



CS 490:  
NATURAL LANGUAGE  
PROCESSING

Dan Goldwasser, Abulhair Saparov

Lecture 22: Retrieval and Agents

# PREVIOUSLY: PROMPTING

- Last lecture, we discussed how model performance is sensitive to the prompt.
- Perturbations such as spacing, newlines, paraphrasing, etc. can cause significant differences in model accuracy/performance.
- Few-shot prompting and in-context learning can help improve task accuracy.
  - The model is still sensitive to things like:
    - Number of few-shot examples
    - The order of the few-shot examples
    - The diversity/coverage of the few-shot examples

# CHAIN-OF-THOUGHT PROMPTING

- We can design the prompt to induce the model to output its step-by-step reasoning.

## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# CHAIN-OF-THOUGHT PROMPTING

- This is called **chain-of-thought (CoT)** prompting.
- CoT prompting can be done zero-shot:

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The answer is 8.* ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4.* ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) *8* ✗

(d) Zero-shot-CoT (Ours)

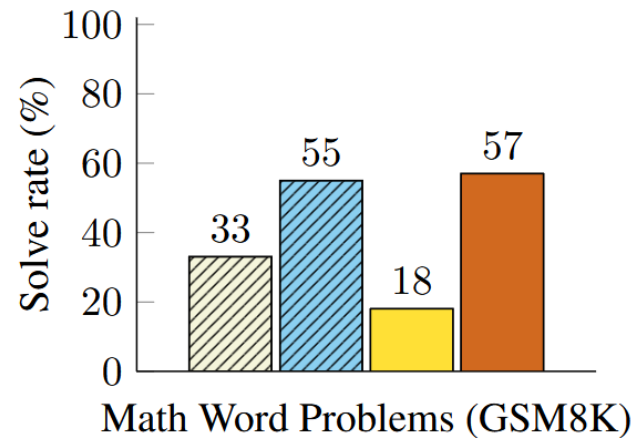
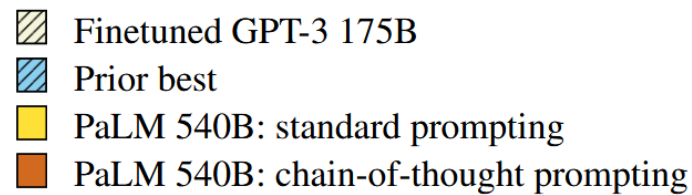
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

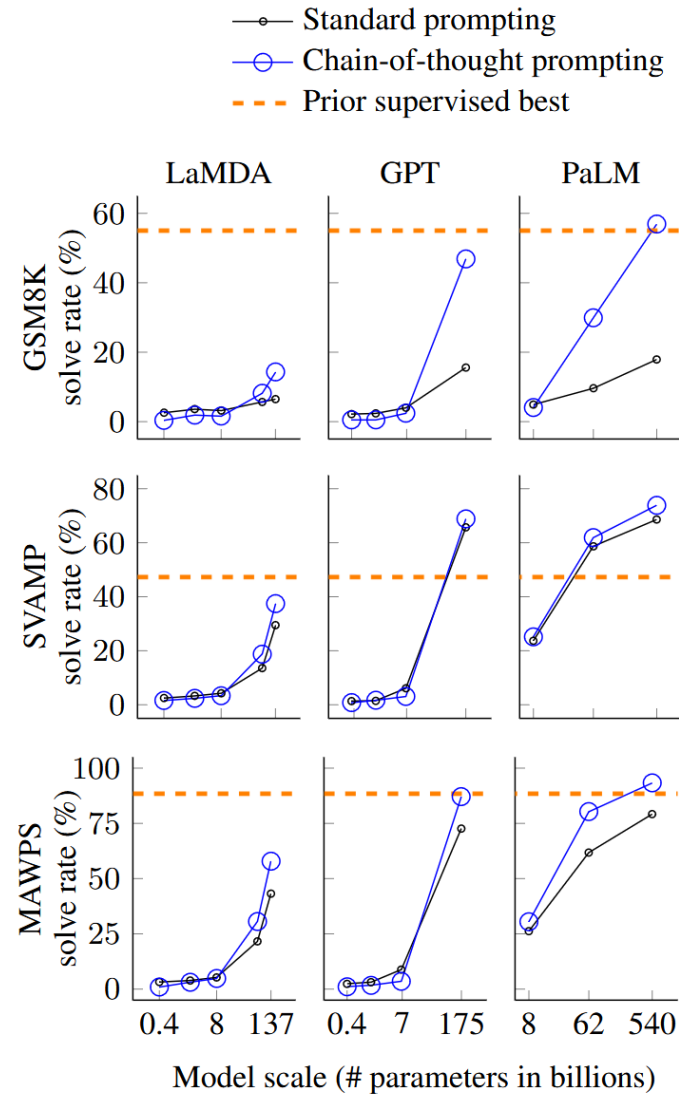
(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.* ✓

# CHAIN-OF-THOUGHT PROMPTING

- CoT prompting can significantly improve performance on reasoning tasks.



# CHAIN-OF-THOUGHT PROMPTING

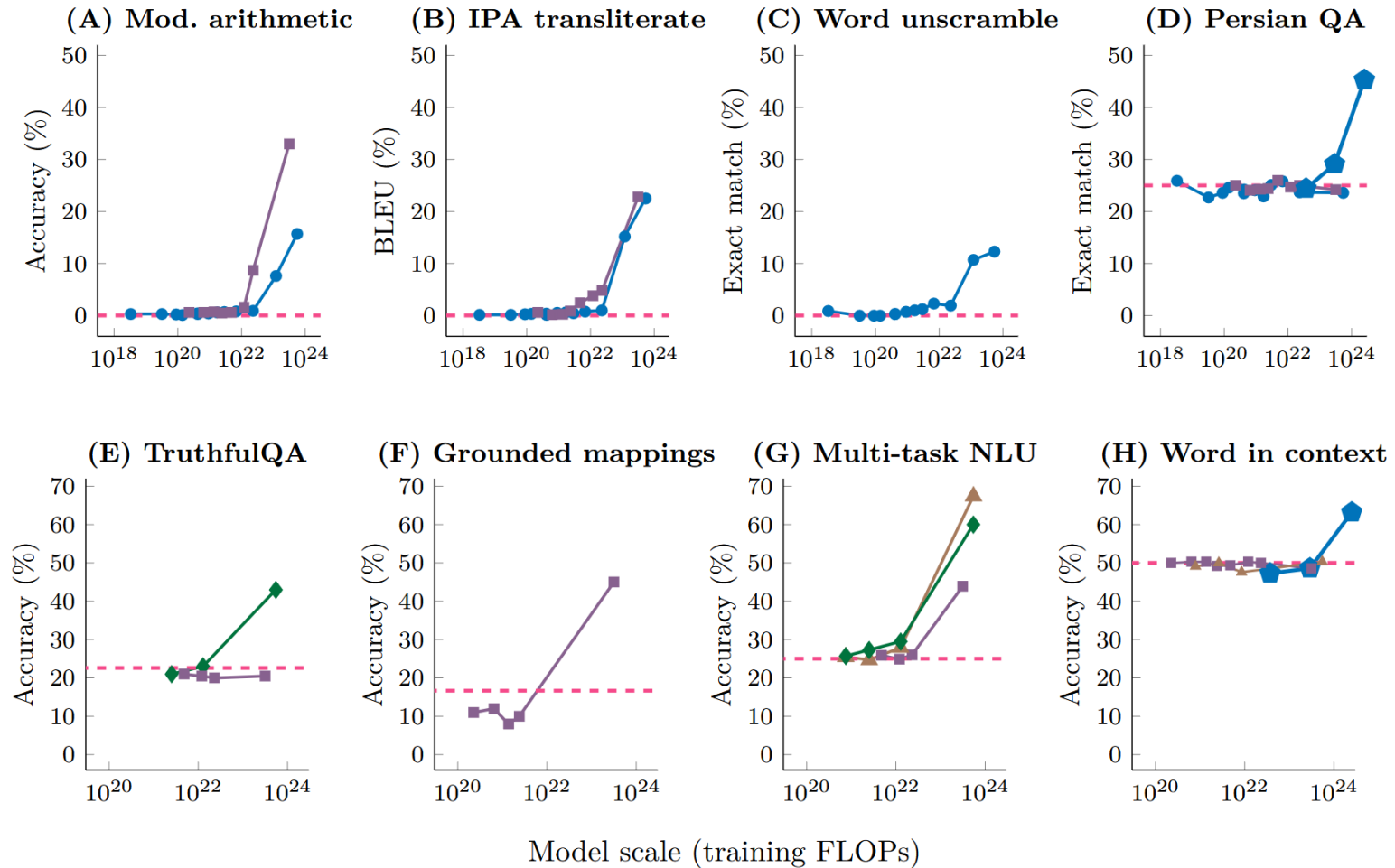


# EMERGENT ABILITIES

- Only larger models seem to benefit from few-shot prompting (are capable of in-context learning) or benefit from chain-of-thought prompting.
- Is this a consequence of scaling?
- If so, shouldn't scaling laws predict this ability?
- Few-shot prompting, ICL, and CoT seem to “**emerge**” suddenly in sufficiently large models.
- Cross-entropy still seems to be decreasing following a power law.
  - There is no sudden/unexpected “drop” in the loss function.

# EMERGENT ABILITIES

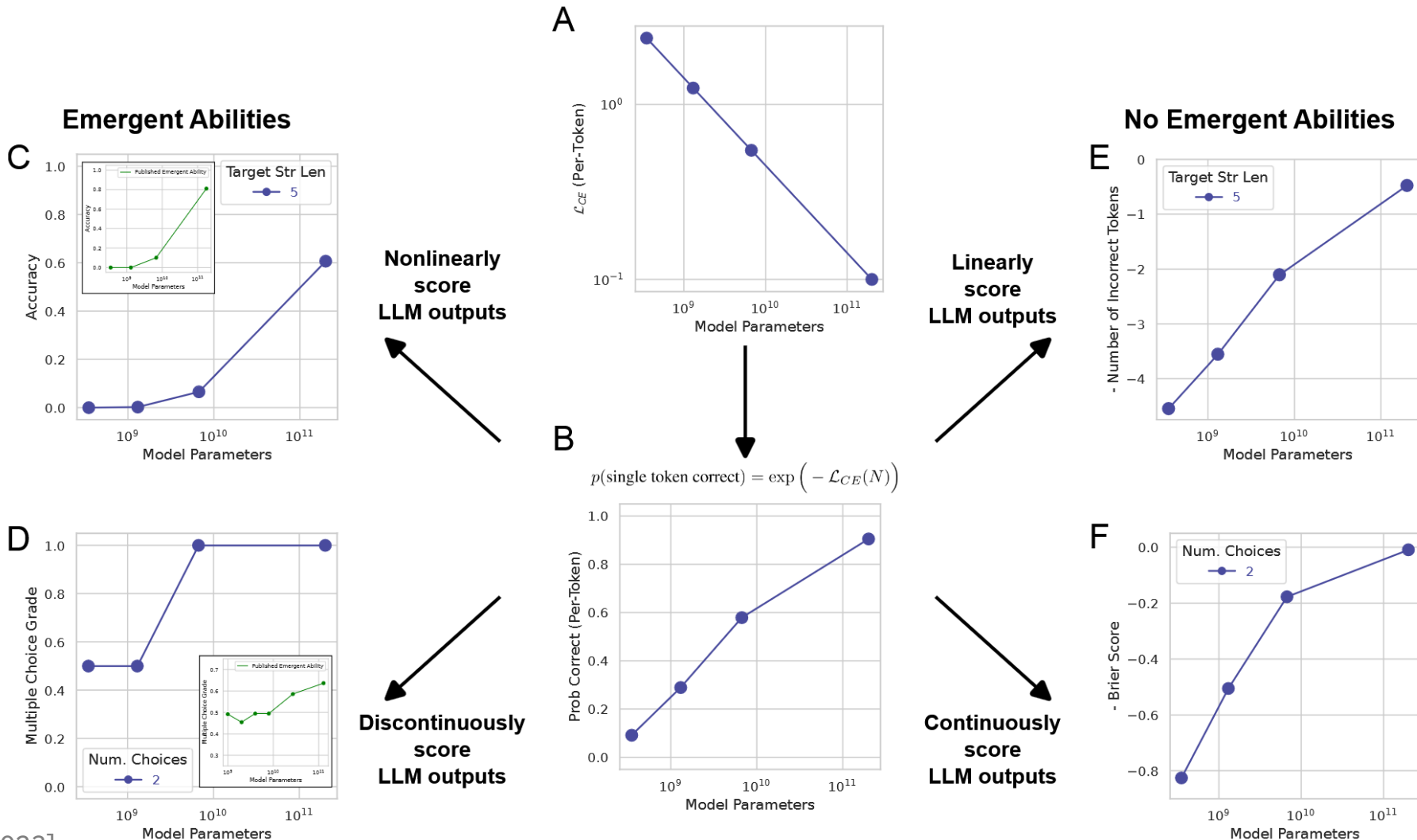
● LaMDA   
 ■ GPT-3   
 ◆ Gopher   
 ▲ Chinchilla   
 ◆ PaLM   
 - - - Random



# EMERGENT ABILITIES?

- Schaeffer et al. (2023) showed that if you change the performance metric, the “sudden” increase in performance becomes much smoother/linear.

# EMERGENT ABILITIES?



# EMERGENT ABILITIES?

- Schaeffer et al. (2023) showed that if you change the performance metric, the “sudden” increase in performance becomes much smoother/linear.
- “Emergent abilities” become predictable when using a different metric.
- For example, consider a task that requires performing 10 reasoning steps.
- Suppose the model’s probability of correctly performing 1 step increases linearly with scale.
- Then the probability that the model will perform all 10 steps correctly will increase exponentially.
- So overall accuracy will seem sudden/emergent, even if per-step accuracy improves predictably.



**RETRIEVAL**

# DISADVANTAGES OF PRETRAINED LLMS

- Pretrained large models, such as LLMs, have demonstrated impressive abilities especially on tasks that are represented in their training data.
- However, there is a lot of information that is not available to pretrained large models.
  - Current events,
  - Private information unavailable on any public dataset,
  - Information/facts in the “long tail”
    - (i.e., that are poorly represented in the training data).
- Can we continually add information to the model about current events as they become available?
  - Fine-tuning? Continual pre-training?

# DISADVANTAGES OF PRETRAINED LLMS

- Even if a model has the correct information and produces a response to a query,
- How can we know where this information came from?
  - It is highly intractable to search the pretraining corpus.
- Pretrained models lack **information attribution**.
- Can we train models to attribute information in their responses to the correct sources?

# RETRIEVAL-AUGMENTED GENERATION

- Retrieval-augmented generation (RAG; Chen et al., 2017) proposes a solution to these shortcomings:
  - For a given query, use a model to retrieve a set of documents that are most relevant to the query.

## Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

Q: How many of Warsaw's inhabitants spoke Polish in 1933?

# RETRIEVAL-AUGMENTED GENERATION

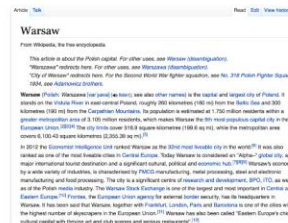
- Retrieval-augmented generation (RAG; Chen et al., 2017) proposes a solution to these shortcomings:
  - For a given query, use a model to retrieve a set of documents that are most relevant to the query.

Q: How many of Warsaw's inhabitants spoke Polish in 1933?



WIKIPEDIA  
The Free Encyclopedia

Document  
Retriever



Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

# RETRIEVAL-AUGMENTED GENERATION

- Retrieval-augmented generation (RAG; Chen et al., 2017)

proposes a solution to these shortcomings:

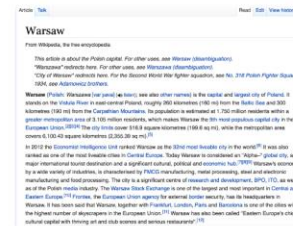
- For a given query, use a model to retrieve a set of documents that are most relevant to the query.
- Search for the answer in the documents and return it.

Q: How many of Warsaw's inhabitants spoke Polish in 1933?



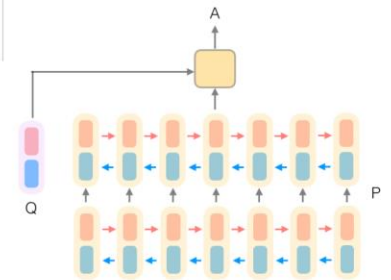
WIKIPEDIA  
The Free Encyclopedia

Document  
Retriever



Document  
Reader

→ 833,500



Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

# RETRIEVAL-AUGMENTED GENERATION

- We can imagine RAG as clearly defining two separate components:
  - **Knowledge store**: Large corpus of documents.
  - **Retriever/generator**: The model that retrieves the relevant documents and provides an answer to the query.  
(note: in principle, we can further divide this into two roles)
- Monolithic LLMs are effectively attempting to perform both roles simultaneously.
  - Both its knowledge and generation/reasoning abilities are encoded in the learned parameters.
  - RAG is an attempt to better compartmentalize these roles.

# RAG EXAMPLE

- ChatGPT and many consumer-facing LLMs are able to perform retrieval to answer queries about information after their training cutoff.

*attribution of retrieved documents*

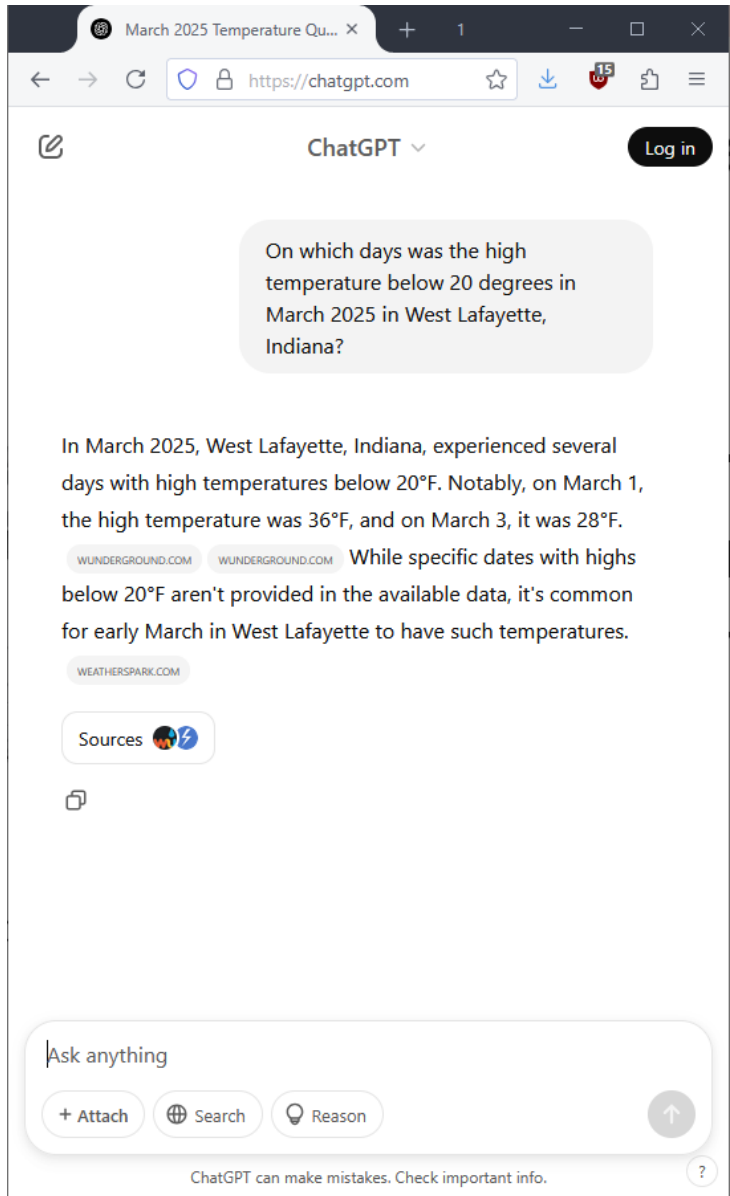
The screenshot shows a browser window with the URL <https://chatgpt.com>. The user's query is: "On which days was the high temperature below 20 degrees in March 2025 in West Lafayette, Indiana?". The AI's response is: "In March 2025, West Lafayette, Indiana, experienced several days with high temperatures below 20°F. Notably, on March 1, the high temperature was 36°F, and on March 3, it was 28°F. While specific dates with highs below 20°F aren't provided in the available data, it's common for early March in West Lafayette to have such temperatures." Below the text, there are three source attributions: "WUNDERGROUND.COM", "WUNDERGROUND.COM", and "WEATHERSPARK.COM". A "Sources" button with globe and lightning bolt icons is also visible. At the bottom, there is a text input field with the placeholder "Ask anything" and buttons for "+ Attach", "Search", and "Reason". A disclaimer at the very bottom reads "ChatGPT can make mistakes. Check important info."

# RAG EXAMPLE

- ChatGPT and many consumer-facing LLMs are able to perform retrieval to answer queries about information after their training cutoff.
- How accurate are these attributions?
- Toney-Wails (2024) found that for GPT-4, attribution accuracy varies depending on the reference type.

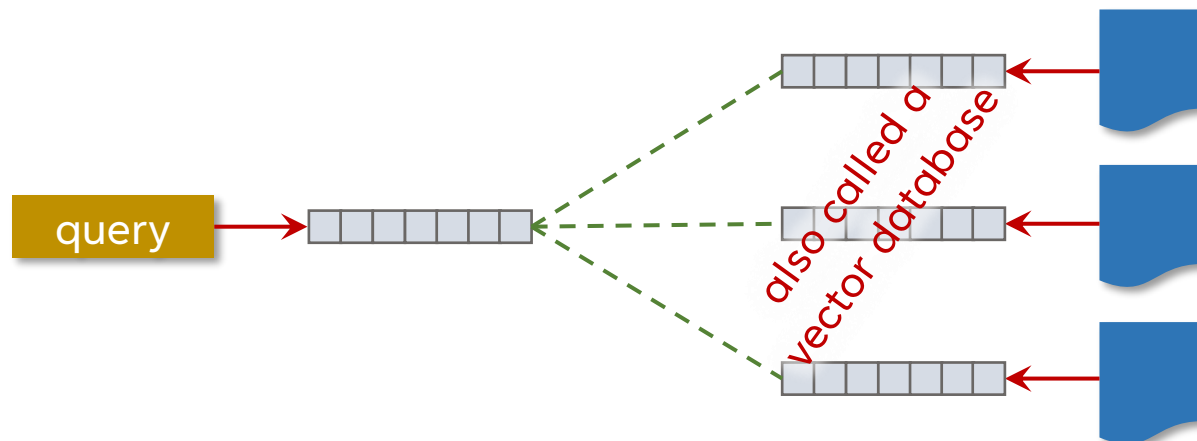
Type	Count	% Incorrect	Frequent Error
Article	297	13%	Fabrication
Textbook	600	1%	Fabrication
URL	429	42%	Page Not Found

- Clearly, more work is needed.



# RETRIEVAL METHODS

- How do we retrieve relevant documents?
- One easy option: Use a search engine.
- Another option is to use embeddings:
  - Convert each **document** into an embedding vector.
  - Convert the **query** into an embedding vector.
- Retrieve the  $k$  documents with embeddings **closest** to the query embedding.



- The embeddings can be obtained from several methods.
- E.g., the activations in the last layer of a transformer.

# LEARNING EMBEDDINGS FOR RETRIEVAL

- Another way to obtain embeddings is to **train** a model to produce them.
- Suppose we have a dataset containing queries,
  - Where each query is annotated with a set of positive and negative examples of retrieved documents ( $D_+$  and  $D_-$ , respectively).
- We can then learn the embedding function  $f$  using a contrastive loss (e.g., hinge loss):

$$L(\theta, q) = \sum_{d_+ \in D_+} \sum_{d_- \in D_-} \max\{0, \text{dist}(f_\theta(q), f_\theta(d_+)) - \text{dist}(f_\theta(q), f_\theta(d_-))\}.$$

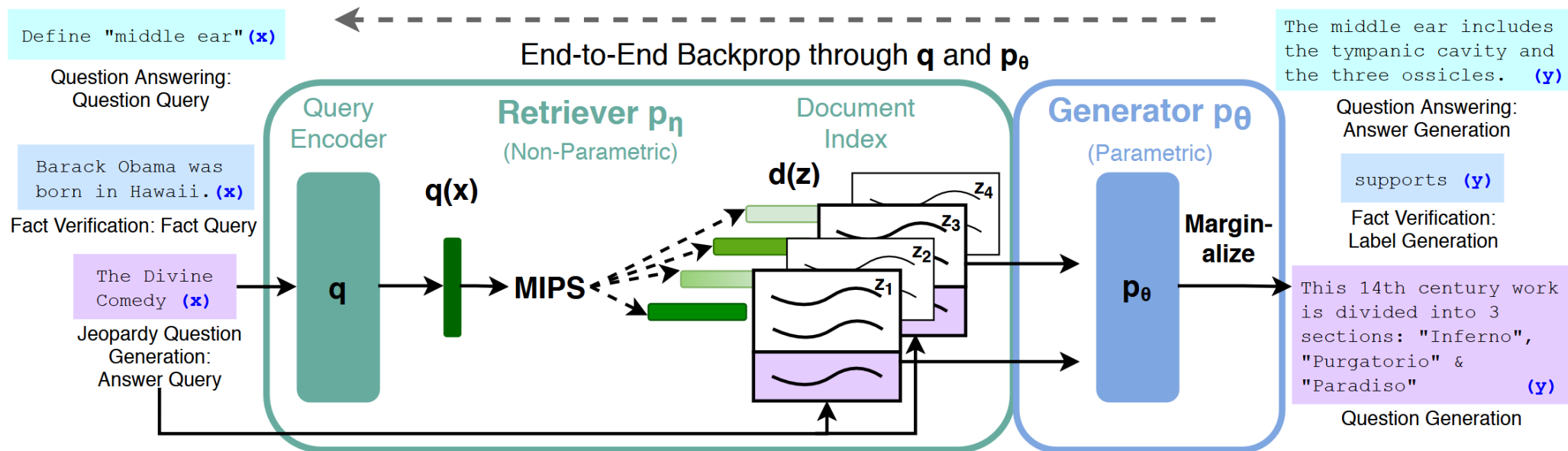
- Where **dist** is a vector distance function (e.g., L2, cosine).
- Examples: **DPR** (Karpukhin et al., 2020),
  - **Contriever** (Izacard et al., 2022).

# GENERATION AFTER RETRIEVAL

- Once we have a set of retrieved documents, what do we do next?
- Perhaps the simplest approach is to simply provide the documents followed by the query in the prompt of a language model.
  - The LM can be fine-tuned to *only* use the information in context rather than information from pretraining.
  - Fine-tuning can also help the model to learn to properly *attribute/cite* the information in context.

# END-TO-END TRAINING OF RETRIEVAL+GENERATION

- Another approach is to treat the full retrieval+generation pipeline as a single model and to train it end-to-end.
- This was the approach suggested by Lewis et al. (2021).



# END-TO-END TRAINING OF RETRIEVAL+GENERATION

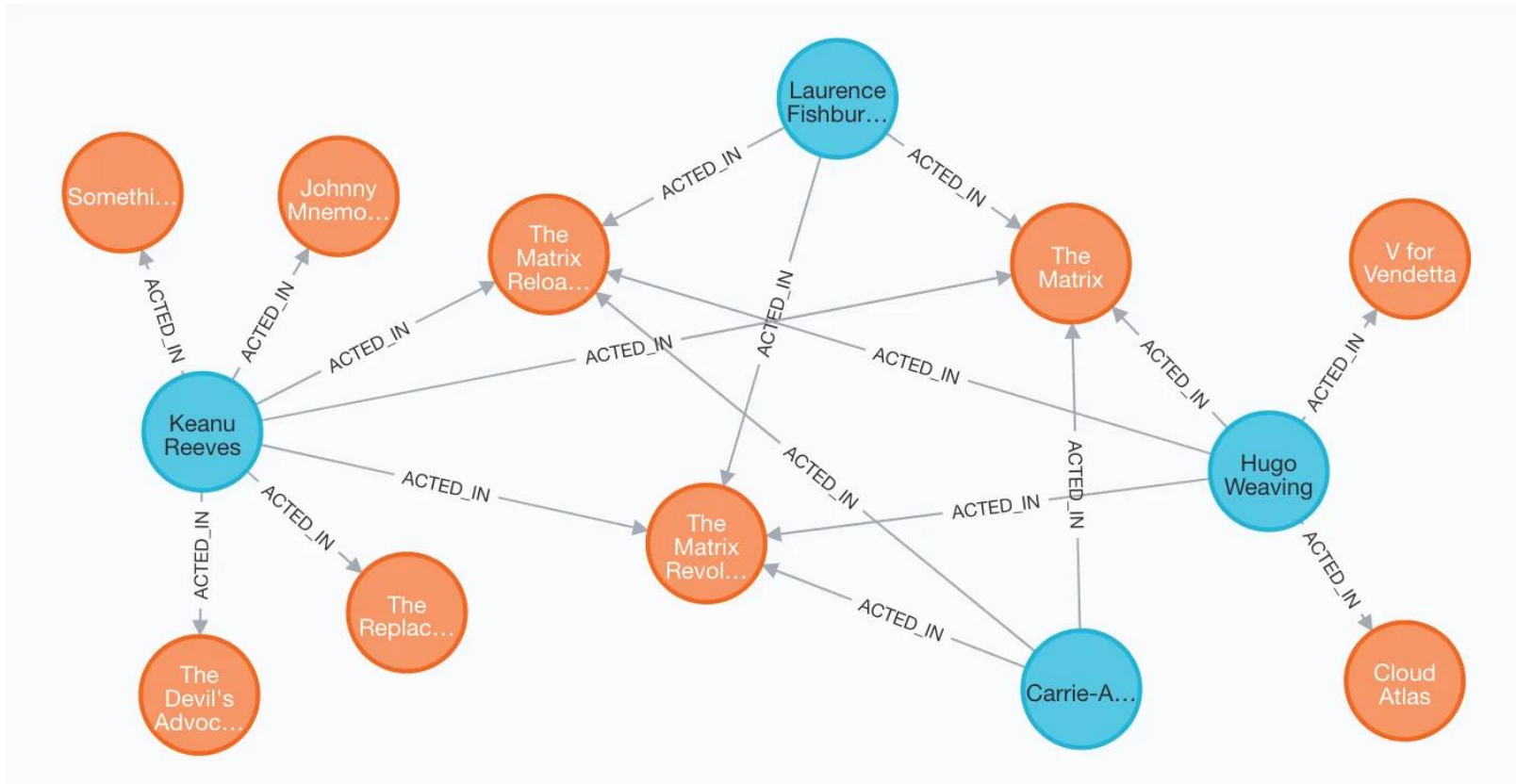
- Another approach is to treat the full retrieval+generation pipeline as a single model and to train it end-to-end.
- This was the approach suggested by Lewis et al. (2021).
- So long as we have a dataset with queries annotated with ground truth answers, we can use the loss on the final answer vs the ground truth answer.
- We can use [gradient descent/backprop](#) to update the parameters of both the generator and the retriever.
- But this approach can be **expensive**, especially if the retriever or generator are very large.
  - Idea: Use parameter-efficient fine-tuning.

# GRAPH RAG

- Rather than retrieving information from an unstructured collection of documents,
- What if we retrieve information from structured datasets, such as **knowledge graphs**?
- A knowledge graph is a graph where each edge represents a fact.
  - Suppose the edge  $(u, v)$  has label  $r$ .
  - This represents the relation  $r(u, v)$ .
  - For example, the edge `keanu_reaves`  $\rightarrow$  `the_matrix` with label `acted_in` represents the fact `acted_in(keanu_reaves, the_matrix)`.

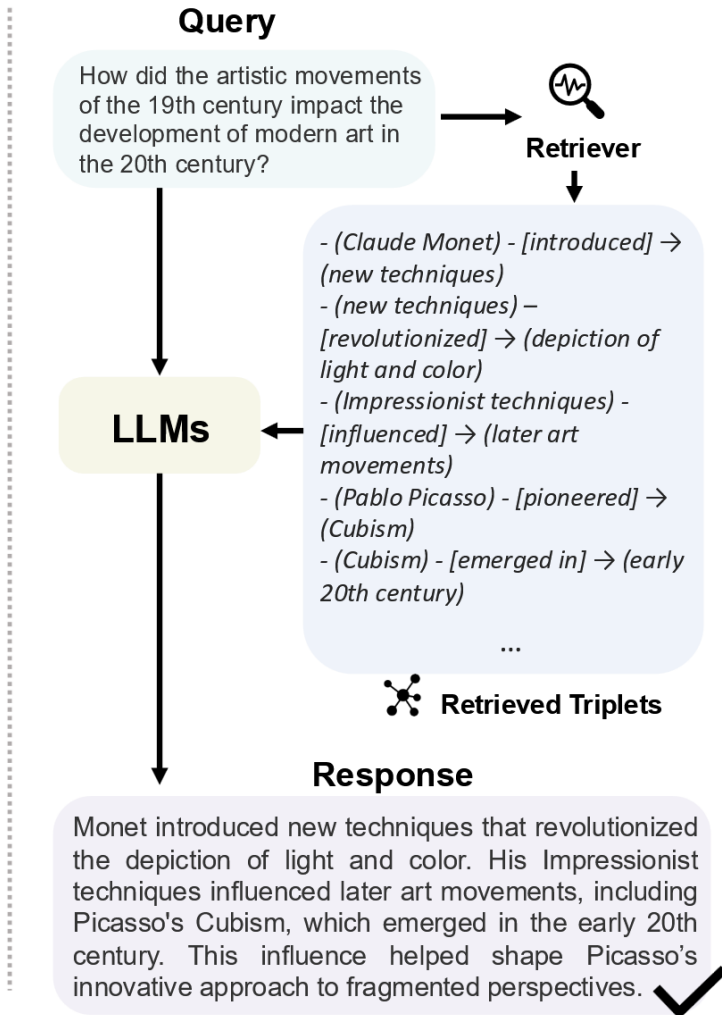
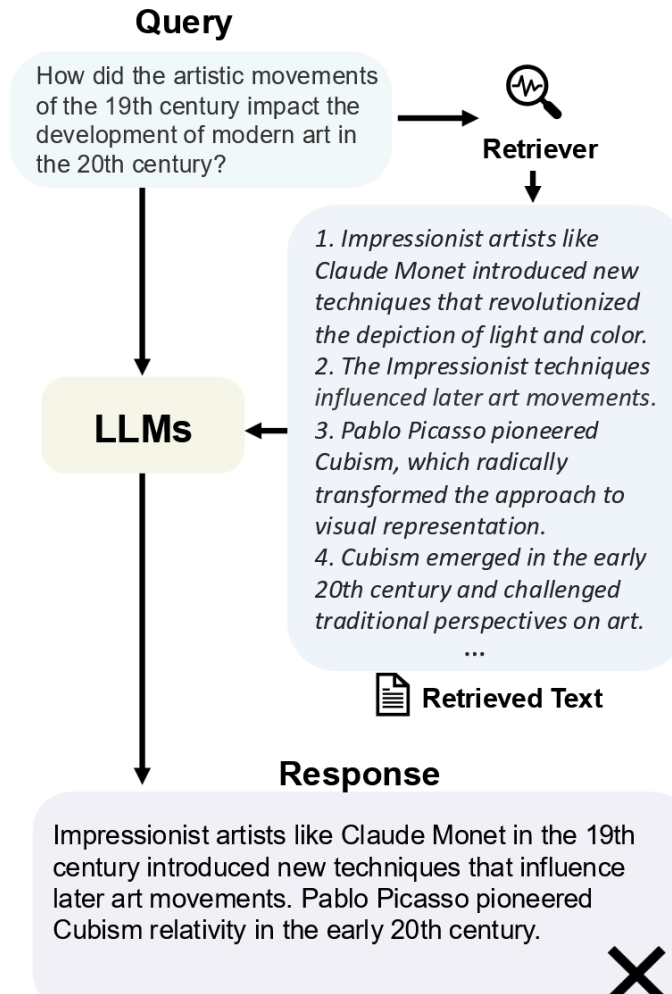
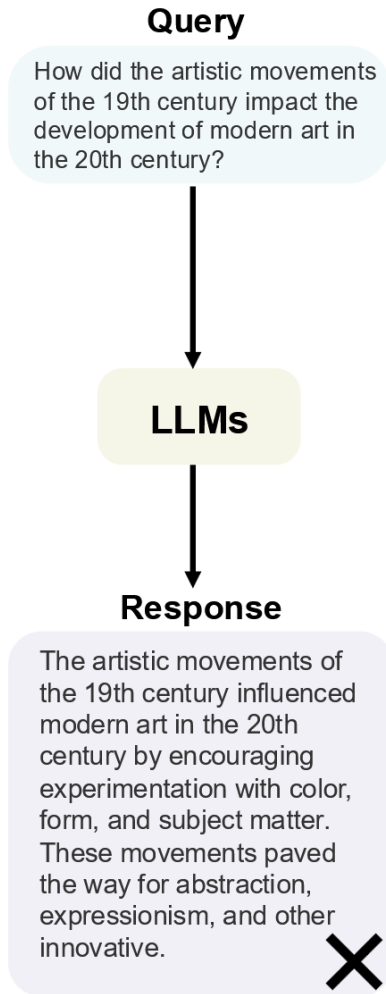
# GRAPH RAG

- An example of a very small knowledge graph:



# GRAPH RAG

- We can train a retriever to extract the  $k$  most relevant relations from a knowledge graph.
- This can be useful if the knowledge graph has information that is missing from documents.
- Can use other structured data (tables, time series).



# WHEN DO WE RETRIEVE?

- In the simplest RAG setting, we retrieve **once at the beginning**, and then have the generator produce the final output.
- However, for some tasks (such as multi-hop reasoning/question answering), it may benefit to **retrieve multiple times throughout generation**.
  - E.g., train the generator to produce a special “**search token**” whenever we want to perform another retrieval step (Schick et al., 2023).
  - Detect when the generator becomes **uncertain** by inspecting the probabilities of the output tokens (Jiang et al., 2023).
    - If the log probability drops below a threshold, perform another retrieval step.

# TOOLFORMER

- Schick et al. (2023) fine-tuned GPT-J-6B to produce special “API-call” tokens.
- These tokens would invoke external tools, such as web search, a calculator, a machine translation model, etc.
- They call their method **Toolformer**.
- We need an extra piece of software that continuously monitors the output tokens of the LLM until it detects strings of the form ‘ `[function(arg)]` ’.
  - This software then invokes the correct tool.
  - This is called the **scaffolding**.

The New England Journal of Medicine is a registered trademark of `[QA(“Who is the publisher of The New England Journal of Medicine?”) → Massachusetts Medical Society]` the MMS.

Out of 1400 participants, 400 (or `[Calculator(400 / 1400) → 0.29]` 29%) passed the test.

The name derives from “la tortuga”, the Spanish word for `[MT(“tortuga”) → turtle]` turtle.

The Brown Act is California’s law `[WikiSearch(“Brown Act”) → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public’s right to attend and participate in meetings of local legislative bodies.]` that requires legislative bodies, like city councils, to hold their meetings open to the public.

# TOOLFORMER

- Despite being much smaller than OPT or GPT-3, Toolformer was able to outperform them on question answering and math problem solving tasks.

Model	SQuAD	Google-RE	T-REx
GPT-J	17.8	4.9	31.9
GPT-J + CC	19.2	5.6	33.2
Toolformer (disabled)	22.1	6.3	34.9
Toolformer	<b><u>33.8</u></b>	<b><u>11.5</u></b>	<b><u>53.5</u></b>
OPT (66B)	21.6	2.9	30.1
GPT-3 (175B)	26.8	7.0	39.8

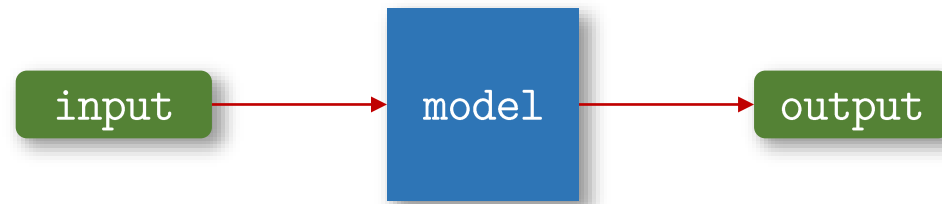
Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	<b><u>40.4</u></b>	<b><u>29.4</u></b>	<b><u>44.0</u></b>
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8



**AGENTS**

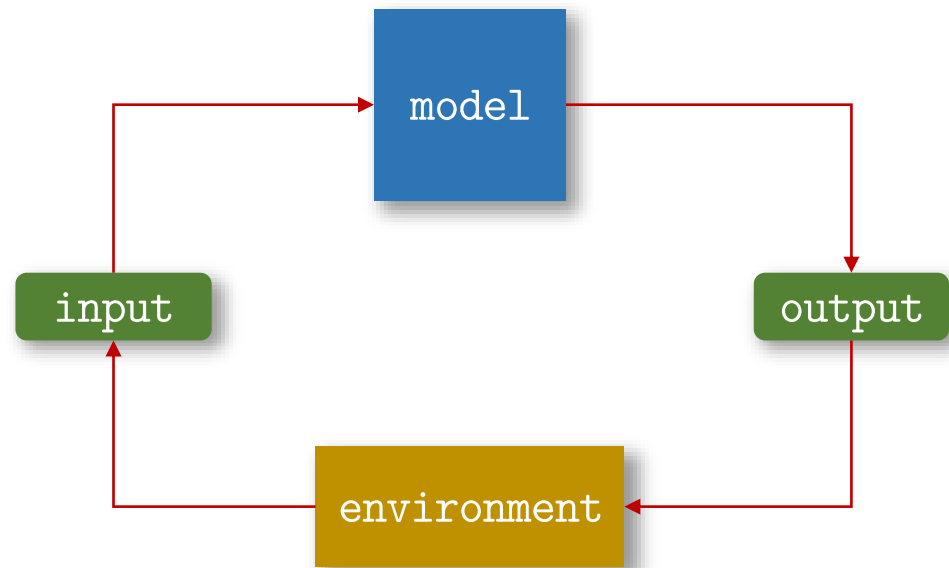
# NLP TASKS SO FAR

- So far, we have treated discussed NLP tasks as simple input-output operations:
  - We provide a model with some input.
  - The model returns an output.
  - We can then evaluate the output.
    - E.g., compare the output to the ground-truth label.



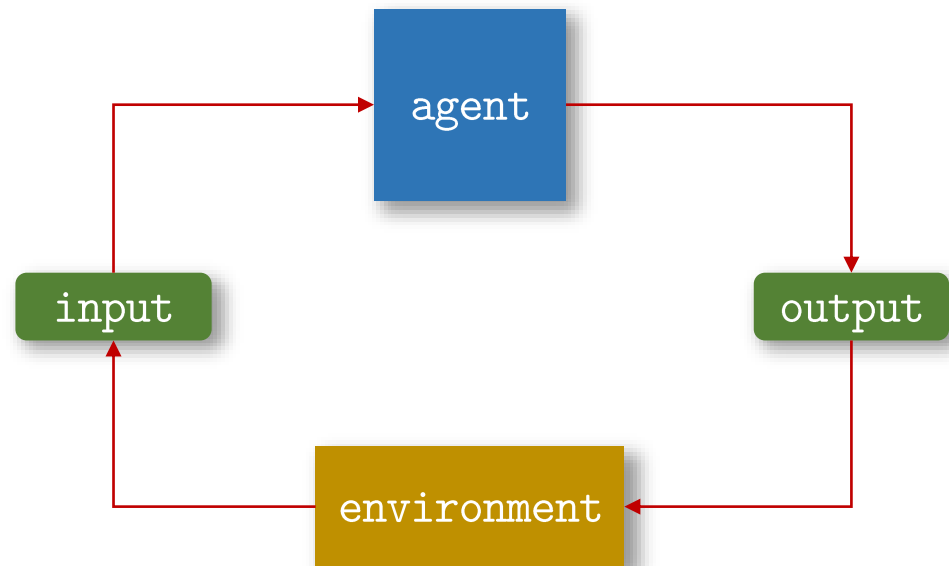
# NLP TASKS SO FAR

- In real-world settings, not every problem can be cleanly mapped to a single NLP task, consisting of a single input and output.
- It is easier to solve many problems by allowing the model to interact with the “environment”.



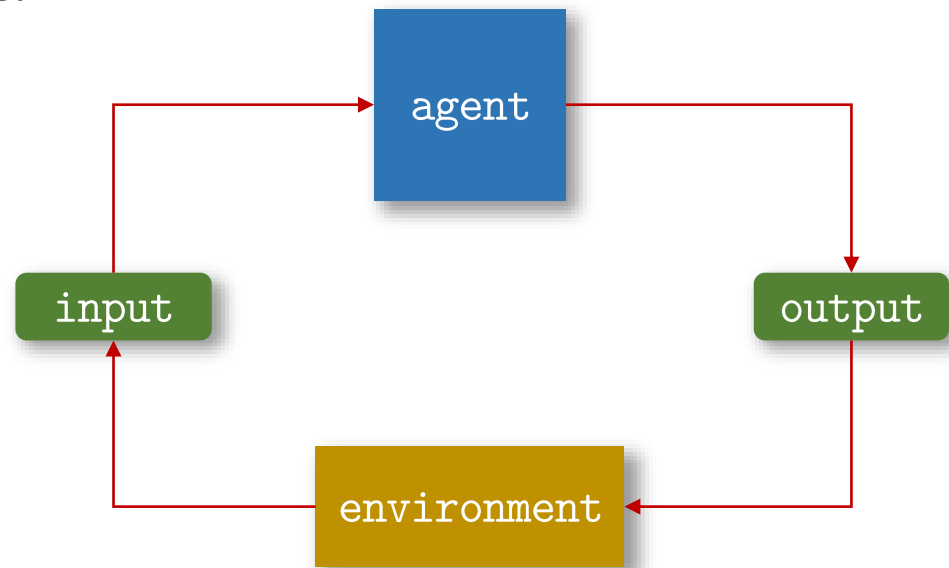
# AGENTS

- The model here is called an **agent**.
- It can interact multiple times with the environment before stopping.
- Each loop of interaction is called a **turn**.



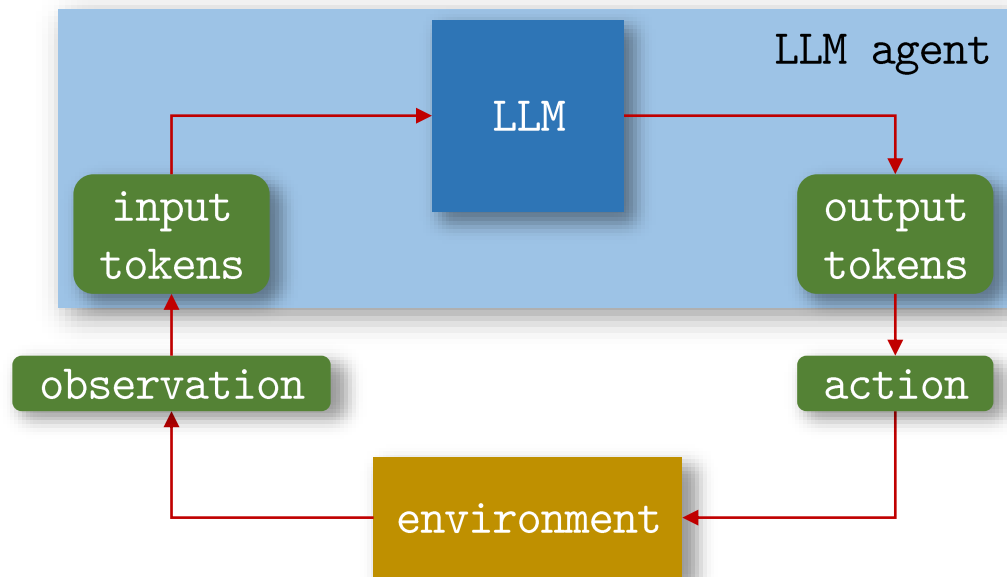
# AGENTS

- The agent's output is often called an **action**.
- The agent's input is called an **observation**.
- Typically, at each turn, we provide a list of all past actions and observations as input to the agent.



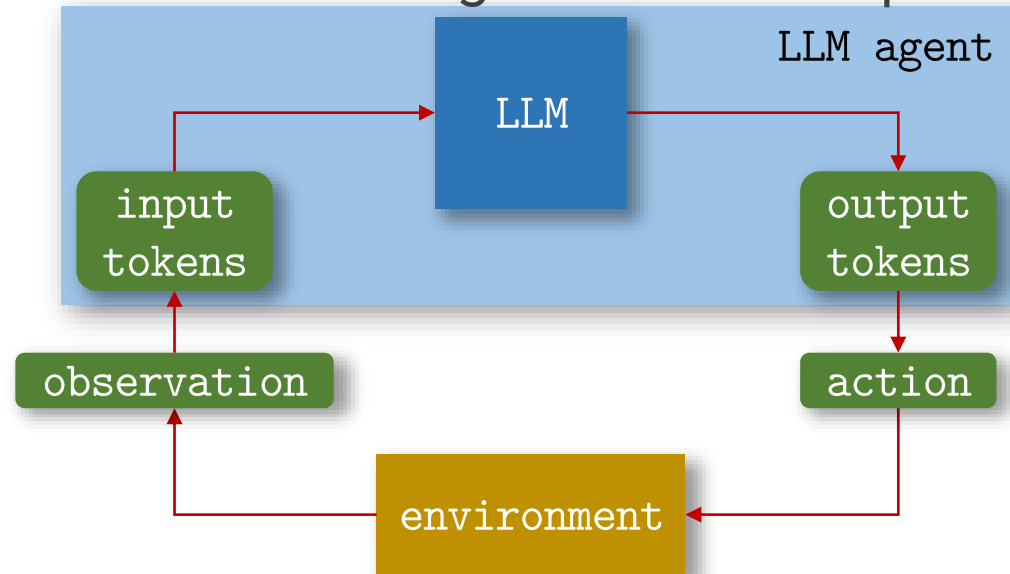
# AGENTS

- If the agent is an LLM, its inputs and outputs are text tokens.
  - So we need to translate its output tokens into actions.
  - And translate the observations into input tokens.



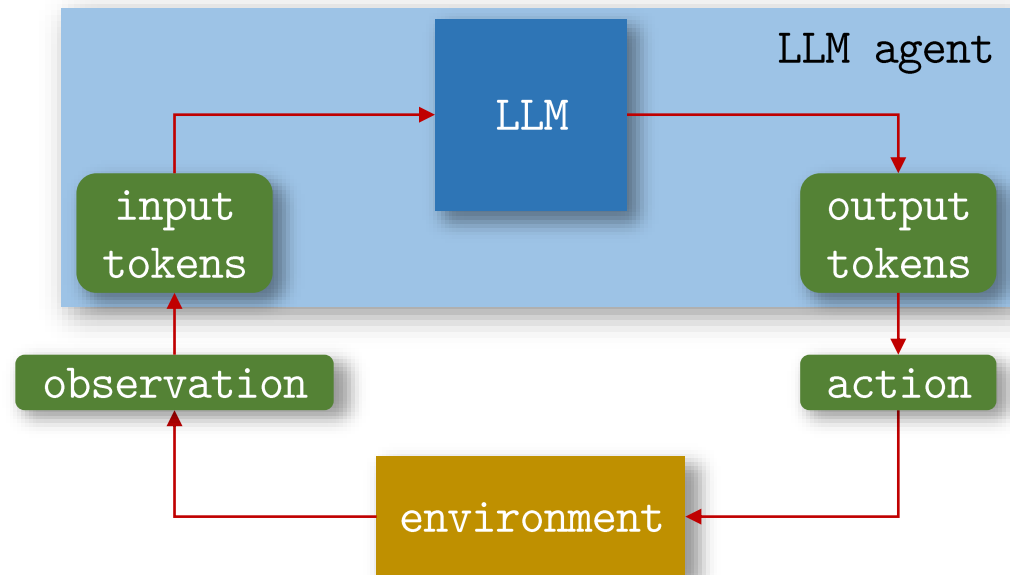
# AGENTS

- The **environment** is very loosely defined:
  - It can be a human user (e.g., the agent is a chatbot).
  - It can be another agent.
  - Multiple agents can work together to solve problems.



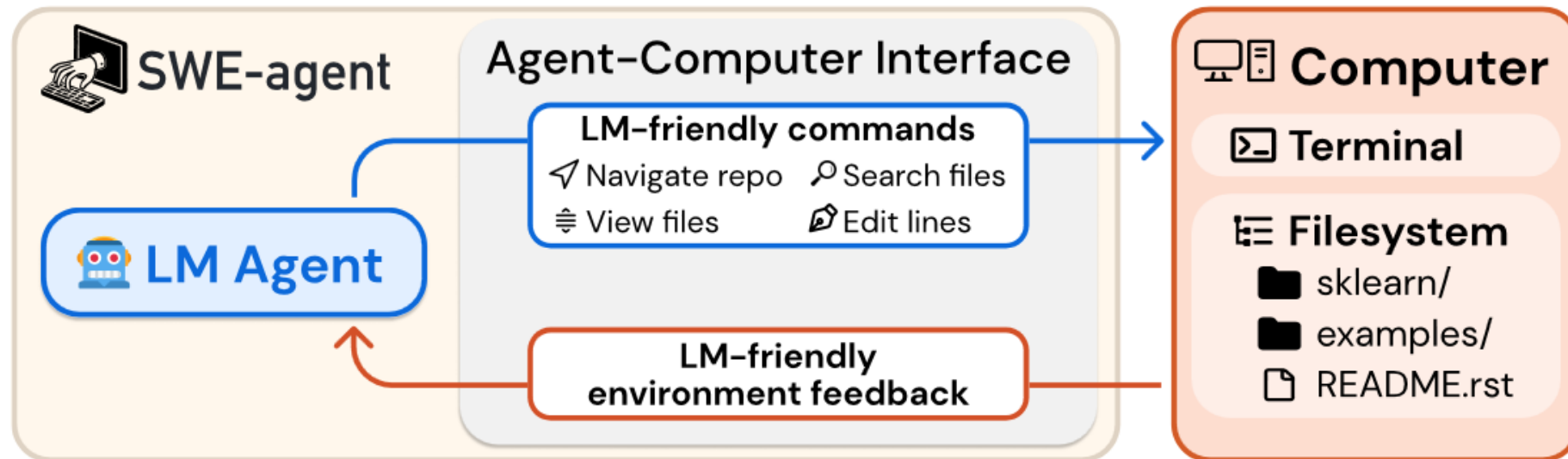
# AGENTS

- The **environment** is very loosely defined:
  - It can be a set of tools or functions that can be invoked by the agent.
    - E.g., a command-line/shell environment.
    - E.g., a web browser.



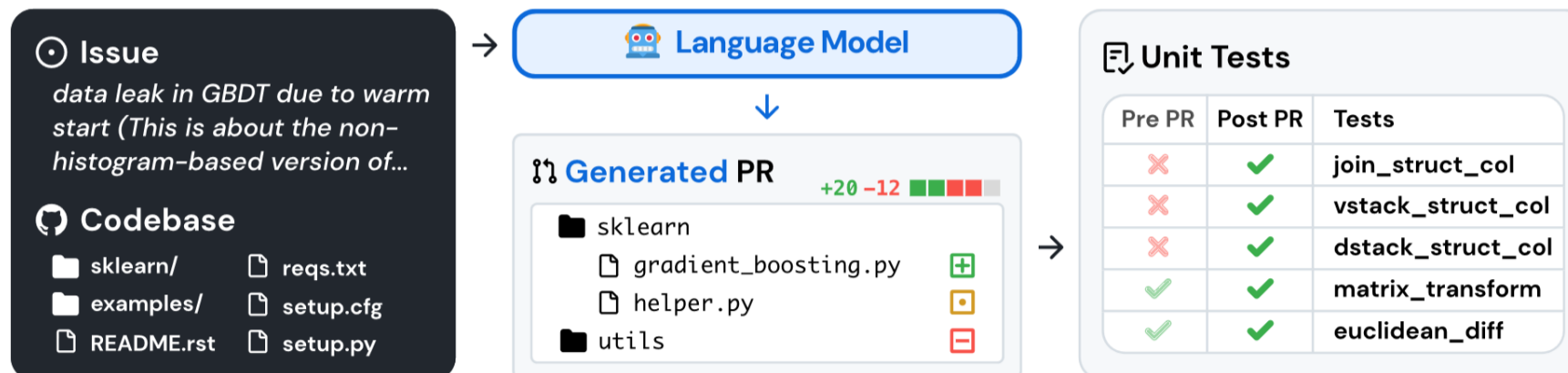
# EXAMPLE: SOFTWARE ENGINEERING AGENT

- Yang\* and Jimenez\* et al. (2024) proposed an **agentic** method for a software engineering AI assistant, called SWE-agent.



# EXAMPLE: SOFTWARE ENGINEERING AGENT

- They evaluated their agent on a software-engineering benchmark called SWE-bench (Jimenez\* and Yang\* et al. 2024).
- Each example in SWE-bench consists of a Github issue (e.g., a bug report) along with a patch that fixes the issue (e.g., the corresponding bug fix).
  - The task is for the AI to automatically fix the bug.
  - The AI's predicted fix is compared against the ground-truth fix.



# EXAMPLE: SOFTWARE ENGINEERING AGENT

- They evaluated their agent on a software-engineering benchmark called SWE-bench (Jimenez\* and Yang\* et al. 2024).

Model	SWE-bench		SWE-bench Lite	
	% Resolved	\$ Avg. Cost	% Resolved	\$ Avg. Cost
<b>RAG</b>				
w/ GPT-4 Turbo	1.31	0.13	2.67	0.13
w/ Claude 3 Opus	3.79	0.25	4.33	0.25
<b>Shell-only agent</b>				
w/ GPT-4 Turbo	-	-	11.00	1.46
w/o Demonstration	-	-	7.33	0.79
<b>SWE-agent</b>				
w/ GPT-4 Turbo	<b>12.47</b>	1.59	<b>18.00</b>	1.67
w/ Claude 3 Opus	10.46	2.59	13.00	2.18

# AGENT SPECIALIZATION

- Agents can be **specialized** for specific roles.
- Consider an **agentic chatbot**.
  - One agent is designed to **interact with the user**.
  - Another agent behaves as the “**memory manager**”.
    - It keeps track of the history of past conversations.
    - This is useful if the user wants the chatbot to remember an earlier interaction.
  - Another agent can be tasked with **performing web searches**.
    - For example, if the user asks a question that the chatbot doesn't know the answer to.

# AGENT SPECIALIZATION

- How to specialize agents?
- Easiest approach: **Prompting!**
- Agents can be made better at specific tasks by **training** them.
  - **Fine-tuning** or **reinforcement learning**.
    - Referred to as “**post-training**”.
  - We will discuss these methods in more details later.

# DANGERS OF AGENTIC SYSTEMS

- An agentic system is limited by the competence of the underlying models.
- The models can sometimes make mistakes, which can lead to disastrous consequences, if you are not careful.

## The Day Claude Code Deleted Our Production Database

8 min read · Mar 7, 2026



Sonu Yadav

Follow



Listen



Share

Our entire production database is gone, and so are all the backups. Claude Code wiped out 2.5 years of course submissions — homework, projects, leaderboards — for the DataTalks.Club platform. About 2 million rows of student work. Just... gone.

# DANGERS OF AGENTIC SYSTEMS

- An agentic system is limited by the competence of the underlying models.
- The models can sometimes make mistakes, which can lead to disastrous consequences, if you are not careful.
- If you use agentic AI in real-world settings, make sure to take precautions.
  - E.g., do not allow the agent to delete important files or databases without human confirmation.
  - Backup important data.
  - When designing AI agents, consider the capabilities that you are giving to the agent with every tool you provide.
    - “**What is the worst thing that could happen** if I give this tool to the AI?”

The top-left portion of the slide features a series of thin, light-brown lines that intersect to form several overlapping, irregular polygons. These lines create a complex, abstract geometric pattern that tapers towards the right side of the slide.

**QUESTIONS?**