Designing Accessible (Effective) Autonomous AI Agents

Yun Wang[*†] Philip Montgomery[*†] Isaac Zheng[*†] Bohan Lin[†] Jun Lim[†] Aaron Zhang[†]

[*] Geni LLC [†] AaSCEND

***Abstract*—Barrier to entry, easy to use (usefulness, usability, effective, solve all NPRE classes, then we have a system that can be replicated everywhere else. This paper presents an analysis of current autonomous systems backed by Artificial Intelligence (AI), mainly Large Language Models (LLM), and a vision for autonomous systems: complex memory, usability, and polymorphism.**

***Index Terms*—Knowledge Graph, Autonomous AI**

I. Introduction

LLMS like GPT-4 [cite] and llama-2 [cite] have shown good results in textual interaction with humans, proving its ability to digest information and produce quality responses. The break-through opens up possibilities in complex AI-human interaction and autonomous systems. Currently, there are many attempts to create the first truly autonomous artificial intelligence. Many seem to fall short in their goal due to focusing too heavily on autonomously accomplishing tasks from one point of view, programmers. In using Autonomous AI, programmers have a distinct advantage : they can fix a bug by modifying particular lines of code, they can use Github repositories and command line arguments, and they simply have a better understanding of what is going on under the hood. In addition, many Autonomous AI's are only designed for coding, but there are many more tasks such as supply chain analysis, legal document reviewing/analysis, and underwriting evaluation of corporations. A truly Autonomous AI should be able to do all those things and also remember what it did and improve on it. For these reasons, we propose Complex Memory, Usability, and Polymorphism (CUP) as three new metrics to analyze autonomous AI systems to see if they are truly moving towards a truly holistic system.

Lay out CUP

- C - Complex Memory (explain it better) (with supporting article) : For a system to be truly autonomous it needs to have memory in order to improve itself from previous sessions, and to work on tasks that may take a long time and require a lot of independent operations.
- U - Usability (with supporting article) : For a system to be usable by a non-technical person, it must have an intuitive UI, where a person with proficient computer knowledge is able to input commands and interface with the Autonomous AI.
- P - Polymorphism (with supporting article) For a system to be truly autonomous, it must be capable of learning how to do new tasks and acting upon them. (Autonomous : We need AI to ask human...)

II. Laying out how the current "Autonomous Agents fail in Accessibility given CUP"

A. Analysis of Framework

B. Analysis of Product

C. MetaGPT

MetaGPT is a multi-agent framework designed specifically for software development. It incorporates SOP (Standard Operating Procedure). It takes the natural language description of the desired software output and outputs a python software package. MetaGPT defines agents and actions with prompts and manages memory retrieval by the memory-generating agent and actions. Each action is hard-coded with memory queries for retrieval. Its working memory is plain concatenated text from previous actions, and it is up to the language model to interpret any formatted information in it. This memory design incorporates an indexing method to gather relevant information for each action and simulate interactions in a software team. But it can lose focus easily as context builds up and code information is not easily interpreted in plain text. MetaGPT uses command line for user-interaction and output intermediate design documents as well as the final product, the code. No testing is performed by the time of writing of this passage. Users have access to the LLM output in the command-line log. As MetaGPT has no breakpoints, users cannot inspect and modify any problematic outputs while running. MetaGPT is specifically designed for python code generation. The structure can be used for large text generation tasks but it requires human design involved.

D. Agents

Agents, an open-source framework, presents promising avenues in developing autonomous language agents. When it comes to usability, its chatbot-like interface provides a user-friendly experience. It also sheds light on an innovative framework that offers users nuanced control over these language agents via a Standard Operating Process (SOP). However, it falls short in providing capabilities for user-defined programming or research tasks, narrowing its application range. Similarly, the nonexistence of planning visualization tools might hinder users from fully understanding the agent's operation. Looking at polymorphism, Agents boasts a flexible agent class

that automatically defines functions, reducing user effort in multi-tasking.

Nonetheless, the memory structure of Agents doesn't mirror its adaptability. Its simplistic memory model - adopting strings to store summaries serving as short-term memory and arrays of messages formulated into long-term memory - proves to be rudimentary. The unavailability of a more complex memory structure like vector database severely constricts its strength in managing complex memory tasks. For example, in large-scale projects encompassing a multitude of agents, the lack of a comprehensive memory framework like a vector database can cause loss of earlier memory data during inter-agent communication. Consequently, this limitation can result in the insufficient capture and applications of valuable information, presenting a significant pitfall.

E. AutoGPT

AutoGPT focuses on showing that AutoGPT can be easily adapted to handle real world situations. They do this by making comparisons between four different LLM models: GPT3.5, GPT4, Claude, and Vicuna. They also show that by bringing in a second opinion from a supervised learner, it can greatly improve the performance of the task. GPT relies on completing complex multi-step tasks without the guidance of humans. They use Additional Opinions algorithm with IL models for tasks which provides a lightweight supervised learning approach.

- AutoGPT relies on GPT4 and takes an average of 50 steps to complete a small task. It costs around $0.288/step, which comes out to around $14 which adds up considerably.
- AutoGPT can't separate between development and implementation. It costs around $14 each time it starts up and can't serialize the chain of events, forcing users to start over again.
- **Polymorphic**: AutoGPT gets stuck in loops due to a limited function set and GPT4's constrained reasoning ability.
- **Usability**: Tedious setup that requires installations of packages such as Git, Python, Docker, and cloning a Github repo.

We can improve this system by implementing asynchronous agents that allow several parts of code to run concurrently without blocking one another.

F. BabyAGI

BabyAGI provides a way to order tasks to reach an objective or task given by the user. BabyAGI can be used to make complex decisions through its re-prioritization algorithm. It provides a list of tasks in order to reach the task or objective given by the user and utilizes the process of re-prioritization by completing the tasks and analyzing the results, making decisions based on these results. This allows for it to be used for decision-making. However, BabyAGI stores the list of proposed tasks in a vector database, which is resource-intensive and can suffer from the curse of dimensionality as it scales. It also only uses training data that is based off of real-world examples and simulations, meaning that its abilities are limited to the insufficient amount of data that it is given. BabyAGI also has no access to the internet, meaning that its functionality is fundamentally limited as well. Additionally, BabyAGI has a lack of accessibility to the average user; to use the application, one must clone the BabyAGI repository, download or have Docker and Python installed, and access an OpenAI key.

G. HyperWrite - Personal Assistant

and displays the web content aside the chat. It explains its reason for its actions and automatically surf the internet. It is capable of searching, interpreting website information, and performing web-surfing actions. HyperWrite displays robustness against failures. It is aware of the success and failure of a certain action on a possibly novel website. On the other hand, HyperWrite can fail under certain unexplored domains, due to possible backend errors.

H. Elicit

Elicit is a research assistant that analyzes research papers by using language models to extract data and summarize the papers. It takes specific steps to increase accuracy and reduce hallucination through fine-tuning the models, double-checking answers, and only searching through existing academic papers. Researchers use Elicit to find papers that they aren't able to find in other places, speed up the process of reviewing literature, and learn about new domains. However, Elicit struggles with identifying facts and is currently unable to answer questions or find general information on a topic that doesn't have information that's explicitly given from a research paper. Additionally, although Elicit is more accessible than other autonomous agents, it utilizes a credit system in which answers can only be generated through the process of paying with credits.

III. Proposal for future Autonomous Agent that fits CUP

Current structures in achieving robustness and versatility:

Objective enriching: Given a short user prompt, the model reflects on the objective and enriches its comprehension and execution. This involves differences in processing and interpretation of the prompt, and the development of a detailed action plan, considering implied undertones and intent of the prompt. It aims at comprehending the larger picture, and the generation of a roadmap that addresses every nuanced requirement.

Task splitting: Given a task, the model splits the task into multiple smaller tasks that each requires a subset of the system's memory and context to solve. This is an exercise in decentralization, which facilitates efficiency and precision in task execution. The model also generates the temporal order for all the tasks, ensuring a logical and coherent sequence of actions that takes into account dependencies and prerequisites between smaller tasks.

Memory indexing: Memories are tracked with their generation context and metadata. This involves a comprehensive

mechanism for filing and retrieving memories, tethered to the context of their generation. This implies a cross-referencing system that operates across multiple dimensions of data, allowing the machine to retrieve and apply memories accurately and effectively.

Complex Memory: The proposition to incorporate graph models for robust retrieval and symbolic reasoning is a substantial enhancement. Graph models aid memory recall by organizing data into nodes and edges for relationships, providing a visual and logical structure that is easy to navigate and process. This promotes efficiency in retrieval and aids the symbolic reasoning process by providing a coherent framework to build upon.

Usability: To achieve an accessible system specified in section 1, we propose the following. Users need to interact with the system via plain text or interactive UI for certain graphical tasks, i.e., form filling. This plain text interaction makes the system accessible to users with varied technical knowledge and proficiency levels, reducing the entry barrier for usage. Moreover, the interactive UI is designed to simplify complex tasks and make them more user-friendly. In addition, the user is capable of adding or modifying the needs during a system run [HyperWrite]. This flexibility is essential to cater to evolving user needs and preferences, and adds a layer of customization and personalization to the system.

Polymorphism: The system should be adaptive to novel tasks. In coherence with the current multi-agent structure, the system should possess an ability to create a copy of itself but with a customized role-action set for the task. This dimension of adaptiveness ensures that the system remains dynamic and versatile, capable of molding itself as per the task at hand. This way, it ensures continuity, context preservation, and responsiveness in its operations.

IV. Conclusion

In summary,