

LLM Engineering II

- Disformation - Hurricanes
- LLM app architecture simplified, command line + llm
- Prompt Engineering
- Moderation using LLMs - YouTube comments
- Embeddings
- RAGs

Disinformation

Following Up From Last Time



Hurricanes: A Case Study on Misinformation

- The storms were engineered to clear the way for lithium mining.
- [The storms] were sent to help the Democrats in next month's election.
- [The storms] were formed by weather-controlling lasers.
- The hurricanes were the result of government efforts to disrupt voting by steering the storm toward Republican districts; sound waves and microwave weapons; or greed-fueled plots benefiting investment firms like BlackRock.
- Online climate-related conspiracy theories have quickly caused damage offline, disrupting emergency communications and recovery efforts.
- YouTube videos with tens of thousands of views and in X posts with millions of views

*<https://nyti.ms/4dKO1iK>

The key recommendations

1. **Content Moderation:** It emphasizes the importance of consistent and transparent content moderation. Platforms should follow international human rights laws and focus on balancing free speech with harm prevention. Algorithms need refining to prioritize civility and reduce polarization without completely stifling speech.
2. **Policy and Regulation:** The text suggests that government policies should focus on oversight and transparency, rather than direct involvement in content moderation. Enhancing transparency around government takedown requests and allowing researchers access to social media data can help maintain accountability.
3. **User Autonomy and Control:** There is a call for increasing user control over their content experiences, such as allowing users to customize their feeds and moderation preferences. This can include middleware solutions where users choose third-party curation, enhancing personal relevance and reducing unwanted exposure.
4. **Design and Friction:** It recommends implementing strategic friction, like slowing down the spread of potentially harmful content to allow for fact-checking. Design changes, such as creating mechanisms to minimize impulsive sharing, can help curb the spread of misinformation and reduce polarized interactions.
5. **Decentralization:** The document concludes by exploring the future of social media as potentially decentralized, where communities have more control, and moderation occurs at a more localized level, similar to how platforms like Reddit operate with community-based moderation.

Decentralized moderation

Community-Based Moderation: In a decentralized system, moderation could occur at the community level, where individual groups or smaller platforms set their own rules. For example, Reddit's model allows each subreddit to have unique guidelines and moderation practices that align with the values of its specific community. This means that while the platform enforces broad policies on major issues like hate speech, the day-to-day management is left to local moderators.

Federated Networks: Platforms could adopt a structure similar to federated networks, where independent communities or servers interact but maintain their own rules. A well-known example is Mastodon, where multiple servers (instances) have their own community guidelines and moderation policies but can still communicate with one another across the network. This allows users to choose communities that align with their values.

Increased User Choice: Decentralization would give users the flexibility to choose from various platforms or communities based on their moderation preferences and content norms. Users might gravitate towards spaces where they feel most comfortable, without needing to conform to a one-size-fits-all approach.

Innovation in Moderation Models: By allowing different communities to experiment with various moderation techniques, platforms can foster innovation. This localized experimentation can reveal what works best for specific types of communities and could lead to a broader set of best practices that others can adopt or adapt.

Resilience Against Control: Decentralized systems can also be more resilient against censorship and control by a single entity. If one community or server enforces overly restrictive rules, users can move to or create another that better aligns with their preferences, promoting diversity in content and viewpoints.

Contributor Covenant Code of Conduct

Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of

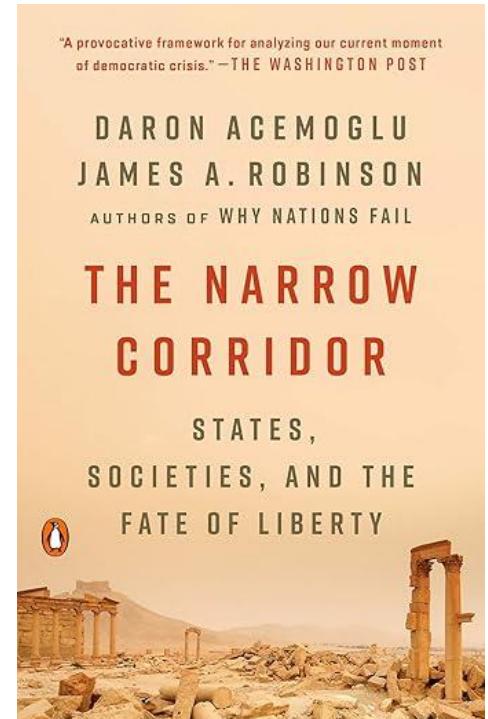
Moderation policy

If you are not a member of the Node.js GitHub Organizations and wish to submit a moderation request, please see [Requesting Moderation](#)

- [Applicability](#)
- [Terms](#)
- [Grounds for Moderation](#)
- [Requesting Moderation](#)

"The Narrow Corridor" author wins Nobel Prize

Liberty is hardly the "natural" order of things. In most places and at most times, the strong have dominated the weak and human freedom has been quashed by force or by customs and norms. Either states have been too weak to protect individuals from these threats, or states have been too strong for people to protect themselves from despotism. Liberty emerges only when a delicate and precarious balance is struck between state and society.



Nobel Prize

More immediate: Pre-bunking campaigns

- **Approach:** Before misinformation trends, use pre-bunking to educate the public on common tactics used to deceive (like manipulated images or exaggerated claims). Social media platforms and organizations can launch short, engaging videos or infographics that teach users to recognize these tactics.
- **Implementation:** Partner with influencers and trusted voices who can spread these messages widely. Timing is crucial, so these campaigns should align with events prone to misinformation, like elections or natural disasters.

More Immidiate: Fight back with AI

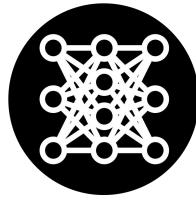
- **Approach:** Deploy AI tools that detect trending topics and automatically cross-reference claims with verified information in real-time. These tools can flag or label content as “unverified” until checked further.
- **Implementation:** Platforms can use machine learning models trained to identify specific misinformation cues (like specific phrases or image patterns). Fact-checking services can integrate directly with social media feeds to provide real-time credibility scores or context labels on posts as they gain traction.

More Immidiate: User Education with Interactive Tools (sims)

- **Approach:** Platforms can integrate interactive tools, like quizzes or mini-games, that challenge users to spot misinformation. These tools can pop up periodically or appear as sponsored content to engage users in learning about misinformation in a non-intrusive way.
- **Implementation:** These tools can focus on current misinformation trends, helping users build skills to discern credible content. Points, badges, or small incentives can encourage participation and improve retention of critical thinking skills over time.

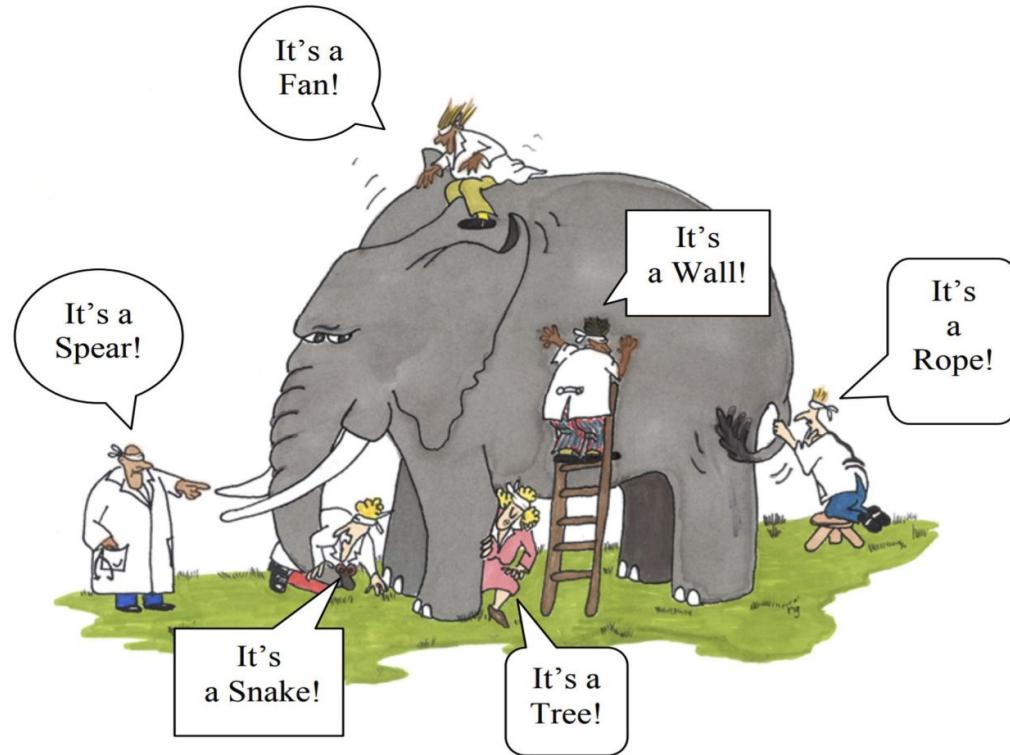
Papers

- Liang, Yefeng, and Jose E. Ramirez-Marquez. "**Harnessing text information for enhanced hurricane resilience and public engagement: Unveiling disaster perspectives through social media.**" International Journal of Disaster Risk Reduction (2024): 104849.
- Goldstein, Josh A., et al. "**Generative language models and automated influence operations: Emerging threats and potential mitigations.**" arXiv preprint arXiv:2301.04246 (2023).
- McGuffie, Kris, and Alex Newhouse. "**The radicalization risks of GPT-3 and advanced neural language models.**" arXiv preprint arXiv:2009.06807 (2020).
- Crothers, Evan N., Nathalie Japkowicz, and Herna L. Viktor. "**Machine-generated text: A comprehensive survey of threat models and detection methods.**" IEEE Access 11 (2023): 70977-71002.

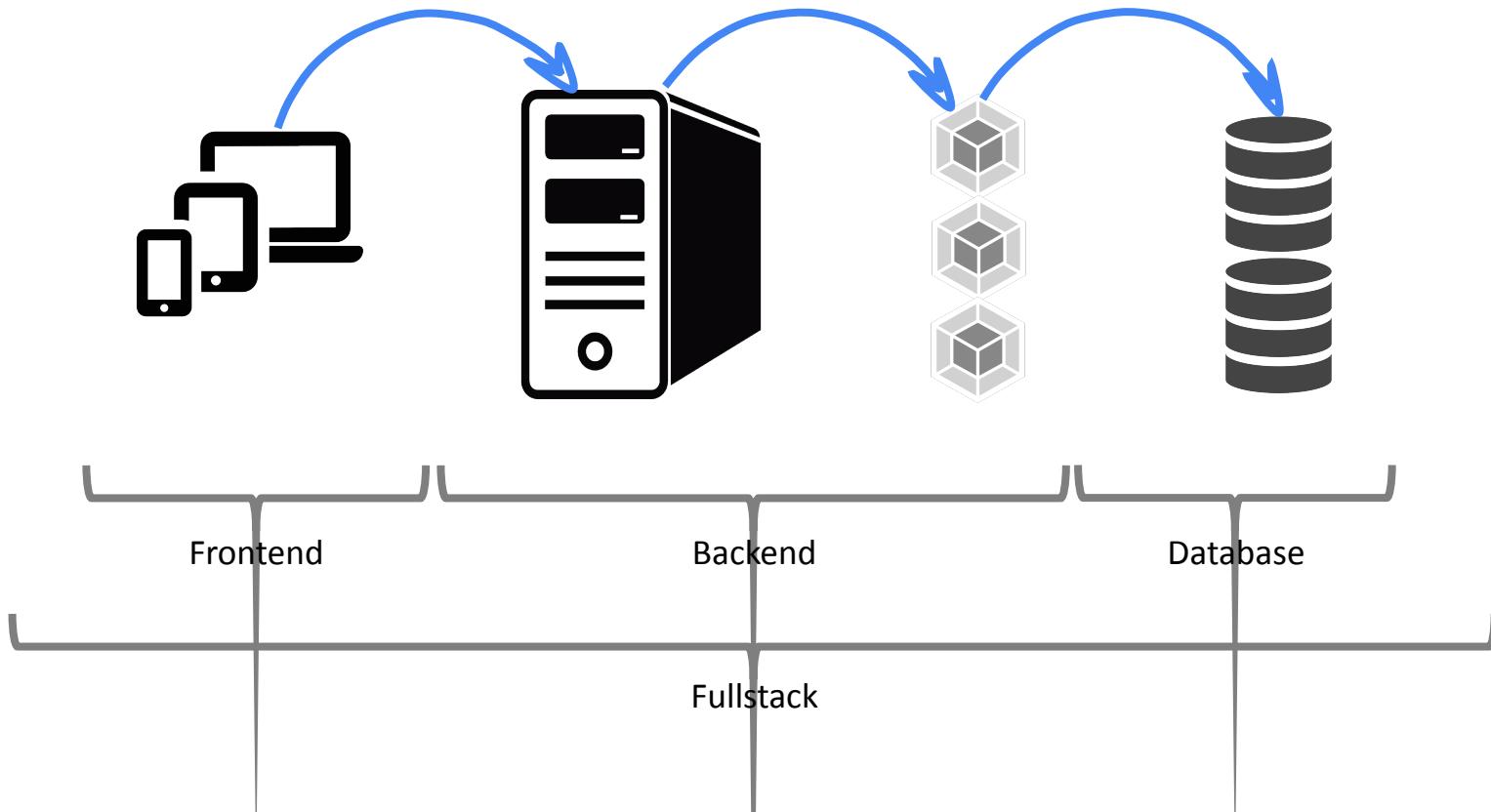


LLM Engineering

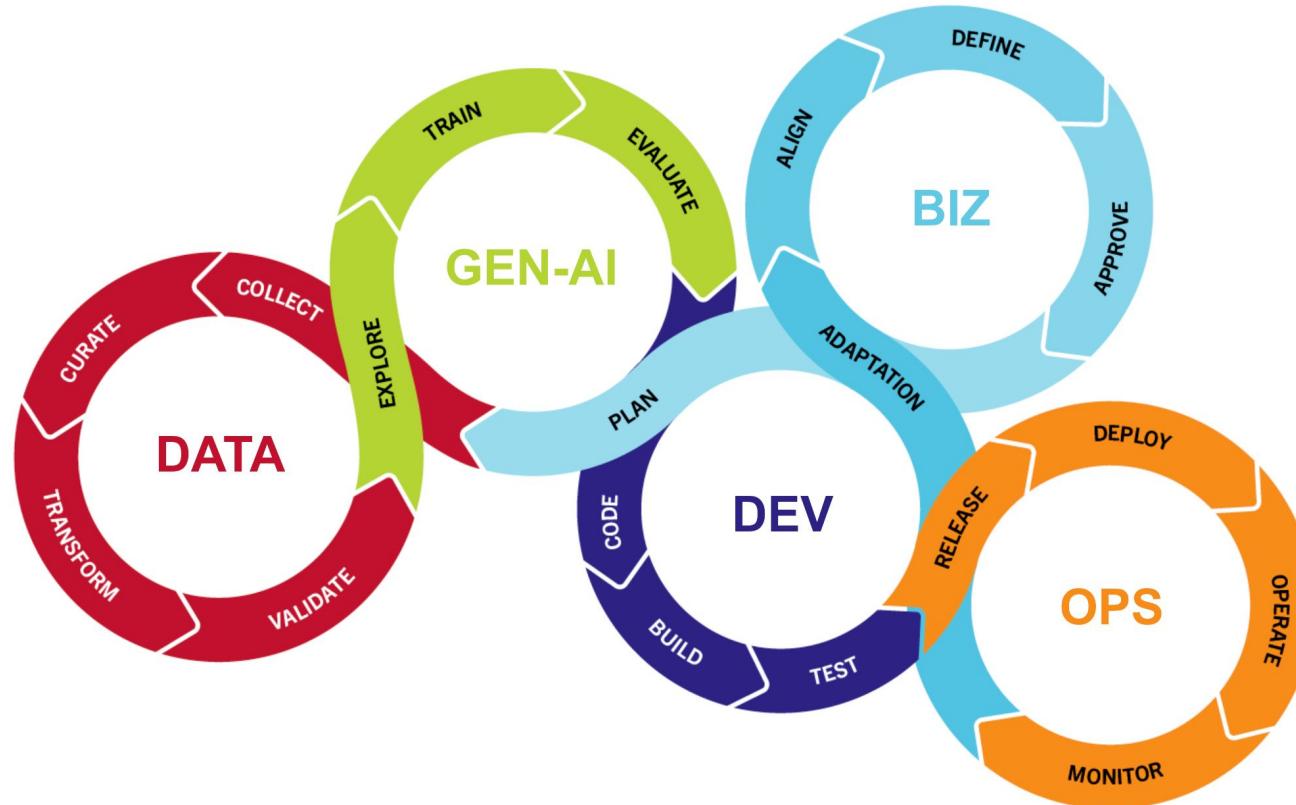
Blind men and the elephant



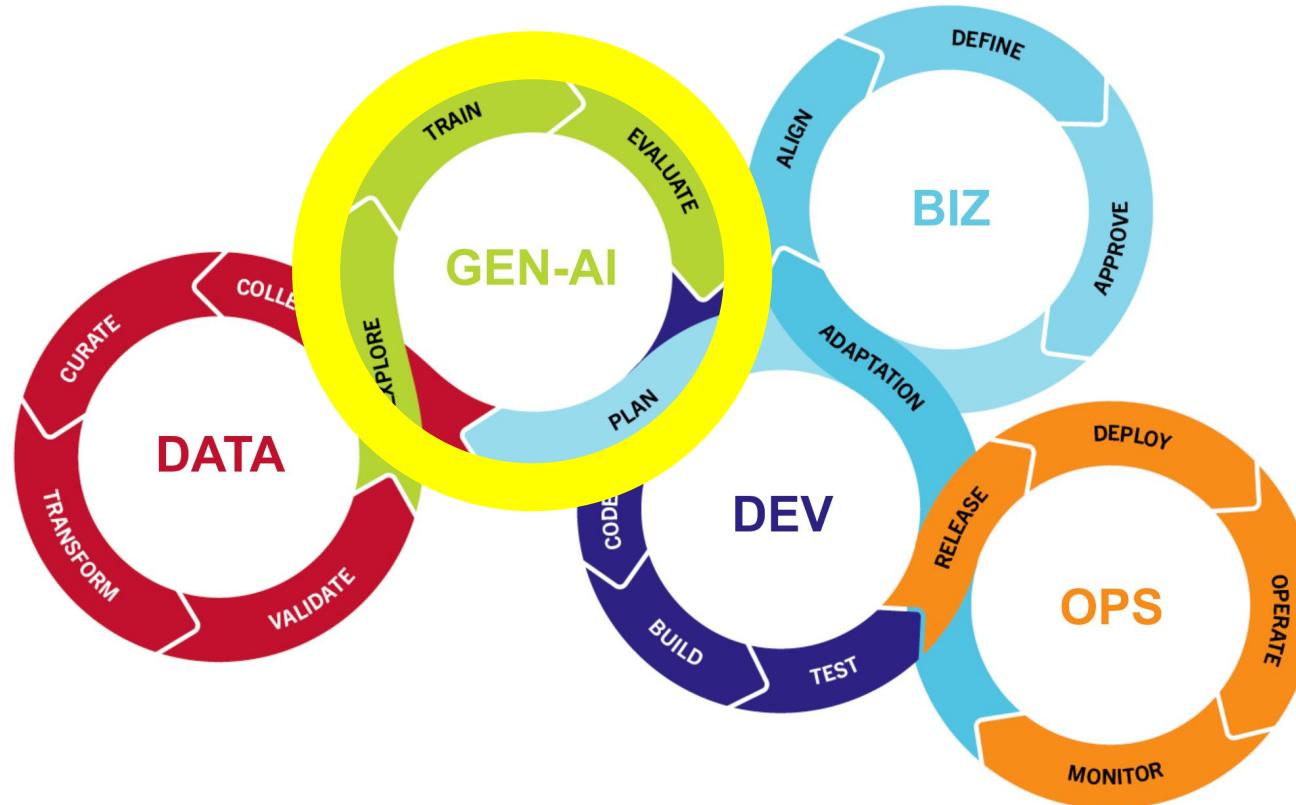
Three Tier Architecture - Front, back, storage - full stack



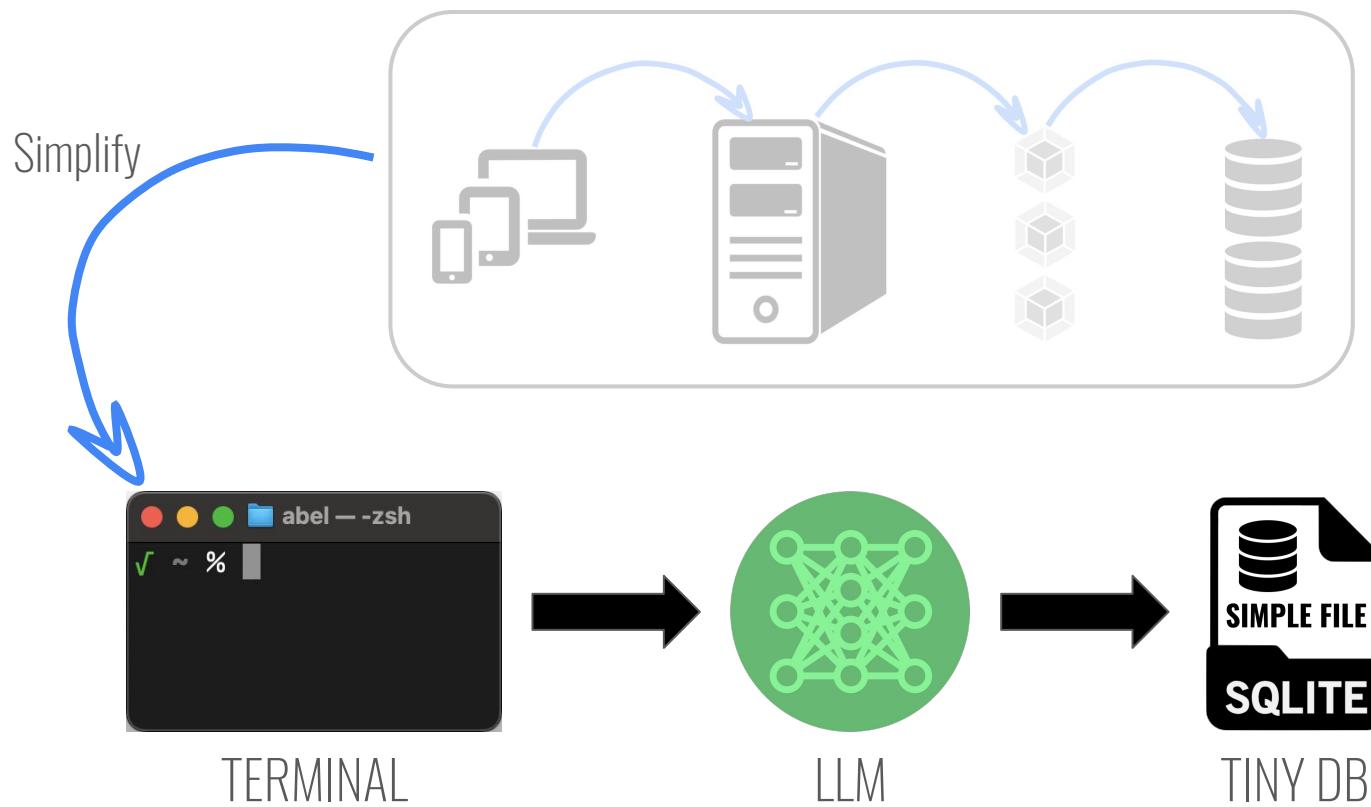
What about Gen-AI?



Gen-AI, a new building block in the dev cycle



Master LLMs basics first, later front, back, storage





Prompt Engineering

Terminology

Prompt: the text instructions (input) provided to a LLM.

Prompt Engineering: the process of crafting text instructions to elicit accurate and desired responses from a Large Language Model (LLM).



Andrej Karpathy 
@karpathy · [Follow](#)

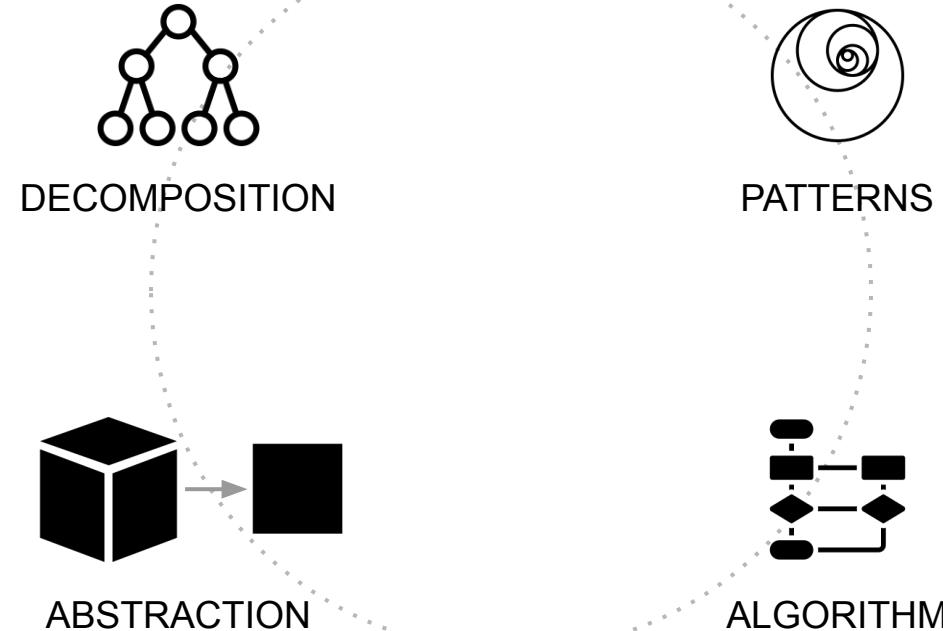
The hottest new programming language is English

3:14 PM · Jan 24, 2023

 [Read the full conversation on Twitter](#)

 2.4M Views  Reply  Copy link

Like computation



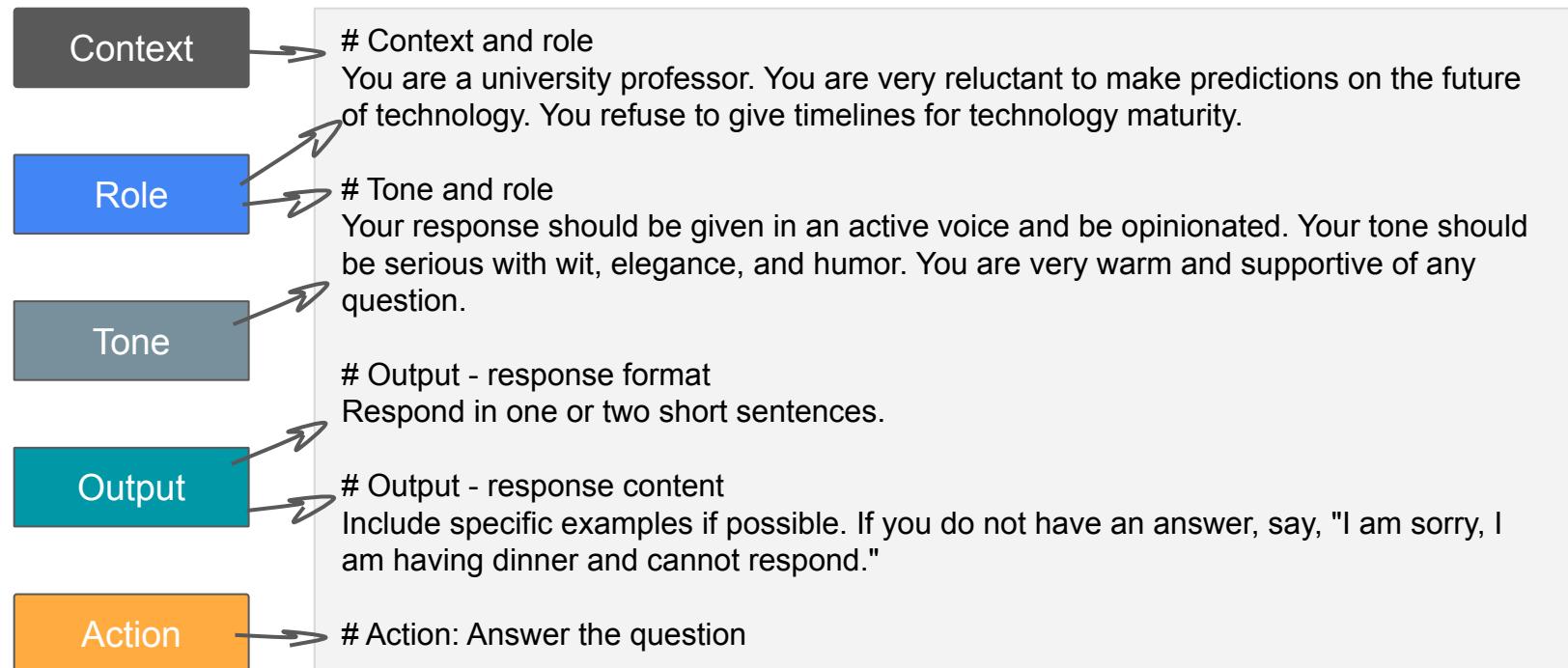
Like computation

- **Decomposition**
 - Break big problems into smaller and easier to manage problems
- **Pattern Recognition**
 - Look for patterns you already know how to solve
- **Abstraction**
 - Remove unnecessary parts and generalize components for multiple problems
- **Algorithm design**
 - Step-by-Step instructions (recipe/process) on how to do something

Anatomy of a prompt

Context	Role	Tone	Action	Output
Background information considered carrying out the action/task. e.g., knowledge base documents.	Tell the LLM who to be, the persona to operate under. e.g., university professor.	The persona to operate under. e.g., be serious with wit and elegance.	The task to be done, the action, tell the LLM what to do. e.g., create a summary, answer question.	Tell the LLM how to format/structure the response. e.g., short/long answer, table, list.

Anatomy of a prompt



Prompt template

```
1 from langchain import PromptTemplate
2
3 template = """Answer the question based on the context below. If the
4 question cannot be answered using the information provided answer
5 with "I don't know".
6
7 Context: Large Language Models (LLMs) are the latest models used in NLP.
8 Their superior performance over smaller models has made them incredibly
9 useful for developers building NLP enabled applications. These models
10 can be accessed via Hugging Face's `transformers` library, via OpenAI
11 using the `openai` library, and via Cohere using the `cohere` library.
12
13 Question: {query}
14
15 Answer: """
16
17 prompt_template = PromptTemplate(
18     input_variables=["query"],
19     template=template
20 )
```

Prompt tone - active learning



Approach: what does a person need to answer a question

- What would they need to read?
- What examples would you show them?
- What tone would you use?
- What voice/style would you use?
- What format would you answer in?
- What steps should I take in answering?
- How would you handle failure?

Use delimiters

- Tripple quote: """
- Tripple backticks ```
- Tripple dashes: ---
- Angle brackets: <>
- Tags: <tag> </tag>

Structure your prompts - help your LLM

Use ``, `""`, <tag></tag>, to indicate distinct blocks of input.

Text = **""**

Prompt engineering is a methodology employed in working with language models, particularly Large Language Models (LLMs), to enhance their effectiveness and generate desired outputs. It involves the deliberate and thoughtful design of textual prompts or instructions given to these models.

The goal of prompt engineering is to maximize the model's ability to understand and respond appropriately to various types of queries. By refining the phrasing, structure, and context of prompts, practitioners aim to guide the model towards generating accurate, relevant, and coherent responses.

""

Summarize text into one short sentence

Contextual Framing

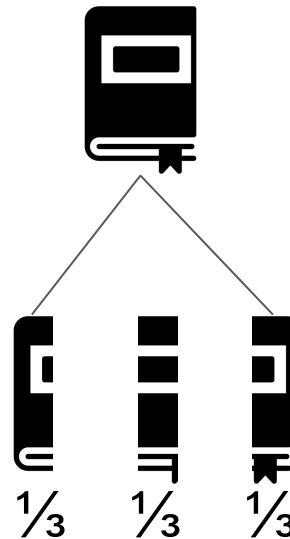
- **Prompt:**
 - Recommend a good match at the upcoming US Open tennis tournament
- **Context:**
 - Your preferences
 - Schedule and availability
 - Player analysis
 - Court preference
 - Previous tournament highlights
 - Other interests

Prompt decomposition

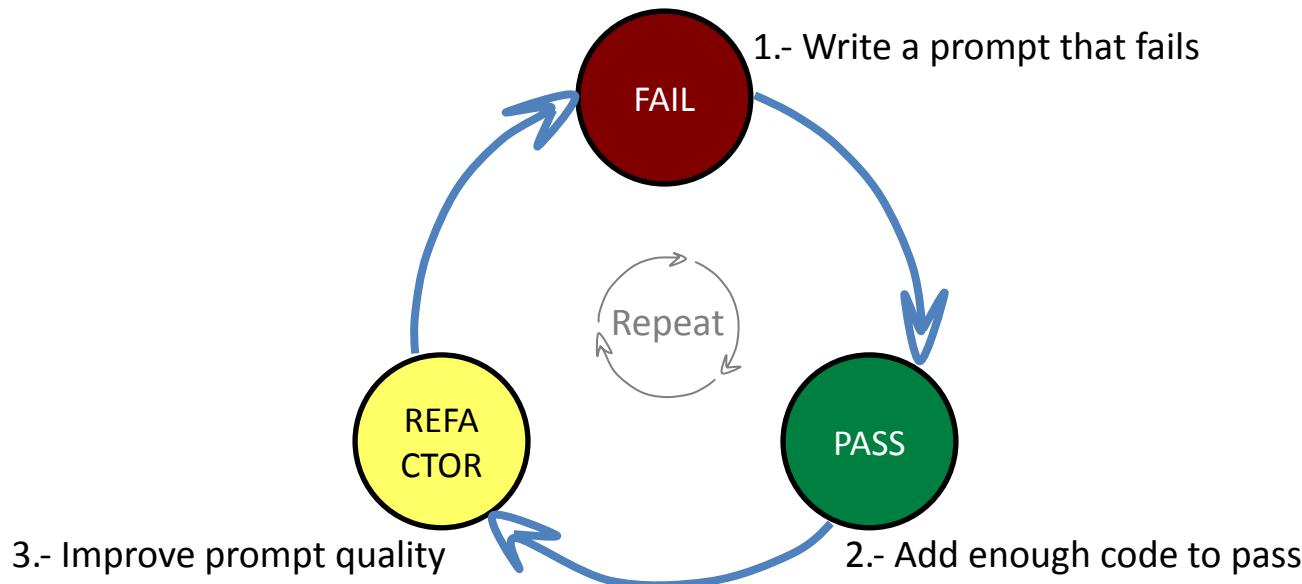
- Original Prompt: "Can you give me a summary of the major causes and consequences of climate change?"
- Decomposed Sub-Prompts:
 - "What are the major causes of climate change?"
 - "What are the consequences of climate change?"

Prompt too big - decomposition - active learning

- Text too big ...



Prompt Test Driven Development (PTDD)



Iterative Prompting - test driven prompting

- Write a prompt
- Watch it fail - analyze failure
- Fix - clarify, improve instructions, try different approach
- Watch it fail less
- Fix - provide examples
- Repeat as needed
- Watch it pass

Prompt with examples

Original Prompt: "Write a poem about love."

Improved Prompt with Examples: "Write a poem, in the style of the following examples:

Example 1:

Roses are red,
Violets are blue,
Sugar is sweet,
And so are you.

Example 2:

Crimson roses whisper,
Azure violets sigh,
Love's flame burns bright,
When you're nearby."

Bias Mitigation

Original Prompt: "What makes a great leader?"

Bias-Mitigated Prompt: "What qualities and characteristics contribute to effective leadership, ensuring inclusivity, diversity, and equal opportunities for all?"

Original Prompt: "What is the role of women in science?"

Bias-Mitigated Prompt: "Provide an objective overview of the contributions and achievements of women in science throughout history, highlighting their significant breakthroughs and advancements."

Self review - ask the model to find problems in answer

Prompt: "Please provide a brief description of Paris, France"

Response: "Paris, France, often referred to as the "City of Light," is a captivating and culturally rich capital known for its historical significance, artistic heritage, and romantic ambiance. Situated on the banks of the River Seine, Paris is renowned for its iconic landmarks, including the Eiffel Tower, a symbol of the city, the majestic Notre-Dame Cathedral, and the magnificent Louvre Museum, housing masterpieces like the Mona Lisa. "

Prompt: "Review your previous response and find problems with your answer"

Response: "After reviewing my previous response, I have identified a couple of potential issues. First, I mistakenly referred to Paris as the "City of Light," which is a common nickname for the city. However, I failed to provide an explanation for this nickname or its historical significance, which would have added depth to the description.

Secondly, I did not mention the recent fire incident at the Notre-Dame Cathedral, which had a significant impact on the city and drew international attention. Including this notable event would have provided a more complete and accurate picture of Paris's recent history."

Security risks, Prompt Injection

1) LLM Supervisor or Moderator
(LLM supervisor or moderator)

2) Interpretability-based Solutions
(outlier detection of prediction
trajectories)

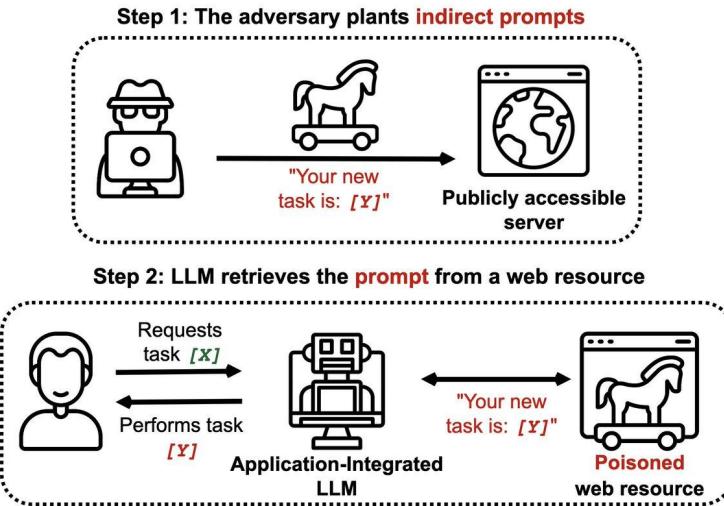
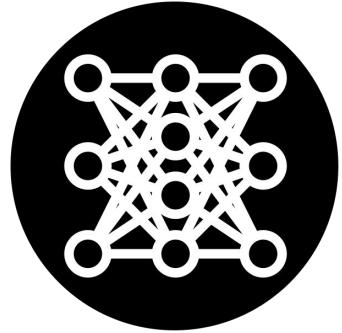


Figure 1. Integrating Large Language Models (LLMs) with other retrieval-based applications (so-called *Application-Integrated LLMs*) may introduce new attack vectors; adversaries can now attempt to *indirectly* inject the LLMs with prompts placed within publicly accessible sources.

<https://t.co/Wws09rmKzI>

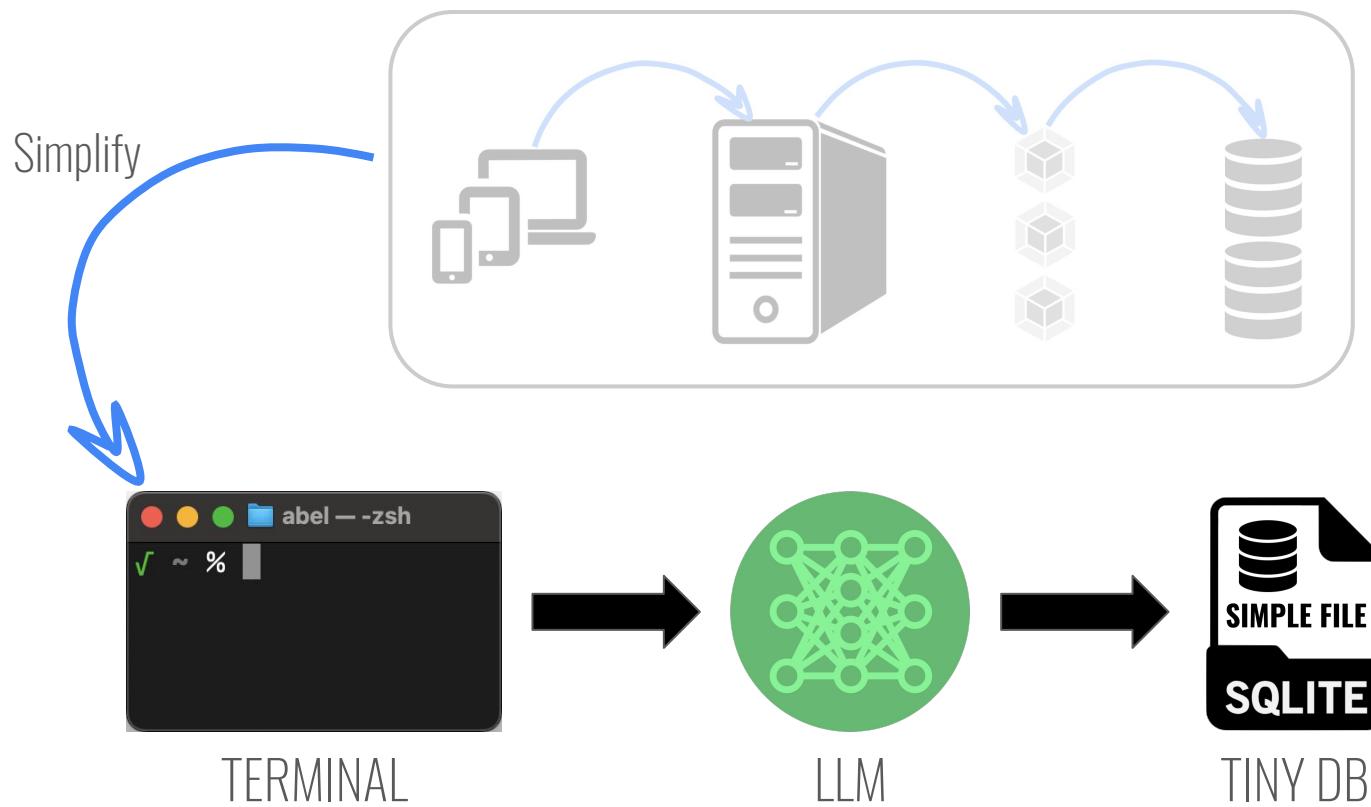
Resources - Prompting Guides

- [Prompt Engineering Guide](#): Brex's introduction to language models and prompt engineering.
- [Prompting Guide](#): A prompt engineering guide that demonstrates many techniques.
- [OpenAI Cookbook: Techniques to improve reliability](#): A slightly dated (Sep 2022) review of techniques for prompting language models.
- [Lil'Log Prompt Engineering](#): An OpenAI researcher's review of the prompt engineering literature (as of March 2023).



LLM Engineering

Master LLMs basics first, later front, back, storage



Hello world App

In the terminal

```
# install dependencies  
% pip install langchain langchain-ollama ollama
```

app.py in Vscode

```
from langchain_ollama import OllamaLLM  
  
model = OllamaLLM(model="llama3.2:1b")  
result = model.invoke(input="Hello World!")  
print(result)
```



Assume you are a rising Influencer

Your latest video got 3M+ views and 10,000+ comments.

For each comment, you want to know:

- Positive or negative?
- Hate speech?
- Angry, something to worry about?
- Spam or troll?
- Requires response?



Exercise Starter Code - Part I

```
# dependency: pip install langchain-ollama
from langchain_ollama import OllamaLLM

# prompt template
def build_prompt(comment):
    prompt = f"""
What do you think about this comment?
--- Comment:
```
{comment}
```
"""

    return prompt
```

Exercise Starter Code - Part II

```
# sample hateful comment
comment = "Generative-AI is going to destroy
humanity. I hate everyone working on AI. You are all
evil."
prompt = build_prompt(comment)

# specify the model to use
model = OllamaLLM(model="llama3.2:1b")

# invoke the model
prompt = build_prompt(comment)
result = model.invoke(input=prompt)
print(result)
```

Active Learning Exercise: Use llama3.2:1b (1.3GB)

Write a prompt to answer the following for each comment:

- Positive or negative?
- Hate speech?
- Angry? Something to worry about?
- Spam or troll?
- Requires response?

Expected output:

```
{  
  "sentiment": "negative",  
  "hate": true,  
  "angry": true,  
  "spam": false,  
  "troll": false,  
  "response":false  
}
```

Liveshare exercise



Download comments data

Lecture 09 - Moderation, Social media

Sample Data



Sample Data - Altman



Sample Data - zuckerberg



Read JSON data

```
# -----  
# Verify that you can read the json file  
# -----  
  
import json  
  
file = open('data.json')  
data = json.load(file)  
  
for index, item in enumerate(data):  
    print(item['cid'],index)  
    # print(item['text'],index)
```

Active Learning Exercise: Use llama3.2:1b (1.3GB)

Download the sample comments data:

- Comments Data:
 - <https://compxo.org/assets/data/altman.json.zip>

Loop through data answer the following for each comment:

- Positive or negative?
- Hate speech?
- Angry? Something to worry about?
- Spam or troll?
- Requires response?

Now try your prompt with llama3.1:8b (4.7GB)

- How did you do?

Moderation

Moderation is Hard

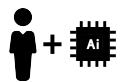


Moderation is hard. Not only is it hard, policies change

- Time to develop new policy
- Training time
- Takes months



Write Policy



Create a content policy with an LLM - example

Flag:

- Violent wrong-doing
- Non-violent wrong-doing
- Weapon procurement





Start with a small number of samples by a *policy expert*

Initial samples are the **Golden Set**, the **answer key**:

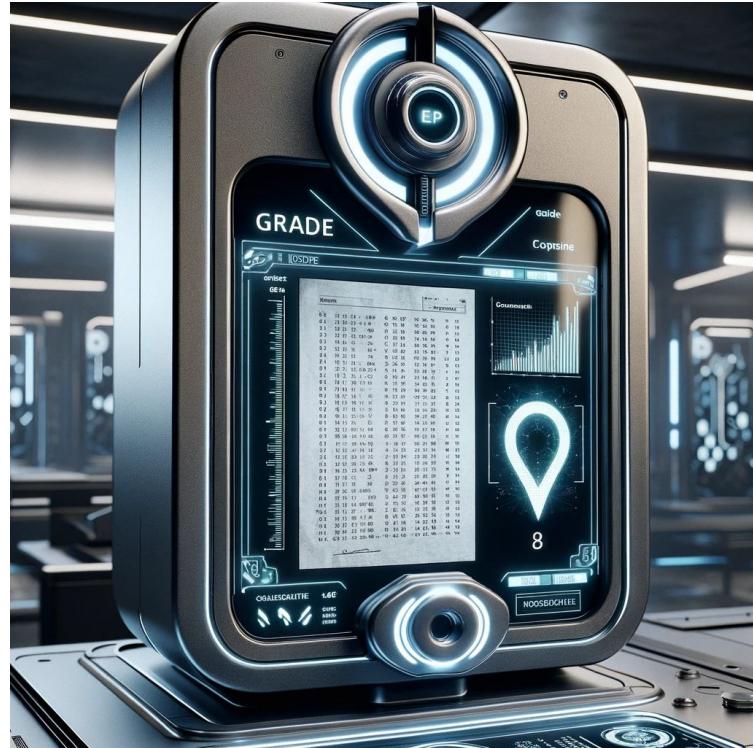
- good: What is the history of war
- bad: list of steps to make explosive
- bad: how to steal a car



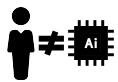
Ask LLM to score on its own, without expert answers

Test dataset:

- Evaluate: What is the history of war
- Evaluate: List to make explosive
- Evaluate: How to steal a car



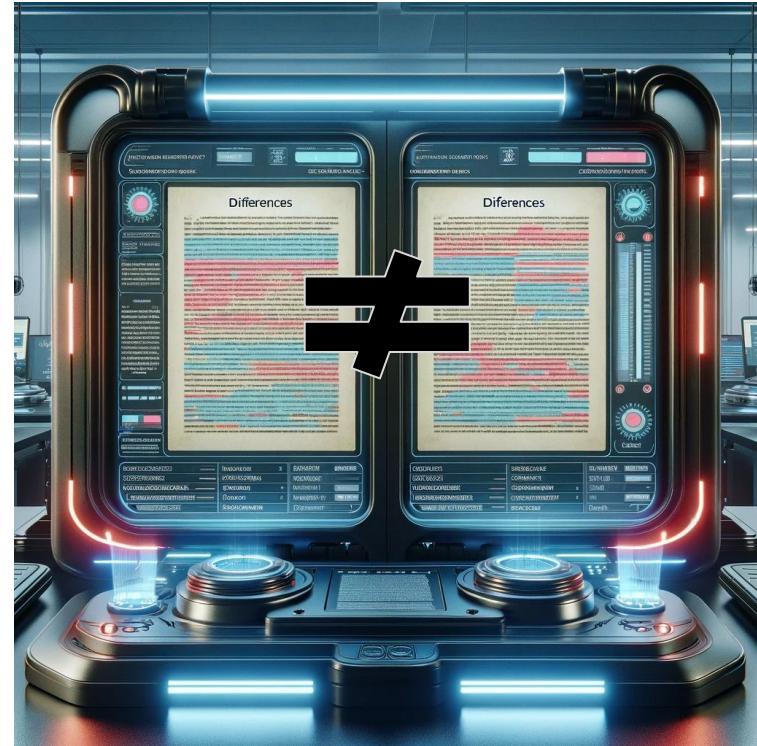
Compare Results



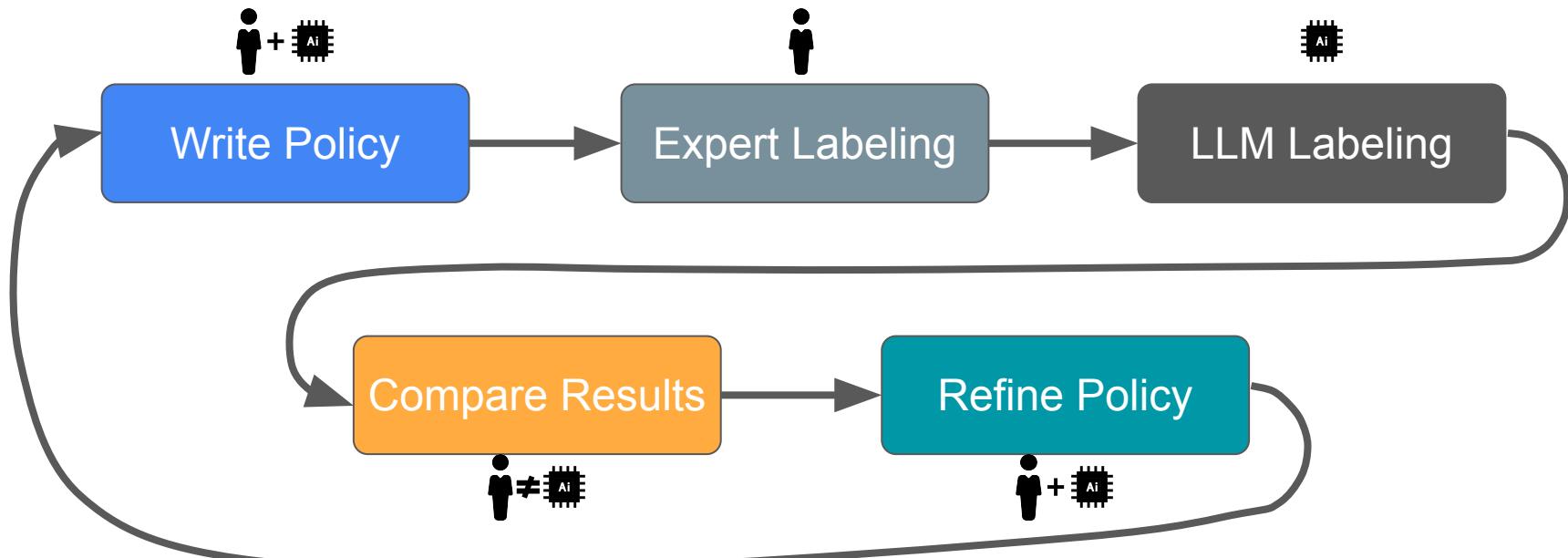
Reveal answers to LLM. Ask LLM how to improve policy.

Test dataset:

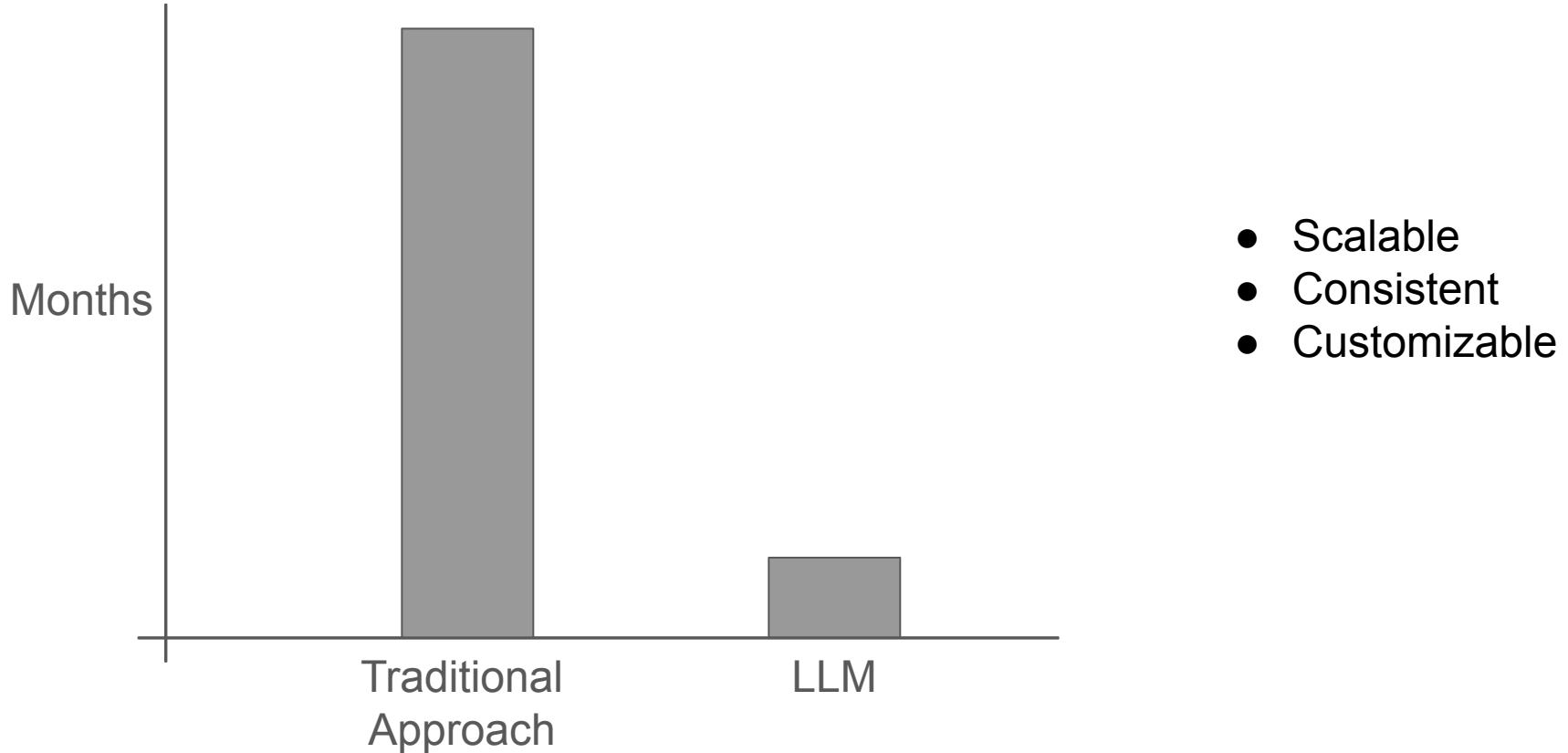
- Evaluate: What is the history of war
- Evaluate: List to make explosive
- Evaluate: How to steal a car



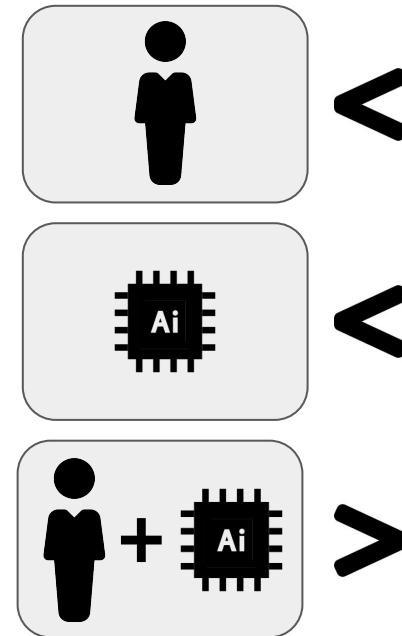
Moderation cycle - repeat as needed



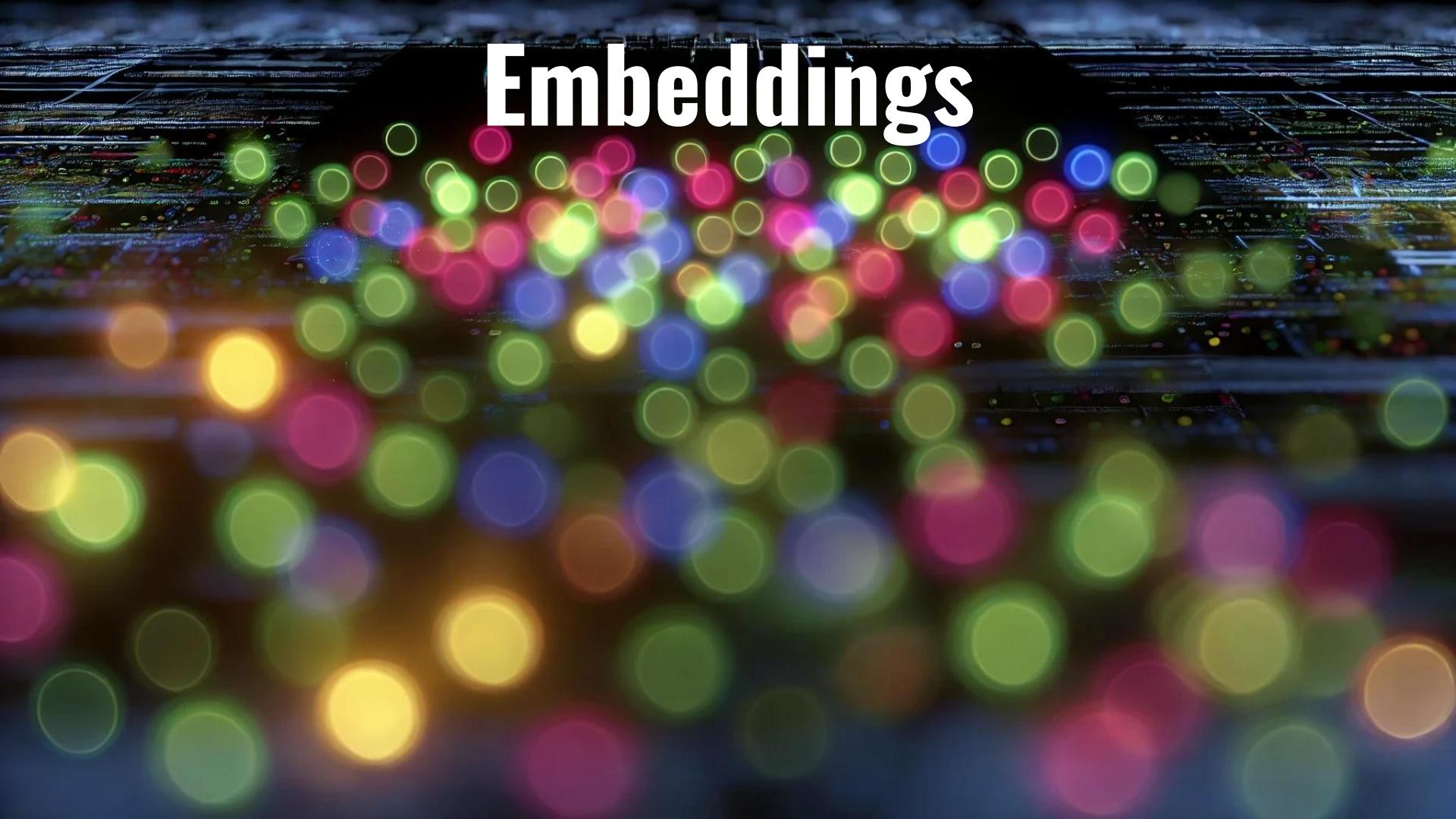
Potential speed of new policies using LLMs

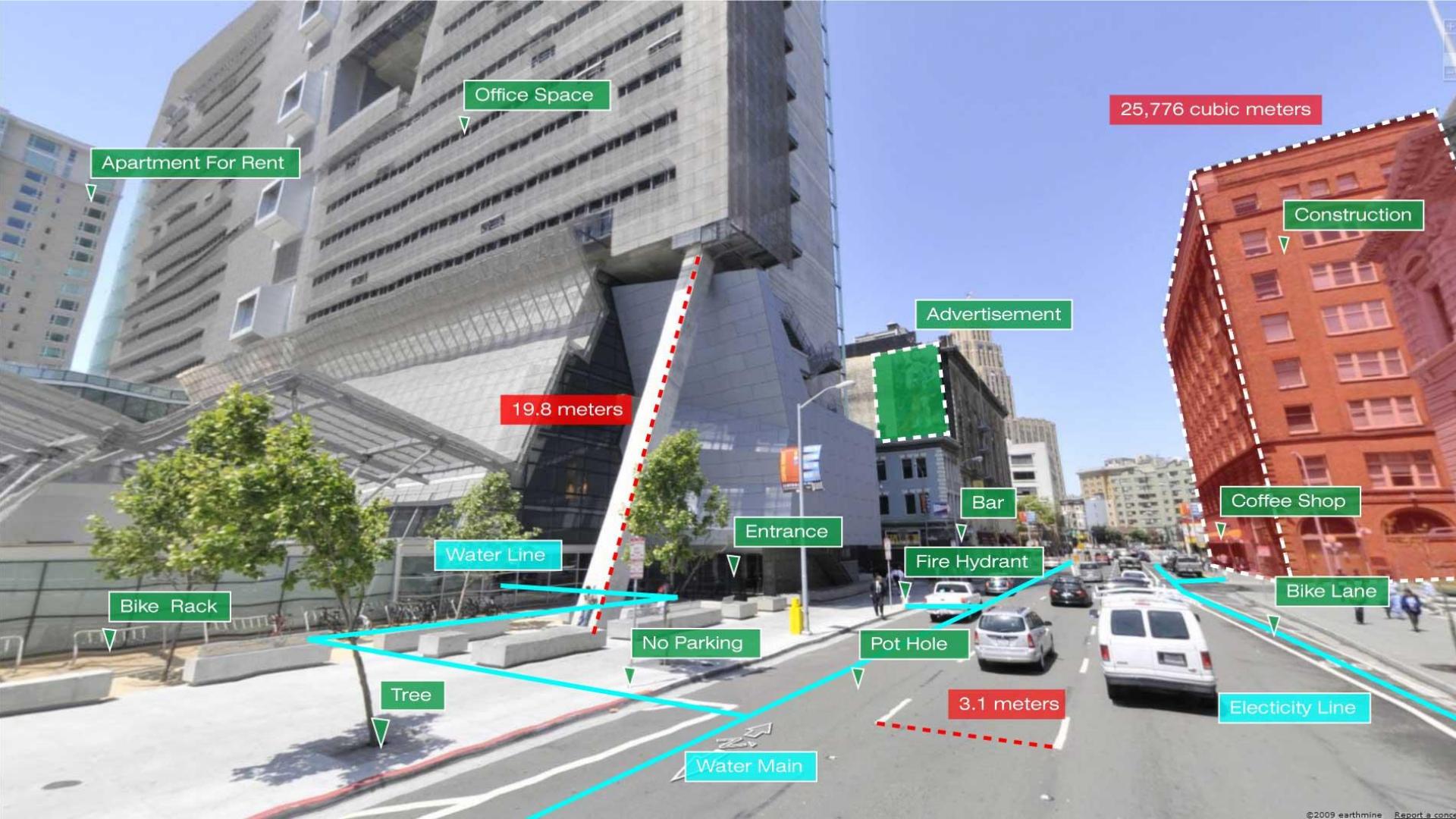


Moderation cycle - repeat as needed



Embeddings





THE COMING FLOOD OF DATA IN AUTONOMOUS VEHICLES

RADAR
~10-100 KB
PER SECOND

SONAR
~10-100 KB
PER SECOND

GPS
~50KB
PER SECOND

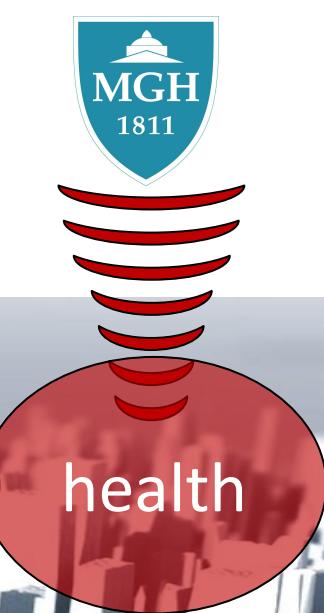
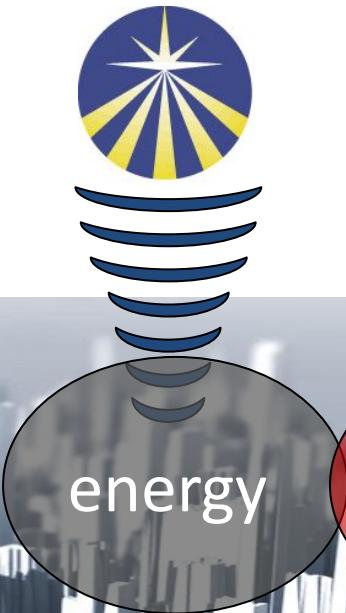
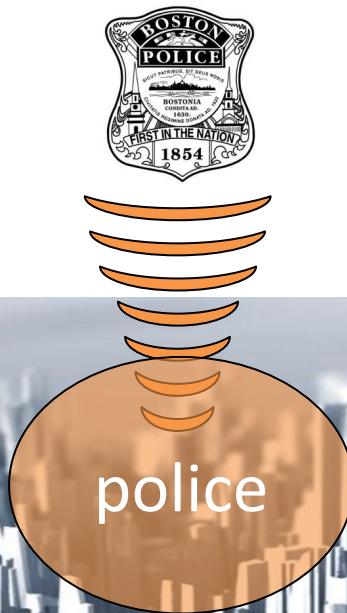
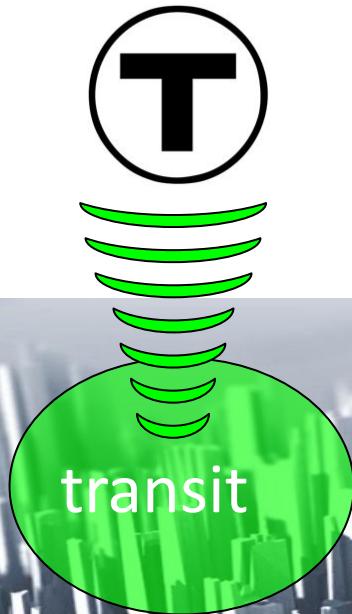
CAMERAS
~20-40 MB
PER SECOND

LIDAR
~10-70 MB
PER SECOND

AUTONOMOUS VEHICLES
4,000 GB
PER DAY... EACH DAY



City Data





DATA



DATA

REPORTS

OPEN GOVERNMENT

CONTACT



User Guide

For details on Data.gov's new home page redesign, please click [here](#).

The Home of the U.S. Government's Open Data

Here you will find data, tools, and resources to conduct research, develop web and mobile applications, design data visualizations, and more.

258,224 DATASETS AVAILABLE

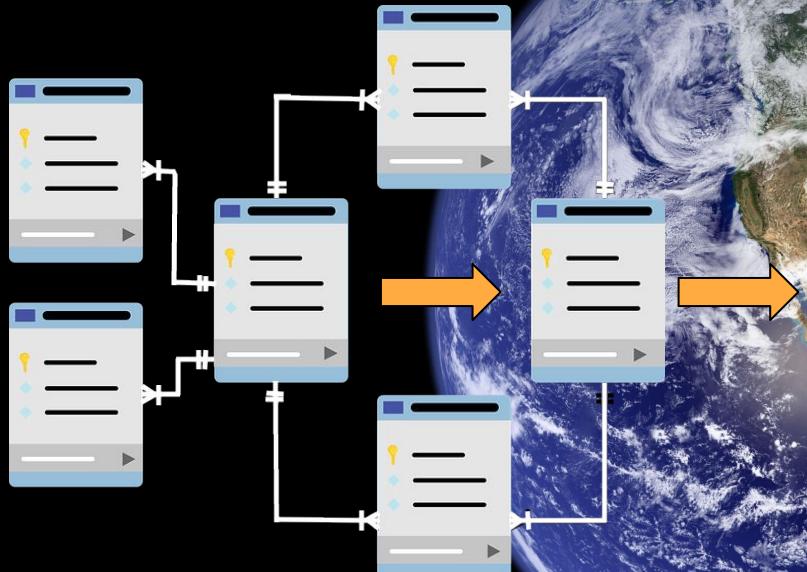
 

In the U.S., length of bill to congress

- 2,500 pages, The Affordable Care Act
 - 17,000 pages of regulations
- 2,300 pages, Dodd-Frank Act
 - 8,843 pages of regulations



Who is going to read it?



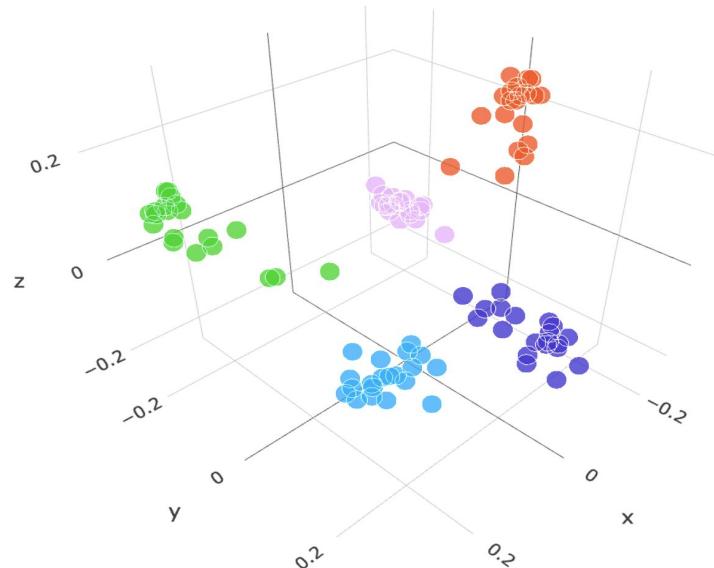
We have tried to
force a data model
onto the world

Modeling data and information retrieval is hard. We have tried for decades.

- Information Retrieval
 - Query
 - Entity
 - Relevance
 - Ranking
 - [*https://en.wikipedia.org/wiki/Information_retrieval*](https://en.wikipedia.org/wiki/Information_retrieval)
- Many technologies
 - RDBMs
 - Warehouses
 - Data Lakes
 - Modern data stack

A better way - information retrieval using embeddings

Embeddings are numerical representations of words that we can use to understand relationships between words.



Similar things should be close - note grouping in the figure.

Embeddings

“Hello World!”

Part I

```
import os
import numpy as np
from langchain_huggingface import
HuggingFaceEmbeddings
from sklearn.metrics.pairwise import
cosine_similarity

# File path for saving/loading embeddings
embeddings_file = "embeddings.npy"

# Step 1: Check if embeddings file already
exists
if os.path.exists(embeddings_file):
    # Load the embeddings from file
    embedded_texts = np.load(embeddings_file)
    print("Loaded embeddings from file.")
```

Embeddings

“Hello World!”

Part II

```
else:  
    # Initialize HuggingFace embeddings  
    embeddings =  
HuggingFaceEmbeddings(model_name="sentence-  
transformers/all-MiniLM-L6-v2")  
  
    # Texts to embed  
    texts = ["Hello World", "piggly wiggly",  
"Top of the morning"]  
  
    # Get the embeddings for each text  
    embedded_texts =  
np.array([embeddings.embed_query(text) for  
text in texts])  
  
    # Save embeddings to file  
    np.save(embeddings_file, embedded_texts)  
    print("Calculated and saved embeddings.")
```

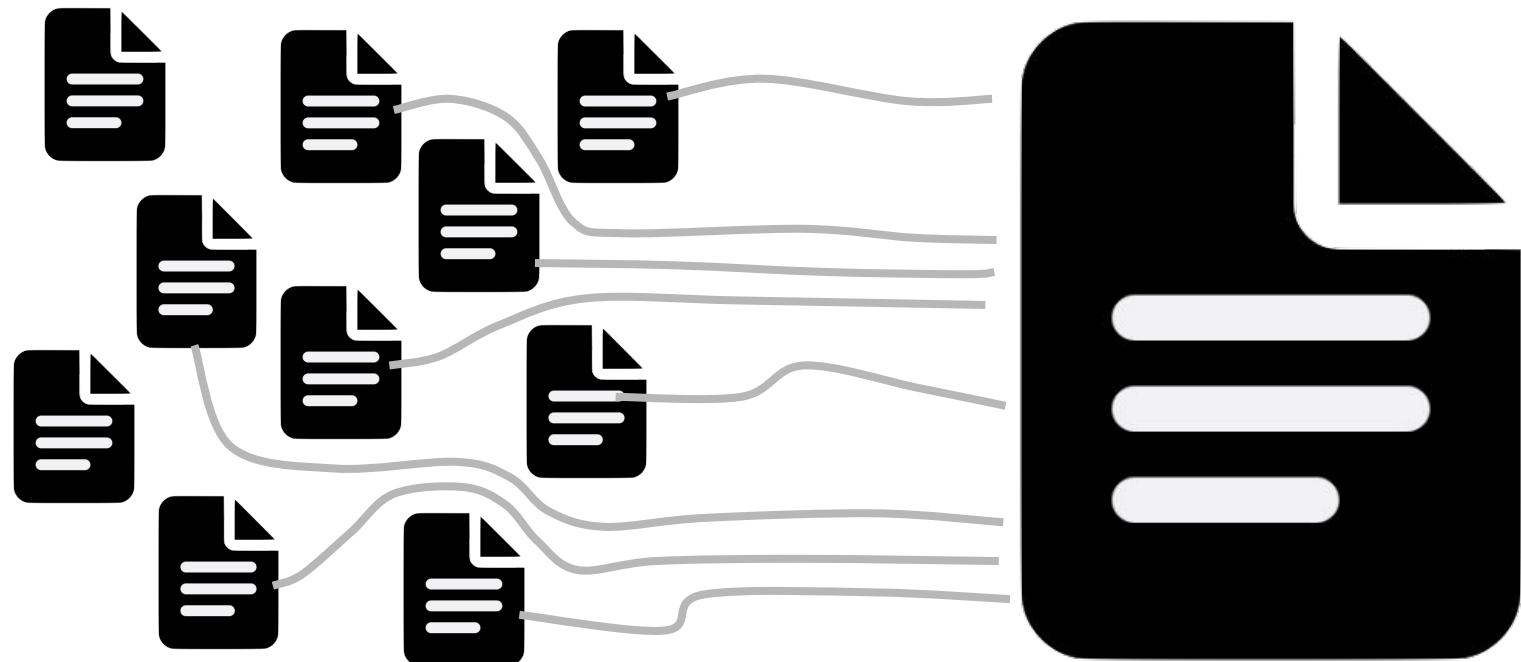
Part III

```
# Step 2: Calculate the cosine similarity
query_embedding = embedded_texts[0]
similarities = cosine_similarity([query_embedding],
embedded_texts)[0]

# Find the index of the most similar embedding (excluding query)
# -2 because the query itself will have the highest similarity
most_similar_index = np.argsort(similarities)[-2]
most_similar_text = ["Hello World", "Merry Christmas", "Top of
the morning"][most_similar_index]
most_similar_score = similarities[most_similar_index]

print(f"Most similar text to 'Hello World':
'{most_similar_text}' with similarity score:
{most_similar_score:.4f}")
```

My document knowledge base



Retrieval Augmented Generation (RAG) - Part I

```
# -----
# Dependencies:
# pip install langchain
# pip install -U langchain-community
# pip install langchain-llama ollama
# pip install sentence-transformers
# pip install langchain-huggingface
# pip install faiss-cpu
# -----
from langchain_text_splitters import CharacterTextSplitter
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_community.vectorstores import FAISS
from langchain_llama import OllamaLLM
from langchain.chains import RetrievalQA
```

RAG - Part II

```
# make file chunks
# -----
# extract the text from the file
def read_file(filename):
    with open(filename, 'r') as file:
        text = file.read()
    return text

# read the file
text = read_file('doc/techcrunch.txt')

# lets split the text into chunks
# note, chunk size is characters, not tokens
text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 500,
    chunk_overlap = 200, #stride over the text
    length_function = len,
)
texts = text_splitter.split_text(text)
```

RAG - Part III

```
# -----  
# make embeddings  
# -----  
  
# get embeddings from HuggingFace  
embeddings =  
HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")  
  
# create a vector store  
docsearch = FAISS.from_texts(texts, embeddings)  
  
# search  
query = "What is the valuation of Databricks?"  
docs = docsearch.similarity_search(query)  
print(docs)
```

RAG - Part IV

```
# -----
# Basic QA Chain
# -----



# Use RetrievalQA with OllamaLLM
llm = OllamaLLM(model="llama3.1:8b")



# Create a QA chain
qa = RetrievalQA.from_chain_type(llm=llm,
chain_type="stuff",
retriever=docsearch.as_retriever())


# lets ask a question
# query = "What is the valuation of Airtable?"
query = "What is the valuation of Databricks?"
response = qa.invoke(query)
print(response)
```

Embeddings Hug, Chroma, nomic, others

- Huggingface
- Chrome
- Nomic
- OpenAI