SAC-GLAM: Improving Online RL for LLM agents with Soft Actor-Critic and Hindsight Relabeling

Loris Gaven
Inria (Flowers)
University of Bordeaux, France
loris.gaven@inria.fr

Clément Romac
Inria (Flowers)
University of Bordeaux, France
Hugging Face

Thomas Carta
Inria (Flowers)
University of Bordeaux, France

Sylvain Lamprier
Univ Angers, LERIA,
SFR MATHSTIC, F-49000 Angers, France

Olivier Sigaud
Sorbonne Université, ISIR, Paris, France

Pierre-Yves Oudeyer
Inria (Flowers)
University of Bordeaux, France

Abstract

The past years have seen Large Language Models (LLMs) strive not only as generative models but also as agents solving textual sequential decision-making tasks. When facing complex environments where their zero-shot abilities are insufficient, recent work showed online Reinforcement Learning (RL) could be used for the LLM agent to discover and learn efficient strategies interactively. However, most prior work sticks to on-policy algorithms, which greatly reduces the scope of methods such agents could use for both exploration and exploitation, such as experience replay and hindsight relabeling. Yet, such methods may be key for LLM learning agents, and in particular when designing autonomous intrinsically motivated agents sampling and pursuing their own goals (i.e. autotelic agents). This paper presents and studies an adaptation of Soft Actor-Critic and hindsight relabeling to LLM agents. Our method not only paves the path towards autotelic LLM agents that learn online but can also outperform on-policy methods in more classic multi-goal RL environments.

1 Introduction

In recent years, LLMs have demonstrated impressive capabilities in solving various tasks, from following instructions to complex reasoning and real-world interactions [Ahn et al., 2022, Li et al., 2022, Wang et al., 2023c,b]. Despite these successes, challenges remain in applying LLMs as agents in sequential decision-making tasks, mainly due to misalignment when understanding environmental states and action spaces. In a recent line of work [Carta et al., 2023, Wen et al., 2024a, Putta et al., 2024, Wen et al., 2024b] the use of online RL has been explored to interactively align LLM agents with complex environments. Following these promising results, one could wonder whether such agents could learn in a more autotelic way where goals are not directly provided by the environment.

A natural next step would be giving these LLM agents more autonomy, allowing them to freely explore environments and set their own goals [Colas et al., 2021]. When designing such agents, prior work notably showed how language can be leveraged to imagine new goals and shape the agent's curriculum [Colas et al., 2022, 2020]. Recent studies successfully showed LLMs could generate

Accepted at the Intrinsically Motivated Open-ended Learning workshop at NeurIPS 2024.

goals [Wang et al., 2023a, Zhang et al., 2024] and reward functions [Fan et al., 2022, Du et al., 2023] for an autonomous agent, leveraging their world-modeling capabilities and commonsense knowledge to drive the agent towards interesting goals. While mixing prior work using LLMs as goal generators and LLMs as RL agents may appear straightforward, several challenges remain in obtaining autotelic LLM agents.

Among them, autotelic architectures usually involve experience replay and hindsight relabelling to best utilize all trajectories, even unsuccessful ones, which can be predominant when agents set their own goals. However, most current approaches to fine-tune LLM agents with RL rely on on-policy algorithms that cannot handle experience replay and trajectory relabeling mechanisms. While a handful of attempts Wen et al. [2024a], Putta et al. [2024] to use off-policy RL on LLMs exist, they consider token-level actions (better suited for text generation tasks) and complex architecture balancing token-level actions and environment-level actions (usually sequences of tokens). As a step towards autotelic LLM agents, this work introduces a version of Soft Actor-Critic (SAC) [Haarnoja et al., 2018] explicitly designed for LLMs as RL agents combining the simplicity of prior on-policy methods focusing on environment-level actions (e.g. GLAM [Carta et al., 2023]) and the advantages of off-policy learning. We then extend it with Hindsight Experience Replay (HER) [Andrychowicz et al., 2017] and show our new method (SAC-GLAM) outperforms GLAM (called PPO-GLAM below) in a classic multi-goal environments while showcasing better sample efficiency while staying on par in time efficiency in spite of the stability and efficiency challenges of using an actor-critic approach with a pre-trained LLM actor and a randomly initialized critic.

2 SAC-GLAM

We consider a textual RL setting where, given a language vocabulary $\mathcal{V}$, the environment returns an observation $o \in \mathcal{V}^N$ and a reward $r \in \mathbb{R}$ after an action $a \in \mathcal{A} \subseteq \mathcal{V}^M$ (i.e., actions are sequences of tokens). The task or goal description $g \in \mathcal{G} \subseteq \mathcal{V}^K$ conditions the reward. This environment can be modeled as a goal-augmented Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \mathcal{O}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{G}$ is the goal space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is the goal-conditioned reward function, $\mathcal{O} : \mathcal{S} \to \mathcal{V}^N$ is the observation function that maps a state to a textual description, and γ is the discount factor.

In such settings, PPO shined for its efficiency and simplicity when finetuning stochastic pre-trained policies such as LLM agents [Carta et al., 2023, Wen et al., 2024b]. However, it cannot leverage HER as analyzed in Appendix G. We propose in sections below an off-policy alternative that keeps GLAM's simplicity on how the LLM is used as a stochastic policy over discrete environment-level actions (i.e. sequences of tokens).

2.1 Soft Actor Critic with an LLM actor

We adapt the discrete version of SAC [Christodoulou, 2019] to our stochastic LLM-based policy. We first follow GLAM's approach to obtaining a stochastic policy with an LLM by computing the probability of each action $a_i \in \mathcal{A}$ from our environment as the probability of its token sequence $a_i = \{w_1, w_2, ..., w_{|a_i|}\}$ (with $w_i \in \mathcal{V}$) to follow a prompt containing both o and g:

$$\mathbb{LP}_{LLM}(a_i|o, g) = \sum_{j=0}^{|a_i|} \log \mathbb{P}_{LLM}(w_j|o, g, w_{<j}) \,. \tag{1}$$

Subsequently, a softmax function is applied to generate a probability distribution over all possible actions and obtain our stochastic policy:

$$\pi(a_i|o, g) = \frac{e^{\mathbb{LP}_{LLM}(a_i|o,g)}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{LP}_{LLM}(a_j|o,g)}} \,. \tag{2}$$

Then, we follow SAC and simultaneously learn both this policy and a Q-function, which we implement as an MLP added on top of the final decoder block of our LLM.

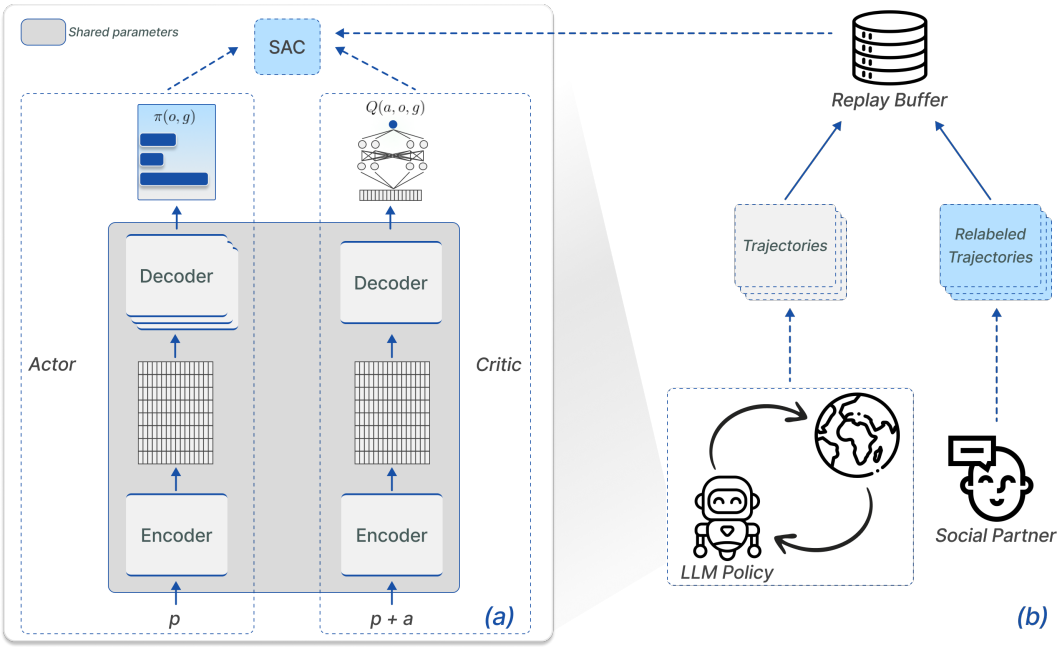

Figure 1: **The SAC-GLAM method.** (a) depicts the agent's architecture when an encoder-decoder LLM is used: the actor computes an action probability as the probability computed by the LLM of the action's tokens to follow the observation and goal concatenated in the prompt p, while the critic computes the Q-value for each action a and the prompt p with an MLP attached to the decoder's last hidden state. (b) illustrates the agent-environment interaction, where trajectories are generated and added to the replay buffer. We used an environment where a social partner relabels these trajectories with hindsight goals.

2.2 Pre-trained policy and randomly initialized critic

When running SAC with a pre-trained LLM as the policy, several unique challenges arise. In SAC, the actor's updates rely on feedback from the critic. While poor estimates from a randomly initialized critic may not significantly impact a randomly initialized policy, our situation is different. We aim to protect the pre-trained LLM actor from harmful updates that could degrade its performance. One potential solution is to introduce a warm-up period where only the critic is trained.

To make this warmup period as short as possible, one must seek for faster critic convergence. First, one could speed up its learning not only backpropagating gradients through the MLP but also through the LLM's weights, while potentially impairing the actor's pre-trained capabilities. Second, while a natural choice would be to have an MLP with $|\mathcal{A}|$ outputs that take as input the representation produced by the LLM for the observation and goal in the prompt, we also investigated leveraging the LLM's capabilities by rather using an MLP with a single output with the LLM encoding the concatenation of the prompt and an action. This requires $|\mathcal{A}|$ forward passes through the LLM and MLP to obtain all the actions' Q-value but may also ease the MLP's task leading to faster convergence.

Finally, the critic's sample efficiency can also be improved after the warmup period. One way to achieve this is through the use of n-step returns with a target expressed as:

$$y_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \mathbb{E}_{a_{t+n} \sim \pi(\cdot|o_{t+n}), g} \left[Q(s_{t+n}, a_{t+n}) - \alpha \log \pi(a_{t+n} \mid o_{t+n}, g) \right] . \quad (3)$$

In the standard SAC algorithm, a single-step return is used, which corresponds to setting $n = 1$. This approach reduces the reliance on bootstrapping by accumulating rewards over multiple steps before bootstrapping, often leading to faster critic convergence. However, n-step returns introduce the risk of older transitions becoming highly off-policy, as intermediate rewards and actions depend on the policy in place when the transitions were collected. To mitigate this issue, we focus on minimizing

the age of the oldest transitions (i.e., the number of policy updates since the transition was recorded). One effective way to achieve this, without altering the update-to-data ratio, is by reducing the size of the replay buffer.

2.3 Hindsight Experience Replay

We augment our SAC-GLAM agent with HER, enabling the algorithm to learn from failed attempts by relabeling portions of the failed trajectories with accidentally reached goals. At the end of an episode, the trajectory with its initial goal and the relabeled sections of the trajectory are added to the replay buffer. When training, we randomly sample a batch transitions from this buffer such that the batch contains 50% of relabeled transitions.

3 Experiments

We evaluated our method in the Playground-text environment, a text-based adaptation (goals, observations, and actions are represented as text) of the original Playground environment [Colas et al., 2020]. We selected this environment because it features a social partner describing goals reached during a trajectory. In this environment, the agent receives a textual goal and must interact with different objects (plants, animals, food and water) to complete the task. It features a sparse reward context where the agent receives 1 if the task is completed and 0 otherwise. In the original Playground environment, the agent moves along the x and y axes to reach objects. In the text-based version, we simplified this by introducing actions like "*Go to {object}*", which directly move the agent to the selected object. The environment features two types of tasks: "*Grasp {object}*" and "*Grow {object}*". Animals can be grown using either water or food, while plants can only be grown using water. We also introduced sequential tasks, where the agent must complete two tasks in the correct order, significantly expanding the task space to $8{,}470$ tasks. A detailed description of the environment is given in Appendix A.

We compared SAC-GLAM with and without HER to PPO-GLAM using Flan-T5-base [Chung et al., 2024] as the pre-trained actor for both approaches. Our results in Figure 2 indicate that for a fixed budget of 400K steps, SAC-GLAM alone outperforms PPO-GLAM in sample efficiency while being slower. When equipped with HER, SAC-GLAM becomes even more sample efficient and achieves comparable performance in terms of training time.

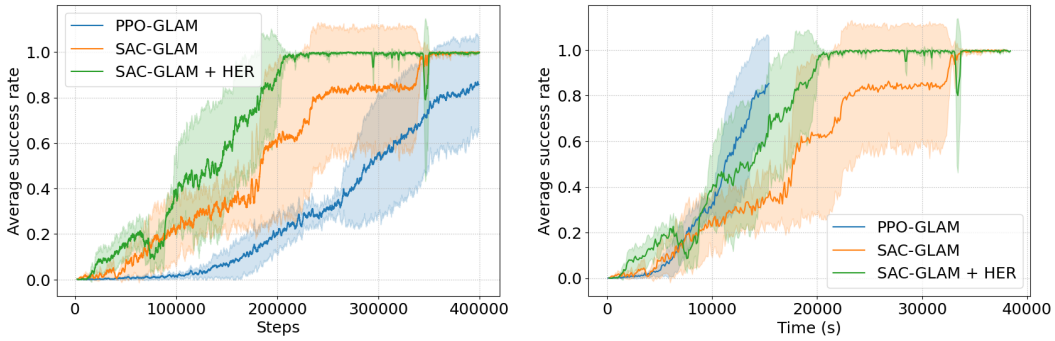

Figure 2: **Performance comparison of SAC-GLAM and PPO-GLAM in the Playground-Text environment.** We show the average success rate as a function of the number of steps (left) and the average success rate over time in seconds (right). The mean and standard deviation are calculated across 4 seeds.

3.1 Ablations

We study in Appendix B different possible architectures for the critic as proposed in Section 2.2. We show that sharing the policy's weights and using an MLP with a single output leads to better sample efficiency and greater stability. We hypothesize that the increased stability is due to the latent representation provided to the MLP, which directly incorporates information about both the prompt and the action.

We also study the impact of the update frequency and number of epochs (Appendix D), replay buffer sampling strategies (Appendix E) and n-step returns (Appendix F) on the critic's convergence speed. We found that doing frequent updates (every 32 steps) with multiple epochs (2), using $n = 3$ and balancing transitions relabeled by HER in batches (50%) was leading to the best performance.

4 Discussion

In this paper, we introduce SAC-GLAM, an off-policy RL approach adapting SAC and HER to LLMs as RL agents. We notably highlight how Actor-Critic methods applied to pre-trained policies such as LLMs introduce challenges in balancing the policy and critic convergence and showed using a warmup period, weights sharing, single Q-value head and n-step returns make SAC-GLAM obtain comparable time efficiency to PPO-GLAM while surpassing it in sample efficiency. Finally, in addition to outperforming PPO-GLAM in a classic multi-goal RL setting, SAC-GLAM also paves the path toward autotelic LLM agents where using HER and off-policy RL is key.

Acknowledgments and Disclosure of Funding

Experiments presented in this paper were carried out using the HPC resources of IDRIS under the allocation 2023-[A0151011996] made by GENCI.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. 2022. URL `http://arxiv.org/abs/2204.01691`.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33:3761–3774, 2020.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. 2021. URL `http://arxiv.org/abs/2012.09830`.

Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Vygotskian autotelic artificial intelligence: Language and culture internalization for human-like AI, 2022. URL `http://arxiv.org/abs/2206.01134`. type: article.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs, 2023. URL http://arxiv.org/abs/2305.14314.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models, 2023. URL http://arxiv.org/abs/2302.06692.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. 35:18343–18362, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018. URL http://arxiv.org/abs/1801.01290.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL http://arxiv.org/abs/2106.09685.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. Pre-trained language models for interactive decision-making. 2022. URL http://arxiv.org/abs/2202.01771.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous AI agents, 2024. URL http://arxiv.org/abs/2408.07199.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL http://arxiv.org/abs/2305.16291.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. JARVIS-1: Open-world multi-task agents with memory-augmented multimodal language models, 2023b. URL http://arxiv.org/abs/2311.05997.

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2023c. URL http://arxiv.org/abs/2302.01560.

Muning Wen, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. Entropy-regularized token-level policy optimization for large language models, 2024a. URL http://arxiv.org/abs/2402.06700.

Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition, 2024b. URL http://arxiv.org/abs/2405.15821.

Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. OMNI: Open-endedness via models of human notions of interestingness, 2024. URL http://arxiv.org/abs/2306.01711.

A Playground-Text

The Playground-Text environment is a text-based environment where goals, observations, and actions are all represented in natural language. The environment consists of a scene with N=6 objects. The agent can move toward these objects using the action "*Go to {object}*", and can interact with them using "*Grasp*" to pick up objects and "*Release*" to put them down. There are four types of objects: furniture, supplies, plants, and animals. The "supplies" category contains two items: food and water. When the agent brings water to a plant, it grows; animals can grow with either water or food. The colors of the objects are irrelevant and serve only as distractions for the agent. Observations from the environment fully describe its state, as shown in Figure 3.

Goal: Grow green mouse then grow blue rose
You see: green water, green mouse, blue rose, red rose, blue lion, red food
You are on: nothing
You are holding: nothing
You have grown: nothing
Action:

Figure 3: **An observation in the Playground-Text environment.** All necessary information is provided in the observation, making the environment fully observable.

There are two types of goals in the environment: "*Grasp {object}*" and "*Grow {object}*." Additionally, we introduce sequential goals such as "*Grow {object A} then grasp {object B}*" or "*Grow {object A} then grow {object B}*." Rewards are set to 0 at each step, with a reward of 1 when achieving the goal. The agent is trained on goals sampled uniformly from all available goals. However, the distribution of goal types is not uniform, with the majority being sequential goals (Figure 4).

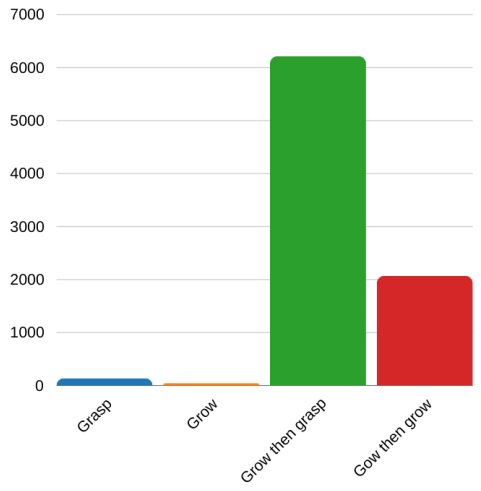

Figure 4: **Proportion of each goal type.** The distribution of goals is highly imbalanced, with sequential goals making up 98% of the total.

B Critic architecture

In our setup, the critic is implemented as an MLP, positioned on top of the final decoder block. We experimented with different methods of integrating the critic with the actor:

1. **Shared parameters with backpropagation:** The critic shares parameters with the actor and allows gradient backpropagation through these shared layers, enabling joint learning of the underlying representations.
2. **Shared parameters without backpropagation:** In this case, the critic shares parameters with the actor, but gradients are backpropagated only through the critic's MLP head, keeping the shared layers frozen.

We also explored two approaches for computing action Q-values:

1. **Observation input:** The critic takes the observation as input and outputs Q-values for each possible action. This method is commonly used in discrete actions value-based algorithms like Mnih et al. [2015].
2. **Observation-action input:** The critic takes the concatenation of the observation and action as input, producing a single Q-value. This approach is more typical in continuous action spaces.

These architecture variations were tested in a simplified version of the environment, which omits sequential goals and uses only $N = 3$ objects (see Figure 5).

As shown in Figure 5, the critic architecture that uses both observation and action as inputs outperforms the one that relies solely on the observation. Additionally, the configuration where gradients are propagated through both the MLP and shared parameters yields better results compared to the version where gradients flow only through the MLP head.

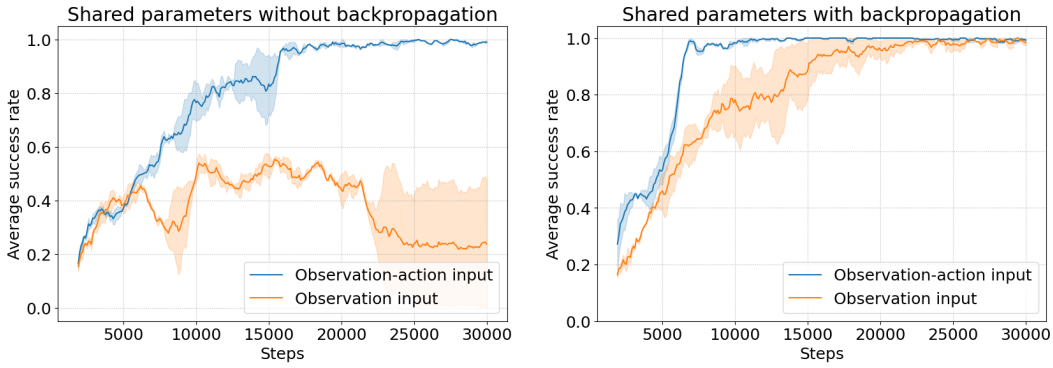

Figure 5: **Comparison of Critic Architectures.** The blue curve represents the architecture where both the observation and action are inputs, producing a single Q-value. The orange curve corresponds to the architecture where only the observation is input, producing Q-values for each action. The left plot illustrates the case where gradients are backpropagated only through the MLP head, while the right plot shows the case where gradients propagate through both the MLP and shared parameters. Mean and standard deviation are computed across two seeds.

C Implementation details

In our approach, the actor is modeled using a pre-trained encoder-decoder LLM, with the critic positioned as an MLP attached to the final decoder block, similar to the GLAM architecture. To optimize GPU VRAM usage during training, we employed LoRA [Hu et al., 2021] and 4-bit quantization techniques as described in [Dettmers et al., 2023].

We implemented SAC with automatic tuning of the entropy coefficient, initializing it at 0.005 and setting a target entropy of 0 for the SAC-GLAM version and initializing it at 0.05 and setting a target

entropy of 0.01 in SAC-GLAM+HER, both using a learning rate of $2\mathrm{e}{-3}$. Both the actor and critic were trained using the Adam optimizer with a learning rate of $1\mathrm{e}{-4}$. Detailed hyperparameters are provided in Appendix H. The critic MLP has two hidden layers, each with 1024 units and ReLU activation functions.

We used a vectorized version of Playground-Text with 32 instances of the environment running (synchronously) in parallel. In order to accelerate the use of an LLM as the trained RL policy (and critic), we leveraged Lamorel[1] as in GLAM and deployed 4 instances of the LLM in parallel (distributing both the forward passes to compute the actions' probability and the training in a Data Parallelism setting). When using Flan-T5 250M, each LLM instance is distributed (Vertical Model Parallelism) over one Nvidia V100 32GB GPUs requiring thus a total of 4 Nvidia V100 32GB GPUs to run an experiment (1 GPU $\times$ 4 LLM instances). In total, each experiment and ablation requires 80 GPU hours per seed on the Nvidia V100 32GB.

D Update frequency and epochs

Update frequency usually plays a key role in Deep RL balancing between sample efficiency and overfitting to data collected. This issue is particularly pronounced in on-policy settings, where updates are based solely on the data gathered between each update. However, when considering very large policy models such as LLMs, properly setting the update frequency also becomes key in off-policy settings as updates may become slower than data collection. Moreover, both PPO and SAC possess a parameter controlling the number of gradient updates performed on collected data, which also affects sample efficiency.

We studied different values for these parameters for both SAC and PPO on a simplified version of the environment. In the case of PPO, we observed that updating the model too frequently (every 1024 steps) leads to the method plateauing at a suboptimal policy. Additionally, performing too many epochs (32) at each update introduces instability in the training process (Figure 6). Concerning SAC, the method benefits from frequent updates with multiple epochs, inducing a balance between sample and time efficiency (frequent updates and multiple epochs imply slower training).

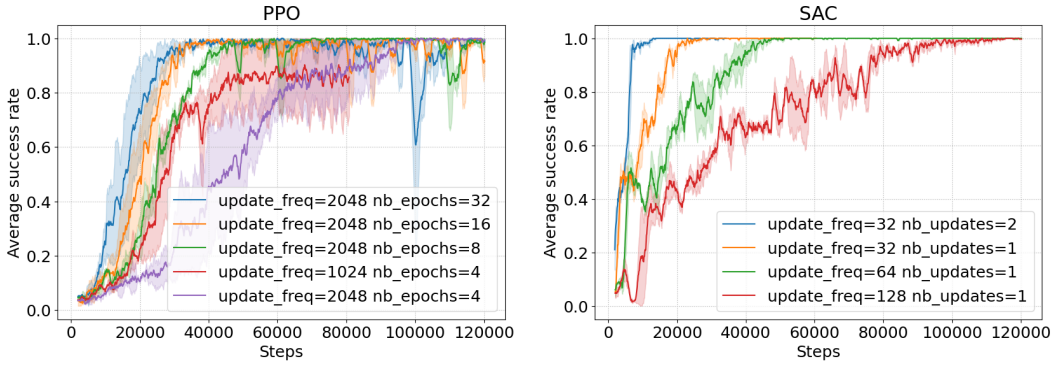

Figure 6: **Comparison of different update frequencies and epochs for PPO and SAC.** In PPO, *nb_epochs* refers to the number of times the model processes each transition during an update, while, for SAC, it represents the number of batches sampled from the replay buffer during each update.

E Batch sampling with HER

In order to optimize sample efficiency, we analyzed the impact of diversity in batches sampled from our replay buffer in order best exploit collected data. We use HER with the *future* strategy proposed in Andrychowicz et al. [2017], using all the hindsight goals returned by the social partner, and found that the replay buffer was predominantly filled with relabeled trajectories, as shown in Figure 7.

This could present two potential issues:

[1] https://github.com/flowersteam/lamorel

1. In our environment, "grasp" goals are subgoals of "grow" goals (i.e. achieving a "grow" goal requires the agent to first grasp an object). Similarly, "grow" goals are subgoals of more complex goals like "grow then grasp" or "grow then grow." This hierarchy means that once a goal is achieved, many of the trajectories added to the replay buffer by HER are from simpler goals, which should be a minority in this environment (Figure 4).
2. Because HER adds only successful trajectories, the replay buffer becomes dominated by positive examples, with very few negative ones. However, negative trajectories are also important in the learning process.

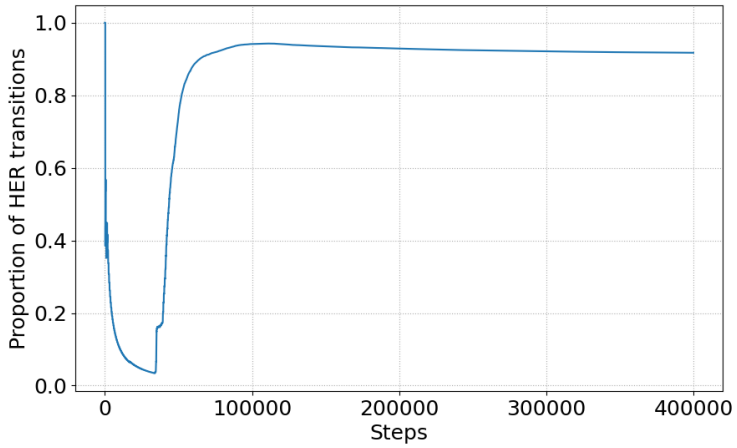

Figure 7: **Proportion of HER transitions in the replay buffer when uniformly sampling.** The plot shows the proportion of transitions relabeled by HER compared to those directly collected from the environment.

To address issue (1), we adjusted the batch sampling strategy to align with the goal distribution of the environment. Instead of uniformly sampling from the replay buffer, batches were sampled with transitions that reflected the goal distribution illustrated in Figure 4. For issue (2), we balanced the batches by sampling 50% of transitions from HER and 50% from the environment. A similar approach was used in Colas et al. [2020], where they also employed HER and sampled an equal proportion of positive and negative transitions, comparable to our method. We compared these methods in Figure 8. These strategies did not affect sample efficiency significantly, so we opted to use the "HER + ratio" strategy in our experiments, as the "prior" method makes the strong assumption of knowing the environmental goal distribution.

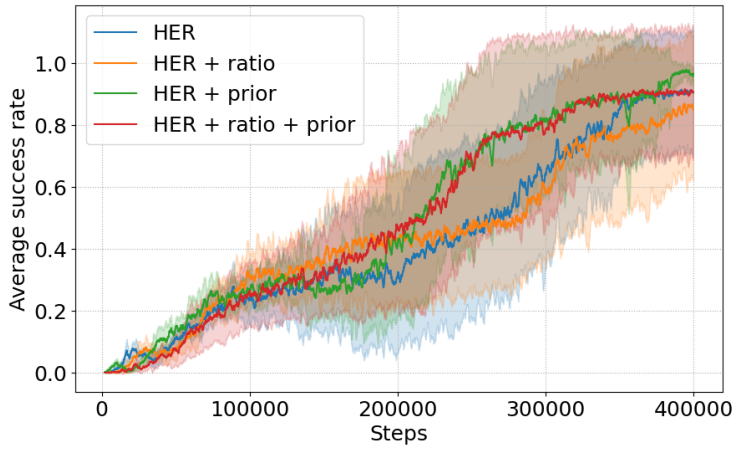

Figure 8: **Comparison of different sampling strategies.** "prior" refers to the sampling strategy used to address issue (1), while "ratio" refers to the strategy used to address issue (2).

F N-step returns parameters

We looked at how the Temporal Difference loss was influencing our sample efficiency. Increasing the n parameter of the n-step return increases the magnitude of the bootstrap, allowing returns to be propagated more quickly to earlier steps in the episode, which speeds up convergence. However, setting n too high leads to highly off-policy updates, introducing instability. We tried to mitigate this instability by using a smaller replay buffer. We compared the performance of our method across different values of n and various replay buffer sizes in Figure 9.

Based on these results, we chose to use 3-step returns in our experiments, as 5-step returns introduced instability. Additionally, we opted for a replay buffer with a capacity of 100000, which provided greater stability compared to using a larger buffer.

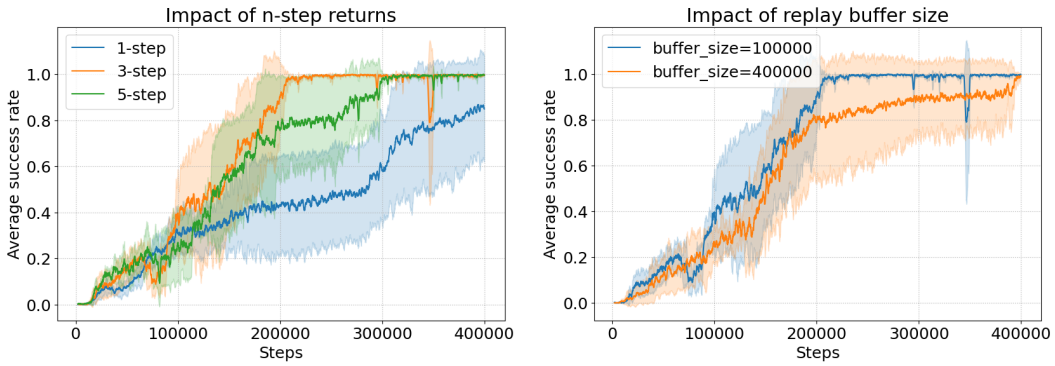

Figure 9: **Comparison of n-step parameters and replay buffer size.** The left plot illustrates the impact of the n parameter (using a replay buffer with a capacity of 100000), while the right plot shows the effect of the replay buffer capacity parameter (when using 3-step returns).

G PPO with HER

While initially designed for off-policy RL methods leveraging experience replay, answering whether HER could be applied to PPO is not straightforward. While PPO is usually presented as an on-policy method, only the first epoch of its update mechanism is strictly on-policy. Subsequent epochs become increasingly off-policy, relying on importance sampling and either clipping or KL regularization to manage the discrepancy.

In the case of the CLIP loss, the clipping mechanism is not engaged in the first epoch because the importance sampling ratio is 1:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]\,. \tag{4}$$

For the KL loss, the KL penalty is 0 in the first epoch:

$$L^{KL}(\theta) = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t + \beta KL(\pi_{\theta_{old}}(., s_t, g_t), \pi_\theta(., s_t, g_t))]\,. \tag{5}$$

The importance sampling ratio is given by $r_t(\theta) = \frac{\pi_\theta(a_t|s_t,g_t)}{\pi_{\theta_{old}}(a_t|s_t,g_t)}$. When using HER, we relabel the goal as g'. Consequently, the importance sampling ratio with HER becomes $r_t(\theta) = \frac{\pi_\theta(a_t|s_t,g'_t)}{\pi_{\theta_{old}}(a_t|s_t,g_t)}$. This adjustment means that even during the first epoch for an update, the ratio is no longer 1.

G.1 HER with the PPO KL Loss

When HER is used, the KL regularization term changes to $KL(\pi_{\theta_{old}}(., s_t, g_t), \pi_\theta(., s_t, g'_t))$. This adjustment forces the policy for g' to converge towards the policy for g, which can be problematic if g' is not a subgoal of g. For example, if g is "Grasp red lion" and g' is "Grasp blue tree", we want two distinctly different policies.

To mitigate this issue, we attempted to filter relabeled goals that are aligned with the original goal. By aligned, we mean goals that are subgoals of the original goal. For instance, if the goal is "Grasp red lion", a valid subgoal could be "Go to red lion." In such cases, the KL term is more appropriate because the optimal policies for g and g' should be similar. However, even with this filtering mechanism, relabeled goals make PPO KL unstable and lead to worse results than PPO KL without HER, as shown in Figure 10.

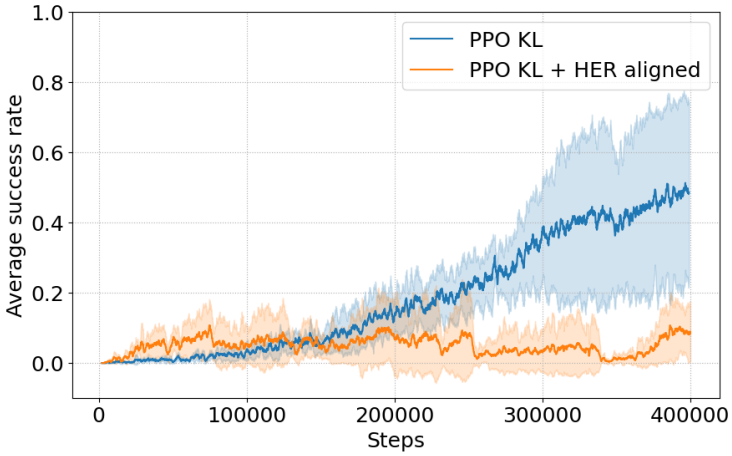

Figure 10: **PPO KL with and without HER.** "HER aligned" refers to relabeling trajectories only with goals that are subgoals of the target goal.

G.2 HER with the PPO CLIP Loss

The situation is slightly different with the clipped version of PPO as, in addition to the alignment between the policies for g and g', the clipping mechanism requires adaptations. Indeed, the clipping mechanism can be conceptualized as a ball with a radius of ϵ around a center of 1 (which should be the value of the ratio at the first epoch), constraining the updated policy from moving too far from the old policy (i.e. the policy used to collect data). However, when using HER, the first ratio is not necessarily 1, and the first ratio may not be at the ball's center or outside of it. We, therefore, tried to adjust the center of the ball to correspond to the value of the ratio at the first epoch rather than 1 in addition to the goal alignment filtering. Results from Figure 11 indicate that using only goal alignment filtering leads to on-par results with PPO without HER and that centering the clipping objective leads to even worse results.

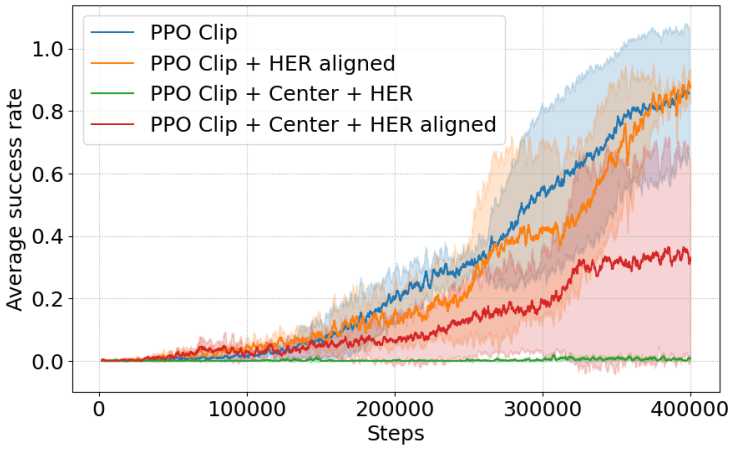

Figure 11: **PPO CLIP with HER.** "Center" refers to using the ratio between the new policy for g' in the clipping mechanism and the old policy for g, instead of 1.

H Hyperparameters

Tables 1, 2 and 3 summarize all the hyperparameters used in the experiments whose results are presented in Figure 2.

Table 1: PPO-GLAM hyperparameters

Variable	**Value**
Number of transitions collected between two updates	2048
Number of epochs per update	16
Batch size	256
Entropy loss coefficient	0.01
Value function loss coefficient	0.5
Discount factor	0.99
Optimizer	Adam
Learning rate	1×10^{-4}
λ factor of the Generalized Advantage Estimator	0.99
Clipping parameter ϵ	0.2
Maximum gradient norm	0.5

Table 2: SAC-GLAM hyperparameters

Variable	**Value**
Update frequency	32
Number of updates	2
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.005
Target entropy	0
Entropy coefficient learning rate	2×10^{-3}
n-step	3
Replay buffer capacity	100000

Table 3: SAC-GLAM + HER hyperparameters

Variable	**Value**
Update frequency	32
Number of updates	2
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.05
Target entropy	0.01
Entropy coefficient learning rate	2×10^{-3}
n-step	3
Replay buffer capacity	100000
Hindsight proportion per batch	0.5