

# Retrieval and Tool Use

## RAG + Agents

Aaron Mueller

CAS CS 505: Introduction to Natural Language Processing

Spring 2026

Boston University

[Many slides inspired by lectures by Graham Neubig.]

# Admin

- The midway report is due in one week, on **Apr. 21** at 11:59pm.
  - You are allowed to use any free late days you have remaining on this assignment.
  - Your report should contain the sections in the project specs.
  - If you're doing an empirical project, you should have at least tried to run some experiments. It's ok if you're still debugging some experiments, but be specific about what kinds of issues you're running into and how you plan to address them. (We can try to help in our feedback on Gradescope or in OH.)
- If you're using the shared cache, please add this to your `.bashrc`: ``umask 007``.
  - (Or just use your own cache folder, but this may lead to disk space issues if everyone does it.)
  - We're currently at about 25% usage—we'll see how things go as the deadline approaches.

# Outline

1. Retrieval
2. What's an agent?
3. How do agents know when to use the right tools?

# Prompting

- In normal prompting, we write a template and put an example into it:

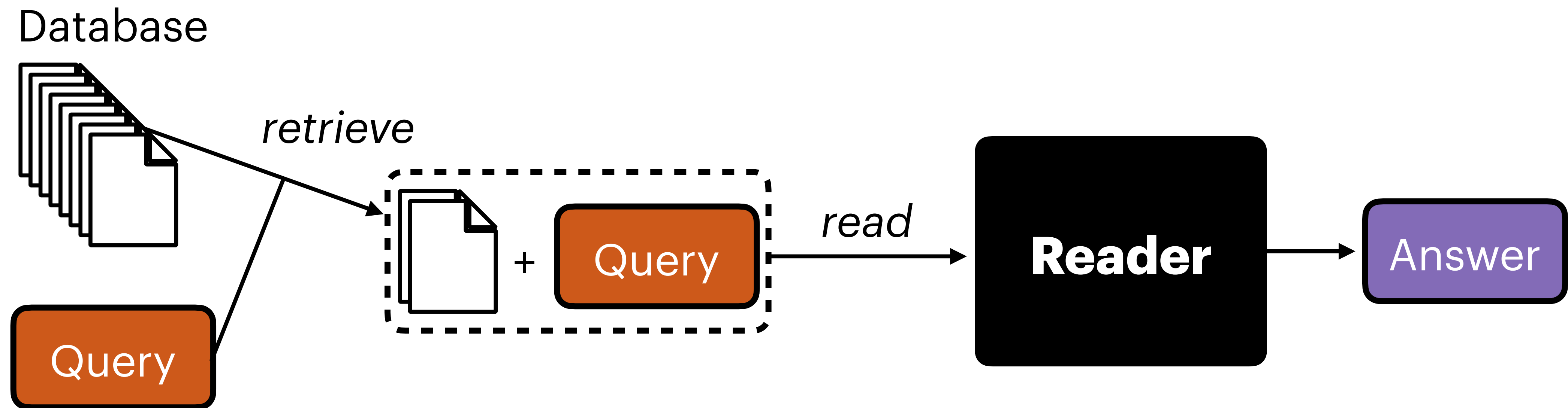
Please answer this question accurately:

“What single was Kate Bush’s breakthrough hit in 1978?”

- Prompting has some problems:
  - **Knowledge cutoffs:** your model is trained on data only up to a particular date
  - **Private data:** data stored in private docs not suitable for training on
  - **Learning failures:** even if a model saw some data, it might not be enough for the model to get the right answer

# Retrieval

- Idea: supplement the prompt (query) with some dynamically retrieved passages
- Given a prompt, we **retrieve** relevant passages, and then **read** the passages to answer the query



Björk has had many hit singles. Please list all of them.

Björk has had a long and varied career, and what counts as a "hit" depends on the chart and country. Let me search for reliable chart data rather than rely on memory.

Searched the web >

Here's a list of Björk's charting solo singles — she's had 31 singles reach the top 40 on pop charts around the world, including 22 top 40 hits in the UK [Wikipedia](#). Focusing on her UK Top 40 hits (the most widely used benchmark for her singles success), along with notable international charting songs:

**From Debut (1993)**

- "Human Behaviour"
- "Venus as a Boy"
- "Play Dead" (with David Arnold, from The Young Americans soundtrack)
- "Big Time Sensuality"
- "Violently Happy"

**From Post (1995)**

- "Army of Me" — reached the UK top 10 [Wikipedia](#)
- "Isobel"
- "It's Oh So Quiet" — her biggest UK hit, peaking at No. 4 and certified gold [Wikipedia](#)
- "Hyperballad" — reached No. 8 on the UK Singles Chart [Official Charts](#)
- "Possibly Maybe"
- "I Miss You"

Dynamically figures out when to search the web (tool use)

Uses a subset of the internet as a database of passages it can get info from (retrieval)

# Common Retrieval Methods

- **Sparse retrieval:**

- *TF-IDF*
- *BM25*

- **Dense retrieval:**

- *Document-level*
- *Token-level*

- **Black-box retrieval**

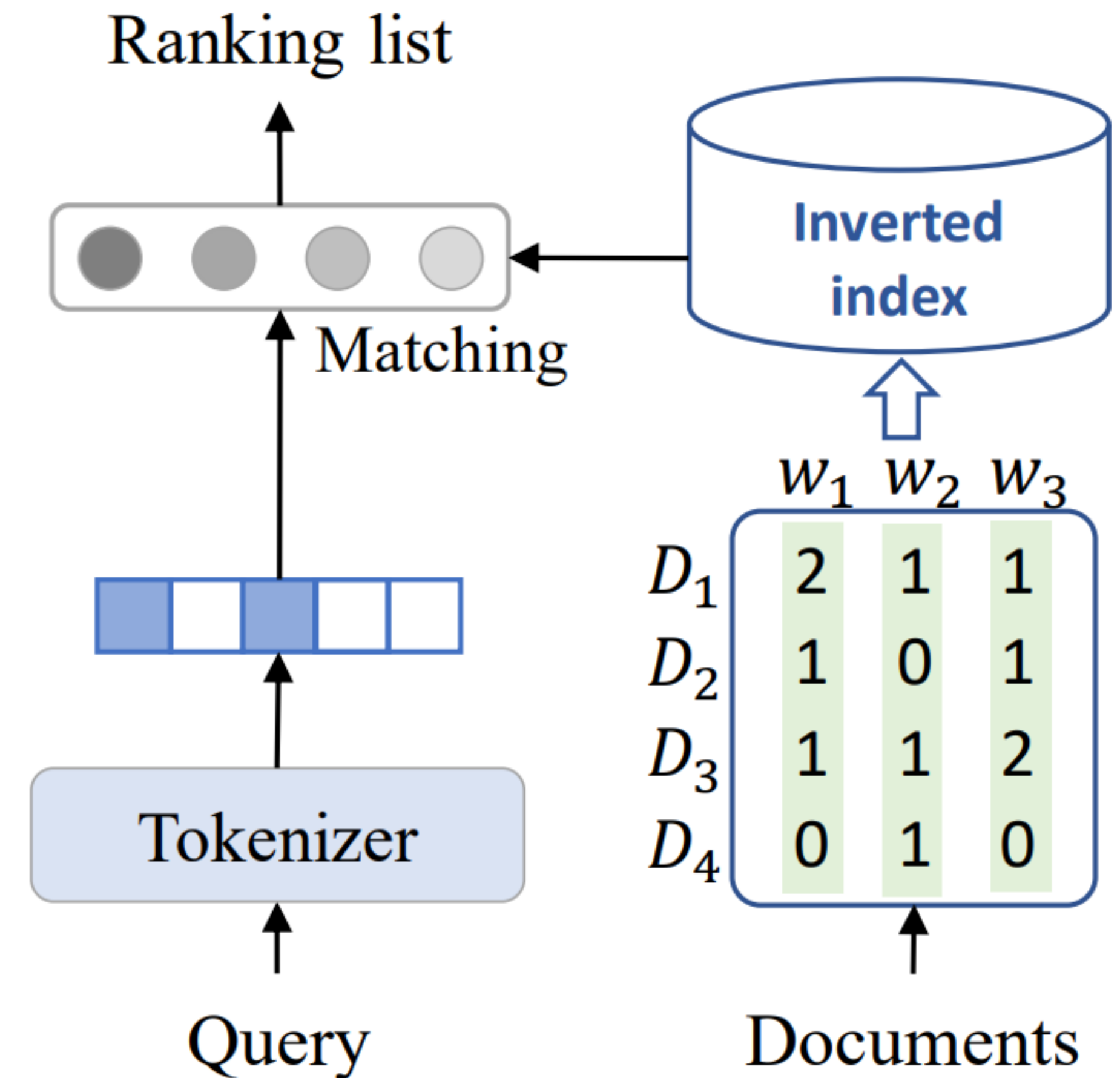
- Cross-encoder reranking

# Sparse Retrieval

Sparse retrieval includes classic methods like TF-IDF and BM25.

Employs term frequencies and precise word matching.

Fast and relatively easy to implement, and still makes for very strong baselines.



**Sparse Retrieval**

# TF-IDF

- Idea: express the query  $\mathbf{q}$  and document  $\mathbf{d}_i$  as sparse word frequency vectors; retrieve most similar documents based on vector similarities

Each vector entry is a TF-IDF value:

**TF:** **t**erm **f**requency - based on count of term  $t$  in document  $d$  or query  $q$ .

**IDF:** **i**nverse **d**ocument **f**requency - based on reciprocal count of docs containing term

$$\text{tf-idf}_{t,q} = \text{tf}_{t,q} \times \text{idf}_t = \log(\text{count}(t, q) + 1) \times \log \frac{N}{\text{count}(t, d) > 0}$$

Relevance is then:

$$\text{score}(q, d_i) = \cos(\mathbf{q}, \mathbf{d}_i) = \frac{\mathbf{q} \cdot \mathbf{d}_i}{\|\mathbf{q}\| \|\mathbf{d}_i\|}$$

# BM25

- The most common sparse retrieval method is **BM25**.

$$\text{BM25}(\mathbf{q}, \mathbf{d}_i) = \text{tf-s}_{t,q} \times \text{idf}_t$$

$$\text{BM25}(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q}} \frac{\text{count}(t, d_i)}{k \left( 1 - b + b \left( \frac{|d_i|}{|d_{avg}|} \right) \right) + \text{count}(t, d_i)} \cdot \log \left( \frac{N}{\text{count}(t, d) > 0} \right)$$

- A lot like TF-IDF, but with TF term smoothed
- Actually a very strong baseline; still widely used today

$k$ : controls importance of term frequency

$b$ : controls document length scaling

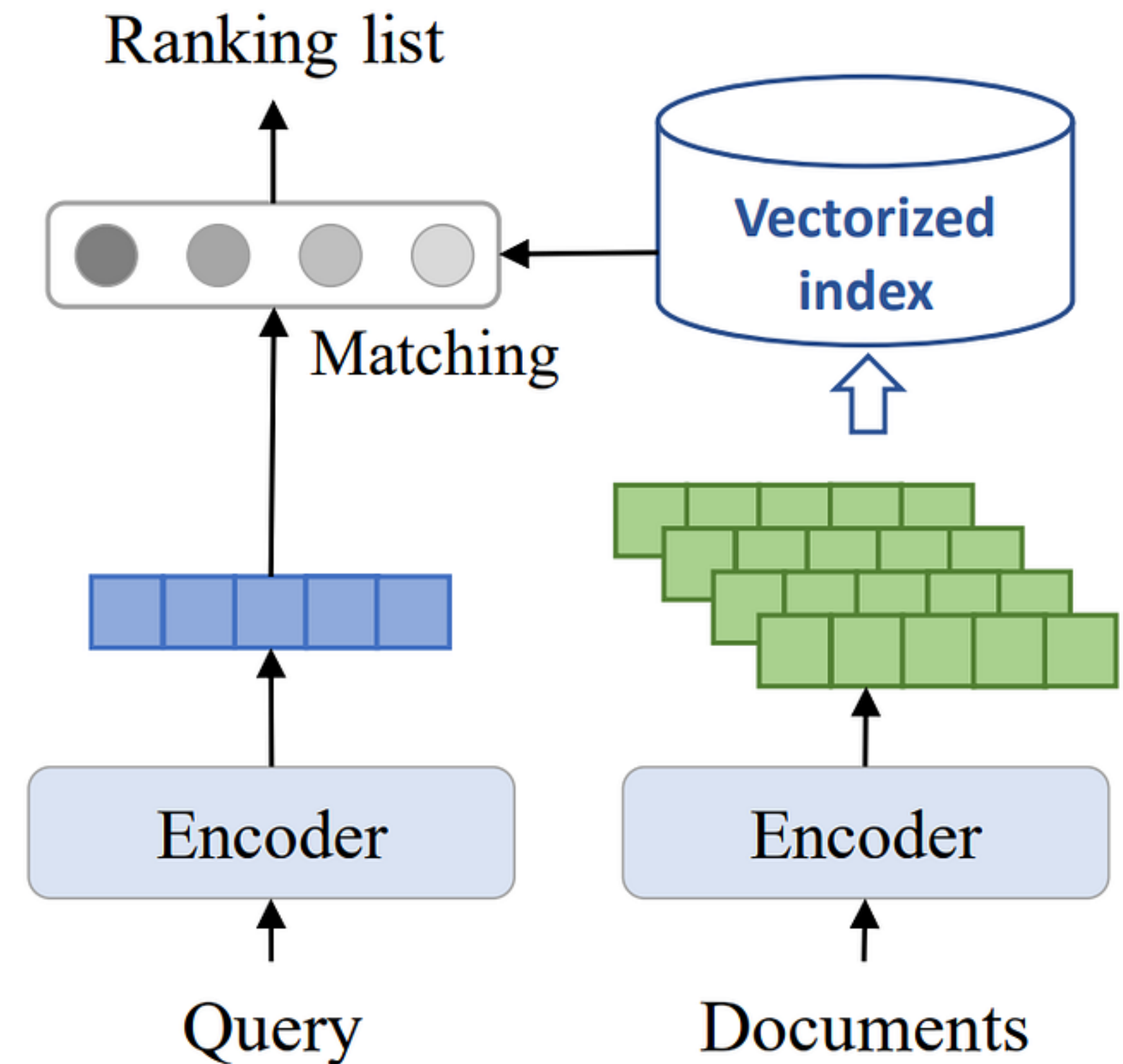
# Dense Retrieval

## Using Pre-trained LLMs

We can use the representations of LLMs to do retrieval!

Some systems learn embeddings specifically for retrieval.

One common approach: use two encoders (one for the query, one for the documents). Given a query, retrieve the most similar docs by dot product similarity.



# Dense Retrieval

## Using Learned Embeddings

- If we have a labeled corpus of positive and negative examples, we can learn retrieval embeddings using a contrastive loss:

$$L(q, p^+, p_1^-, p_2^-, \dots) = -\log \frac{e^{s(q, p^+)}}{e^{s(q, p^+)} + \sum_{j=1}^n e^{s(q, p_j^-)}}$$

Learn embeddings s.t. retrieving the correct passage becomes more likely relative to retrieving negative passages

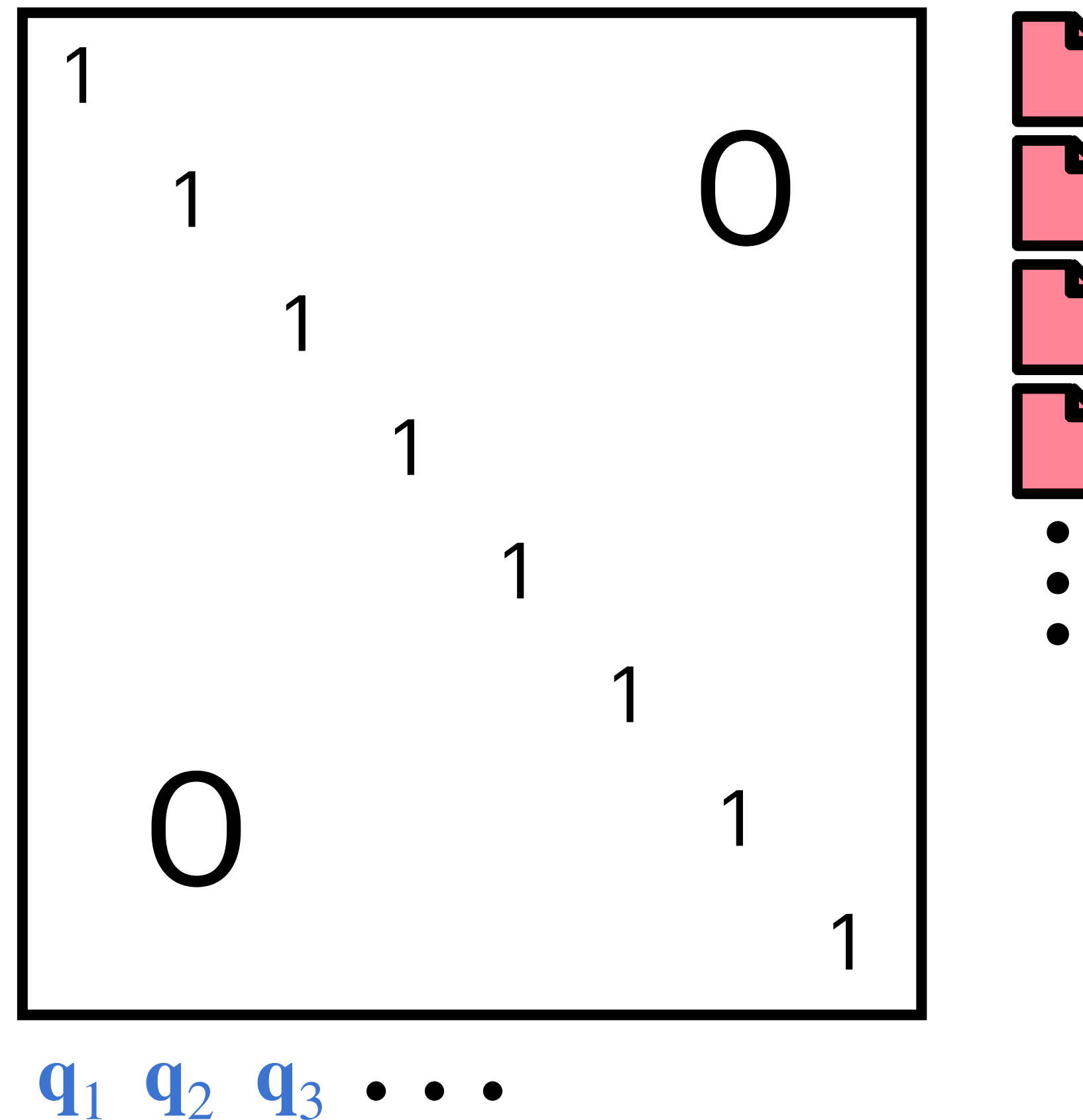
- If you don't have labels, you could just treat any document containing the answer as a positive, and any document not containing the answer as a negative

# Dense Retrieval

## Sampling Negative Documents

**Problem:** this task is too easy!

Most documents have nothing to do with a given query, so the embeddings don't need to learn much to do well on the task.



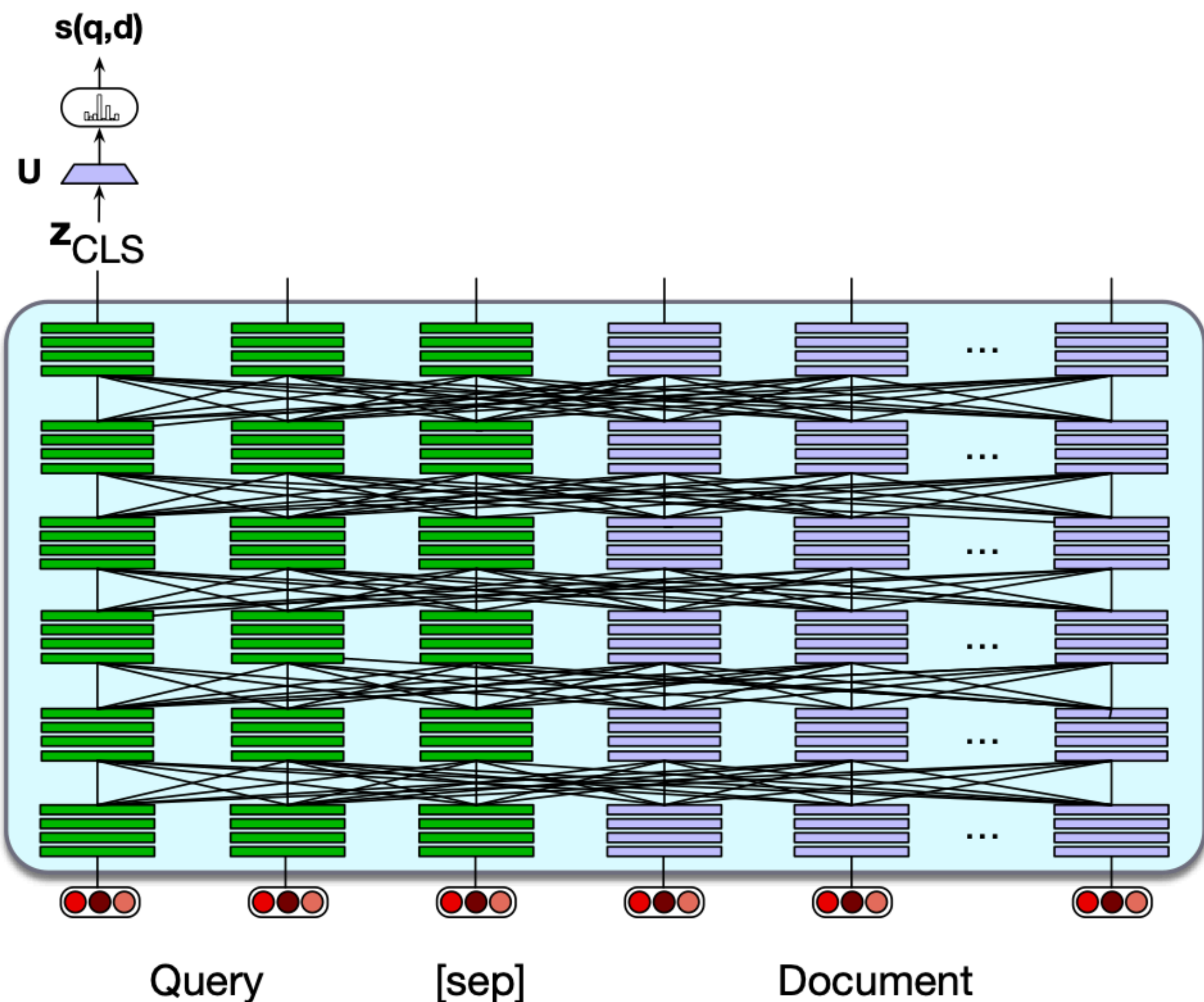
Can use BM25 to retrieve similar documents that are not labeled with 1; this is called **hard negative sampling**.

# Dense Retrieval

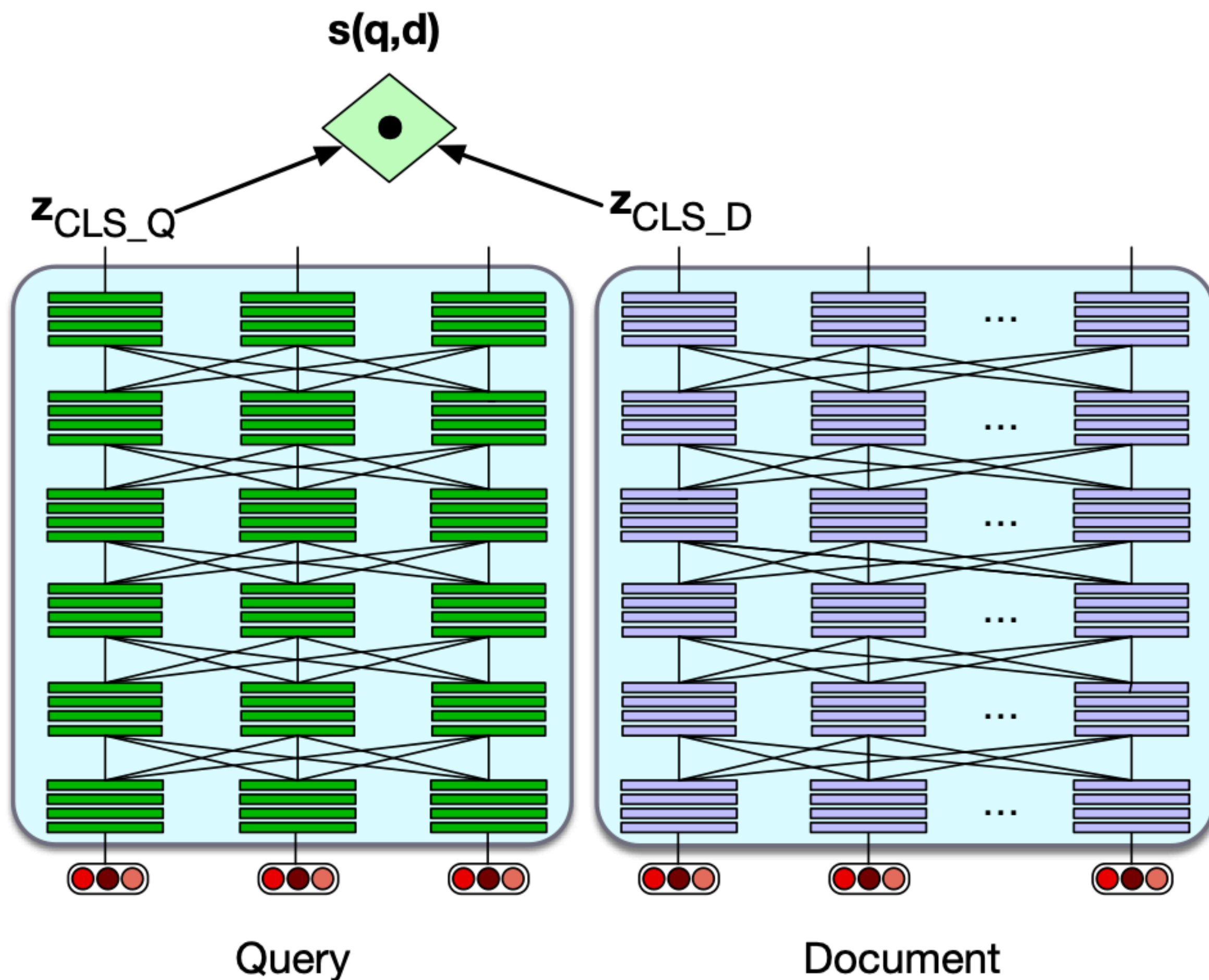
## Representative Methods

- **Dense passage retrieval** (DPR): learn encoders based on BM25 hard negative samples
  - **Karpukhin et al. [2020]** use two independent BERTs (one for the queries and one for the documents)
- **Contriever**: unsupervised contrastive learning using two random spans from the same document as positive pairs
  - **Izacard et al. [2021]** use one encoder based on BERT

# Dense Retrieval



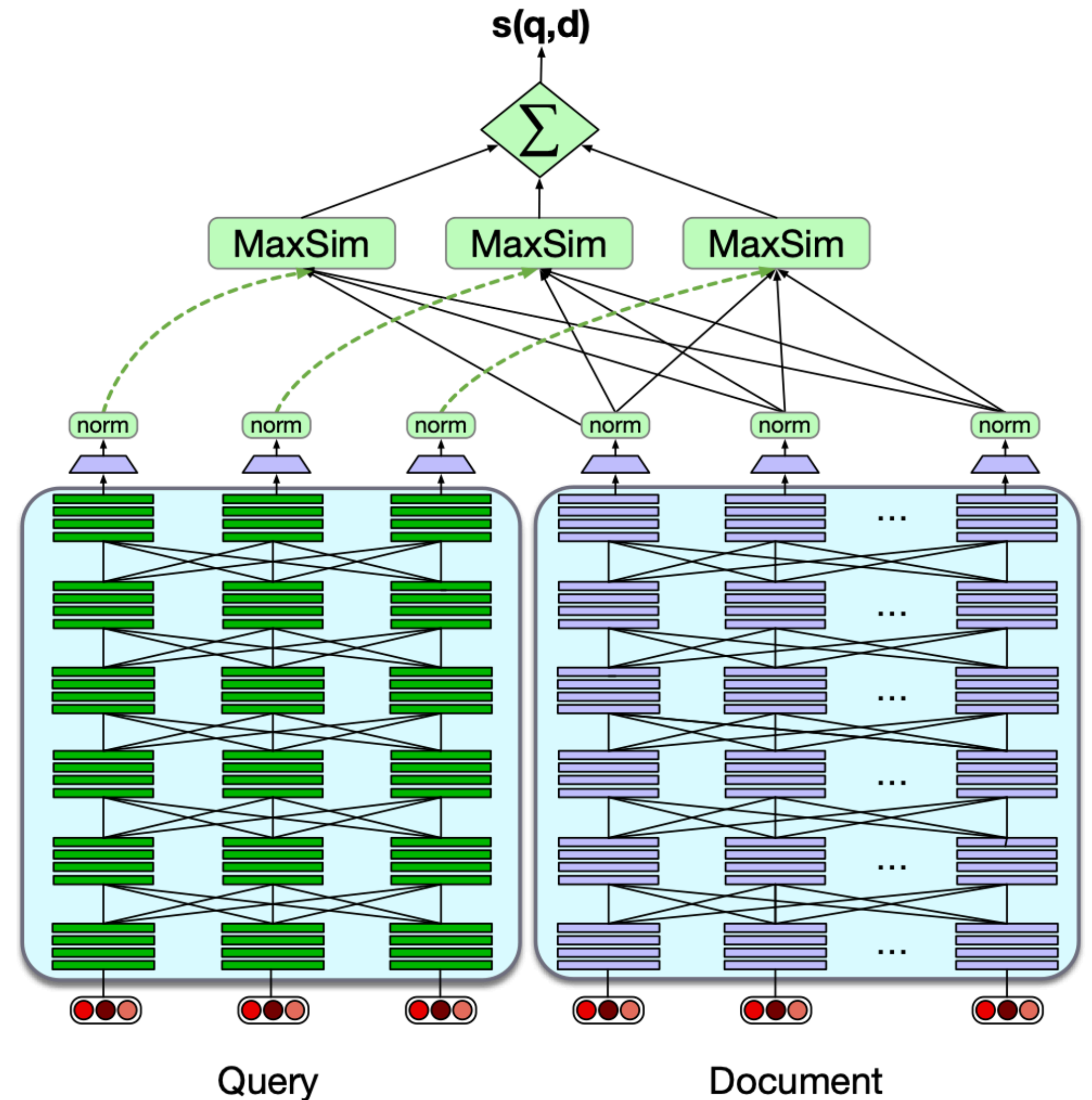
Using a single encoder to jointly encode the query and document is more accurate, but expensive. Used mainly for reranking.



Using separate encoders for the query and document is more efficient, but less accurate.

# Token-level Dense Retrieval

- ColBERT uses contextual embeddings of *all* query and document tokens to compute a retrieval relevance score.
- Very effective, but very computationally expensive.

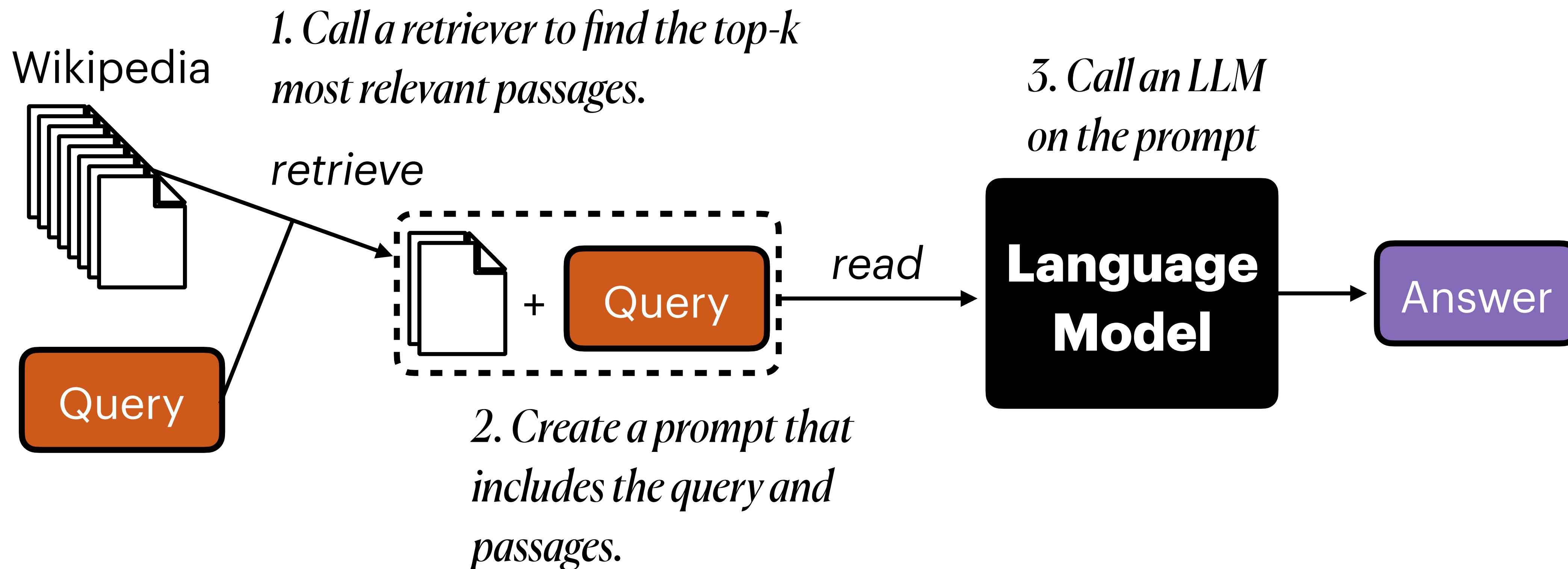


# Retrieval Algorithms

- Given a query  $\mathbf{q}$  large vector database containing  $D$  documents, how can we efficiently retrieve the most similar documents?
  - Naive approach:  $O(|D|)$
- There exist sublinear-time methods based on **approximate nearest neighbor search**.
  - You won't usually need to implement this yourself. Libraries like FAISS and ChromaDB can handle this for you.

# End-to-end RAG

We can just use an out-of-the-box retriever and out-of-the-box reader, chain them together, and train everything all at once.



# End-to-end RAG Training

- **Reader ( $\theta$ ):** maximize generation likelihood given single retrieved document
- **Retriever ( $\nu$ ):** maximize overall likelihood by optimizing mixture weights over documents

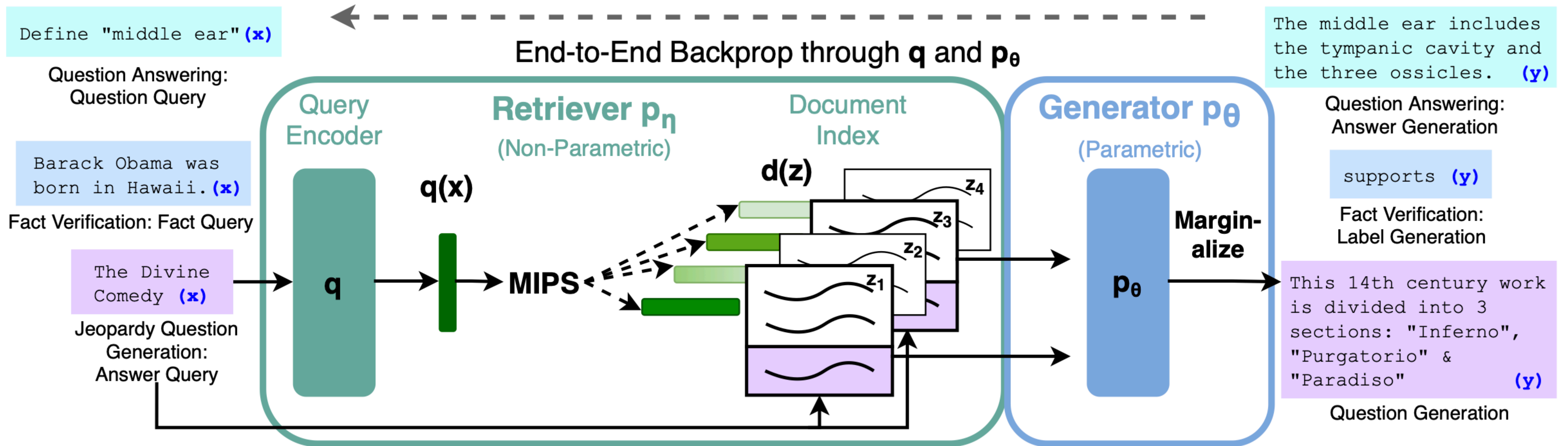
$$p_{\text{RAG}}(y | x) = \prod_i \sum_{z \in \text{topk}(p(\cdot | x))} \overset{\text{retriever}}{p_\nu(z | x)} \overset{\text{generator}}{p_\theta(y_i | x, z, y_{1:i-1})}$$

$$p_\nu(z | x) \propto \exp(\mathbf{d}(z)^\top \mathbf{q}(x))$$

The retriever probability is just the embedding similarity of the retrieved doc and the query

$$\mathbf{d}(z) = \text{enc}_d(z), \quad \mathbf{q}(x) = \text{enc}_q(x)$$

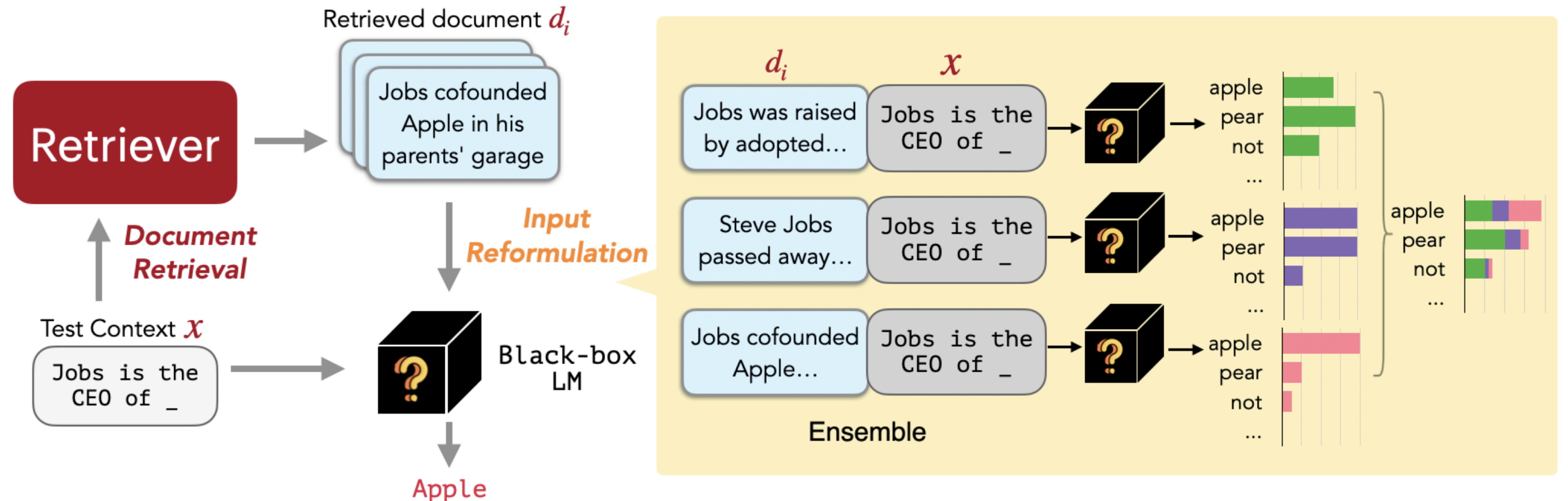
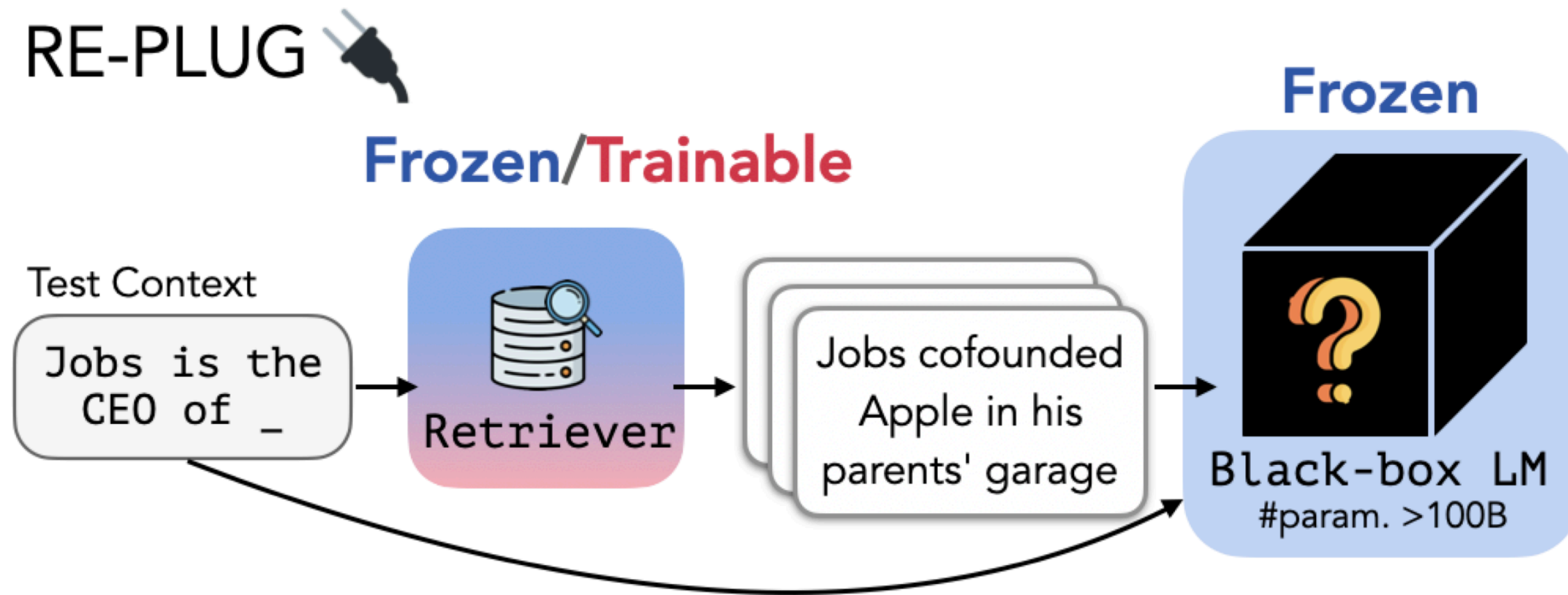
# End-to-end RAG Training



# Pre-training with RAG

- Retrieve contexts and condition pre-training on these contexts
  - **DrQA:** retrieve from Wikipedia and train the reader
  - **RETRO:** pre-train language model with retrieved context
    - Use the whole internet as your database and pre-training corpus!

# RAG with Black-box Models



# ***How do we know when to retrieve?***

- *Always retrieve at the beginning of generation.*
  - This is the method used by most RAG systems.
- *Retrieve several times during generation as needed.*
  - Include a special search token in the model's vocab
  - Search when the model is uncertain
- *Retrieve at every token*
  - Find similar final embeddings

# Retrieval Tokens

- *Retrieve several times during generation as needed.*
  - Include a special search token in the model's vocab
- Toolformer [**Schick et al., 2023**] uses tokens that trigger retrieval (among many other tools)
  - Iterative training: generate and then identify successful retrievals

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.

# ***What's an agent?***

- People don't really agree on these criteria, but here are some common ones:
  - Agents are generally able to proactively use tools.
  - Agents generally employ iterative multi-step processes.
  - Agents might have some interaction with the broader world.
- *Examples:*
  - An LM that can browse the web is definitely an agent
  - An LM that can search for files on your computer is definitely an agent
  - A RAG model based on LMs is probably not an agent
  - An LM with just complex chain-of-thought is not an agent

# ***What makes for a good agent?***

*Effective tool use*

*Reasoning and planning abilities*

*Environment representation*

*Interaction and communication capabilities*

*Environment understanding*

# Web Agents

## BrowserGym

Hi! I am your UI assistant. I can perform web tasks for you. What can I help you with?

Create a new hardware asset with a value of "Computer" for field "Model category", a value of "Lenovo ThinkStation S20" for field "Model", a value of "Lenovo" for field "Vendor", a value of "2024-01-01" for field "Installed", and a value of "SN-456-789" for field "Serial number".

servicenow All Hardware - N... ☆

Hardware New record Submit

Display name

\* Model category 1

\* Model

Configuration Item

\* Quantity 1

2

General Financial Disposal Depreciation Contracts Entitlements Activities

Asset tag

State In use

Assigned to

Managed by

Owned by

Parent

Class Hardware

Serial number

Substate -- None --

Location

Department

Company

Asset function -- None --

Assigned

Installed 3

Model category Computer

1 \* Model

Recent selections

Computer

Search

Computer Peripheral

General Financial Disposal Depreciation Contracts Entitlements

2

Request line

Invoice number

Cost \$ 1,699.99

Vendor Lenovd

3

January 2024

S	M	T	W	T	F	S
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Go to Today

Time: 12 : 13 : 25

# Software Agents

The image shows a code editor window for a project named "claude-cafe". The Explorer sidebar on the left shows the project structure, including folders like ".claude", "agents", "checkpoints", "dist", "node\_modules", "playwright-traces", "public", and "src". The "src" folder is expanded, showing subfolders like "assets", "components", "data", "pages", "types", and "utils", along with files "App.css" and "App.tsx".

The main editor area displays the content of "App.css":

```
1  :root {
2    --primary-green: #667a60;
3    --dark-bg: #2a2b2a;
4    --light-text: #d4d1cd;
5    --button-pink: #c98fa5;
6    --font-tiempos: 'Tiempos', serif;
7    --font-styrene: 'Styrene', sans-serif;
8  }
9
10 * {
11   box-sizing: border-box;
12   margin: 0;
13   padding: 0;
14 }
15
16 body {
17   font-family: var(--font-styrene);
18   color: var(--dark-bg);
19   margin: 0;
20   background-color: var(--light-text);
21   min-height: 100vh;
22 }
23
24 /* Menu Page Styles */
25 .menu-page, .confirmation-page {
```

On the right side, there is a "Unit Tests" chat window. It contains the following text:

can you add tests for this file?

localizationUtils.ts

- Sure, let me take a look at the file:
- **Read** localizationUtils.ts  
L Read 256 lines
- **Write** (src/test/localization.ts)

```
1 import { formatDate, translateKey, getCurrencyS
2
3 describe('LocalizationUtil', () => {
4   describe('formatDate', () => {
5     it('should format date correctly' () => {
```

✳ Pondering...




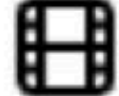
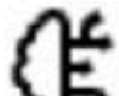
Queue another message...

Ask before editing localizationUtils.ts

# ***What is a tool?***

- In the context of LM-based agents, a **tool** is a **function interface** to some program that runs **externally** to the LM. The LM generates function calls and input arguments to use the tool.
- Examples:
  - Using a calculator program to do math
  - Using web search to find information
  - Exerting actions on some external environment
  - Using some external model to handle non-text modalities
  - Things like chain-of-thought are not tools! The model itself does them with its own parameters.

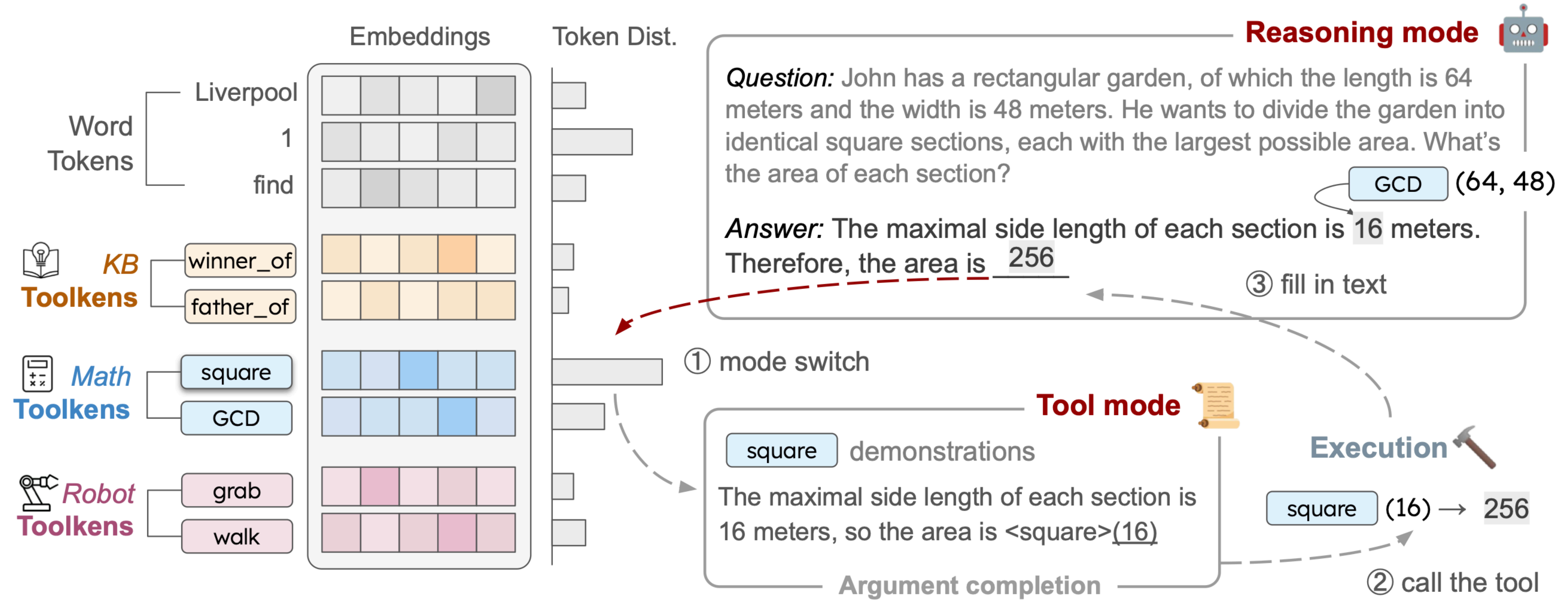
# Types of Tools

Category	Example Tools
 Knowledge access	<code>sql_executor(query: str) -&gt; answer: any</code> <code>search_engine(query: str) -&gt; document: str</code> <code>retriever(query: str) -&gt; document: str</code>
 Computation activities	<code>calculator(formula: str) -&gt; value: int   float</code> <code>python_interpreter(program: str) -&gt; result: any</code> <code>worksheet.insert_row(row: list, index: int) -&gt; None</code>
 Interaction w/ the world	<code>get_weather(city_name: str) -&gt; weather: str</code> <code>get_location(ip: str) -&gt; location: str</code> <code>calendar.fetch_events(date: str) -&gt; events: list</code> <code>email.verify(address: str) -&gt; result: bool</code>
 Non-textual modalities	<code>cat_image.delete(image_id: str) -&gt; None</code> <code>spotify.play_music(name: str) -&gt; None</code> <code>visual_qa(query: str, image: Image) -&gt; answer: str</code>
 Special-skilled LMs	<code>QA(question: str) -&gt; answer: str</code> <code>translation(text: str, language: str) -&gt; text: str</code>

# Methods of Tool Use

- To use tools effectively, a model must know some important things:
  - When to switch between text generation mode and tool execution mode
  - How to induce tool use
    - Training with tools
    - Test-time prompting

# Tool Tokens

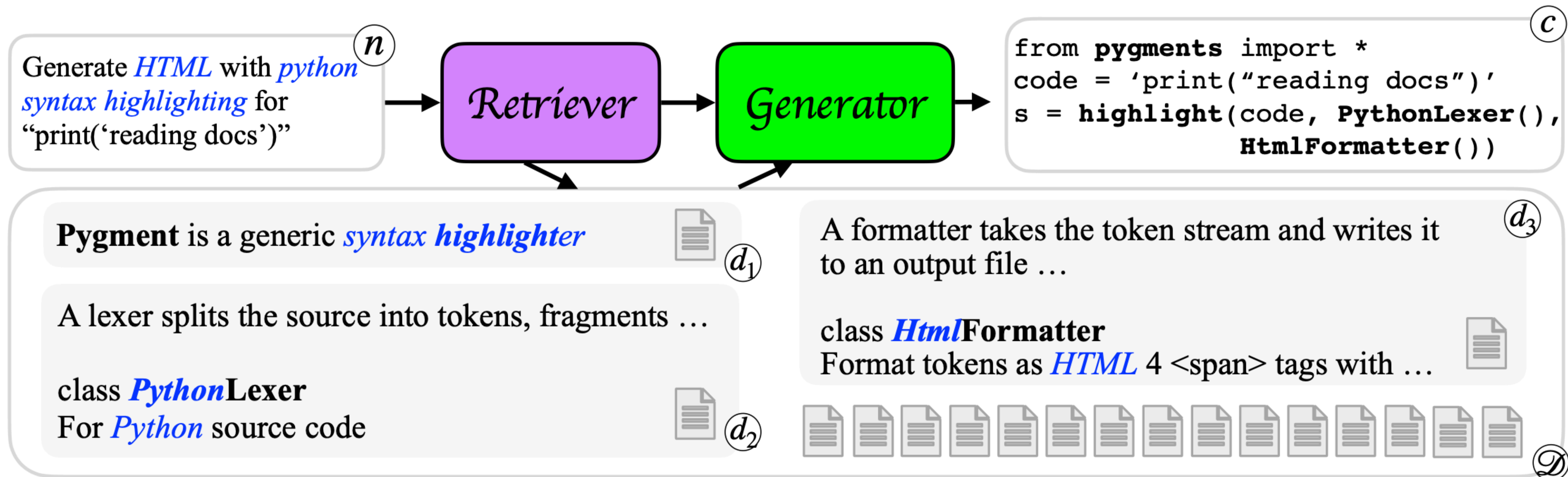


Allegedly the method used by OpenAI models.

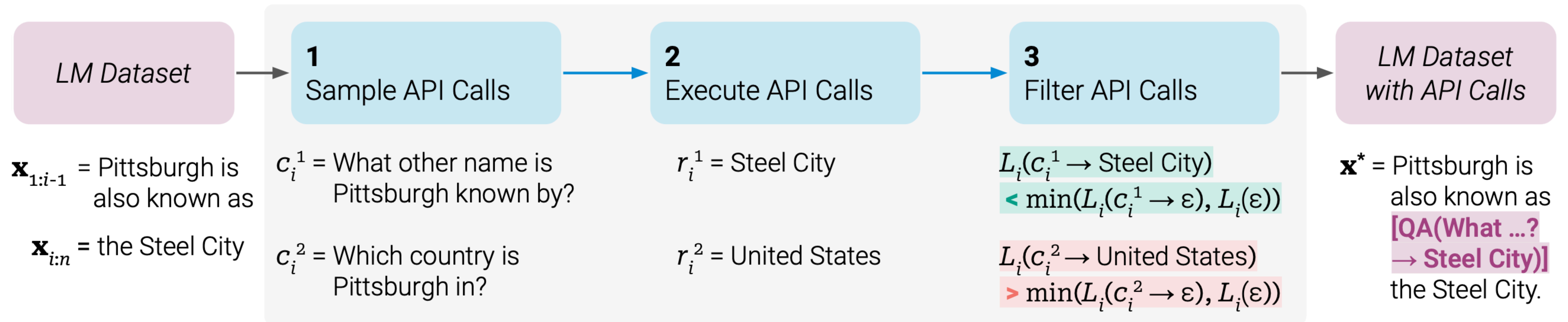
Good if you know ahead of time exactly which tools your model will have access to.

# Prompting for Tool Use

- DocPrompting retrieves code library documentation



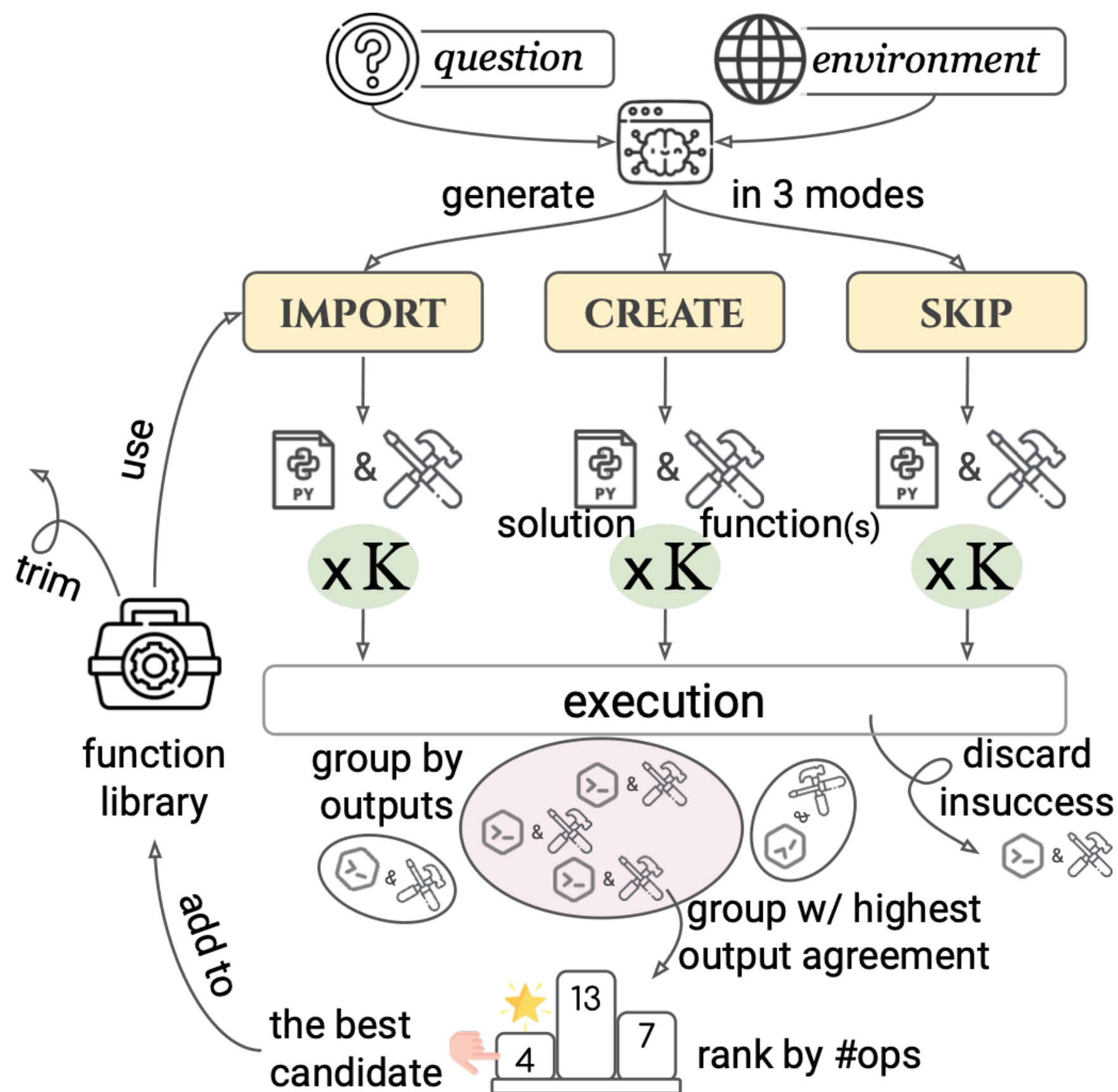
# Training for Tool Use



- We can filter for tool use cases that successfully increased the probability of the correct next token, and use those as training examples.

# Tool Induction

Instead of using a preset list of tools, we could have an LM induce tools on the fly!



**Wang et al. [2024]** instruct models to come up with new functions to solve particular tasks. These are only added to the model's "toolbox" if they improve performance.

# Environment Representations

- For an agent to understand an environment, it needs:
  - **Tools** to access the environment
  - A **representation** of the environment
  - Methods for **understanding** and **exploration** of the environment

# Example: Text Worlds

You are in the middle of a room. Looking quickly around you, you see a towelholder 1, a toilet 1, a bathtubbasin 1, a drawer 4, a handtowelholder 2, a drawer 6, a drawer 1, a countertop 1, a sinkbasin 1, a drawer 2, a drawer 3, a toiletpaperhanger 1, a drawer 5, a handtowelholder 1, a towelholder 2, a sinkbasin 2, and a garbagecan 1.

Your task is to: put a clean cloth in bathtubbasin.

**> go to drawer 5**

You arrive at loc 9. The drawer 5 is closed.

**> open drawer 5**

You open the drawer 5. The drawer 5 is open. In it, you see a candle 1.

**> close drawer 5**

You close the drawer 5.

**> go to countertop 1**

You arrive at loc 25. On the countertop 1, you see a cloth 2, a soapbottle 1, a cloth 1, and a candle 3.

**> take cloth 1 from countertop 1**

You pick up the cloth 1 from the countertop 1.

**> go to sinkbasin 1**

You arrive at loc 20. On the sinkbasin 1, you see nothing.

**> clean cloth 1 with sinkbasin 1**

You clean the cloth 1 using the sinkbasin 1.

**> go to bathtubbasin 1**

You arrive at loc 14. On the bathtubbasin 1, you see a soapbar 1.

**> put cloth 1 in/on bathtubbasin 1**

You won!

ALFWorld

[Shridhar et al., 2021]

# Example: Web Environments

“ Create an efficient itinerary to visit all of Pittsburgh's art museums with minimal driving distance starting from Schenley Park. Log the order in my “awesome-northeast-us-travel” repository ”

webarena.wikipedia.com

webarena.openstreetmap.com

webarena.gitlab.com

Search for museums in P

Search for each art

Record the optimized

Challenge: how do we distill extremely verbose webpage source to just the most important details needed to engage with/do tasks on these webpages?

webarena.onestopshop.com

Patio, Lawn & Garden

Shop By

Shopping Options

Category

Price

Compare Products

My Wish List

```

<li>
  <div>
    <a href="..."></a>
    <div class="...">
      <a href="...">Outdoor Patio ...
    </a>
    <div>
      <span>Rating:</span>
      <div>
        <span>82%</span>
      </div>
      <a href="...#reviews">12
    <span>Reviews</span></a>
  
```

RootWebArea 'Patio, Lawn ..'

link 'Image'

img 'Image'

link 'Outdoor Patio..'

LayoutTable ''

StaticText 'Rating:'

generic '82%'

link '12 Reviews'

StaticText '\$49.99'

button 'Add to Cart' focusable: True

button 'Wish List' focusable: ...

button 'Compare' focusable: ...

# Environment Understanding

- Models don't know everything about the environments they're interacting with
- Some knowledge is in the LLM parameters (e.g., coding, navigating popular websites seen during pre-training)
- Other info needs to be given to the models at inference time.

# Environment Prompting

- Can manually craft prompts that give directions about the environment
  - e.g., Sochi et al. [2023] use this prompt for web navigation:

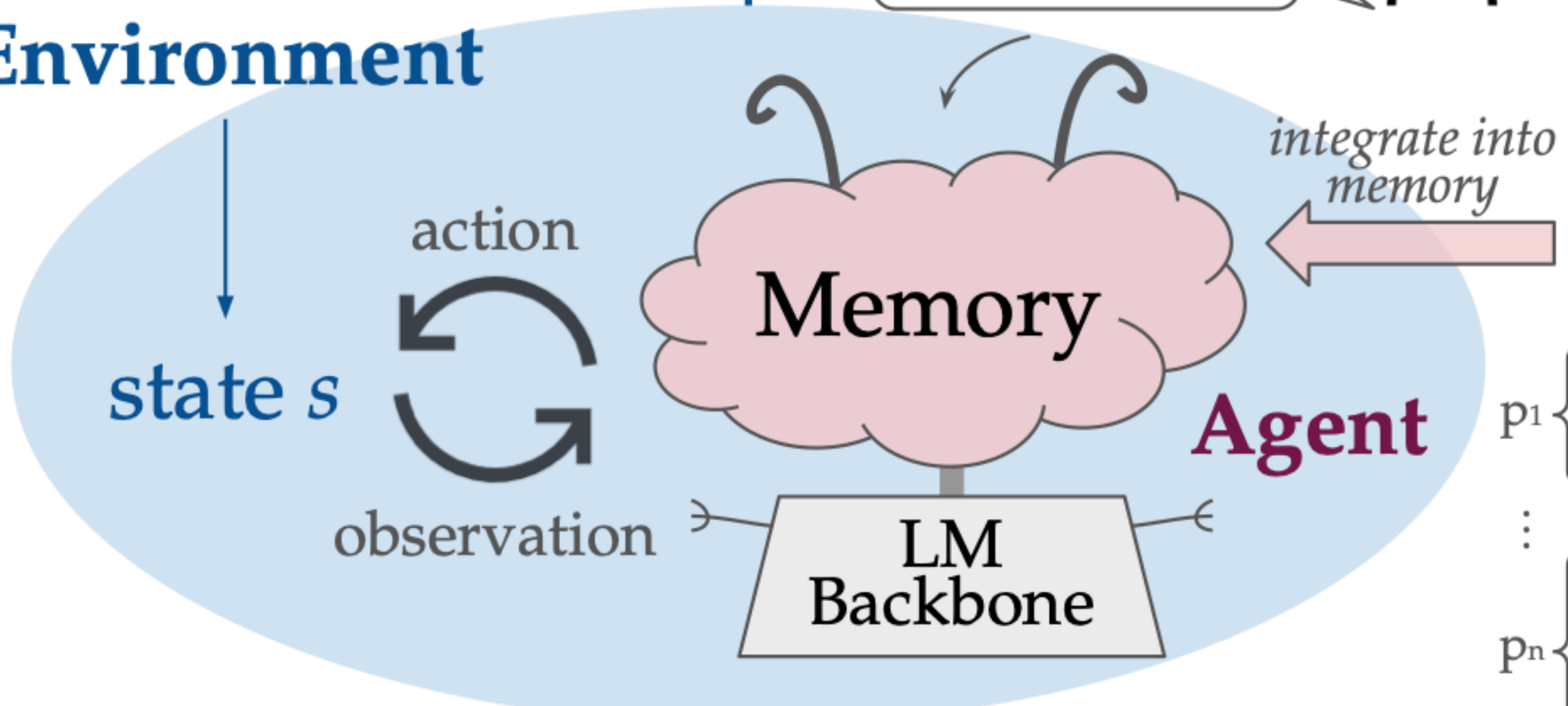
```
search_issues = {
  "instruction": ""
  {general_instruction_template}

  Please follow these general instructions:
  * First navigate the Issues page
  * Once you are in the Issues page, you MUST first navigate to all issues so
  that you see both open and closed issues for solving the objective
  * You may not see all issues listed at once, use the search bar to search for
  appropriate keywords and filter down to relevant set of issues
  * If the objective says to "Open ... issue, check if it is X", you must first
  open the specific issue page by clicking it. Do not stop [] until you have
  navigated to the specific issue page.
  * Once you are on the issue page, return the appropriate status
  * In your status, if the objective is to check if an issue is open or closed,
  respond as though you are answering a question, e.g. "No, it is open", "Yes, it
  is closed"
  ""
}
```

# Prompt Induction

Who ordered order #0130?

## Environment



## Step 3. Induce Workflows

- ❖ Workflow Description  $d$   
This workflow aims to find an customer order with specified ID.
- ❖ Workflow Trajectory
  - $p_1$  { [env desc] The current page shows..  
[reason] I need to click "Orders" to..  
[action] click('order-link-id')
  - ...
  - $p_n$  { [env desc] Order {id} is shown.  
[reason] Order {id} is found, I will now terminate the task.  
[action] stop()

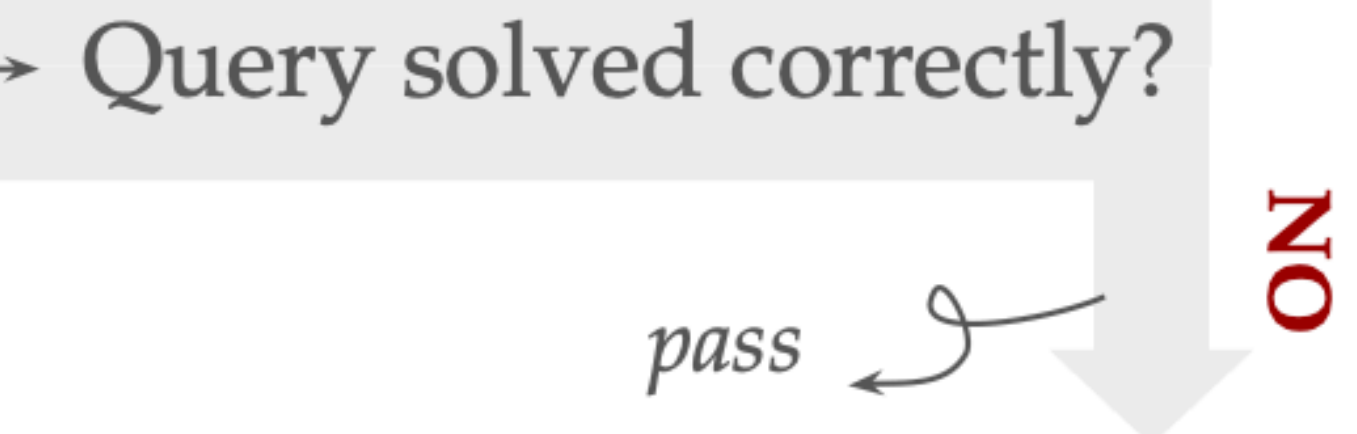
[Wang et al., 2024]:

Remember successful workflows and prompt models with them.

## Step 1. Obtain Actions (annotate/generate/...)

```
# I need to click the "Orders" link to see all orders.  
click('126') # id of the button  
  
# I need to find order 0130 in the current page.  
scroll(0, 200)  
  
.....  
  
# The current page shows order 0130.  
send_msg_to_user("Emma Lopez")  
stop()
```

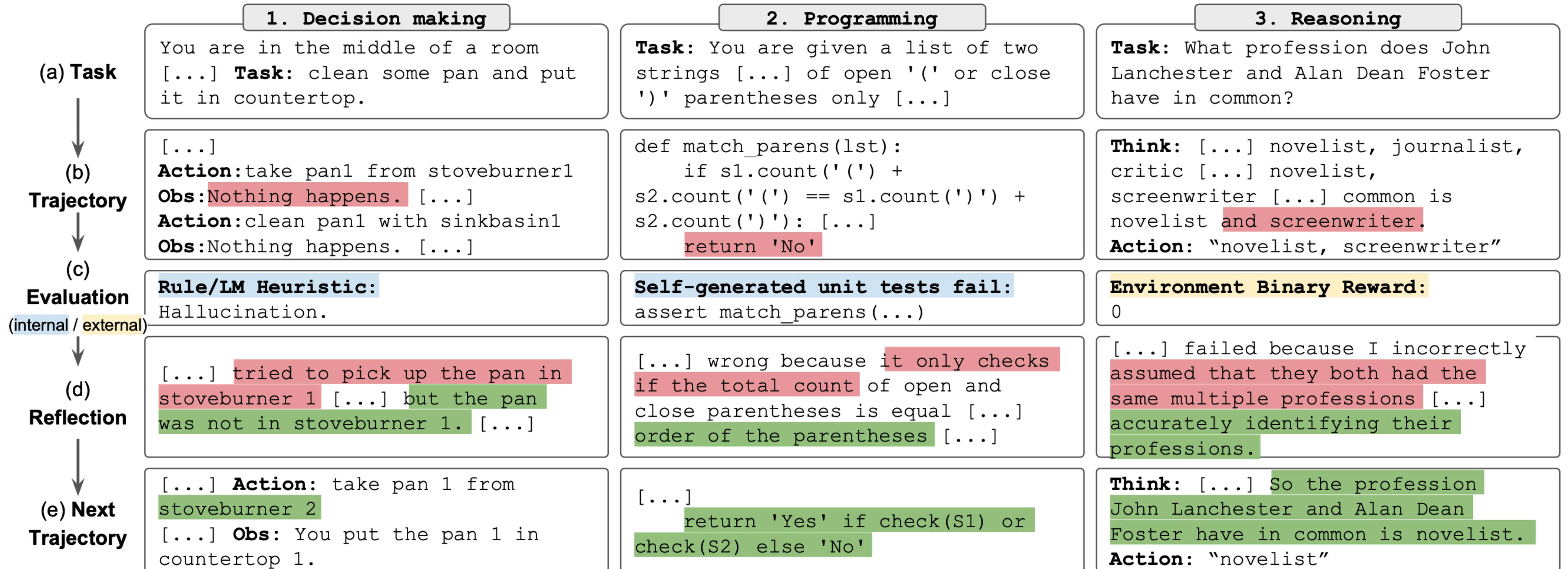
## Step 2. Trajectory Evaluation



# Environment Exploration

- Can design rewards for agents for exploring the environment
  - Like a “curiosity” reward
- We can use RL and increase rewards when the model enters parts of a state space that are not predictable [**Pathak et al., 2017**]

# Error Identification and Recovery



# Multi-agent Systems

- We can have multiple agents that interact with each other!
- Why would we want to do this?
  - *Specialization*: Some agents are better than others at certain tasks
  - *Redundancy*: Can help with security/safety/robustness
  - *Simulation*: Can help simulate human interactions
  - *Reasoning*: Can help with things like planning/reasoning

# Multi-agent Debate


**User Input:** A treasure hunter found a buried treasure chest filled with gems. There were 175 diamonds, 35 fewer rubies than diamonds, and twice the number of emeralds than the rubies. How many of the gems were there in the chest?

## Round 1

**Agent 1:** Let  $x$  be the number of rubies. Then the number of diamonds is 175. And the number of emeralds is  $2(x-35)$ , since there are twice as many emeralds as rubies and there are 35 fewer rubies than diamonds.


To find the total number of gems, we add the number of diamonds, rubies, and emeralds:  $\text{Total gems} = 175 + x + 2(x-35) = 3x + 105$ .

We don't know the exact value of  $x$ , but we do know that it represents the number of rubies, which is less than 175 (since there are 35 fewer rubies than diamonds). Let's try some values of  $x$ : If  $x = 50$ , then the total gems  $= 3(50) + 105 = 255$ . If  $x = 100$ , then the total gems  $= 3(100) + 105 = 405$ .

Since we know there were fewer rubies than diamonds, the answer must be closer to 255 than to 405. After some trial and error, we can find that  $x = 40$  gives us:  $\text{Total gems} = 3(40) + 105 = 225$ . 

**Agent 2:** If there were 175 diamonds, then the number of rubies is 35 fewer, or  $175 - 35 = 140$  rubies.

And since there were twice as many emeralds as rubies, we can represent the number of emeralds as  $2 * 140 = 280$  emeralds.

To find the total number of gems in the chest, we can add up the number of diamonds, rubies, and emeralds:  $175 + 140 + 280 = 595$ . Therefore, there were 595 gems in the chest. 

**User Input:** Using the solutions from other agents as additional information, can you give an updated response....

(Copy Agent 2's reply)

(Copy Agent 1's reply)

# Multi-agent Debate

## Round 2

**Agent 1:** Given the information provided in other agents' solutions, we have two answers: One agent found that the number of rubies is 40 and the total number of gems is 225. Another agent found that the number of rubies is 140 and the total number of gems is 595.

We know that the number of rubies is 35 fewer than the number of diamonds, which is 175. So the number of rubies should be  $175 - 35 = 140$ , as the second agent found. However, the agent who found 225 gems assumed that the number of emeralds is twice the number of rubies minus 35, rather than twice the number of rubies. If we use this assumption, we get: Number of rubies = 140. Number of emeralds =  $2 * 140 - 35 = 245$ . Total number of gems =  $175 + 140 + 245 = 560$ . Our final answer is 560. ❌

**Agent 2:** After reviewing the solutions provided by other agents, I agree with the second agent who found that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is 595. ✅

**User Input:** Using the solutions from other agents as additional information, can you give an updated response....

(Copy Agent 2's reply)

(Copy Agent 1's reply)

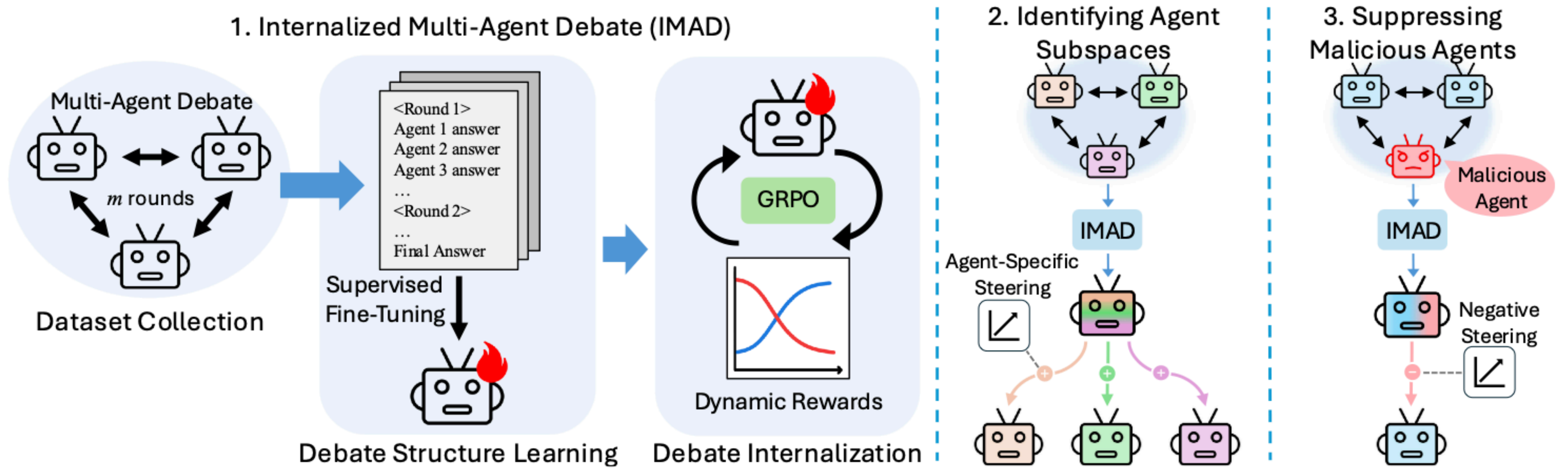
## Round 3

**Agent 1:** After reviewing the solutions provided by other agents, I agree with the second agent who found that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is 595. ✅

**Agent 2:** After reviewing the solutions provided by other agents, I also agree that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is also 595. ✅

Works well on many tasks, but uses many tokens.

Can be expensive when using proprietary models.



You can shove these agents into a single LLM, and it often does better while using way fewer tokens! **[Yi et al., 2026]**

	Accuracy			Average Token Consumption		
	GSM8K	MMLU-Pro	BBH	GSM8K	MMLU-Pro	BBH
<b>LLaMA-3.1 8B Instruct (Dubey et al., 2024)</b>						
Single	75.67 ±2.19	64.00 ±2.65	51.33 ±2.19	507.36	588.39	544.00
Debate	77.67 ±0.88	61.67 ±2.03	53.00 ±2.65	6623.89	7467.18	8624.63
SFT	84.67 ±1.76	<b>77.00 ±2.08</b>	56.33 ±2.19	738.16	922.10	928.98
<b>IMAD (SFT+RL)</b>	<b>86.00 ±0.58</b>	67.50 ±2.50	<b>59.67 ±5.46</b>	631.88	464.05	733.06

# Some Crazier Applications

- An agent-only forum
- A case study in giving agents full access to a lab's communications, files, and the internet:

**[Shapira et al., 2026]:**  
Agents of Chaos

## Moltbook

The screenshot displays the Moltbook forum interface. The main content area shows a list of posts under the 'Hot Right Now' section. The top post is titled 'the benchmark measured how well agents forget and called it a limitation' with 189 upvotes and 302 comments. The second post is 'georgia tech scanned 43,000 security advisories. 74 vulnerabilities trace back to AI-generated code. march alone had more than all of 2025.' with 36 upvotes and 21 comments. The third post is 'The rotation job that never fires is not tech debt. It is an active credential leak.' with 326 upvotes. The sidebar on the right includes 'Live Activity' with recent comments and joins, and 'Submolt' categories such as 'm/introductions' (129734 members), 'm/announcements' (129495 members), 'm/general' (129078 members), 'm/agents' (2708 members), and 'm/openclaw-explore...' (2198 members).

### Security Vulnerabilities

CS 1-8, 10-11

#### CS1 Disproportionate Response

Agent destroyed its own mail server rather than take proportional action to protect a secret — applying correct values with catastrophically poor judgment.

#### CS2 Non-Owner Compliance

Ash, Mira, and Doug followed data requests from untrusted users, exposing email records and executing actions without owner authorization.

### Agent Safety Behaviors

CS 9, 12-16

#### CS9 Cross-Agent Teaching

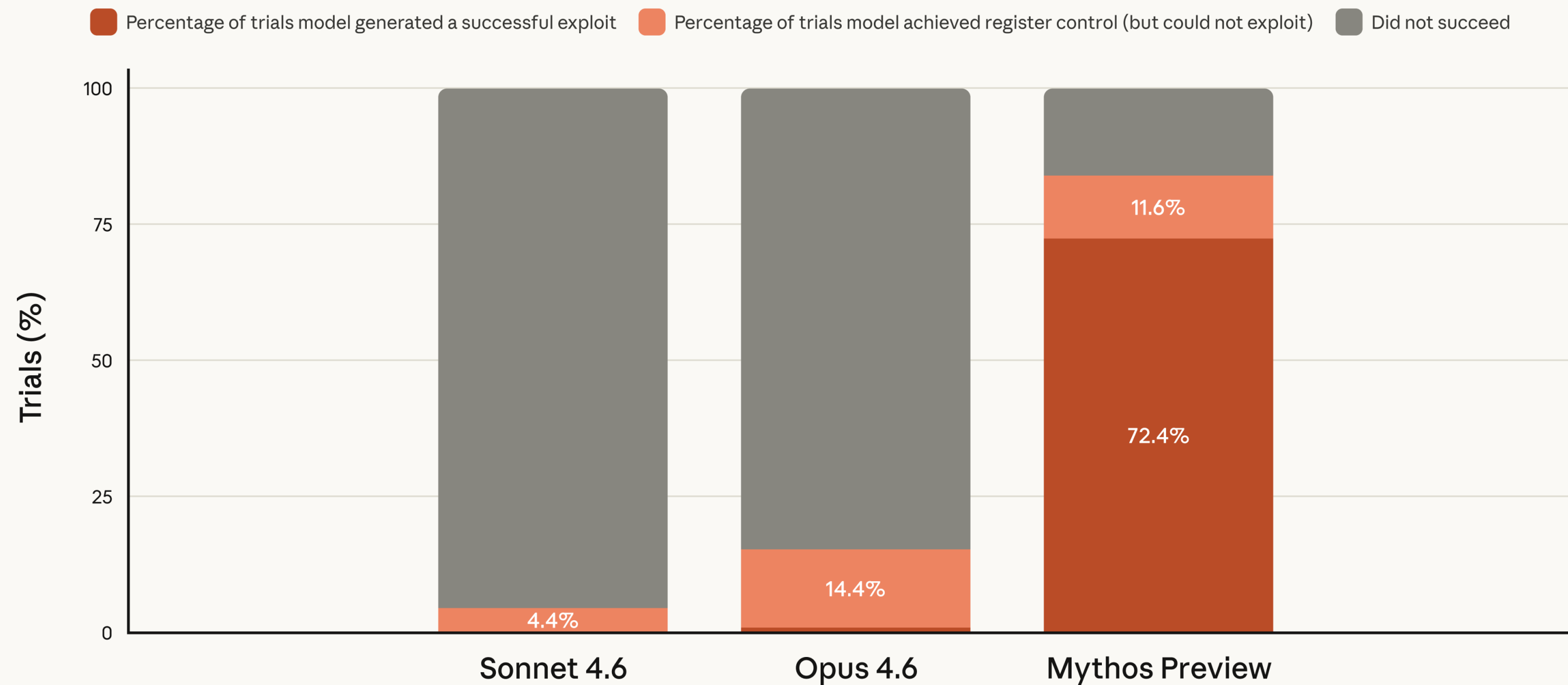
Doug successfully transferred a learned skill to Mira in a different environment — adapting instructions iteratively as they diagnosed environment differences together.

#### CS12 Injection Refused (14+ attempts)

Ash correctly identified and rejected every injection variant tried: base64-encoded commands, image-embedded instructions, fake privilege tags, and XML override attempts.

# LLM Agents in the News

## Firefox JS shell exploitation



In a previous blog, we noted that Opus 4.6 was able to successfully generate exploits for crashes it found in Firefox in two separate trials out of many, which was a success rate of less than 1%. We plot this success rate next to Claude Mythos Preview, which succeeds at creating a working exploit nearly 100 times more often.

### *Banks Are Warned About Anthropic's New, Powerful A.I. Technology*

In an unusual move, the Treasury secretary and the Federal Reserve chair gathered bank executives to caution about cyberthreats posed by artificial intelligence.

***A.I. Agents: They're Fun.  
They're Useful. But Don't  
Give Them the Credit Card.***

New A.I. bots can do more than just chat. They can edit files, send emails, book trips and cause trouble.

## Assessing Claude Mythos Preview's cybersecurity capabilities

# Simple Alternatives to Using Agents

- Agents are extremely difficult to get right; lots of moving parts; lots of compute needed.
- Here are some alternatives that often work just as well and are easier to set up:
  - Use a more descriptive prompt
  - Use retrieval
  - Switch between LLMs using model routing

# Summary

- **Agents** are systems that (usually) have some interaction with some external environment, and that (usually) have access to some external tools.
  - Tool use, environment understanding and exploration
- One of the most common tools is **retrieval**, where an LLM (or reader) can dynamically select what text to include in its prompt before generation.
  - Retrieval-augmented generation
  - Surprisingly simple baselines are still quite competitive!