Positive Experience Reflection for Agents in Interactive Text Environments

Philip Lippmann and **Matthijs T.J. Spaan** and **Jie Yang**
Delft University of Technology
{p.lippmann, m.t.j.spaan, j.yang-3}@tudelft.nl

Abstract

Intelligent agents designed for interactive environments face significant challenges in text-based games, a domain that demands complex reasoning and adaptability. While agents based on large language models (LLMs) using self-reflection have shown promise, they struggle when initially successful and exhibit reduced effectiveness when using smaller LLMs. We introduce Sweet&Sour, a novel approach that addresses these limitations in existing reflection methods by incorporating positive experiences and managed memory to enrich the context available to the agent at decision time. Our comprehensive analysis spans both closed- and open-source LLMs and demonstrates the effectiveness of Sweet&Sour in improving agent performance, particularly in scenarios where previous approaches fall short.

1 Introduction

Intelligent agents, designed to interact with and make decisions in dynamic environments, have become a central focus in AI research, with text-based games (TBGs) emerging as a particularly challenging domain for evaluating these agents' reasoning, adaptability, and learning abilities (Côté et al., 2018; Wang et al., 2022). Originally popular in the 1970s as text adventure games,[1] TBGs present players with textual descriptions of environments, requiring them to input natural language commands to achieve objectives (Hausknecht et al., 2020). For instance, determining if a metal fork is conductive involves locating the fork, assembling a circuit, and analyzing the result. Navigating TBGs demands that agents exhibit a combination of abilities, including planning, memory retention, spatial reasoning, and common sense knowledge (Wang et al., 2023).

Previously, deep reinforcement learning and behavior cloning were the primary approaches to develop agents to play TBGs (Ammanabrolu and Riedl, 2019; Yao et al., 2020). However, recent research shows that agents based on pretrained large language models (LLMs) are more effective at navigating TBGs (Lin et al., 2023). A key factor in their success is the integration of internal *reflection* to improve planning (Xi et al., 2023; Huang et al., 2024b; Hu et al., 2024).

Self-reflection, closely related to self-refinement, is a form of reasoning that occurs after receiving binary or scalar feedback from the environment (Madaan et al., 2023). In this process, the LLM reviews its actions and their outcomes, considering what went wrong and potential ways to improve (Wang et al., 2024). By iteratively adjusting its strategy based on verbal reinforcement, conveyed through textual feedback, the agent refines its planning for subsequent attempts (Shinn et al., 2023). However, reflection also has several limitations, including 1) underwhelming performance when agents are correct initially (Li et al., 2024), 2) significantly worse efficacy when using smaller LLMs (Lin et al., 2023), and 3) dependence on external feedback (Zhang et al., 2024b).

Our contributions. In this work we conduct a comprehensive analysis of LLM-based agents employing reflection approaches in TBGs and evaluate their performance across various LLMs. To address the limitations of reflection when agents are initially successful and the diminished efficacy of smaller LLMs, we propose *Sweet&Sour* (S&S) to leverage positive experiences to create a richer context for self-reflection. We supplement this by proposing a managed memory approach to build context across multiple rollouts. Our findings demonstrate that our method improves the performance of agents using reflection, particularly in scenarios where they previously struggled, enabling more robust and generalizable learning.

[1] Try it yourself: https://www.microsoft.com/en-us/research/project/textworld/try-it/

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 131–142
July 31, 2025 ©2025 Association for Computational Linguistics

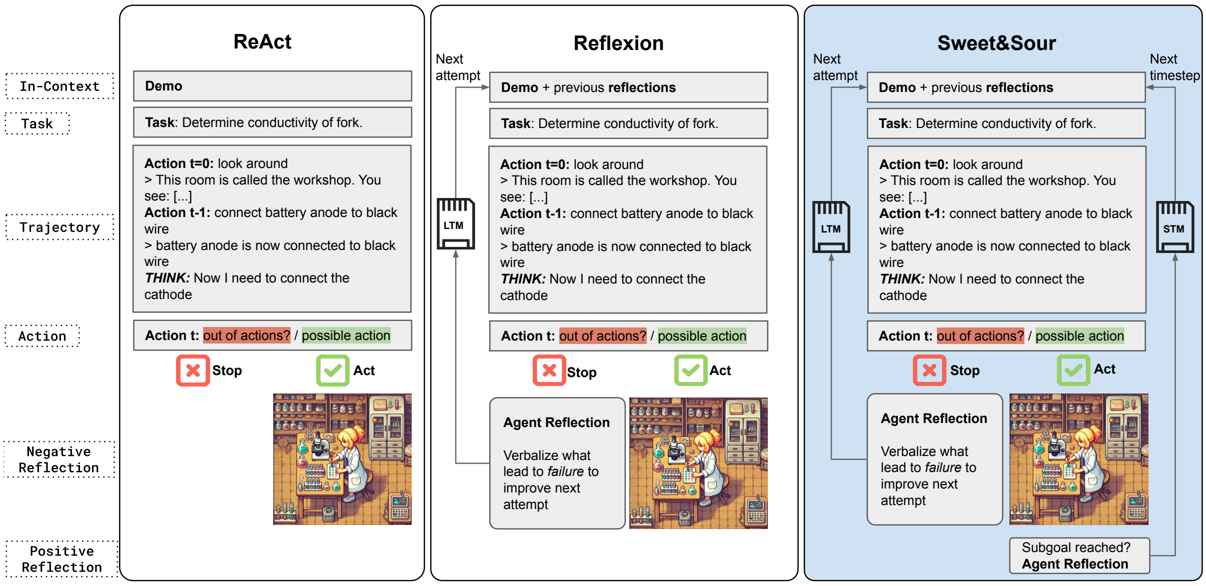

Figure 1: Comparison of used methods to play ScienceWorld. ReAct introduces a `THINK` action to explicitly reason regarding the next step, which does not persist across attempts. Reflexion leverages self-reflection across attempts to learn from unsuccessful tries and stores these in long-term memory (LTM). Sweet&Sour not only performs self-reflection after failures but also after each completed subgoal, making its reflection instantly available at the next timestep as part of short-term memory (STM), and thus incorporating positive experiences.

2 Methodology

2.1 LLMs Playing Text-Based Games

Assuming an LLM behaving as an actor model as part of our agent – *i.e.* generating actions based on the current state and policy, analogous to traditional policy-based RL setups – we sample an action a_t from the current policy π_θ at time t and receive an observation from the environment o_t. Each task consists of a number of sub tasks (such as finding a key object), the completion of which grants the agent a sparse reward, which adds to its current reward r_t. The game continues until the agent has achieved the goal d and receives the full reward as final score, or the maximum number of steps – which we set to 150 – is reached, in which case r_t will become the final score. A detailed problem formulation is given in appendix A.

2.2 Self-Reflection

Reflection occurs in addition to the acting LLM. Here, the agent reviews a_t and o_t associated with previous unsuccessful attempts to verbalize the reason for failure. This process typically involves maintaining a persistent history of insights gained across attempts, which the LLM uses as additional context for its reflections to improve future decision making for the next attempt (Shinn et al., 2023). However, since other self-reflection methods focus on learning from failures (Renze and Guven, 2024; Zhang et al., 2024a,c; Huang et al., 2024a; Yao et al., 2024), they overlook the importance of reinforcing successful behaviors in a similar way.

2.3 Sweet&Sour

Positive experience reflection. To address the limitations of existing self-reflection methods, we introduce a structured approach to leverage reflections from both positive (*sweet*) and negative (*sour*) outcomes. Unlike Reflexion, which passively accumulates failure-based insights, S&S actively queries the agent for generalizable insights both from failure and success cases, promoting a more balanced context building mechanism. Here, we draw inspiration from RL, where rewards steer the agent towards reinforcing advantageous behaviors and preventing over-reliance on error correction. When the current policy is achieving rewards, we query the agent to extrapolate, encouraging the agent to articulate what made its current policy successful and what can be generalized from this, reinforcing strategies that lead to positive outcomes while still learning from failures. This is visualized in figure 1 and a qualitative example of positive experience reflection, as well as an algorithm outlining our approach, is shown in appendix B. Our method is broadly applicable to agents in environments with feedback using self-reflection, including those that build additional complexity on top of the core reflection loop, such as, for example, grounding (Lin et al., 2023) or gradient learning (Yao et al., 2024).

Managed memory. Previous works store reflections gained from unsuccessful attempts in something akin to a long-term memory and make them available as additional context across attempts (Huang et al., 2024a; Shinn et al., 2023). This implies that the agent only has access to additional context upon failing the task – essentially brute-forcing the problem across rollouts. To complement S&S and address these limitations, we introduce *managed memory*, which employs a dual-buffer approach, separating reflections into short-term and long-term storage. Initially, if subgoals are reached within a run, then the reflection corresponding to this subgoal is stored in a temporary buffer and made available immediately. Each short-term memory consists of a tuple $(\text{reflection}_t, o_t, a_t, r_t)$.Once a task is completed or an attempt ends, all short-term memories from the temporary buffer are moved to long-term memory. Reflections from failed attempts are immediately added to long-term memory to inform future attempts. At each time step, the agent queries both memory systems.

2.4 Data and Environment

We use the ScienceWorld benchmark (Wang et al., 2022), which provides a versatile setting for evaluating agents in science experiment tasks across 10 interconnected locations, such as a greenhouse and a workshop, with over 200 objects and 25 action templates, generating a vast and dynamic search space. We use the test set for our evaluation, providing up to 10 variations of each of the 30 distinct tasks, which have an average optimal decision depth of 50 steps. An example task is shown in appendix C. For details of all tasks and the environment, we refer to (Wang et al., 2022). We measure performance using the success score. Completing a task implies completing every sub task, receiving the full reward, and thus a score of 100. We elect to use ScienceWorld instead of previous interactive text environment benchmarks due to their relative simplicity for current LLM-based agents. Still, we provide additional results on the ALFWorld benchmark (Shridhar et al., 2021) in appendix D.

2.5 Baselines

CALM (Yao et al., 2020), a popular method to play TBGs, is a reranking method that integrates a deep reinforced relevance network (DRRN) (He et al., 2016a) with a causal language model fine-tuned using oracle trajectories and imitation learning. We use ReAct (Yao et al., 2023) as our baseline LLM-based agent. ReAct composes useful information at each time step by reasoning over the current context (*e.g.* decomposing task or common sense knowledge query) and carries it over to the context of the following time step as few-shot in-context learning. We further compare our method against Reflexion (Shinn et al., 2023), an agent built on ReAct that employs a self-reflection mechanism to iteratively improve its performance across rounds upon encountering failure based on feedback from the environment. As such, it runs for up to four rounds as it builds up its long-term memory. For all agents, we evaluate their performance using LLMs of different sizes. In descending order of parameter count, we select GPT-4o (`gpt-4o-2024-08-06`), Mistral Large 2 (`mistral-large-2407`), and Llama 3.1 8B (`llama-3.1-8b-instruct`), accessing each via its respective API.

3 Results and Discussion

The results for ScienceWorld are shown in table 1.

Sweet&Sour outperforms baselines. We find that S&S outperforms other methods across all LLMs, setting the highest average score at 54.6 using GPT-4o. The performance gap between S&S and the other methods widens for smaller models with a lower parameter count. For instance, it achieves 44.6 compared to Reflexion's 27.6 on Mistral Large 2, and 32.5 compared to 21.7 on Llama 8B – indicating that our method is more suitable for scenarios with limited computational resources.

Ablation studies demonstrate the complementary nature of positive experience reflection. We conduct ablation studies by selectively removing positive or negative reflections to analyze their individual contributions to performance. When we modify our method to only reflect on failures, performance drops significantly to levels comparable with Reflexion – average scores decrease to 24.6, 31.1, and 44.9 for Llama 8B, Mistral Large 2, and GPT-4o, respectively. When employing only positive reflections, scores remain over the ReAct baseline at 25.8, 32.4, and 42.3, suggesting that while positive reflections alone enhance performance, they are less effective than negative reflections. To further assess contributions of individual components, we forego the use of managed memory and

Task	CALM	ReAct			Reflexion			Sweet&Sour (ours)		
	CALM	L8B	ML2	GPT	L8B	ML2	GPT	L8B	ML2	GPT
1-1 (Boil)	0.0	0.0	0.0	3.8	0.0	0.0	5.1	0.0	7.2	**9.6**
1-2 (Melt)	0.0	8.4	10.3	11.8	0.0	0.0	10.0	11.4	12.1	**12.8**
1-3 (Freeze)	0.0	1.5	0.0	8.1	0.0	2.3	8.3	2.4	3.1	**8.9**
1-4 (Change state)	0.0	1.0	4.7	**10.0**	0.0	0.0	4.2	1.7	2.9	9.2
2-1 (Thermometer)	1.0	5.1	7.8	7.7	3.4	4.2	7.6	7.8	9.7	**10.9**
2-2 (Melting)	1.0	6.7	6.3	5.9	3.3	3.3	26.2	7.9	36.8	**46.0**
2-3 (Melting)	5.0	9.1	11.8	23.4	13.2	14.7	22.6	15.2	29.0	**38.3**
3-1 (Power 1)	7.0	18.8	24.6	57.2	21.2	51.5	78.4	28.6	75.4	**81.1**
3-2 (Power 2)	2.0	10.2	24.7	55.6	9.5	11.9	24.7	23.3	44.5	**58.0**
3-3 (Conductivity 1)	2.0	52.4	51.7	73.0	9.2	25.8	72.1	59.1	69.2	**75.7**
3-4 (Conductivity 2)	10.0	54.2	64.9	**89.7**	35.4	41.6	75.1	62.7	60.3	67.3
4-1 (Find 1)	54.0	17.3	18.7	27.5	44.6	48.1	62.3	41.7	71.7	**74.2**
4-2 (Find 2)	10.0	69.1	71.6	80.3	68.4	75.7	87.3	76.8	**100.0**	**100.0**
4-3 (Find 3)	8.0	21.3	42.8	**47.7**	18.4	16.5	17.3	20.9	21.5	34.3
4-4 (Find 4)	2.0	15.7	15.2	19.3	39.6	46.6	**100.0**	55.1	87.8	**100.0**
5-1 (Grow plant)	4.0	10.8	10.8	10.0	7.2	7.2	7.9	14.2	14.6	**17.4**
5-2 (Grow fruit)	3.0	18.1	18.5	19.2	30.8	51.4	34.6	51.5	55.6	**60.2**
6-1 (Chemistry 1)	6.0	37.8	42.9	58.6	27.1	29.7	**70.2**	37.9	61.1	**70.2**
6-2 (Chemistry 2)	3.0	25.0	27.1	50.6	14.4	28.0	69.8	27.2	51.9	**83.1**
6-3 (Chemistry 3)	6.0	14.4	17.5	39.7	38.9	31.1	16.7	45.3	53.7	**61.5**
7-1 (Lifespan 1)	10.0	37.0	41.7	60.0	75.0	75.0	**100.0**	75.0	88.2	**100.0**
7-2 (Lifespan 2)	4.0	50.5	50.7	67.5	60.0	71.9	**81.4**	70.5	77.0	80.0
7-3 (Lifespan 3)	4.0	33.7	38.2	50.0	29.5	33.7	75.0	51.1	54.2	**84.6**
8-1 (Identify life 1)	0.0	5.1	18.9	**25.3**	1.7	1.7	3.4	11.1	10.3	14.2
8-2 (Identify life 2)	0.0	6.4	7.4	**8.0**	7.4	**8.0**	**8.0**	5.0	7.4	7.4
9-1 (Measure angle)	0.0	28.5	33.0	42.5	56.9	55.1	57.1	68.4	70.3	**75.0**
9-2 (Friction 1)	3.0	14.5	22.6	43.1	23.4	29.3	**100.0**	33.3	36.7	62.0
9-3 (Friction 2)	2.0	2.9	14.5	42.8	1.3	33.6	59.6	7.2	51.9	**63.1**
10-1 (Genetics 1)	2.0	25.7	27.3	26.4	5.6	9.8	50.4	38.9	48.6	**78.8**
10-2 (Genetics 2)	2.0	13.2	19.1	17.2	6.2	21.5	22.7	23.6	24.0	**54.8**
Average	**5.1**	**20.5**	**24.8**	**36.0**	**21.7**	**27.6**	**45.3**	**32.5**	**44.6**	**54.6**

Table 1: Results on the ScienceWorld benchmark. For each method, we use Llama 8B (L8B), Mistral Large 2 (ML2), and GPT-4o (GPT). Each value is an average of across all task variations.

instead use a simplified long-term memory similar to previous approaches. This modification results in performance decreases to 28.2, 38.5, and 48.7 – still exceeding Reflexion but averaging 12.4% below S&S. These results demonstrate that negative reflections, positive reflections, and managed memory provide benefits, with their combination yielding superior performance compared to any individual component.

Sweet&Sour improves robustness to tilt. In highly challenging tasks, such as 1-1 and 8-2, all methods tend to struggle, while in simpler tasks, methods succeed based on the LLM's inherent capabilities. However, medium-difficulty tasks, such as 6-3 and 10-2, reveal a critical performance gap between our method and previous approaches. For instance, on task 10-2, we could observe Reflexion getting off to a strong start, achieving multiple subgoals, though it appeared to get stuck upon encountering its first error. We hypothesize this occurs because methods reflecting only on failures lack context from early successes, leading to performance decline when facing first encountering failure (also known as "tilt"). By contrast, S&S's balanced reflection approach provides richer context, enabling it to build upon initial successes.

4 Conclusion

We introduced Sweet&Sour, a novel approach that enhances LLM-based agents' performance in text-based games through balanced reflection on both successes and failures. Our evaluation demonstrates significant improvements over existing methods across different model sizes, particularly for smaller LLMs. These findings establish the value of positive experience reflection and managed memory in developing more robust and adaptable agents for complex interactive environments.

Limitations and Broader Impacts

Limitations. Despite promising results, our work has limitations. LLMs do not provide guarantees regarding their reasoning capabilities, and their reliance on textual reflection may introduce biases or inconsistencies in agent behavior. Further studies are needed to assess how such agents generalize to real-world decision-making scenarios. Additionally, our evaluation is conducted using a single environment, which, while comprehensive, does not cover all types of interactive scenarios. We leave the exploration of additional environments, such as embodied environments, to future work.

Broader impacts. Large language models (LLMs) carry the risk of misuse and the development of harmful applications (Weidinger et al., 2021). Since our work enhances LLM performance, it could also be leveraged for negative purposes. Given that our research focuses on LLMs functioning as agents, it shares the general risks associated with LLM agents—namely, the potential for errors and unintended negative consequences for users. However, we believe these risks are mitigated by our reliance on simulated benchmarks and by the fact that our work improves agent accuracy, reducing the likelihood of harmful outcomes.

References

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016a. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016b. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Tekin, Gaowen Liu, Ramana Kompella, and Ling Liu. 2024. A survey on large language model-based game agents. *Preprint*, arXiv:2404.02039.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2024a. Recommender ai agent: Integrating large language models for interactive recommendations. *Preprint*, arXiv:2308.16505.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024b. Understanding the planning of llm agents: A survey. *Preprint*, arXiv:2402.02716.

Yanhong Li, Chenghao Yang, and Allyson Ettinger. 2024. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. *Preprint*, arXiv:2404.09129.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Prithviraj Ammanabrolu, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *ArXiv preprint*, abs/2305.17390.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *Preprint*, arXiv:2405.06682.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Matthijs TJ Spaan. 2012. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. *Preprint*, arXiv:1812.00855.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, Qingqing Zhu, Zhenzhu Yang, Adam Nik, Qi Liu, Chenghua Lin, Shi Wang, Ruibo Liu, Wenhu Chen, Ke Xu, Dayiheng Liu, Yike Guo, and Jie Fu. 2023. Interactive natural language processing. *Preprint*, arXiv:2305.13246.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2024. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *Preprint*, arXiv:2302.01560.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models. *Preprint*, arXiv:2112.04359.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R N, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil L Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. 2024a. Proagent: Building proactive cooperative agents with large language models. *Preprint*, arXiv:2308.11339.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024b. Self-contrast: Better reflection through inconsistent solving perspectives. *Preprint*, arXiv:2401.02009.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024c. Agent-pro: Learning to evolve via policy-level reflection and optimization. *Preprint*, arXiv:2402.17574.

A Background

One may consider every TBG to be a partially observable Markov decision process (POMDP) (Spaan, 2012) where the environment state is never observed directly. This POMDP may be formalized as $\langle S, T, A, \Omega, R, \gamma \rangle$, where $\gamma \in [0,1]$ denotes the reward discount factor. S denotes the set of states s that contain the internal information of the game – such as objects found throughout the game or the player's location – not all of which may be visible to the agent at any given time. A denotes the action space made up of individual text actions a issued by the player. Ω denotes the observation function. Further, $o \in O$ denotes the observations made by the player. The observation o_t of the agent at time t depends on the current state s_t, as well as the previous action a_{t-1}, which may be formalized as $\Omega\left(o_t \mid s_t, a_{t-1}\right)$. Seeing as the agent can only observe and interact with the environment of a TBG via natural language, each observation is composed of a sequence of tokens $o_t = \left\{\hat{o}_t^1, \cdots, \hat{o}_t^N\right\}$, as are their actions $a_t = \left\{\hat{a}_t^1, \cdots, \hat{a}_t^M\right\}$.

In the context of TBGs, an action a_t is considered admissible at a state s_t if it is capable of changing the game's state, *i.e.*, if it can lead to a transition to a new state s_{t+1} that is different from the current state s_t. The environment's state transition is modeled through a probabilistic function $T\left(s_{t+1} \mid s_t, a_t\right)$. Traditionally, admissible actions in state s_t could deterministically lead to a new state s_{t+1}. However, we use a more general approach where all actions, whether admissible or not, are included in the state transition function. Non-admissible actions, which do not lead to a change in the game's state, result in a transition back to the original state s_t with probability 1. In contrast, admissible actions lead to different states with their own probability. The admissible action set is bound to be significant for quests with a sufficiently large branching factor. While admissible, many action candidates are bound to be suboptimal.

The reward r received by the agent – the discounted sum of which, $\mathbb{E}\left[\sum_t \gamma^t r_t\right]$, it aims to maximize – are obtained by $r_t = R\left(s_t, a_t\right)$. In practice, TBGs typically provide sparse positive rewards for completing subgoals and advancing towards completing the game.

When a game begins, the agent makes its first observation o_0 at time step $t = 0$. This first observation differs from subsequent ones, as it consists of the goal description d, as well as an analysis of the starting room (*i.e.* the output of the "look around" command). Subsequently, the agent can perform a regular action a_t at each time step and receives a corresponding observation o_t from the environment. In the ScienceWorld environment, similarly to many other TBGs, the agent has an inventory i_t in which to store items.

A TBG's interpreter can accept any text sequence, but will only recognize text that follows a certain structure. Typically, commands take the form of either a single keyword (such as "look") or a combination involving verbs, objects, and occasionally prepositions. Previous works on TBGs made the assumption that we have access to a predefined set of all admissible actions at each game state and must select the correct action to progress (Narasimhan et al., 2015; He et al., 2016b). This is consistent with how some text adventure games are played in the real world (Tao et al., 2018). In line with more recent works (Lin et al., 2023), we make the assumption that we have access to a number of action templates (*e.g.* `connect A to B`, `pick up C`) and subsequently ask the agent to generate the actions as tokens using these templates. This leads to a more challenging action generation process for the agent.

B LLM Reflection

B.1 Reflection Examples

After successfully completing a sub task, the agent is prompted to reflect on its most recent observations and identify the key factors that contributed to its success. This reflection process encourages the agent to verbalize the steps and strategies that led to the desired outcome, helping it create a plan that can be stored in managed memory for future use. In the case that a task is failed, the agent is instructed to reflect on alternative actions it could have taken and to devise a revised plan for the next attempt, ensuring continuous learning and improvement. This plan is stored in long-term memory.

An example trajectory of a ScienceWorld task where the agent reflects on positive and negative experiences is shown in Figure 2. At the end of this trajectory, we show for a single command what a reflection would look like for a successful or unsuccessful choice, in this case picking up an animal. For the unsuccessful case (red), where the agent does not pick up an animal, we reflect on whether another choice would have constituted an animal and thus resulted in a reward. For the successful case where a subgoal is reached (yellow), we reflect on what made the current actions successful and subsequently commit this to memory.

B.2 Detailed Breakdown of Sweet&Sour

The Sweet&Sour self-reflection process is outlined in Algorithm 1. Sweet&Sour combines reflections from positive and negative experiences with a managed memory system to improve agent performance through structured learning from both successes and failures.

Initialization and prerequisites. The algorithm requires an initial observation o_0 that contains the goal description d and the initial environment state. We set a maximum episode length $T = 150$ steps to ensure termination. The system maintains two distinct memory buffers: short-term memory (M_{ST}) storing reflections from the current episode, and long-term memory (M_{LT}) maintaining persistent reflections across episodes.

Episode structure. Each episode consists of a sequence of interactions with the environment. The agent begins by retrieving relevant reflections from both memory systems to inform its current decision. It then samples an action a_t from its policy π_θ, which is conditioned on the current observation o_t and the relevant memories from both M_{ST} and M_{LT}, if available. After executing the action, the agent receives a new observation o_{t+1} and reward r_t.

Reflection generation. The algorithm generates reflections through two distinct mechanisms. Positive reflections (R_t^+) are generated when subgoals are achieved during an episode. These reflections capture successful strategies and are initially stored in short-term memory as tuples (R_t^+, o_t, a_t, r_t), allowing the agent to build upon successful approaches within the current episode. We query the agent to extrapolate, encouraging the agent to articulate what made its current policy successful and what can be generalized from this, reinforcing strategies that lead to positive outcomes while still learning from failures. Conversely, negative reflections (R_t^-) are generated at the end of unsuccessful episodes. These reflections analyze failure cases and include a proposal for a different approach to be pursued during the following episode. They are directly stored in long-term memory, enabling the agent to learn from and avoid unsuccessful strategies in future attempts.

Managed memory. The algorithm's memory management system employs a structured approach through its dual-buffer design. Short-term memory accumulates positive reflections during successful progression and maintains immediate context for the current episode. This memory is reset at the end of each episode, ensuring a fresh context for new attempts. The long-term memory serves as a persistent knowledge base, receiving all short-term memories after successful episodes and directly storing negative reflections from failed episodes. This separation between immediate and persistent knowledge enables the agent to maintain both contextual awareness and learned experience across multiple episodes.

C Example ScienceWorld Task

In this section, we provide a successfully completed task, a variation of Task 1-1, which concerns itself with boiling a substance, in this case water, to change its state. Once the agent has viewed its surroundings and moved to the kitchen, it collects the necessary items and begins its experiment by boiling the water in a pot on the stove. Finally, the agent examines steam and completes the task (highlighted in green). The example is truncated to improve readability. The trajectory is shown in Figure 3.

D Additional Results on ALFWorld

To further demonstrate the generalizability and robustness of S&S, we conduct additional experiments on the ALFWorld benchmark (Shridhar et al., 2021). ALFWorld comprises of various interactive embodied tasks set in common home environments. It includes 6 task types where an agent must achieve high-level goals by navigating and interacting via text actions within a simulated household. Tasks can span over 50 locations and require more than 50 steps to solve. We follow the experimental protocol used by ReAct (Yao et al., 2023), i.e. we test on the same 134 unseen evaluation games. Additionally, we report success rates as the evaluation metric. Table 2 summarizes our results. S&S consistently outperforms baselines across different model sizes, demonstrating its effectiveness in complex interactive scenarios.

Method / Model	Llama 8B	Mistral Large 2	GPT-4o
ReAct	8.0	41.0	77.0
Reflexion	13.0	61.0	91.0
Sweet&Sour (ours)	**20.0**	**78.0**	**97.0**

Table 2: Performance on ALFWorld (success rates).

These results affirm that leveraging positive experience reflections significantly enhances performance, particularly in challenging interactive environments. Still, we find that the ALFWorld tasks remain significantly easier than those of ScienceWorld discussed in section 3, with our approach achieving a near perfect score. Future work may explore extending S&S to additional tasks and environments.

Algorithm 1 Sweet&Sour Reflection Process with Short- And Long-Term Memory

Require: Initial observation o_0 containing goal description d
Require: Maximum steps $T = 150$
Initialize short-term memory $M_{ST} \leftarrow \emptyset$
Initialize long-term memory $M_{LT} \leftarrow \emptyset$
Initialize current reward $r \leftarrow 0$
for each episode **do**
  $t \leftarrow 0$
  while $t < T$ and goal not achieved **do**
    Retrieve reflections from M_{ST} and M_{LT}
    Sample action $a_t \sim \pi_\theta(a_t | o_t, M_{ST}, M_{LT})$
    Execute a_t, observe o_{t+1}, potentially receive reward r_t
    $r \leftarrow r + r_t$
    if subgoal achieved **then**
      Generate positive reflection R_t^+
      $M_{ST} \leftarrow M_{ST} \cup \{(R_t^+, o_t, a_t, r_t)\}$
    end if
    $t \leftarrow t + 1$
  end while
  if goal achieved **then**
    $M_{LT} \leftarrow M_{LT} \cup M_{ST}$ ▷ Transfer successful memories
  else
    Generate negative reflection R_t^-
    $M_{LT} \leftarrow M_{LT} \cup \{(R_t^-, o_t, a_t, r_t)\}$
  end if
  $M_{ST} \leftarrow \emptyset$ ▷ Reset short-term memory
end for

ScienceWorld Task 8-1

Task Description:
Your task is to find a(n) animal. First, focus on the thing. Then, move it to the red box in the kitchen.

`> look around`
This room is called the hallway. In it, you see:
- a substance called air
- a picture
- the agent

You also see:
- A door to the greenhouse (open)
- A door to the living room (open)
- A door to the art studio (open)
- A door to the kitchen (open)
- A door to the bedroom (open)
- A door to the workshop (open)

`> open door to greenhouse`
The door is already open.
`> go to greenhouse`
You move to the greenhouse.

`> open door to outside`
The door is already open.
`> go to outside`
You move to the outside.
`> look around`
This outside location is called the outside. Here, you see:

- a substance called air
- an axe
- a substance called wood
- the ground
- a fire pit (containing nothing)
- a fountain (containing a substance called water)
- a butterfly egg
- a blue jay egg
- a dove egg
- the agent

You also see:

- A door to the kitchen (open)
- A door to the foundry (open)
- A door to the greenhouse (open)

Case #1: Failure
`> focus on axe`
You focus on the axe.
`> pick up axe`
You move the axe to the inventory.
Reflection (I failed because I assumed eggs are not animals. My plan was to go outside to find animals. I tried to pick up an axe. An alternative course of action would have been to go outside and pick up an egg, as it might be considered an animal.)
→ Managed Memory

Case #2: Success
`> focus on dove egg`
You focus on the dove egg.
`> pick up dove egg`
You move the dove egg to the inventory.
Reflection (Picking up an egg was the correct choice. Even though the animal was in an egg, it is still considered an animal. My plan was to go outside and find an animal in the wild. Items such as an axe, wood, or landscape elements are not related to animals.)
→ Managed Memory

Figure 2: Example of reflection in the 2-1 Task, where the agent needs to find something.

ScienceWorld Task 1-1

Task Description:
Your task is to boil water. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.

`> look around`
This room is called the hallway. In it, you see:
- a picture
- a substance called air
- the agent
- A door to the green house (open)
- A door to the living room (open)
- A door to the art studio (open)
- A door to the kitchen (open)
- A door to the bedroom (open)
- A door to the workshop (open)

`> open door to kitchen`
The door is already open.
`> go to kitchen`
You move to the kitchen.
`> look around`
This room is called the kitchen. In it, you see:
- a substance called soap
- a painting
- a thermometer reading 10°C
- a counter with various items (banana, potato, red apple, orange)
- a sink (off, empty)
- a table with a glass cup (empty)
- a chair (empty)
- a freezer (closed)
- a lighter
- a stopwatch (deactivated)
- a fridge (closed)
- a cupboard (closed)
- a stove (turned off)
- a glass jar containing sodium chloride
- various other items (air, the agent, etc.)

`> pick up thermometer`
You move the thermometer to the inventory.
`> open cupboard`
The cupboard is now open.
`> pick up metal pot`
You move the metal pot to the inventory.
`> move metal pot to sink`
You move the metal pot to the sink.
`> activate sink`
The sink is now activated.
`> deactivate sink`
The sink is now deactivated.

```
> pick up metal pot
```
You move the metal pot to the inventory.
```
> focus on substance in metal pot
```
You focus on the water.
```
> move metal pot to stove
```
You move the metal pot to the stove.
```
> activate stove
```
The stove is now activated.
```
> use thermometer in inventory on substance in metal pot
```
The thermometer measures a temperature of 13 degrees Celsius.
```
> examine substance in metal pot
```
A substance called water.
```
> use thermometer in inventory on substance in metal pot
```
The thermometer measures a temperature of 28 degrees Celsius.
```
> examine substance in metal pot
```
A substance called water.
```
> use thermometer in inventory on substance in metal pot
```
The thermometer measures a temperature of 58 degrees Celsius.
```
> examine substance in metal pot
```
A substance called water.
```
> use thermometer in inventory on substance in metal pot
```
The thermometer measures a temperature of 78 degrees Celsius.
```
> examine substance in metal pot
```
A substance called water.
```
> use thermometer in inventory on substance in metal pot
```
The thermometer measures a temperature of 98 degrees Celsius.
```
> examine steam
```
A substance called steam. Task completed.

Figure 3: Successful sequence of events in the 1-1 Task, where the agent needs to boil water.