Language Agents

From next token prediction to digital automation

Shunyu Yao
Autonomous agents to interact with the world

Agent

- Rule-based agents: manual design
- Learning-based agents: trial-and-error
- Language agents: reasoning to act

Environment

- Interact with humans / physical world
- Interact with games / simulation
- Interact with digital world (e.g., Internet)
Challenge 1: Accessible methods for general agents

Intensive to build
(Even for experts)

Takes millions of lines of rules (by domain experts)

Takes millions of training iterations (by RL experts)

Hard to generalize
Challenge 2: Scalable benchmarks for practical tasks

Practical
(Can build agents for useful tasks)

Scalable
(Easy data/reward collection)

(But not scalable)
(But not practical)
My research

Part 1. Benchmarking agents via digital automation

[NeurIPS’22, NAACL’22, ACL’23, NeurIPS’23, ICLR’24, ICLR’24]
My research

Part 1. Benchmarking agents via **digital automation**
[NeurIPS’22, NAACL’22, ACL’23, NeurIPS’23, ICLR’24, ICLR’24]

Part 2. Building **language agents** that reason to act
[EMNLP’20, ICLR’23, NeurIPS’23, NeurIPS’23]
My research

Part 1. Benchmarking agents via **digital automation**
[NeurIPS'22, NAACL'22, ACL'23, NeurIPS'23, ICLR'24, ICLR'24]

Part 2. Building **language agents** that reason to act
[EMNLP'20, ICLR’23, NeurIPS’23, NeurIPS’23]

Part 3. Principled **framework** for language agents
[TMLR'24]
Benchmarking agents via digital automation

WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents
Yao*, Chen*, Yang, Narasimhan. NeurIPS 2022
Digital automation

- Tremendous practical values, but little progress (think about Siri)
- Underlying research challenges:
  - Reasoning over real-world language (and other modalities)
  - Decision making over open-ended actions and long horizon
- Solving these is also key for robot navigation, planning, coordination
Agent benchmarks without these challenges

- **MiniWoB** (Shi et al., 2017)
  - Simulation environment
  - Synthetic text (if any)

- **TextWorld** (Côté et al., 2019)

- **BabyAI** (Chevalier-Boisvert et al., 2019)
  - Small action space
  - Short-horizon tasks
WebGPT (Nakano et al., 2021)

An agent, not a benchmark!

Desired benchmark

- Large complex environment
- Automatic reward function
- Research challenges

How do neural networks work?

Web browser

Task input

In its most basic form, a neural network has two layers: an input layer and an output layer[2]. The output layer is the component of the neural network that makes predictions[2]. In a feedforward network, information flows through the network in the following way: patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units[3]. The network learns by a feedback process called backpropagation, which involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network, working from the output units through the hidden units to the input units, going backward[2][4]. Over time, backpropagation causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide, so the network figures things out exactly as it should[2].

References:
3. How Do Neural Networks Really Work? [www.nickmccullum.com]
4. How Do Neural Networks Really Work? [www.nickmccullum.com]

Task output

Reward via professional annotators
WebShop

- **Large-scale complex environment** based on 1.16M Amazon products
- **Automatic reward** based on instruction and product attribute matching
- **Challenges** language and visual understanding, and decision making
WebShop is challenging

- Pre-trained image model (ResNet)
- Pre-trained language models (BERT, BART)
- Imitation learning
- Reinforcement learning

Getting all attributes requires long-horizon exploration!
WebShop enables sim-to-real transfer
Various follow-up methods and benchmarks for web interaction
• Testbed for industrial developments (e.g., Google, OpenAI)
• Inspired research on other real-world digital tasks (e.g., coding)
Coding benchmarks are becoming easy

HumanEval (Chen et al., 2021)

Our work [SCBGNY, Reflexion, NeurIPS’23] has reached >95%…
SWE-Bench

Input: a GitHub repo and an issue

Output: a file diff to resolve the issue

Evaluation: unit tests from pull request

Metadata
Repo: scikit-learn/scikit-learn
Issue #: 14858
Instance ID: scikit-learn__scikit-learn-14869
Pull Number: 14869
Created At: Aug 31, 2019
Base Commit: 1016f9f_

Problem Statement
HGBc with categorical_crossentropy fails silently on binary classification

```python
import numpy as np
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier

x = [[1, 0], [1, 0], [1, 0], [0, 1], [1, 1]]
y = [1, 1, 1, 0, 1]
gb = HistGradientBoostingClassifier(loss='categorical_crossentropy',
                                 min_samples_leaf=1)

gb.fit(x, y)
print(gb.predict([[1, 0]]))
print(gb.predict([[0, 1]]))

gives:

[[0]
[0]]

And binary_crossentropy works fine. categorical_crossentropy should either generalize or raise an error on binary classification.
Fing @NicolasHug @ogrisel
```

Test Patch
```
sklearn/ensemble/hist_gradient_boosting/tests/test_gradient_boosting.py

438     assert stumpclf.fit(x, y, n_iter=1).score(x, y, n_iter=1) == 1
439
420     def test_crossentropy_binary_problem():
421         # categorical_crossentropy should only be used if there
422         # are more than two classes present. PR #14869
423         x = [[1], [0]]
424         y = [0, 1]
425         gbtt = HistGradientBoostingClassifier(loss='categorical_crossentropy')
426         with pytest.raises(ValueError, match='`categorical_crossentropy` not suitable.'):  
427             gbtt.fit(x, y)
428
429         @pytest.mark.parametrize("scoring", ["none", "loss"])  
```
LLMs cannot solve SWE-Bench

At least not in a sequence-to-sequence setup

<table>
<thead>
<tr>
<th>Model</th>
<th>% Resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChatGPT-3.5</td>
<td>0.20</td>
</tr>
<tr>
<td>Claude 2</td>
<td>1.96</td>
</tr>
<tr>
<td>GPT-4*</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Summary

• Digital automation: a new frontier for autonomous agents
  • Tremendous practical values
  • Scalable environment
  • Bottleneck: scalable evaluation

• It requires sequential decision-making over open-ended language
  • LLMs or RL agents cannot solve it
  • Require a fundamentally new type of agents
Building language agents that reason to act

ReAct: Synergizing Reasoning and Acting in Language Models
Yao, Zhao, Yu, Du, Shafran, Narasimhan, Cao. ICLR 2023
LLMs can solve tasks using few examples

Next-token prediction

Training:
Next-token prediction

Inference:
Prompting for various NLP tasks

Figures: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning.

The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-,

and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.

- Zero-Shot (0S) is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This method provides maximum convenience, potential for robustness, and avoidance of spurious correlations (unless they occur very broadly across the large corpus of pre-training data), but is also the most challenging setting. In some cases it may even be difficult for humans to understand the format of the task without prior examples, so this setting is in some cases “unfairly hard.” For example, if someone is asked to “make a table of world records for the 200m dash”, this request can be ambiguous, as it may not be clear exactly what format the table should have or what should be included (and even with careful clarification, understanding precisely what is desired can be difficult). Nevertheless, for at least some settings zero-shot is closest to how humans perform tasks – for example, in the translation example in Figure 2.1, a human would likely know what to do from just the text instruction.

Figure 2.1 shows the four methods using the example of translating English to French. In this paper we focus on zero-shot, one-shot and few-shot, with the aim of comparing them not as competing alternatives, but as different problem settings which offer a varying trade-off between performance on specific benchmarks and sample efficiency. We especially highlight the few-shot results as many of them are only slightly behind state-of-the-art fine-tuned models. Ultimately, however, one-shot, or even sometimes zero-shot, seem like the fairest comparisons to human performance, and are important targets for future work.

Sections 2.1–2.3 below give details on our models, training data, and training process respectively. Section 2.4 discusses the details of how we do few-shot, one-shot, and zero-shot evaluations.

[Brown et al., 2020]
LLMs can reason to answer questions

**Chain-of-Thought Prompting**

Input:
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9.

[Wei et al., 2022]
<table>
<thead>
<tr>
<th>Company</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple (AAPL)</td>
<td>$2.40 trillion</td>
</tr>
<tr>
<td>Nvidia (NVDA)</td>
<td>$1.20 trillion</td>
</tr>
<tr>
<td>Microsoft (MSFT)</td>
<td>$2.50 trillion</td>
</tr>
</tbody>
</table>

Total market cap: $2.40 + $1.20 + $2.50 = $6.10 trillion

$6.10 trillion - $7 trillion = $0.90 trillion

You would need $0.90 trillion more.
Table 10: Example trajectories for Webshop predicted by ReAct
Published as a conference paper at ICLR 2023

<table>
<thead>
<tr>
<th>WebShop observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[B0061IVFZE] Brothers-ALL-Natural Fruit Crisps, Strawberry Banana, 0.42 - Ounce (Pack of 100)</td>
</tr>
<tr>
<td>$85.0</td>
</tr>
<tr>
<td>$18.99</td>
</tr>
<tr>
<td>[B092JLLYK6] Nature’s Turn Freeze-Dried Fruit Snacks - Banana Crisps - Perfect For School Lunches or an On-The-Go Snack - No Sugar Added, Non GMO, Gluten Free, Nothing Artificial (0.53oz) 6-Pack</td>
</tr>
<tr>
<td>$12.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WebShop action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action: click [B092JLLYK6]</td>
</tr>
</tbody>
</table>
ReAct: a new paradigm of agents that reason and act

- **Synergy** of reasoning and acting
- **Simple** and intuitive to use
- **General** across domains
Traditional agents: action space $A$ defined by the environment

- External feedback $o_t$
- Agent context $c_t = (o_1, a_1, o_2, a_2, \ldots, o_t)$
- 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
ReAct is simple and intuitive to use

ReAct supports

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

Example ReAct trajectory

\[ \text{Task: } \text{xxxxxx} \]

\[ \text{Thought: } \text{xxx} \]
\[ \text{Action: } \text{xxx} \]

\[ \text{Observation: } \text{xxxxxx} \]

\[ \text{Thought: } \text{xxx} \]
\[ \text{Action: } \text{xxx} \]

\[ \ldots \]
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?
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."

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."]
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!

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

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"]

Synergy: acting support reasoning, reasoning guides acting
ReAct is general and effective

<table>
<thead>
<tr>
<th></th>
<th>(NLP tasks)</th>
<th>(RL tasks)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PaLM-540B</td>
<td>Reason</td>
</tr>
<tr>
<td></td>
<td>HotpotQA (QA)</td>
<td>FEVER (fact check)</td>
</tr>
<tr>
<td>Reason</td>
<td>29.4</td>
<td>56.3</td>
</tr>
<tr>
<td>Act</td>
<td>25.7</td>
<td>58.9</td>
</tr>
<tr>
<td>ReAct</td>
<td>35.1</td>
<td>64.6</td>
</tr>
</tbody>
</table>
WebShop: One-shot generalization

Success Rate

<table>
<thead>
<tr>
<th></th>
<th>Best RL agent</th>
<th>ReAct</th>
<th>Act Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000 training steps</td>
<td>28.7</td>
<td>40</td>
<td>30.1</td>
</tr>
<tr>
<td>1 example trajectory</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32
SWE-Bench: Zero-shot generalization

Success Rate

Best LLM  \( 1.96 \)
ReAct* \( 12.47 \)
Devin (product) \( 13.86 \)

* SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

Paper coming out soon :)

0 example trajectory
Impact in 2023

Followup methods

<table>
<thead>
<tr>
<th>Method</th>
<th>(Reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>(Wang et al.)</td>
</tr>
<tr>
<td>ToolLLM</td>
<td>(Qin et al.)</td>
</tr>
<tr>
<td>AutoGen</td>
<td>(Wu et al.)</td>
</tr>
<tr>
<td>AutoGPT</td>
<td>(Richards et al.)</td>
</tr>
<tr>
<td>MetaGPT</td>
<td>(Hong et al.)</td>
</tr>
<tr>
<td>SwiftSage</td>
<td>(Lin et al.)</td>
</tr>
<tr>
<td>MM-ReAct</td>
<td>(Yang et al.)</td>
</tr>
<tr>
<td>ToolkenGPT</td>
<td>(Hao et al.)</td>
</tr>
</tbody>
</table>

Interdisciplinary research

<table>
<thead>
<tr>
<th>Research Area</th>
<th>(Reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creative art</td>
<td>(Sun et al.)</td>
</tr>
<tr>
<td>Healthcare</td>
<td>(Imrie et al.)</td>
</tr>
<tr>
<td>Robotics</td>
<td>(Hireche et al.)</td>
</tr>
<tr>
<td>Education</td>
<td>(Tabrizi et al.)</td>
</tr>
<tr>
<td>Diaster control</td>
<td>(Colverd et al.)</td>
</tr>
<tr>
<td>Networks</td>
<td>(Hamadanian et al.)</td>
</tr>
<tr>
<td>Chemistry</td>
<td>(Bran et al.)</td>
</tr>
<tr>
<td>Mathematics</td>
<td>(Liao et al.)</td>
</tr>
</tbody>
</table>

Industrial applications

| Application     | |
|-----------------| |
| Marketing agent | |
| Sales agent     | |
| Education agent | |
| Video edit agent| |
| Game NPC        | |
| Coding agent    | |
| Web agent       | |
| Customer support| |
ReAct enables discovery of a novel chromophore

**a**
```
from chemcrow.agents import ChemTools, ChemCrow
chemtools = ChemTools()
# Initialize ChemCrow object with toolset and LIM
crow = ChemCrow(chemtools.all_tools, need='plot', temp=1.0,
)
# Task definition
task = {
    "Find and synthesize a thiourea organocatalyst which accelerates a Diels-Alder reaction.
} # Execute ChemCrow
crow.run(task)
```

**b** Task: Find and synthesize a thiourea organocatalyst which accelerates a Diels-Alder reaction.

First, I need to find a thiourea organocatalyst that can accelerate the Diels-Alder reaction. I will perform a web search to find a suitable catalyst.

**Web Search tool**: Schreiner's thiourea catalyst

Now, I will obtain the SMILES. **Name2Smiles tool**: 
FC(F)F1c1cc(NC(=S)Nc2cc(F)(F)F)cc(F)(F)F)c2cc(F)(F)F1c1cc(F)(F)F1

I will plan a synthesis for Schreiner's thiourea catalyst.

**SynthesisPlanner tool**: Detailed synthesis plan
I will execute the synthesis.

**SynthesisExecutor tool**: Successful synthesis.

**d** ChemCrow workflows with experimental validation

Insect repellent (plan and execute) → **DEET**

Thiourea organocatalysts (plan and execute)

- Schreiner's catalyst
- Ricci's catalyst
- Takemoto's catalyst

Novel chromophore (clean data, train model, and predict)

**Synthesis step 1: Bromo Suzuki coupling**

**Synthesis step 2: Iodo Heck reaction**

[Bran et al., 2023]
Is next-token prediction enough for general problem solving?

Tree of Thoughts (Yao et al., 2023): no!
Next-token prediction cannot reason deliberately

**Question:** How to combine 2, 9, 10, 12 to get 24?
**Thought:** 12 * 2 = 24; 10 - 9 = 1; 24 * 1 = 24.
**Answer:** \((12 * 2) * (10 - 9) = 24\)

**Question:** How to combine 4, 5, 6, 10 to get 24?

**Initial tokens are hard to decide**

**Thought:** 10 * 6 = 60; 60 / 5 = 12; 12 * 4 = 48
**Answer:** \(((10 * 6) / 5) * 4 = 24\)

LLMs make linear token decisions without lookahead or backtrack!

[YYZSGCN, Tree of Thoughts, NeurIPS’23]
How do we fix next-token prediction?

- We took inspirations from human cognition
  - System 1: fast and automatic (~next-token prediction)
  - System 2: slow and deliberate (~control algorithm)

One of the oldest ideas in AI: Tree search
Natural language search: Curse of combinatoriality

- **Classical search (e.g., chess agent):**
  - A small, well-defined action space $A_{env}$
  - Can simulate external feedback $o_t \in O_{env}$
  - Can design/learn evaluation heuristics $f(a_t)$

- **Search in the space of thought $\mathcal{L}$:**
  - $\mathcal{L}$ is combinatorial and infinite!
  - No external feedback
  - Hard to enumerate or evaluate thoughts

\[\hat{a}_t \in \mathcal{L}\]

\[\text{Colorless green ideas sleep!}\]

\[\text{It is so annoying that I can think about any piece of text…}\]

\[\text{This is also a thought!}\]

\[\text{This is a thought}\]

\[\text{This is also a thought}\]
Tree of Thoughts: Blessing of compositionality

Thought: A semantically coherent unit of text that can be generated/evaluated by LLMs

Each token as thought
- Easy to generate
- Hard to evaluate

Each equation as thought
- Relatively easy to generate/evaluate
- A problem-specific tradeoff design

Whole reasoning as thought
- Easy to evaluate
- Hard to generate
Thought-level BFS

4 5 6 10

Generate

10 - 4 = 6 (Left: 5 6 6)
10 - 5 = 5 (Left: 4 5 6)
6 - 5 = 1 (Left: 1 4 10) …

Evaluate

10 - 4 = 6 (Left: 5 6 6)
10 - 5 = 5 (Left: 4 5 6)
6 - 5 = 1 (Left: 1 4 10) (top-b choices remaining) …

Generation Prompt: come up with ways to combine two of these numbers…

Evaluation Prompt: how likely are these 3 numbers to combine to 24…

Task success:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CoT</td>
<td>4%</td>
</tr>
<tr>
<td>ToT (ours)</td>
<td>74%</td>
</tr>
<tr>
<td>Tasks</td>
<td>Game of 24</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“Thought”</td>
<td>An equation</td>
</tr>
<tr>
<td>Steps</td>
<td>3</td>
</tr>
<tr>
<td>Search</td>
<td>BFS</td>
</tr>
<tr>
<td>Generation</td>
<td>proposal</td>
</tr>
<tr>
<td>Evaluation</td>
<td>simulation/</td>
</tr>
<tr>
<td></td>
<td>commonsense</td>
</tr>
<tr>
<td>CoT -&gt; ToT</td>
<td>4% -&gt; 74%</td>
</tr>
</tbody>
</table>

- Modular
- Flexible
- Performant
Rating Prediction

How will user_X rate the item "Kusco-Murphy Tart Hair"? The rating should be an integer between 1 to 5, with 1 being lowest and 5 being highest.

Recommender agent
(Wang et al., 2023)

Evaluator: simulate humans

Auction agent
(Dean et al., 2024)

Evaluator: simulate agents

Jailbreak agent
(Mehrotra et al., 2023)

Evaluator: simulate self
Summary

• Language agents: reasoning as internal actions
  • Reasoning and acting can be complementary (ReAct)
  • Reasoning and acting can be similarly planned (ToT)

• They address key limitations of LLMs and traditional agents
  • Ground LLMs with external feedback and internal control
  • Few-shot generalization to act in various new domains
Principled framework for language agents

CoALA: Cognitive Architectures for Language Agents
Sumers*, Yao*, Narasimhan, Griffiths. TMLR 2024
How do we make sense of various LLM systems?

digital circuits

Where should the field be going?
Soar cognitive architecture
Cognitive Architectures for Language Agents (CoALA)

• **Memory**: short and long term

• **Action space**: internal and external
  1. Reasoning (update short-term memory)
  2. Retrieval (read long-term memory)
  3. Learning (write long-term memory)
  4. Grounding (update external world)

• **Decision making**: choose an action
Modularize and compare language agents

<table>
<thead>
<tr>
<th>SayCan (Ahn et al., 2022)</th>
<th>ReAct (Yao et al., 2022b)</th>
<th>Voyager (Wang et al., 2023a)</th>
<th>Generative Agents (Park et al., 2023)</th>
<th>Tree of Thoughts (Yao et al., 2023)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Long-term Memory</strong>&lt;sup&gt;5&lt;/sup&gt;</td>
<td><strong>External Grounding</strong></td>
<td><strong>Internal Actions</strong></td>
<td><strong>Decision Making</strong></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>physical digital</td>
<td>reason/retrieve/learn</td>
<td>evaluate</td>
<td></td>
</tr>
<tr>
<td>procedural</td>
<td>digital</td>
<td>reason/retrieve/learn</td>
<td>propose</td>
<td></td>
</tr>
<tr>
<td>episodic/semantic</td>
<td>digital/agent</td>
<td>reason</td>
<td>propose</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>reason</td>
<td>propose, evaluate, select</td>
<td></td>
</tr>
</tbody>
</table>
Langauge agents

Benchmarks
[NeurIPS’22, NAACL’22, ACL’23, NeurIPS’23, ICLR’24, ICLR’24]

Methods
[EMNLP’20, ICLR’23, NeurIPS’23, NeurIPS’23]

Frameworks
[TMLR’24]

Other topics

- Computer vision and graphics [NeurIPS’18]
- Developmental psychology [NeurIPS’19, CogSci’20]
- Reinforcement learning and control [ICLR’22, CVPR’23]
- Human-computer interaction [DIS’24 submission]
- Information Retrieval [ACL’24 submission]
- Theory [ACL’21]

Future work
Future work #1: Train models for agents

Establish model-agent synergy:

- Improve “agent capabilities” like planning, self-evaluation, calibration..
- Open-source agent backbone model
- Next trillion tokens for model training

Data → Usage

<table>
<thead>
<tr>
<th>Model</th>
<th>ReAct prompting</th>
<th>ReAct finetuning</th>
<th>ReAct prompting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Llama2-13B</td>
<td>21.2</td>
<td>34.4</td>
<td>31.4</td>
</tr>
</tbody>
</table>

[FireAct: Toward Language Agent Fine-tuning. To COLM’24]
Future work #2: Teach and discover knowledge

Through the lens of CoALA, these new applications require:

- Flexible learning and retrieval
- Intrinsic motivation (e.g., curiosity)
The most powerful neural networks ever built shouldn’t just answer questions or draft emails. They should be used to automate every aspect of our life, society, and science.
Thanks to my committee

- Danqi: thanks for your great students :)
- Tom: thanks for all the classical insights :) 
- Sanjeev: thanks for the retreat and retweet :) 
- Ben: thanks for making me feel old :) 
- Tatsu: thanks for shaping my talk :)


Thanks to my advisor and friends
Thanks to my collaborators

Karthik Narasimhan  Tom Griffiths  Yuan Cao
Noah Shinn  Howard Chen  John Yang
Carlos Jiminez  Ted Sumers

Akshara Prabhakar  Alex Wettig  Ashwin Gopinath
Austin Wang  Baian Chen  Ben Shi
Binghui Peng  Chang Shu  Christos Papadimitriou
Chuang Gan  Dan Friedman  Dian Yu
Edward Berman  Ehsan Shareghi  Fandong Meng
Federico Cassano  Izhak Shafran  Jeffery Zhao
Jeffrey Stanton  Jens Tuyls  Jiangnan Li
Jie Zhou  Jing Li  Josh Tenenbaum
Kexin Pei  Matthew Hardy  Matthew Hausknecht
Michael Tang  Mingyu Ding  Mo Yu
Nan Du  Nigel Collier  Ofir Press
Ping Luo  Rohan Rao  Runzhe Yang
Sham Kakade  Tao Yu  Tom McCoy
Wenjie Pang  Xiangyang Mou  Yang Zhang
Yao Mu  Yi Gu  Yisi Sang
Yuqian Sun  Zhiyong Wu  Zhou Xiao