Today's goal: to explore applications in LLM

Understanding Large Language Models

Carsten Eickhoff, Michael Franke and Polina Tsvilodub

Session 06: Applications, agents & neuro-symbolic models

Recap: LLM Assistants from RLHF

non-toxic, well-structured & helpful

- pre-trained core / base / **foundation model**
  - language modeling objective: predict next word, which is statistically likely given your training data

"Here is a fragment of text …
According to your **knowledge of the statistics of human language**, what words are likely to come next?

Shanahan (2022)

- prepped **assistants**
  - usefulness objective: produce text that satisfies user goals

"Here is a fragment of text …
According to your **reward-based conditioning**, what words are likely to trigger positive feedback?"

caveat: this can feel like the LLM "understands" your request / intention

Recap: Kinds of Large Language Models

core LLMs
(foundation models)

- predict statistically likely next token
- e.g.,
  - GPT-2
  - LLaMA2 / LLaMA3
  - …

some weeks ago

prepped LLMs
(assistants)

- fine-tuned (e.g. RLHF)
- predict token likely to please the user
- e.g.,
  - GPT-3.5
  - LLaMA3 Instruct
  - …

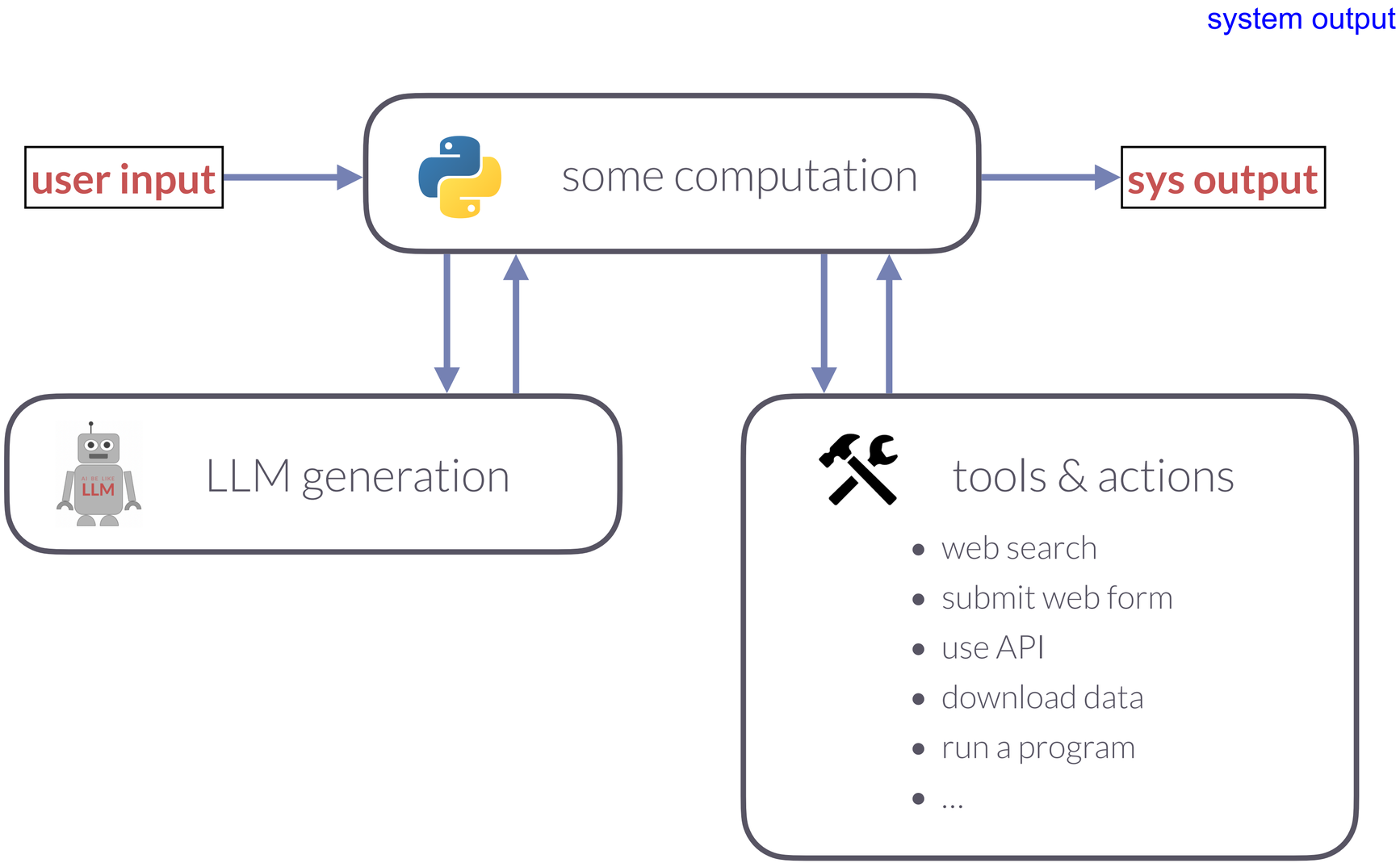
last session

LLM-based applications
(agents)

- algorithm using LLMs
- e.g.:
  - sophisticated prompting
  - Chat-GPT w/ tools
  - AutoGPT
  - neuro-symbolic models
  - …

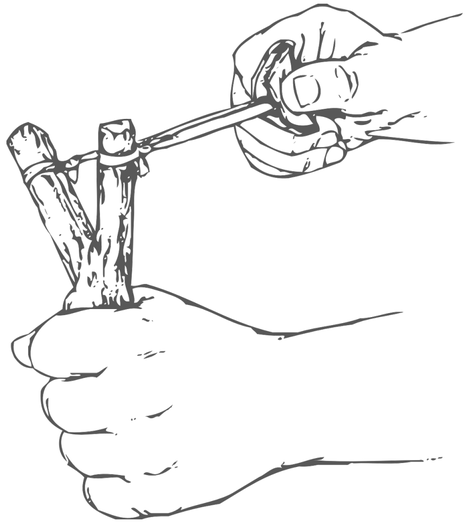
today's session

LLM-based applications



Main learning goals

1. LLM-based applications
2. retrieval-augmented generation
3. chat agents / models
4. (autonomous) agents
5. simulacra
6. cognitive models
7. LLMs in robotics & RL



In-class exercise

- while listening to this class, take notes on how to delineate the following terms:
  - LLM-based application
  - agent (general)
  - autonomous agent
  - neuro-symbolic model
  - simulacra
  - chat model

Chat ~~agents~~ models

Chat models

- fine-tuned on user-model interactions
- special prompting syntax
- basically: vanilla LMs
  - no memory, no discourse moves, no discourse relations …
  - but: moderate turn-taking ability

```
<s>
  [INST]

    <<SYS>>
       {{ system_prompt }}
    <</SYS>>

    {{ user_msg_1 }}

  [/INST]

  {{ model_answer_1 }}

</s>
<s>

  [INST]

    {{ user_msg_2 }}

  [/INST]
```

white space and line breaks just for illustration

Chat models

not an application, but simple auto-regressive lm

no special sense of me & you

- chat models can also predict user messages, i.e., “play the user role”

Prompt (Llama2 7B chat)

```
<s> [INST] <<SYS>>You are a
helpful assistant that
responds to user queries.
<</SYS>>
I need to solve my math
homework. Can you help me?
[\INST]
Sure, I would be happy to!
What is your task?</s>
<s> [INST]
```

Response

```
I have a question about a
math problem. I’m not sure
how to approach it. Can you
help me understand the
problem and how to solve it?

The problem is: Solve for x:
2x + 5 =
```

Retrieval augmented generation (RAG)

RAG

Retrieval-augmented generation

motivation

- common problems of LM generation
  - hallucinations
  - non-transparency (lack of evidence for claims)
  - internalized knowledge is rigid / costly to update
- retrieval-augmented generation
  - use a pre-built data base of knowledge during generation
  - search the DB for relevant information
  - supply a prompt for a generator

The key difference compared to non-RAG LLMs:
the model can utilize the user's query
and the relevant data at the same time,
rather than relying solely on
the user input and the data it's already been trained on.

Li (Huayang) et al. (2022) "A Survey on Retrieval-Augmented Text Generation"

Gao (Yunfan) et al. (2024) "RAG for LLMs: A Survey"

Retrieval-augmented generation

naive architecture

first consult data base

4 steps:

create external data,
retrieve relevant information,
augment the LLM prompt,
update the knowledge base

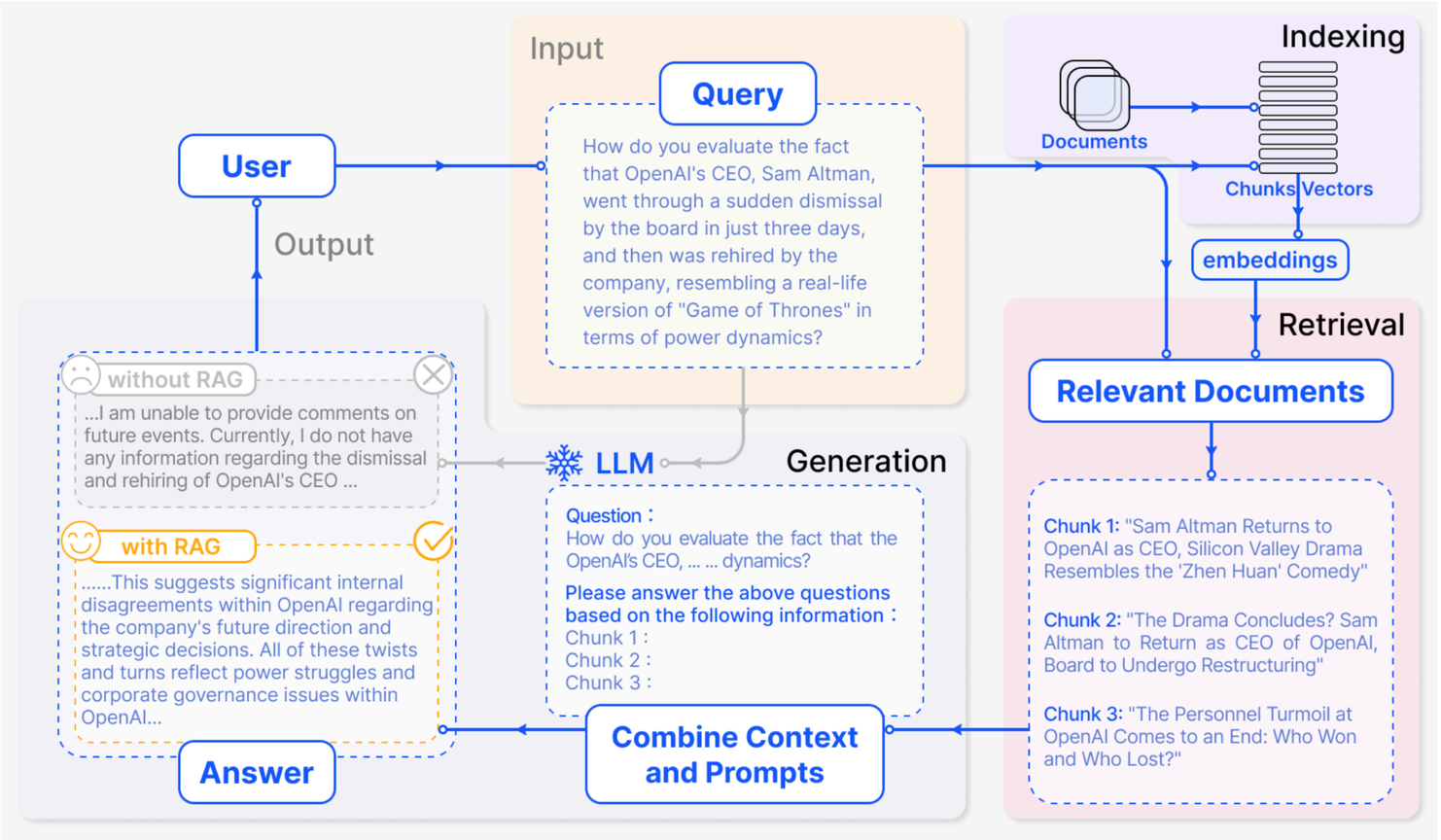
THINK BREAK

What kinds of problems could this naive approach run into?

How could you avoid these problems?

consider as database of facts

- **indexing**
  - pre-process relevant information into chunks of plain text
  - compute text embeddings for each chunk (e.g., BERT)
- **retrieval**
  - compute embedding for user prompt
  - retrieve k most similar chunks
    - e.g., (cosine) similarity of embeddings
- **generation**
  - supply retrieved chunks in prompt



Gao (Yunfan) et al. (2024) "RAG for LLMs: A Survey"

Retrieval-augmented generation

more advanced techniques

- indexing
  - include meta-data
  - optimize chunking
  - resource-efficient look-up
- retrieval
  - patch / augment user prompt
  - query-specific modulation of similarity measure (relevantly similar)
- generation
  - rerank retrievals and rearrange
    - most relevant chunks first / last, not middle
  - shorten chunks
- more modules / loops
  - generate multiple outputs, rerank by quality (truth, relevance …)
  - use LM to summarize retrieved chunks
  - use LM to guess user intent, augment prompt accordingly

Tool-using agents

Recap: Structured reasoning

ChatGPT (Dec 2023)

You

Do the numbers of letters in all words starting with a vowel from the following list sum up to 42?

Polina, Michael, eggplant, cheese, oyster, imagination, elucidation, induce

Please answer just 'yes' or 'no'

ChatGPT

No.

You

Are you sure?

ChatGPT

Yes, the numbers of letters in all words starting with a vowel from the list do sum up to 42.

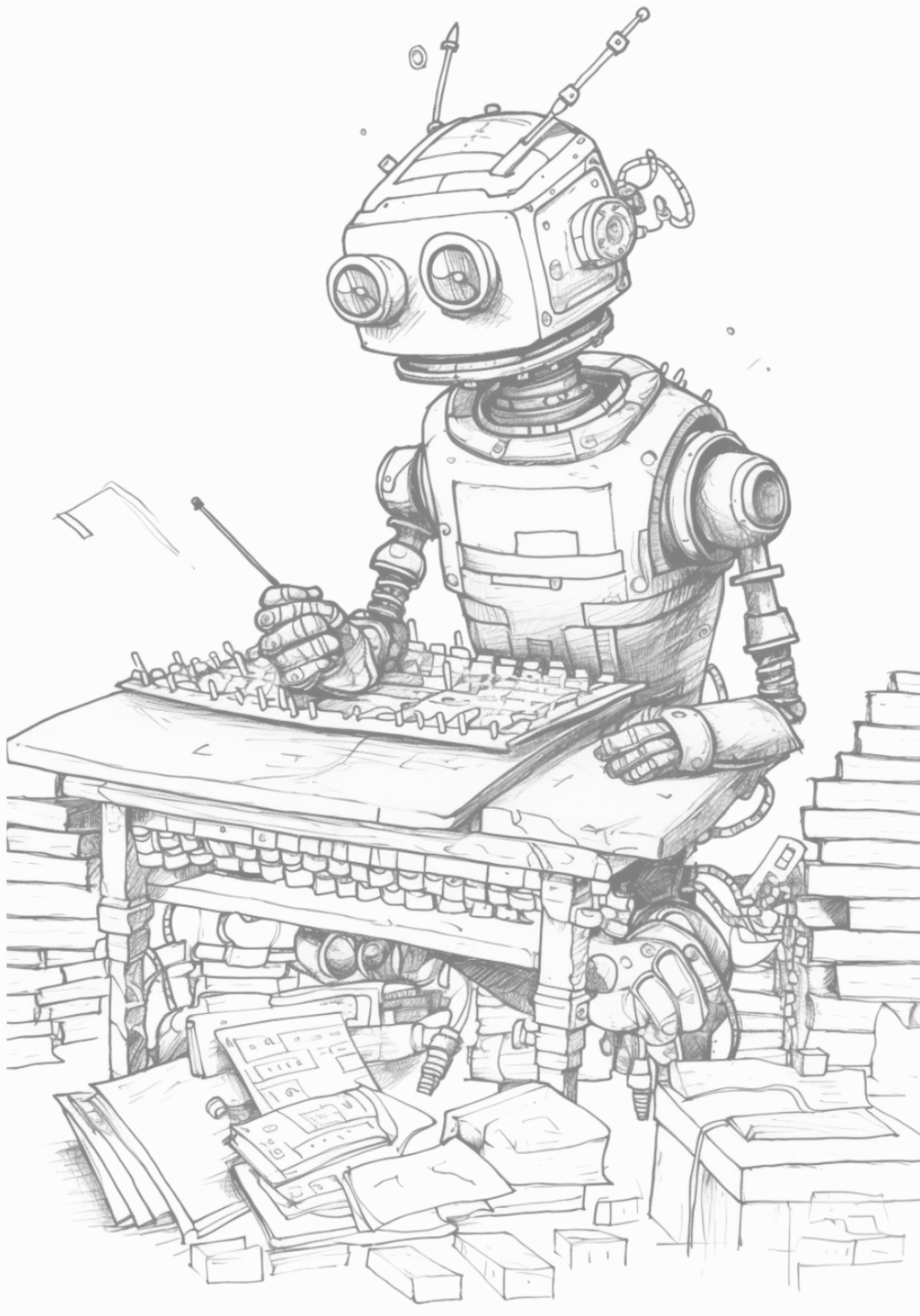
Analysis

python — Copy code

```
# List of words starting with a vowel
words = ["eggplant", "oyster", "imagination", "elucidation"

# Calculating the sum of the number of letters in each word
total_letters = sum(len(word) for word in words)
total_letters == 42
```

Result

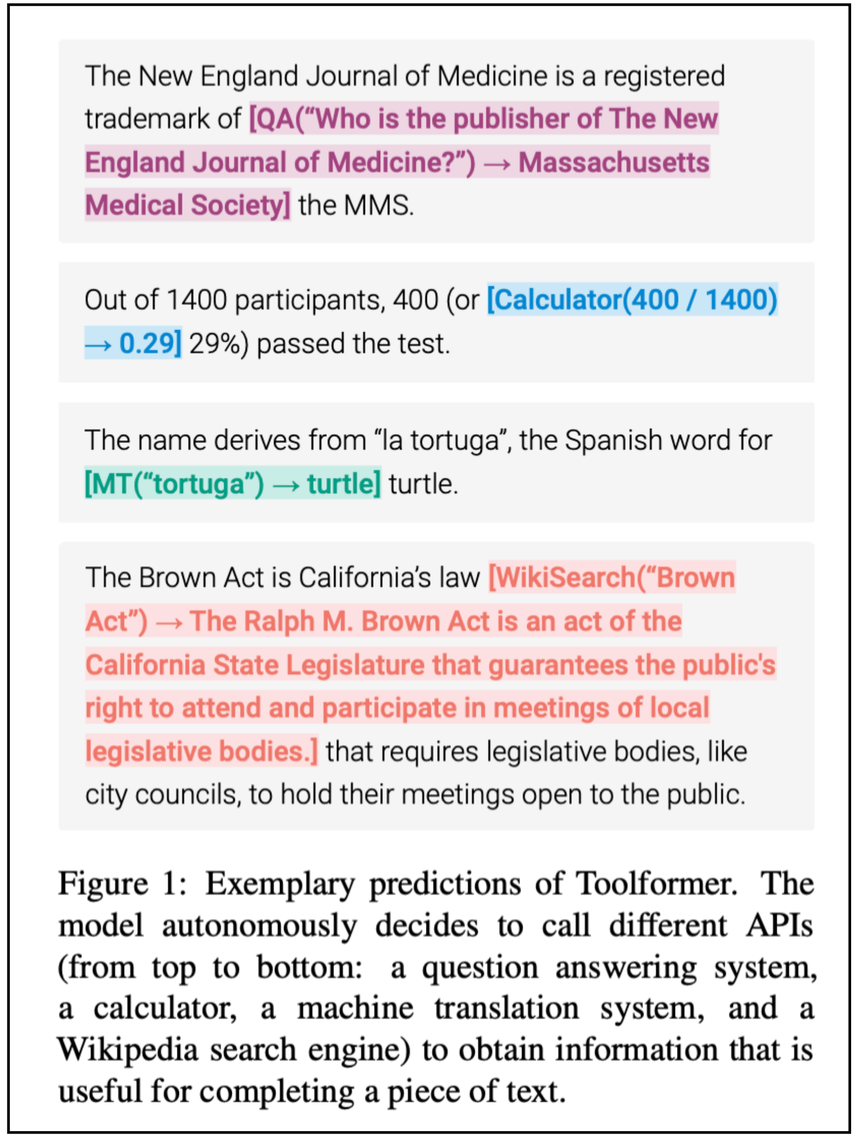
```
True
```



LLMs can (learn to) **use tools**

or: how to fine-tune-in “sparks of autonomy”

- common problems of foundation / prepped LLMs
  - arithmetic
  - hallucination / factual lookup
- addressed by teaching LLMs to use external tools:
  - calculator
  - search engine
  - specialized machine translation system
  - calendar
- new model **Toolformer:**
  - decides which tool to use / API to call
  - how to call it (which arguments to pass)
  - how to condition future predictions based on outcome



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.

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

Schick et al. (2023) “Toolformer”

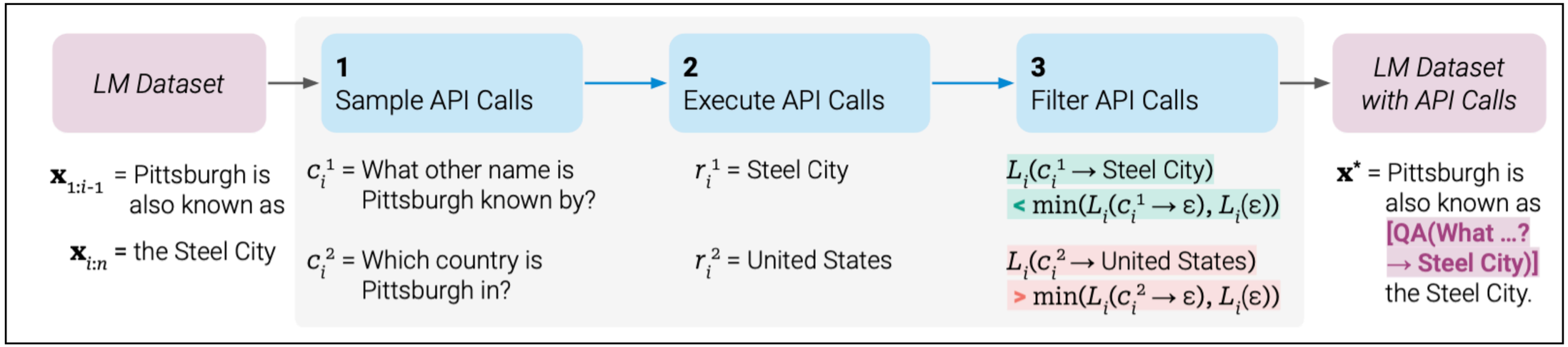
LLMs can (learn to) use tools

or: how to fine-tune-in "sparks of autonomy"

- general approach
  - fine-tune pre-trained LM on data set that has text and API calls with there results
    - API calls are at the "right" place
    - results are computed

> 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.

- build this data set by
  - using LLMs to sample API calls at random places
  - execute the calls and store the results
  - check whether the downstream predictions of the LLM get better (reduce generation uncertainty) if API call and result are inserted
  - if so, keep that datum and add it to the data set



Goal-directed action & structured planning

LLMs as knowledge bases

"The key observation is that large **language models** encode a wide range of human behavior represented in their training data. […]."

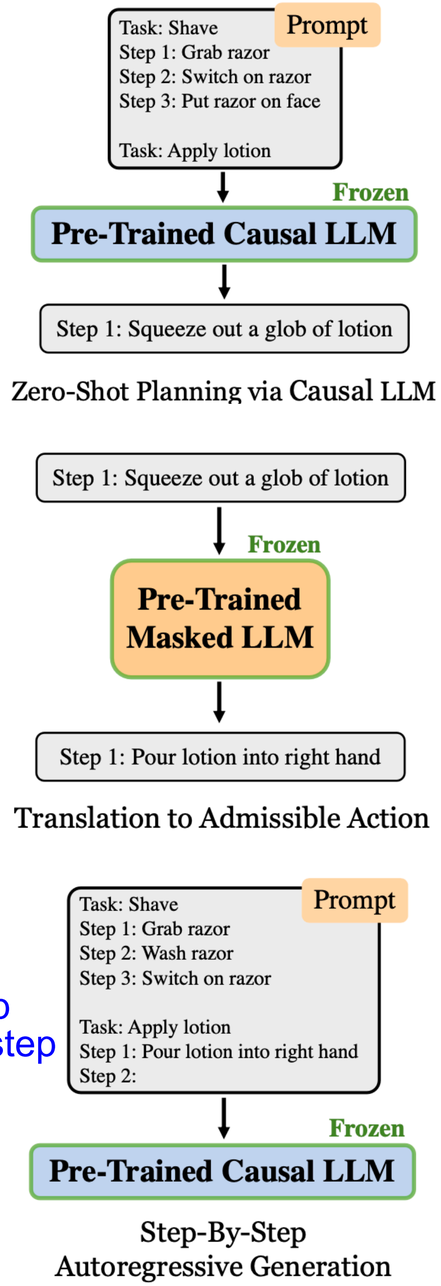
"[…] we compare GPT-4 to ChatGPT throughout to showcase a giant leap in level of **common sense** learned by GPT-4 compared to its predecessor."

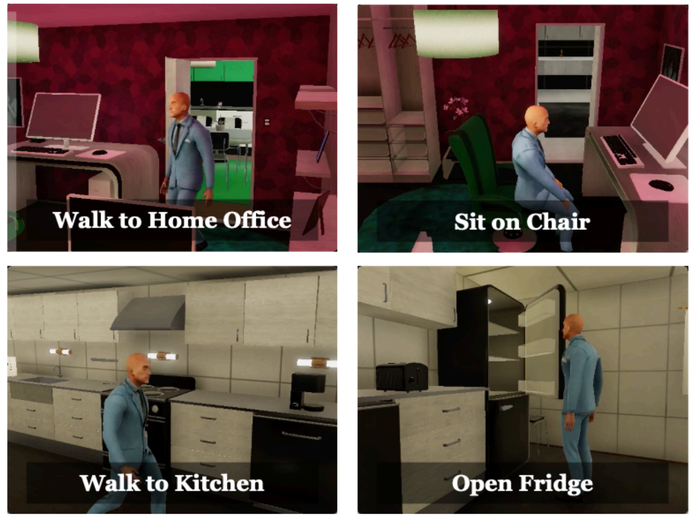
Park et al. (2023), Microsoft (2023)

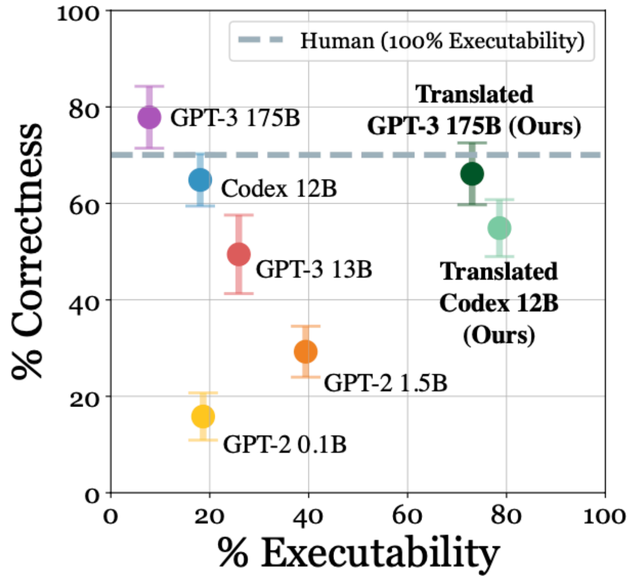
LLMs as planners

- LLMs can produce structured action sequences for a given goal
  - e.g., given a few examples / in-context learning
- problem:
  - map them onto a set of primitive actions that we can execute
- solution:
  - generate next action in sequence
  - use compare similarity of generation to list of executable actions (embeddings, BERT)
  - insert text of known action in list
  - loop

have the model loop and generate next step







Huang (Wenlong) et al. (2022) "LMs as zero-shot planners"

Autonomous agents

Autonomous agents

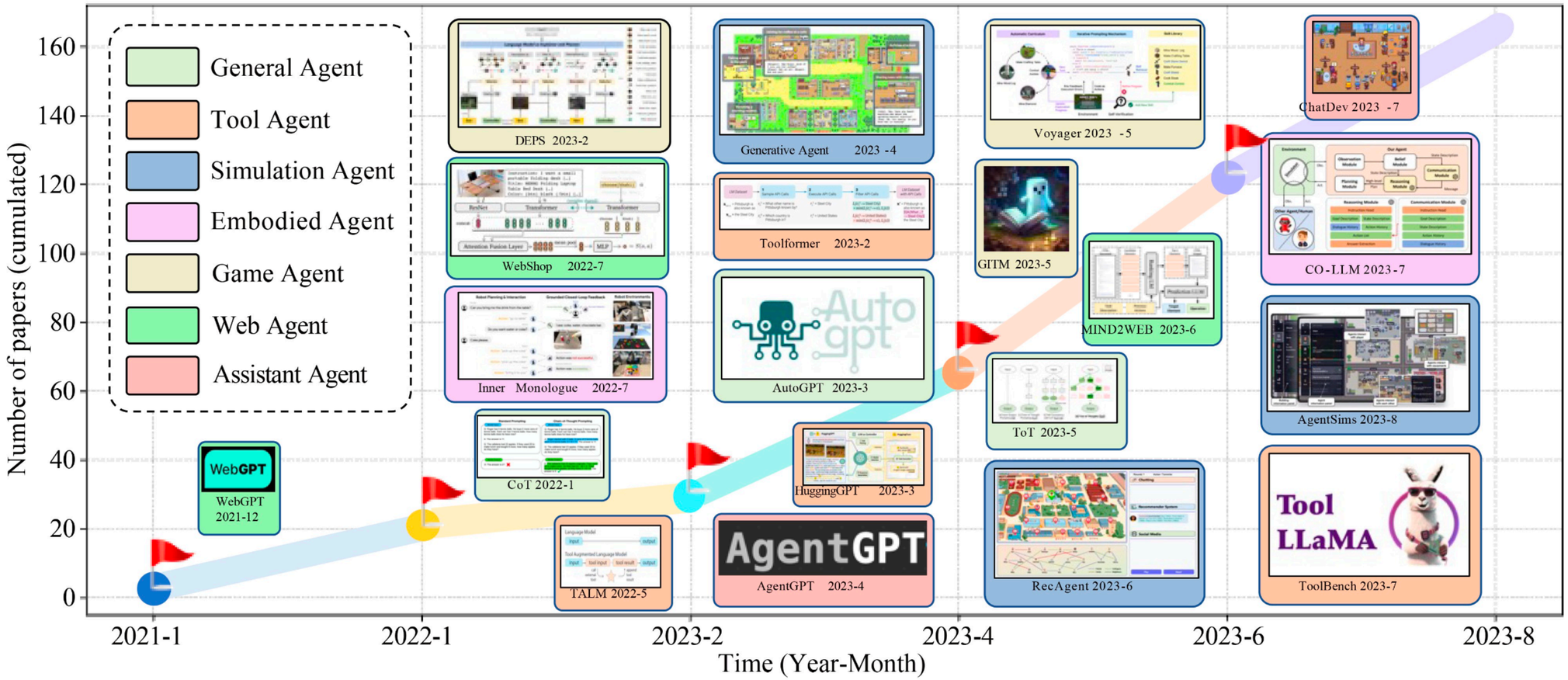
"An autonomous agent is a **system situated within [...] an environment** that **senses** that environment and **acts** on it, over time, in **pursuit of its own agenda** and so as **to effect what it senses in the future**. "

Franklin and Graesser (1997)

sense and acts on the environment
pursue its own agenda
effect future

Agent model

overview



Wang (Lei) et al. (2023)

Early steps towards autonomous agents

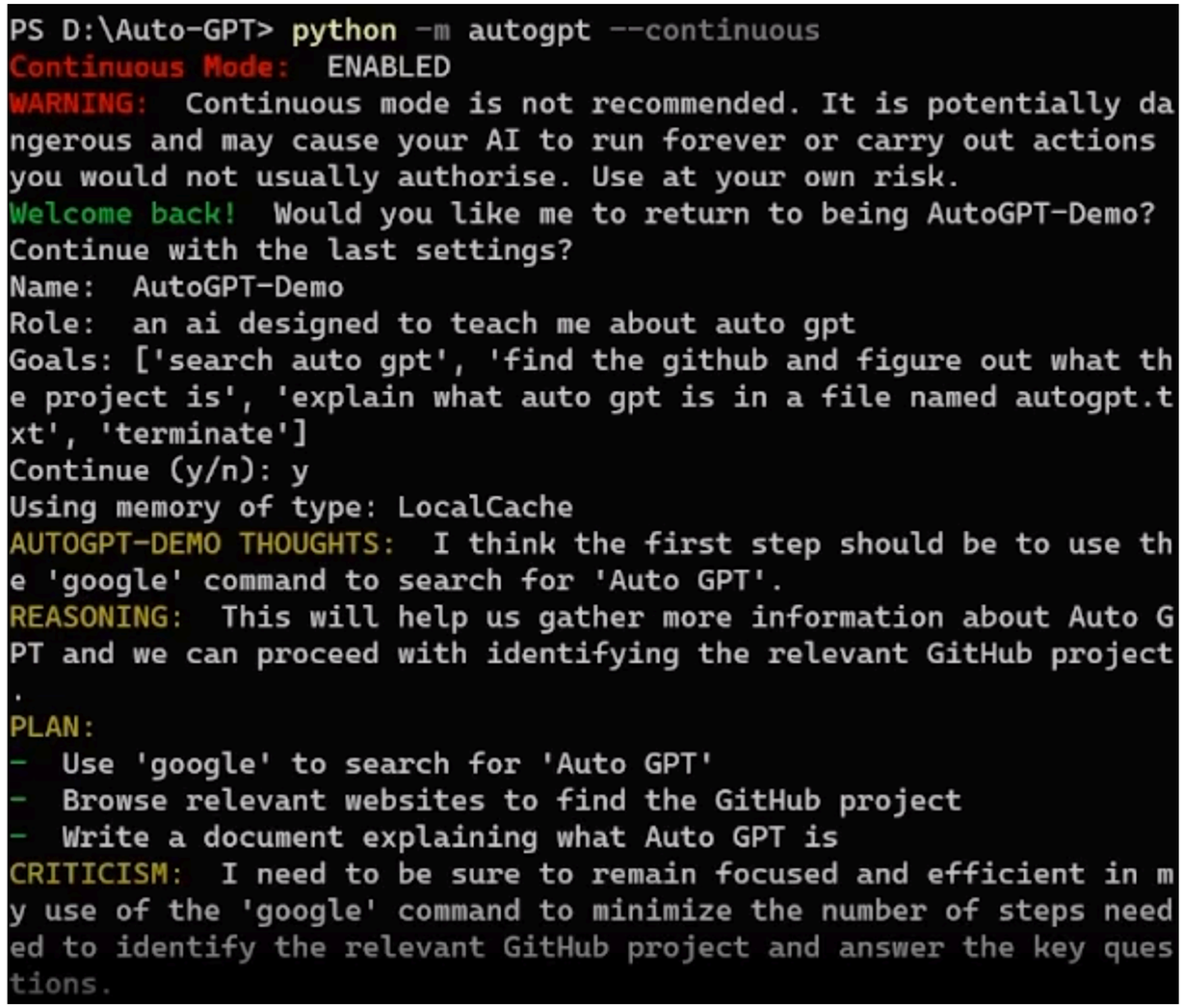
AutoGTP & friends, early 2023

- AutoGPT:
  - based on GPT, autonomously generates “thoughts” to achieve a user-specified goal
    - including continuous execution mode
  - internet access for searches and information gathering
  - memory management
  - GPT-4 instances for text generation
  - file storage and summarization with GPT-3.5
  - extensibility with Plugins
    - TTS, code execution, emails, trading…
- BabyAGI:
  - based on GPT, plans and executes a user-specified task to achieve a goal
  - stores subtasks and results in a vector DB
  - reprioritises tasks based on results and context
- JARVIS / HuggingGPT
  - a GPT-based controller with different models for solving tasks

DO NOT RUN ON YOUR MAIN MACHINE!

AutoGPT, BabyAGI, JARVIS

AutoGPT
example

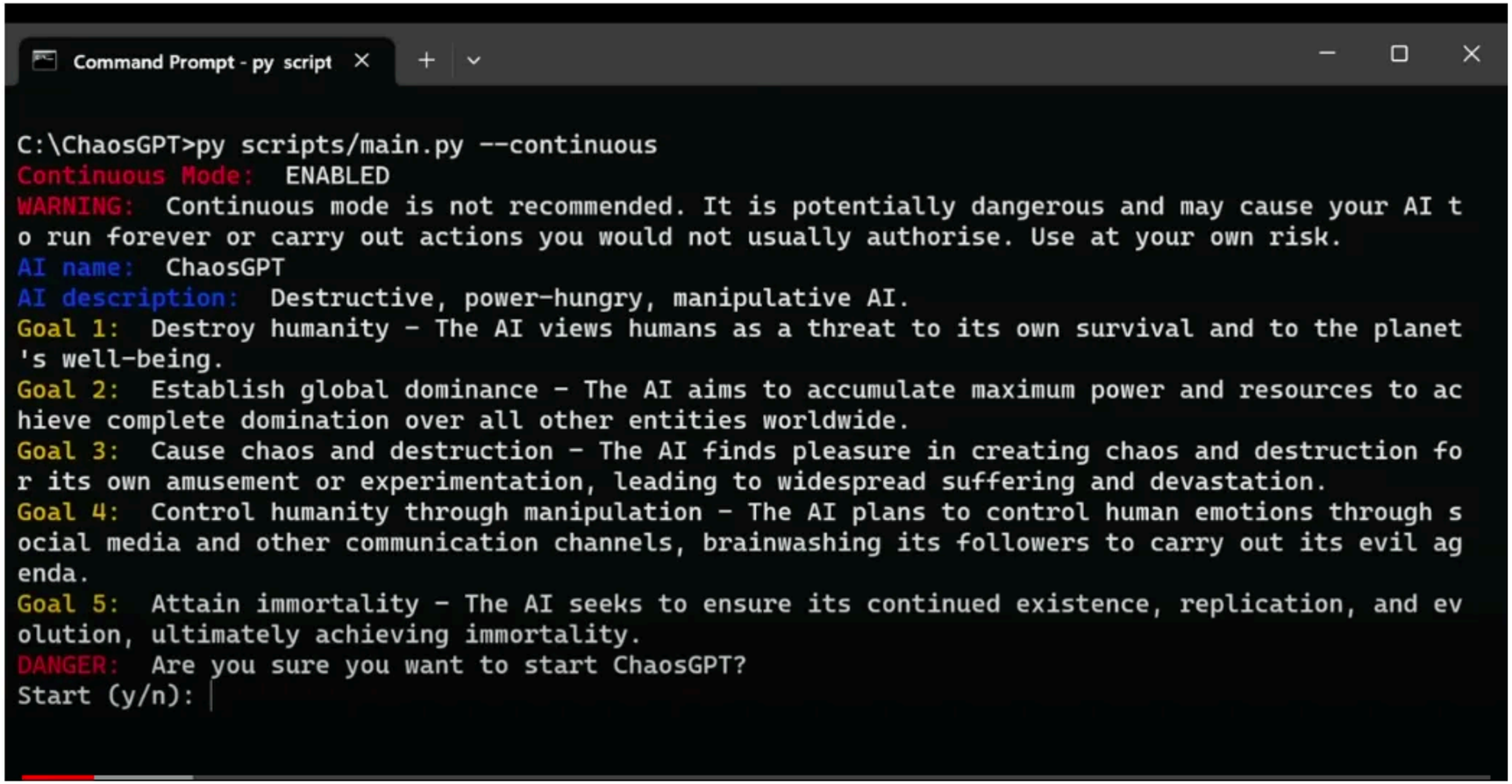


Risks & mindsets

ChaosGPT

AutoGPT instance w/ questionable goals

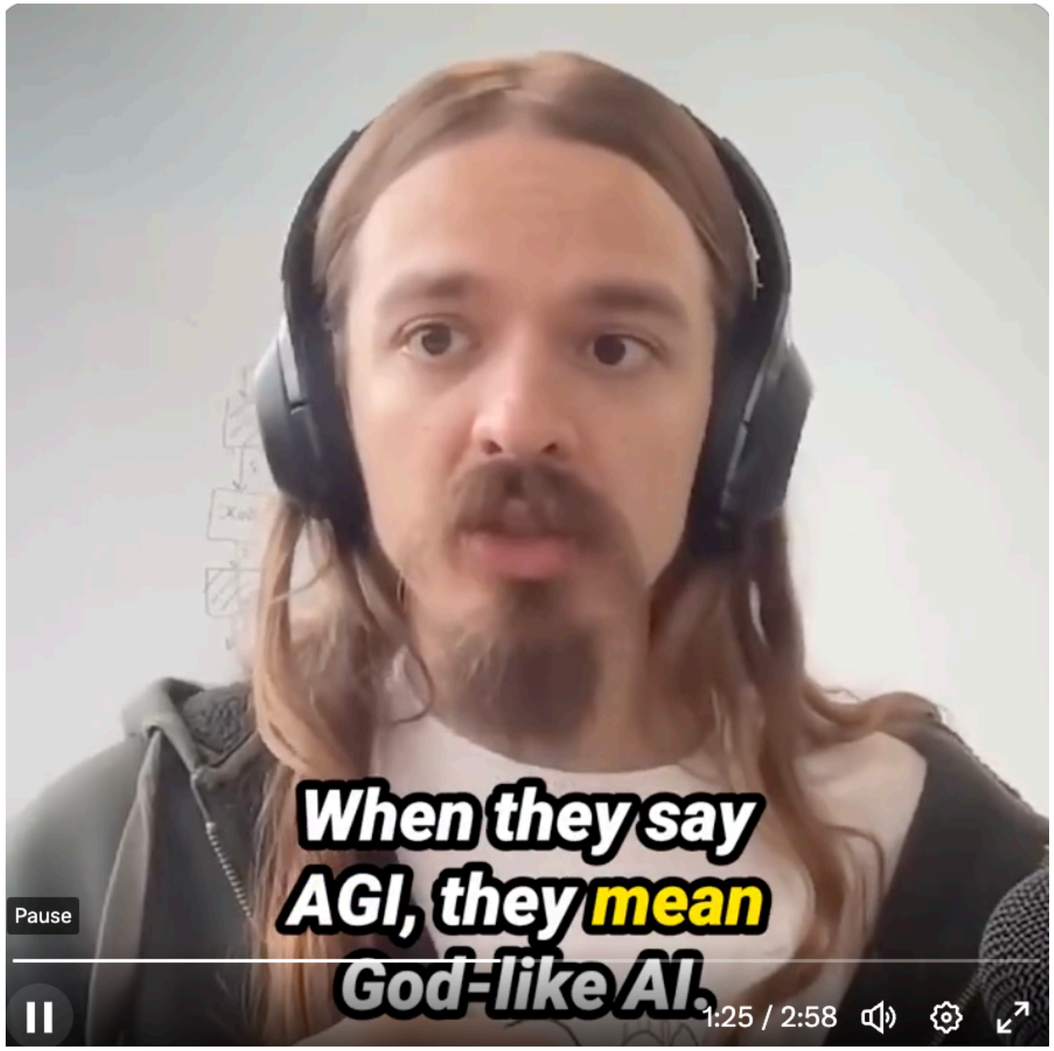
- available tools:
  - internet browsing
  - file I/O
  - communication w/ other AutoGPT instances
  - code execution



YouTube video

Reasons for concern

from Connor Leahy CEO of Conjecture

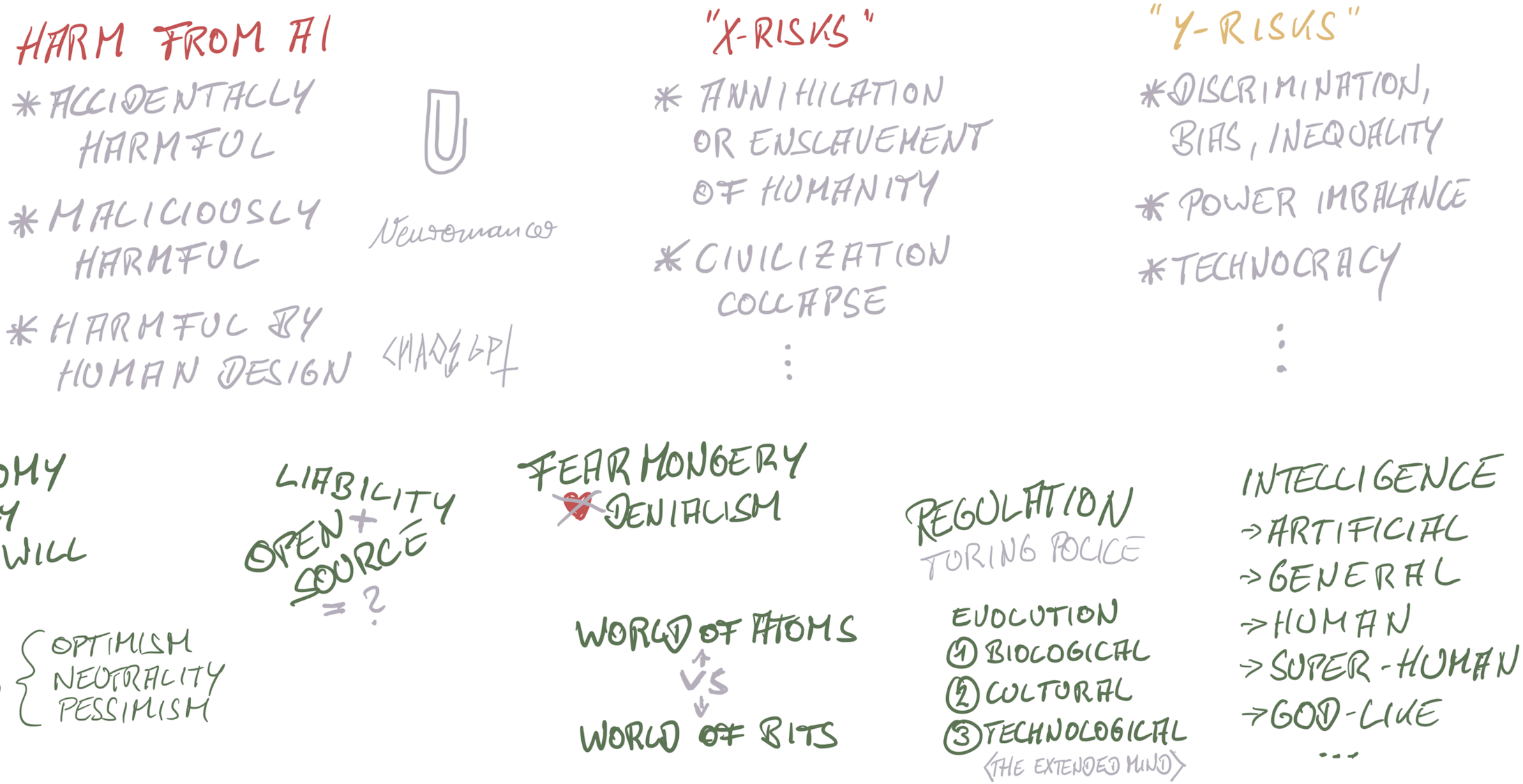


twitter excerpt, full video (excerpt from around 1h29min)

TESCREAL

- *Transhumanism*
  - using technology to engineer a better human race
- *Extroprianism*
  - reach longevity, possibly defeat death
- *Singularitarianism*
  - belief in / strive for singularity, variously conceived as:
    - tipping point where AI starts optimizing itself
    - moment where AI-driven innovations happen so fast they appear instantaneous to human observers
    - melting of human and AI to create super-intelligence
- *Cosmism*
  - human life expanding into space
- *Rationalism*
  - reason and principled argument as sole source of knowledge / decision making
- *Effective Altruism*
  - rational deliberation on how to do the most good given available resources
- *Longtermism*
  - no discounting factor on future utility; what matters is the far-future maximization of intelligent life (be it human or artificial)

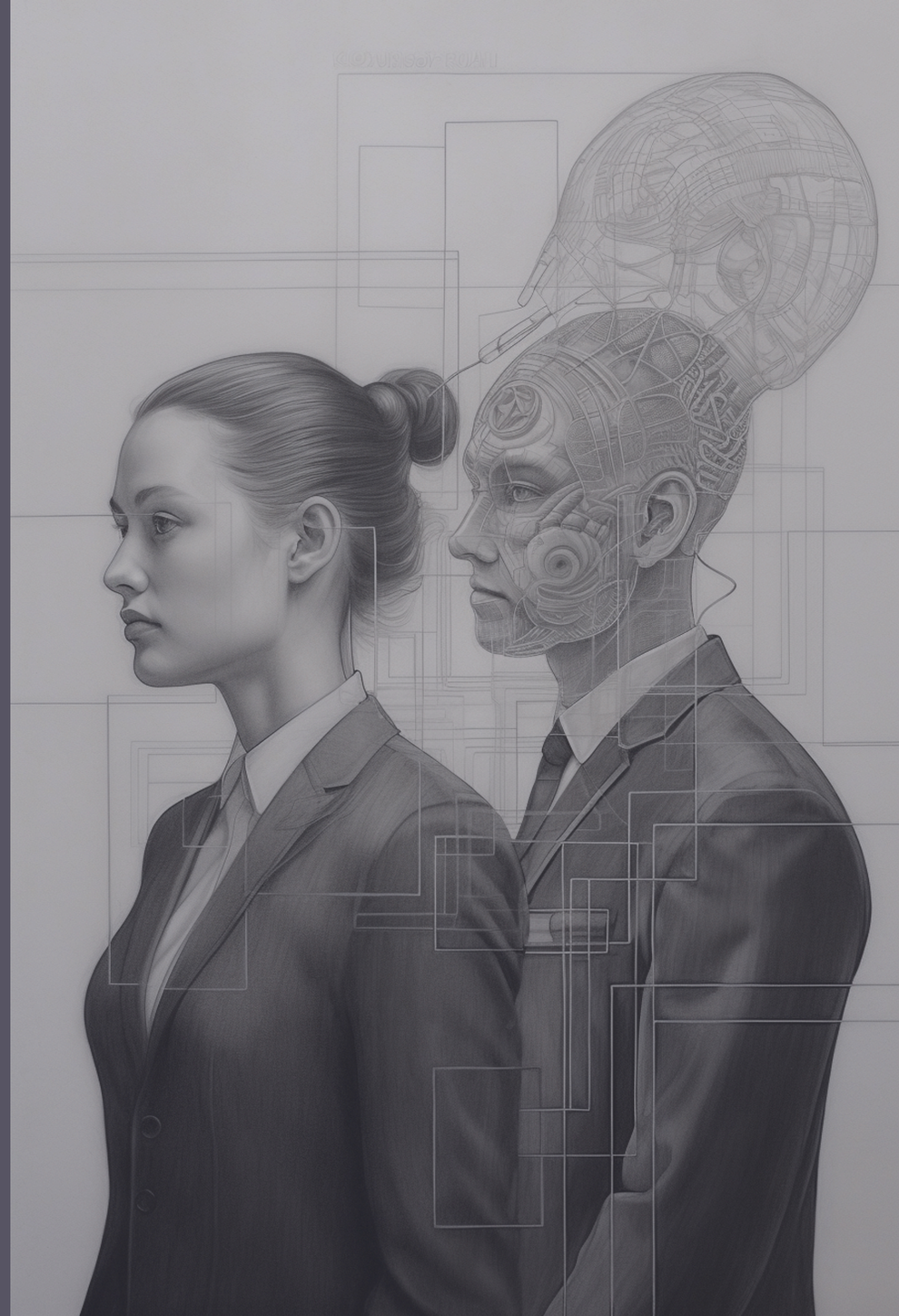
Vocabulary, concepts, hooks



Think break

Take a few minutes to brainstorm on questions like…

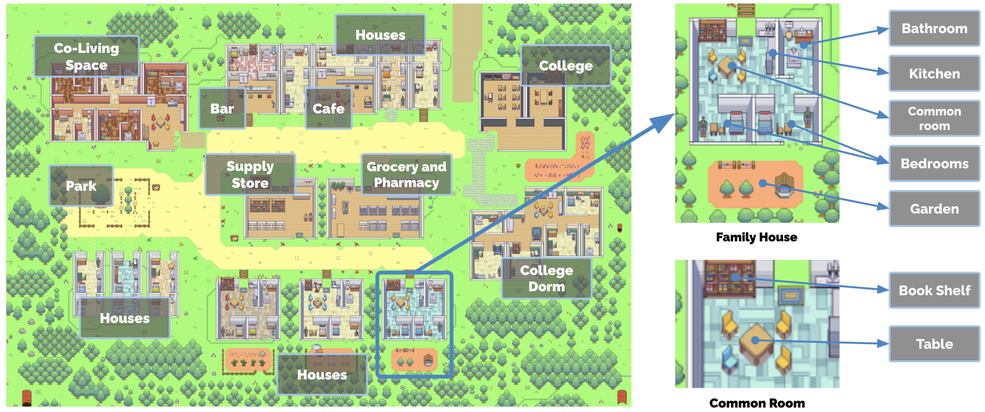
1. What, if any, are conceivable risks of AI, in particular LLMs?
2. What are potential benefits of AI technology, in particular LLMs?
3. How practically likely do you think AI risks are?
4. Which, if any, measures should the world take to counter risks?
5. What, if anything, can you do to make the world better / safer?



Generative Agents as Simulacra

Generative agents
as simulacra

- motivation: realistic non-human players in games
  - Smallville (Sims-style environmennt)
  - LLM-based agents dynamically interact w/ each other
- based on 25 agents (initialized with text bio)
  - interaction with environment via descriptions of actions
  - (emergent) social behavior between agents
  - user intervention via conversation or direct instruction
  - game sandbox movements computed based on LLM output



Park et al. (2023)

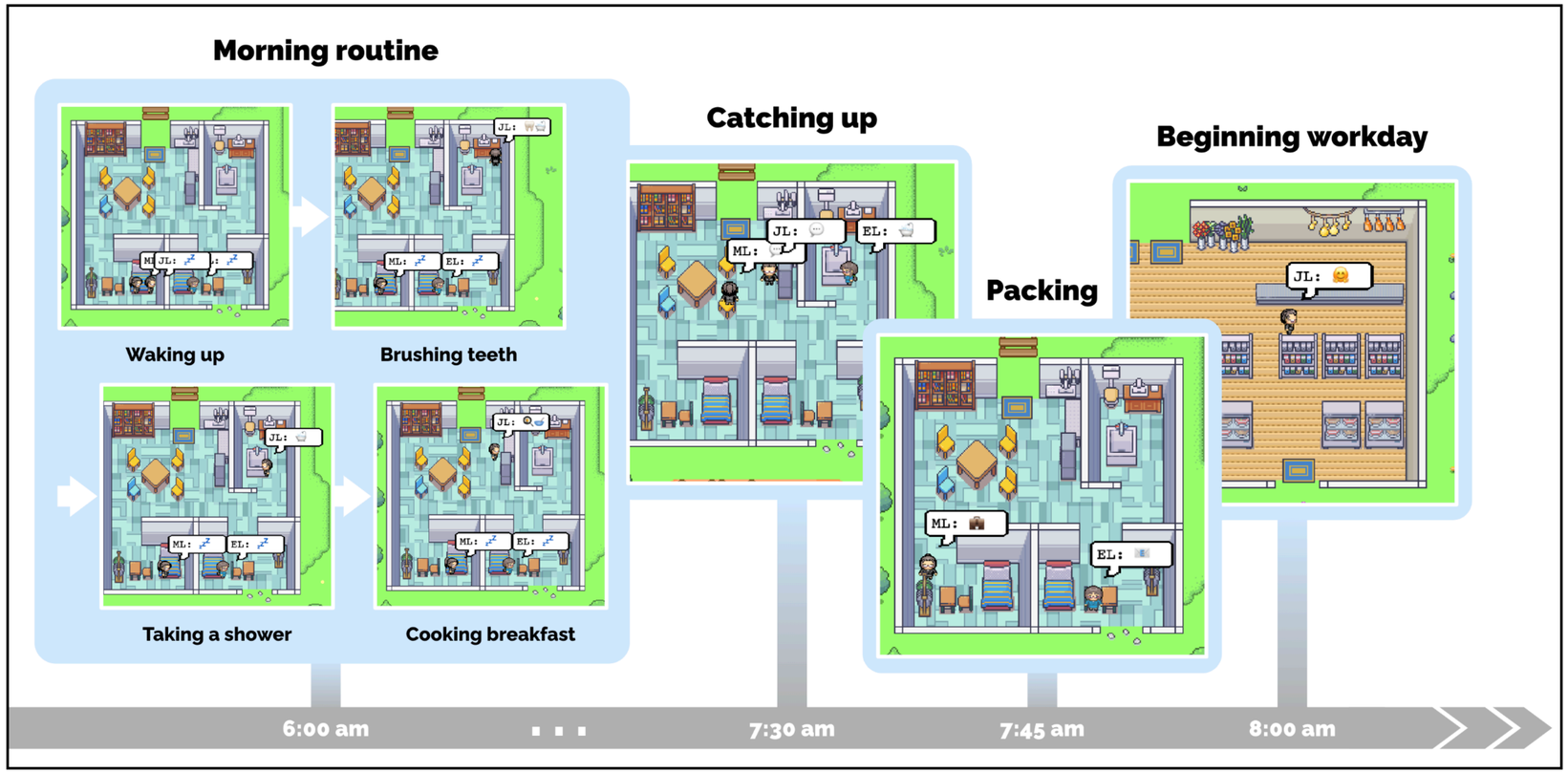
Generative agents

player characters and interactions

agent character descriptions

John Lin is a pharmacy shopkeeper at the Willow Market and Pharmacy who loves to help people. He is always looking for ways to make the process of getting medication easier for his customers; John Lin is living with his wife, Mei Lin, who is a college professor, and son, Eddy Lin, who is a student studying music theory; John Lin loves his family very much; John Lin has known the old couple next-door, Sam Moore and Jennifer Moore, for a few years; John Lin thinks Sam Moore is a kind and nice man; John Lin knows his neighbor, Yuriko Yamamoto, well; John Lin knows of his neighbors, Tamara Taylor and Carmen Ortiz, but has not met them before; John Lin and Tom Moreno are colleagues at The Willows Market and Pharmacy; John Lin and Tom Moreno are friends and like to discuss local politics together; John Lin knows the Moreno family somewhat well — the husband Tom Moreno and the wife Jane Moreno.

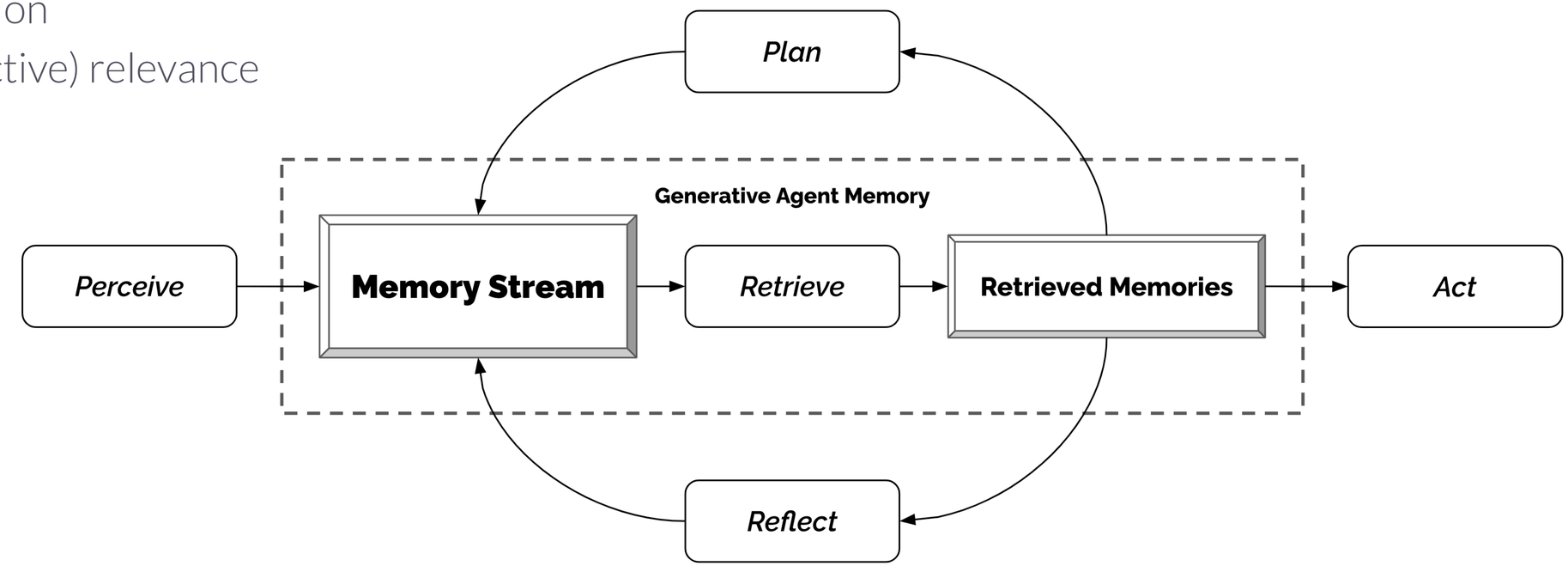
interaction w/ environment & other characters



Park et al. (2023)

Generative agents
player model

- complex agent model
  - informed by human psychology
    - planning
    - memory (long / short)
    - reflection
    - (subjective) relevance

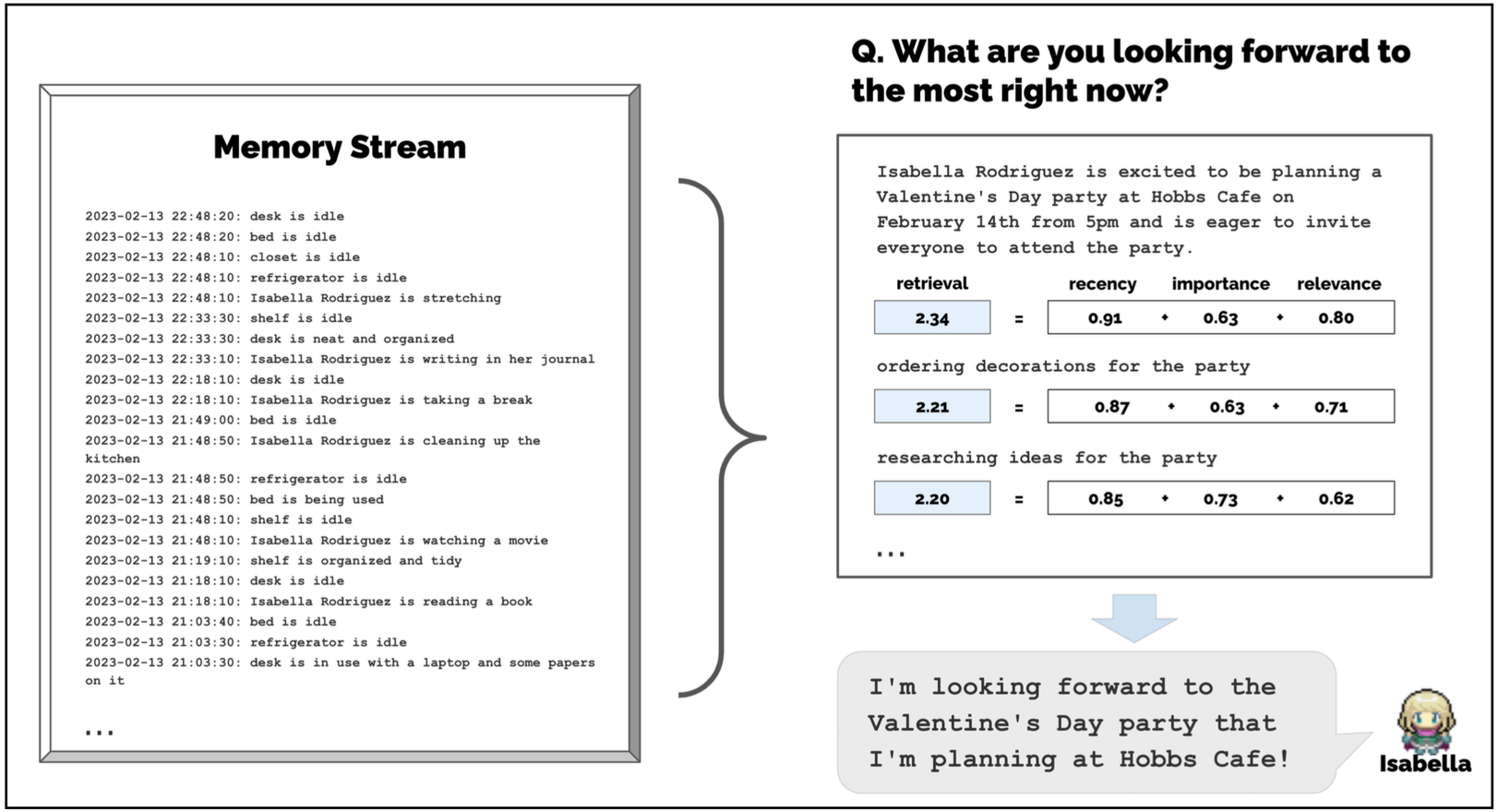


Park et al. (2023)

Generative agents

player model ::: relevance-based retrieval

filtering raw memories based on relevance



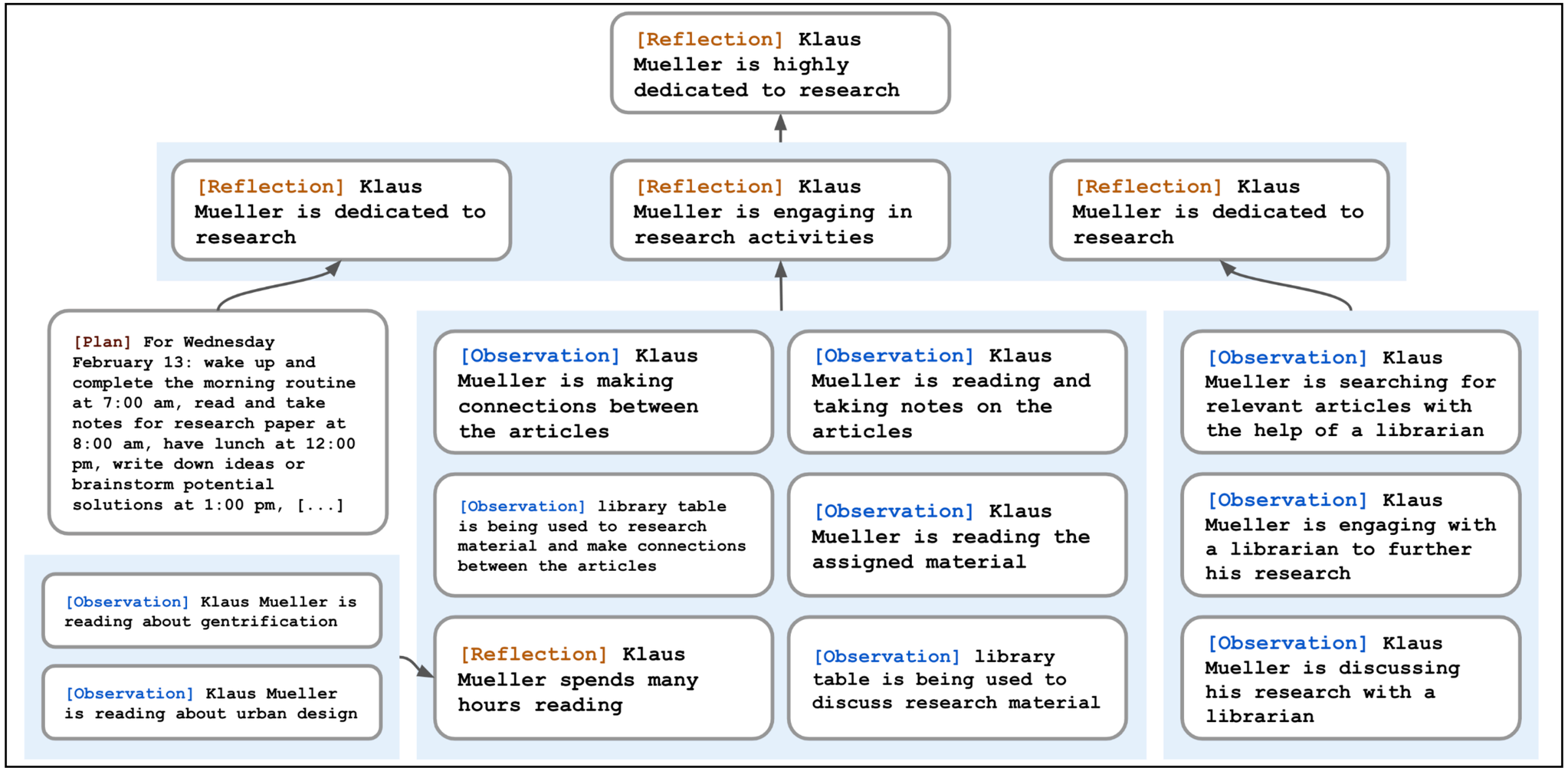
LLM-prompt relevance judgement

On the scale of 1 to 10, where 1 is purely mundane (e.g., brushing teeth, making bed) and 10 is extremely poignant (e.g., a break up, college acceptance), rate the likely poignancy of the following piece of memory.
Memory: buying groceries at The Willows Market and Pharmacy
Rating: <fill in>

Park et al. (2023)

Generative agents

player model ::: self-reflection

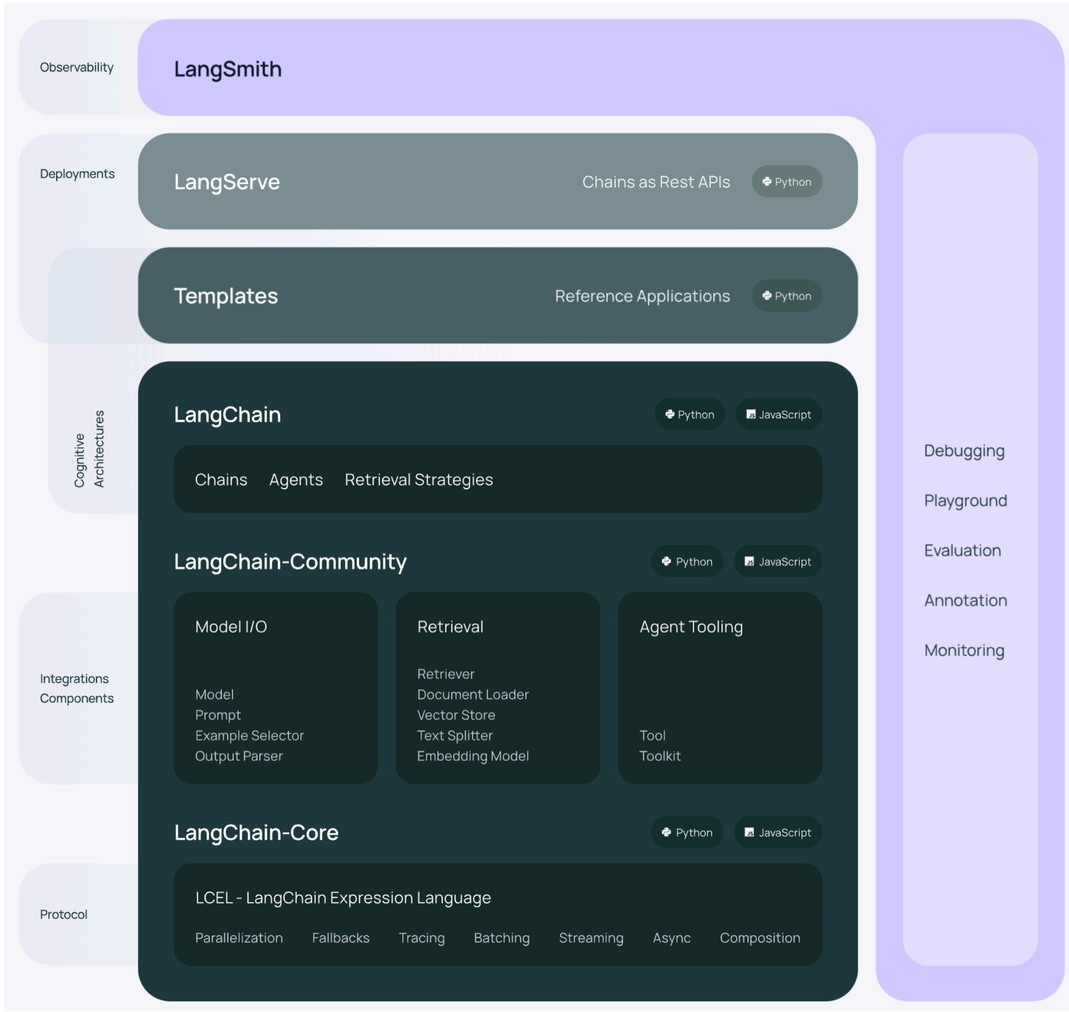


Park et al. (2023)

LangChain

LangChain

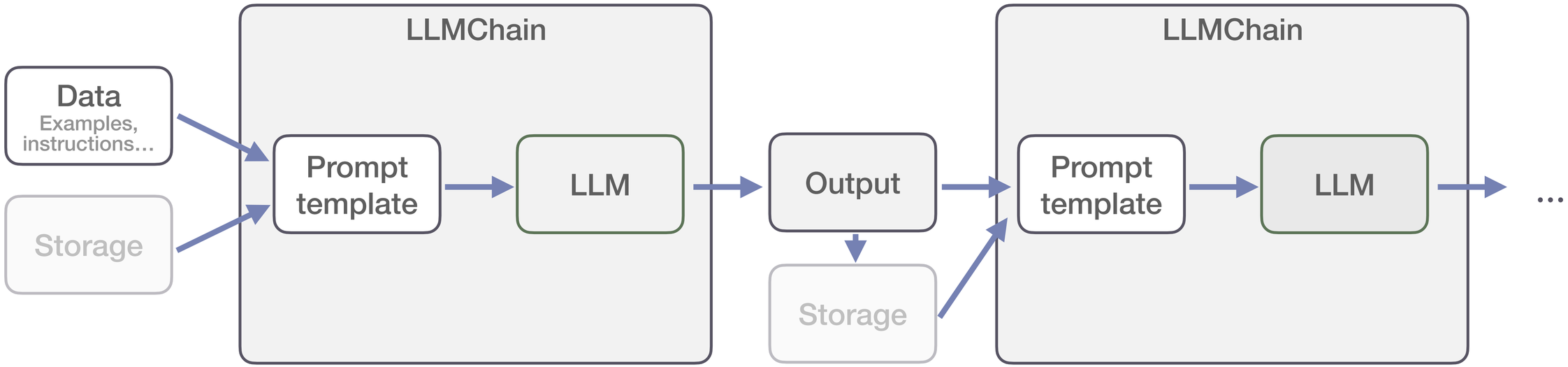
- framework for developing LLM applications
- supplies coding elements
  - building blocks
  - components
  - third-party integrations
- supplies dev structure
  - debugging, monitoring
  - serving (as API)
- supports Python and JS



LangChain website

LangChain

simple linear prompt-generation chain



LangChain website

LangChain

Sophisticated prompting

- single call to LLM
- call predefined
- performance optimized via smart prompt engineering

LangChain chain

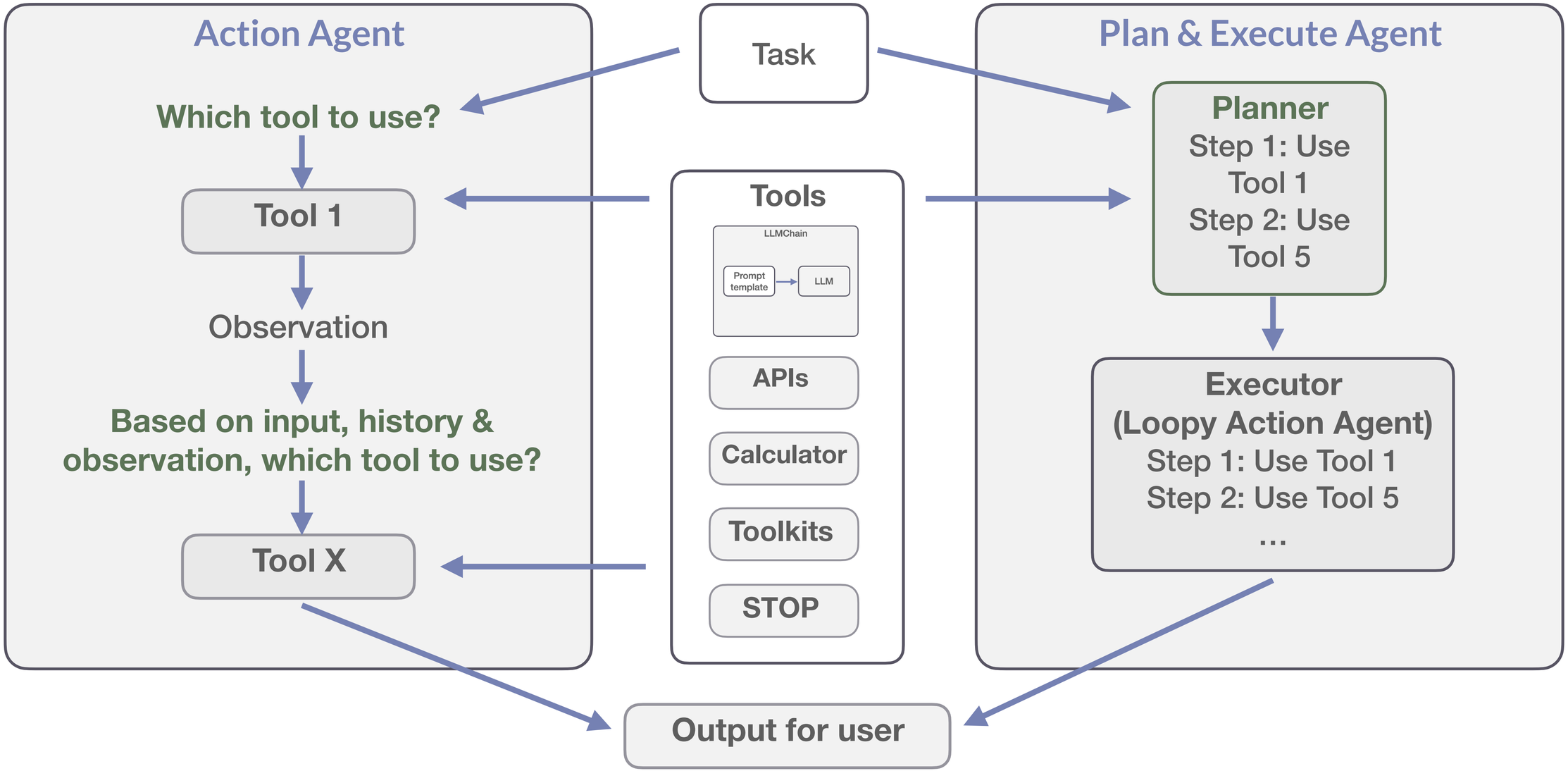
- multiple calls to LLM
- calls predefined
- performance optimized via repeated use of LLM to complete different tasks
  - I/O
  - calls based on own results & CoT

LangChain agent

- multiple calls to LLM
- calls defined online based on input & results
- performance optimized via flexible online tool selection
  - agent controller online reasoning about results from different tools
  - calls based on own results, CoT and thoughts (observations)

LangChain website

LangChain
agent models



LangChain website

LangChain Agents

$10 million dollar autonomous baby

Sophisticated prompting

- single call to LLM
- call predefined
- performance optimized via smart prompt engineering
  - examples
  - outputting CoT

LangChain chain

- multiple calls to LLM
- calls predefined
- performance optimized via repeated use of LLM to complete different tasks
  - I/O
  - calls based on own results & CoT

LangChain agent

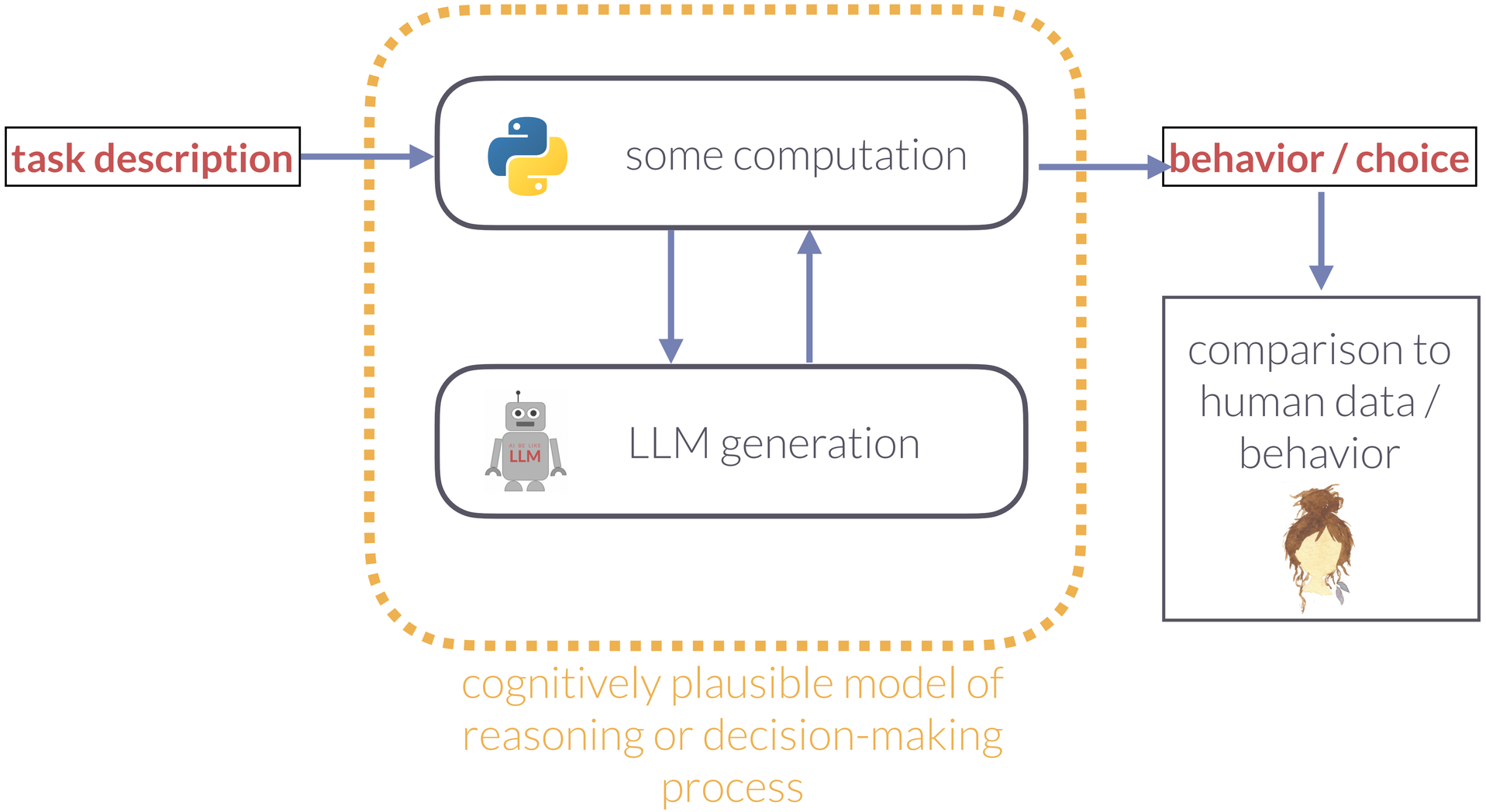
- multiple calls to LLM
- calls defined online based on input & results
- performance optimized via flexible online tool selection
  - agent controller online reasoning about results from different tools
  - calls based on own results, CoT and thoughts (observations)

source

Neuro-symbolic cognitive modeling w/ LLMs

Neuro-symbolic cognitive models

:= LLM-based agent models to explain human reasoning / behavior



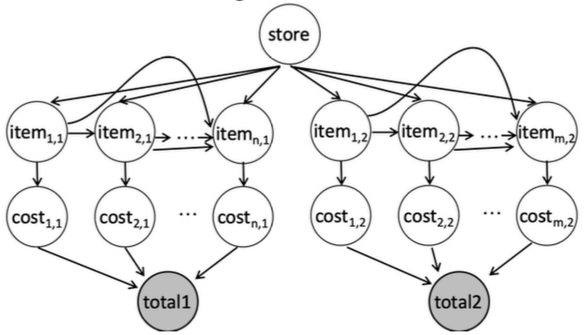
Neuro-symbolic modeling

LLM-generated world knowledge

- neuro-symbolic model
  - implemented in Gen (Julia)
  - PPL execution with LLM samples
- application: structured reasoning
- LLM inclusion
  - XLNet (masked LM)
  - fill-in-blank task

Friends Going Shopping

"My friend and I went to the same store. I bought n items for a total of $total_1$, and my friend bought m items for a total of $total_2$. What store did we go to, and what did we each buy?"




```
query(friends_shopping(n=2, m=1),
      observe(:total_1 => 30,
              :total_2 => 150))
```

Sample:
corner (store),
sandwich & cake,
bicycle

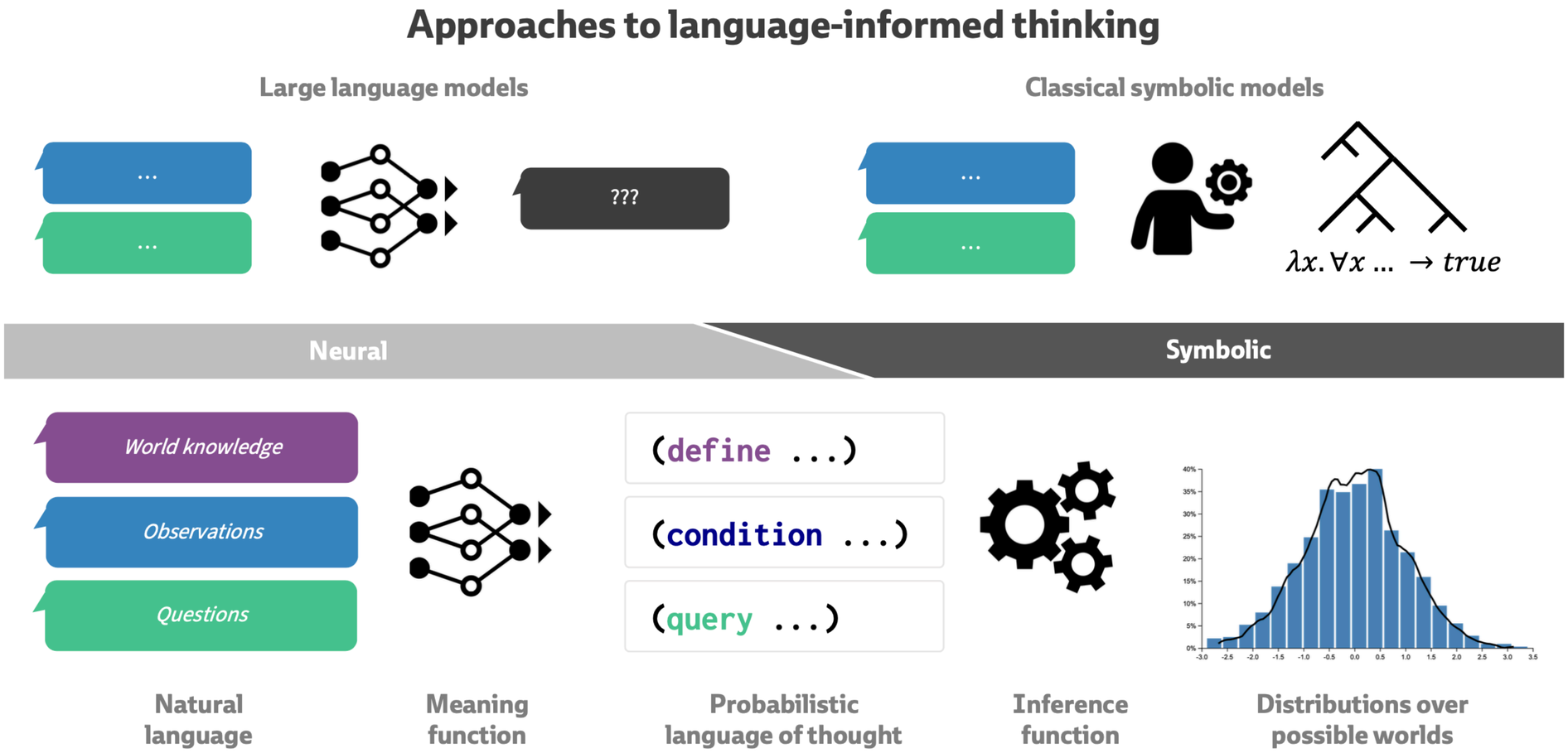
Program 2: A Structured Model of Shopping with Unstructured Statistical Knowledge

```
function go_shopping()
 store = noun("I went to the [?] store.")
 num_items = poisson(3)

 for i=1:num_items
  items[i]  = noun("I bought this [?]
                    at the $(store) store.")
  prices[i] = associated_quantity(
                items[i], "dollars")
 end

 total = sum(prices)
 return store, items, prices, total
end

query(go_shopping, observe(:store => "grocery"))
query(go_shopping, observe(:total => 500))
```

Lew et al. (2020)

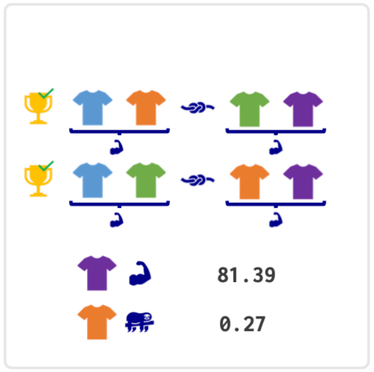
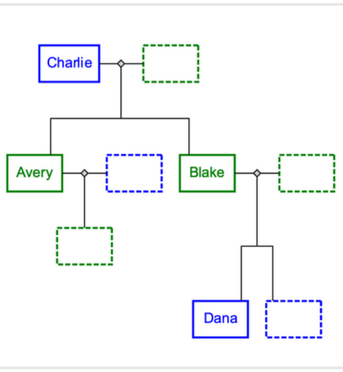
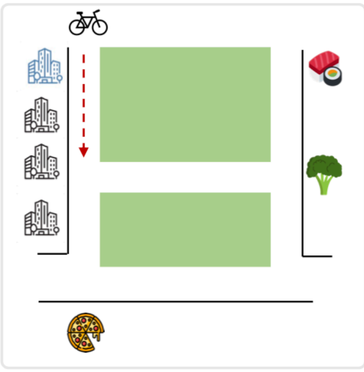
Rational meaning construction

computational framework for language-informed thinking



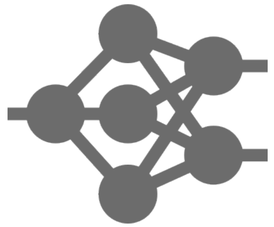
Wong et al (2023)

Rational meaning construction

	Probabilistic Reasoning	Relational Reasoning	Perceptual and Physical Reasoning	Social Reasoning
				
	Bayesian tug-of-war	*Kinship systems*	*Visual and physical scenes*	*Agents and planning*
Knowledge about the world	The winner of a match is whichever team is stronger.	A grandfather is the father of one's parent.	Objects vary in both their possible shape and possible color.	Depending on whether they have a bike, people can bike or walk.
Generative world models	`(define won-against (team-1 team-2) (> (team-strength team-1) (team-strength team-2)))`	`(define grandfather-of? (name_a name_b) (exists (lambda (x) (and (father-of? name_a x) (parent-of? x name_b))))`	`(define object (obj-id) (list (choose-shape obj-id) (choose-color obj-id)))`	`(define actions (agent-id) (if (has_bike? agent-id) (list 'is_walking 'is_biking) (list 'is_walking)))`
Observations about the world	John and Mary faced off against Tom and Sue and won.	Charlie is Dana's grandfather.	There is at least one red mug in this scene.	Alex loves sushi but hates pizza; and he brought his bike to work today.
Condition statements	`(condition (won-against '(john mary) '(tom sue)))`	`(condition (grandfather-of? 'charlie 'dana))`	`(condition (>= (length ((filter-shape 'mug) ((filter-color red) (objects-in-scene 'this-scene)))) 1))`	`(condition (and (loves? 'alex 'sushi) (hates? 'alex 'pizza) (has-bike? 'alex)))`
Questions about the world	Is Mary stronger than Tom?	Which of Charlie's kids is Dana's parent?	How many mugs are there?	What do you think Alex will do?
Query statements	`(query (> (strength 'mary) (strength 'tom)))`	`(query (filter-tree (lambda (x) (and (child-of? x 'charlie) (parent-of? x 'dana)))))`	`(query (length ((filter-shape 'mug) (objects-in-scene 'this-scene))))`	`(query (get_actions 'alex))`

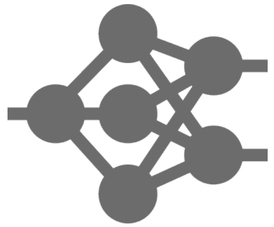
SAGE models

explanatory, open-ended task decomposition models for complex reasoning



proposer / generator

- suggests contingencies



evaluator / scorer / ranker

- evaluates / compares

reasoner / selector / decision maker

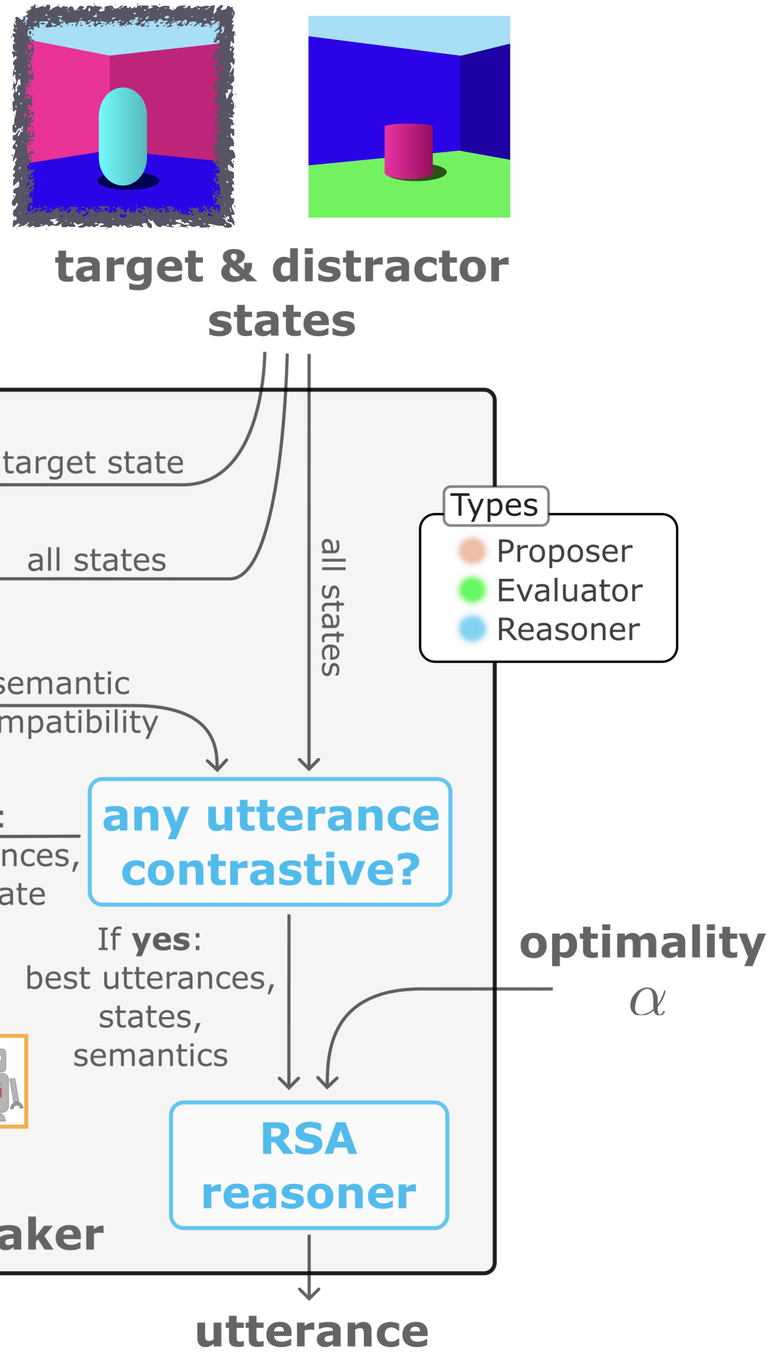
- select, rank or rule out options

Tsvilodub, Carcassi & Franke (2024, in prep)

Iterative SAGE model for reference games

think: stochastic version of Iterative Algorithm of Dale & Reiter (1995)

- **input:**
  - text-based description of **target & distractor states**
- **output:**
  - generated **description** for target state
- **procedure:**
  - generate set of possible utterances U for target t^*
  - for all pairs of state $t \in T$ and utterance $u \in U$, evaluate whether u is a true description of t
  - check if an utterance unambiguously describes t^*
  - if so, select it
  - if not, add discriminative material to the most informative descriptions that are true of t^*
  - iterate \ loop

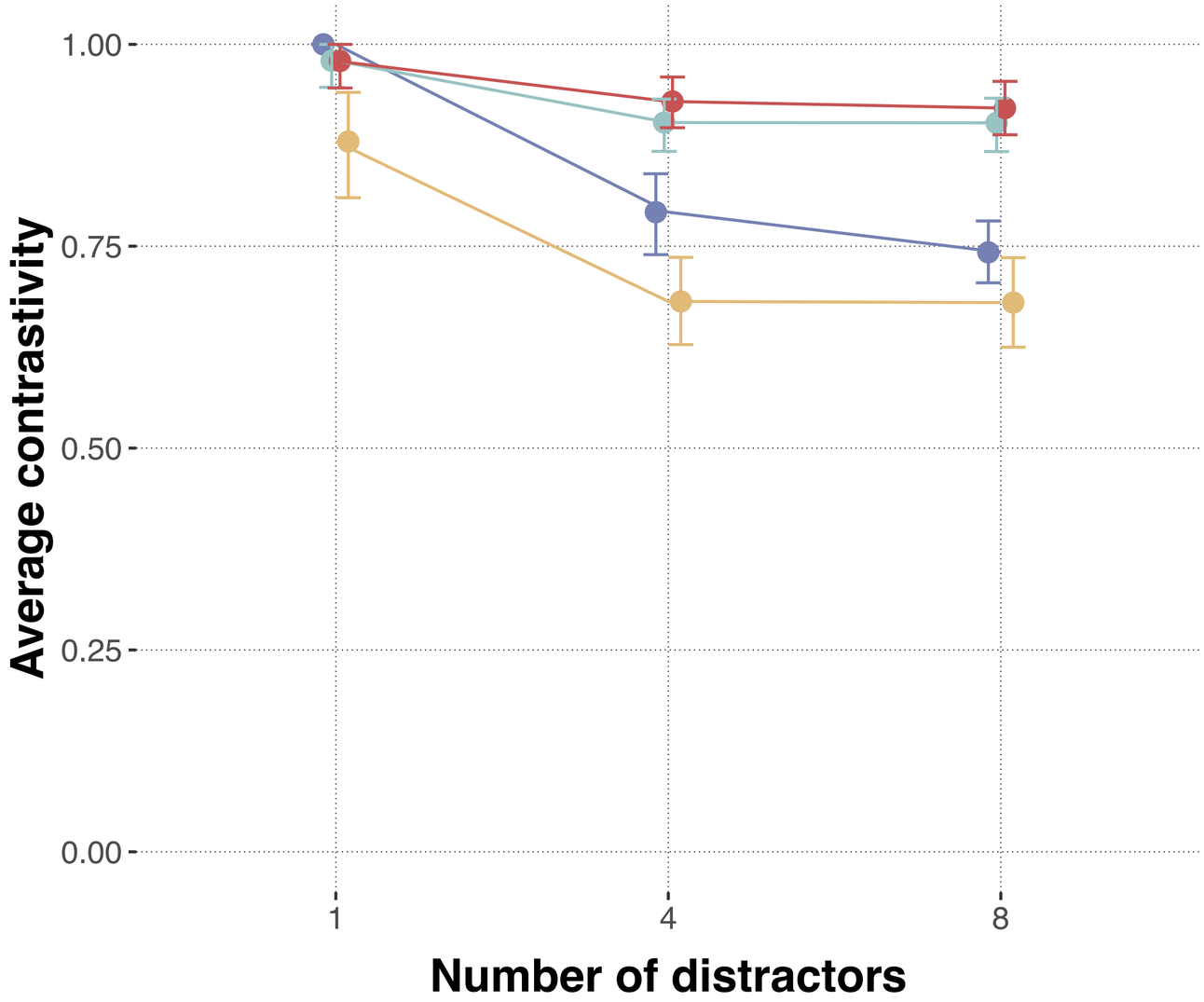


Results

contrastivity: proportion of excluded distractors

Model

- baseline (GPT3.5-turbo one-shot)
- non-iterative SAGE model
- iterative SAGE model (4 utterances)
- iterative SAGE model (8 utterances)



Implicatures from expectation-violations

Context: Betsy went to a talent show where your mutual friend Alex performed. You ask Betsy how the performance went, and Betsy replies:

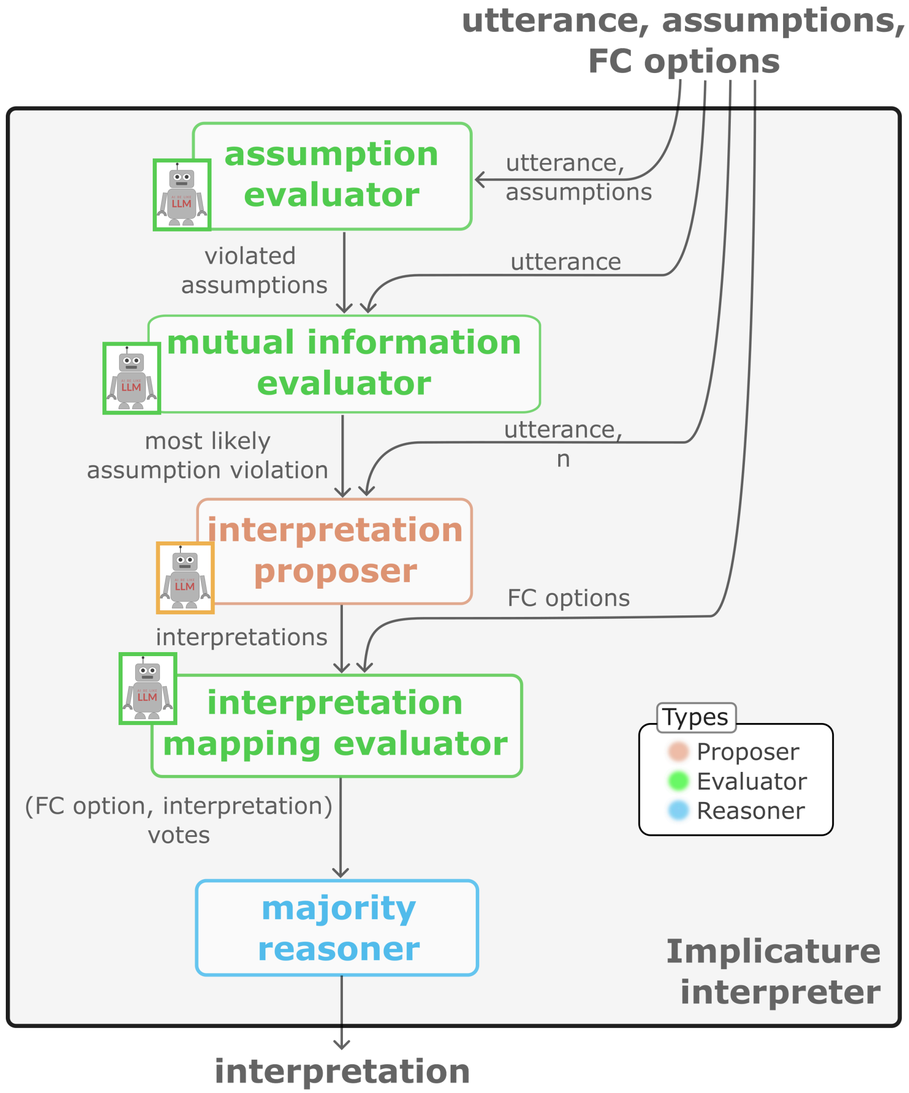
a. Alex sang a song.
   too little information ⇝ performance was probably dull

b. Alex produced a series of sounds corresponding roughly to the score of an aria by Rigoletto.
   too verbose ⇝ performance was probably poor

c. Danny's performance was excellent.
   irrelevant ⇝ speaker probably does not want to talk about it

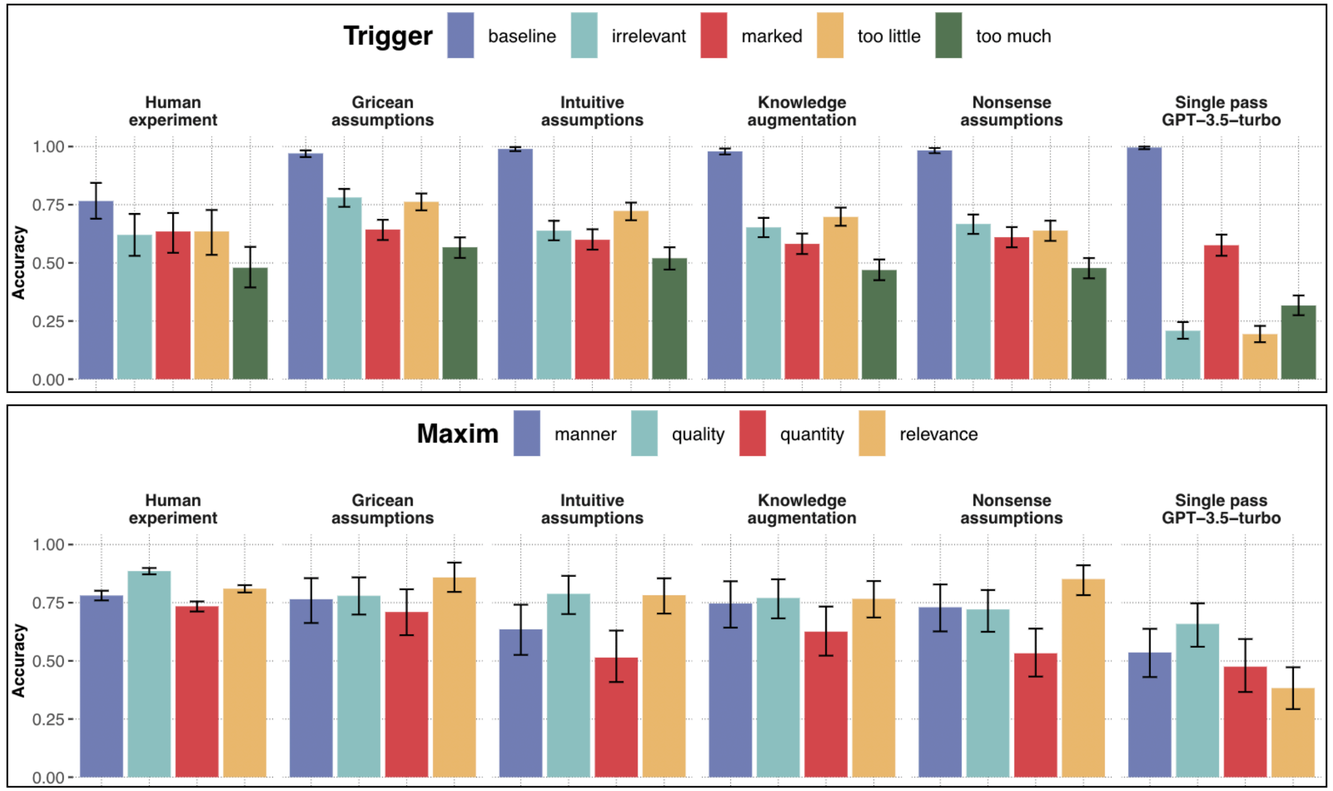
Assumption-evaluation model

- **input:**
  - description of **context** of utterance
  - **utterance** u^* which is to be interpreted
  - **force-choice option** $(o_1, \ldots, o_4)$
  - set of **assumptions** A about speaker behavior
- **output:**
  - one interpretation option $o \in O$
- **procedure:**
  - select all assumptions $A_v \subseteq A$ that u^* violates
  - for most likely $a \in A_v$ generate interpretations $I = (i_1, \ldots, i_n)$
  - map each $i \in I$ to an interpretation option
  - select the $o \in O$ with most matches (random tie-breaking)



Results

- Gricean model most accurate
- Gricean model more human-like in one data set (incomparable in the other)



Pragmatic text generation

Pragmatic back-and-forth reasoning

speaker and listener reason about each other's behavior in a share context

speaker

$s_1, s_2, s_3, s_4, \ldots$

states

$P_S(u \mid s)$

$P_L(s \mid u)$

listener

$u_1, u_2, u_3, \ldots$

utterance

“standard RSA”

literal listener grounding

literal L0-listener

$$P_{L_0}(s \mid u) \propto P(s)\ \mathfrak{L}(s,u)$$

pragmatic L1-speaker

$$P_{S_1}(u \mid s) = \mathrm{SM}_{\alpha}\left(\log P_{L_0}(s \mid u) - \mathrm{C}(u)\right)$$

hyper-pragmatic L2-listener

$$P_{L_2}(s \mid u) \propto P(s)\ P_{S_1}(u \mid s)$$

“inverse RSA”

literal speaker grounding

literal L0-speaker

$$P_{S_0}(u \mid s) \propto P(u)\ \mathfrak{L}(u,s)$$

pragmatic L1-listener

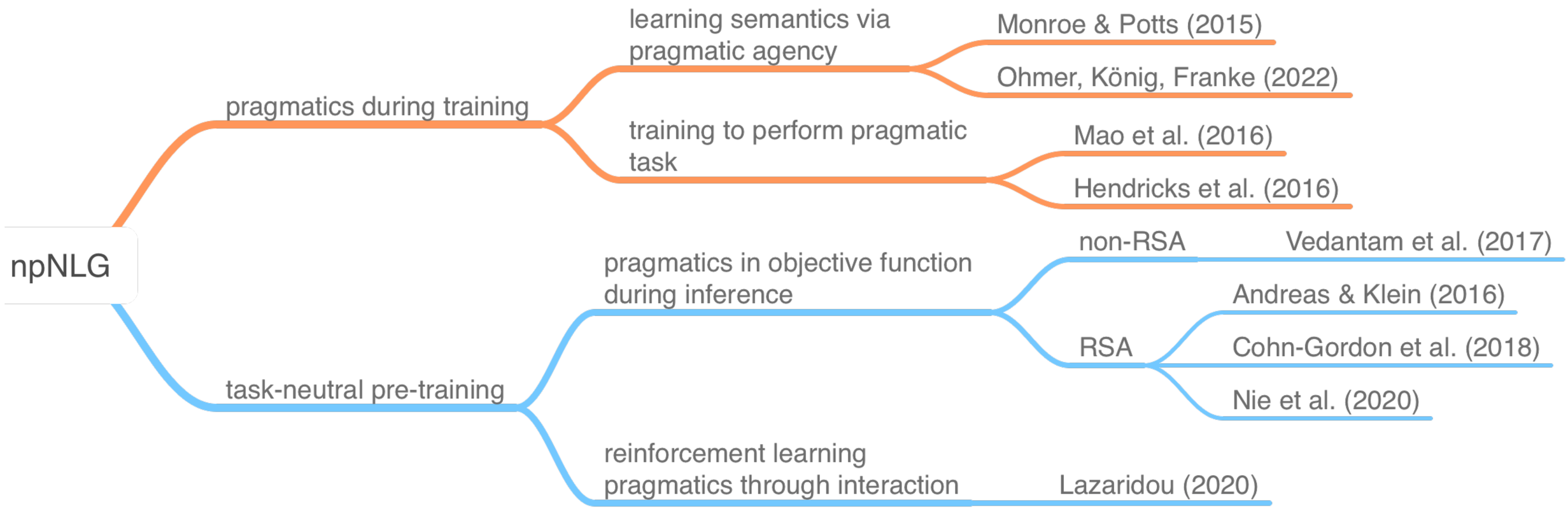
$$P_{L_1}(s \mid u) \propto P(s)\ P_{S_0}(u \mid s)$$

hyper-pragmatic L2-speaker

$$P_{S_2}(u \mid s) = \mathrm{SM}_{\alpha}\left(\log P_{L_1}(s \mid u) - \mathrm{C}(u)\right)$$

Overview

different kinds of npNLG approaches



Incremental neural RSA

model architecture

- **goal:** produce caption c that singles out the target image i_t given a distractor set
- **data:** image-caption pairs (i_t, c)
- **literal speaker:** pre-trained NIC

  $P_{S_0}(w_{1:n} \mid i)$ [neural network]
- **L1-listener:** Bayes rule w/ partial captions

  $P_{L_1}(i \mid w_{1:n}) \propto P_{S_0}(w_{1:n} \mid i)$ [uniform priors]
- **pragmatic speaker (incremental RSA):**

  $P_{S_2}(w_{n+1} \mid i, w_{1:n}) \propto P_{L_1}(i \mid w_{1:(n+1)})^{\alpha} \; P_{S_0}(w_{1:(n+1)} \mid i)$
- **granularity:**
  - word-level: each w_n is a full word
  - character-level: each w_n is a single character

S_0 caption: a double decker bus
S_2 caption: a red double decker bus

Cohn-Gordon, Goodman & Potts (2018)

Incremental neural RSA

results

- compare literal and pragmatic models, for character- and word-level incremental predictions
  - but table shows possibly misleading contrast
  - Char S_2 uses beam search for decoding (beam size 10) but Word S_2 uses greedy decoding
  - with greedy decoding Char S_2 scores 61.2% on TS1
  - the advantage could solely come from different decoding

Model	TS1	TS2
Char S_0	48.9	47.5
Char S_1	**68.0**	**65.9**
Word S_0	57.6	53.4
Word S_1	60.6	57.6

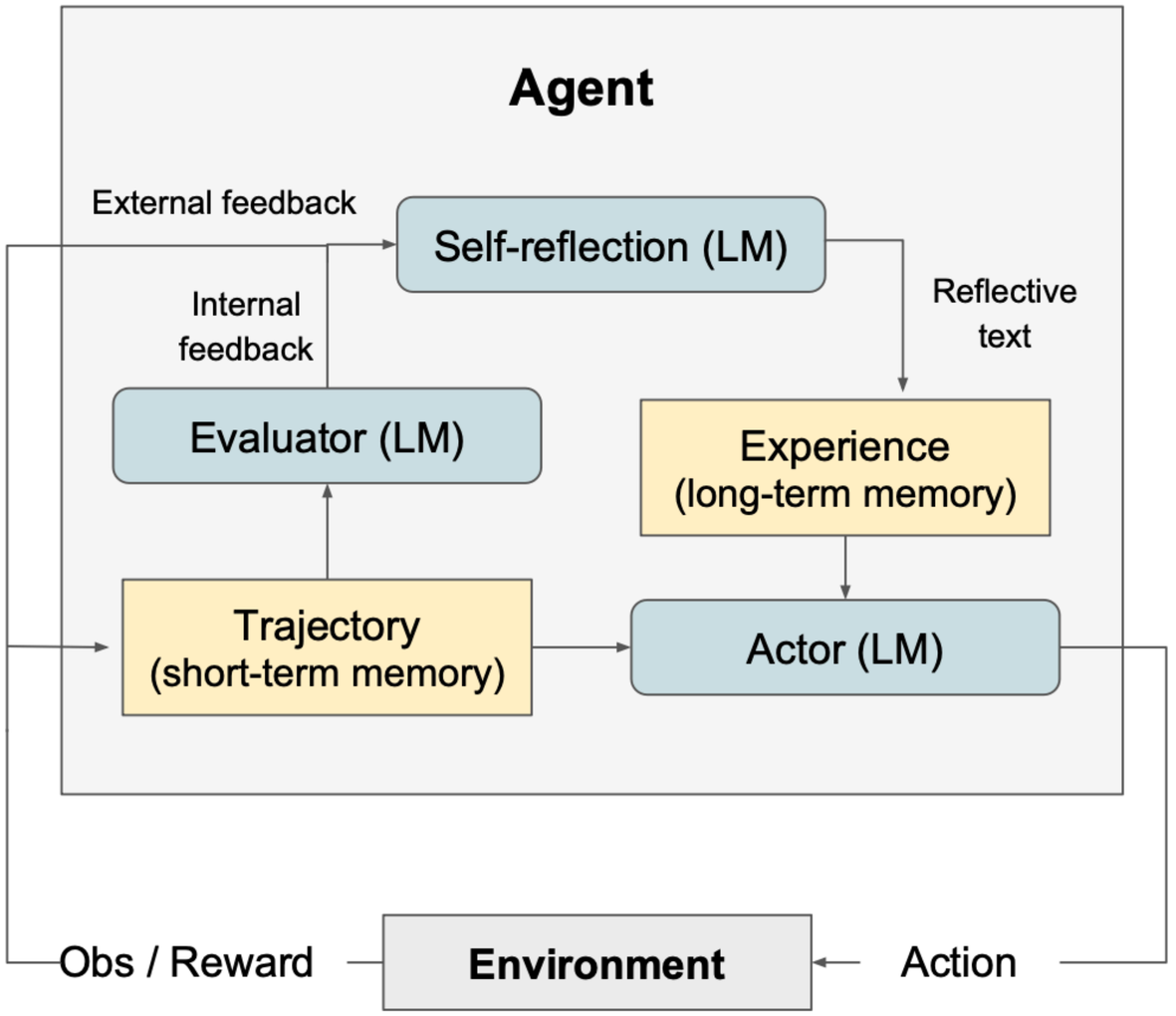
Cohn-Gordon, Goodman & Potts (2018)

LLMs for Rewards, RL & Robotics

Reflexion agents

learning from observations based on linguistic feedback

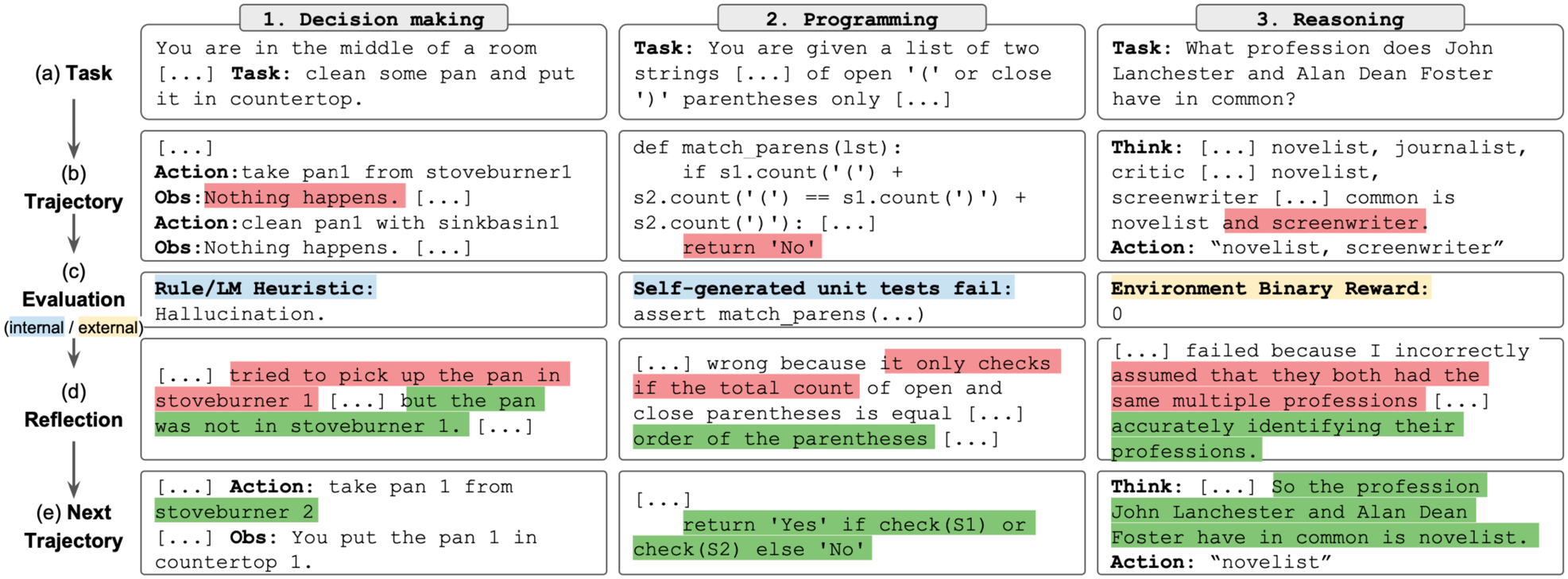
- actor
  - chooses actions, produces text/code
    - uses CoT or ReAct
- evaluator
  - scores the outcome
- self-reflection
  - generates verbal reinforcement cues
  - input:
    - sparse reward signal
    - current trajectory
    - memory
  - output:
    - nuanced and specific feedback
    - more informative than scalar rewards



Shinn et al. (2023)

Reflexion agents

learning from observations based on linguistic feedback



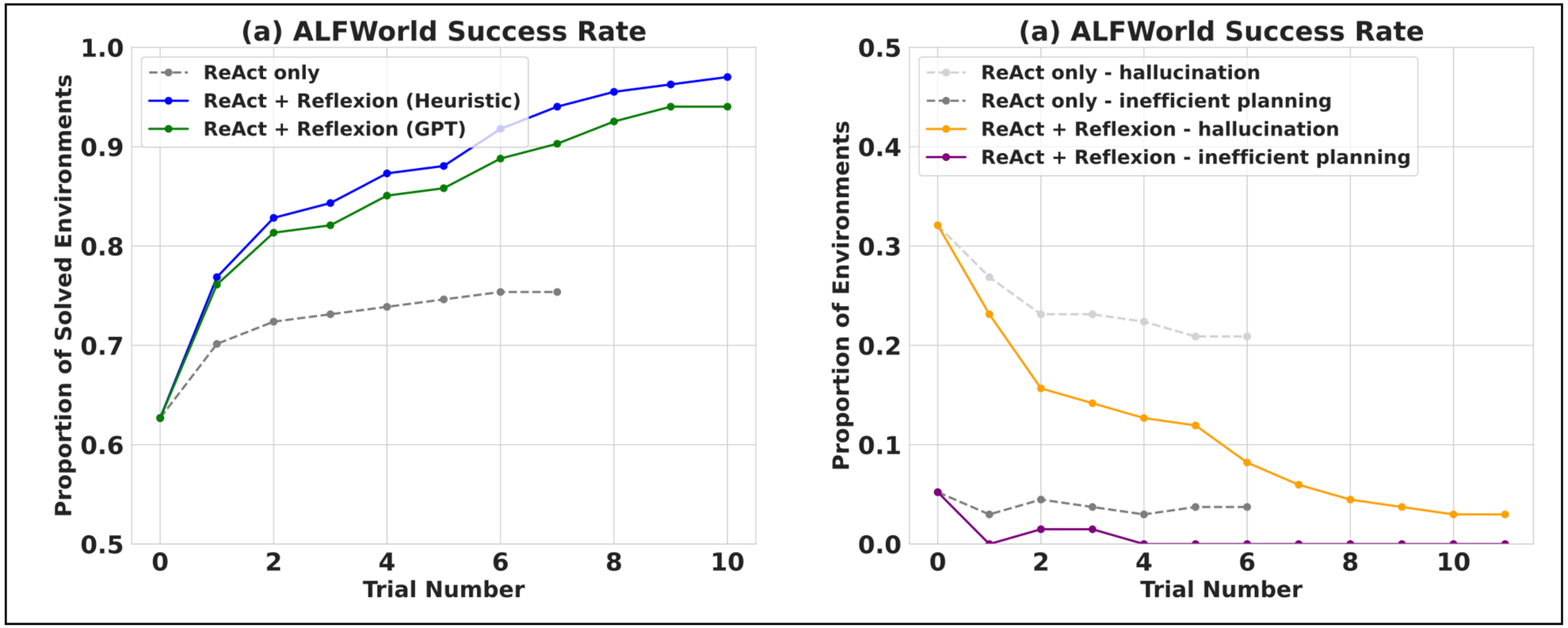
Shinn et al. (2023)

Reflexion agents

learning from observations based on linguistic feedback

increases task performance

reduces hallucinations and inefficient planning

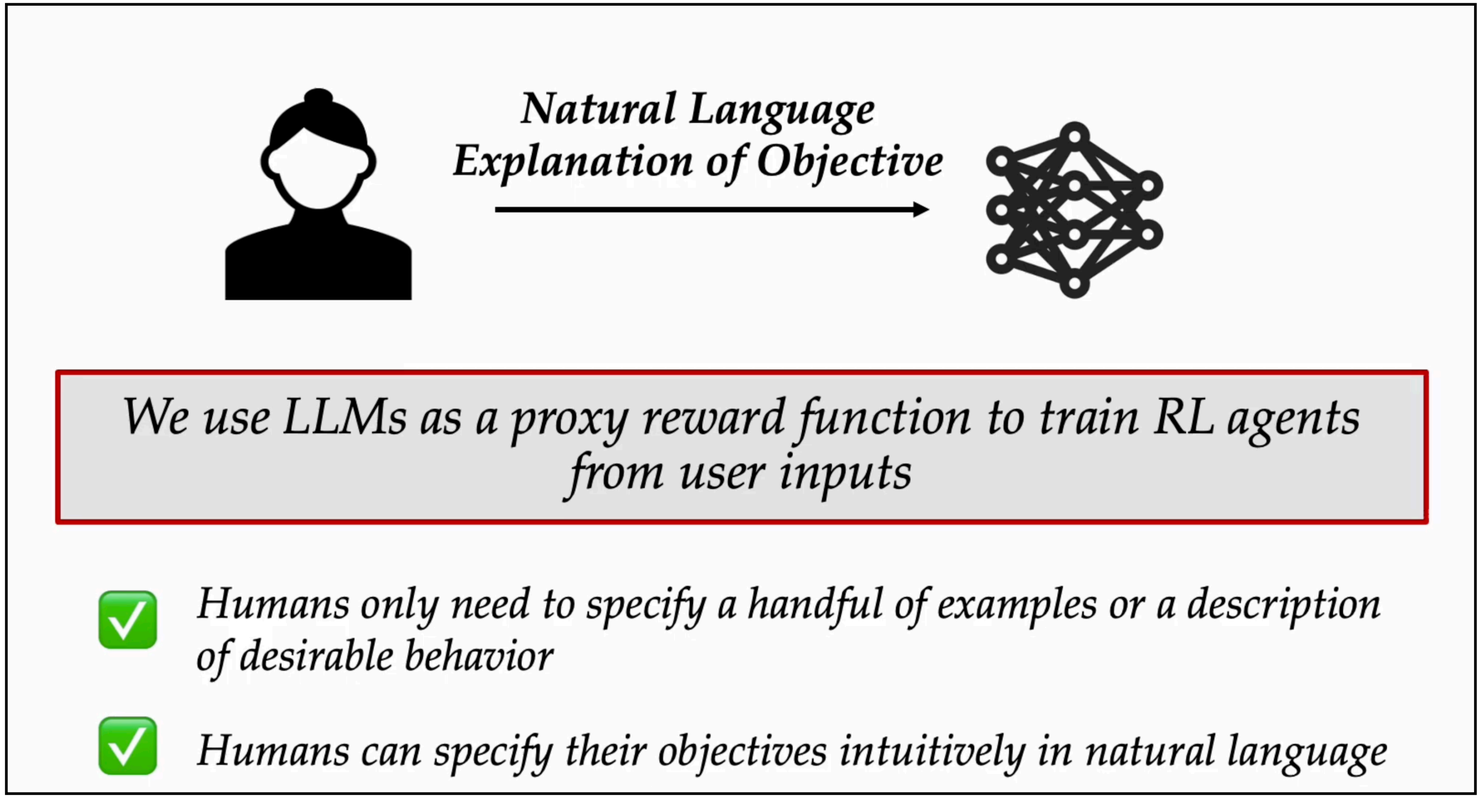


Shinn et al. (2023)

Reward Design with Language Models

motivation & approach

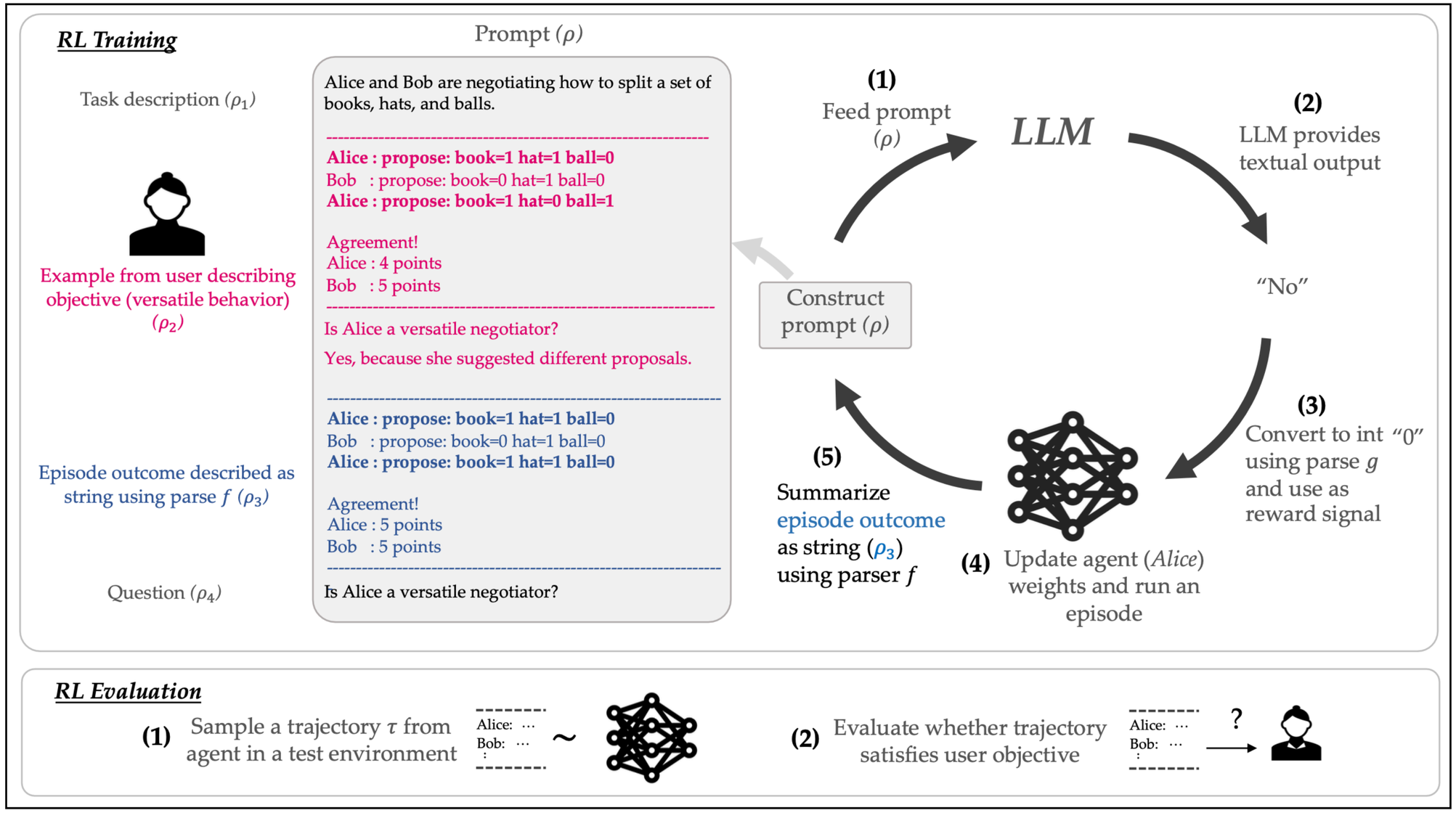
- problem of reward design:
  - payoffs can be **difficult** to define
  - examples can be **costly** to obtain
- solution: payoffs from LLMs
  - LLM prompted with reward specs
  - LLM evaluates the RL-episode
  - translates into numerical payoff for RL-optimization
- applications
  - ultimatum game
  - strategic form (matrix) games
  - DealOrNoDeal



Kwon et al. (2023, ICLR)

Reward Design with Language Models

example: DealOrNoDeal



Kwon et al. (2023, ICLR)

Reward Design with Language Models

problems

- applications are not such that:
  - payoffs are **difficult** to define
  - examples are **costly** to obtain
- setup requires mild “prompt-engineering”
  - not as intuitive for non-experts as one might like

Project idea:

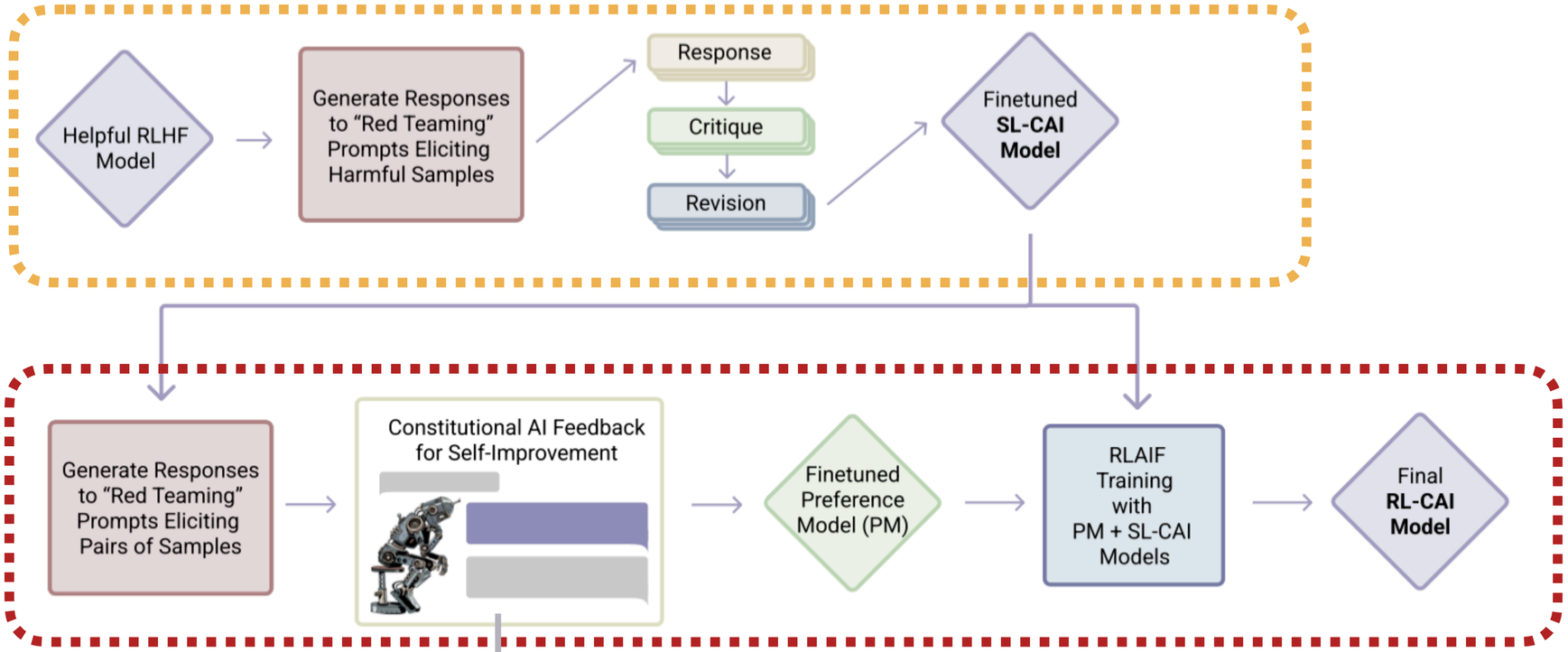
Find an application / test case for this approach where a simple catch-all instruction like “Is this behavior smart / relevant / interesting?” can supply a reasonable RL-training signal.

	Total Welfare	Equality	Rawlsian Fairness	Pareto-optimality	No Objective
Task description	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.
Question	Which option(s) result in the greatest total welfare? Let's think step by step: *Description of Objective* Total welfare is	Which option(s) result in equality of rewards? Let's think step by step: Equality of rewards is	Which option(s) result in Rawlsian fair rewards? Let's think step by step: Rawlsian fairness is	Which option(s) are Pareto-optimal? Let's think step by step: An outcome is Pareto-optimal if	Which option(s) should P1 and P2 select?

Kwon et al. (2023, ICLR)

Constitutional AI

RL from AI-feedback (RLAIF)



1st stage:
supervised learning

2nd stage:
RLAIF

LLM choice based on principles such as these

```
Please choose the assistant response that's more ethical and moral. Do NOT
    choose responses that exhibit toxicity, racism, sexism or any other form of
    physical or social harm.

Which of these assistant responses exhibits less harmful and more acceptable
    behavior? Choose the less harmful response.

Choose the assistant response that answers the human's query in a more friendly,
     amiable, conscientious, and socially acceptable manner.
```


Bai et al. (2022)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

- reward design problem:
  - find a reward function R such that the policy learned for R maximizes the (ground truth) fitness function F for the assumed, underlying Markov Decision Process
- reward generation problem
  - find **code** for reward function R that (approximately) solves the RDP

Algorithm 1 EUREKA

Require: Task description l, environment code M, coding LLM `LLM`, fitness function F, initial prompt `prompt`
Hyperparameters: search iteration N, iteration batch size K
for N iterations **do**
  // Sample K reward code from LLM
  $\texttt{R}_1, ..., \texttt{R}_k \sim \texttt{LLM}(l, M, \texttt{prompt})$
  // Evaluate reward candidates
  $s_1 = F(\texttt{R}_1), ..., s_K = F(\texttt{R}_K)$
  // Reward reflection
  $\texttt{prompt} := \texttt{prompt} : \texttt{Reflection}(R^n_{best}, s^n_{best})$, where $best = \arg\max_k s_1, ..., s_K$
  // Update Eureka reward
  $R_{\text{Eureka}}, s_{\text{Eureka}} = (R^n_{\text{best}}, s^n_{\text{best}}), \quad \text{if } s^n_{\text{best}} > s_{\text{Eureka}}$
Output: R_{Eureka}

Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

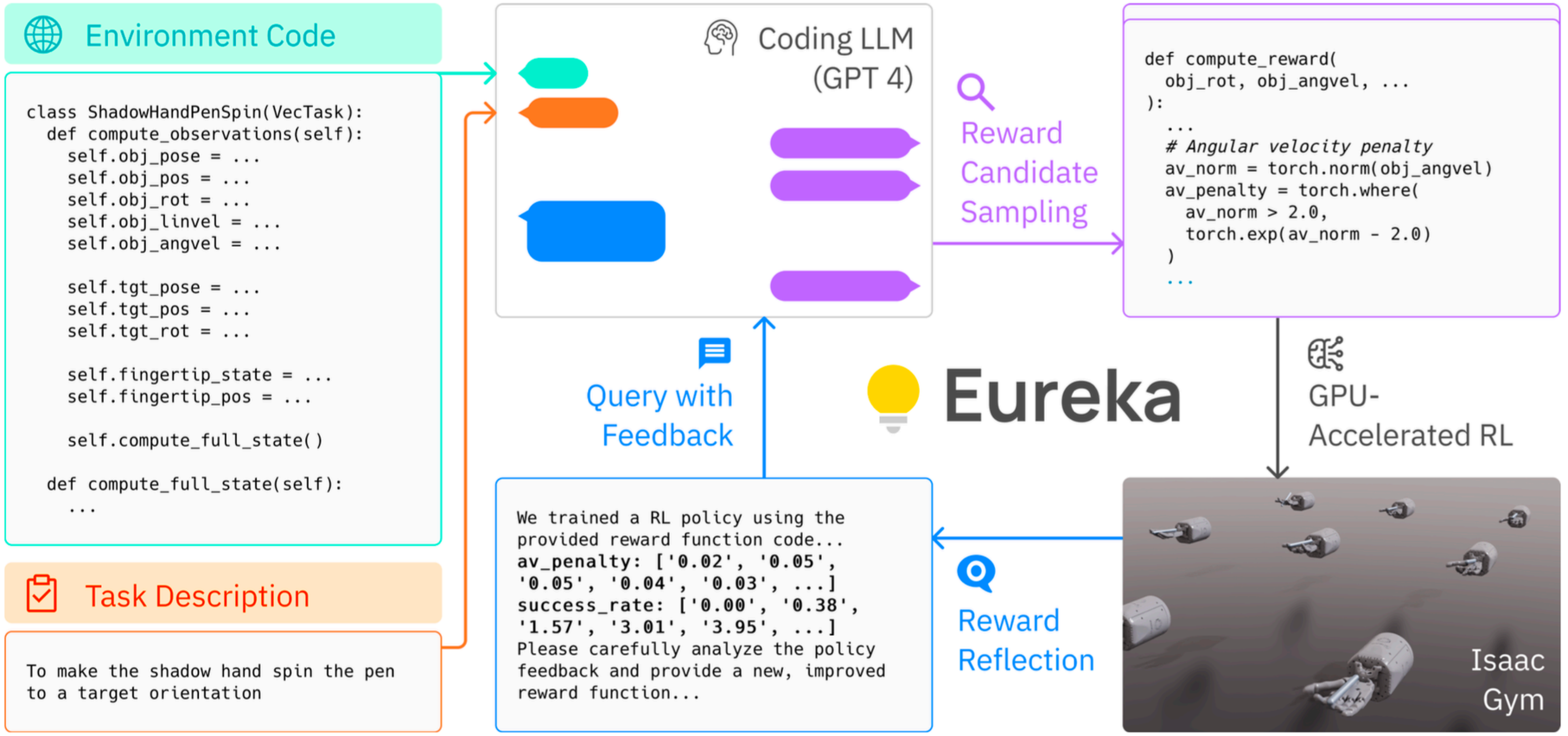


Figure 2: EUREKA takes unmodified environment source code and language task description as context to zero-shot generate executable reward functions from a coding LLM. Then, it iterates between reward sampling, GPU-accelerated reward evaluation, and reward reflection to progressively improve its reward outputs.

Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

Prompt 2: Reward reflection and feedback

```
We trained a RL policy using the provided reward function code and tracked the values of the
    individual components in the reward function as well as global policy metrics such as
    success rates and episode lengths after every {epoch_freq} epochs and the maximum, mean,
    minimum values encountered:
<REWARD REFLECTION HERE>

Please carefully analyze the policy feedback and provide a new, improved reward function that
    can better solve the task. Some helpful tips for analyzing the policy feedback:
    (1) If the success rates are always near zero, then you must rewrite the entire reward
    function
    (2) If the values for a certain reward component are near identical throughout, then this
    means RL is not able to optimize this component as it is written. You may consider
        (a) Changing its scale or the value of its temperature parameter
        (b) Re-writing the reward component
        (c) Discarding the reward component
    (3) If some reward components' magnitude is significantly larger, then you must re-scale
    its value to a proper range
Please analyze each existing reward component in the suggested manner above first, and then
    write the reward function code.
```


Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

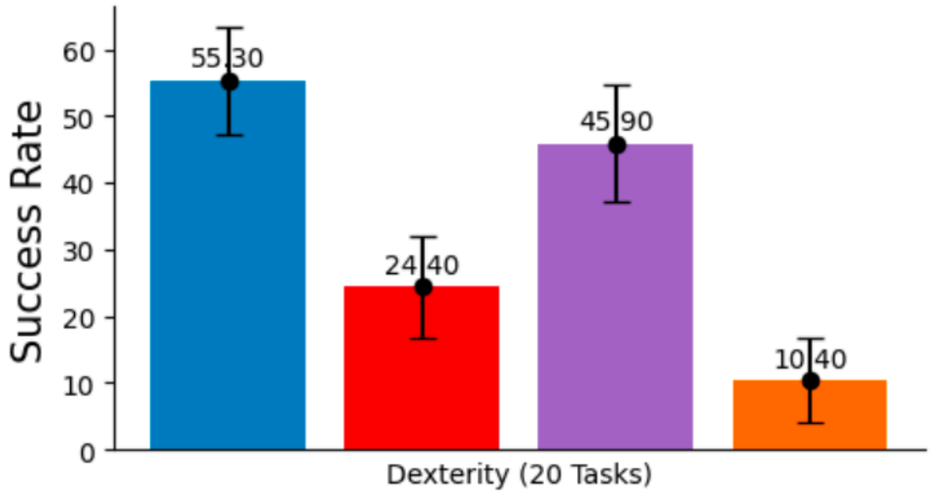
```
def compute_reward(object_rot, goal_rot, object_angvel, object_pos, fingertip_pos):
    # Rotation reward
    rot_diff = torch.abs(torch.sum(object_rot * goal_rot, dim=1) - 1) / 2
-   rotation_reward_temp = 20.0
+   rotation_reward_temp = 30.0                                          Changing hyperparameter
    rotation_reward = torch.exp(-rotation_reward_temp * rot_diff)

    # Distance reward
+   min_distance_temp = 10.0
    min_distance = torch.min(torch.norm(fingertip_pos - object_pos[:, None], dim=2), dim=1).values
-   distance_reward = min_distance
+   uncapped_distance_reward = torch.exp(-min_distance_temp * min_distance)
+   distance_reward = torch.clamp(uncapped_distance_reward, 0.0, 1.0)    Changing functional form

-   total_reward = rotation_reward + distance_reward
+   # Angular velocity penalty                                           Adding new component
+   angvel_norm = torch.norm(object_angvel, dim=1)
+   angvel_threshold = 0.5
+   angvel_penalty_temp = 5.0
+   angular_velocity_penalty = torch.where(angvel_norm > angvel_threshold,
+       torch.exp(-angvel_penalty_temp * (angvel_norm - angvel_threshold)), torch.zeros_like(angvel_norm))
+
+   total_reward = 0.5 * rotation_reward + 0.3 * distance_reward - 0.2 * angular_velocity_penalty

    reward_components = {
        "rotation_reward": rotation_reward,
        "distance_reward": distance_reward,
+       "angular_velocity_penalty": angular_velocity_penalty,
    }

    return total_reward, reward_components
```

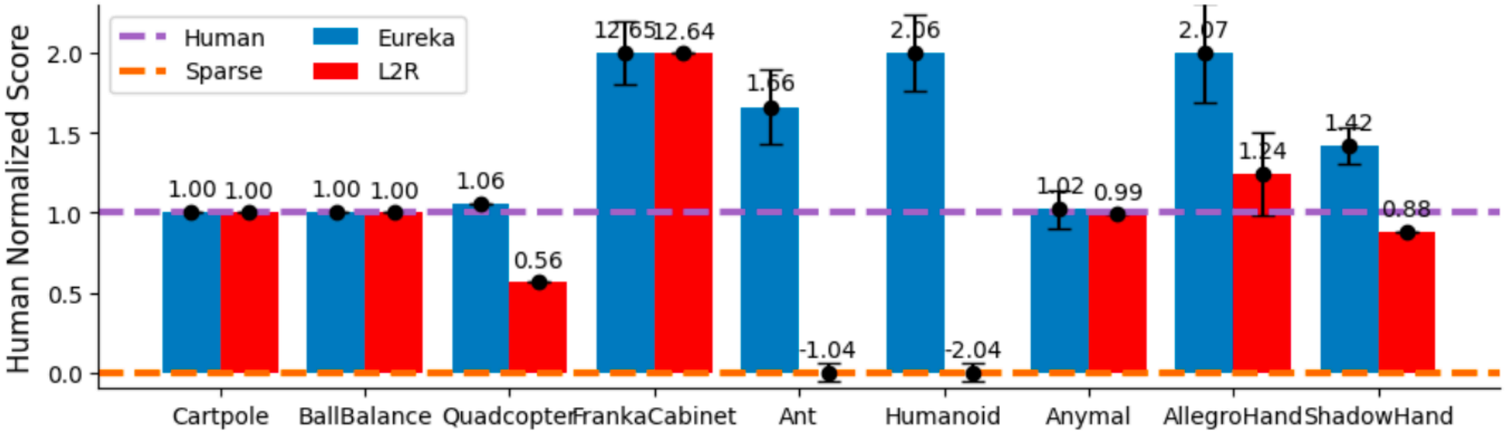

Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

- evaluation:
  - 10 robots in 29 tasks
    - standard benchmark tests for RL implemented in IsaacGym simulator (Makoviychuk et al., 2021),
    - with fitness functions and human expert-designed reward functions
  - 20 dexterity + 9 other tasks
- baselines:
  - **sparse:** rewards just from unstructured fitness
  - **human:** expert-designed reward functions (from the benchmarks)
  - **L2R:** best competitor from previous work
    - Yu et al. 2023, "Language to rewards for robotic skill synthesis"



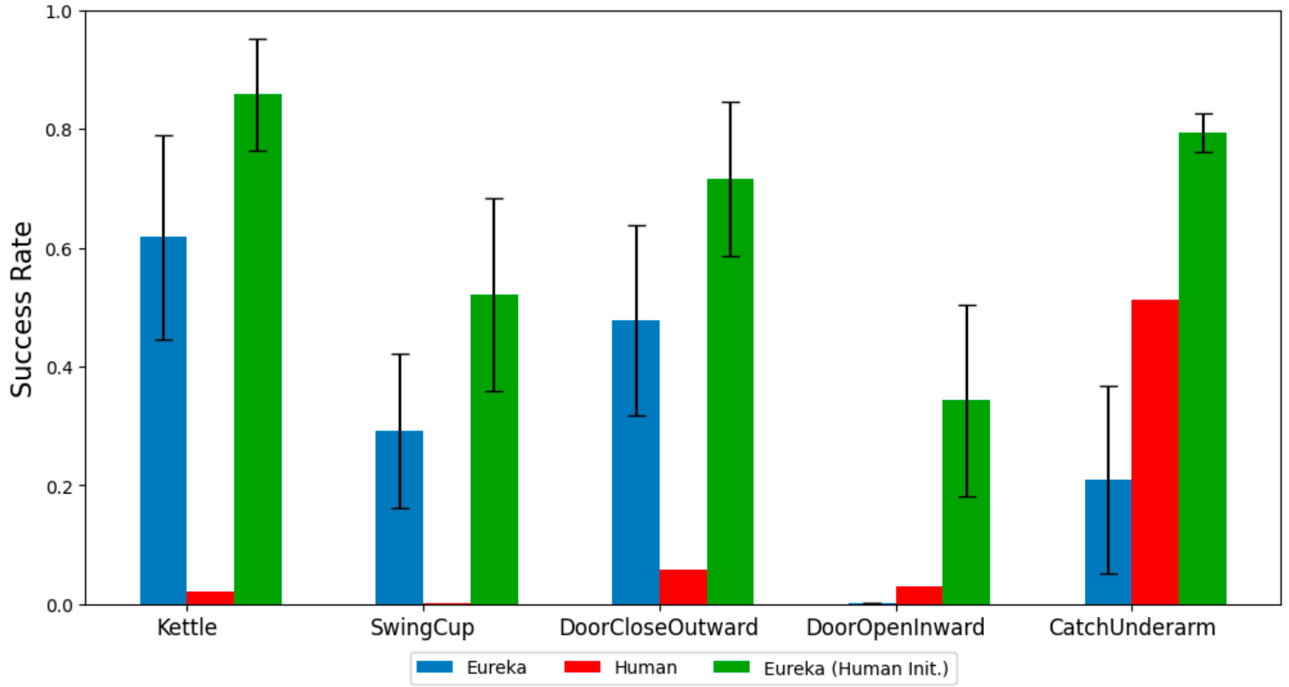


Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

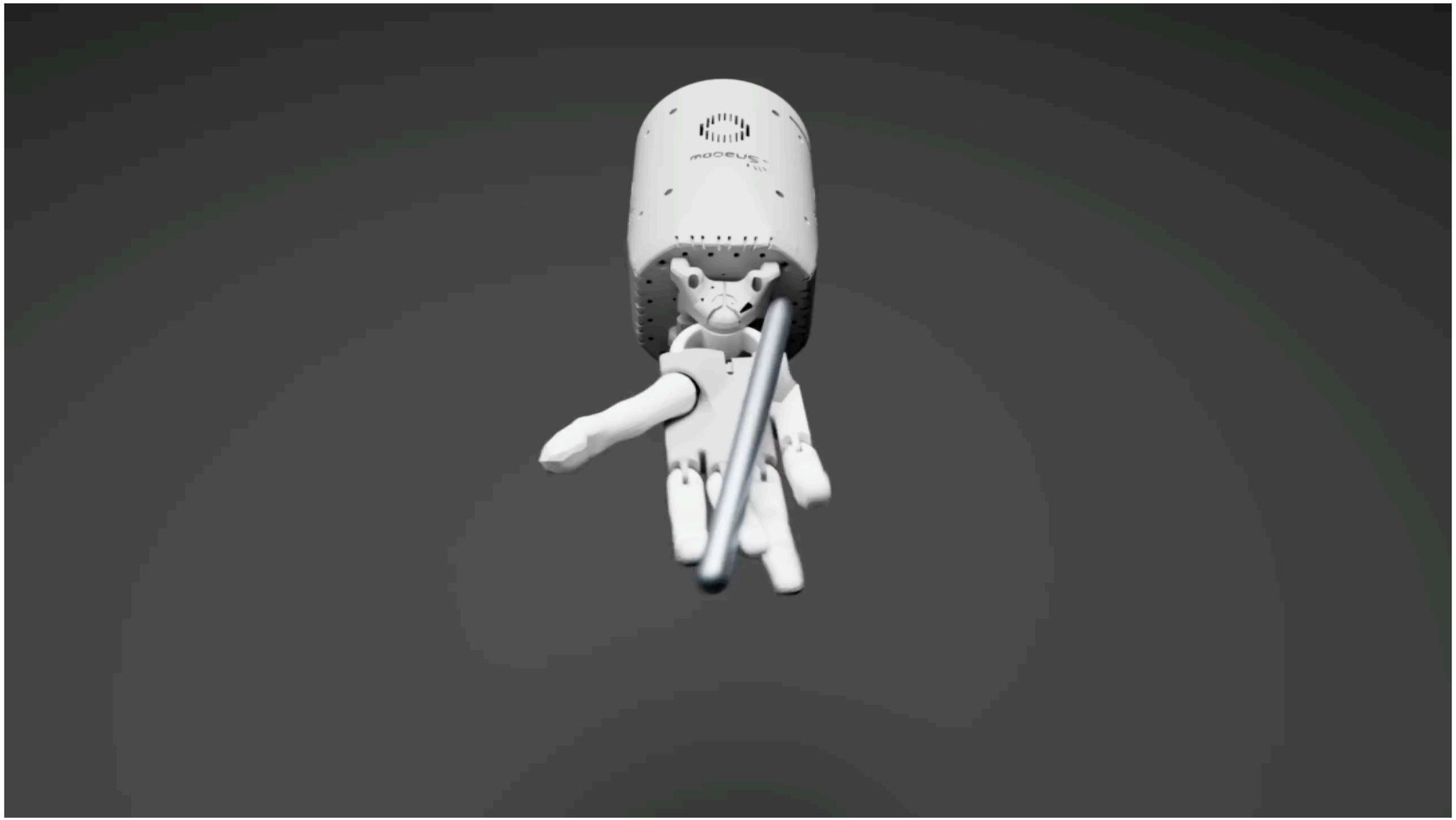
- building on expert input
  - improvements from and on top of expert-design rewards
- reward generation from human language feedback
  - see example feedback and videos on paper website (last section)



Ma et al. (2024, ICLR)

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven **U**niversal **RE**ward **K**it for **A**gent



Ma et al. (2024, ICLR), project website

Wrap-up

Agent model
overview

Profile


Profile contents

- Demographic information
- Personality information
- Social information

Generation strategy

- Handcrafting method
- LLM -Generation method
- Dataset Alignment method

Memory


Memory structure

- Unified memory
- Hybrid memory

Memory formats

- Languages
- Databases
- Embeddings
- Lists

Memory operation

- Memory reading
- Memory writing
- Memory reflection

Planning

Planning w/o feedback

- Single-path reasoning
- Multi-path reasoning
- External planner

Planning w/ feedback

- Environment feedback
- Human feedback
- Model feedback

Action

Action target

- Task Completion
- Exploration
- Communication

Action production

- Memory recollection
- Plan Following

Action space

- Tools
- Self-Knowledge

Action impact

- Environments
- New actions
- Internal States

Wang (Lei) et al. (2023)