LLM App Intro

9.28

Content

- LLM applications
    - AutoGPT
    - A list of llm applications
- What’s the magic?
    - Prompt templates
- How do we find these applications?
    - Github search
- How do we find these templates?
    - static analysis/instrumentation
- What’s our ultimate goal?
    - A brief intro

AutoGPT

- https://github.com/Significant-Gravitas/AutoGPT
- Demo

A list of LLM applications

- AutoGPT: https://github.com/Significant-Gravitas/AutoGPT
- GPT Academic: https://github.com/binary-husky/gpt_academic
- Task Matrix: https://github.com/microsoft/TaskMatrix
- LangChain: https://github.com/langchain-ai/langchain
- Wolverine: https://github.com/biobootloader/wolverine
- Llama index: https://github.com/jerryjliu/llama_index
- LLM-ToolMaker: https://github.com/ctlllll/LLM-ToolMaker
- Aider: https://github.com/paul-gauthier/aider
- Camel: https://github.com/camel-ai/camel

What’s the magic here?

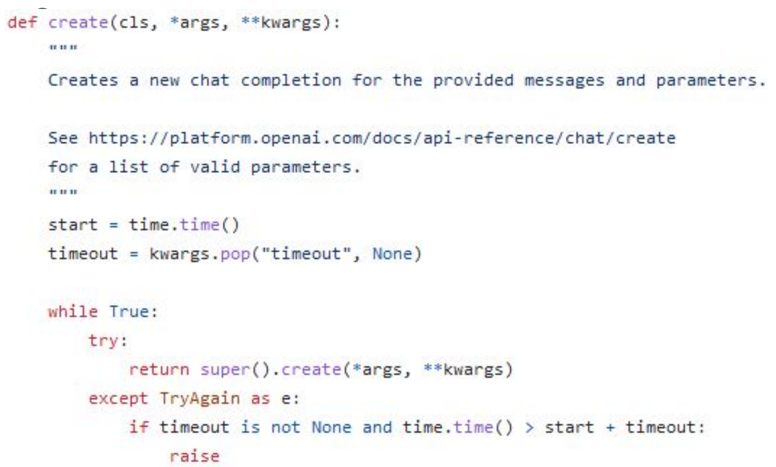
- Call OpenAI API (ChatGPT) with prompt templates
- Back to AutoGPT…

How to collect these applications?

- Search “openai.ChatCompletion.create” with Github API call
- Seem to be easy, but several challenges to automate the collection
    - return limit
    - filter unrelated repos

How to find these prompt templates?

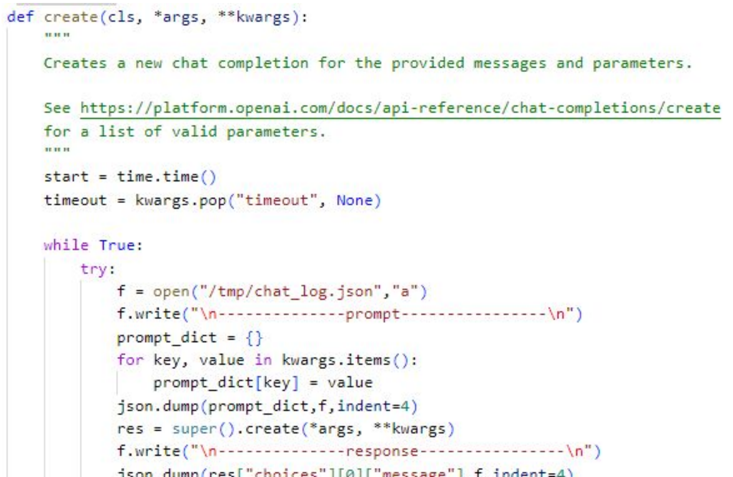
- Static analysis
  - Actually we’ve already done that a little bit…
  - Hard to automate this process…
- Instrumention
  - In all the applications, they use the OpenAI API call: openai.ChatCompletion.create(...)
  - https://github.com/openai/openai-python/blob/main/openai/api_resources/chat_completion.py



```
def create(cls, *args, **kwargs):
    """
    Creates a new chat completion for the provided messages and parameters.

    See https://platform.openai.com/docs/api-reference/chat/create
    for a list of valid parameters.
    """
    start = time.time()
    timeout = kwargs.pop("timeout", None)

    while True:
        try:
            return super().create(*args, **kwargs)
        except TryAgain as e:
            if timeout is not None and time.time() > start + timeout:
                raise
```



```
def create(cls, *args, **kwargs):
    """
    Creates a new chat completion for the provided messages and parameters.

    See https://platform.openai.com/docs/api-reference/chat-completions/create
    for a list of valid parameters.
    """
    start = time.time()
    timeout = kwargs.pop("timeout", None)

    while True:
        try:
            f = open("/tmp/chat_log.json","a")
            f.write("\n-------------prompt---------------\n")
            prompt_dict = {}
            for key, value in kwargs.items():
                prompt_dict[key] = value
            json.dump(prompt_dict,f,indent=4)
            res = super().create(*args, **kwargs)
            f.write("\n-------------response---------------\n")
            json.dump(res["choices"][0]["message"],f,indent=4)
```

What’s our ultimate goal?

- Let’s say we have two prompts:
    - My name is Jiayi. Please give me a cool nickname.
    - My name is Junchen. Please give me a cool nickname.
- In GPT models, we have to do calculations for each token.
- Furthermore, later tokens are dependent on previous tokens.
    - “Please” for Jiayi and Junchen are different
- Our current idea: only recompute some of the imporatnt later tokens
    - E.g., only recompute “cool name”