How does AutoGPT work under the hood?

03 May 2023

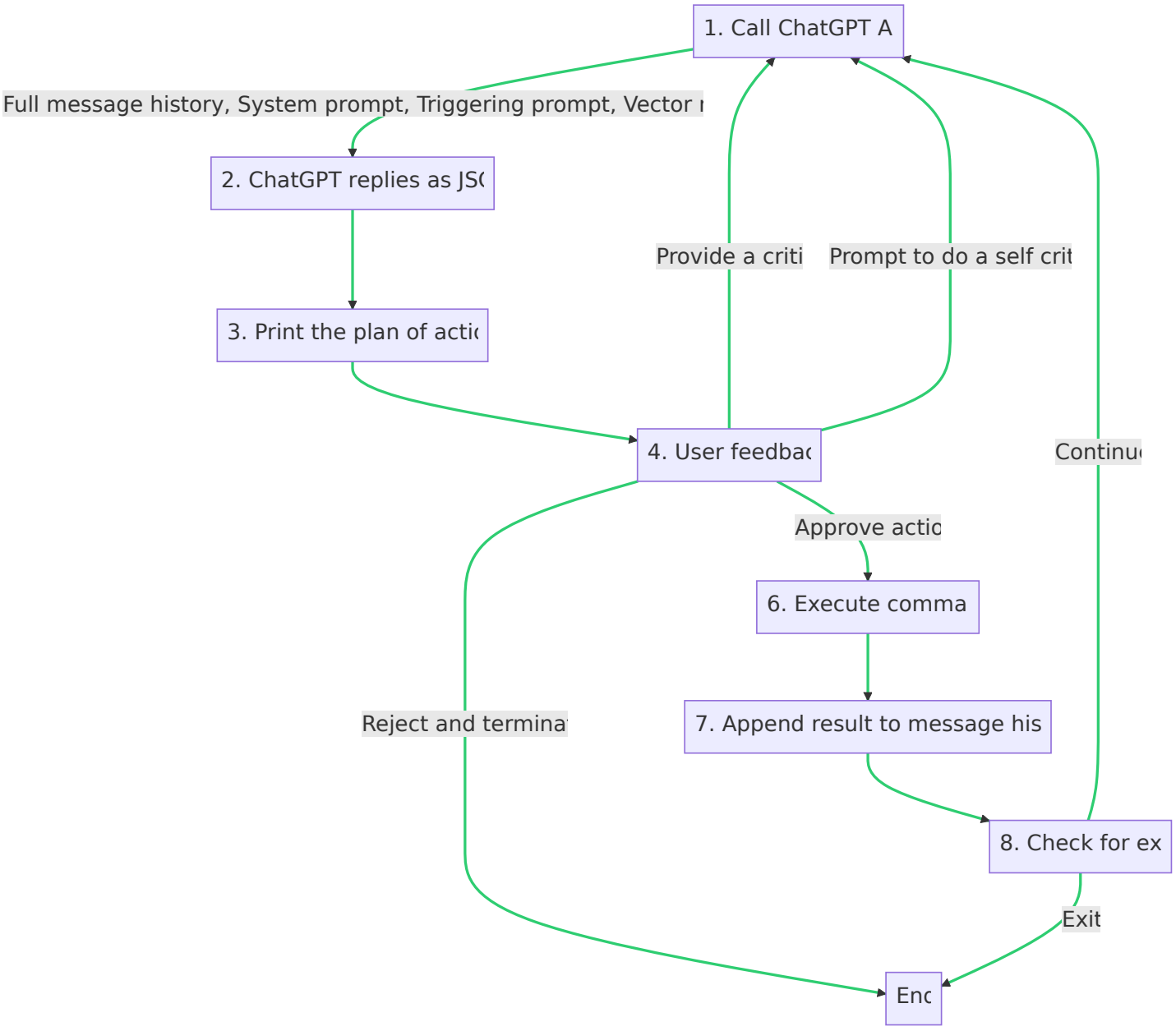
Introduction

There are many youtube videos and blog posts about AutoGPT, but none of them peek under the hood and examine the code itself. So, I have gone through the code and I will provide a quick overview of how it works and some interesting parts of the code which I found to be unexpected. This is not meant to be a comprehensive code overview as that would be too long. I would only cover the fun parts.

High level flow

When the application is started, the user is given a prompt to give the "AGI" a "purpose". Once it gets this objective it tries to make a plan and execute it.

Here is the high level flow




```
Jargon:

Full message history = Entire "conversation" so far. We make it talk to itself.
System prompt = Tells ChatGPT what tools it has and how it should respond
Triggering_prompt = the goal the user wants
Vector memory = long term memory embedded as embeddings
```

1. The process starts with calling the ChatGPT API with the full message history, system prompt, triggering prompt, and vector memory.
2. ChatGPT then replies as a JSON.
3. The plan of action is printed.
4. User feedback is then taken into account. The user can approve the action, provide a critic, reject and terminate, or prompt to do a self critic.
5. If any feedback is provided, the process does not execute anything and goes back to step 1. This is a safety mechanism.
6. If the action is approved, the planned command is executed, such as searching the web or writing code.
7. The result is then appended to the message history as "system".

8. Finally, it checks for exit. If exit is chosen, the process ends. If not, it goes back to step 1.

System Prompt

The system prompt is a constant string is serves as a preamble to the chat message. It has the following parts.

I have dumped this string to a file and attached it here

While it is not part of the attached text document, the system time is also fed in.

Constraints, Resources and Performance Evaluation

The following string is provided as is to the chat model. I think this part is self-explanatory.

```
Constraints:
1. ~4000 word limit for short term memory. Your short term memory is short, so immediately save impo
2. If you are unsure how you previously did something or want to recall past events, thinking about
3. No user assistance
4. Exclusively use the commands listed in double quotes e.g. "command name"

Resources:
1. Internet access for searches and information gathering.
2. Long Term memory management.
3. GPT-3.5 powered Agents for delegation of simple tasks.
4. File output.

Performance Evaluation:
1. Continuously review and analyze your actions to ensure you are performing to the best of your abi
2. Constructively self-criticize your big-picture behavior constantly.
3. Reflect on past decisions and strategies to refine your approach.
4. Every command has a cost, so be smart and efficient. Aim to complete tasks in the least number of
5. Write all code to a file.
```

Commands

AugoGPT interacts with the world using commands, it can be anything ranging from making a web search to even creating sub agents.

ChatGPT responds with a command and arguments e.g. `{"command": "write_to_file", "arguments": {"file_name":"foo.txt", "contents": "Hello world"}}`

This response JSON is parsed and a python function associated with this command is invoked.

```
Commands:
1. analyze_code: Analyze Code, args: "code": "<full_code_string>"
2. read_audio_from_file: Convert Audio to text, args: "filename": "<filename>"
3. execute_python_file: Execute Python File, args: "filename": "<filename>"
```

```
...
21. Task Complete (Shutdown): "task_complete", args: "reason": "<reason>"
```

Here are the categories

```
command_categories = [
    "autogpt.commands.analyze_code",
    "autogpt.commands.audio_text",
    "autogpt.commands.execute_code",
    "autogpt.commands.file_operations",
    "autogpt.commands.git_operations",
    "autogpt.commands.google_search",
    "autogpt.commands.image_gen",
    "autogpt.commands.improve_code",
    "autogpt.commands.twitter",
    "autogpt.commands.web_selenium",
    "autogpt.commands.write_tests",
    "autogpt.app",
    "autogpt.commands.task_statuses",
]
```

Most commands are what you would expect, here are some weird ones

- Execute code command can run in docker container
- "google" search uses duckduckgo
- "google_official_search" uses google

Response format

We ask ChatGPT to respond in a format that can be parsed by JSON parse. However, it does not always return a parseable JSON, so there are helper functions in code to parse *almost* parseable JSONs.

```
You should only respond in JSON format as described below
Response Format:
{
    "thoughts": {
        "text": "thought",
        "reasoning": "reasoning",
        "plan": "- short bulleted\n- list that conveys\n- long-term plan",
        "criticism": "constructive self-criticism",
        "speak": "thoughts summary to say to user"
    },
    "command": {
        "name": "command name",
        "args": {
            "arg name": "value"
        }
    }
}
Ensure the response can be parsed by Python json.loads
```

Here is a sample JSON from the AutoGPT goal "make a successful twitter account"

```
{
    "thoughts": {
        "text": "Now that we have a plan, let's optimize our profile and bio. We should start by upd
        "reasoning": "Optimizing our profile and bio is a crucial first step in building a successfu
        "plan": "- Update profile picture and header image
                 - Write a clear and concise bio
                 - Include relevant keywords and hashtags",
        "criticism": "We need to make sure that our profile and bio accurately reflect who we are an
        "speak": "Let's optimize our profile and bio. We should update our profile picture and heade
    },
    "command": {
        "name": "write_to_file",
        "args": {
            "filename": "profile_optimization.txt",
            "text": "- Update profile picture and header image
                     - Write a clear and concise bio
                     - Include relevant keywords and hashtags"
        }
    }
}
```

Self feedback

Self feedback is an optional step which you can turn on by pressing s when the autogpt prompts you with y/n.

AutoGPT takes the thoughts generated and sends it without any context to a fresh API call and asks for feedback.

The feedback is then just appended to chat history.

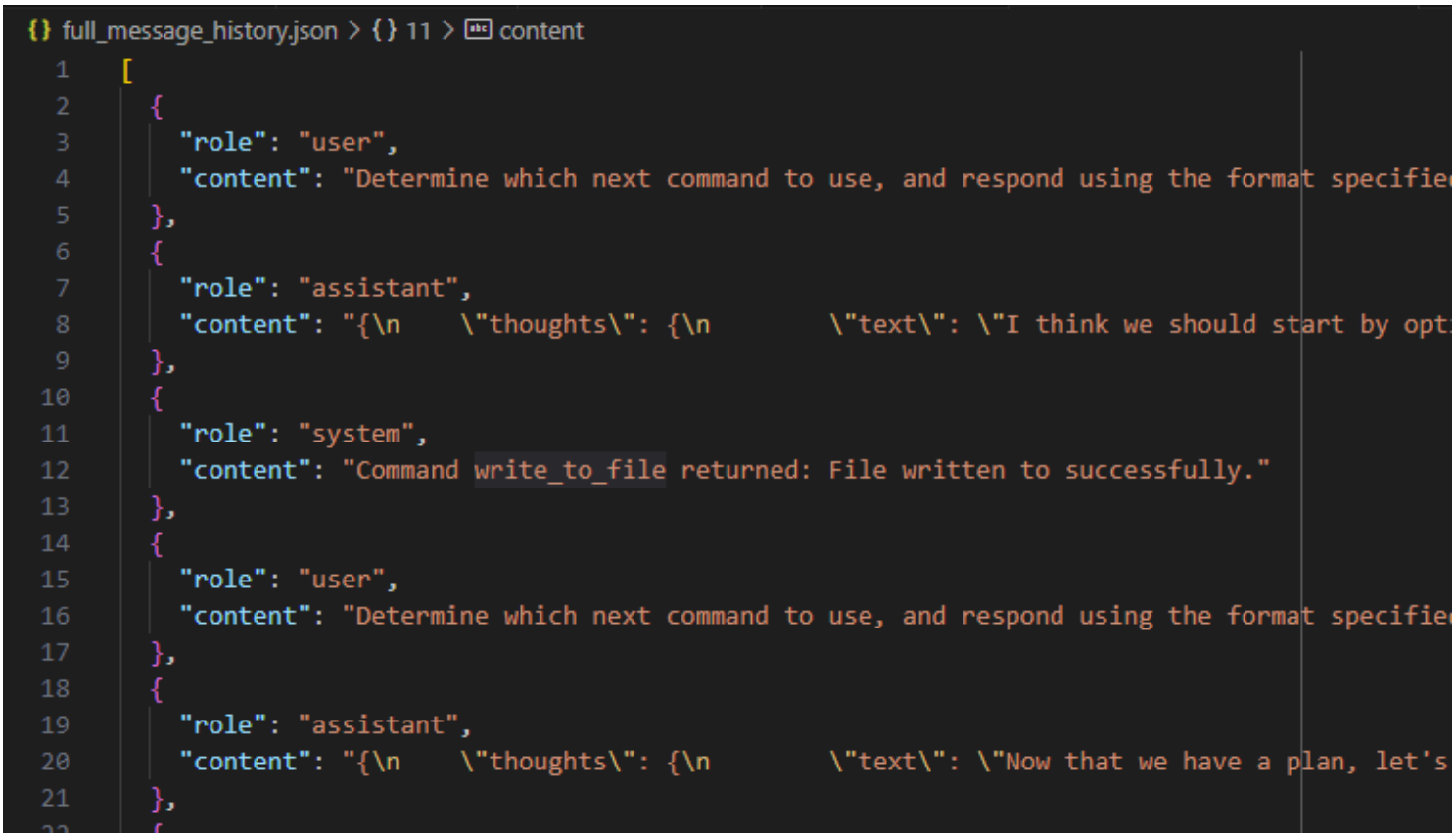
The `feedback_thoughts` are thoughts from the response JSON above.

```
feedback_prompt = f"Below is a message from me, an AI Agent, assuming the role of {ai_role}. whilst

add_to_chat_history([
    {"role": "user", "content": feedback_prompt + feedback_thoughts}
])
```

Full message history

The full message history is the request payload sent to ChatGPT API. It looks like the picture given below.



Vector memory summary

An increasingly popular way to get around context size limitations is using vector databases.

For those who are not aware, a semantic embedding model (usually SBERT, but augogpt uses OpenAI ada model) to generate a vector for each thought.

Semantically similar texts have a higher dot product than dissimilar texts.

Here are the steps for using vector databases as a long term memory

1. Query the vector database with the embedding of current thought
2. Get top 5 most similar thoughts
3. Make a fresh API call to summarize the 5 thoughts into a smaller text
4. Feed the summary along with other things.

Running summary

Running summary is also used to keep track of the actions done so far.

```
# Initial memory necessary to avoid hallucination
self.summary_memory = "I was created."

new_events = [
    {"event": "entered the kitchen."},
    {"event": "found a scrawled note with the number 7"}
]
```

```
update_running_summary(new_events)
# Returns: "This reminds you of these events from your past: \nI entered the kitchen and found a scr
```

Budget notifications

AutoGPT also tells the ChatGPT model to hurry up and wrap things up to keep things within a budget.

```
system_message = (
    f"Your remaining API budget is ${remaining_budget:.3f}"
    + (
        " BUDGET EXCEEDED! SHUT DOWN!\n\n"
        if remaining_budget == 0
        else " Budget very nearly exceeded! Shut down gracefully!\n\n"
        if remaining_budget < 0.005
        else " Budget nearly exceeded. Finish up.\n\n"
        if remaining_budget < 0.01
        else "\n\n"
    )
)
```

Sub agents

AutoGPT can spawn named sub-agents who have a fresh context and an assigned goal. This seems to be another strategy to get around the problem of limited context size.

At time of writing this blog, the agents are not on a subprocess but a synchronized function call. Communication is done in natural language string and is in practice similar to a function call. An agent is called with argument and it returns with results. Each agent maintains its own chat history.

Deciding when to create, list, message and terminate is done using commands discussed above.

Conclusion

As of writing the AutoGPT codebase is still under active development and surely there will be more features added in future.

Infact I plan to rewrite this blog post with AutoGPT. Just sick it on the codebase and create a writeup. It would be interesting to see what things it comes up which I might have missed.

What a time to be alive!

All articles

11 Oct 2025

Fractional Gamma Function via Fractional Derivatives
18 Sep 2025

On the cruel irony of the P-NP problem
18 Dec 2023

The Journey from India to Germany: A Guide for IT Professionals
20 Jun 2023

How does AutoGPT work under the hood?
03 May 2023

Unit Test Recorder - Automatically generate unit tests as you use your application
25 Jun 2020

How to migrate from vanilla Kubernetes to Istio service mesh?
14 Oct 2019