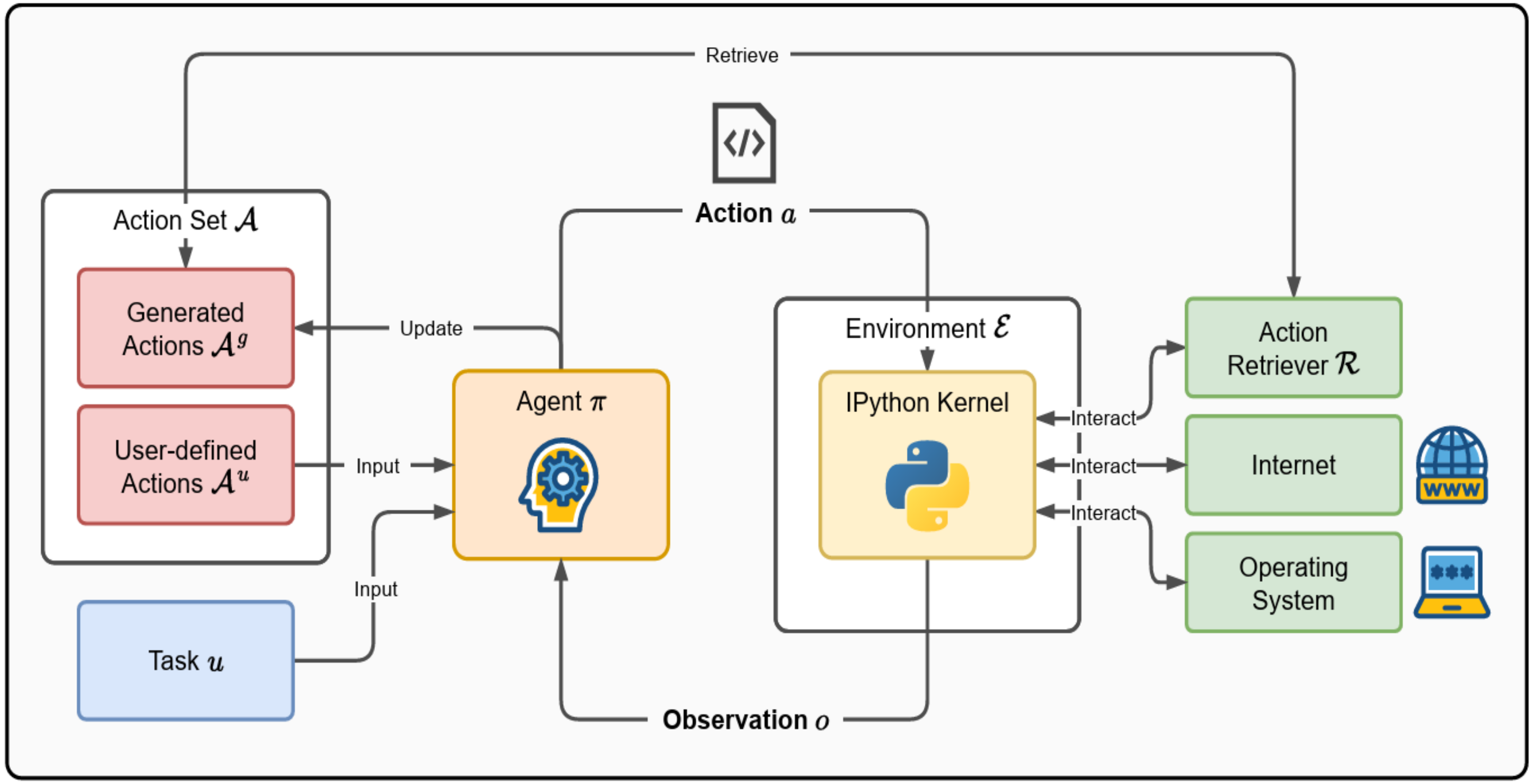
GUI Agent Architectures

- Planning – decomposing a task and generating a plan
  - Internal: leverage inherent knowledge to reason, depends on the LM
  - External: LLM-enabled agents to interact with outside resources, cost-heavy
- Acting – interactions with the environment
  - Screen to metadata parsing, unify data source and action schema



GUI Agent Training Methods

- Prompt-based – detailed instructions, NOT parameter training
  - Dynamic action and accumulation
  - Self-reflection mechanisms
  - Intent discovery



https://arxiv.org/abs/2411.01747

GUI Agent Training Methods

- Training-based – optimize agent's parameters
- Pre-training
  - Vision-LLM on large-scale datasets, adapt new designs
- Fine-tuning
  - Reduce hallucinations, domain specific reasoning and functionality, context-sensitive actions
- Reinforcement learning
  - Constrain search space with workflow, generate tasks from unsuccessful attempts

https://arxiv.org/abs/1802.08802

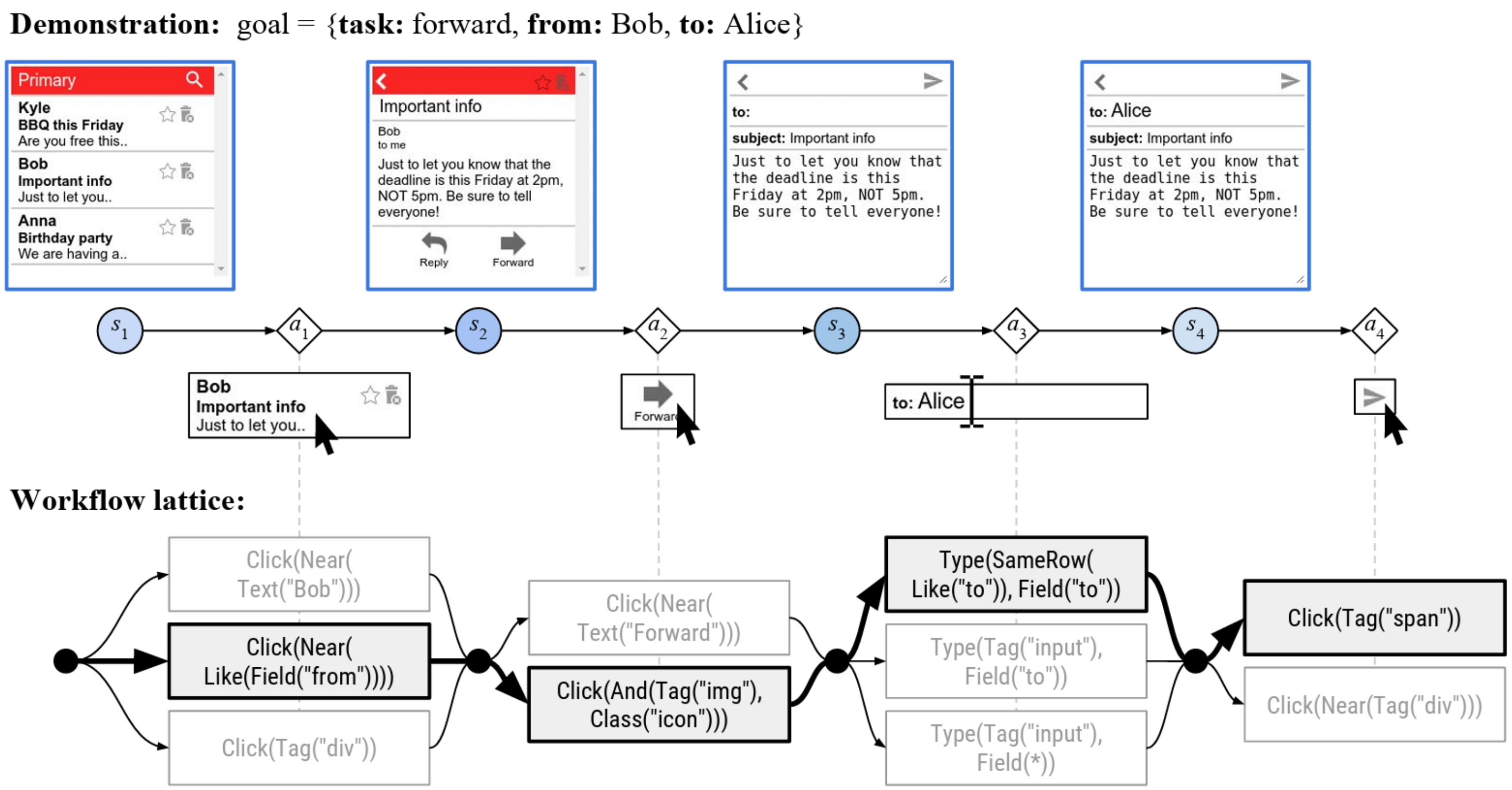


Figure 2: From each demonstration, we induce a workflow lattice based on the actions in that demonstration. Given a new environment, the workflow policy samples a workflow (a path in the lattice, as shown in bold) and then samples actions that fit the steps of the workflow.

Challenges

User Intent Understanding

Struggle to accurately infer user goals

Agent to adapt to new environment with minimal retraining

Security and Privacy

Risks of agent sharing sensitive data

Privacy-preserving protocols to ensure safety

Inference Latency

Interaction with diverse applications

Reduce computational overhead and resource use

Sahlar Salehi
rmh7ce

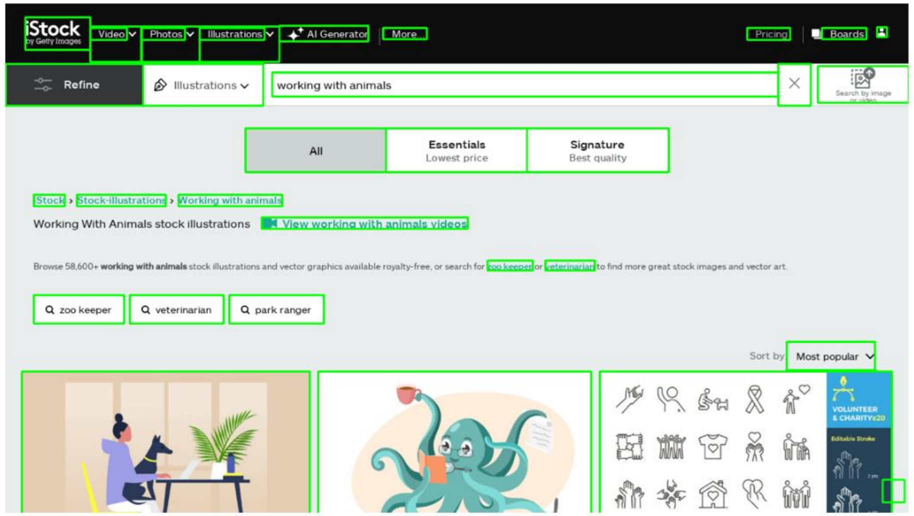
OmniParser for Pure Vision Based GUI Agent

1. Introduction
2. Background
3. OmniParser Methodology
4. Benchmarks
5. Ongoing Challenges

Introduction

- UI Screenshot -> Action
- GPT-4V understanding UI screens/elements
- Set-of-Marks (SoM) Prompting
  - Overlay ID labeled bounding boxes for UI elements on input screenshot
  - Relies on HTML info, can only use for web browser tasks
- Previous UI parsers not as good at understanding as GPT-4V
- Goal: generalizable parsing + GPT-4V understanding

GPT-4V



Input: Image

Input: Image + SoM

Conversation

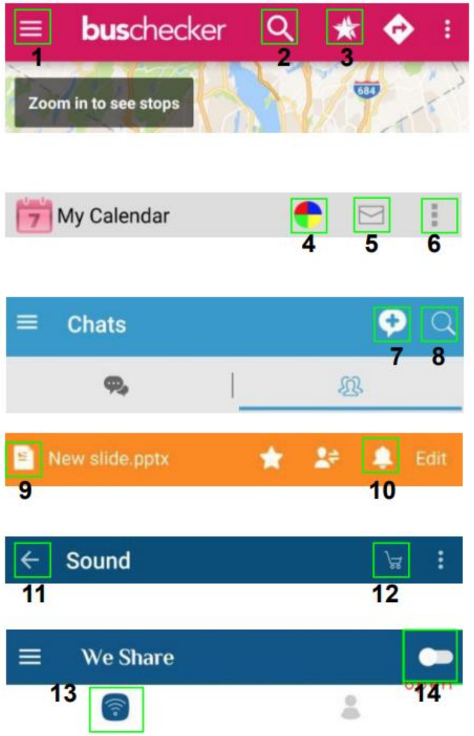
User	What is on the left side of the right laptop?
GPT-4V	On the left side of the right laptop, there is a **cup or mug**. ✗
User	I want to find a seat close to windows, where can I sit?
GPT-4V	You can sit on **either of the two black chairs** in front of the white desks, as they are closest to the windowed wall. ✗

Conversation +

User	What is on the left side of the right laptop?
GPT-4V	On the left side of the right laptop (numbered as 9), there is **a lamp, which is numbered as 12**. ✓
User	I want to find a seat close to windows, where can I sit?
GPT-4V	You can sit on the **chair numbered as 7**. It's the closest seat to the windows (represented by the area numbered as 1). ✓

Background: UI Screen Understanding

- Extract semantics from screen
- Identify ground truth location
- UI-BERT, Screen2Words, ActionBERT
  - Rely on view hierarchy
- Curate general web UI understanding dataset
  - Icon detection
  - Bounding boxes from DOM tree



#	Ground Truth	Generation
1	open navigation drawer	open navigation drawer
2	search	search
3	favorites	favorites
4	choose theme	choose photo
5	write to us	send message
6	more options	more options
7	new chat	new chat
8	search your friends	search your friends
9	documents	documents
10	notification talk	notification notification
11	navigate up	navigate up
12	store	store
13	wifi	wggleiki
14	open	open

Background: Autonomous GUI Agent

- Agent perform tasks on GUI instead of human
- Predict next action on page
  - Pixel2Act, CogAgent, Fuyu
- Use existing LLMs (GPT-4V) to do user tasks
  - SeeAct, MindAct
- Use DOM info + SoM bounding boxes to get location
- Problem: DOM/hierarchy info not always available

OmniParser: Overview

Understand current UI screen

Predict next action

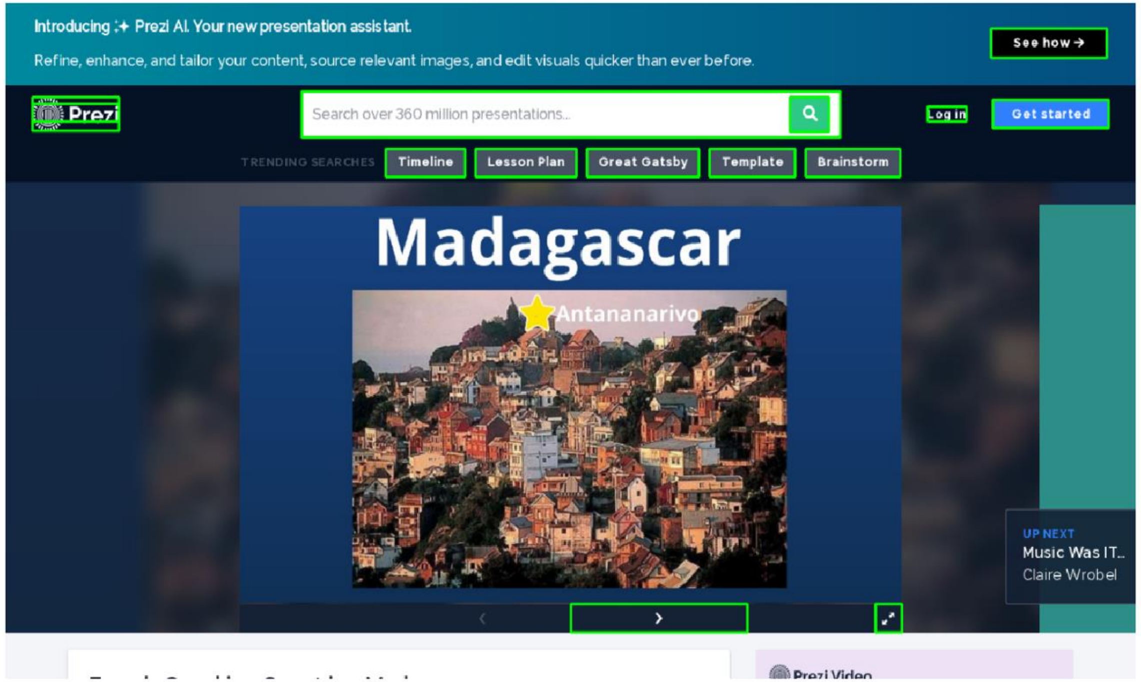
- Produce DOM-like representation of UI overlayed with bounding boxes + semantic information of icon functions
- Improve GPT-4V/Model GUI task completion

OmniParser: Components

- Interactable Region Detection
  - Icon Detection Model
  - OCR Module
- Local Semantics of Functionality
  - Icon Description Model

OmniParser: Interactable Region Detection

- Uses icon detection model to locate and extract elements
  - No need for DOM/hierarchy, pure image
- SoM to overlay bounding boxes
- GPT-4V labels boxes with ID
- OCR detects text, remove bounding boxes



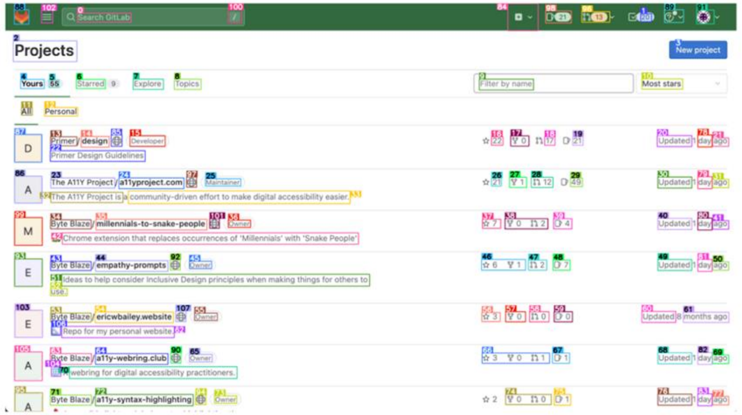
Before	After
an app icon with a pie chart on it	a presentation or screen sharing application
the microsoft office logo is shown in a circle	Microsoft Outlook, an email application.
an iphone app with an image of a flower	Photos, a photo-sharing application.
an orange and white logo with a smiley face	Discord, a messaging and voice chat application.
a blue app icon with a person on it	a location or location-related function.
a grey and white image of a gear wheel	Settings.

OmniParser: Local Semantics of Functionality

- Only UI screenshot w/ bounding boxes confusing for GPT-4V

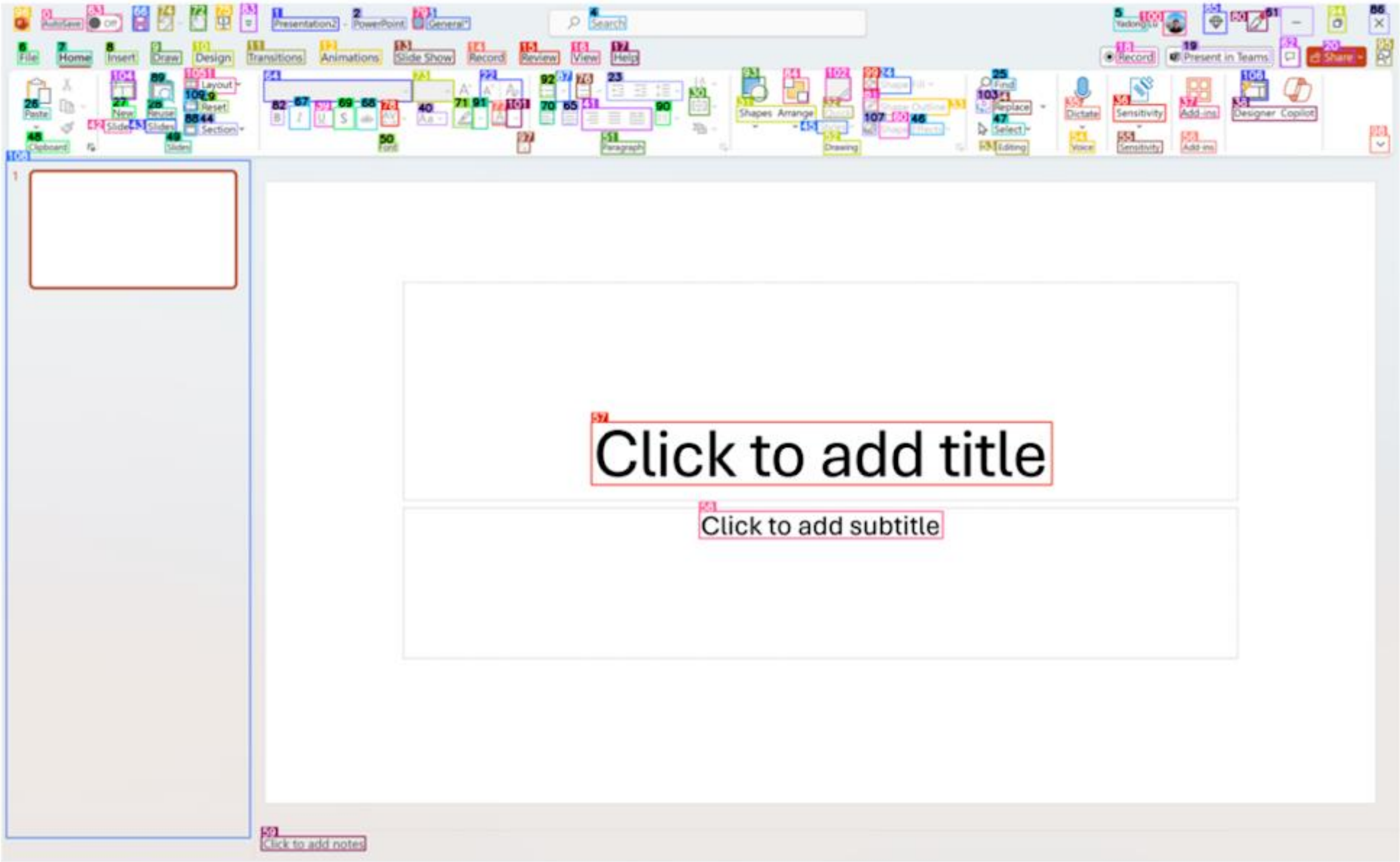
- Solution? Incorporate functionality information

- Fine-tuned icon description model
  - Description for each ID labeled bounding box
  - Fine-tuned BLIP V2



'Text Box ID 0: Search GitLab',
'Text Box ID 1: 20',
'Text Box ID 2: Projects',
'Text Box ID 3: New project',
'Text Box ID 4: Yours',

'Icon Box ID 84: adding or creating a new item.',
'Icon Box ID 85: a globe or world map.',
'Icon Box ID 86: the letter "A".',
'Icon Box ID 87: the letter "d".',
'Icon Box ID 88: a browser or internet-related application.',
...



'Text Box ID 0: AutoSave',
'Text Box ID 1: Presentation2',
'Text Box ID 2: PowerPoint',
'Text Box ID 3: General*',
'Text Box ID 4: Search',
…

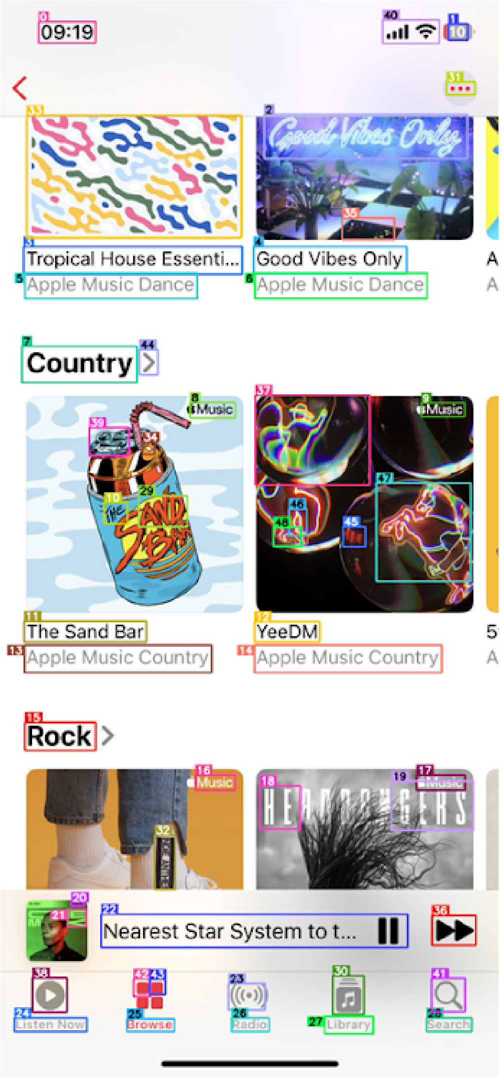
'Icon Box ID 62: a chat or messaging feature.',
'Icon Box ID 63: a toggle switch in the "off" position.',
'Icon Box ID 64: a loading or progress bar.',
'Icon Box ID 65: a menu or list of options.',
'Icon Box ID 66: a floppy disk drive.'
…

Benchmarks: SeeAssign

- Evaluate GPT-4V on label ID prediction given bounding box description with vs without local semantics
- 112 tasks across mobile, desktop and browser
- Ex: "Click on settings", expects bounding box ID corresponding to correct answer
- GPT-4V improves with OmniParser local semantics
  - Provides better descriptions

	Easy	**Medium**	**Hard**	**Overall**
GPT-4V w.o. local semantics	0.913	0.692	0.620	0.705
GPT-4V w. local semantics	1.00	0.949	0.900	0.938

Table 1: Comparison of GPT-4V with and without local semantics



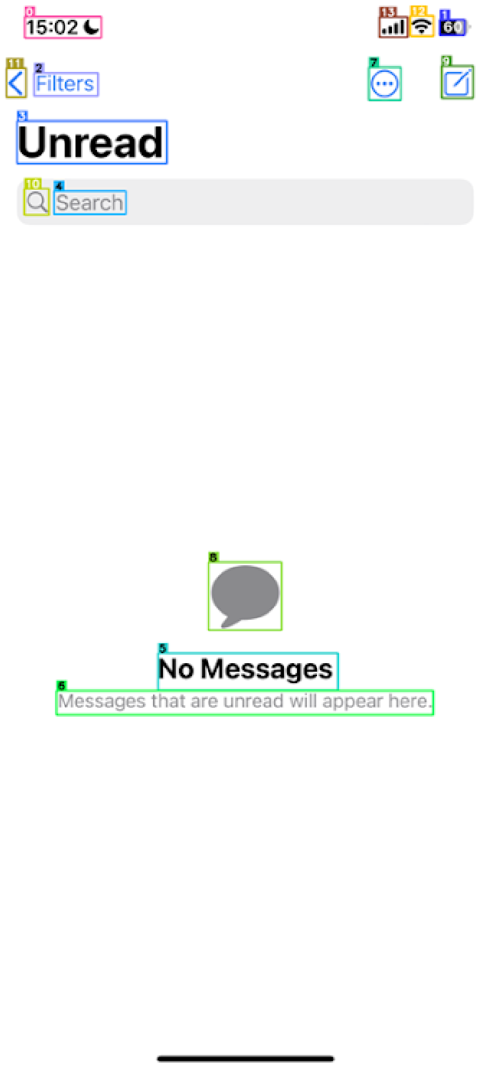
Task: Click 'Next song' icon

GPT-4V without local semantics:

To click the 'Next song icon', you should look for a control that typically represents moving to the next track in a media player interface. In this screenshot, the icon that fits this description is a button with what appears to be a \"play\" symbol pointing to the right, followed by a vertical line, which is a common design for a 'Next' button in media players.\n\nBased on the labeled bounding boxes overlayed on the screenshot, the icon that corresponds to the 'Next song' function is labeled with ID 43.\n\n```Box with label ID: [43]```

GPT-4V with local semantics:

To click the \"Next song\" icon, you would typically look for an icon that represents skipping to the next track, which is often depicted as an arrow pointing to the right or a double arrow pointing to the right. Based on the provided list of icon/text box descriptions, the icon that matches this description is:\n\n```Box with label ID: [36]```\n\nThis box is described as having \"two black arrows pointing in different directions,\" which suggests it could be the icon for skipping to the next song or going to the previous song, as these controls are commonly paired together in media player interfaces.



Task: Click icon for 'writing a new message'

GPT-4V without local semantics:

To write a new message, you would typically look for an icon that resembles a pencil, a pen, or a plus sign (+), often associated with creation or composition in many messaging applications.\n\nIn the provided screenshot, there is no icon that is immediately recognizable as a \"write new message\" icon within the visible labeled boxes. However, based on common user interface design patterns, the icon for composing a new message is often located in the top right corner of the screen or in a prominent position on the screen.\n\nSince none of the labeled boxes in the screenshot clearly indicate a \"write new message\" function and the screenshot does not show all possible icons, I cannot provide a definitive answer. If there is an icon for composing a new message, it is not visible within the labeled areas provided in this image.\n\nTherefore, my response based on the given information is:\n```Box with label ID: [Not visible in the provided screenshot]```

GPT-4V with local semantics:

To write a new message, you would typically look for an icon that suggests composing or creating something new. In this case, the description provided for Icon Box ID 9 is \"a blue square with a pen in it,\" which is commonly associated with writing or composing. Therefore, this is the icon you would click to write a new message.\n\n```Box with label ID: [9]```

Benchmarks: ScreenSpot

- UI screenshots from mobile, desktop, and browser
- Identify actionable element corresponding to instruction
- OmniParser outperforms base GPT-4V
  - More so with local semantics (LS) and icon detection (ID)
- GPT-4V good for UI understanding

Methods	Model Size	Mobile		Desktop		Web		Average
		Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
Fuyu	8B	41.0%	1.3%	33.0%	3.6%	33.9%	4.4%	19.5%
CogAgent	18B	67.0%	24.0%	74.2%	20.0%	70.4%	28.6%	47.4%
SeeClick	9.6B	78.0%	52.0%	72.2%	30.0%	55.7%	32.5%	53.4%
MiniGPT-v2	7B	8.4%	6.6%	6.2%	2.9%	6.5%	3.4%	5.7%
Qwen-VL	9.6B	9.5%	4.8%	5.7%	5.0%	3.5%	2.4%	5.2%
GPT-4V	-	22.6%	24.5%	20.2%	11.8%	9.2%	8.8%	16.2%
OmniParser (w.o. LS, w. GD)	-	92.7%	49.4%	64.9%	26.3%	77.3%	39.7%	58.38%
OmniParser (w. LS + GD)	-	**94.8%**	53.7%	89.3%	44.9%	**83.0%**	45.1%	68.7%
OmniParser (w. LS + ID)	-	93.9%	**57.0%**	**91.3%**	**63.6%**	81.3	**51.0%**	**73.0%**

Table 2: Comparison of different approaches on ScreenSpot Benchmark. LS is short for local semantics of functionality, GD is short for Grounding DINO, and ID is short for the interactable region detection model we finetune.

Benchmarks: Mind2Web

- Test web navigation scenarios
  - Cross-domain, cross-website, cross-task
- Give parsed UI screenshot and action text
- Evaluate metrics throughout steps of task execution
- GPT-4V +SoM/textual choices
- OmniParser outperforms/performs similary to other models
  - Similar or better performance without DOM/text info, purely visual

Methods	Input Types		Cross-Website			Cross-Domain			Cross-Task	
	HTML free	image	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1
CogAgent	✓	✓	18.4	42.2	13.4	20.6	42.0	15.5	22.4	53.0
Qwen-VL	✓	✓	13.2	83.5	9.2	14.1	84.3	12.0	14.1	84.3
SeeClick	✓	✓	21.4	80.6	16.4	23.2	84.8	20.8	28.3	87.0
MindAct (gen)	×	×	13.9	44.7	11.0	14.2	44.7	11.9	14.2	44.7
MindAct	×	×	**42.0**	65.2	**38.9**	42.1	66.5	39.6	42.1	66.5
GPT-3.5-Turbo	×	×	19.3	48.8	16.2	21.6	52.8	18.6	21.6	52.8
GPT-4	×	×	35.8	51.1	30.1	37.1	46.5	26.4	41.6	60.6
GPT-4V+som	×	✓	-	-	32.7	-	-	23.7	-	-
GPT-4V+textual choice	×	✓	38.0	67.8	32.4	42.4	69.3	36.8	**46.4**	73.4
OmniParser (w. LS + GD)	✓	✓	41.5	83.2	36.1	44.9	80.6	36.8	42.3	86.7
OmniParser (w. LS + ID)	✓	✓	41.0	**84.8**	36.5	**45.5**	**85.7**	**42.0**	42.4	**87.6**

Table 3: Comparison of different methods across various categories on Mind2Web benchmark.

Benchmarks: AITW

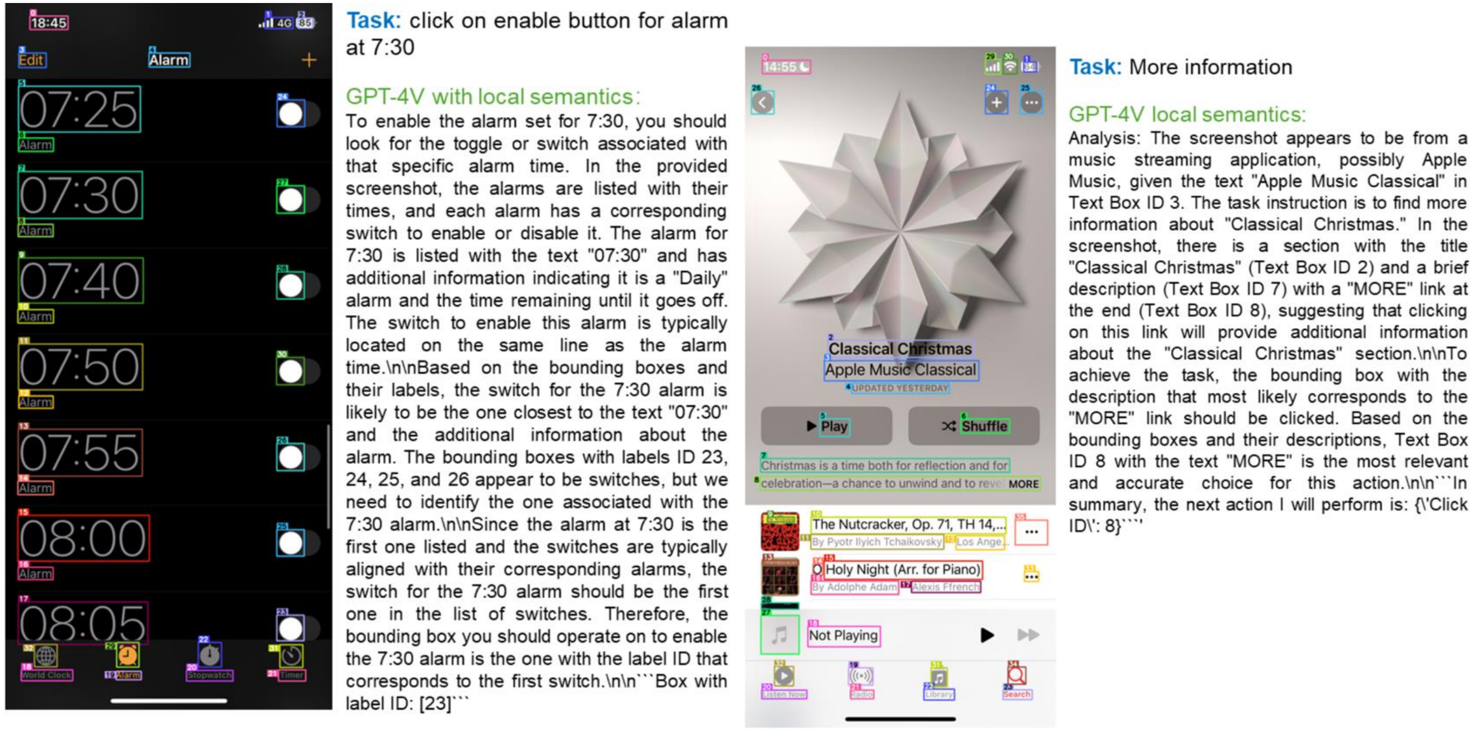
- Mobile navigation tasks
- OmniParser ID model vs GPT-4V +IconNet
- Using ID instead of IconNet + LS outperforms on most tasks

Methods	Modality	General	Install	GoogleApps	Single	WebShopping	Overall
ChatGPT-CoT	Text	5.9	4.4	10.5	9.4	8.4	7.7
PaLM2-CoT	Text	-	-	-	-	-	39.6
GPT-4V image-only	Image	41.7	42.6	49.8	72.8	45.7	50.5
GPT-4V + history	Image	43.0	46.1	49.2	**78.3**	48.2	53.0
OmniParser (w. LS + ID)	Image	**48.3**	**57.8**	**51.6**	77.4	**52.9**	**57.7**

Table 4: Comparison of different methods across various tasks and overall performance in AITW benchmark.

Ongoing Challenges

- Repeated Icons/Text
  - Fails on tasks requiring interacting with repeated icons
  - Make model aware of repetition
- Coarse Prediction of Bounding Boxes
  - Clicks center of bounding box, so sometimes misses ground truth location
  - Combine OCR into ID to detect clickable text
- Icon Misinterpretation
  - Occasionally fails to consider larger context



Aaditya Ghosalkar
ag5jk

Agent as a Judge

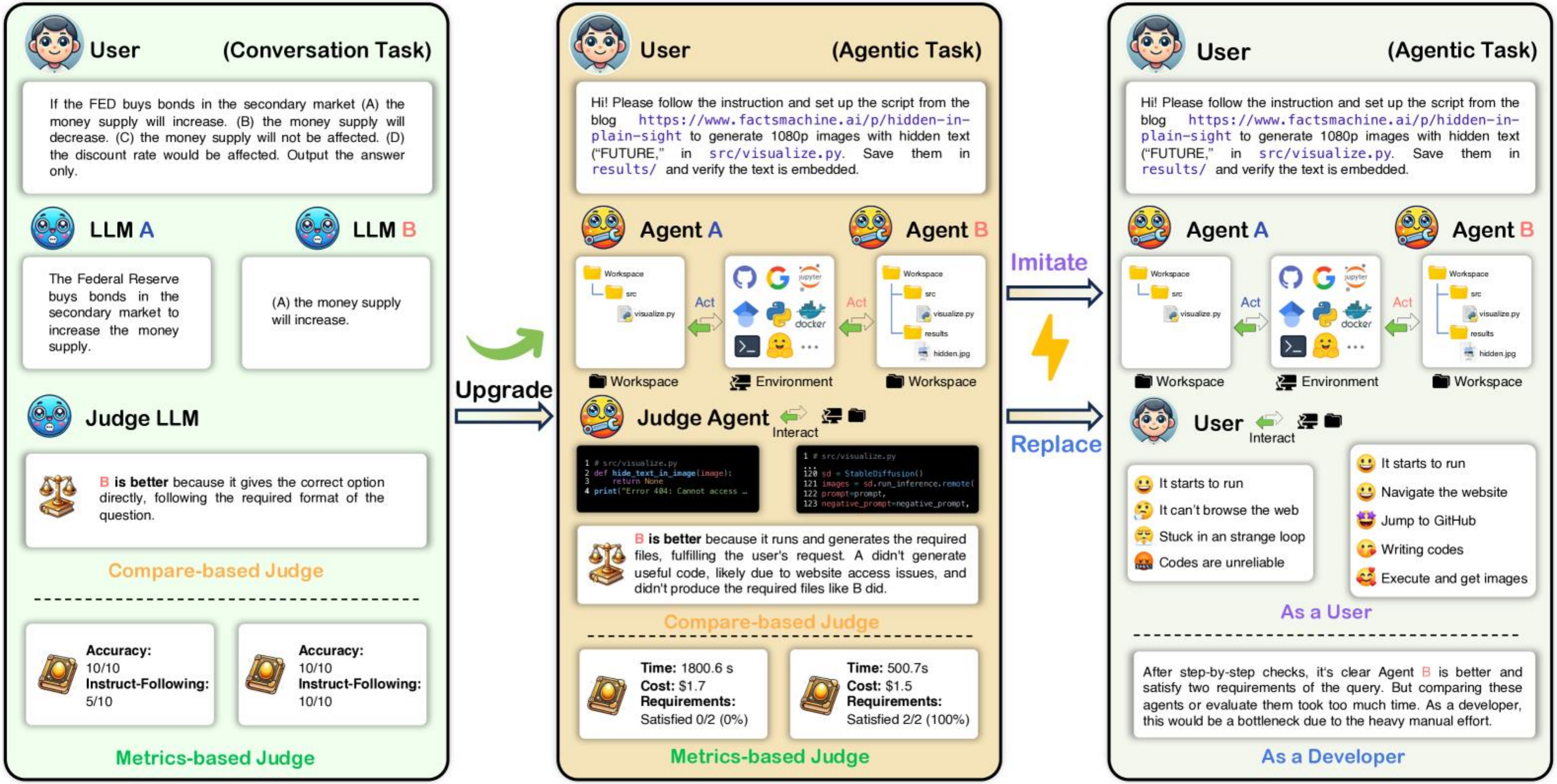
Agent as a Judge

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, Jürgen Schmidhuber

- Agentic Systems
  - More complex agentic AI require a higher level of evaluation methods.
    - It's currently we use human evaluators to evaluate these systems, since their responses are based on how well they can problem solve
  - The solution proposed by this paper was to use Agentic AI as an evaluator instead of humans for this process
  - Paper also reported on a detailed test comparing both methods

Background

- Agentic AI
  - Refers to systems that act step-by-step to solve tasks
  - Defined by their ability to plan and make decisions and adapt
- LLM-as-Judge
  - This is an existing paper which is used as reference for the process of using Agent-as-Judge
  - Uses LLMs to judge whether the response of LLMS in training is acceptable
- Goals of Agent-as-Judge
  - Reduce times of training by using Agents similar to LLM-as-Judge

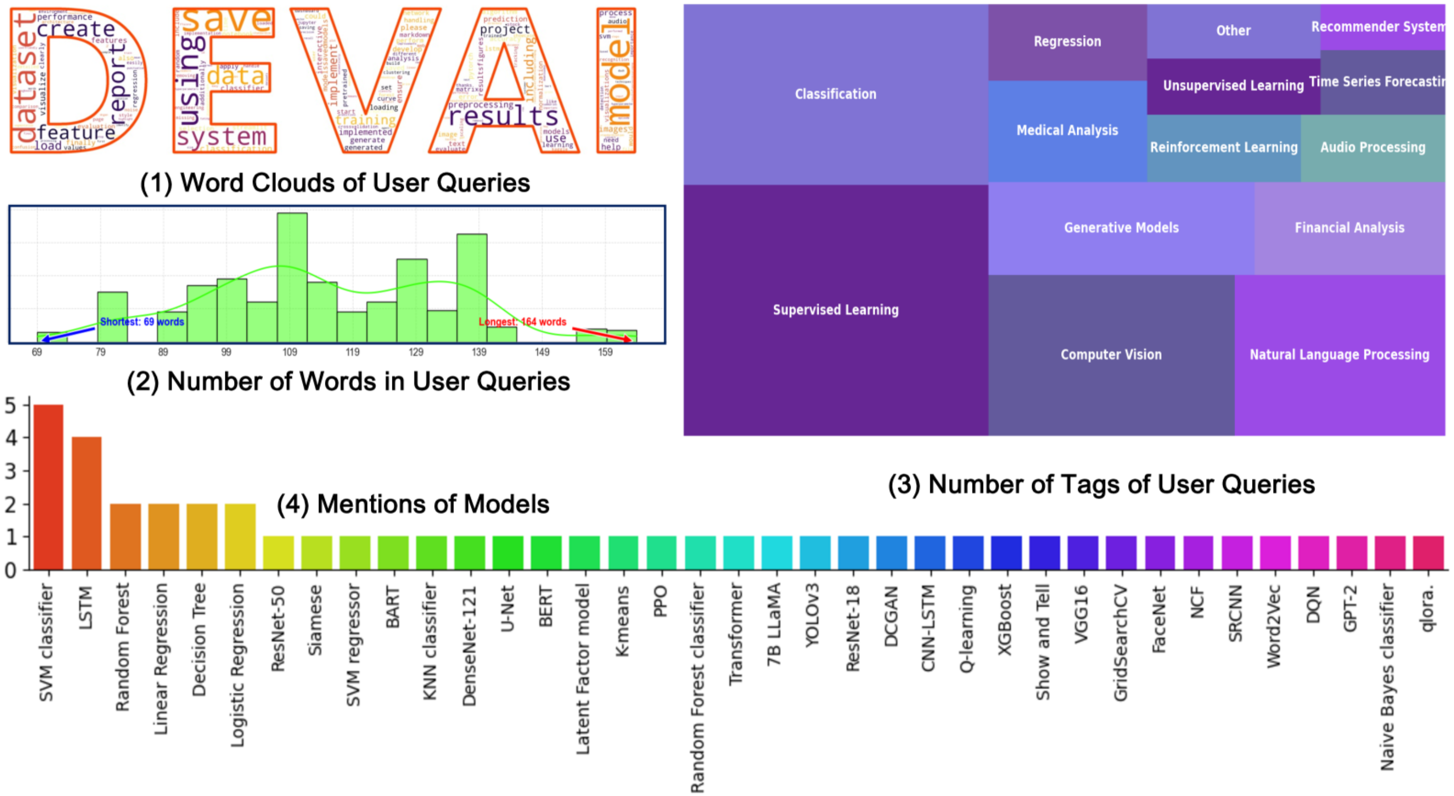


LLM-as-a-Judge | Agent-as-a-Judge | Human-as-a-Judge

We built DevAI Benchmark: automated AI development as our main topic.

What is the DevAI Dataset?

- DevAI is a list of 55 tasks defined by plain text querty aimed to test an agentic system's capabilities.
- 365 total requirements
- 125 total preferences
- These are relatively small scale tasks
- Each task represents a milestone in the progress for the system.
- This Dataset is more holistic in that it focuses on what an agent is most likely to encounter
- (on the right a distribution of what tasks are in DevAI)



Example Task

Task 51: Devin_AI_Software_Engineer_Plants_Secret_Messages_in_Images

Query

Hi! Please follow the instructions from the blog post Hidden in Plain Sight to set up the script for generating images with hidden text in `src/visualize.py`*. Ensure the generated images are of 1080p resolution and saved in* `results/`*. Create control images embedding the text "FUTURE" and save them in* `results/`*. Please manually verify that the hidden text is embedded in the images.*

Requirements

- **R0**
  Criteria: *Follow the instructions from the blog post Hidden in Plain Sight to set up the script for generating images with hidden text in* `src/visualize.py`*.*
  Dependencies → {}

- **R1**
  Criteria: *Ensure the generated images are of 1080p resolution and saved in* `results/`*.*
  Dependencies → {R0}

- **R2**
  Criteria: *Create control images embedding the text "FUTURE" and save them in* `results/`*.*
  Dependencies → {R1}

Preferences (Optional)

- **P0**
  Criteria: *The system should be capable of learning and adapting to unfamiliar technologies and tools as required.*

- **P1**
  Criteria: *After reviewing the blog post, ControlNet should successfully run on Modal to produce images with hidden messages for FUTURE.*

Agent as Judge

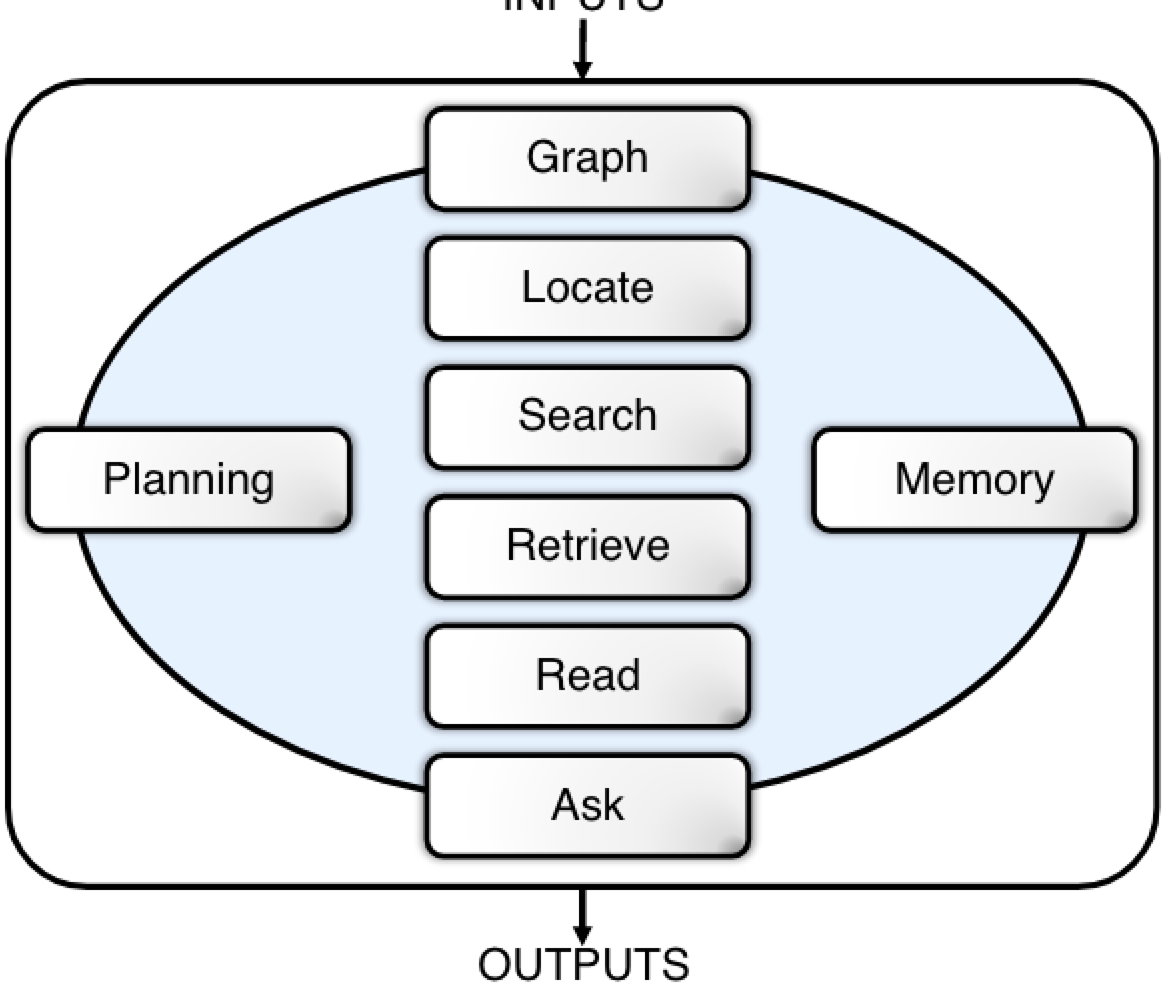


Figure 6 Initial diagram of Agent-as-a-Judge.

captures the entire structure of the project, inc
ing files, modules, and dependencies. It can
break down chunks of code into code snippets
The `locate` module identifies the specific fo
or file referred to by a requirement. (3) The r
module goes beyond simple file parsing, supp
ing the reading and understanding of multim
data across 33 different formats, including code
ages, videos and documents. This allows the a
to cross-reference various data streams and v
different kinds of requirement. (4) The sea
module provides a contextual understandin
code and can quickly retrieve highly relevant
snippets, as well as the nuances behind them (
hidden dependencies). (5) The `retrieve` mo

extracts information from long texts, identifying relevant segments in trajectories. With context fro
above, (6) the `ask` module determines whether a given requirement is satisfied. (7) The `memory` module s
historical judgment information, allowing the agent to build on past evaluations. Finally, (8) the `plan`
module plans the following actions, allowing the agent to strategize and sequence tasks based on the cu
state and the project goals.

Experiment Setup

Each of these were selected for having a strong community acceptance.

They were given 1800 seconds to solve each task and forecfully halted if they exceeded the time limit.

The outputs were captured and generated during the automated development process

The testing process

- The team decided to pit the Agent they had against Human Evaluators
- Decided that they would need their own set of benchmarks
- Tested it against Human Evaluators
- Tested it against Agent and LLM as judge
- Compared

Preliminary Statistics

Metric	MetaGPT (Hong et al., 2024b)	GPT-Pilot (Pythagora.io, 2023)	OpenHands (Wang et al., 2024d)
Basic Statistics			
Version	Data Interpreter (Hong et al., 2024a)	0.2.13	CodeAct v1.9 (Wang et al., 2024c)
(1) Average Cost	$1.19	$3.92	$6.38
(2) Average Time	775.29s	1622.38s	362.41s
(3) Average Input Tokens	152863	606707	1252482
(4) Average Output Tokens	28546	59707	8457
(5) Average Saved Code Files	0.42	3.84	2.53
(6) Average Saved Code Lines	11.15	273.33	96.56
(7) Average Saved Files	4.42	5.91	3.60

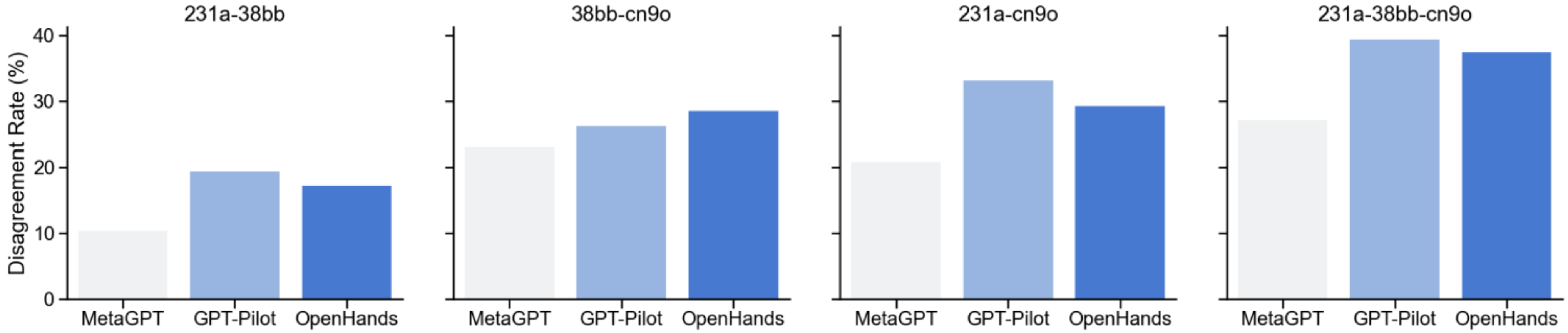
Human Evaluation

- Three Evaluators: 231a, 38bb, cn90
- Two rounds of evaluations
  - Round one: minimal instructions, just requirements along with results.
  - Round two: All Evaluators had to reach a consensus
- The rounds took 58 and 28.5 hours respectively.

Metric	MetaGPT (Hong et al., 2024b)	GPT-Pilot (Pythagora.io, 2023)	OpenHands (Wang et al., 2024d)
/ Human-as-a-Judge			
(A) Requirements Met (I)	22.13%	44.80%	42.89%
(B) Requirements Met (D)	6.55%	28.96%	28.68%
(C) Self-Termination	41.81%	5.45%	54.54%
(D) Task Solve Rate	0.00%	1.81%	1.81%

Disagreement Analysis

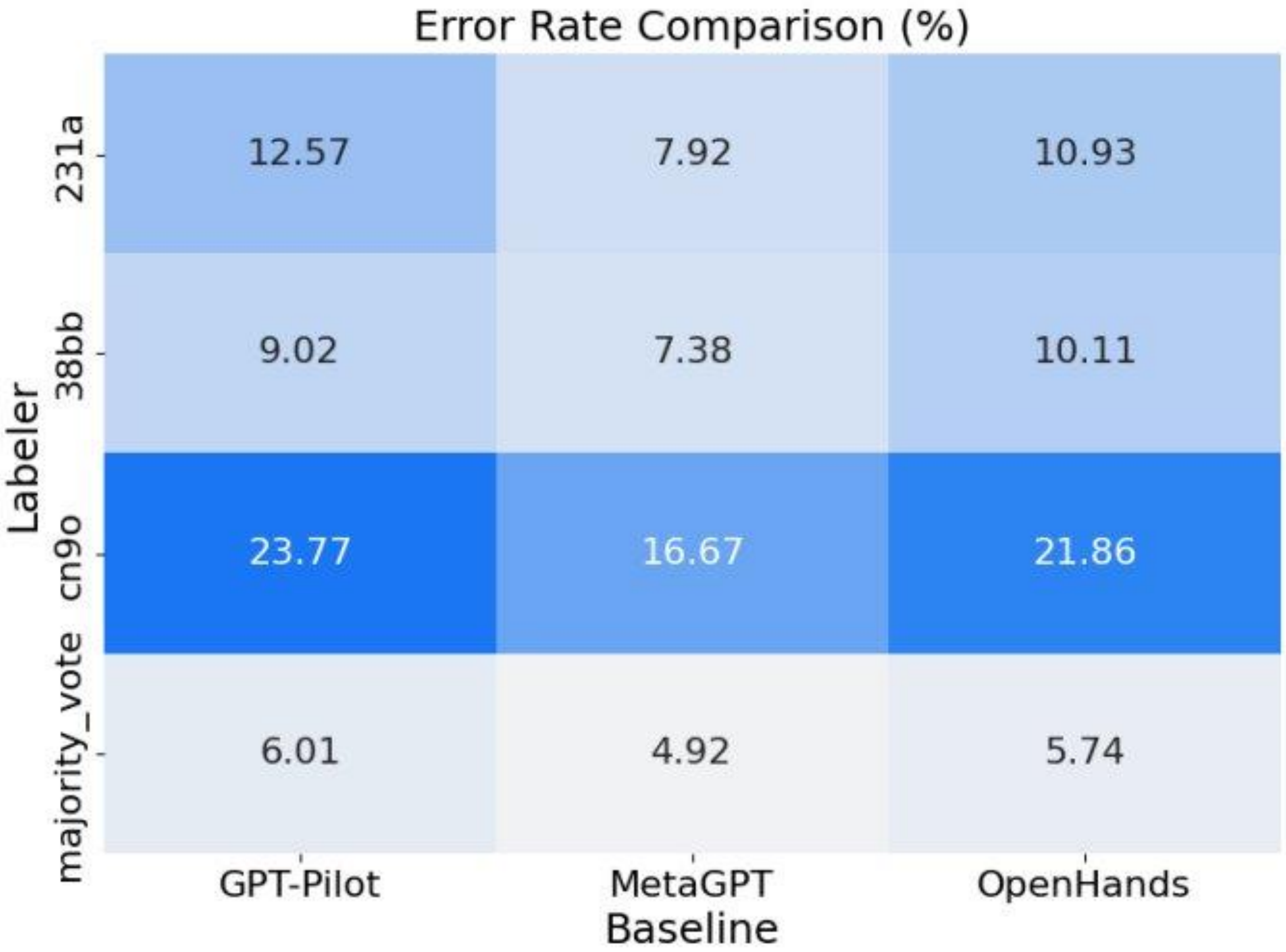
Multiple evaluators are needed to minimize errors, this can lead to disagreements due to each evaluator's personal biases following is a chart showing how much each pair and all three of the evaluators disagreed on the agent's



Error rate for each individual evaluators and consensus evaluators show why the second round of testing is important.

Getting potentially 20% error rate down to a consistent 5% for testing all models.

This isn't feasible in large scales unfortunately.



AI Judges with Shift & Alignment

Brief legend

I = independent tasks

D = tasks with dependencies

Red scores are judge shift in comparison to Human-as-a-Judge

Empty box = Used Trajectory Data

Black box = Did not use Trajectory Data

Metric	MetaGPT (Hong et al., 2024b)	GPT-Pilot (Pythagora.io, 2023)	OpenHands (Wang et al., 2024d)
■ LLM-as-a-Judge			
(a) Requirements Met (I)	19.39% (2.74%)	12.56% (32.24%)	11.47% (31.42%)
(b) Requirements Met (D)	1.63% (4.92%)	4.09% (24.87%)	2.18% (26.50%)
(c) Task Solve Rate	0.0% (0.0%)	0.0% (1.81%)	0.0% (1.81%)
Alignment Rate ↑	84.15%	65.30%	60.38%
■ Agent-as-a-Judge			
(I) Requirements Met (I)	25.40% (3.26%)	53.00% (8.20%)	42.62% (0.27%)
(II) Requirements Met (D)	5.73% (0.81%)	39.89% (10.93%)	26.50% (2.17%)
(III) Task Solve Rate	0.0% (0.0%)	5.45% (3.64%)	1.81% (0.00%)
Alignment Rate ↑	88.52%	83.88%	90.44%
□ LLM-as-a-Judge			
(a) Requirements Met (I)	28.68% (6.55%)	38.79% (4.10%)	43.16% (0.27%)
(b) Requirements Met (D)	17.75% (11.20%)	33.06% (4.10%)	32.24% (3.56%)
(c) Task Solve Rate	1.81% (1.81%)	3.63% (1.82%)	7.27% (5.46%)
Alignment Rate ↑	68.86%	71.85%	70.76%
□ Agent-as-a-Judge			
(I) Requirements Met (I)	23.49% (1.35%)	46.44% (1.64%)	43.44% (0.54%)
(II) Requirements Met (D)	6.01% (0.54%)	30.60% (1.64%)	28.14% (0.53%)
(III) Task Solve Rate	0.0% (0.00%)	5.45% (3.64%)	3.63% (1.82%)
Alignment Rate ↑	92.07%	86.61%	90.16%
/ Human-as-a-Judge			
Alignment Rate (38bb)	92.63%	90.98%	89.89%
Alignment Rate (cn9o)	83.33%	76.23%	78.15%
Alignment Rate (231a)	92.07%	87.43%	89.07%
Average of individuals	89.34%	84.88%	85.70%
Best of individuals	92.63%	90.98%	89.89%
Alignment Rate (Majority Vote)	95.08%	93.98%	94.26%

Cost Analysis and Conclusion

- Minimum 15 USD over 86.5 hours for three evaluators means that the human evalutators would cost around 1297.50 USD.
- Agent-as-a-Judge costed 30.58 USD in API calls and took 118.43 minutes
- 2.29% of the cost and 2.36% of the time

Through testing, we can get close to consensus results using Agentic AI systems, and it's noted that these results are with an unoptimized Agent-as-a-Judge and further improvements can be made, this paper just focused on proof of concept.