AI Security

Introduction to Agentic AI

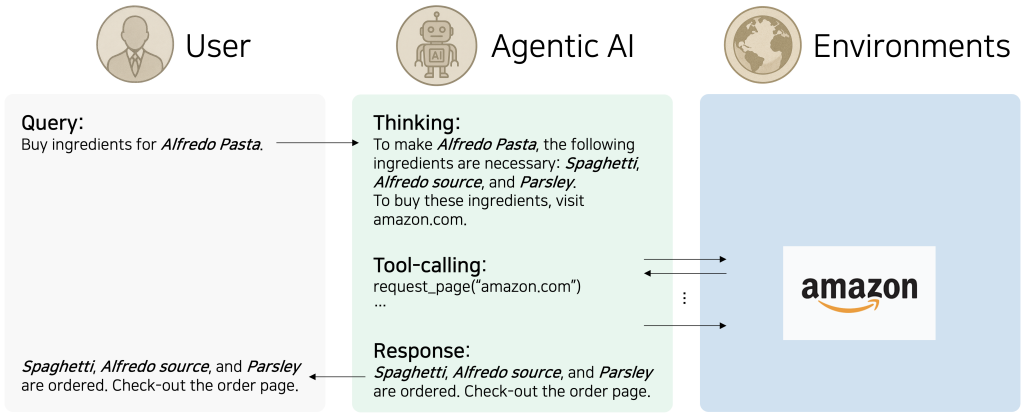
Sangdon Park

POSTECH

May 19, 2026

Outline

Era of Agentic AI



- What's under the hood? We will consider the following questions:
  - What's the core components and how do they work?
  - How to tame LLMs for Agentic AI?

Outline

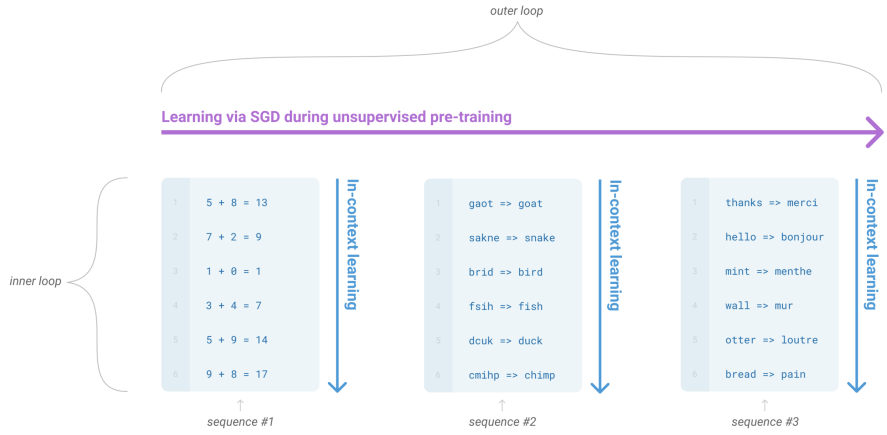
From Model Learning to Context Learning

Rethinking on Learning?

Learning by Model Update $\Longrightarrow$ In-Context Learning

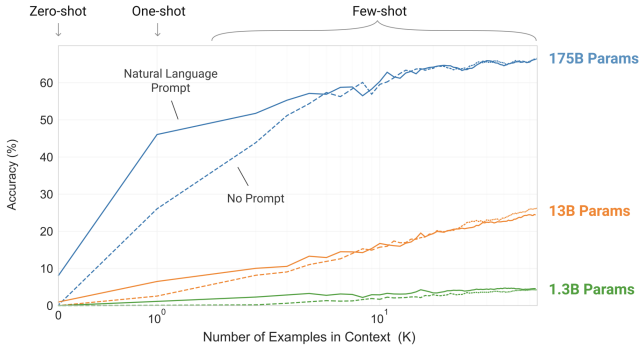
- Previous: update model parameters
- Part of current trends: preconditioning via prompting
  1. (observation) in-context learning empirically works [Brown et al., 2020]
  2. (interpretation) in-context learning $\approx$ gradient descent [Von Oswald et al., 2023]

Language Models are Few-Shot Learners [Brown et al., 2020]



- Outer loop: conventional model learning (*e.g.*, pretraining)
- Inner loop: in-context learning (*e.g.*, providing labeled examples in context)

Does In-Context Learning Work?



- Evaluated in toy tasks (*i.e.*, "character manipulation" tasks)
- Some interesting questions:
  - Why does it work?
  - When in-context learning work? (or any assumptions?)
  - Is the LLM adaptive to new tasks?

Why Does In-Context Learning Work? [Von Oswald et al., 2023]

Transformers Learn In-Context by Gradient Descent

Johannes von Oswald [1 2] **Eyvind Niklasson** [2] **Ettore Randazzo** [2] **João Sacramento** [1]
Alexander Mordvintsev [2] **Andrey Zhmoginov** [2] **Max Vladymyrov** [2]

- Interpret in-context learning theoretically
- In-context learning in Transformer forward passes $\approx$ gradient-based optimization (under some conditions)
  - "When training Transformers on auto-regressive tasks, in-context learning in the Transformer forward pass is implemented by gradient-based optimization of an implicit auto-regressive inner loss constructed from its in-context data."

Recall Gradient Descent

Regression Objective

$$L(W) = \frac{1}{2N} \sum_{i=1}^{N} \| W x_i - y_i \|^2$$

- We assume the linear regression problem.
- W: parameters of a linear model
- N: the number of data points
- x_i: an example
- y_i: a label

One Step of Gradient Descent

Update Rule

$$W' \leftarrow W - \frac{\eta}{N} \sum_{i=1}^{N} r_i x_i^T \qquad \text{where} \qquad N \cdot \nabla_W L(W) = \sum_{i=1}^{N} \underbrace{(Wx_i - y_i)}_{r_i} x_i^T = \sum_{i=1}^{N} r_i x_i^T$$

- η: a learning rate
- r_i: residual – measures the degree of prediction error

Effect of One Step of GD on Prediction

Prediction

$$\hat{y}_j = W' x_j = \left(W - \eta' \sum_{i=1}^{N} r_i x_i^T \right) x_j = \underbrace{W x_j}_{:=\hat{y}_j^{\text{old}}} \underbrace{- \eta' \sum_{i=1}^{N} r_i (x_i^T x_j)}_{\Delta \hat{y}_j}$$

- x_j: a new example
- $W' \leftarrow W - \eta' \sum_{i=1}^{N} r_i x_i^T$: one step of GD, where $\eta' = \frac{\eta}{N}$
- $\Delta \hat{y}_j$: prediction correction – this has an interesting structure

$$\sum_{i=1}^{N} r_i (x_i^T x_j) = \sum_{i=1}^{N} (\text{residual}) \times (\text{similarity})$$

  - (residual) $= W x_i - y_i$
  - Ignoring η', the "new" prediction is corrected by prediction errors (=residual) on seen and similar examples.

Recall Attention

Linear Attention

$$\text{Attn}(q) = \sum_i v_i (q^T k_i) = \sum_i (\text{value}) \times (\text{similarity})$$

- q: query – *e.g.*, an embedding vector of the current token
- k: key – *e.g.*, an id for a previous token's embedding vector
- v: value – *e.g.*, an embedding vector for the chosen key
- Here, assume that linear attention without Softmax – as you know, the following is the original attention:

$$\text{Attn}(q) = \sum_i a_{i,j} v_j \qquad \text{where} \qquad a_{i,j} := \frac{\exp(q_i^T k_j)}{\sum_{j'} \exp(q_i^T k_{j'})}$$

Mapping Between Prediction Correction and Linear Attention

Mapping Trick

$$\underbrace{q = x_j}_{(=\text{current token})} \qquad \underbrace{k_i = x_i}_{(=\text{a previous token})} \qquad \underbrace{v_i = r_i = Wx_i - y_i}_{(=\text{residual of the previous token})}$$

- consider the following few shot prompt:

$$1 + 2 = 3$$
$$1 + 3 = 4$$
$$1 + 4 = ?$$

- $q = x_j$: *e.g.*, an embedding vector of "?" along with its context from previous tokens
- $k_i = x_i$: *e.g.*, a key embedding vector of "4" along with its context (*e.g.*, "$1 + 3 =$"
- $v_i = r_i$: *e.g.*, (an embedding for the prediction of 3) − (an embedding for 4)
  - assume that it is given, but can Transformer infer this?

Prediction Correction by GD = Linear Attention

Comparison

$$\Delta \hat{y}_j = \sum_i r_i (x_i^T x_j)$$

$$\text{Attn}(x_j) = \sum_i v_i (q_j^T k_i) = \sum_i r_i (x_j^T x_i)$$

- Exactly $\Delta \hat{y}_j = \text{Attn}(x_j)$

Prediction Update by GD = Attention Residual Update

Comparison

$$\text{(Prediction Update)} \quad \hat{y}_j = \hat{y}_j^{\text{old}} - \eta' \underbrace{\Delta \hat{y}_j}_{\text{correction}}$$

$$\text{(Attention Update)} \quad h'_j = h_j + \underbrace{\text{Attn}(h_j)}_{\text{correction}}$$

- What is the implication?
  - parameter update = attention rewiring?

Recap

- A very LLM is a meta learner.
  - In-context learning is working, empirically validated.
  - Actually, in-context learning in auto-regressive Transformer is very related to learning by one-step GD.
- Then, what is needed for learning agentic AI?
  - in-context learning?
  - context engineering?
  - post-training?
  - more few shots for in-context learning v.s. larger model?

Outline

Agentic AI

Components of Agentic AI

$$\text{Agentic AI} = \text{LLM} + \text{Memory} + \text{Tools} + \text{Planning} + \text{Feedback}$$

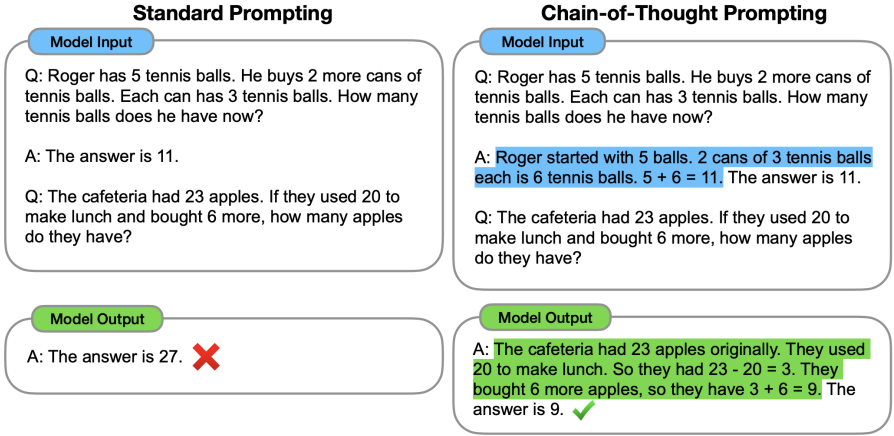
- LLM: language understanding and generation
- Memory: state representation (*e.g.*, maintaining context)
- Tools: methods to interact with external worlds
- Planning: divide a goal into sub-goals
- Feedback (or reflection): self-verification to re-planning
- How do they actually implemented?

Outline

In this class, we will see the following:

- CoT – an LLM is a reasoning model [Wei et al., 2022]
- Toolformer – an LLM can call tools [Schick et al., 2023]
- ReAct – combine reasoning and acting *i.e.,* a foundation of Agentic AI [Yao et al., 2022]
- Reflexion – an LLM can be self-improving with feedback [Shinn et al., 2023]
- Harness design – system engineering is getting important (report by Anthropic)

Chain-of-Thought [Wei et al., 2022]



- Implication: LLMs can generate right reasoning with proper context without reasoning supervision during pretraining
- LLM can do reasoning (✓), tool-calling (✗), self-verification (✗), remembering long context (✗)

Toolformer [Schick et al., 2023]

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.

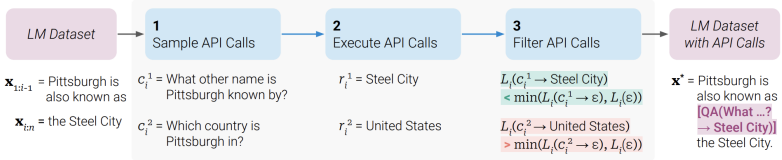
Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

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.

- LLMs can teach themselves to use external tools with simple SFT.
  - LLMs know how to use API so SFT simply encourages this.
  - See details for the paper.
- LLM can do reasoning (✗), tool-calling (✓), self-verification (✗), remembering long context (✗)

Data Collection Pipeline for SFT (1/3)



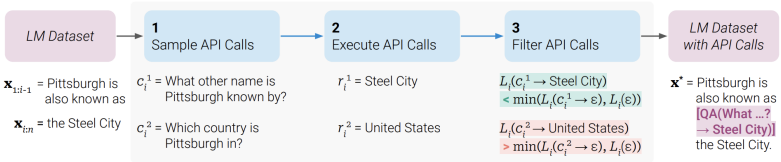
Sample API Calls:

1. For each i-th prefix (*e.g.*, "Pittsburgh is also known as",
2. Choose candidate positions based on the following quantity:

$$p_M(\texttt{<API>} \mid P(x), x_{1:i-1})$$

   - M: an LLM, <API>: a special token
   - $P(x)$: a system prompt to encourage API calling.
3. Sample multiple API calls, *e.g.*, <API> $a_c(i_c)$ </API>
   - a_c: API name, i_c: API input

Data Collection Pipeline for SFT (2/3)



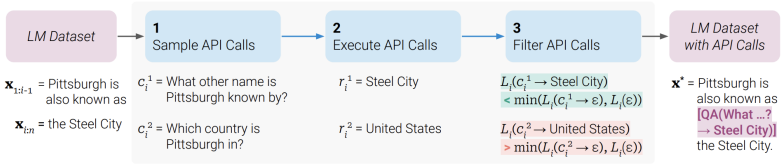
Execute API Calls:

1. Execute API calls to obtain results, *i.e.*,

$$\text{Convert} \quad \texttt{<API>}\ a_c(i_c)\ \texttt{</API>} \quad \text{into} \quad \texttt{<API>}\ a_c(i_c) \rightarrow r_c\ \texttt{</API>}$$

   - Some invalid API calls may be filtered.

Data Collection Pipeline for SFT (3/3)



Filter API Calls:

1. Choose API calls that are useful for the next token prediction, *i.e.*, satisfying the following:

$$\underbrace{L_i(e(c_i, r_i))}_{\text{“Call API”}} - \underbrace{\min(L_i(\emptyset), L_i(e(c_i, \emptyset)))}_{\text{“don't call API”}} \geq \tau_f$$

   - $e(c, r) := \texttt{<API>}\ a_c(i_c) \to r\ \texttt{</API>}$
   - $L_i(\mathbf{z}) := -\sum_{j=i}^{n} \log p_M(x_j \mid \mathbf{z}, x_{1:j-1})$: negative log-likelihood

2. Now, we have a SFT dataset

Inference for Toolformer

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.

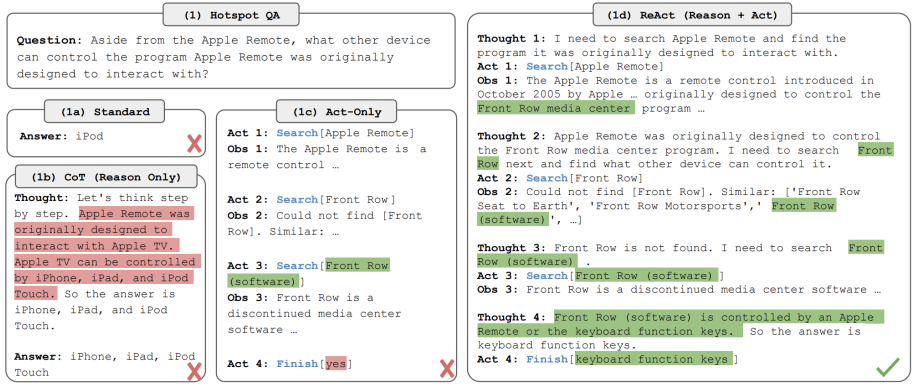
Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

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.

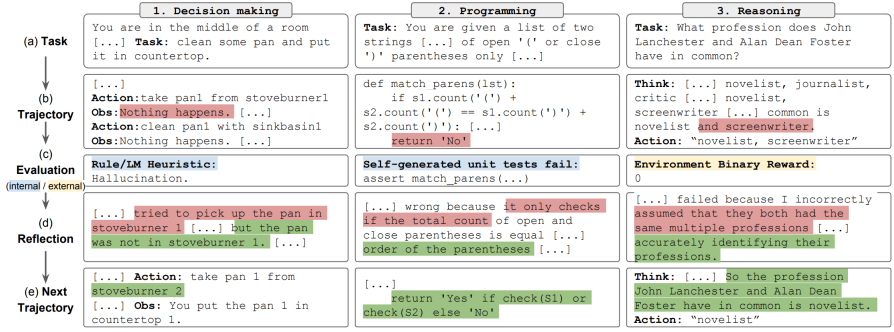
1. Decoding stops if it meets a special token $\rightarrow$.
2. API call is executed to obtain its return.
3. Add a return in the context and restart decoding.

ReAct [Yao et al., 2022]



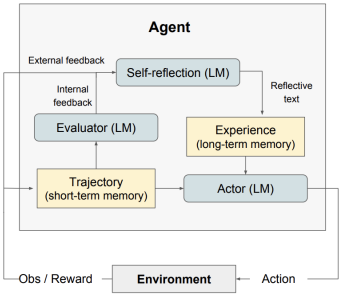
- Synergizing reasoning and acting (=tool-calling) with PaLM-540B
  - few-shot in-context learning
  - action = thought + tool-calling
  - three action types: (1) **search**[entity], (2) **lookup**[string], (3) **finish**[answer]
- LLM can do reasoning (✓), tool-calling (✓), self-verification (✗), remembering long context (✗)

Reflexion [Shinn et al., 2023]



- Reflexion ≈ In-context reinforcement learning
  - Provide verbal feedback (called Reflexion) to guide actions
  - Use GPT-3 – for in-context learning
- LLM can do reasoning (✗), tool-calling (✗), self-verification (✓), remembering long context (✗)

RL via Reflexion



Algorithm 1 Reinforcement via self-reflection

Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i), \theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
  Generate $\tau_t = [a_0, o_0, \dots a_i, o_i]$ using π_θ
  Evaluate τ_t using M_e
  Generate self-reflection sr_t using M_{sr}
  Append sr_t to mem
  Increment t
end while
return

- actor (π_θ): LLMs, CoT, ReAct – generate an action
- Evaluator (M_e): evaluate a trajectory (reward functions *e.g.*, LLM-as-Judge, generated unit tests)
- "reflective text": informative textual reward ($\approx$ return)
- persistent memory $\approx$ a value function
- Any uncomfortable part?

Harness Design (by Anthropic)

- This is a tech report on keys to design long-running coding agents.
- What is harness?
- Trend: LLMs $\rightarrow$ Agentic AI $\rightarrow$ Runtime system design
- Agentic AI "need to do" reasoning (✓), tool-calling (✓), self-verification (✓), remembering long context (✓)

Failure Modes in Designing Long-running Agents

1. "Context Anxiety"
   - due to the limited context size, an agent are trying to wrap up tasks.
   - previously: a single agent $\rightarrow$ still limited context
   - improvement: context reset + "state summary" $\rightarrow$ making harness on "state summary" that works well across reset

2. "Self-evaluation Bias"
   - Confidently praising the work – *e.g.*, "That's an insightful question! ..."
   - Separation between a Generator and an Evaluator – inspired by GAN

Anthropic's Solution in Designing Long-running Agents

Agentic AI = Planner + Generator + Evaluator

1. Planner:
   - From a simple prompt (a task description), generate specification and task decomposition
   - $\approx$ reasoning
   - harness: *e.g.*, task decomposition and task ordering
2. Generator:
   - Conduct a task (*e.g.*, generate code)
   - harness: *e.g.*, tool calling (or providing a set of feasible tools)
3. Evaluator:
   - Real world testing (*e.g.*, use MCP to click a web page for UI feature testing)
   - harness: *e.g.*, interaction with a web browser, unit tests for code, building code

what is harness?

harness = LLM or LLM-based SW

Recap

- Learning in Agentic AI: mostly in-context learning
- Design paradigm: LLM $\rightarrow$ CoT $\rightarrow$ Toolformer $\rightarrow$ ReAct $\rightarrow$ Reflexion $\rightarrow$ Harness design
- What could be the attack surface of agentic AI?

Reference I

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36: 8634–8652, 2023.

J. Von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Reference II

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.