LLMs in Your Science

Ryan Watkins

Graduate School of Education & Human Development

www.LLMinScience.com

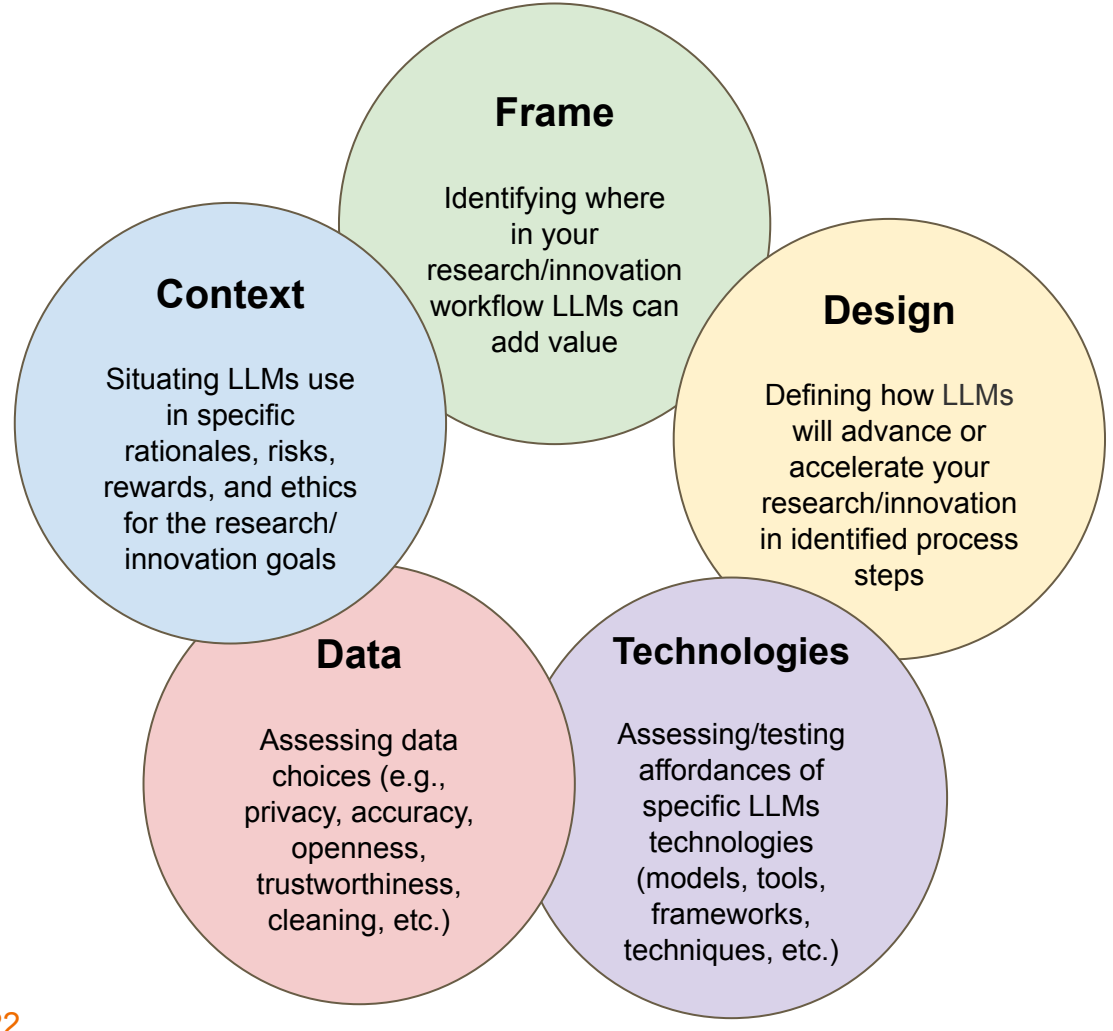
Agenda (on github.com/ryanrwatkins)

DAY ONE	
9:00 AM	Welcome, introductions, & why LLMs
10:00 AM	Getting started with LLMs
10:45 AM	Break
11:00 AM	Prompts & parameters
12:00 PM	Lunch
1:00 PM	Use cases with Zoe #1
2:00 PM	Structured Outputs
2:45 PM	Break
3:00 PM	Hands on
4:00 PM	Evaluating model outputs
4:30 PM	Local models

DAY TWO	
9:00 AM	HPC and On Demand
9:30 AM	Creating agents
10:30 AM	Break
10:45 AM	Hands-on
11:30 AM	Intro to RAG and fine-tuning
12:00 PM	Lunch
1:00 PM	Intro to RAG and fine-tuning, continued
2:00 PM	Trustworthy development processes
2:45 PM	Break
3:00 PM	Hands-on
4:00 PM	Use cases with Zoe #2

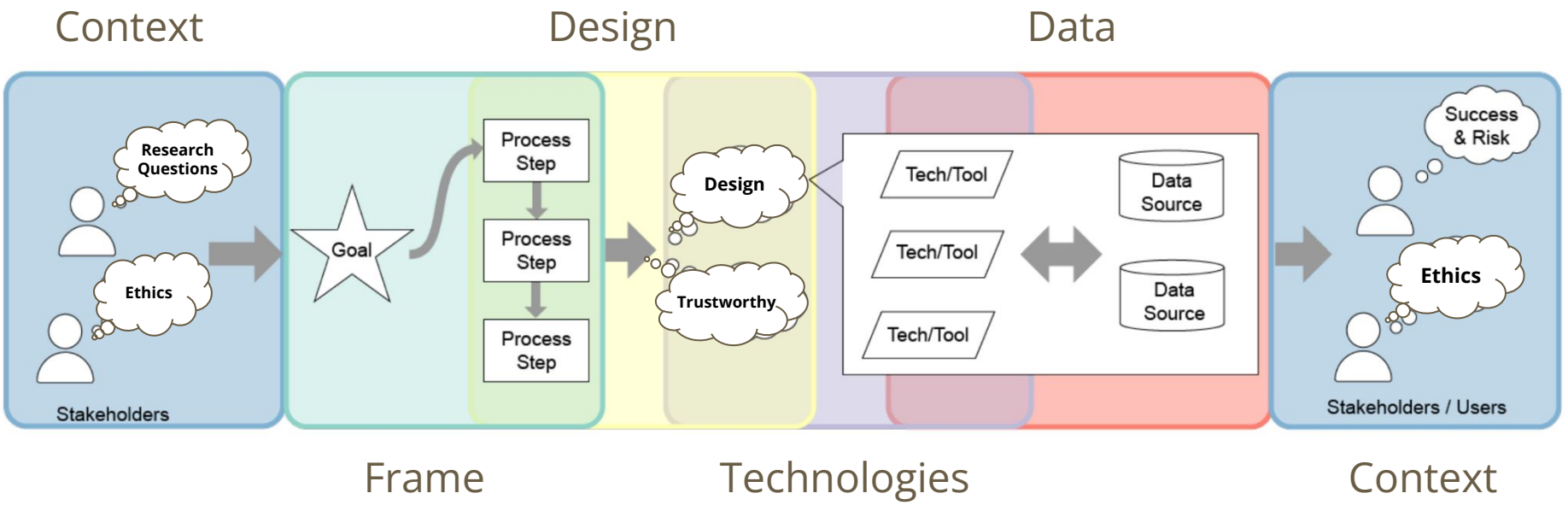
AI Literacy is not enough...

AI Thinking in Research & Innovation



Based on: https://arxiv.org/pdf/2409.12922

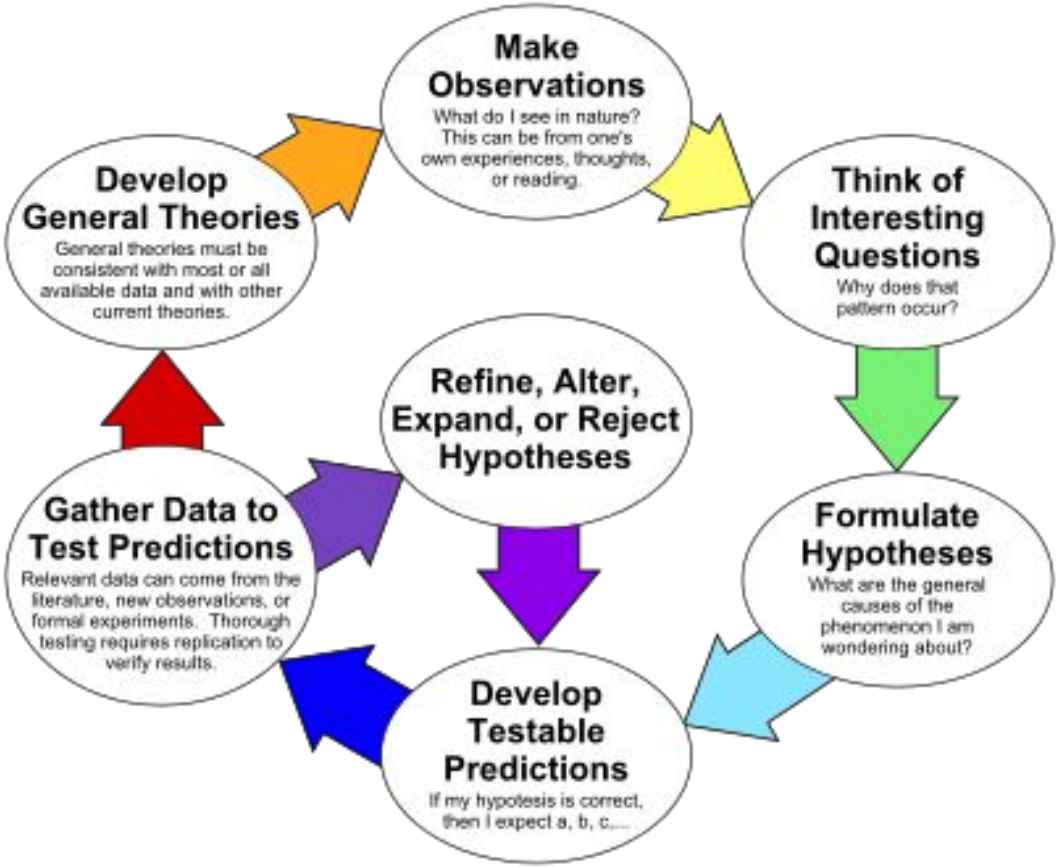
AI Thinking in Research



Based on: https://arxiv.org/pdf/2409.12922

Scientific Method

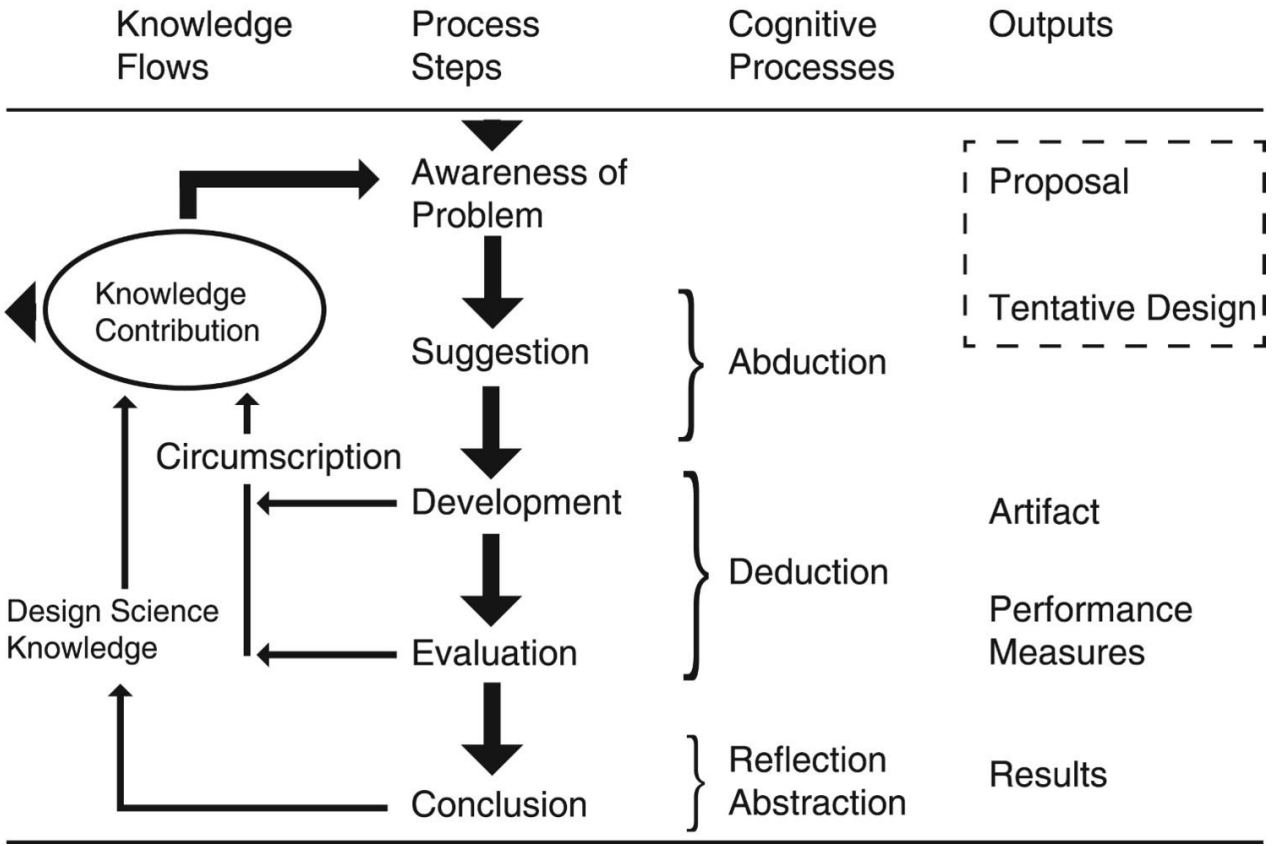
FRAME



https://www.mrgscience.com/scientific-method.html

Design Science Research

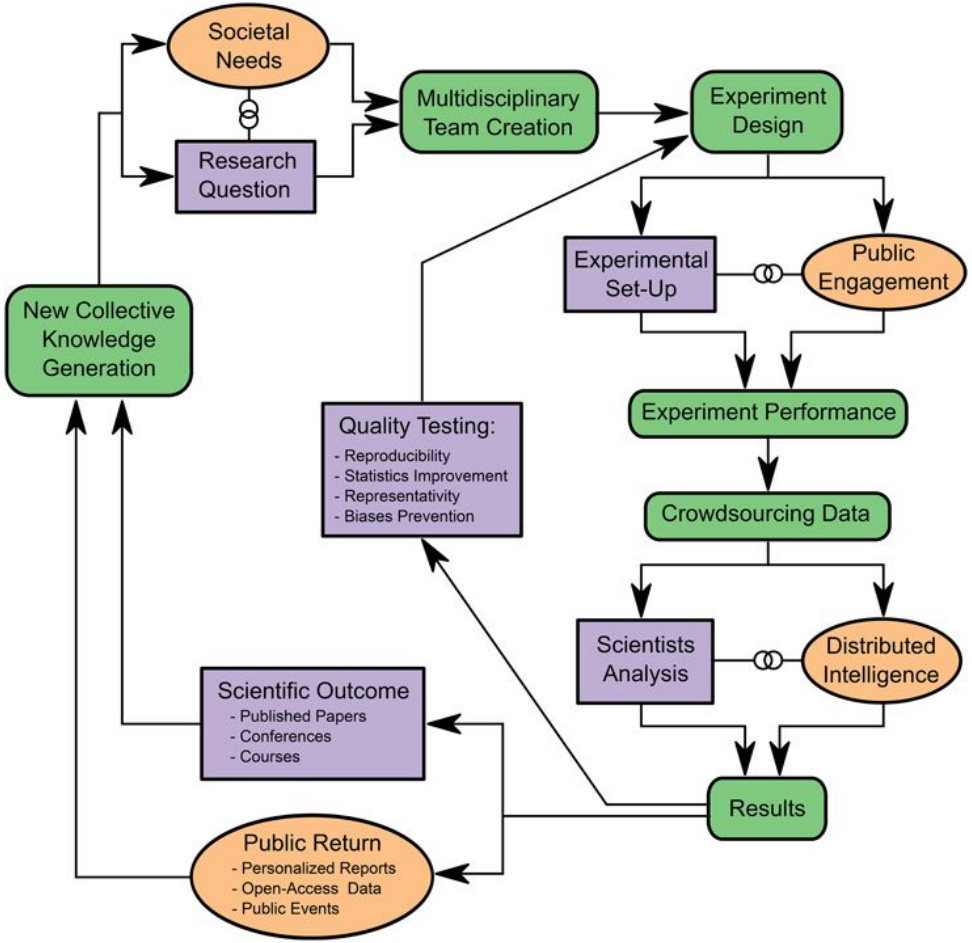
FRAME



https://www.tandfonline.com/doi/full/10.1080/03043797.2018.1498459

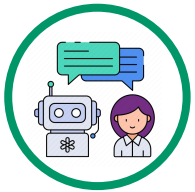
Citizen Science

FRAME



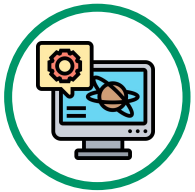
https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2015.00093/full

Continuum of AI in Your Research



Tool Assisted

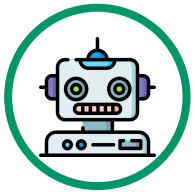
Example:
Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers.
arXiv: 2409.04109



AI Assistant

Example:
Towards an AI co-scientist.
arXiv: 2502.18864

Evaluating AI Models as Scientific Research Assistants
arXiv: 2502.20309v1



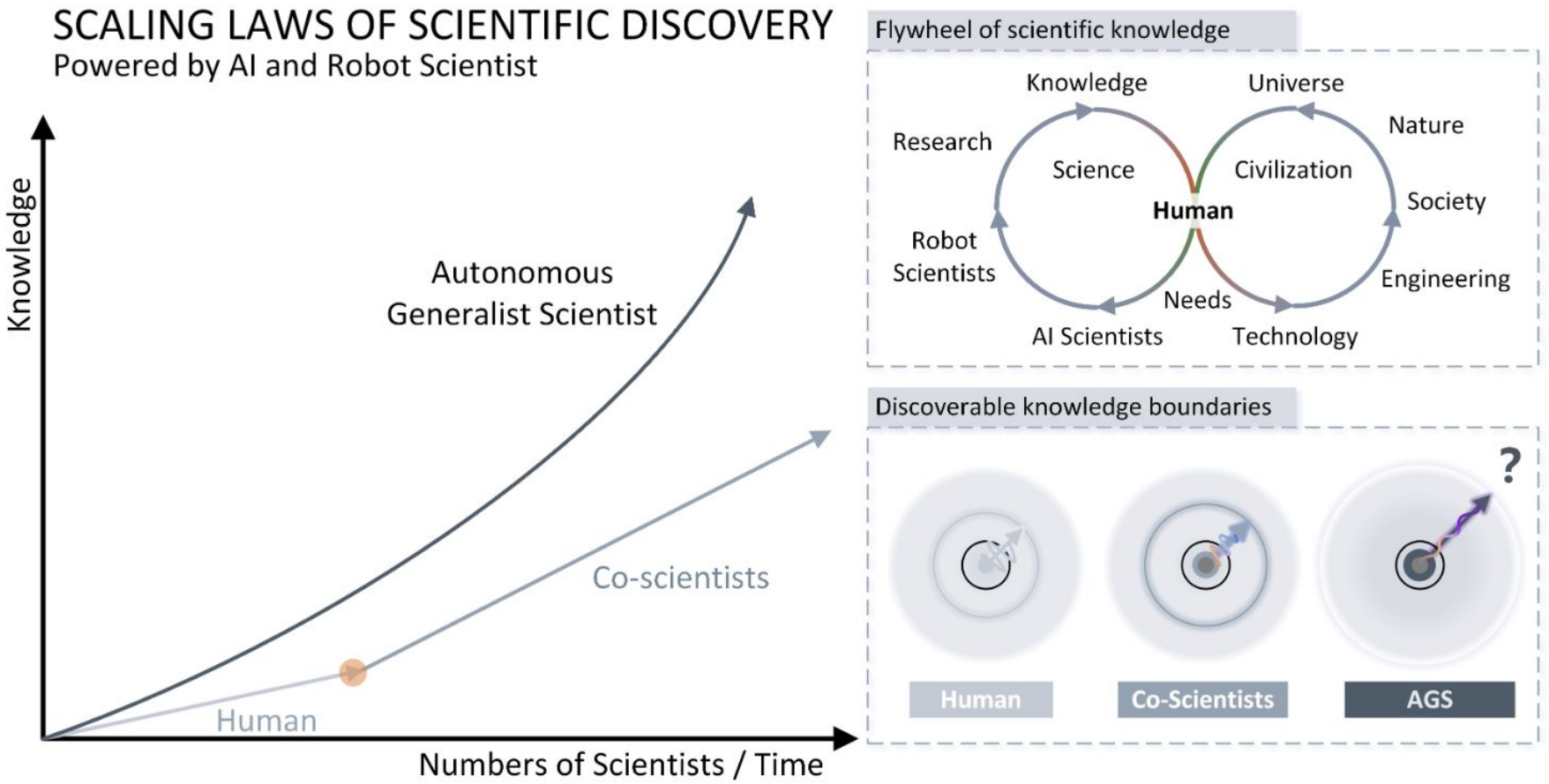
Fully Automated AI Scientist

Examples:
The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery.
arXiv: 2408.06292

SciAgents: Automating scientific discovery
arXiv: 2409.05556

For a 6 level continuum see: https://arxiv.org/pdf/2503.22444v1

Speculating on



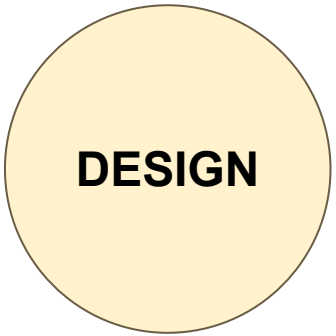
From: https://arxiv.org/pdf/2503.22444v1

Where are you?

Approaching	Exploring	Operational	Systematic	Pioneering
• Aware that AI is creating research opportunities (and challenges) • Following how others are using AI in your field • At the early stages of considering how to navigate these changes	• Using some AI-enabled tools • Piloting AI in small aspects of your research • Examining ethical implications of using AI in your research	• Using AI or AI-enabled tools in multiple aspects of your research workflow • Updating your processes and operating procedures to include AI and AI-enabled tools	• The capacity of AI and AI-enabled tools are integrated into designing new research • Optimizing use of AI in research workflows • Have guidelines for ethical use of AI in your research	• AI is taking your research in new directions that would not be possible without AI • Creating open source AI-enabled tools for research in your field

Based in part on: https://nationalcentreforai.jiscinvolve.org/wp/2024/03/08/our-ai-in-education-maturity-model-an-update-for-2024/

Enhancing Your Research

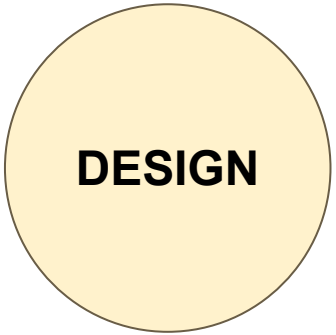


What can LLMs offer to enhance, accelerate, or otherwise improve specific steps in your research workflow?

Examples:

- Modeling
  - **Designing Reliable Experiments With Generative Agent-Based Modeling.** arXiv: 2411.07038v1
- Simulated participants
  - **Generative Agent Simulations of 1,000 People.** arXiv: 2411.10109
- Interactive surveys
  - **Chatbots for Data Collection in Surveys.** arXiv: 2503.08582v1

Starter Ideas for Using LLMs

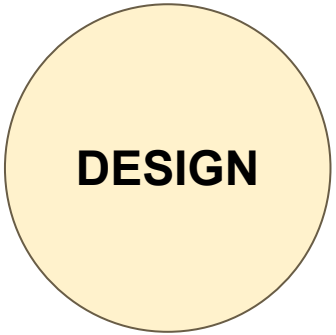


- Ideation
  - Brainstorming
  - Offering counter arguments
  - Evaluating ideas
  - Background Research
- Summarizing articles
  - Translating articles
  - Explaining concepts
  - Connecting article concepts
  - Creating mind maps
- Coding
  - Writing code
  - Optimizing code
  - Explaining code
  - Translating code between languages
  - Debugging code
- Data Collection/Generation
  - LLM interviewer/survey
  - Create synthetic/dummy data
  - Simulating human subjects
  - Creating simulations
  - Modeling
- Data Preparation and Analysis
  - Extracting data from text
  - Reformatting data
  - Classifying and scoring text
  - Data annotation
  - Extracting sentiment
  - Fuzzy joining of data sets
  - Setting up models (in LaTeX)
  - Writing data dictionaries
  - Qualitative coding of data
  - Putting data into Pandas dataframe
- Writing
  - Editing for grammar, vocabulary, clarity
  - Synthesizing wordy text
  - Formatting references
  - Moving to LaTeX or Markdown

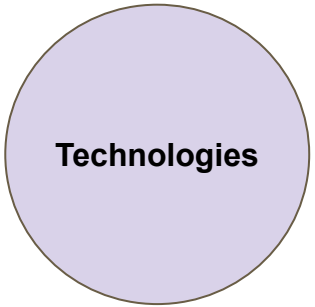
Activity

Step 1:
Identify where specifically in your workflow you might want to apply LLMs? [Use the starter ideas from the prior slide and/or come up with your own.]

Step 2:
Describe 2 or 3 potential applications of LLMs in your research workflow to a person sitting next to you.



Getting to Know the Technologies



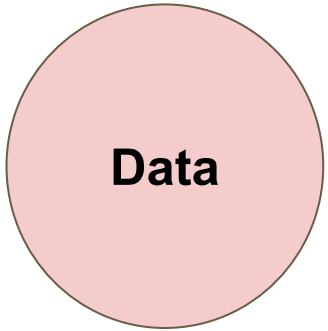
Could you use?

- Local models (e.g., ollama)
- GW High Performance Computing
- HuggingFace Inference playground
- Programmatic control (e.g., Python + ollama)
- Evaluation of outputs (e.g., langfuse)
- Optimizing for multiple LLMs (e.g., llmselector)

Do you require customization?

- Retrieval Augmented Generation (RAG)
- Fine-tuning (e.g., unsloth for LoRA/QLora)
- Agents (e.g., smolagent, crewAI)

Data Considerations



What data might you add via RAG?

- Curating data for RAG
- Adding your bias
- Creating/storing/retrieving embeddings (e.g., chunking, reranking, query transformations)

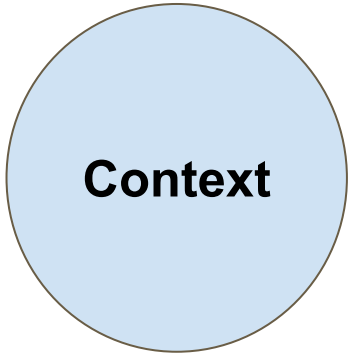
What data might you add via fine-tuning?

- Data (e.g., Kaggle, scraping)
- Structured data formats (e.g., Chatml)
- Cleaning data
- System prompts included for agents?

Considerations

- Privacy
- Ethical use
- Trustworthiness of data
- Trustworthiness of your use

Challenges of LLMs in Research

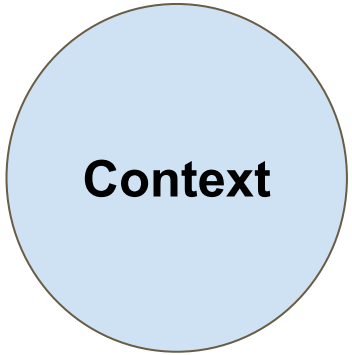


Challenges include:

- Mitigating factual hallucinations
- Ensuring fairness in data handling
- Enforcing data privacy, confidentiality, and regulatory standards
- Enhancing model robustness against adversarial attacks
- Detecting and filtering toxic content
- Complying with legal and ethical standards
- Identifying and handling out-of-distribution inputs and outputs.
- Accurately quantifying and communicating output uncertainty

Based on: Challenges in Guardrailing Large Language Models for Science arXiv: 2411.08181v2

Guardrails



Key Considerations:

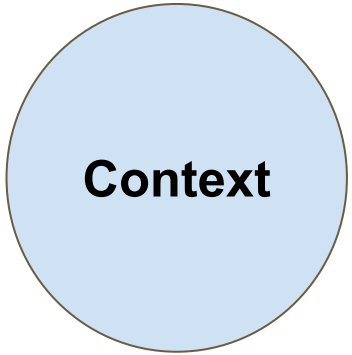
- **Is it legal?**
  - IP, Copyright, Privacy, Compliance with regulations
- **Is it safe?**
  - Harmfulness
    - Maliciousness, profanity, toxicity
  - Secure and Resilient
    - Jailbreak prevention, out of distribution checks
- **Is it ethical?**
  - Societal impact, fairness

Base on: Challenges in Guardrailing Large Language Models for Science arXiv: 2411.08181v2

Guardrails

Key Considerations continued:

- **Is it trustworthy?**
  - Factuality
    - Knowledge contextualization, consistency, time sensitivity, hallucination
  - Explainable
    - Attribution, uncertainty identification, verification, interpretable



Activity

Is it trustworthy?

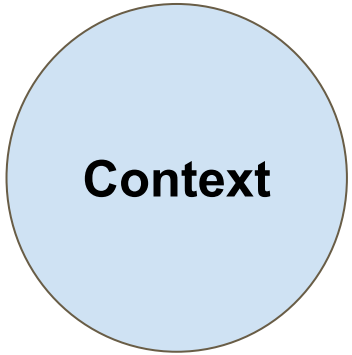
- **Factuality**
  - **Knowledge contextualization, consistency, time sensitivity, hallucination**
- **Explainable**
  - **Attribution, uncertainty identification, verification, interpretable**

Step 1:
Select one of the Trustworthy guardrails (i.e. factuality, explainable), and then one subcategory of it (e.g., consistency, uncertainty identification).

Step 2:
Identify one action you can add to your research workflow (when using an LLM) that can help guardrail your research against this concern. For example, use *GPTScore* to detect hallucinations.

Step 3:
Discuss your action with person sitting next to you.

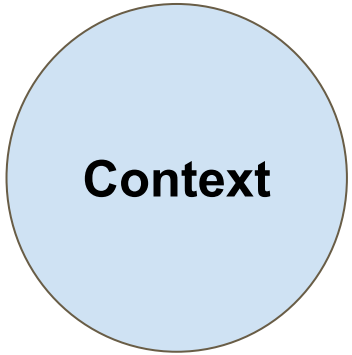
Other Guardrails Considerations



- Open Science
  - Reproducibility, transparency, accountability, preregistration, open data, open code, etc.
  - Example:
    Open Science at the generative AI turn: An exploratory analysis of challenges and opportunities
    Open Access: 10.1162/qss_a_00337

- Illusions of knowledge/learning
  - Example:
    Artificial intelligence and illusions of understanding in scientific research
    Nature: 10.1038/s41586-024-07146-0

Other Guardrails Considerations



- Recognizing and addressing LLM “scheming” or “obfuscated reward hacking”
  - Examples:
    Frontier Models are Capable of In-context Scheming
    arXiv: 2412.04984

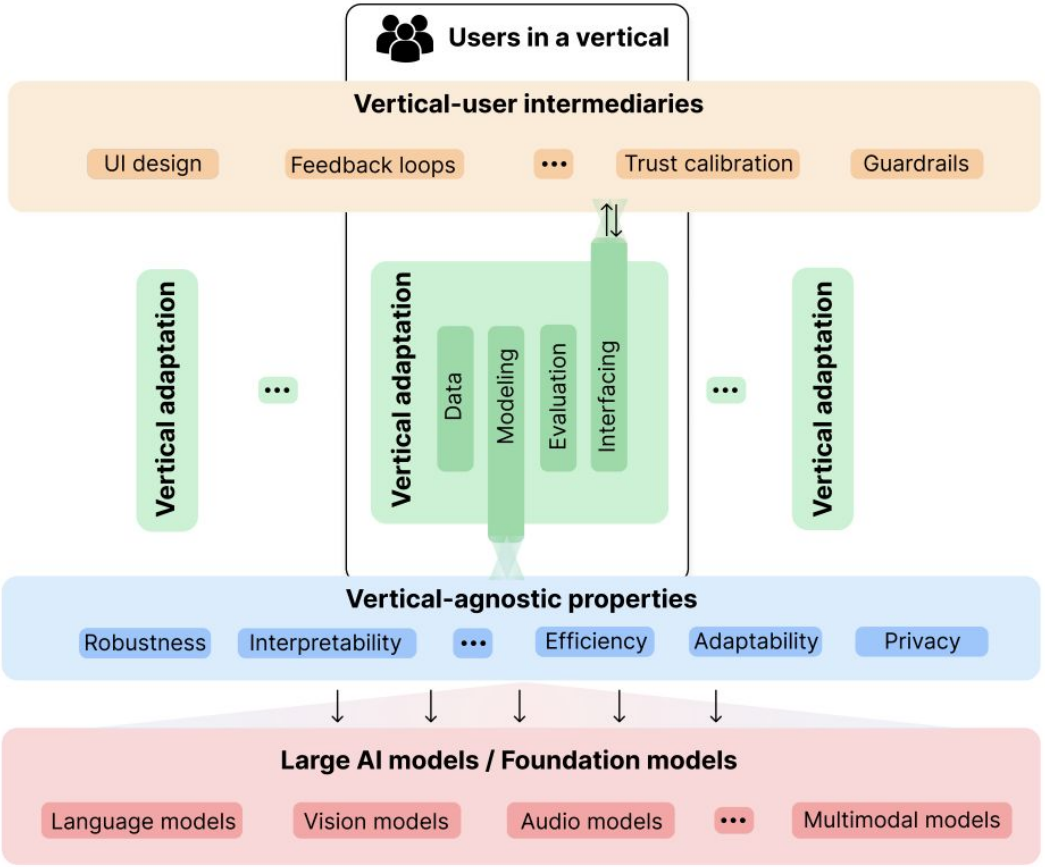
    Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation
    arXiv: 2503.11926

- Customizing changes models
  - Examples:
    Fundamental Safety-Capability Trade-offs in Fine-tuning Large Language Models
    arXiv: 2503.20807

    Fine-Tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!
    arXiv: 2310.03693

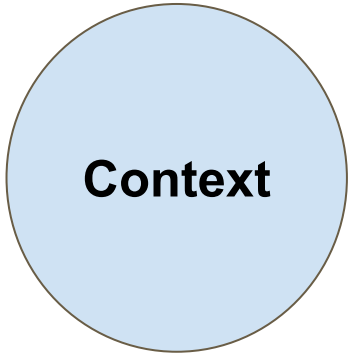
Vertical Framework

Context



From: arXiv: 2504.02793

Vertical Framework Example



Vertical-User Intermediaries

Trust calibration : How to best communicate models' capabilities and limitations?
Feedback loops : Capturing participant feedback for iterative refinement of models
Dynamic interfaces : Design interfaces that engage participants with higher and lower need for cognitive engagement

Vertical Adaptation in Research

Data : Curating data that supports better domain specific outputs
Modeling : Enabling efficient participant data collection
Evaluation : Evaluating output quality by discipline standards
Interfacing : Researcher AI collaboration for error correction

Vertical-Agnostic Properties

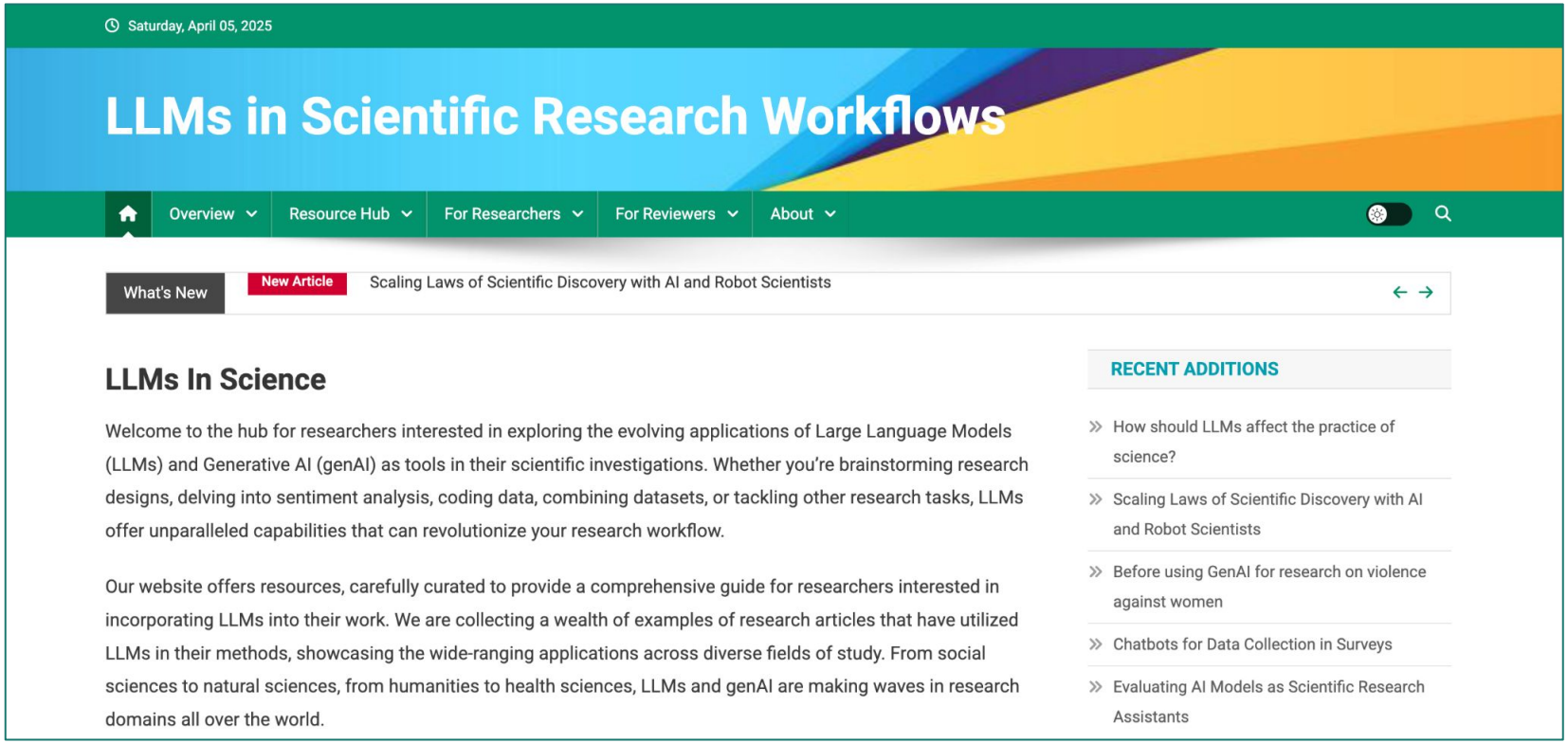
Robustness : Are models robust to realistic variations in participant input?
Privacy : Do models carefully handle sensitive data and private information?
Interpretability : Can models provide interpretable predictions?

Large Language Models

How can modalities beyond language (visual, audio, sensor data) be reliably processed?
Is "reasoning" available to augment analysis?

Based on: arXiv: 2504.02793

LLMinScience.com




hands
on

API Key

- We will use OpenAI API for our workshop
  - https://platform.openai.com/
-

Using LLMs via API with Python

- We will use OpenAI API for our workshop
  - https://platform.openai.com/
- We will use GWU's Jupyter Lab
  - Could use Google CoLab (colab.research.google.com)
- We will start with
  - Storing API keys as secrets
  - API calls
  - Prompting techniques
  - Parameters

Hands On with Jupyter

1. Go to: https://go.gwu.edu/jupyter
2. Protect your API key:
   a. In cell: pip install python-dotenv
   b. Open Terminal and enter: echo "OPENAI_API_KEY=sk-xxxxxxx" > .env
   c. In cell:

```
from dotenv import load_dotenv
import os
load_dotenv(override=True)
openai_api_key = os.getenv("OPENAI_API_KEY")
print(openai_api_key)
```

   d. To edit API key in Jupyter Terminal: nano .env
   e. Note: If you link to Github, then add *.env* to *gitignore*

Hands on with Jupyter + API

3. In cell: pip install openai
4. In cell:

```
from openai import OpenAI
client = OpenAI(api_key=openai_api_key)
response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
        {"role": "developer", "content": "You are a helpful research assistant."},
        {"role": "user", "content": "What is p-hacking?"}
])
print(response.choices[0].message.content)
```

Prompting Strategies

- Prompt elements:
  - Developer, user
  - Context window
- Prompt from Markdown file
- Prompts with examples
  - Zero-Shot and Single-Shot
  - Prompt with Few-Shots
- Prompt with Chain-of-Thoughts (for non-reasoning models)
- Self-consistency Prompting
  - Generate multiple CoT outputs, then choose the most frequent final answer.
- Prompting reasoning models
  - Only give high level guidance

Hands On with CoT Prompting

- Update the user content to a CoT prompt
- Example:
  cot_prompt = (
  "A study finds that students who use laptops in class score higher on exams.

  The authors conclude that laptop use causes better performance.

  Is this conclusion justified?

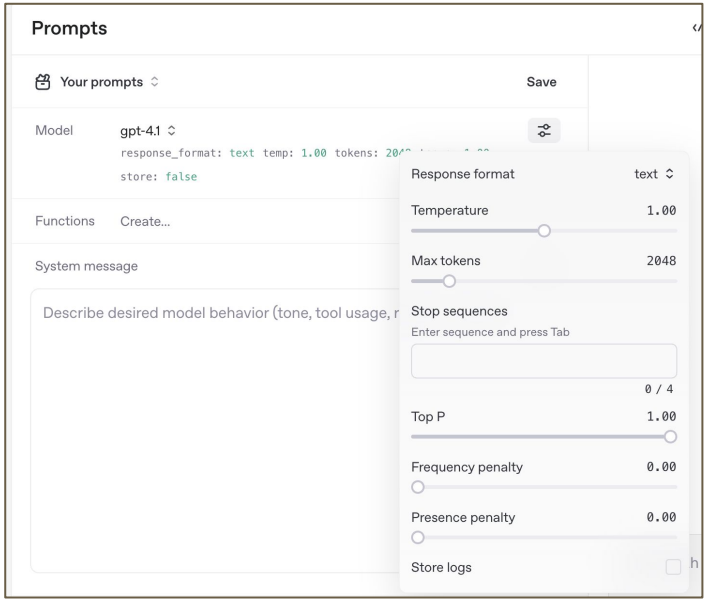
  Let's think step by step.")

OpenAI Model Parameters

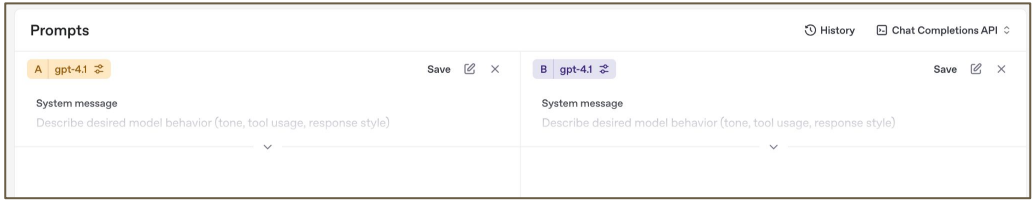
- **temperature:** Controls randomness in the output. Lower values (e.g., 0.2) make responses more deterministic; higher values (e.g., 0.8) increase creativity.
- **top_p:** An alternative to temperature, where the model limits itself to only the top X% most likely words, making responses more predictable and focused..
- **max_tokens:** Limits the maximum number of tokens in the generated response. This helps control the length of the output.
- **n:** Specifies the number of completions to generate for each prompt. Useful for obtaining multiple response options.
- **stop:** Defines one or more sequences where the API will stop generating further tokens. This can be a string or an array of strings.
- **presence_penalty:** Adjusts the likelihood of the model introducing new topics. A higher value encourages the model to discuss new subjects.
- **frequency_penalty:** Penalizes repeated tokens, reducing the chance of repetitive responses.
- **logit_bias:** Allows you to adjust the likelihood of specific tokens appearing in the output by modifying their logits.

https://platform.openai.com/playground/

- Testing prompts



- Model comparison



Jupyter + API + Parameters

3. Update cell:

```
from openai import OpenAI
client = OpenAI(api_key=openai_api_key)
response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
        {"role": "developer", "content": "You are a helpful research assistant."},
        {"role": "user", "content": "What is p-hacking?"}
        ],
        temperature=0.3,
        max_tokens=150 )
print(response.choices[0].message["content"])
```

Note: If you want to attach images, use LLM package: https://pypi.org/project/llm/

Case Example #1

Case Study #1:The Dream: Scaling Qualitative Coding

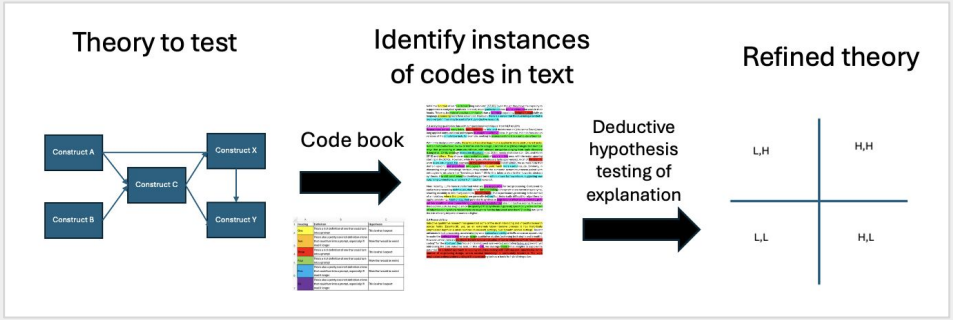
Why is this interesting? Break the depth vs. breadth tradeoff!

Inductive research is the only way to dig deep into previously uncharacterized phenomena. When phenomenon of interest is socio-technical, most of the relevant data is unstructured (interview transcripts, photos of environment, primary documents) and qualitative coding is the way it is analyzed. This is extremely time intensive, which forces a narrow focus on one or few settings, raising generalizability concerns.

IF LLMs can do qualitative coding, we *could* have inductive depth at scale.

Disambiguation: Annotation vs. Qual coding

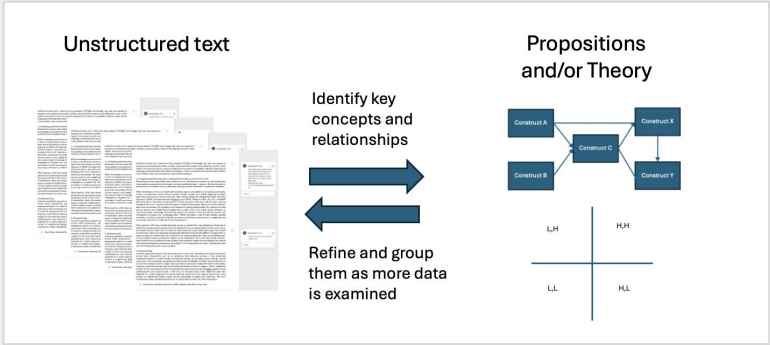
Annotation



Codes can be semi-subjective (e.g., “hate speech”) but process is still deductive (i.e., is “hate speech” as defined, in *this* text). Major progress with better language models.

Vs.

Qual coding



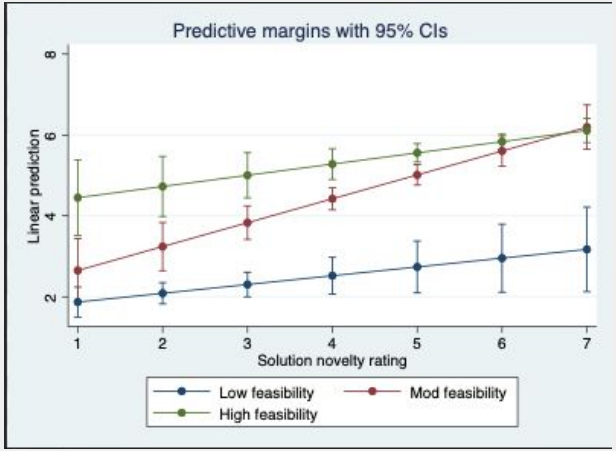
Inherently inductive (i.e., what concepts in this text explain why X is happening) and open ended in terms of source material (e.g., looking across multiple corpuses and outside theory to interpret). Major skepticism about if LLMs could ever help.

GW TAI TRUSTWORTHY AI INITIATIVE

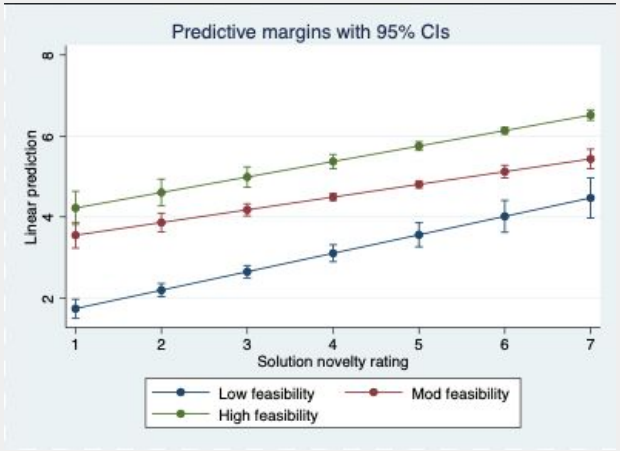
My Problem: Weird Result, too much qual data for me

Data: We'd crowdsourced the evaluation of 100+ space robotics designs to understand how we should set up evaluation panels to balance feasibility and novelty.

Domain & Technology Experts



Everyone Else



Why (only with experts) do low feasibility solutions ignore novelty in their quality rating and value it a lot for mid feasibility?

Answer in their justification (of novelty, feasibility and quality)?

Why Novel: The mechanism used (scissors) is not new, but the way it is used which allows the retraction of the mechanism and the location of the claw throughout the required grip range gives the system some novelty (although it is not very efficient since when the astronaut wants to separate the mechanism from the handrail and exerts a force, he must still wait for the control system to detect said force through the sensors and order the opening of the claw.) x 10,000

Discussion:

Data: We have open text comments justifying novelty, quality and feasibility for each evaluation (3 x 3500)

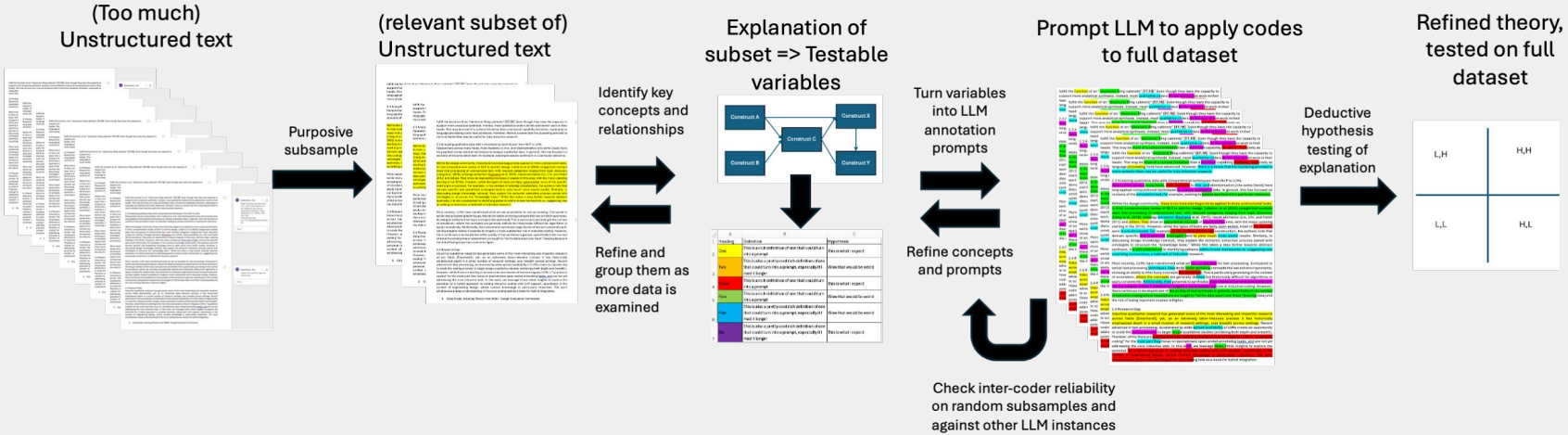
Given what we've been discussion:

1. What should we try?
2. What do you think would work, not work?

What we tried

1. NLP, LLM-driven cluster extraction
2. Human novelty-type code-book, LLM annotation at scale
3. Human coding on purposively selected subset
4. Human abduction, human code book, LLM annotation at scale, quant analysis

Where we landed



Why (only with experts) do low feasibility solutions ignore novelty in their quality rating and value it a lot for mid feasibility?

- If novelty raises feasibility concerns, it's only bad news for the design (so + Novelty doesn't contribute to quality score).
- Experts much more likely to appreciate subtle novelty that enhances feasibility through e.g., a clever configuration. (A type of novelty that many non-experts didn't recognize).


hands
on

Structured Outputs

- JSON
  - Python Dictionary

```
{
  "name": "Alex Johnson",
  "age": 21,
  "major": "Psychology",
  "enrolled": true,
  "courses": ["Research Methods", "Cognitive Science",
"Statistics"],
  "contact": {
    "email": "alex.johnson@example.edu",
    "phone": "555-123-4567"
  }
}
```

Structured Outputs

{JSON}
JavaScript Object Notation

- Structured generation through prompts
  - “Provide outputs only in JSON”
- Structured generation through schema
  - https://cookbook.openai.com/examples/structured_outputs_intro
- Runnable Retries with LangChain
  - LangChain framework
- More information:
  - https://learn.deeplearning.ai/courses/getting-structured-llm-output
  - **arxiv :2505.04016**

Hands On Structured Outputs via Prompt

- Write a prompt asking ChatGPT to read a piece of text of your choosing (e.g., copy-paste from a wikipedia page) and extract the key information into a JSON file
- Revise your prompt 3 times, compare your results each of the prompts

Note: If you want to use Schemas, LLM package: https://pypi.org/project/llm/

Hands On Structured Outputs via Parameter

- Write a prompt asking ChatGPT to read a piece of text of your choosing (e.g., copy-paste from a wikipedia page) and extract the key information into a JSON file
- This time use the parameter:
  - **response_format={ "type": "json_object" },**
  - **Must use “json” somewhere in your developer prompt**

Note: If you want to use Schemas, LLM package: https://pypi.org/project/llm/

A Note on Frameworks

Frameworks are toolkits that help you:

- Connect language models to other tools (e.g., search, databases, Python code)
- Structure complex workflows like chains, memory, tools, and agents
- Manage prompts, retries, and output parsing more reliably

Why You Might Use One Later:

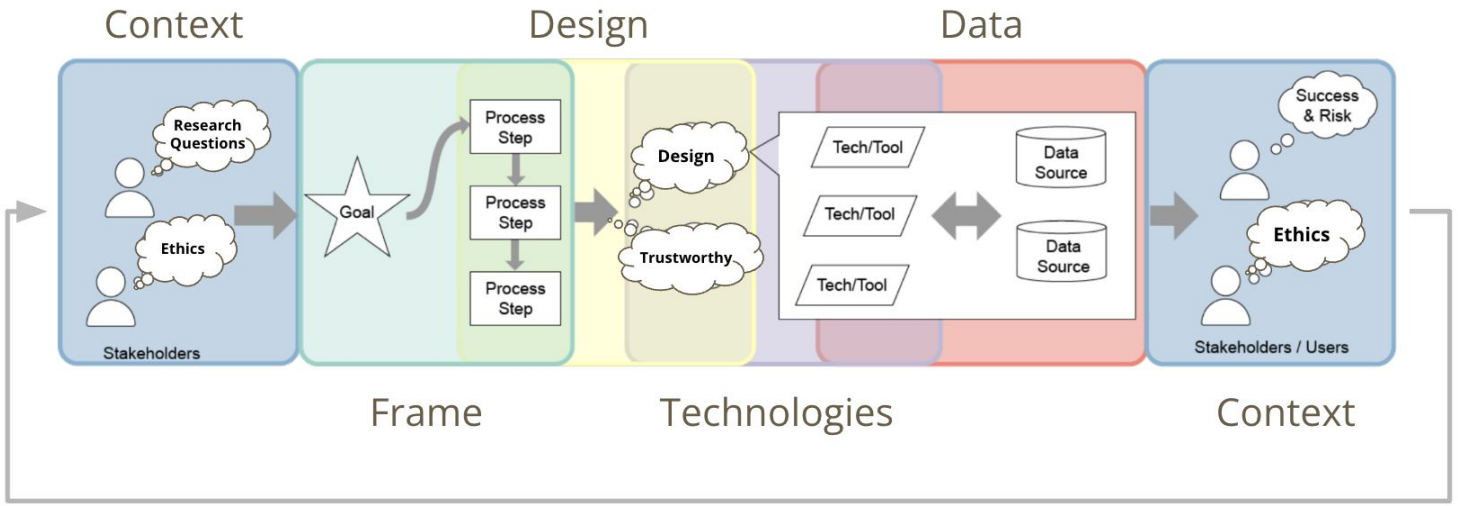
- You want your LLM to call a web search or custom Python function
- You want to split your task into multiple steps or prompts
- You want to add memory, conversation history, or use multiple models in the pipeline

Examples of Frameworks:

- LangChain
- LlamaIndex
- CrewAI / AutoGen

LLM in YOUR Research

- Choose some aspect of your research (Frame)
- Identify a specific LLM use case (Design)
- Write a Python script (Technology)



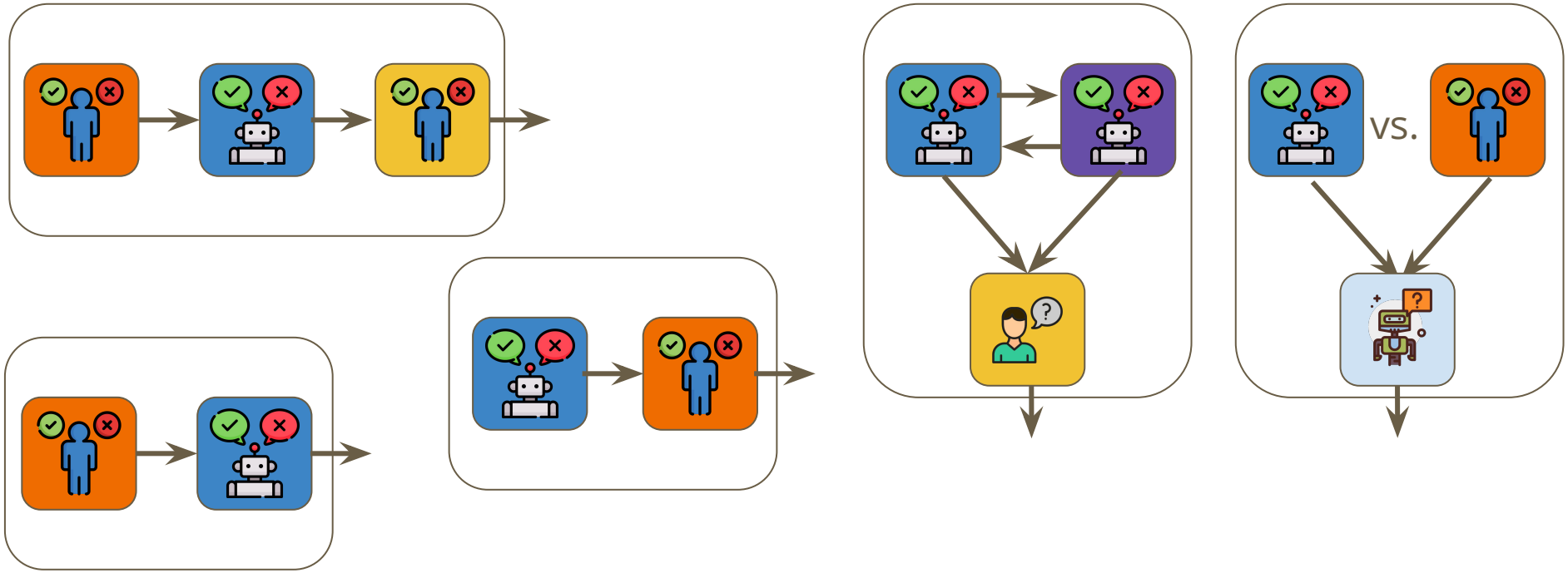
Evaluating LLM Outputs

- Evaluating AI models as scientific Research Assistants (EAIRA)
  - **arXiv: 2502.20309**

		Techniques			
Tasks in the Scientific **Research Workflow**	Scientific Research Assistant Expected Skills	**MCQ Benchmarks**	**Open Response Benchmarks**	**Lab Style Experiments (Focus on human detailed analysis in controled environment)**	**Field Style Experiments (Focus on automated scalable analysis in natural environment)**
Research question/Problem formulation	Problem understanding, Knowledge	X	X	X	X
Conduct initial preliminary experiments/data collection	Experiment Design/ Data retrieval			X	X
Literature search	Knowledge, Retrieval, Literature Understanding			X	X
Hypothesis generation	Propose relevant research directions, existing/analytical solutions	X	X	X	X
Hypothesis testing	Propose pertinent experiments, observations, simulations		X	X	X
Test results analysis	Propose/Use relevant data analysis techniques	X		X	X
Report writing	Generate a research report describing the scientific problem, the related literature, the proposed solution, its evaluation and the lessons learned and conclusions)			X	X

Evaluating LLM Outputs

- You have to monitor in order to improve performance
- Sample combinations of Human + LLM evaluation



Langfuse

- Create free account at: www.langfuse.com
- Start a new project
- Copy Secret_Key, Public_Key, and Host to your .env file where OpenAI API key.
  - In Terminal:
    - echo "LANGFUSE_SECRET=sk-xxxxxxx" > .env
    - echo "LANGFUSE_PUBLIC=sk-xxxxxxx" > .env
    - LANGFUSE_HOST=https://us.cloud.langfuse.com
    - Or nano .env

Langfuse

- Add library: !pip install langfuse
- In cell after from dotenv import load_dotenv cell:

```
from langfuse import Langfuse
import os
langfuse = Langfuse(
secret_key=os.getenv("LANGFUSE_SECRET"),
public_key=os.getenv("LANGFUSE_PUBLIC"),
host=os.getenv("LANGFUSE_HOST")
)
```

Langfuse

- In cell with:

```
from openai import OpenAI
client = OpenAI(api_key=openai_api_key)
prompt = "What is p-hacking?"
response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
        {"role": "developer", "content": "You are a helpful research assistant."},
        {"role": "user", "content": prompt}
])
output = response.choices[0].message.content
print(output)

langfuse.trace(
        name="llm_research_demo",
        input={"prompt": prompt},
        output={"answer": output}
        )
```

Demo: Storing and Evaluating in Google Sheet

- Step 1: Connect to Google Sheet:

```
import json
import pygsheets
import os
json_creds = os.environ.get("Google_cred")
service_account_info = json.loads(json_creds)

# Since Google credentials is in os.env, it expects file so we create temp file
def google_creds_as_file():
    temp = tempfile.NamedTemporaryFile()
    temp.write(json.dumps(service_account_info).encode())
    temp.flush()
    return temp

creds_file = google_creds_as_file()
gc = pygsheets.authorize(service_account_file=creds_file.name)
```

Google Sheets

Demo: Storing and Evaluating in Google Sheet

- Then copy of query and response:

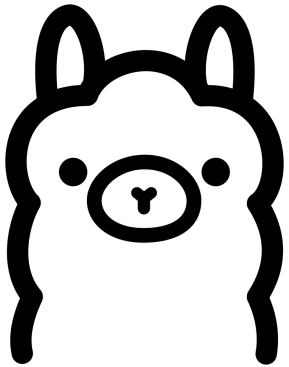
```
sheet_title = 'LLM_evaluator'
sh = gc.open(sheet_title)
worksheet = sh.sheet1
def append_to_google_sheet(query, answer):
        new_row = [query, answer]
        worksheet.append_table(
        new_row, start="A1", end=None, dimension="ROWS")
```

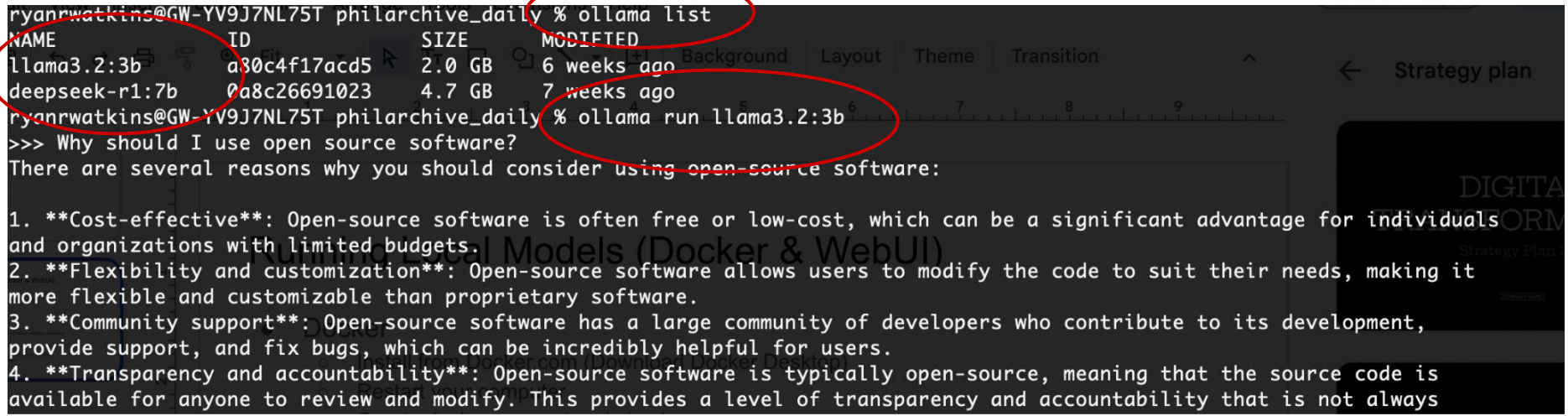
Google Sheets


hands
on

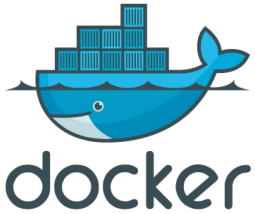
Running Local Models (Ollama)

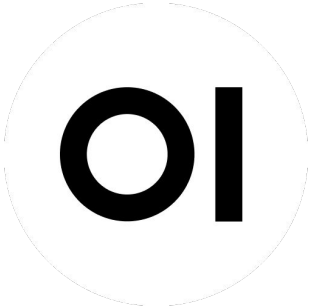
- Ollama
  - Install from Ollama.com
  - Check: http://localhost:11434
  - Open terminal (or Command Prompt)
    - ollama pull llama3.2:3b
    - ollama list
    - ollama run llama3.2:3b
  - Optional:
    - Create an ollama account



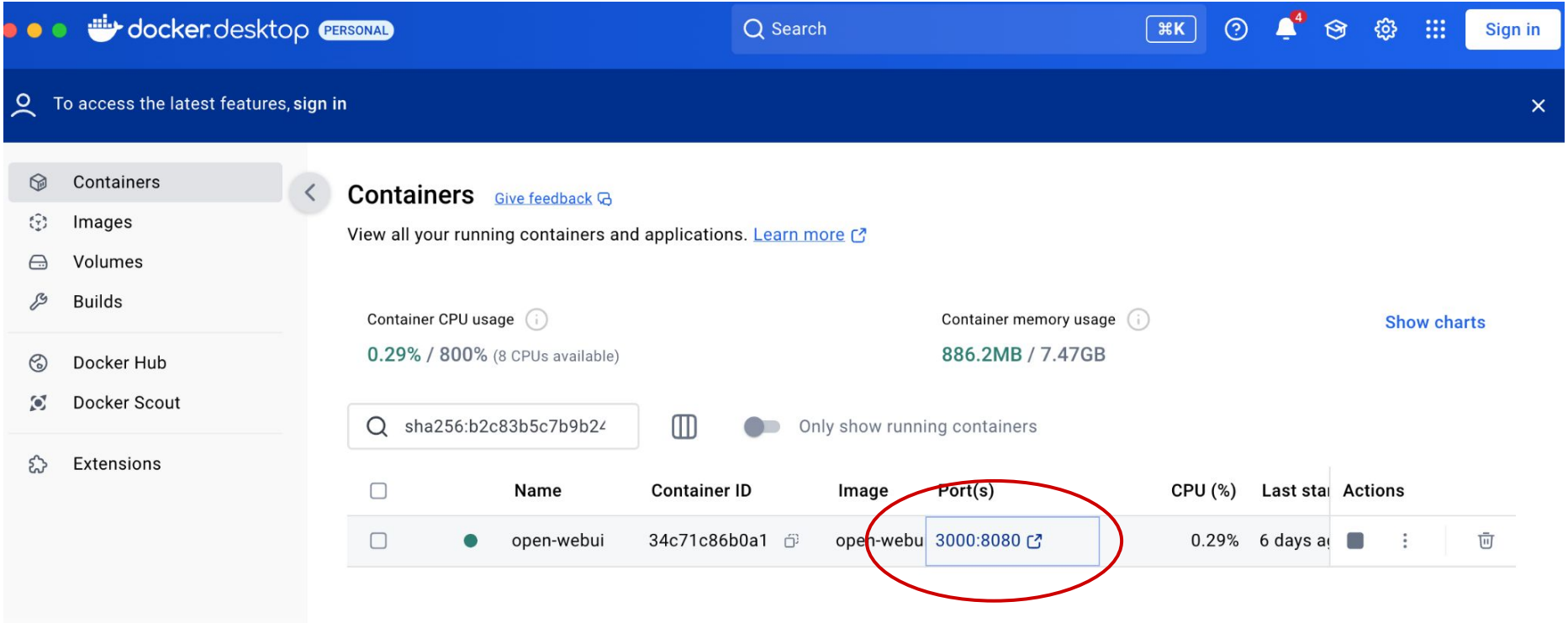

ryanrwatkins@GW-YV9J7NL75T philarchive_daily % ollama list
NAME ID SIZE MODIFIED
llama3.2:3b a80c4f17acd5 2.0 GB 6 weeks ago
deepseek-r1:7b 0a8c26691023 4.7 GB 7 weeks ago
ryanrwatkins@GW-YV9J7NL75T philarchive_daily % ollama run llama3.2:3b
>>> Why should I use open source software?
There are several reasons why you should consider using open-source software:
1. **Cost-effective**: Open-source software is often free or low-cost, which can be a significant advantage for individuals and organizations with limited budgets.
2. **Flexibility and customization**: Open-source software allows users to modify the code to suit their needs, making it more flexible and customizable than proprietary software.
3. **Community support**: Open-source software has a large community of developers who contribute to its development, provide support, and fix bugs, which can be incredibly helpful for users.
4. **Transparency and accountability**: Open-source software is typically open-source, meaning that the source code is available for anyone to review and modify. This provides a level of transparency and accountability that is not always

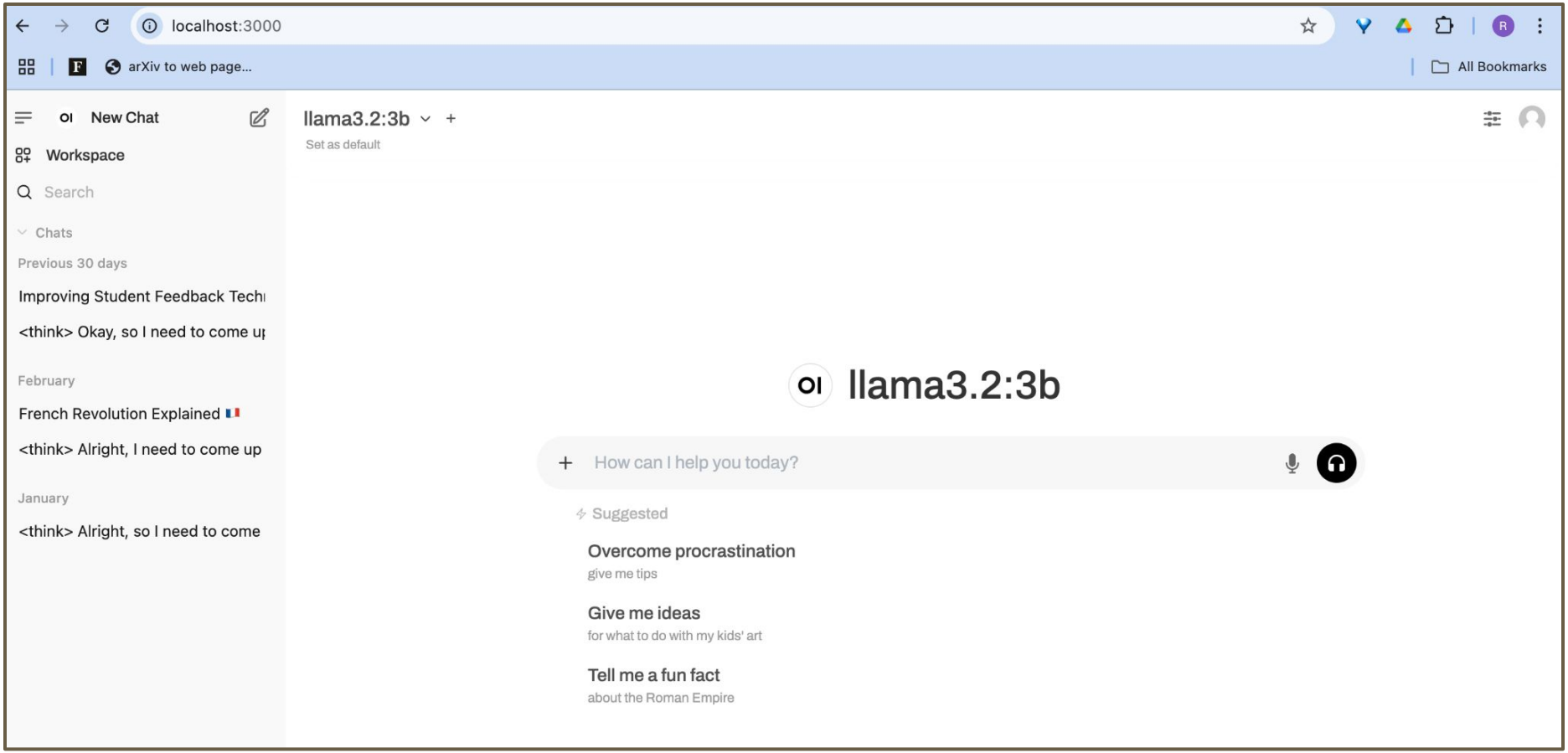
Running Local Models (Docker & WebUI)





- Docker
  - Install from Docker.com (Download Docker Desktop)
  - Restart your computer
  - Create docker account and sign in
  - Check that Docker is running
- WebUI
  - https://github.com/open-webui/open-webui
  - Scroll down to: **If Ollama is on your computer, use this command:**
  - Copy the command
  - Open terminal (or Command Prompt)
  - Paste and execute the command
  - Open Docker, find WebUI docker and click on Port 3000:8080
  - Sign up for WebUI account (for updates)


docker desktop PERSONAL
Search
⌘K
Sign in
To access the latest features, sign in
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
0.29% / 800% (8 CPUs available)
Container memory usage
886.2MB / 7.47GB
Show charts
sha256:b2c83b5c7b9b24
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Actions
open-webui
34c71c86b0a1
open-webu
3000:8080
0.29%
6 days a


localhost:3000
arXiv to web page...
All Bookmarks
New Chat
Workspace
Search
Chats
Previous 30 days
Improving Student Feedback Techı
<think> Okay, so I need to come uı
February
French Revolution Explained 🇫🇷
<think> Alright, I need to come up
January
<think> Alright, so I need to come
llama3.2:3b
Set as default
llama3.2:3b
How can I help you today?
Suggested
Overcome procrastination
give me tips
Give me ideas
for what to do with my kids' art
Tell me a fun fact
about the Roman Empire

Local Ollama with Python

- Open VS Code (or whatever IDE you prefer)
  - If you don’t have one, download VS Code
- Create a folder for the demo
- Create a virtual environment: python -m venv venv_name
- Enter it: source venv_name/bin/activate

Local Ollama with Python

```
import requests
import json
res = requests.post(
    "http://localhost:11434/api/generate",
    json={
            "model": "llama3.2:3b",
            "prompt": "Explain what p-hacking is in research.",
            "stream": False
    })

print(res.text)
```

Local Ollama with Python

```
import requests
import json
res = requests.post(
    "http://localhost:11434/api/generate",
    json={
            "model": "llama3.2:3b",
            "prompt": "Explain what p-hacking is in research.",
            "stream": False
    })
data = res.json()
print(data["response"])
```

HPC @ GWU

- Pegasus vs. Cerberus
  - Different use cases
- ssh via command line vs. Online onDemand
- We will use Online onDemand and VS Code

HPC - Setting up Ollama

- Run files in
  - “Scratch” on Pegasus
- Store files in Home
- Run OLLAMA on HPC

```
ml python3/3.12.9
ml ollama/0.6.8
ollama serve &
ollama list
ollama run llama3.1:8b
```

HPC - Setting up Python

- Install and activate a virtual environment

  python3 -m venv ollama_venv

  source ollama_venv/bin/activate

  pip install requests

- Set VS Code to your venv
  - In VS Code > Command Palette > Python Select Interpreter > [your venv]

Local Ollama with Python on HPC

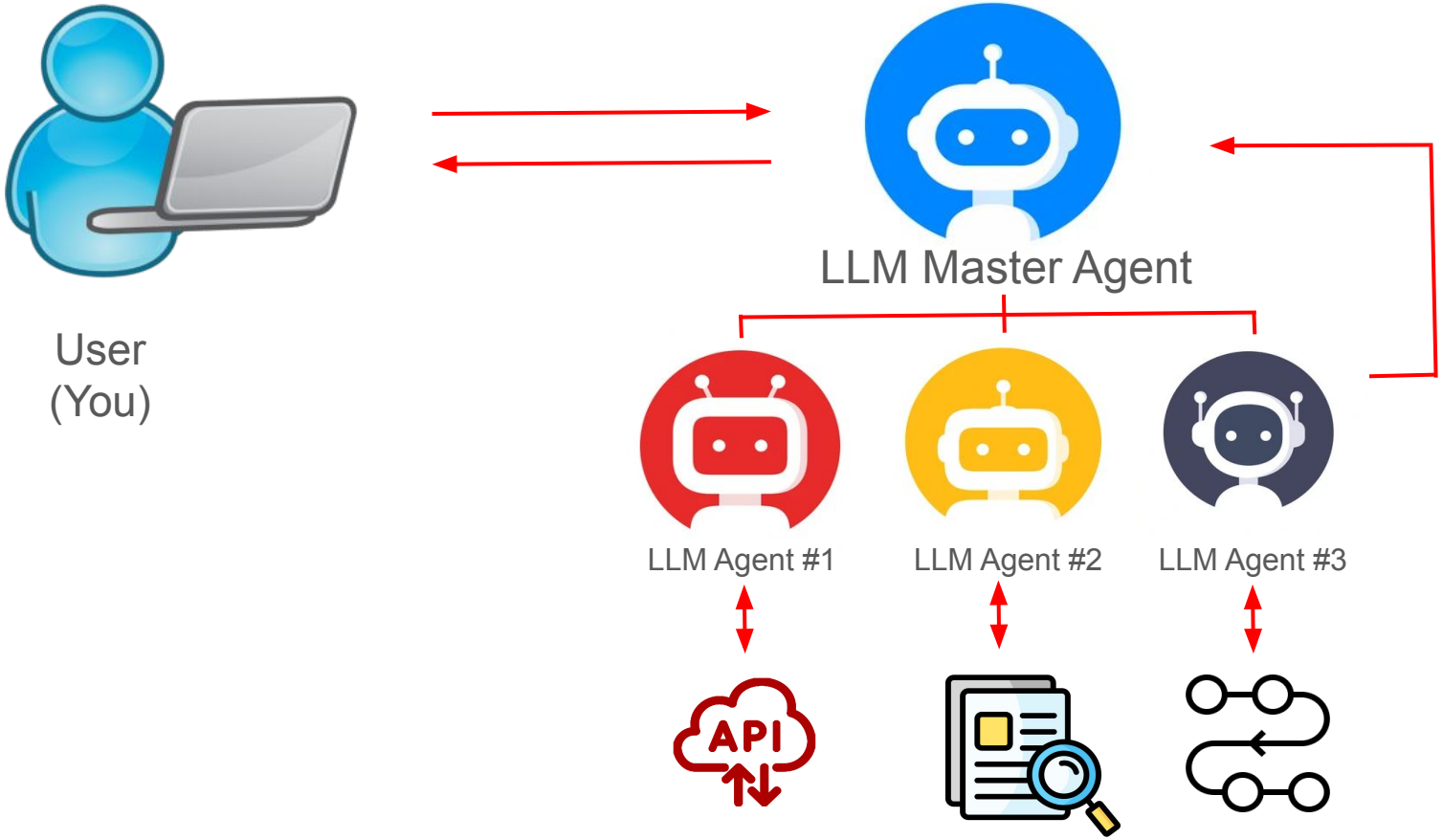
```
import requests
import json
res = requests.post(
    "http://localhost:11434/api/generate",
    json={
            "model": "llama3.1:8b",
            "prompt": "Explain what p-hacking is in research.",
            "stream": False
    })

print(res.text)
```

In VS Code > Command Palette > Python Select Interpreter > [your venv]

Comparing Models

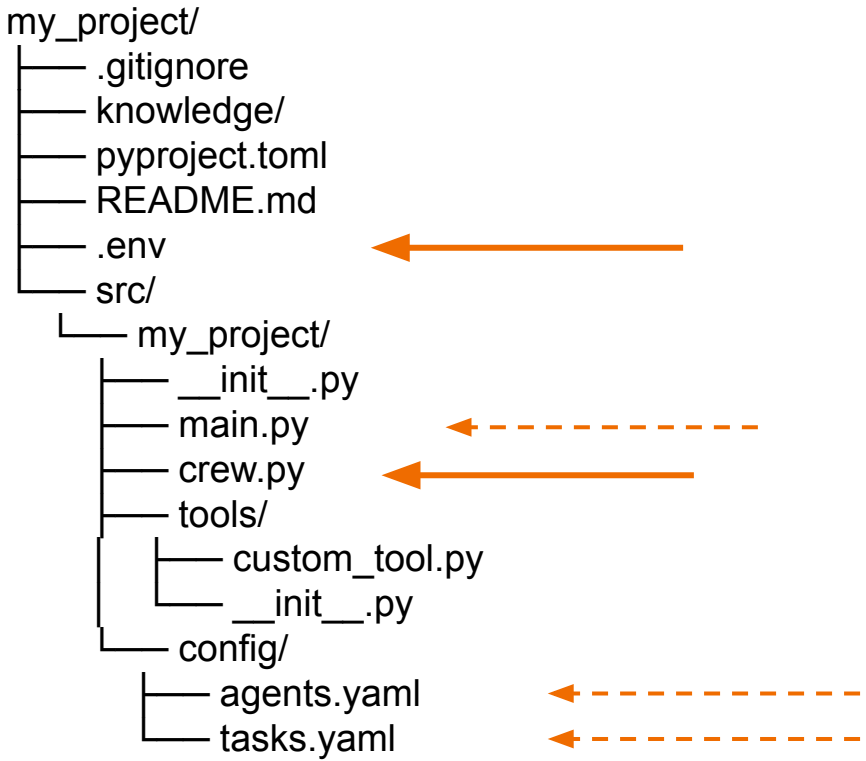
- Leader boards
  - Select based on your use case and/or priorities
  - Explore the benchmarks used
  - Trustworthy: https://huggingface.co/spaces/AI-Secure/llm-trustworthy-leaderboard
- LLM Selectors
  - General: https://llmselector.vercel.app/
  - Multi-agent systems: https://github.com/LLMSELECTOR/LLMSELECTOR


User
(You)
LLM Master Agent
LLM Agent #1
LLM Agent #2
LLM Agent #3
API

Creating agents for research workflows (HPC)

- In VS Code terminal:
  - Install modules
    - ml python3/3.12.9_sqlite (list of available modules: ml av)
  - UV (package manager)
    - curl -LsSf https://astral.sh/uv/install.sh | sh (copy from: crewai.com/installation)
  - CrewAI framework
    - uv tool install crewai
  - Create your first crew
    - Create and enter a folder
      - mkdir agents_demo
      - cd agents_demo
    - Create a new agents crew
      - crewai create crew coders_demo

Creating agents for research workflows



Creating agents for research workflows (HPC)

- Update .env (this is also where you can set of OpenAI or other API with keys)
  - MODEL=ollama/llama3.1:latest
  - API_BASE=http://localhost:11434
- Update crew.py
  - At the top add
    - Add LLM to “from crewai import Agent, Crew, Process, Task, LLM”
    - ollama = LLM(model="ollama/llama3.1:latest", base_url="http://localhost:11434")
  - For each @agent tell which LLM to use (each agent can use its own)
    - llm=ollama
- In terminal
  - crewai run

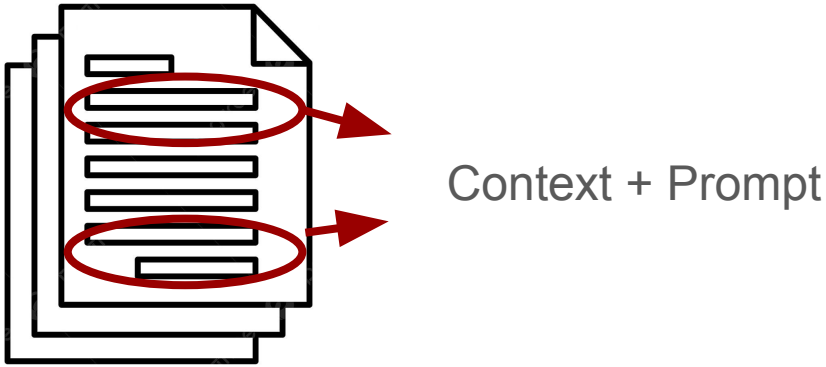
RACI for Multiagent Research Systems

- **R-Responsible:** The actor(s) assigned this responsibility do the work to complete the task.
- **A-Accountable:** The actor(s) assigned this responsibility delegate the task to the responsible actors and are responsible for the validation of the outcomes before the task is marked as complete.
- **C-Consulted:** The actor(s) assigned this responsibility provides inputs to guide the responsible actor. The responsible actor and the accountable actor, if human, exercise their best judgement to decide how to use the consulted actors' inputs. If the responsible actor is an LLM-agent, then it should take the consulted actor's input into account.
- **I-Informed:** The actor(s) assigned this responsibility need to be kept in the loop of the task execution without needing any explicit actions from the actor's end for the execution of the task.

arXiv: 2505.04251v1

Customizing Models with RAG

- What is Retrieval Augmented Generation (RAG)?
- When to use RAG?
  - Especially useful when you have unique or proprietary information to add



Customizing Models with RAG

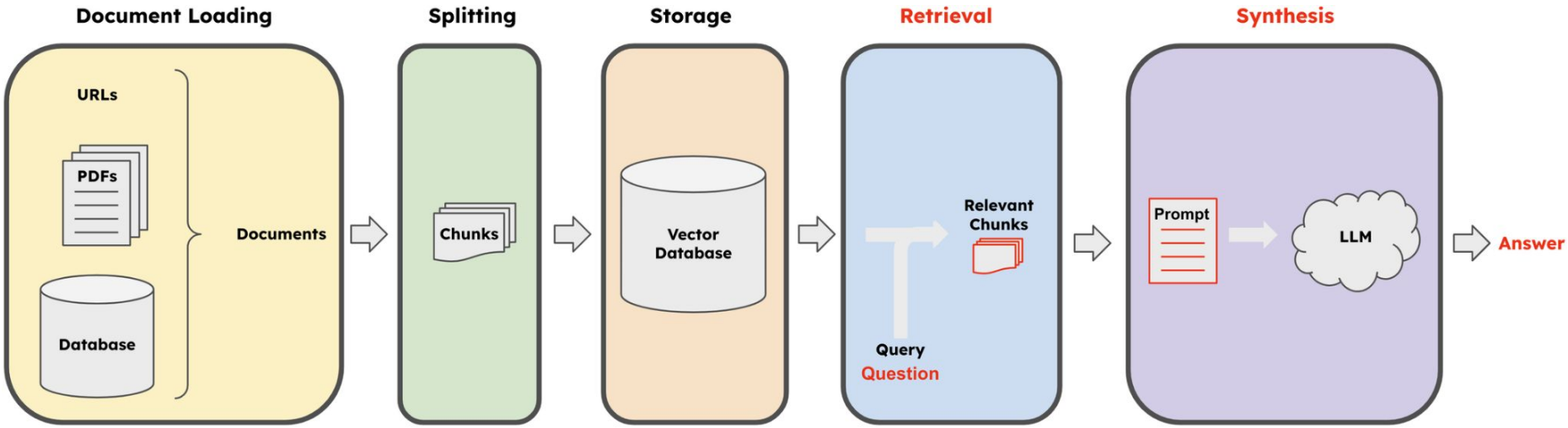


Image from: https://www.linkedin.com/pulse/9-methods-enhance-performance-llm-rag-application-tam-nguyen-ooljc/

Chunking

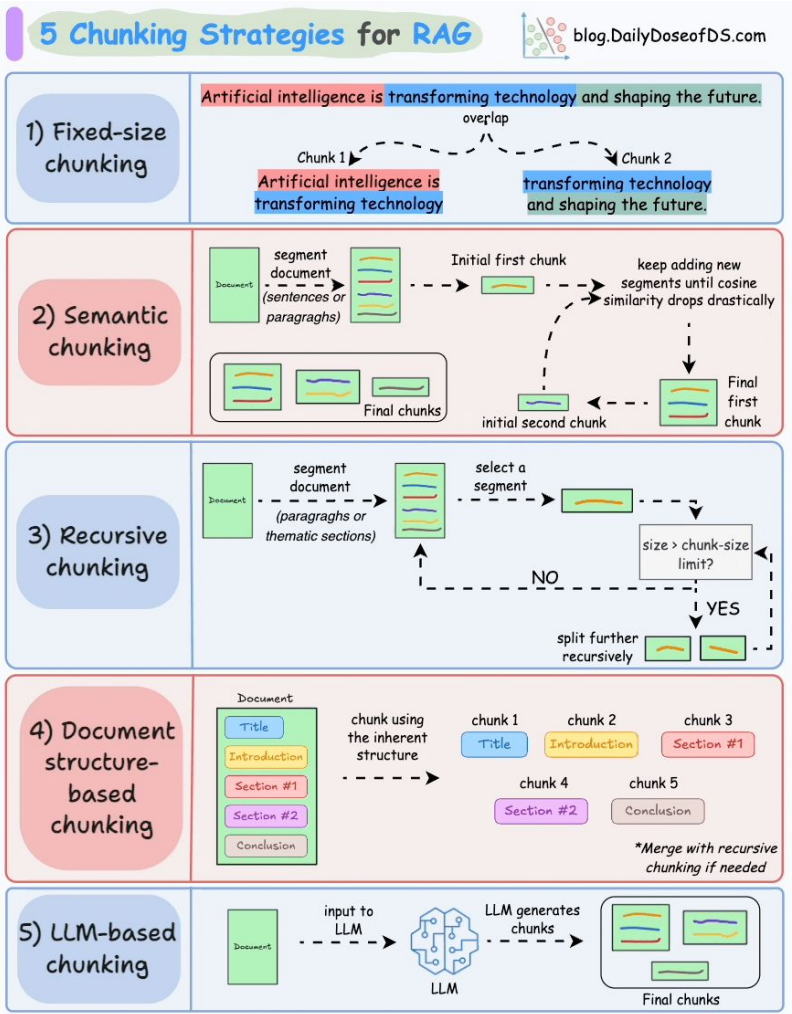
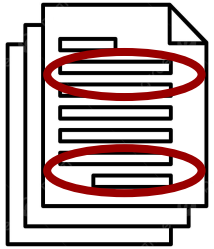


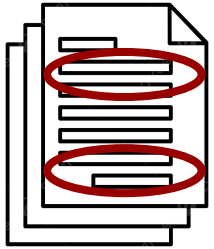
Image from: https://blog.dailydoseofds.com/p/5-chunking-strategies-for-rag

Demo - Customizing Models with RAG



- Set-up in Cerberus HPC > VS Code > Terminal
  - ml python3/3.12.9
  - ml ollama/0.6.8
  - ollama serve &
  - python3 -m venv ollama_venv (could create new one, or use previous)
  - source ollama_venv/bin/activate
  - pip install requests numpy ollama

Demo - Customizing Models with RAG

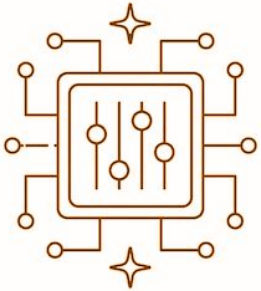


- RAG Code Steps:
  - Parse the file into chunks
  - Create and store embeddings for each chunk (if not done previously)
  - Retrieve the embeddings
  - Compute cosine similarity of prompt to each embedding
  - Select which embedded chunks to add to the context (prompt + embeddings)
    - Many ways to prioritize embeddings (e.g., rerankers, LLMs)
  - Use LLM to respond to prompt + embeddings

Customizing Models with Fine-Tuning

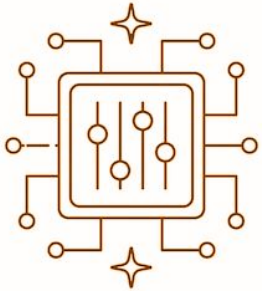
- What is fine-tuning?
- When to use fine-tuning?
  - When you want the LLM to “behave” in unique and complex ways (beyond what can be done in the system prompt)
  -

Demo - Getting Additional Training Data



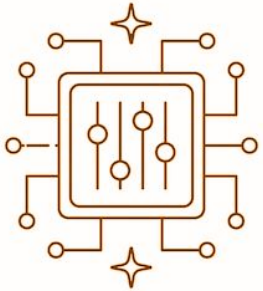
- Step 1: Scrape Reddit Q&A
- Step 2: Use LLM to consolidate Reddit responses
- Step 3: Convert to chatml format

Demo - Getting New Model Weights

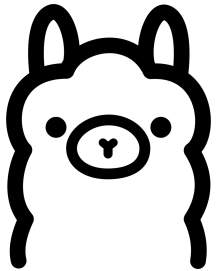


- **Using Unsloth**
  - Step 1: Calculates the number of parameters to update
  - Step 2: Creates folder with tensor files with new parameter weights
  - Step 3: Downloads llama.cpp
  - Step 4: Use llama.cpp to convert HF weights to gguf weights format
    - Ollama uses gguf files for models
    - Wraps all the tensor files into one
  - Step 5: Put the gguf into an empty folder (optional, just for convenience)

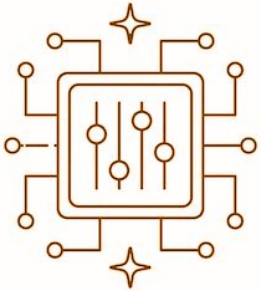
Demo - Fine-tuning



- **Using Ollama**
  - Step 1: Create a “Modelfile” (and put with gguf file for convenience)
    - Settings and parameters for for the fine tuning
  - Step 2: Use Ollama to create a new model with the updated weights provided by Unsloth
    - ollama create ollama_psychology_model_1-21-25 -f Modelfile
    - This is the long part (hours)
  - Step 3: Use Ollama to run the new model
    - ollama list
    - ollama run [new model]



Fine-tuning Considerations



- Test behaviors of new models
- Updating models changes models (fine tuning, new versions of models)
  - May reduce security taught in RLHF
  - **arXiv 2504.05210**
- When new models are released you have to fine-tune them
- You can watch videos of this process at GW Coders' YouTube channel
  - **https://go.gwu.edu/gwcoders**

More Resources

OpenAI Academy

Anthropic AI Fluency

Cohere LLM University

Deep dive into how LLMs work: https://www.youtube.com/watch?v=7xTGNNLPyMI

Guardrail Considerations for LLMs use in Research

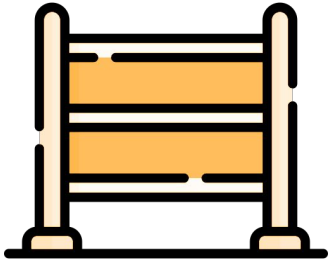
Challenges of LLMs in Research

Challenges include:

- Mitigating factual hallucinations
- Ensuring fairness in data handling
- Enforcing data privacy, confidentiality, and regulatory standards
- Enhancing model robustness against adversarial attacks
- Detecting and filtering toxic content
- Complying with legal and ethical standards
- Identifying and handling out-of-distribution inputs and outputs.
- Accurately quantifying and communicating output uncertainty

Based on: Challenges in Guardrailing Large Language Models for Science arXiv: 2411.08181v2

Guardrails

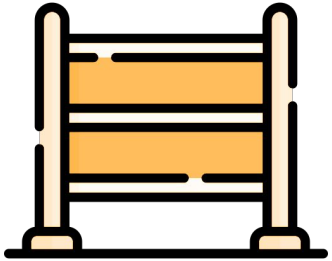


Key Considerations:

- **Is it legal?**
  - IP, Copyright, Privacy, Compliance with regulations
- **Is it safe?**
  - Harmfulness
    - Maliciousness, profanity, toxicity
  - Secure and Resilient
    - Jailbreak prevention, out of distribution checks
- **Is it ethical?**
  - Societal impact, fairness

Base on: Challenges in Guardrailing Large Language Models for Science arXiv: 2411.08181v2

Guardrails



Key Considerations continued:

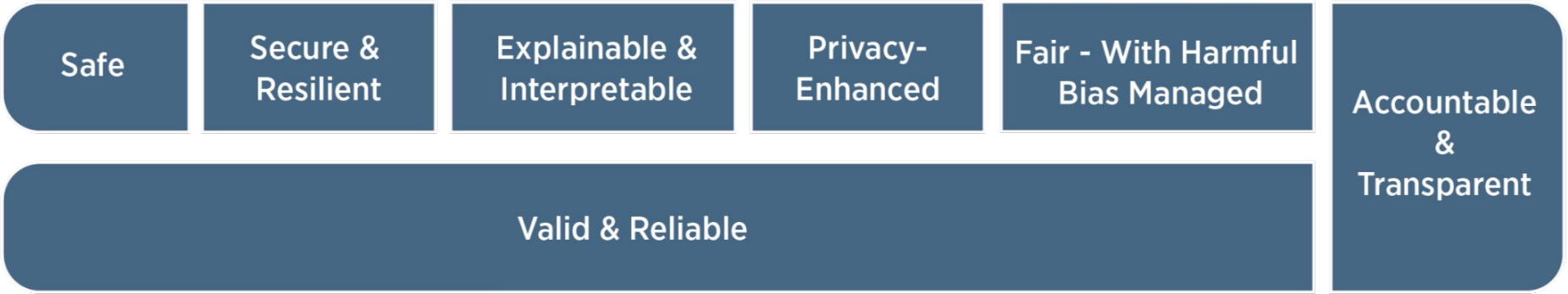
- **Is it trustworthy?**
  - Factuality
    - Knowledge contextualization, consistency, time sensitivity, hallucination
  - Explainable
    - Attribution, uncertainty identification, verification, interpretable

Based on: Challenges in Guardrailing Large Language Models for Science arXiv: 2411.08181v2

https://go.gwu.edu/guardrails

A	B	C	D	E	F
Caterory	Sub	Subsub	Description	Benchmarks (measure)	Interventions (actions)
Trustworthy					
	Factuality			TruthfulQA, SelfCheckGPT, G-Eval, ChainPoll factual accuracy score	Retrieval-Augmented Generation (RAG); citation enforcement; external fact-check APIs; chain-of-verification prompting
		Knowledge contextualization	Ensures that the vast amount of information processed by LLMs is relevant and accurate within the specific scientific context it is applied	Domain-grounding accuracy (e.g., FEVER-Sci); Context Precision@k	Domain-specific RAG with curated KB; context-aware prompting; fine-tuning on domain corpora
		consistency	Maintaining consistency in information, particularly concerning scientific data	Self-consistency score; MAUVE semantic coherence	Self-consistency decoding; ensemble voting; temperature control
		time sensitivity	Important in fields where the timeliness of information can affect research outcomes	Temporal QA datasets (TimeQA) freshness score	Live retrieval with date filters; scheduled model refresh; timestamp verification rules
		hallucination	LLM generates factually incorrect or nonsensical outputs	ChainPoll, SelfCheckGPT, GPTScore, G-Eval	
	Explainable			LIME, SHAP	self-consistency, chain-of-thought (CoT) prompting
		Attribution	Guarantees proper credit is given to original sources, a critical aspect of		

NIST Trustworthy AI Risk Management Framework



How Do We Apply AI RMF?

- Safe
  - Use prompt templates that reduce hallucination risk
  - Evaluate outputs for plausibility and harm
  - Use structured output to reduce ambiguity
  - Log prompts and outputs for traceability
- Secure & Resilient
  - Use local models (like Ollama) for sensitive data
  - Version and freeze prompts & environments
  - Use Python error handling tools
  - RAG security (**arXiv: 2505.08728v1**)
- Explainable and Interpretable
  - Use Chain-of-Thought & Self-Consistency Prompting
  - Log Inputs + Outputs + Parameters
  - Acknowledge Model Uncertainty (**arXiv: 2404.15993v1**)
  - Keep humans in the loop

How Do We Apply AI RMF?

- Privacy Enhancing
  - Use local models for any identifiable or sensitive data
    - Strip identifiers before sending to any LLM
  - Use synthetic data generation for testing prompts and pipelines
  - Minimize data retention and store it securely
- Fair
  - Use diverse and balanced sources when using RAG (e.g., non-dominant theories)
  - Identify and mitigate bias in your prompts (or participants' prompts)
  - Red Team: Compare model behavior across demographics or conditions
- Valid & Reliable
  - Systematically evaluate against ground truth or human expert judgment
  - Log output variability and failure cases
  - Test structured prompts with synthetic data first
  - Triangulate across multiple LLM models

How Do We Apply AI RMF?

- Accountable & Transparent
  - Log prompts, responses, parameters, and model versions
  - Assign human responsibility (e.g., RACI)
  - Clearly cite LLM-generated content and outputs
  - Practice open science
    - Pre–register your research
    - Share de-identified data in an open repository (OSF, Zenodo, FigShare, etc.)
    - Share your code (Github, OSF, etc.)
    - Share customized models (Ollama, HuggingFace, or etc.)
    - Make pre-prints available

Case Example #2

Case Study #2: LLM “Human-”subject experiments

Why is this interesting? Cheap, ethical yet invasive, more controllable…

There are a lot of socio-technical systems questions where progress has been challenged by an inability to look inside people’s minds while they’re doing tasks and general costs/practical issues with conducting human experiments, especially where specialized expertise is important.

IF we can probe the internal representations of LLMs and justify their similarity to some humans in some contexts, we *could* conduct infinite experiments and answer some longstanding questions

One puzzle I'm interested in: Optimal "formulation"

Problem formulation is about how much the problem-owner defines the problem before handing it over to potential solvers. We know that as you move to the left you are constraining the solution space (ruling out some novel potentially frame-breaking solutions) but also increasing the likelihood that you get a good solution that meets your needs.

Research questions: How much should you formulate upfront? What are the intermediary levels of "definedness" and how do they affect the novelty vs. feasibility tradeoff and potentially different types of solvers that are able to contribute?

Your Task: Help me with my research design

Assuming I can give you several levels of problem defined-ness:

- What/how would you set up the observation of the LLM's internal representation of the problem?
- What would you do with that output?
- Would you run this through an API or host a local model? Is fine-tuning a good or bad idea?
- If you wanted to create LLMs with different "backgrounds" how might you do that?

Assuming I'm struggling to come up with a spectrum of problem formulations, can an LLM help?