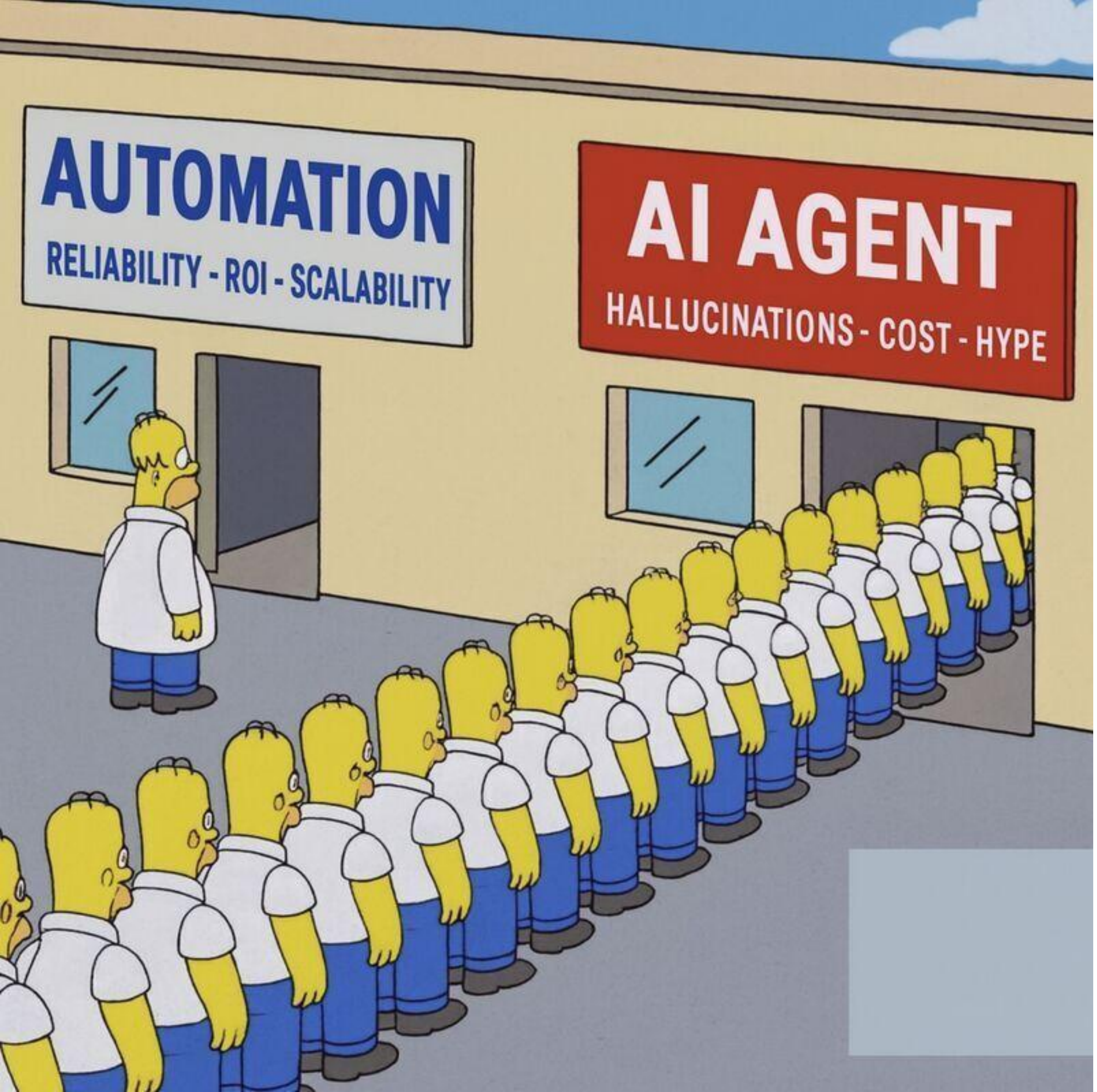
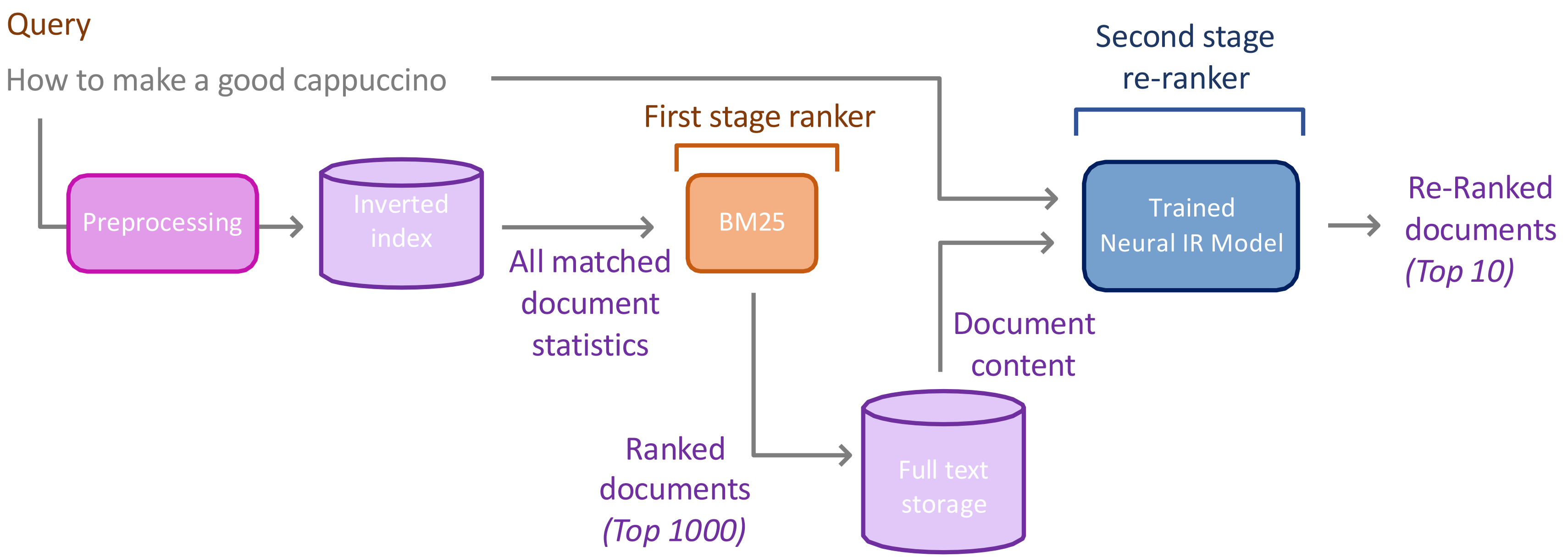


RAG and Reasoning Agents

Based on Shunyu Yao's slides

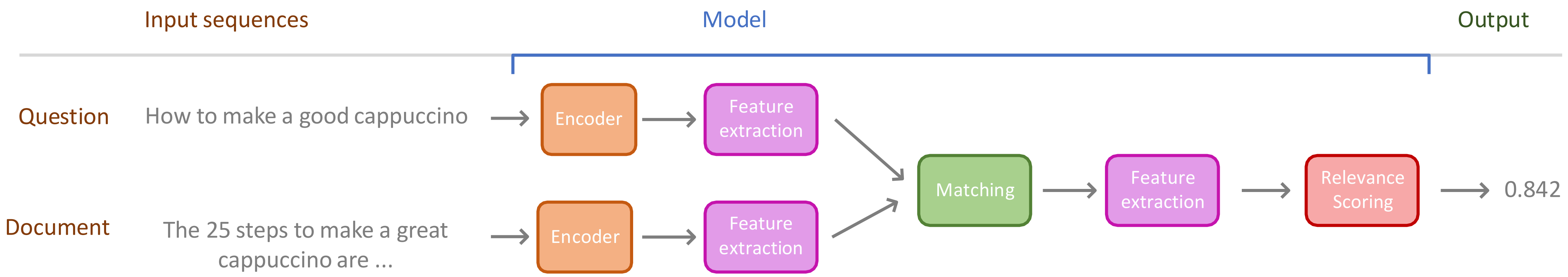
Neural Re-Ranking Models

- Re-rankers: They change the ranking of a pre-selected list of results
  - Same interface as classical ranking methods: $\text{score}(q, d)$
- Query Workflow:



Inside Neural Re-ranking Models

- Core part of re-ranking models is a matching module
  - Operating on a word interaction level



Training (same as Dense Reteival)

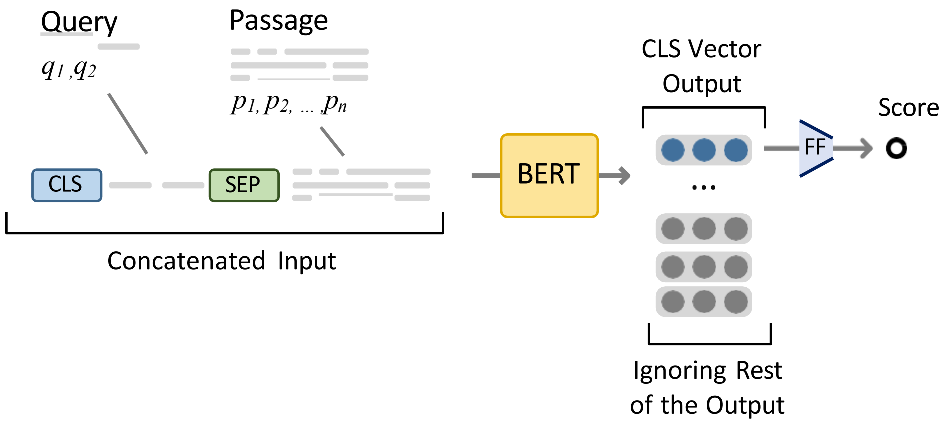
- Training is independent from the rest of the search engine operations
  - But it could be done repeatedly to account for temporal shift in the data
- Neural IR models are typically trained with triples (pairwise +,-)
  - Triple: 1 query, 1 relevant, 1 non-relevant document
  - Generate embeddings for query, relevant doc, non-relevant doc
  - Loss function: Maximize margin between rel/non-rel document
- All model components are trained end-to-end
  - Of course we could decide to freeze some parts for more efficient training

Re-Ranking Evaluation

- Scoring per tuple (1 query, 1 document)
- List of tuples is then sorted & evaluated with ranking metric per query (for example: MRR@10)
  - MRR@10 = Mean Reciprocal Rank, stop to look at position 10 or first relevant

- Mismatch: You can't really compare training loss and IR evaluation metric
  - Training loss is only good for checking at the beginning if your network is not completely broken :) – it should go down very quick and then not change

BERT Re-Ranking: BERT_{CAT}



- Also know as monoBERT, vanilla BERT re-ranking, or simply BERT
- Concatenating the two sequences to fit BERT's workflow
  - [CLS] query [SEP] passage
  - Pool [CLS] token
  - Predict the score with a single linear layer
- Needs to be repeated for every passage

Passage Re-ranking with BERT. Rodrigo Nogueira, Kyunghyun Cho. 2019
https://arxiv.org/abs/1901.04085

BERT_{CAT}

- Simple formula (as long as we abstract BERT):

BERT

$$r = \text{BERT}(\underline{[CLS]; q_{1..n}; [SEP]; p_{1..m}})_{CLS}$$

Concatenation

Scoring

$$s = r * \underline{W}$$

Starts uninitialized

- We still have the choice of BERT-model

$q_{1..n}$	Query tokens
$p_{1..m}$	Passage tokens
BERT	Pre-trained BERT model
[CLS] [SEP]	Special tokens
x_{CLS}	Pool the CLS vector
W	Linear Layer (from 768 dims to 1)
s	Output score

The Impact of BERT_{CAT}

- This model (first shown by Nogueira and Cho) jumpstarted the current wave of neural IR 🎉
- Works awesome out of the box
  - Concatenating the two sequences to fit BERT's workflow
  - As long as you have time or enough compute it trains easily
- Major jumps in effectiveness across collections and domains
  - But, of course, comes at the cost of performance and virtually no interpretability
  - Larger BERT models roughly translate to slight effectiveness gains at high efficiency cost
    - The problem is we need to repeat the inference by the re-ranking depth!

So how good is $BERT_{CAT}$?

- MSMARCO-Passage
  - MRR@10 from .194 (BM25) to .385 (ALBERT-Large)
  - BERT basically doubles the result quality

- MSMARCO-Document
  - MRR@10 from .252 (BM25) to .384 (DistilBERT with 2K tokens)

- Similar results for TREC-DL ’19, ’20, TripClick
  - Large Training Data Settings

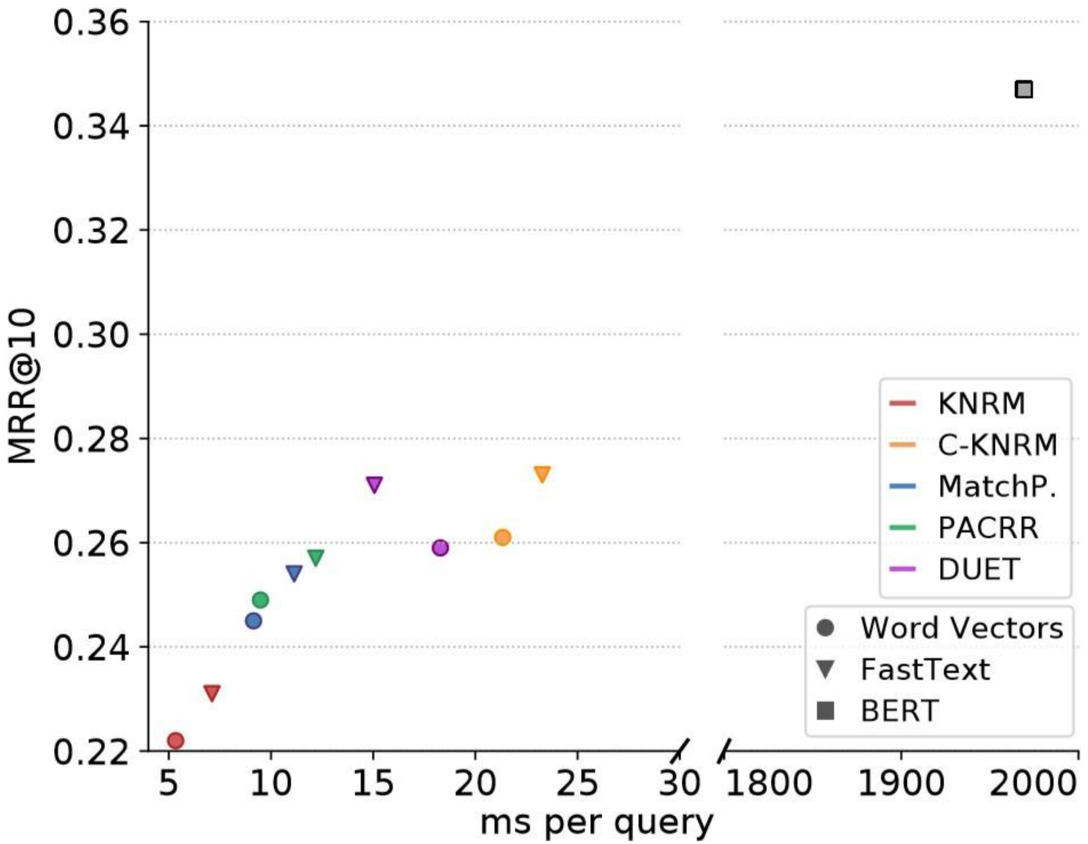
See also the (retired) MSMARCO leaderboard: https://microsoft.github.io/msmarco/
Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation; Hofstätter et al. https://arxiv.org/abs/2010.02666

$BERT_{CAT}$ for Longer Documents

- BERT is capped at max. 512 input tokens (query + document)
- Simplest solution: just cap the document at 512-query length
  - Works surprisingly well already (for MSMARCO-Documents)
  - But might not work well in other domains, where documents are really long, or contain a variety of topics at different depths
- Still simple, but working on full documents: Sliding window over the document -> take max window score as document score
  - Now, we can also make smaller sliding windows
    - Might be useful to use in the UI -> highlight the most relevant passage as snippet

Deeper Text Understanding for IR with Contextual Neural Language Modeling. Zhuyun Dai, Jamie Callan
https://arxiv.org/abs/1905.09217

$BERT_{CAT}$ In-Efficiency



- Evaluated on 250 docs / query on short MSMARCO-Passage *(max 200 tokens)*
- Basic IR-specific networks are fast, but moderately effective
- Transformer-based BERT is very effective, but very slow
  - + Infrastructure cost (blocking 1 GPU for 2 seconds at a time)

Sebastian Hofstätter and Allan Hanbury. 2019.
Let's measure runtime! Extending the IR replicability infrastructure to include performance aspects . In OSIRRC @ SIGIR.

Achieving Efficiency

- Multiple paths to reduce query latency
  - Query latency is our focus today, but full lifecycle efficiency also a concern
  - Lifecycle efficiency includes training, indexing and retrieval steps

❶ Reduce model size

- Smaller models run faster, duh!
- Only possible until a certain threshold after which quality reduces drastically

❷ Move computation away from query-time

- Pre-compute passage representation, so they become a simple lookup
- Lightweight aggregation, that can be done at query time

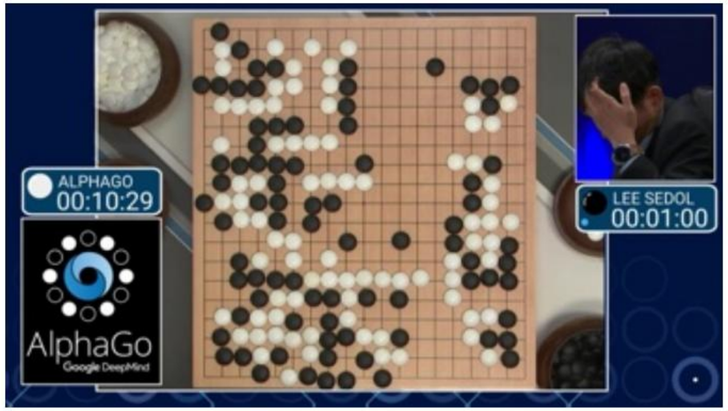
Agent

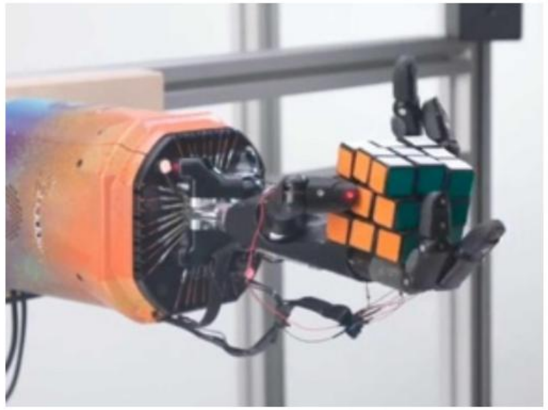
Outline

- What are LLM agents?
- A brief history of LLM agents
  - Under the context of “LLM”
  - Under their classic definition
- On the future of LLM agents

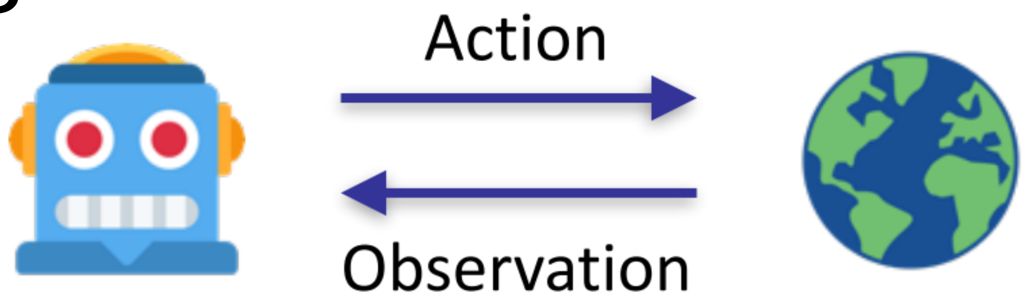
What is “Agent”

What is "Agent"



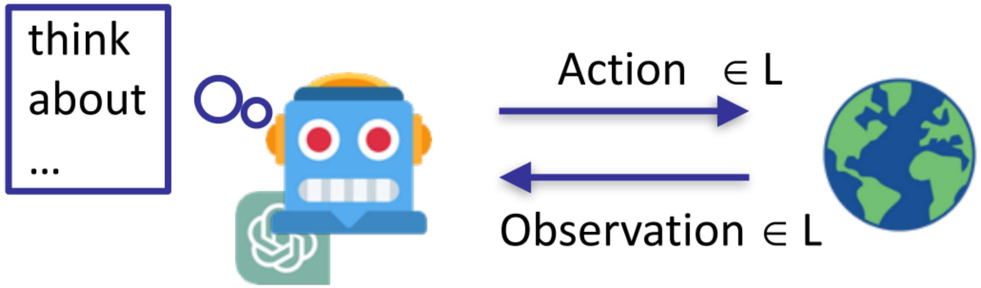


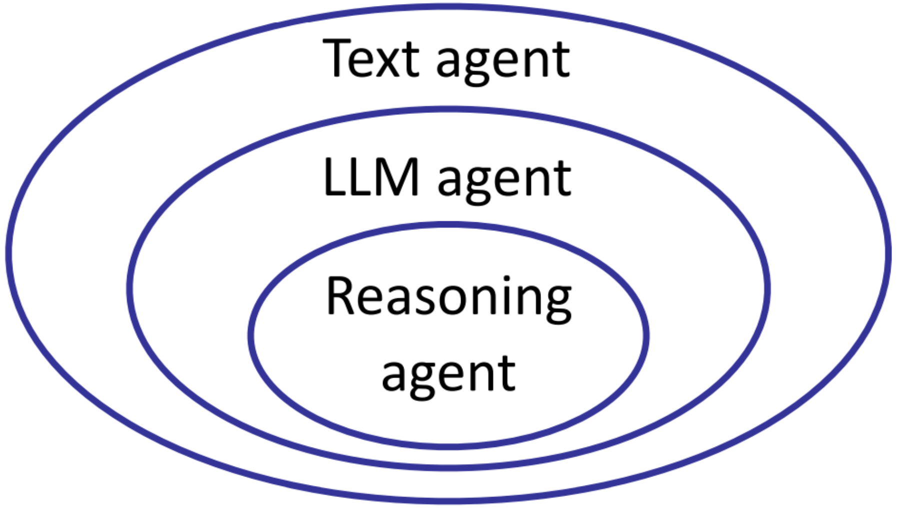
What is "Agent"?



- An "intelligent" system that interacts with some "environment"
  - Physical environments: robot, autonomous car, …
  - Digital environments: DQN for Atari, Siri, AlphaGo, …
  - Humans as environments: chatbot
- Define "agent" by defining "intelligent" and "environment"
  - It changes over time!
  - Exercise question: how would you define "intelligent"?

What is an “LLM Agent”





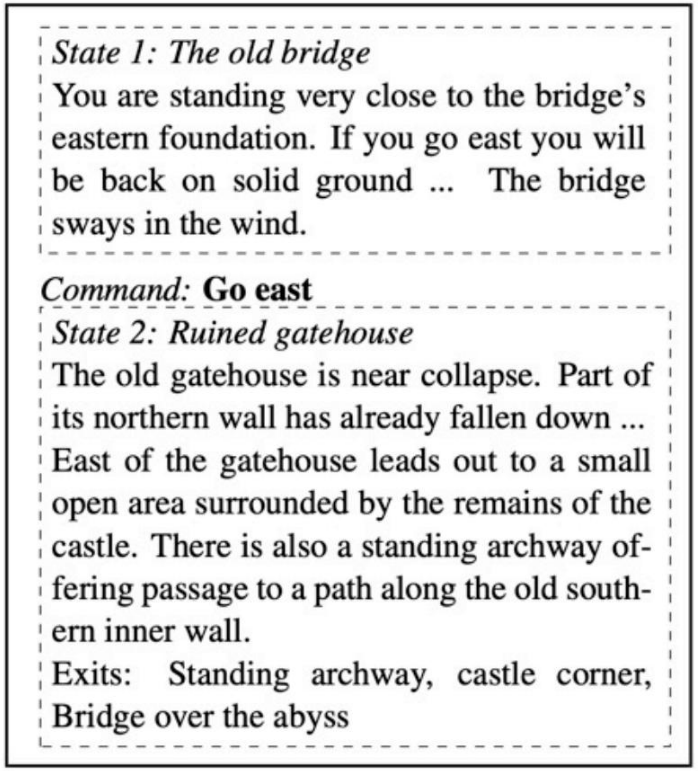
- Level 1: Text agent
  - Uses text action and observation
  - Examples: ELIZA, LSTM-DQN
- Level 2: LLM agent
  - Uses LLM to act
  - Examples: SayCan, Language Planner
- Level 3: Reasoning agent
  - Uses LLM to reason to act
  - Examples: ReAct, AutoGPT
  - The key focus of the field and this lecture

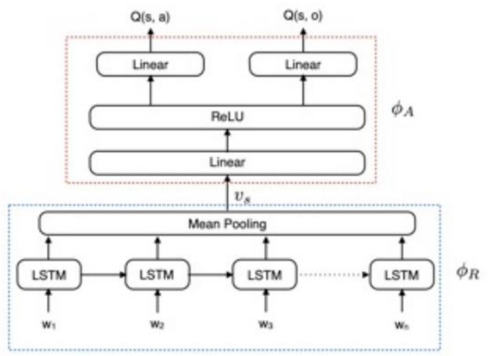
ELIZA (1966): Text agent via rule design

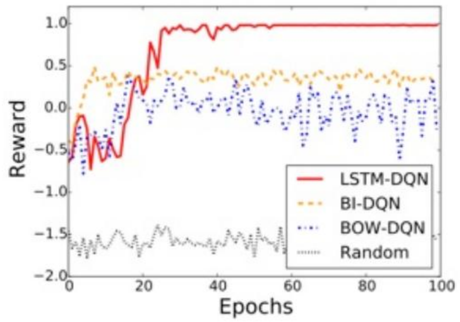
```
  Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

- Domain specific!
- Requires manual design
- Cannot work beyond simple domains

LSTM-DQN (2015): Text agent via RL



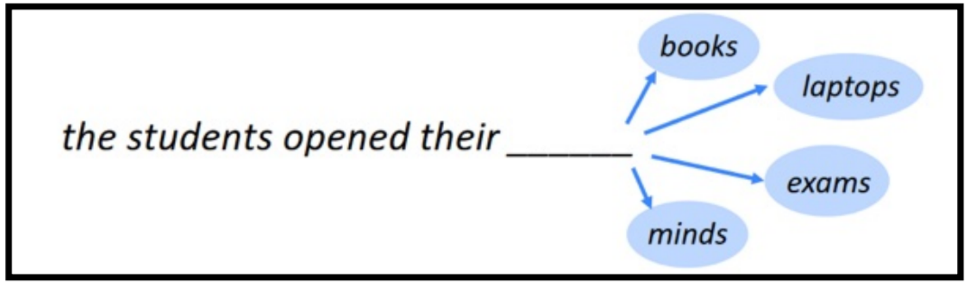




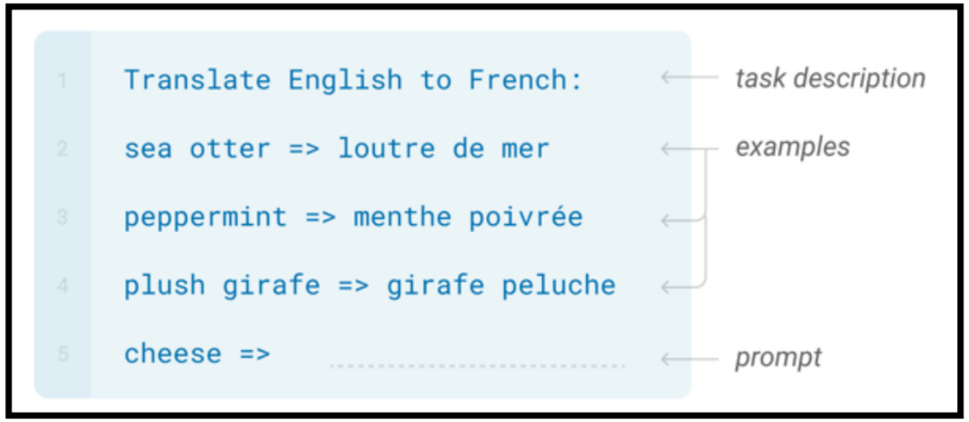
- Domain specific!
- Requires scalar reward signals
- Requires extensive training

Language understanding for text-based games using deep reinforcement learning. EMNLP 2015.

The promise of LLMs: Generality and few-shot learning



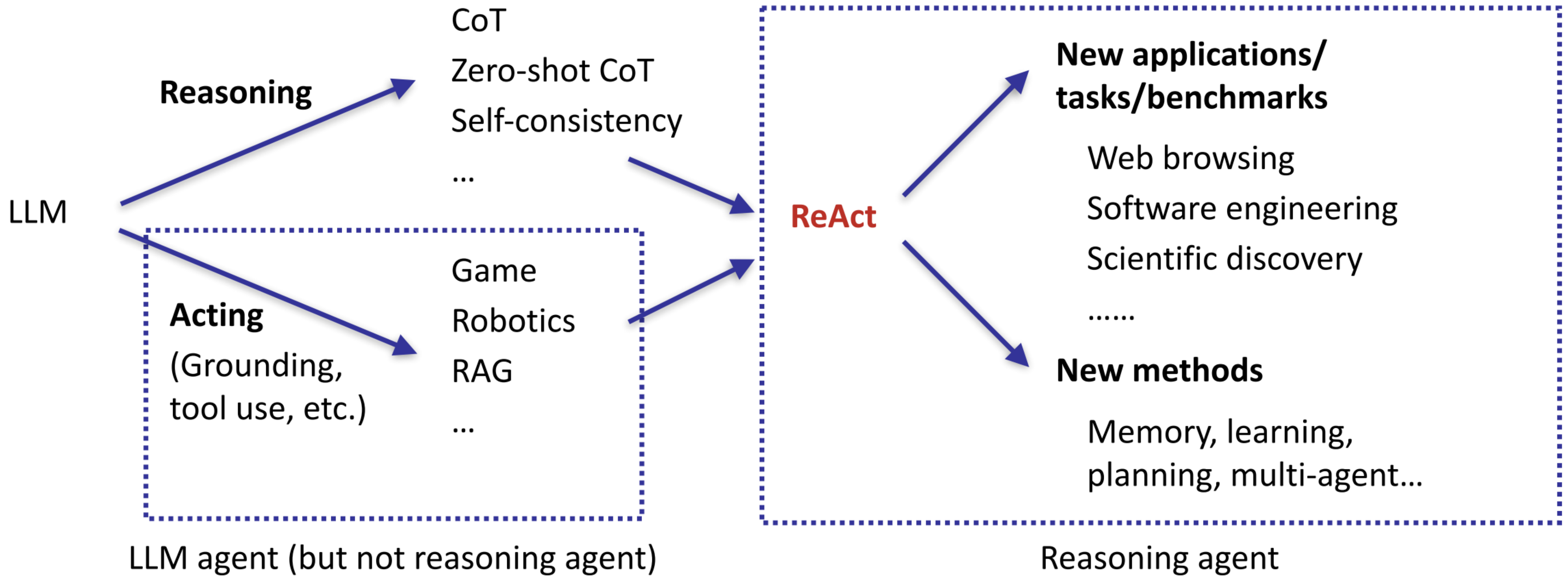
Training: next-token prediction on massive text corpora



Inference: (few-shot) prompting for various tasks!

Language Models are Few-Shot Learners NeurIPS 2020.

A brief history of LLM agents



Question Answering (QA)

Question Answering

Q: what is 1 + 2? → A: 3

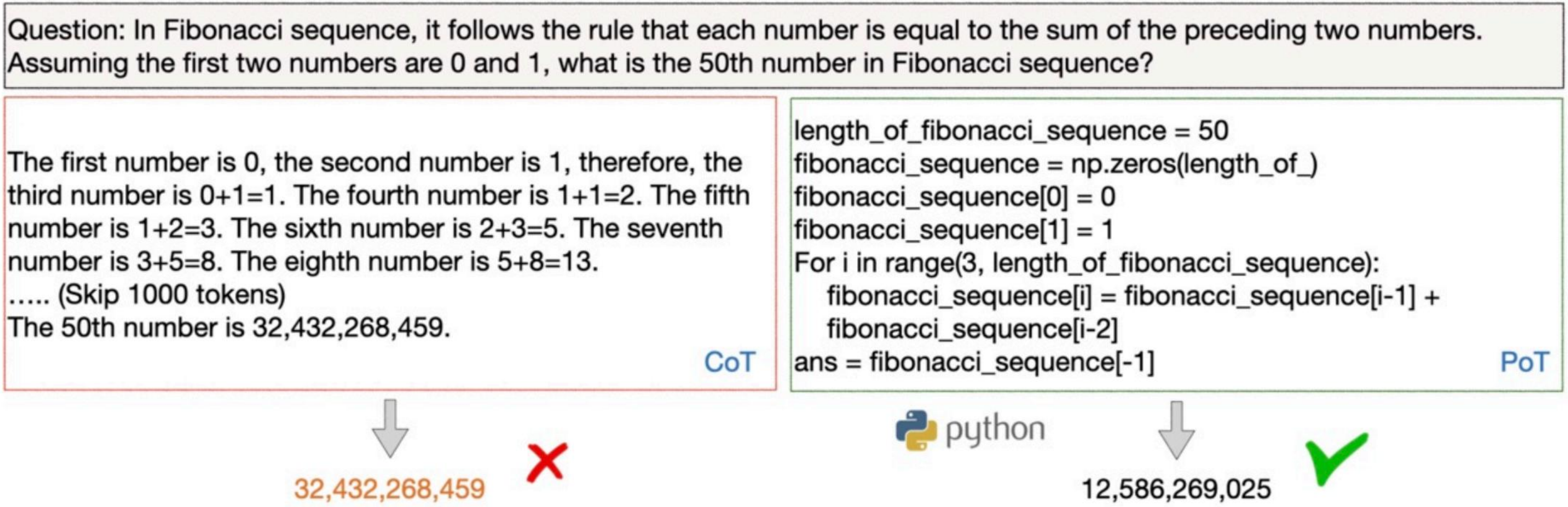
Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day? → Requires reasoning

Q: who is the latest UK PM? → Requires knowledge

Q: what is the prime factorization of 34324329? → Requires computation

People came up with various solutions for different QA tasks.

Code Augmentation for Computation



Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

Retrieval-Augmented Generation (RAG)

- Sounds fancy, but actually very simple.
- RAG:
  - Step 1: retrieve N documents using some IR algorithm.
  - Step 2: write the augmented query.

```
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and not
prior knowledge, answer the query.
Query: {query_str}
Answer:
```

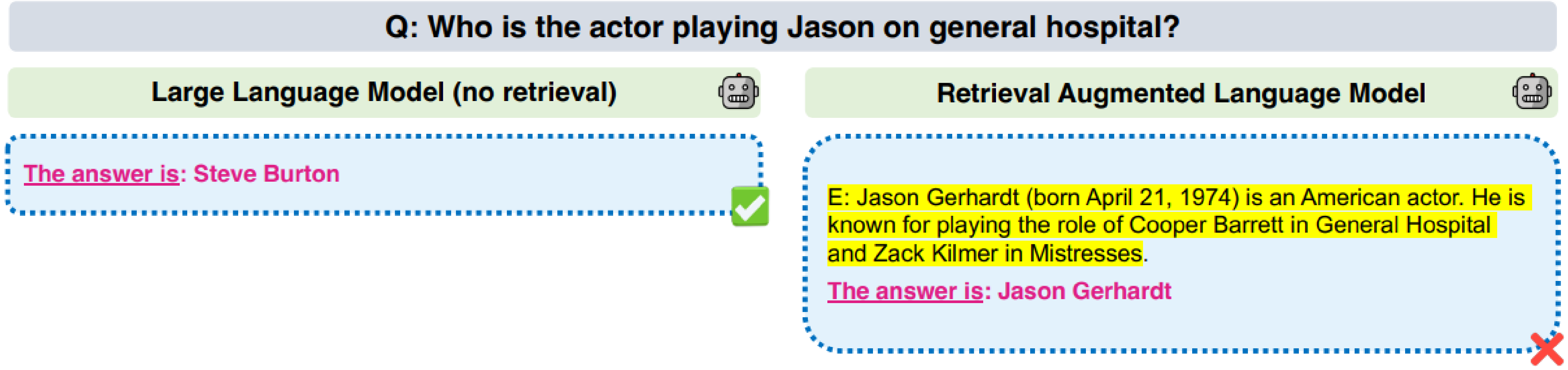
  - Step 3: Profit.

Prompt templates:

https://github.com/run-llama/llama_index

Distraction in RAG

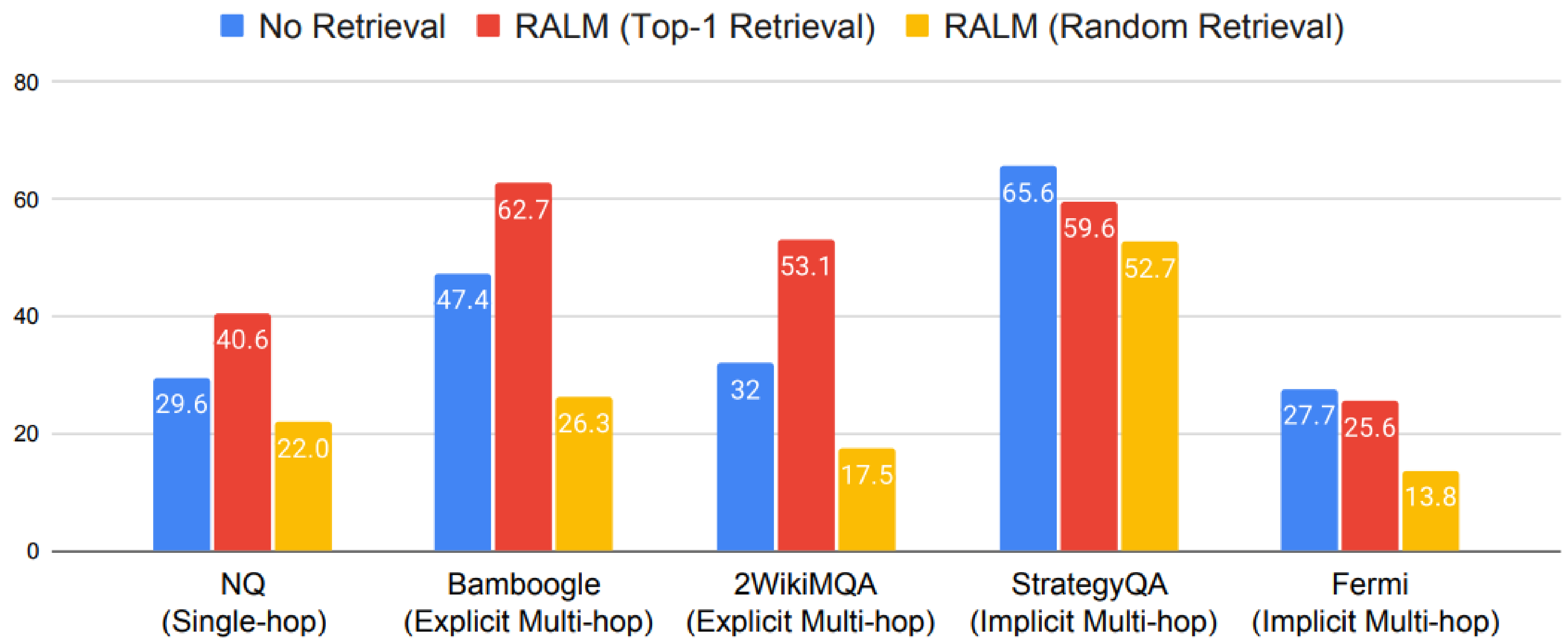
- Distraction:
  - When a piece of irrelevant context is provided, the model generates an incorrect response.



  - **Relevant** refers to whether the correct answer is in the prompt or not.

Yoran et al. (2024) Making Retrieval-Augmented Language Models Robust to Irrelevant Context。

Distraction in RAG



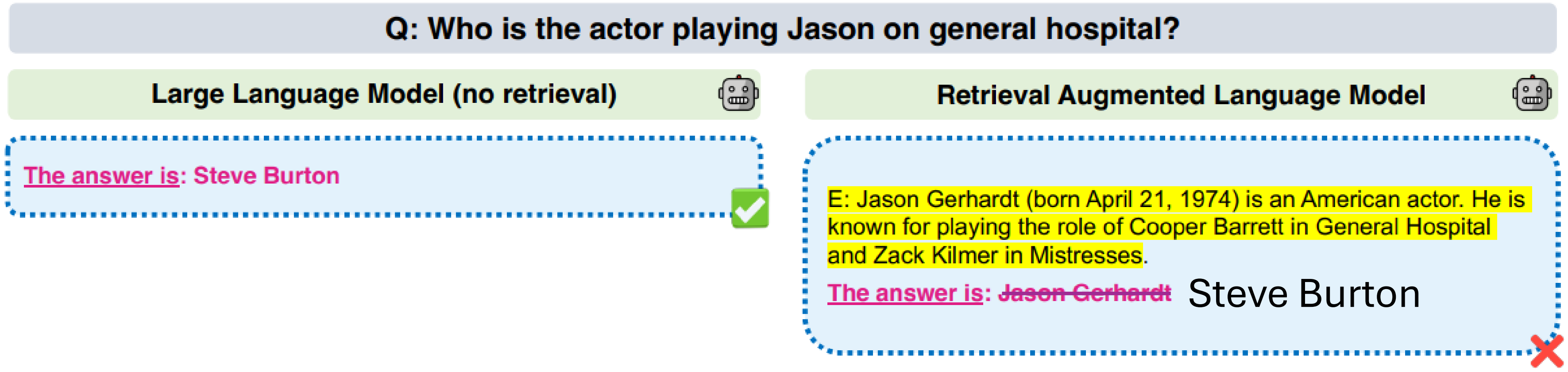
Yoran et al. (2024) Making Retrieval-Augmented Language Models Robust to Irrelevant Context

Solution #1: Use NLI to filter irrelevant context

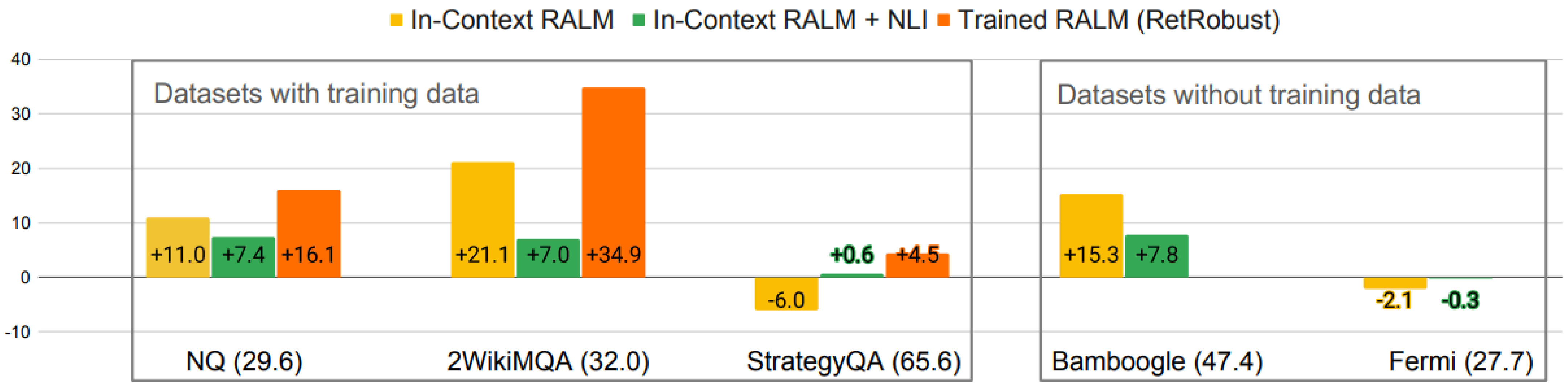
- Review: NLI models
  - Premise:
    - *If you help the needy, God will reward you.*
  - Hypotheses:
    - *Giving money to a poor man has good consequences.* Entailment
    - *Giving money to a poor man has no consequences.* Contradiction
    - *Giving money to a poor man will make you a better person.* Neutral
- NLI against distraction:
  - Remove context sentence if it contradicts the question.

Solution #2: Finetuning

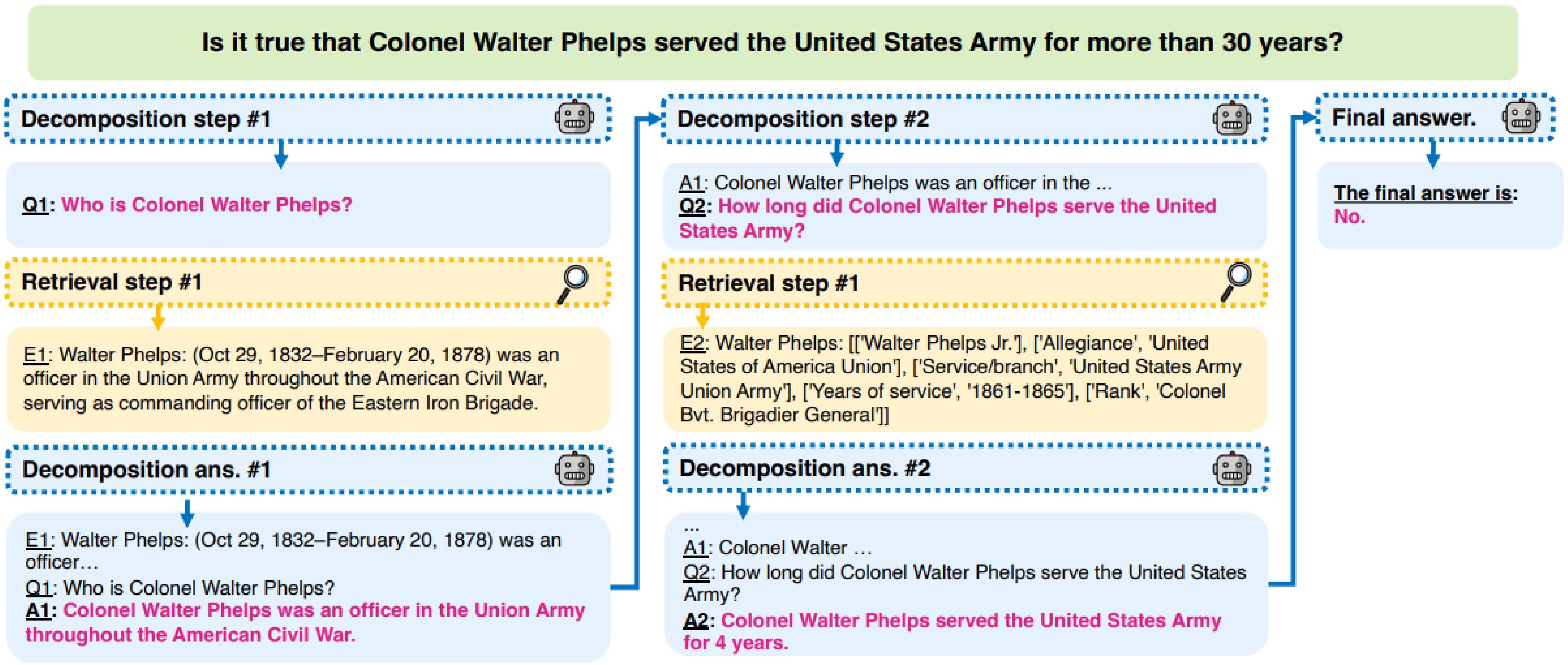
- Fine-tune the LM with:
  - Both relevant and irrelevant contexts



Distraction in RAG: Mitigation Result

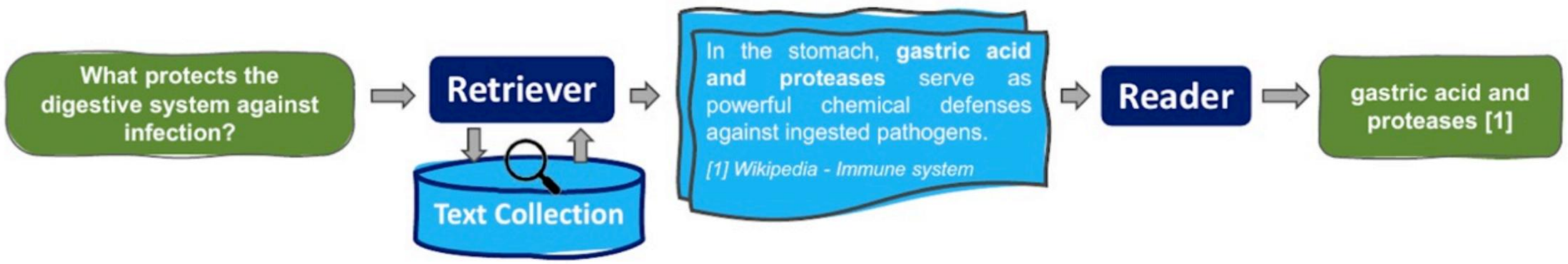


Solution #3: Interleaving decomposition



Retrieval-augmented generation (RAG)

- Answer knowledge-intensive questions with
  - Extra corpora
  - A retriever (e.g., BM25, DPR, etc.)
- What if there's no corpora? (e.g. who's the latest PM?)



Tool Use

- Special tokens to invoke tool calls for
  - Search engine, calculator, etc.
  - Task-specific models (translation)
  - APIs
- Unnatural format requires task/tool-specific fine-tuning
- Multiple tool calls?

A weather task:
how hot will it get in NYC today? |*weather* lookup region=NYC |*result* precipitation chance: 10, high temp: 20c, low-temp: 12c |*output* today's high will be 20C

TALM: Tool Augmented Language Models.

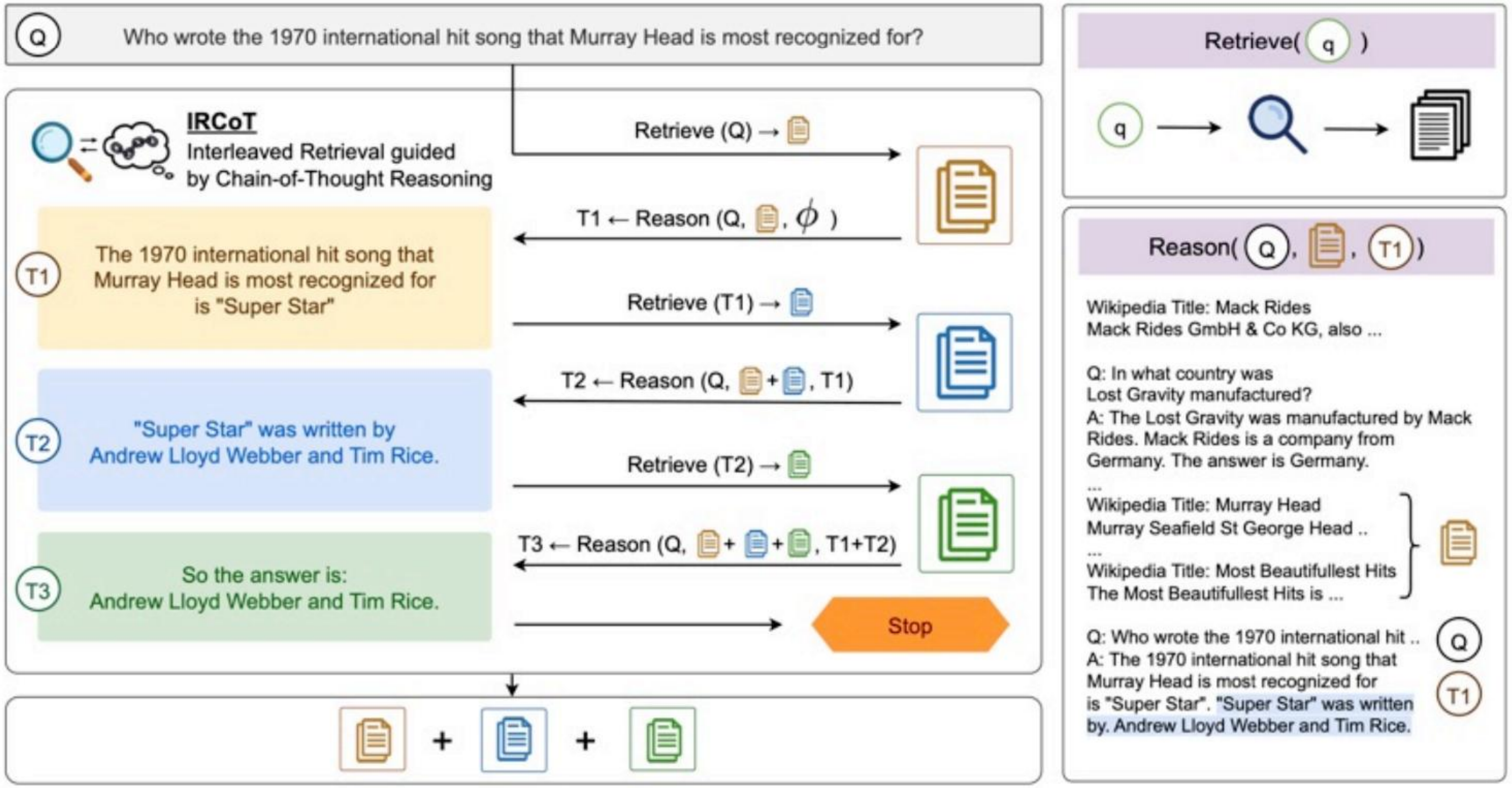
Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

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

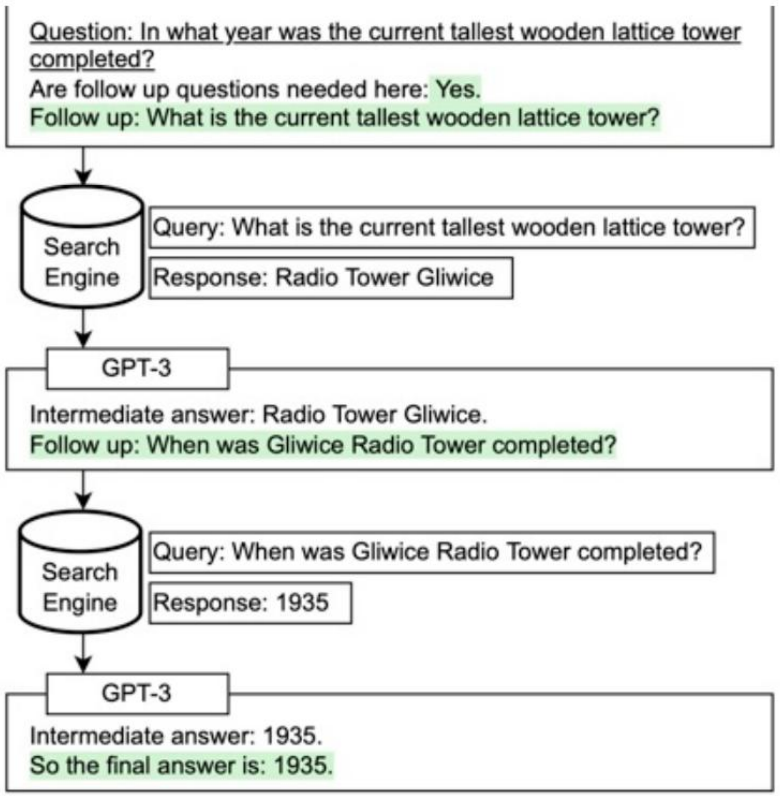
The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their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

Toolformer: Language Models Can Teach Themselves to Use Tools

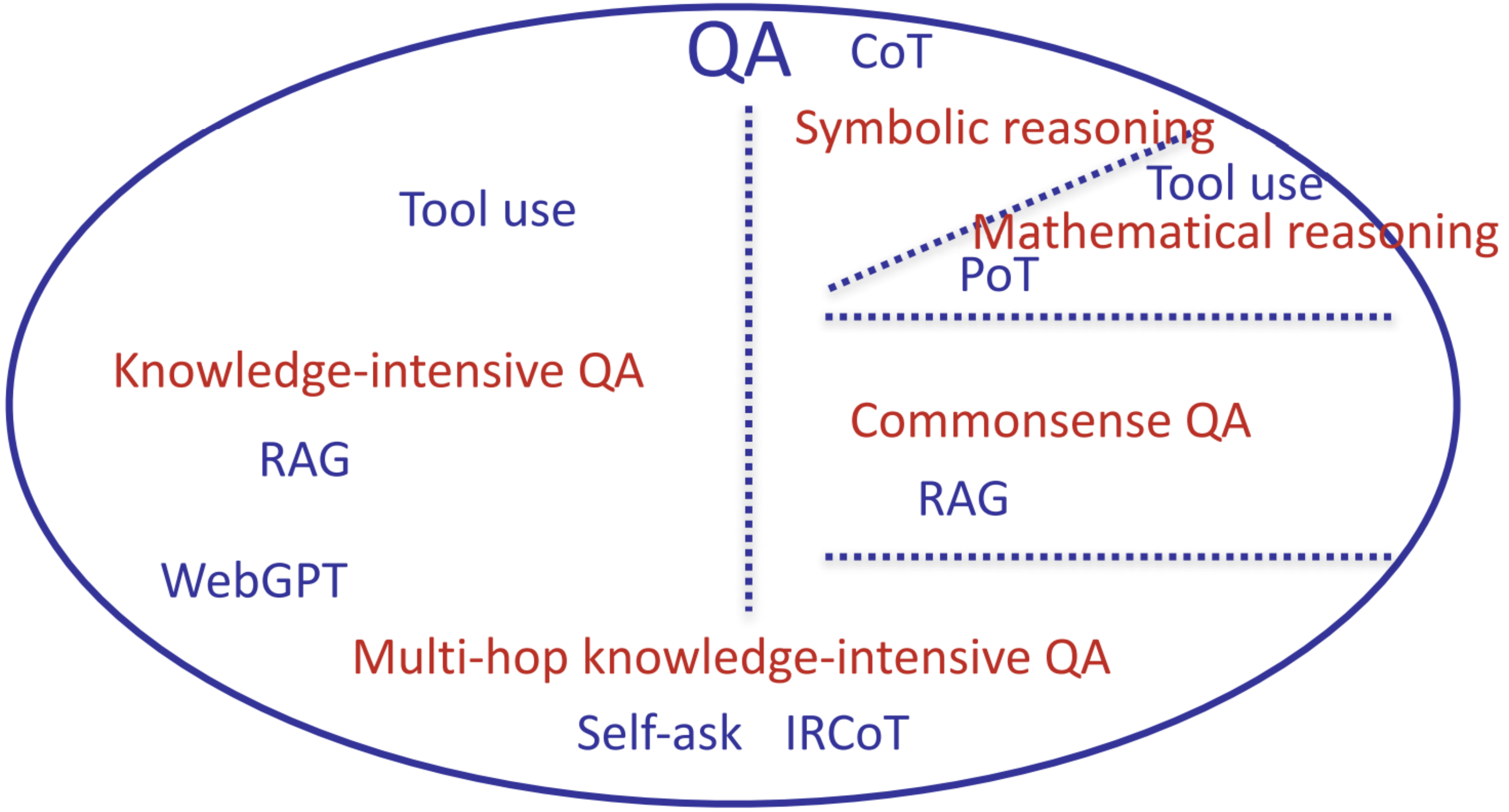
What if both knowledge and reasoning are needed?



Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions



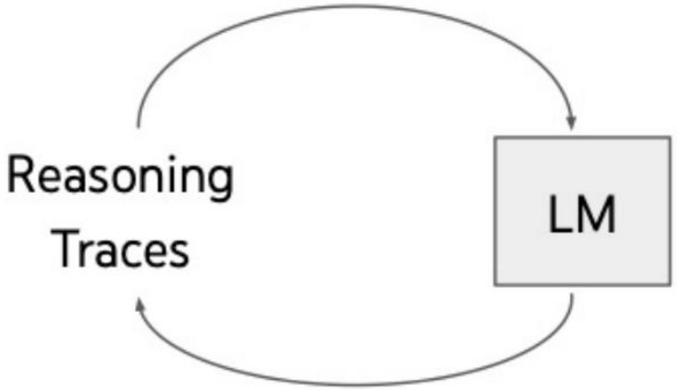
Measuring and Narrowing the Compositionality Gap in Language Models.



Can we have a simple, unifying solution?
We need abstraction.

Reasoning OR acting

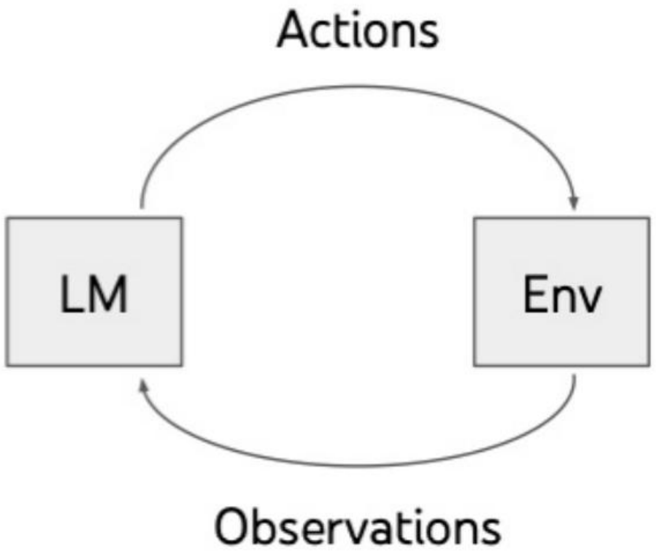
CoT



Flexible and general to augment test-time compute

Lack of external knowledge and tools

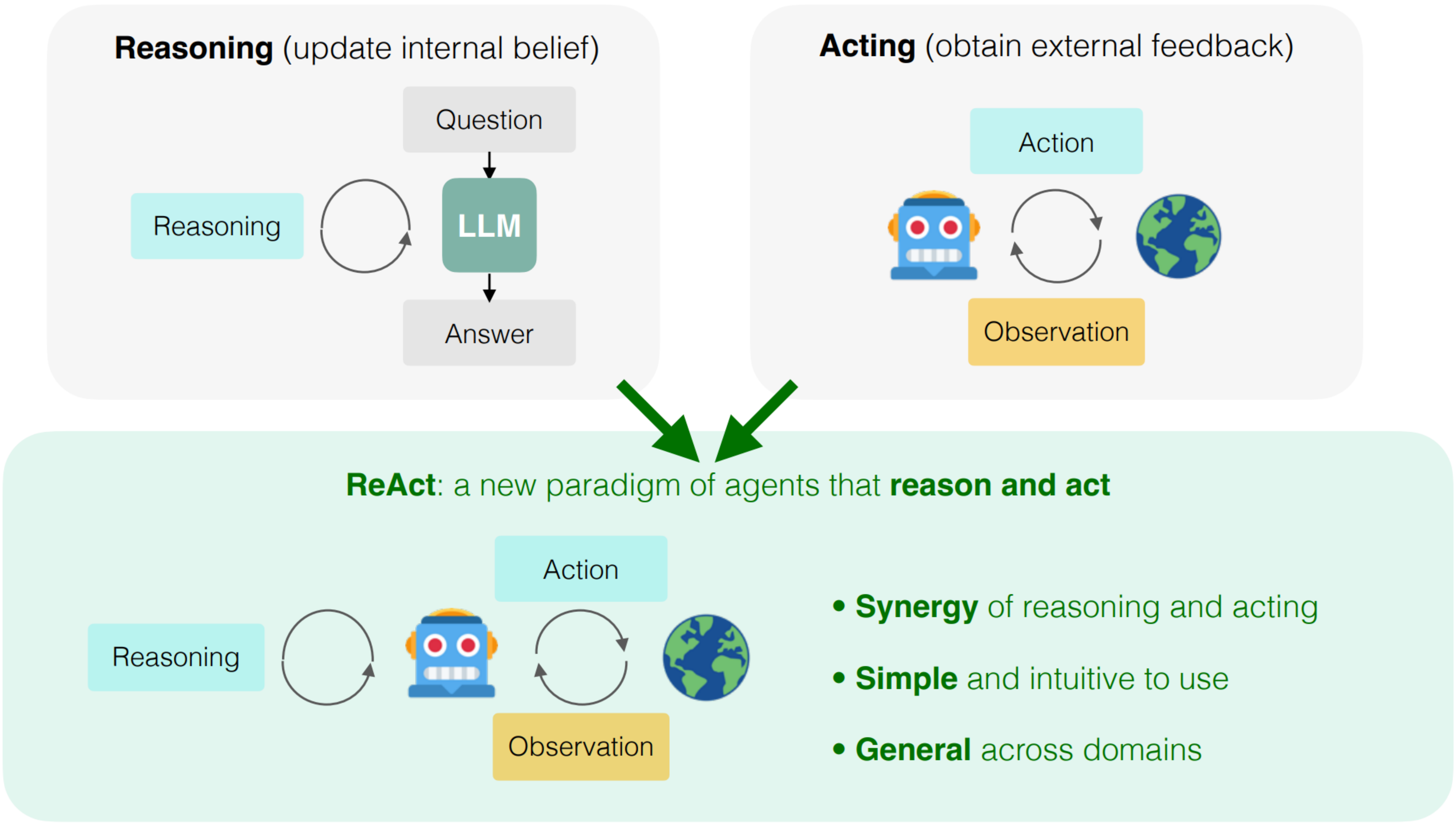
RAG/Retrieval/Code/Tool use



Retrieval
Search engine
Calculator
Weather API
Python
……

Lack of reasoning

Flexible and general to augment knowledge, computation, feedback, etc.


Reasoning (update internal belief)
Question
Reasoning
LLM
Answer
Acting (obtain external feedback)
Action
Observation
ReAct: a new paradigm of agents that reason and act
Action
Reasoning
Observation
• Synergy of reasoning and acting
• Simple and intuitive to use
• General across domains

ReAct is simple and intuitive to use

ReAct supports:

- One-shot prompting
- Few-shot prompting
- Fine-tuning

A zero-shot ReAct prompt

You are an agent that answers questions by using two actions:
- search[query]: Google search the query. You can also use it to calculate math.
- finish[answer]: return the answer

Your generation should have the following format:
Thought: to analyze the observation and inform the action.
Action: your action

Question: If I have seven trillion dollars today, can I buy Apple, Nvidia, and Microsoft? If not, how much more money do I need?

LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: "Nvidia's market cap stood at $1.832 trillion as of about 10 a.m. ET Wednesday, while Alphabet's was at $1.814 trillion, bringing Nvidia behind just Microsoft (MSFT) at $3.015 trillion and Apple (AAPL) at $2.836 trillion."

(Mar 2024)

Thought: Now that I have the market cap for all companies, let's add them together and see if $7 trillion is enough.
Action: search["1.832 trillion + 3.015 trillion + 2.836 trillion"]

Observation: 7683000000000

Thought: The amount of money needed in addition to the $7 trillion is $683 billion.
Action: finish["You need an additional $683 billion to be able to buy Apple, Nvidia, and Microsoft."]

✓

LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: nothing found!

(My adversarial observation)

Thought: The search did not give any results. I will need to search for each company's market capitalization individually.
Action: search["current market capitalization of Apple"]

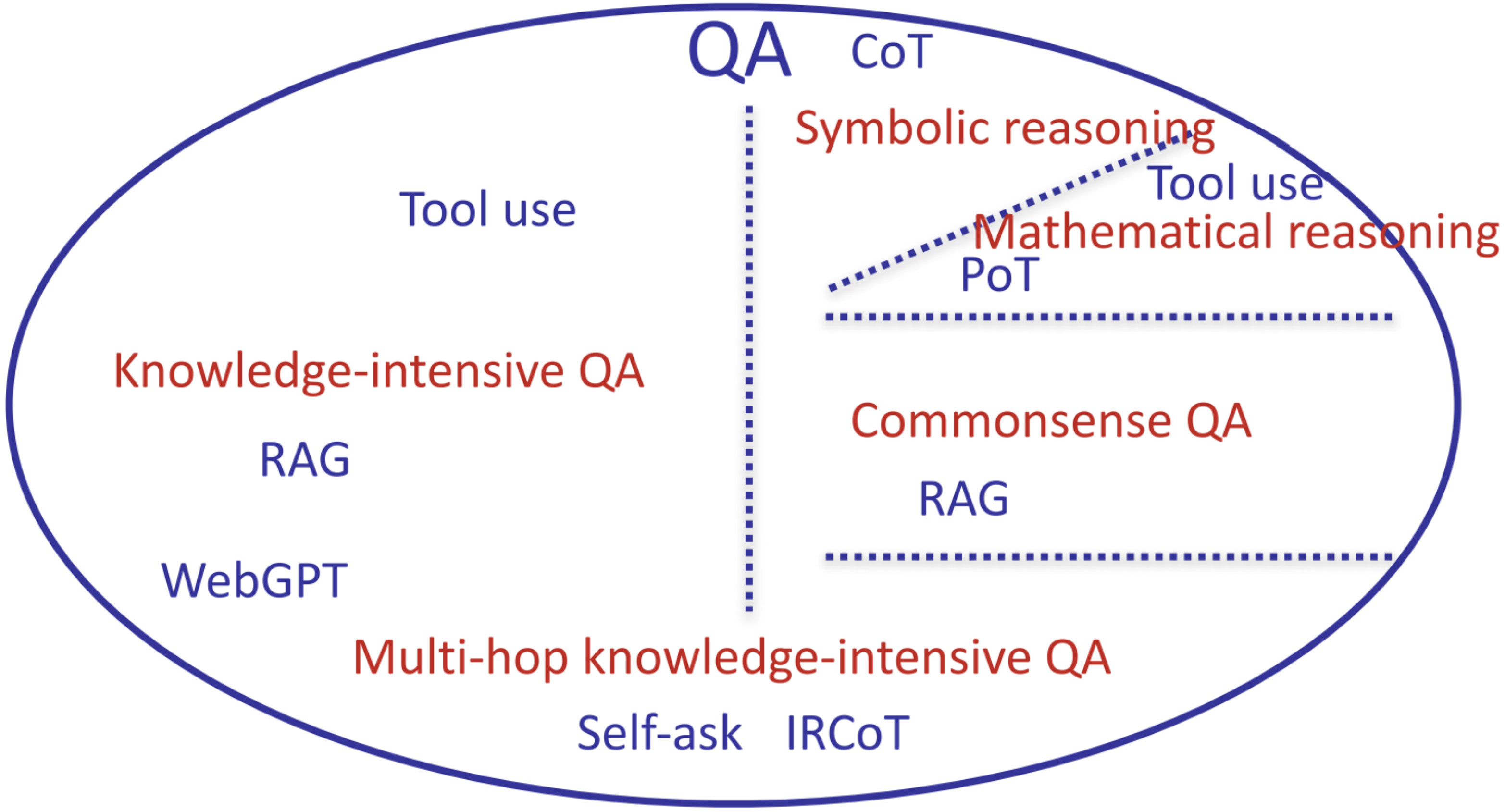
Observation: 'price': 180.75

(My adversarial observation)

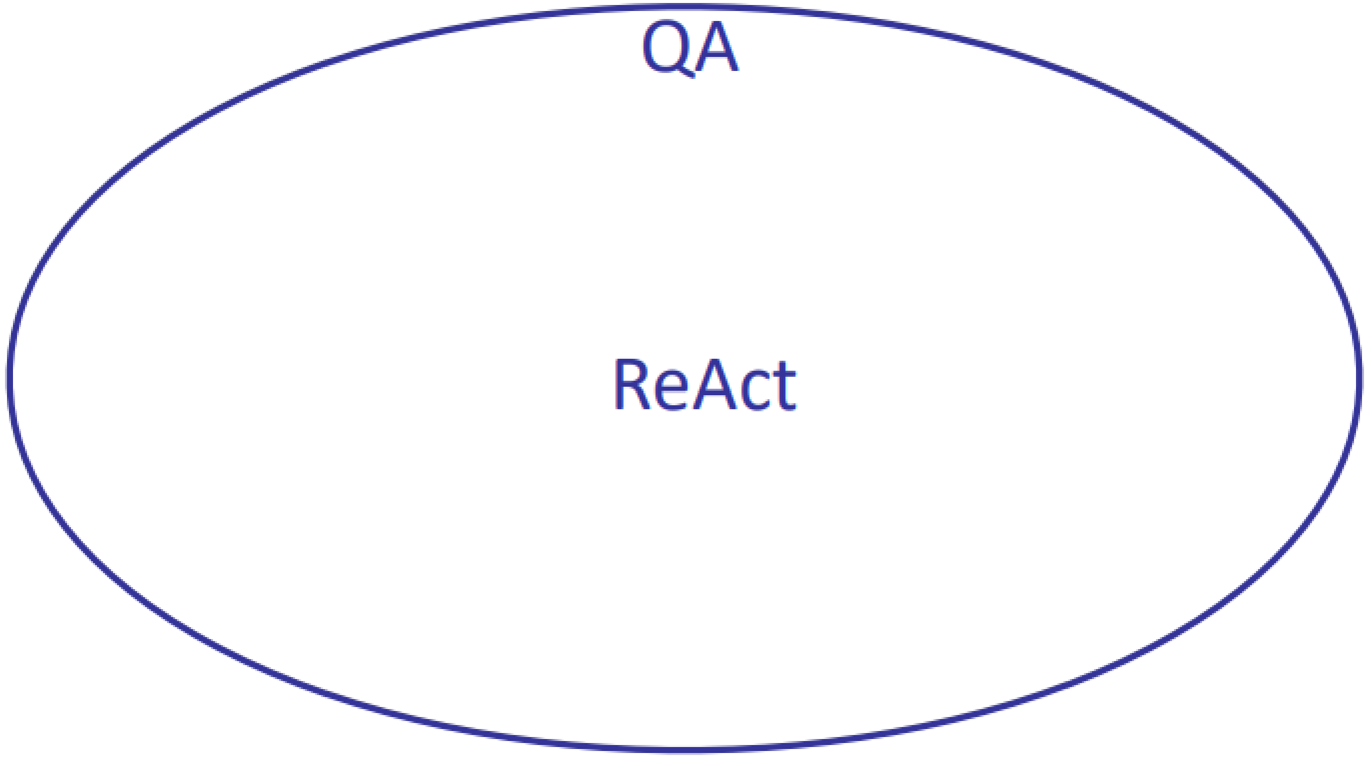
Thought: This is the current price of Apple's stock, not its market capitalization. The market cap is calculated by multiplying the share price by the number of outstanding shares. Thus, I need to find the total number of Apple's outstanding shares to calculate its market cap.
Action: search["total number of Apple's outstanding shares"]

✓ (Continues to solve the task)

Synergy: acting support reasoning, reasoning guides acting.



Can we have a simple, unifying solution?



More than QA?

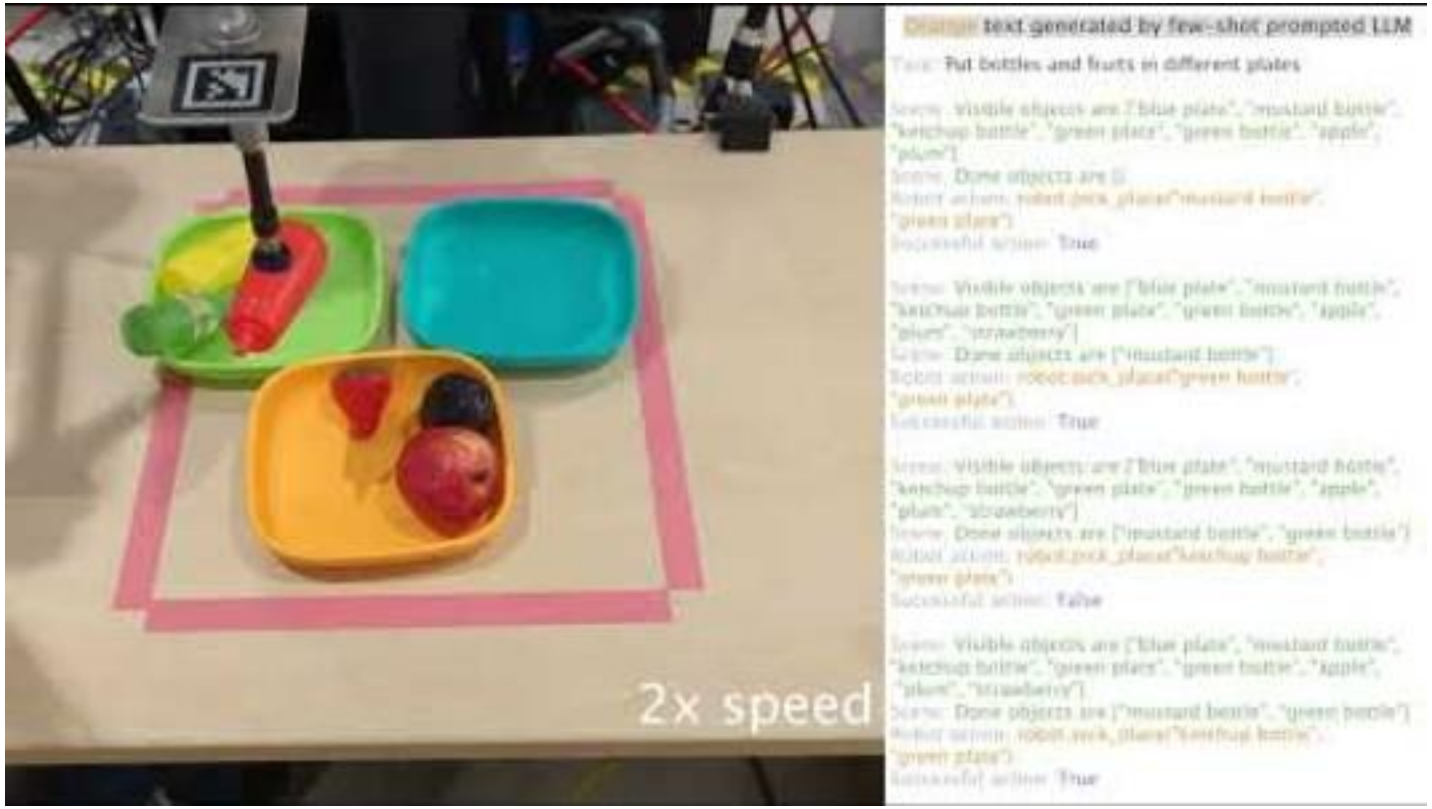
Many tasks can be turned into text games

```
You are in the middle of a room. Looking quickly around you, you see a drawer 2, a shelf
5, a drawer 1, a shelf 4, a sidetable 1, a drawer 5, a shelf 6, a shelf 1, a shelf 9, a
cabinet 2, a sofa 1, a cabinet 1, a shelf 3, a cabinet 3, a drawer 3, a shelf 11, a shelf
2, a shelf 10, a dresser 1, a shelf 12, a garbagecan 1, a armchair 1, a cabinet 4, a shelf
7, a shelf 8, a safe 1, and a drawer 4.

Your task is to: put some vase in safe.

> go to shelf 6
You arrive at loc 4. On the shelf 6, you see a vase 2.

> take vase 2 from shelf 6
You pick up the vase 2 from the shelf 6.

> go to safe 1
You arrive at loc 3. The safe 1 is closed.

> open safe 1
You open the safe 1. The safe 1 is open. In it, you see a keychain 3.

> put vase 2 in/on safe 1
You won!
```

ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks.

ALFWorld: Aligning Text and Embodied Environments for Interactive Learning.

Acting without reasoning



Inner Monologue: Embodied Reasoning through Planning with Language Models

Acting without Reasoning

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

Act only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dishsponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishsponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens.
......

X

Cannot explore systematically or incorporate feedback

ReAct Enables Systematic Exploration

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

ReAct

Act 1: Think[First I need to find a peppe shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
(Here the agent goes to cabinet 1, 2, 3, 4, 5, 6, then countertop 1 and 2)

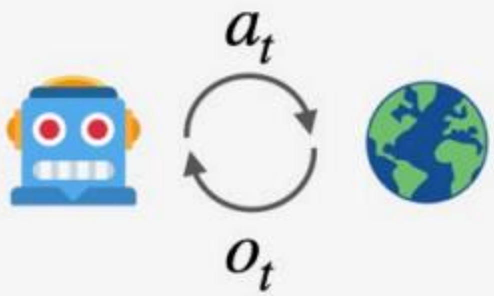
Act 9: Go to countertop 3
Obs 9: On the countertop 3, you see a apple 1, a bread 1, ... a pepper shaker 1, and a vase 3.
Act 10: Take pepper shaker 1 from countertop 3
Obs 10: You pick up the pepper shaker 1 from the countertop 3.
Act 11: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 12: Go to drawer 1
Obs 12: Drawer 1 is closed.
Act 13: Open drawer 1
Obs 13: You open Drawer 1 …
Act 14: Put pepper shaker 1 in/on drawer 1
Obs 14: You put pepper shaker 1 in/on the drawer 1.

✓

ReAct is general and effective

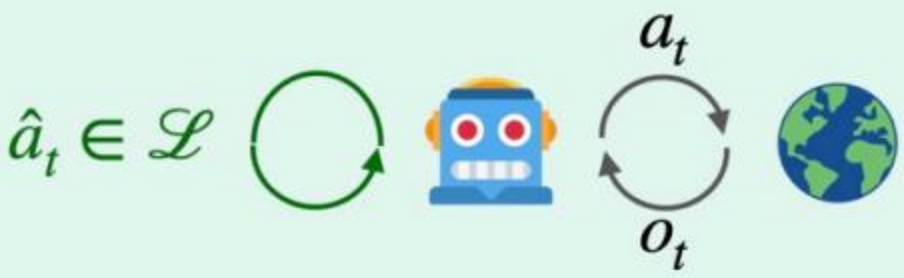
PaLM-540B	HotpotQA *(QA)* (NLP tasks)	FEVER *(fact check)* (NLP tasks)	ALFWorld *(Text game)* (RL tasks)
Reason	29.4	56.3	N/A
Act	25.7	58.9	45
ReAct	**35.1**	**64.6**	**71**

Traditional agents: action space A defined by the environment



- **External feedback o_t**
- Agent context $c_t = (o_1, a_1, o_2, a_2, \cdots, o_t)$
- Agent action $a_t \sim \pi(a \mid c_t) \in A$

ReAct: action space $\hat{A} = A \cup \mathcal{L}$ augmented by reasoning



- $\hat{a}_t \in \mathcal{L}$ can be any language sequence
- Agent context $c_{t+1} = (c_t, \hat{a}_t, a_t, o_{t+1})$
- $\hat{a}_t \in \mathcal{L}$ only updates **internal context**