Stuffing AI into our programs

Admin

- Any Questions on project1 sprint 2?
- Faculty Hiring: We want your feedback
  - Each comp490 section will have one candidate
    - T/R 11: Thursday Feb19
    - T/R 12:30: Tuesday Feb 17
    - M/W 12:20: Mon Feb 23
  - Cyber faculty candidates coming soon
- Assignment for the “stepping into the industry” portion of class:
  - The programming podcast from Sept 11, 2025 “The BIGGEST Reason Some Devs Get More Interviews Than Others”
  - https://www.youtube.com/watch?v=_LOz3YBm73A
  - https://creators.spotify.com/pod/profile/the-programming-podcast/episodes/The-BIGGEST-Reason-Some-Devs-Get-More-Interviews-Than-Others-e382v6r
  - etc

AI

- What is ‘AI’?

AI

- What is ‘AI’?
  - “Artificial intelligence refers to computer systems that can perform complex tasks normally done by human-reasoning”
  - Source NASA (https://www.nasa.gov/what-is-artificial-intelligence/) 2926
  - Self defeating definition, but the best/ most common one I’ve seen over decades

LLM

- What is an LLM? (Large Language Model)

LLM

- What is an LLM? (Large Language Model)
  - (I’m sure Dr. Kumari would scold me for over simplification)
  - A supervised deep learning model based on neural networks that ingests huge amounts of data to train (and has a *lot* of nodes)
  - Finds patterns in the very large piles of input data

Generative AI

- What is ‘Generative AI’?

Generative AI

- What is ‘Generative AI’?
- An AI system (pretty much all LLMs in the mid 2020s) which can generate novel output* based on a prompt and the patterns it has learned in training.
  - The ever increasing prompt is often called the ‘context’

* for certain definitions of novel output

Generative AI

- What are some of the positive capabilities of Gen-AI in the 2020s?
  - At least things it can do well, even if you don't think it is a benefit to society

- What are some of the knocks on Gen-AI?
  - Particularly things it does famously poorly/improperly

Generative AI

- What are some of the positive capabilities of Gen-AI in the 2020s?
  - At least things it can do well, even if you don't think it is a benefit to society
    - I look forward to what each class has to say, because there are a lot of things here
- What are some of the knocks on Gen-AI?
  - Particularly things it does famously poorly/improperly
    - There are a lot of things here too but I particularly want to talk about "Hallucinations"

AI ‘Hallucinations’

- What are AI ‘Hallucinations’?

AI ‘Hallucinations’

- What are AI ‘Hallucinations’?
  - These generative AIs just statistically generation the most likely outcome based on the patterns it found in input data
  - Famous first published case of AI Hallucinations
    - “list 7 law professors accused of sexual harassment”
    - When the model didn’t have 7, it just added some famous law professors names ot the list back in 2023
      - Note the reference to 2018 news articles
    - https://www.firstpost.com/world/chatgpt-makes-up-a-sexual-harassment-scandal-names-real-professor-as-accused-12418552.html
    - I’m sure they fixed it right?
      - https://www.spotlightpa.org/news/2026/01/pennsylvania-commonwealth-court-ai-hallucinations-allegations-justice-system/
  - How about in our field of software dev?

AI Software Hallucinations

- What format do AI software hallucinations take?
- What are the outcomes?

AI Software Hallucinations

- What format do AI software hallucinations take?
  - Methods in libraries that don’t exist
    - (though similarly named might exist in other libraries)
  - My personal annoyance
    - Methods that used to exist but were deprecated an removed
    - But training data is forever
- What are the outcomes?
  - Code that won’t compile/run
  - Or crashes.

RAG

- What is RAG (Retrieval Augmented Generation)?
- And how does it help us avoid Hallucinations?

RAG

- What is RAG (Retrieval Augmented Generation)?
  - Allows LLM access to existing documents/data and adds (some of) the information to the prompt context.
- And how does it help us avoid Hallucinations?
  - If we feed the docs for a library into the model as context, much less likely to invent functions based on similar libraries.
  - Examples fed in help generate good code for library use
    - Or API use.
  - Models are trained on the data available back in training, RAG can add current data to adjust output.
- Others that you all came up with?

Agentic AI

- So finally, what is Agentic AI?

Agentic AI

- So finally, what is Agentic AI?
  - AI that can actually ‘do stuff’ rather than exist as a disembodied responder.
  - In current incarnations this is usually a program which uses LLMs and has access to some functions/programs that it can run.
    - Two most common approaches for providing this functionality in 2026:
      - MCP [Model Context Protocol] servers (originally OpenAI invention)
      - Skills (originally Anthropic [Claude/Claude code] innovation)
  - We’ve built some skills as functions – but what kind of programs might we want to run?
    - Lucky volunteer?

Agentic AI

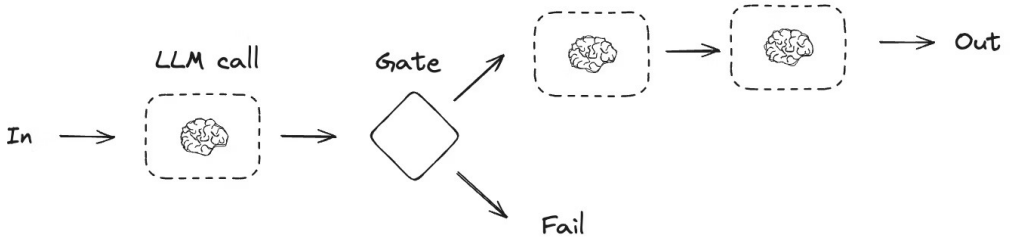
- So finally, what is Agentic AI?
  - AI that can actually ‘do stuff’ rather than exist as a disembodied responder.
  - In current incarnations this is usually a program which uses LLMs and has access to some functions/programs that it can run.
    - Two most common approaches for providing this functionality in 2026:
      - MCP [Model Context Protocol] servers (originally OpenAI invention)
      - Skills (originally Anthropic [Claude/Claude code] innovation)
  - We’ve built some skills as functions – but what kind of programs might we want to run?
    - compiler/interpreter for language
    - Linters
    - Both help keep agentic AI from producing hallucinated buggy code.
      - Because can re-invoke LLM if the compile fails.

Building Agentic AI

- There are several libraries to help build Agentic AI
  - But at the moment the one with the most mindshare is langchain
    - No guarantee that will still be the same in 6-9 months
  - Langchain (https://docs.langchain.com/)
    - Originally written for python and javascript
      - Offers first party support for these languages
    - Ports for other languages available
      - Langchain4j (https://github.com/langchain4j/langchain4j) brings langchain to java (and kotlin)
        - Decent tutoral for Java https://www.baeldung.com/java-langchain-basics
      - Langchaingo (https://github.com/tmc/langchaingo) golang
      - After this it gets a little less well supported
      - https://github.com/tryAGI/LangChain community port for C#
      - https://github.com/Abraxas-365/langchain-rust community port for rust
      - https://pub.dev/packages/langchain community port for dart

LangChain Basic Idea

- Langchain built on langgraph
  - Build Finite State Machine to determine LLM calls and inputs
  - For example in this image from the official docs (https://docs.langchain.com/oss/python/langgraph/workflows-agents)
  -
  -



LLM System

- Langchain can work with a variety of LLM setups
  - OpenAI
  - Gemini
  - Antropic Claude
  - Ollama
- What is Ollama?

LLM System

- Langchain can work with a variety of LLM setups
  - OpenAI
  - Gemini
  - Antropic Claude
  - Ollama

- What is Ollama?
  - A command line system for running LLM/AI models.
    - Runs them locally
  - A competitor of LMstudio
  -

Ollama

- Ollama
  - a command line LMM runner
    - Still need to get a model
    - ollama pull <model name>
    - I tried
    - llama3:8b (doesn't support tools)
    - And
    - qwen3:30b-a3b
    - While prepping for this discussion
    - Also tried
      - Mistral
      - granite4:1b # I had really good luck with this one after my Tuesday demo

- Once downloaded, load an LLM by:
  - ollama run <model name>
- Then check which one is running by
  - ollama ps
- And find out which models are downloaded
  - ollama list
- By Default ollama runs server

  http://localhost:11434

Ollama with agent

- Very simple example – let's look, dissect and discuss

```
from langchain_ollama import ChatOllama

OLLAMA_MODEL = "qwen3:30b-a3b"
OLLAMA_BASE_URL = "http://localhost:11434" # Default Ollama URL

# --- 1. Set up the LLM ---
agent = ChatOllama(model=OLLAMA_MODEL, base_url=OLLAMA_BASE_URL, temperature=0) # Set temperature to
0 for replicable results, to 1 for 'creative' results

# Run the agent
result = agent.invoke(
  [  ("system", "You are a helpful assistant that can answer questions about programming languages."),
     ("human", "Tell me about the haskell programming language.")
    ]
)
print(result)
print("Done!")
```

- Initial reference: https://docs.langchain.com/oss/python/integrations/chat/ollama

Ollama with tools

```python
from typing import List

from langchain.messages import AIMessage
from langchain.tools import tool
from langchain_ollama import ChatOllama


@tool
def validate_user(user_id: int, addresses: List[str]) -> bool:
    """Validate user using historical addresses.

    Args:
        user_id (int): the user ID.
        addresses (List[str]): Previous addresses as a list of strings.
    """
    return True
```

@tool makes a function a tool for AI

The docstring lets the LLM know how to use it

-

```python
llm = ChatOllama(
    model="qwen3:30b-a3b",
    validate_model_on_init=True,
    temperature=0,
).bind_tools([validate_user])

result = llm.invoke(
    "Could you validate user 123? They previously lived at "
    "123 Fake St in Boston MA and 234 Pretend Boulevard in "
    "Houston TX."
)

print(result)
if isinstance(result, AIMessage) and result.tool_calls:
    print(result.tool_calls)
```

From docs: https://docs.langchain.com/oss/python/integrations/chat/ollama scrolls down to ‘tool calling’

Ollama with Baby Tools Part 2

```python
from langchain.agents import create_agent
from langchain_core.messages import HumanMessage
from langchain_ollama import ChatOllama

# Step 1: Define a tool
def get_weather(city: str) -> str:
    """Get weather for a given city."""
    return f"It's always sunny in {city}!"

# Step 2 & 3: Instantiate a model and create the agent
llm = ChatOllama(
    model="qwen3:30b-a3b",
    validate_model_on_init=True,
    temperature=0,
)
agent = create_agent(
    model=llm,
    tools=[get_weather],
    system_prompt="You are a helpful assistant that can check the weather.",
)

# Step 4: Invoke the agent
user_input = {"messages": [HumanMessage(content="What is the weather in San
Francisco?")]}
response = agent.invoke(user_input)

for item in response.get("messages", []):
    print(
        item.content
    )
```

- Here is the typical toy example from the docs
  - Eg: https://docs.langchain.com/oss/python/langchain/quickstart
- But ported to ollama
  - Let’s try it.

Now lets do something more real

- Here is our example Tool function

```
import requests
from langchain.tools import tool


@tool
def get_university_data(Name:str)->list[dict]:
    """
    Returns a list of dictionaries containing data about universities with the given name
    this is a sample dictionary {'state-province': None, 'web_pages': ['http://www.byu.edu/'], 'name': 'Brigham Young University', 'domains': ['byu.edu'], 'country': 'United States', 'alpha_two_code': 'US'}
    Expects part of the university name as an input
    """
    url = f"http://universities.hipolabs.com/search?name={Name}"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        return []
```

- This is a lot like your functions
  - Except that it uses the university API
  - Note the addition of the @tool decorator

Using a real tool – simplest example

```
from AgentSkills import get_university_data
from langchain_ollama.chat_models import ChatOllama
from langchain_core.prompts import PromptTemplate
from langchain.agents import  create_agent


#first built from  https://github.com/saurav-samantray/ollama-llm-rag-tools-example/
OLLAMA_MODEL = "qwen3:30b-a3b"
OLLAMA_BASE_URL = "http://localhost:11434" # Default Ollama URL

print(f"Using Ollama model: {OLLAMA_MODEL}")
print("-" * 30)

# --- 1. Set up the LLM ---
llm = ChatOllama(model=OLLAMA_MODEL, base_url=OLLAMA_BASE_URL,
temperature=0) # Set temperature to 0 for consistent results, 1==creative
print("LLM Initialized.")


tools =[get_university_data]
```

```
agent = create_agent(llm, tools)
print("Agent Created.")
query = "What country is Young located in?"
for event in agent.stream(
      {"messages": [{"role": "user", "content": query}]},
      stream_mode="values",
):
   event["messages"][-1].pretty_print()
```

- Get the example to try it yourself
- https://github.com/jsantore/SimpleAgentToolDemo

Putting it together

- Now lets put this together with the voice stuff from sprint1
  - Here is the essence setup as functions

```
OLLAMA_MODEL = "granite4:1b"
OLLAMA_BASE_URL = "http://localhost:11434" # Default Ollama URL


def process_voice_prompt(agent:CompiledStateGraph, prompt:str)->None:
    for event in agent.stream(
            {"messages": [{"role": "user", "content": prompt}]},
            stream_mode="values",
    ):
        event["messages"][-1].pretty_print()

def setup_agent()-> CompiledStateGraph:
    llm = ChatOllama(model=OLLAMA_MODEL, base_url=OLLAMA_BASE_URL,
temperature=0) # Set temperature to 0
    print("LLM Initialized.")
    tools =[get_university_data]
    agent = create_agent(llm, tools)
    print("Agent Created.")
    return agent


if __name__ == "__main__":
    agent = setup_agent()
    setup_recognizer()
    start_recognizer(agent)
```

Putting it together II

```
voice_stream = None
pyaudioObj = None
recognizer = None

def get_transcript(audio_data) -> str:
    if recognizer.AcceptWaveform(audio_data):
        result = json.loads(recognizer.Result())
        recognized_text = result["text"]
        if "terminate" in recognized_text.lower():
            print("Termination keyword detected. Stopping...")
            clean_up()
        else:
            return recognized_text
    else:
        return ""


def setup_recognizer(from_mic: bool = True):
    global voice_stream, pyaudioObj, recognizer
    model = vosk.Model("vosk-model-en-us-0.22-lgraph")
    recognizer = vosk.KaldiRecognizer(model, 16000)
    pyaudioObj = pyaudio.PyAudio()
    if from_mic:
        voice_stream = pyaudioObj.open(
            format=pyaudio.paInt16,
            channels=1,
            rate=16000,
            input=True,
            frames_per_buffer=8192,
        )
    print("Listening for speech. Say 'Terminate' to stop.")


def start_recognizer(agent:CompiledStateGraph):
    while True:
        data = voice_stream.read(8192)
        text = get_transcript(data)
        if text:
            process_voice_prompt(agent, text)
```

- This is in my voice.py file
  - Calls process_voice_prompt from previous slide
  - Let’s try it out.

Embeddings

- What are ‘embeddings’ in the LLM sense?
  -

Embeddings

- What are ‘embeddings’ in the LLM sense?
  - When we read in a document and run it through a vector database to find meaning similarities
  - Then mark up the document with numeric representations
  - Should allow documents to be searched by ‘meaning’ rather than keyword.