Self-Generated In-Context Examples Improve LLM Agents For Sequential Decision-Making Tasks

Vishnu Sarukkai, Zhiqiang Xie, Kayvon Fatahalian   Stanford University

Self-Improvement Algorithm

We present a **RAG-based self-improving LLM Agent** that remembers its attempts at prior tasks, and intelligently retrieves the **most relevant experiences** from its **continually growing memory**

Algorithm 1 ReAct-style Agent Loop

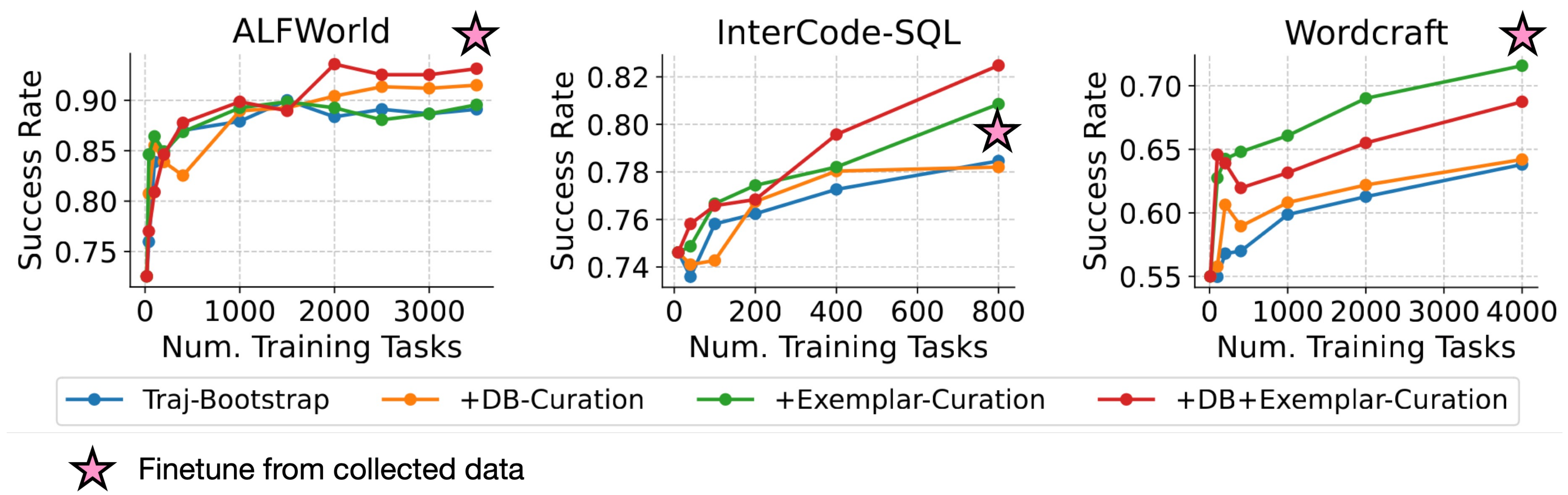
function AGENT($g, \mathcal{D}, \mathcal{E}, T$)
- $C_p \leftarrow \texttt{Retrieve}(\mathcal{D}, keys = [g])$ ▷ Retrieve for plan
- $p \leftarrow \texttt{LLM}_{\text{plan}}(g, C_p)$ ▷ Generate initial plan
- Initialize $\tau \leftarrow (g, p, \{\}, -)$
- $o_1 \leftarrow \mathcal{E}.\texttt{obs}()$
- $C_1 \leftarrow \texttt{Retrieve}(\mathcal{D}, keys = [g, p, o_1])$ ▷ Retrieve for current observation
- **for** $t = 1$ to T **do**
  - $r_t \leftarrow \texttt{LLM}_{\text{reason}}(\tau, o_t, C_t)$ ▷ Generate reasoning
  - $C_{t+1} \leftarrow \texttt{Retrieve}(\mathcal{D}, keys = [g, p, r_t])$ ▷ Retrieve for current reasoning
  - $a_t \leftarrow \texttt{LLM}_{\text{act}}(\tau, o_t, r_t, C_{t+1})$ ▷ Decide action
  - $o_{t+1}, \text{done}, s \leftarrow \mathcal{E}.\texttt{step}(a_t)$ ▷ Execute action in environment
  - $\tau \leftarrow \tau \cup (o_t, r_t, a_t)$
  - **if** done **then**
    - **return** $(g, p, \{(o_i, r_i, a_i)\}_{i=1}^{t}, s)$
- **return** $(g, p, \{(o_i, r_i, a_i)\}_{i=1}^{T}, 0)$ ▷ Failed due to timeout

Agent Performance Scales with Tasks Attempted

Naively adding solved tasks to an ever-growing task database scales performance: using Traj-Bootstrap, task accuracy improves from 73% to 89% on ALFWorld, 74% to 78% on IC-SQL, and 55% to 64% on Wordcraft

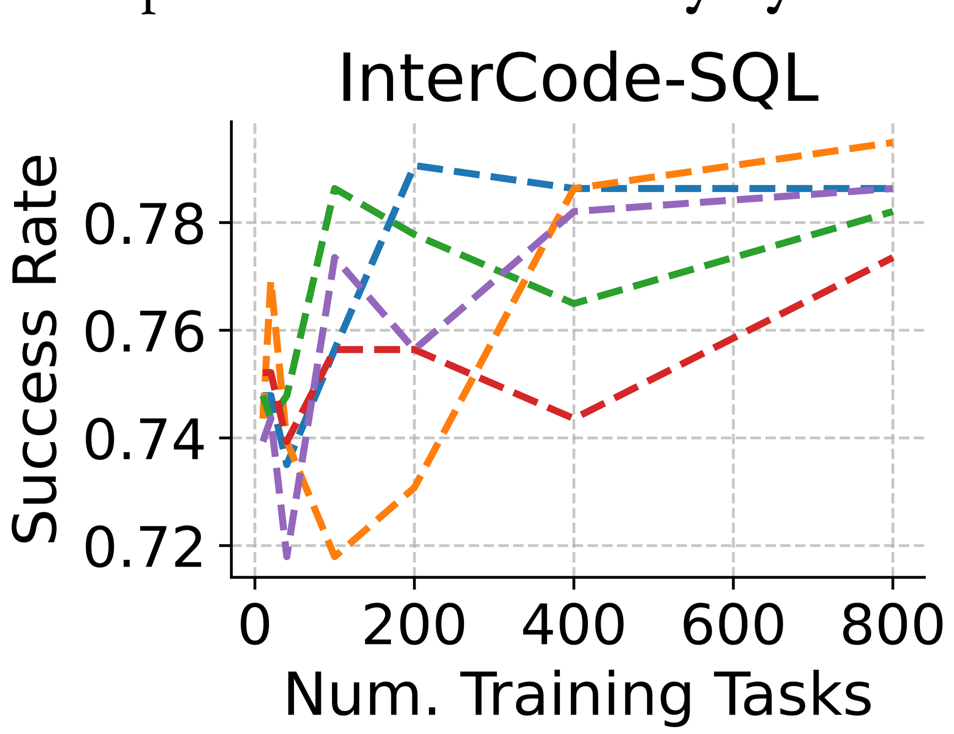
Task quantity and quality both matter: adding on both our data-curation strategies boosts performance further—from 89% to 93% on ALFWorld, from 78% to 82% on IC-SQL, from 64% to 68% on Wordcraft.

Self-generated data can be used for model finetuning: after finetuning gpt-4o-mini on the data collected by +DB+Exemplar-Cur, we obtain accuracy similar to our in-context agent



Self-Collected Data Varies in Quality

Similar to gradient-based RL, continual learning via self-collected in-context examples **varies in accuracy by trial**



Data Quality AND Quantity Matter

We optimize for data quality along two axes:

1. **DB-Curation**: like Population-Based Training (PBT), **run agents in parallel and drop the "worst" agents**
2. **Exemplar-Curation**: filter the best examples in the DB via a **metric of in-context effectiveness**:

$$Q(\tau) = \frac{\sum_{i \in \mathcal{R}(\tau)} o_i \cdot f_i(\tau)}{\sum_{i \in \mathcal{R}(\tau)} f_i(\tau)}$$

Comparison to Other Approaches

Scaling Approach	Performance Boost (δ)
Traj-BS+DB-Cur +Ex-Cur	20
Test-time scaling (pass@4)	21
Model Improvement (4o-mini to 4o)	15
Task-Specific Engineering	18

On ALFWorld, running our algorithm with GPT-4o-mini—rather than upgrading to GPT-4o—would **save over $500,000 across one million tasks** while **also delivering better task accuracy**.

Read the paper for details on...

- **Prompting philosophy:** we always retrieve k in-context examples, and the only content changed in the prompts are the in-context examples. **No task-specific prompting, no context bloat!**
- **Multi-key retrieval:** how we set up a retrieval mechanism accounting for cosine similarity across **several different retrieval keys**
- **Downstream use cases of the data:** we can use our collected databases to help **predict task success**—valuable signal for model routing
- **Experiments with open models:** our algorithm works with models from OpenAI, Anthropic, Mistral, etc.

Paper

Blog

Code