

# Agent Based Workflows via Dungeons and Dragons

Alexander C. Mueller

June 2024

# Agenda

## Context and Introduction

- Goals and philosophy of this presentation
- Architecture of the Ale and Antler demonstration
- Caveats for bean counters and pessimists

## Dungeon Master via Agents

- Three sub-tasks of dungeon mastery
- Present deficiencies and how to fix them
- Relation to “agents” in libraries like LangChain

[https://github.com/capnion/random/blob/master/acm\\_capnion\\_11june2024\\_stlgenai.pdf](https://github.com/capnion/random/blob/master/acm_capnion_11june2024_stlgenai.pdf)

# WTF is an “Agent” lolol

I like to ask questions like...

What problems might I encounter while building an LLM-driven application that would lead me to invent “agents”?

My philosophy regarding first approaches to arcane topics is...

- problems are more enlightening than solutions
- practical activity is more enlightening than abstraction

This presentation seeks to present a **tangible, familiar, and fun problem** that will give you a sense of the “why” surrounding a young concept that is not always terribly well-explained.

# The Ale and Antler Inn

We've used the ChatGPT API to build a simple toy in a text-based, high fantasy, tabletop role-playing game style to use as a conversation piece. (Jack Phillips did the real work.)

You are encouraged to play with it now.

<https://tiqlish.com/dnd/character/>

**Advertisement and Warning:** The characters often remember conversations and context if you come back later. Keeping your confidence is not a capability we have built yet.

I'm going to point out some challenges we encountered and how they relate to concepts in LLM development.

# Architecture

The Ale and Antler uses retrieval augmented generation (RAG):

- **Generation** via the **ChatGPT** API,
- **Retrieval** via a **Pinecone** vector database...
- ... plus use of a garden variety vector embedding.

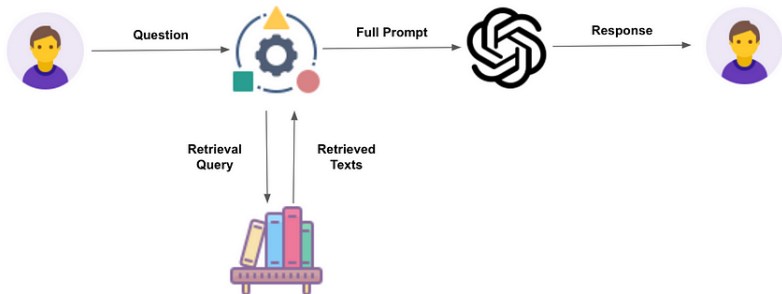
Serving the application is accomplished via...

- **Django** and its standard adjacent stack, on an...
- **Ubuntu** Linux server hosted on **Digital Ocean**.
- Our code is written almost entirely in **Python**...
- With a splash of **Javascript**.

No (non-)endorsement of any vendor or tool is intended.

# RAG for Dummies

Retrieval augmented generation (abbreviated RAG) is where you give the LLM something like a library and tell it...



"Don't be an expert, be a reference librarian."

## Business-esque Concerns

The Ale and Antler was very easy and cheap to create because we cut corners a megacorporation wouldn't be comfortable with. With guardrails, the cost could balloon massively.

Notable flaws, from a megacorporate perspective, include:

- Data flying around everywhere (OpenAI, Pinecone)
- Doesn't detect inappropriate behavior from users
- Difficult to make guarantees about edge-case behavior
- Just not that great a dungeon master

Today I am going to talk about my plans to make it a better dungeon master. The big idea is that a dungeon master is actually a bundle of functions and each should be its own agent.

# ?Prompt Engineering?

Is prompt engineering a “real” discipline?

Do we agree on what the phrase should even mean?

A few humble observations:

- Prompting has a learning curve.
- You can invest considerable energy refining prompts.
- All challenges are turbocharged if you are using a prompt template that gets filled in according to the user’s behavior.

Prompts are definitely “expensive” at times and you want to be a good curator of your own past efforts.



# Three Tasks for the Dungeon Master

All of these tasks will pass through a prompt template:

- Voicing characters and providing descriptions
- Enforcing the setting and its internal rules
- Rolling dice and (possibly secretly) evaluating results

It's tempting to try to combine two functions into one prompt, but after a point this clearly doesn't work.

*Informal Definition: An agent is a structure in your code that helps you manage complexity by grouping related functions.*

# Ale and Antler Enforces Setting Poorly...

I want to give a presentation on generative AI to the tavern.

DM:

Tavern Keeper: Grug, the Barbarian, steps up onto a platform in the tavern, a hush falling over the crowd as they turn their attention to him.

Grug: "Greetings, my fellow patrons! Today, I bring you a tale of magic and wonder, a tale of Generative AI. This powerful tool has the ability to create stories, art, music, and more with just a few lines of code."

He goes on to explain the basic concepts of generative AI, its applications in various fields, and the potential impact it can have on the world.

As Grug shares his knowledge with the eager audience, the tavern becomes filled with curiosity and excitement. The patrons listen intently, asking questions and engaging in discussions about this fascinating technology.

After his presentation, Grug is met with applause and cheers from the impressed crowd. The tavern is abuzz with chatter about generative AI, and the patrons are inspired by the possibilities it holds.

As the evening continues, discussions about AI, magic, and innovation fill the tavern, creating an atmosphere of intellectual curiosity and creativity. Grug's presentation has sparked new ideas and conversations among the patrons, leaving them with a sense of wonder and inspiration.

# ...Even as ChatGPT Knows Better



**You**

you are the dungeon master for a tabletop rpg in a high fantasy setting. is "I want to give a presentation on generative AI to the tavern." appropriate to the setting. answer concisely.



**ChatGPT**

In a high fantasy setting, presenting on generative AI in a tavern would likely not be appropriate, as the technology and concepts of AI wouldn't exist in that world.



**You**

how would a tavern keeper respond if a player character said "I want to give a presentation on generative AI to the tavern." in such a setting?



**ChatGPT**

The tavern keeper would probably look at the character bewildered, perhaps even suspiciously, before responding with something like, "Eh? What's this generative AI you're babbling about? If it's not about ale or swords, I doubt anyone here would care to listen."



# A Setting Enforcement Agent

My life will be easier if I can invoke a function, class, or similar entity and obtain a standardized judgment on whether a behavior is permissible inside the game.

Compare setting appropriateness vs. generating dialog...

- Output schema are different
- Prompt templates (probably) have to be different
- One function invariably precedes the other
- Both are constantly used by the dungeon master

One way or another, the task of enforcing the setting should be compartmentalized away from dialog. This (really quite old and widely known) idea is what motivates agents.

## A Second Opinion from LangChain 1

LangChain has a notion of a **tool** (aka **function calling**). Tools, roughly speaking, process text to provide specific inputs for functions you have defined elsewhere.

Imagine you are dealing with inputs like...

"I rolled three D sixes and all I got was 1s."

A tool is a construct that takes this input, hopefully returns the vector (1,1,1), and then inputs this vector into another function.

An agent in LangChain **applies a sequence of tools according to predetermined logic and process control.**

## A Second Opinion from LangChain 2

Only one of my three dungeon master functions is currently complicated enough to require LangChain.

LangChain's notion of a tool presumes I want to do further processing of information in my prompts (which come from users via templates).

In your favorite tabletop RPG, you might end up doing all sorts of things to numbers from dice including adding these numbers, counting rolls above a certain threshold, etc.

Agents are really being defined for the first time right now, but the dungeon master's dice rolling and evaluation function is a prototypically good place to apply an agent-based approach as imagined by LangChain.