LLM Agents

Yashila Bordag

What are Agents?

Agent - “An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.”

- Agent = architecture + program
- Agent Program/Agent Function - the algorithm which decides what actions to take based off of the current set of precepts (observations)

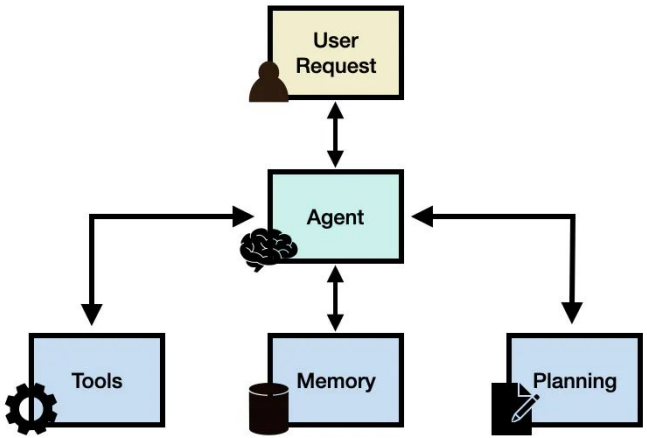
Goal-based agent - an agent which is given a particular goal state and uses the current model/precepts to choose actions which minimize the distance to the goal

Utility-based agents - an agent which calculates a rough heuristic based on previous experience and distance from the goal state to choose actions

(Russel, Norvig, Artificial Intelligence A Modern Approach, 4th ed.)

What are LLM Agents?

LLM Agent - a system which can execute complex tasks using a structured prompting system which recursively calls an underlying LLM instance



- system usually includes modules for planning, memory, tool use, and retrieval
- LLM acts as central agent function
- A well-structured and accurate user request is the agent’s goal

Limitations of LLMs

Why do we need to create so much structure for LLMs to solve these tasks?

- LLMs have limited single step reasoning capacity (although it is growing larger with better models), and with complex queries and too many entities, might hallucinate answers or forget to answer parts of questions
- Without structure and prompt guardrails, LLMs can hallucinate and produce incorrect answers, poorly reasoned answers, or provide irrelevant information
- Allows us to add real time data and inputs into LLM decision making process
- Allows for observability into the LLM ‘thought’ process
- Dramatically improves the abilities of even relatively small LMs (~7B parameter models) making these systems accessible to a wider variety of compute environments

Agentic Reasoning - Chain of Thought, Tree of Thought

LLMs can generate seemingly plausible reasoning when prompted to explain something. This ability was explored and augmented in the Chain of Thought and Tree of Thought papers among others

- Chain of Thought shows that LLMs produce considerably better outputs when prompted to provide reasoning based on a few incontext examples.
- Self-consistency showed that when generating several reasoning chains, the chains which had the modal answer were considerably more likely to be correct
- Tree of Thought explored the use of “thought” trees which allowed for backtracking on reasoning chains, which showed considerable improvement in complex tasks involving planning

Agentic Reasoning - ReAct, Chain of Abstraction

LLMs tend to perform better and hallucinate less when provided structured inputs.

- ReAct explores the use of action and observation interleaving with planning to improve LLMs ability to solve complex tasks. This closely parallels the hypothesis-action-observation paradigm which drives many utility-based models.
- Chain of Abstraction builds on previous work with agentic tool use and ReAct to train LLMs to create abstract reasoning chains for which tools can be selected and then observations can be gathered to continue the reasoning process

LLM Tool Use

LLM Reasoning Performance

- Pros : Verbal Reasoning
- Cons : Non-verbal Reasoning (eg. Quantitative Reasoning), Reasoning on out-of-distribution data

However, LLMs can use verbal reasoning to abstract out inputs for well-defined tools (functions) and use the observed outputs (eg. retrieved data) to complete tasks

Some papers also examine having LLMs define the subroutines necessary to gather necessary data for reasoning (PAL, Program of Thoughts)

Agentic Memory - Motivations

LLMs have two primary memory issues:

- Context length
- Prioritizing context information

Agentic systems tend to generate very long contexts lengths via the prompts and outputs from the reasoning process. Without appropriate context summarization, only models with very long context windows could be used for agents.

Several implementations of agents allow agents to write in a ‘scratchpad’ which is appended to the prompt window for the next reasoning step

MemGPT develops the concept of virtual context management which parallels virtual memory management done by operating systems.

Agentic Memory - Techniques

Main goal of memory management would be to optimize on reducing retrieval complexity for relevant information

- Agent Scratchpad - agents are allowed to write whatever is deemed necessary
- RAG system - agents are given access to a data source (such as a vector database) and allowed to store and extrapolate information from this source
- Knowledge Graph RAG system - similar to the last technique, knowledge graphs can be more effectively interfaced for both relevant information and conceptually indirectly relevant information, allowing for more complex outputs

Multi-agent Systems

Agents can be defined for specific tasks and asked to complete these tasks in order or in tandem, providing information to each other. This can take on a few different forms:

- Multi-agent systems - these define simple agents which are orchestrated to complete super-tasks given a sub-task workflow (AutoGen, CrewAI, LangChain, etc.)
- Agentic interfaces for single agent systems - this approach proposes creating access points for more complex agents to talk to make requests

Review of Agentic Ability / Summary

How agents improve on LLMs

- Have much deeper, more directed reasoning ability and are able to parse considerably more difficult material and tasks
- Have access to tools and can gather information indirectly
- Can be connected to very large data sources

Applications:

- Structuring Data
- Automating Workflows
- Complex Data Generation

Review of Agentic Ability / Summary

Agents

- Can access large amounts of data
- Can process unstructured data
  - Generate informed content
  - Structure incoming content
  - Extract subtext or make other conjectures about content
- Can repeat workflows at very large scale

Big Picture View on Using Agents

For Agents to work consistently they need guardrails, highly structured prompts, goals and higher level reasoning inputs

How should agent scope be defined and used?