Reinforcement Learning for Aligning Large Language Models Agents with Interactive Environments: Quantifying and Mitigating Prompt Overfitting

Mohamed Salim Aissi[1], Clement Romac[2,3], Thomas Carta[3], Sylvain Lamprier[4], Pierre-Yves Oudeyer[3], Olivier Sigaud[1], Laure Soulier [1], Nicolas Thome [1,5]
[1]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France [2]Hugging Face
[3]Inria (Flowers), University of Bordeaux, France [4]Univ Angers, LERIA, Angers, France
[5]Institut universitaire de France (IUF)
{salim.aissi, laure.soulier, olivier.sigaud, nicolas.thome}@isir.upmc.fr
{clement.romac, thomas.carta, pierre-yves.oudeyer}@inria.fr
{sylvain.lamprier}@univ-angers.fr

Abstract

Reinforcement learning (RL) is a promising approach for aligning large language models (LLMs) knowledge with sequential decision-making tasks. However, few studies have thoroughly investigated the impact on LLM agents capabilities of fine-tuning them with RL in a specific environment. In this paper, we propose a novel framework to analyze the sensitivity of LLMs to prompt formulations following RL training in a textual environment. Our findings reveal that the performance of LLMs degrades when faced with prompt formulations different from those used during the RL training phase. Besides, we analyze the source of this sensitivity by examining the model's internal representations and salient tokens. Finally, we propose to use a contrastive loss to mitigate this sensitivity and improve the robustness and generalization capabilities of LLMs.

1 Introduction

LLMs have demonstrated emergent abilities in tasks like summarization, translation or chain-of-thought reasoning (Singhal et al., 2023; Yao et al., 2024; Wei et al., 2022). Their versatility suggests they possess some common sense knowledge and reasoning capabilities (Li et al., 2021; Huang and Chang, 2022), making them potential agents for tasks such as sequential decision-making (Zeng et al., 2023; Zhao et al., 2024; Yao et al., 2022). In this context, LLMs predict actions based on a prompt, i.e., a textual description of the agent's goal and of the environment, e.g., the agent's state and its possible actions (see Figure 1).

However, LLMs often face grounding issues, i.e. they suggest actions that may be unsuitable in the current situation. This is mainly due to the inadequacy of the common knowledge embedded in the LLM to the components or dynamics of the environment at hand (Ahn et al., 2022; Huang et al., 2023). Solutions to this challenge include combining language with other modalities into multimodal models (Jiang et al., 2022; Driess et al., 2023; Li et al., 2023), using dedicated grounding modules (Huang et al., 2023; Yang et al., 2023), or employing in-context learning (ICL) to correct prompts after spotting mistakes (Shinn et al., 2023; Yao et al., 2023). To explicitly integrate the specificities of the environment into the LLM itself, an emerging strategy involves using reinforcement learning (RL) to fine-tune LLMs and integrate knowledge from executing actions (Carta et al., 2023; Tan et al., 2024). While those methods improve agent performance, it remains unclear how RL impacts the inherent knowledge of the LLM and whether the model gains generalization capabilities.

In this paper[1], we study the sensitivity of LLMs to prompt formulations and its impact on knowledge acquisition. We define a set of different prompt formulations to evaluate LLM performance when prompts are changed, and also analyze how the LLM represents these prompts. Our findings reveal that the performance of LLMs is highly sensitive

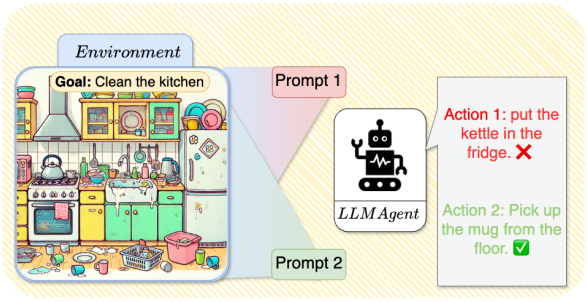

Figure 1: **Sequential decision making with LLMs.** An LLM agent interacts with an environment by means of a prompt, i.e., a textual representation of the scene contextualizing its goal, state description, and possible actions. Depending on the prompt, the LLM agent might choose different actions. In this paper, we show that LLMs fine-tuned with reinforcement learning tend to overfit to the specific prompts they have been trained on, and propose solutions to mitigate this effect.

[1]Code can be found here

to prompt variations, suggesting that fine-tuning only induces superficial updates and fails to improve the acquisition of new knowledge about the environment. By analogy with observational overfitting in RL (Song et al., 2019), we refer to this phenomenon as *prompt overfitting*. With this in mind, our paper proposes two main contributions:

• We design experiments to measure prompt overfitting issues of LLMs in interactive environments. The study reveals that fine-tuned LLMs heavily depend on the training prompts, exhibiting a significant drop in "zero-shot" performance when using new prompt formulations. To further analyze this behavior, we thoroughly analyze latent representations and salient tokens in LLMs, both showing a strong bias towards the prompt formulation.

• We propose a solution for mitigating prompt overfitting with a contrastive regularization loss that makes the latent representation of the LLM invariant to prompt formulations. This solution significantly improves the zero-shot performance and the robustness to prompt variations, as well as the acquisition of new knowledge about the environment.

Altogether, our work contributes to a better understanding of the obstacles one must face when leveraging RL to improve the abilities of LLM agents to act appropriately in interactive environments.

2 Related work

Grounding LLMs for sequential decision-making. When solving sequential decision-making tasks, an LLM acting as an agent must leverage its common sense knowledge and adapt to the environment it interacts with. Several approaches have been explored to align the LLM with its environment. A first approach, called SayCan (Ahn et al., 2022) filters out inefficient actions, depending on affordances captured from interactions. The resulting agents can solve sequential decision-making problems, but the involved LLM does not benefit from interaction feedback. Alternatively, research has focused on functionally grounding LLMs in interactive environments using online RL to align text processing with external dynamics (Carta et al., 2023; Tan et al., 2024; Wen et al., 2024; Zhou et al., 2024; Abdulhai et al., 2023). For instance, the pioneering GLAM method (Carta et al., 2023) enables LLM agents to learn policies through environmental interactions. Similarly, Szot et al. (2024) apply RL with Visual Language Models (VLMs) to solve embodied tasks but do not directly ground the VLM. Instead, they train randomly initialized neural networks on top of the VLM. Parallel to these, several works (Wang et al., 2023c,a,b) leverage prompting methods to inform the LLM about the consequences of its actions in the environment or even mix RL and prompting (Yan et al., 2023). Finally, in Xiang et al. (2023), an LLM agent collects embodied experiments in an environment to enhance its modeling abilities such as counting and tracking objects, proposing plans, etc. However, most existing works only employ a single prompt formulation, assuming that the LLM will adapt to the given format. In contrast, our study evaluates LLM performance across multiple prompts, aiming to understand how the model handles variations in formulation.

LLMs' Prompt Sensitivity. LLMs have shown remarkable capabilities to generate text and solve tasks in zero-shot and few-shot scenarios. To enhance their performance and consistency, various prompting methods have been developed (Pengfei et al., 2021). In addition, multiple studies have highlighted the sensitivity of LLMs to minor perturbations in the prompt, leading to substantially different outputs (Zhao et al., 2021; Sclar et al., 2023; Salinas and Morstatter, 2024). This sensitivity impairs the reliability and robustness of these models. Indeed, certain input-agnostic sequences could trigger specific outputs, further illustrating the brittleness of LLMs to prompt modifications (Wallace et al., 2019). The sensitivity of LLMs persists regardless of model size, the number of examples, or the type of instruction provided (Sclar et al., 2023; Zhao et al., 2021; Loya et al., 2023). These studies also reveal poor performance consistency across models using the same prompt. To improve the LLM outputs, a recalibration can estimate and adjust for the model's biases with additional parameters, which mitigates the effects of prompt sensitivity (Zhao et al., 2021). Our work extends this research by evaluating the performance of LLM agents across various prompt formulations and by mitigating prompt overfitting to preserve semantic consistency in interactive environments.

3 Problem Statement and Methods

3.1 Problem statement

To investigate the impact of RL on the knowledge of LLM agents in interactive environments, we operate in a textual RL setting. Given a vocabulary of tokens V, our experimental framework can be

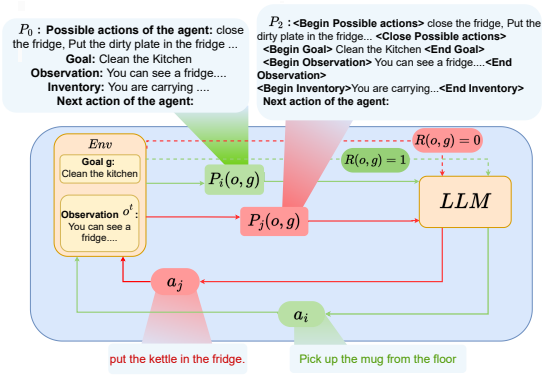

Figure 2: **The fine-tuning framework**: we use an LLM as an agent policy in a textual environment Env. The Env provides a fixed goal description g for the current episode, a description of the agent's observation o, and a scalar reward $R(o,g)$ for the current step. The goal and observation are formatted using a prompt formulation P_i. Our experiments reveal that an LLM fine-tuned on P_i succeeds with prompts formatted with P_i but fails with prompts formatted with $P_j \neq P_i$.

conceptualized as a partially observable Markov decision process $M = (S, A, T, R, G, O, \gamma)$ with S the state space, $A \subset V^N$ the action space, $G \subset V^N$ the goal space, $T : S \times A \mapsto S$ the transition function, $R : S \times G \mapsto S$ the goal-conditioned reward function, $Obs : S \mapsto V^N \equiv O$ the observation function that maps a state s to a textual description and γ the discount factor. For a trajectory also called rollout in RL $\tau = (s_1, a_1, \ldots, s_H, a_H)$ of length H, we note $R_g(\tau) = \sum_{t=1}^{H} \gamma^t R(o,g)$ its cumulative discounted reward given a goal $g \in G$. The optimal policy is $\pi^* = \arg\max_\pi \mathbb{E}_{g\in G, \tau\sim\pi(\tau|g)}[R_g(\tau)]$, with $\pi(\tau|g)$ the probability of τ following π. On top of the above framework, we study the impact of prompt formulations $P = \{P_i\}_{i\in\{1,\cdots,n\}}$, where $P_i : O \times G \mapsto \mathcal{P} \subset V^N$ formats text entries from observations and goals as prompt inputs. We assume that all formulations from P preserve information, i.e. any optimal policy π_i^* using the prompt formulation P_i from P can obtain the same amount of rewards as an optimal policy π^* acting on original observations and goals: $\forall g \in G, \forall i \in \{1, \cdots, n\}, \mathbb{E}_{\tau\sim\pi_i^*(\tau|g)}[R_g(\tau)] = \mathbb{E}_{\tau\sim\pi^*(\tau|g)}[R_g(\tau)]$.

Based on this, we analyze the sensitivity of the policy π to variations in prompt formulation compared to the one used during training. We define prompt overfitting as a scenario where, for a given policy π trained using a specific formulation P_i, there exists a prompt formulation $P_j \in P$ such that the expected reward for π on P_i is significantly higher than the expected reward for π on P_j.

3.2 Fine-tuning LLM agents with RL

First, we define a policy π, based on an LLM for solving tasks in M, given any prompt formulation $P_i \in P$. For any goal $g \in G$ and observation $o \in O$, we note $\mathbb{P}_{LLM}(w_k|p_i^{o,g}, w_{<k})$ the probability of token w_k for prompt $p_i^{o,g}$ generated by the prompt formulation P_i and the decoding history $w_{<k}$. We also define $\mathbb{P}_{LLM}(a|p_i^{o,g}) = \Pi_{k=0}^{|a|}\mathbb{P}_{LLM}(w_k|p_i^{o,g}, w_{<k})$ the decoding probability of the sequence corresponding to action $a \in A$ given prompt $p_i^{o,g}$. Following (Tan et al., 2024), we use a normalized decoding probability $\mathbb{P}^{norm}_{LLM}(a|p_i^{o,g}) = \mathbb{P}_{LLM}(a|p_i^{o,g})^{\frac{1}{|a|}}$ to better balance actions of various sizes. From these quantities, we build the policy as $\pi(a|p_i^{o,g}) = \mathbb{P}^{norm}_{LLM}(a|p_i^{o,g})/Z_i^{o,g}$, where $Z_i^{o,g} = \sum_{a\in A}\mathbb{P}^{norm}_{LLM}(a|p_i^{o,g})$ is the partition function. Then, we follow the recent textual RL works for sequential decision-making of Carta et al. (2023) and Tan et al. (2024) to train our LLM agents using Proximal Policy Optimization (PPO) (Schulman et al., 2017). Given a subset of prompt formulations $P_{ppo} \subseteq P$, we note $\pi_{P_{ppo}}$ an LLM agent of parameters θ trained by minimizing the PPO loss $PPO_{loss}(\theta, P_{ppo})$ from rollouts using prompt formulations P_{ppo}. Each rollout τ is obtained for a given prompt formulation $P_i \in P_{ppo}$ uniformly sampled from P_{ppo}, and is used by the agent during training.

3.3 Mitigating Prompt Overfitting with Contrastive Learning

After analyzing prompt overfitting within the proposed framework, we add a contrastive loss to bring closer the latent representations $z_\theta(p_i^{o,g})$ and $z_\theta(p_j^{o,g})$ of the same observation-goal pair (o,g) across prompts $p_i^{o,g}$ and $p_j^{o,g}$, compared to latent representations $z_\theta(p_i^{o',g'})$ of other observations and goals (o',g') of prompt $p_i^{o',g'}$. That is, we aim to minimize:

$$C^{(i,j)}(\theta) = \mathop{\mathbb{E}}_{\substack{(o,g)\sim d^{\pi_{P_{ppo}}} \\ (o',g')\sim d^{\pi_{P_{ppo}}}}} \max\Big(\Delta(z_\theta(p_i^{o,g}), z_\theta(p_j^{o,g})) - \Delta(z_\theta(p_i^{o,g}), z_\theta(p_i^{o',g'})) + 1), 0\Big) \quad (1)$$

where $z_\theta(p_i^{o,g})$ is the latent representation of textual prompt $p_i^{o,g}$ produced by prompt formulation P_i applied to the observation-goal pair (o,g), $\Delta(z,z') = ||(z) - (z')||_2^2$ is the Euclidean distance between two latent representations z and z', and $d^{\pi_{P_{ppo}}}$ is the joint stationary observation-goal distribution of a policy using P_{ppo}. In practice, we sample

pairs (o, g) from the PPO rollouts, supposed to follow $d^{\pi_{P_{ppo}}}$.
Our final loss L jointly optimizes PPO_{loss} with the contrastive loss $C^{(i,j)}(\theta)$ as follows:

$$L(\theta, P_{ppo}, P_c) = PPO_{loss}(\theta, P_{ppo}) + \frac{\alpha}{|P_c|^2} \sum_{P_i, P_j \in P_c^2} C^{(i,j)}(\theta) \quad (2)$$

where α is a parameter regulating the impact of $C^{(i,j)}$, P_{ppo} represents the set of prompts used for fine-tuning the LLM policy with the PPO loss, and P_c represents the set of prompts used for the contrastive regularization $C^{(i,j)}$. Following (Ni et al., 2022), $C^{(i,j)}$ is only applied to the representation of the first token of the prompt. Appendix C provides implementation details about contrastive regularization.

4 Experimental Protocol

In this section, we introduce an evaluation methodology for analyzing the impact of fine-tuning LLMs with RL in textual environments. Our experiments aim to address three research questions:

• **Q1: Prompt sensitivity:** Are LLMs sensitive to different prompt formulations, and how does this sensitivity influence their ability to generalize across various prompt formulations?

• **Q2: State representation:** How do LLMs encode the state space in their hidden representation, and what is the topology of the latent space?

• **Q3: Impact of prompt information on action choice:** After fine-tuning with various prompts, on which parts of the prompt does the LLM agent focus for task completion?

4.1 Environments

We conduct our experiments in two textual environments: 1) BabyAI-Text, a mini-grid environment used in (Carta et al., 2023) where an agent navigates through limited actions, and 2) the Medium difficulty TWC Environment, proposed in (Murugesan et al., 2021) (noted TWC-Medium), targeted to solve household tasks getting a scene description and a list of possible actions. These two environments assess complementary skills in terms of semantics: BabyAI-Text requires exploring and understanding the arrangement of objects, whereas TWC-Medium requires common sense knowledge and reasoning. See Appendix A for more details.

4.2 Prompt Design

To use LLMs as policies, we define four distinct prompt formulations to gather several pieces of information from the environment: the goal, possible actions, an inventory and textual observations. The first prompt formulation P_0 follows a format where pieces of information are separated by line breaks in the following order: possible actions, goal, observation, and inventory. P_1 is similar to P_0 but switches the order of the information. P_2 employs a more rigid syntax with delimiter tags, similar to an XML file. Finally, P_3 removes all rigidity in syntax and follows a natural language format, paraphrased by a prompt writer. Examples of P_0 and P_2 in TWC-Medium can be found in Figure 2 and detailed descriptions of all prompt formulations are provided in Appendix A.3.

4.3 Training and Evaluation Details

Training: We consider several LLMs including encoder-decoder architectures (Flan-T5 78M, 780M, and 2.7B) (Chung et al., 2022) and decoder-only architectures (GPT-Neo 1.3B, LLama 7B) (Black et al., 2022; Touvron et al., 2023). We present in the main paper the results obtained by Flan-T5 78M and 780M also used in (Carta et al., 2023) and GPT-Neo 1.3B. Additional results on other LLMs are presented in Appendix B.1. We consider different fine-tuning scenarios denoted as $\sigma^{P_c}_{P_{ppo}}$. For brievity, we omit to specify P_c if no contrastive regularization is applied. Also, we denote as σ_{zs} the zero-shot scenario, that corresponds to the pre-trained LLM agent without any fine-tuning (i.e., both P_c and P_{ppo} are empty). Three scenarios are mainly considered for fine-tuning LLM agents in our experiments: 1) the one prompt scenario σ_0 where the LLM is fine-tuned by PPO on prompt P_0 only, 2) the multiple prompt scenario $\sigma_{0:3}$ where the LLM is fine-tuned by PPO on prompts P_0 to P_3, and 3) the contrastive scenario $\sigma_0^{0:3}$ where the LLM is fine-tuned by PPO with P_0 only, but using an additional contrastive loss considering prompts P_0 to P_3. Fine-tuning is performed in both environments for 500k steps across 5 seeds.

Evaluation: We evaluate the zero-shot scenario σ_{zs} and the fine-tuned LLMs ($\sigma^{P_c}_{P_{ppo}}$) to assess the performance on both seen and unseen prompts during training. In addition, for the scenarios $\sigma_{0:3}$ and $\sigma_0^{0:3}$, we define a new prompt formulation P_4 to analyze generalization to unseen prompts. P_4 follows a different template with changes to the section's names, reordering, and an additional context tag (details about this prompt in Appendix A.3).

4.4 Metrics

We assess LLM agents around four axes:

• **Performance Related to Training Task:** We define a success of an LLM agent in the environment as a case where a trajectory is successful if it reaches its goal g. We deduce two metrics: 1) (**SR**): the success rate of the agent and 2) ($\overline{\textbf{SR}}$): the mean success rate across all prompt formulations and episodes.

• **Exploring Hidden Representations:** To delineate the disparities between prompt formulations, we compute the cosine similarity of their latent representations pre and post fine-tuning. Given a set of observation-goal pairs $\Gamma = \{(o, g)\}$[2], the similarity between representations using different inputs formatted by a single prompt formulation P_0 is defined as

$$Intra(P_i) = \frac{1}{|\Gamma|^2 - |\Gamma|} \sum_{\substack{(o,g)\in\Gamma \\ (o',g')\in\Gamma\setminus\{o,g\}}} cos\left(z_i^{o,g}, z_i^{o',g'}\right).$$

Besides, we assess the similarity between representations from different prompt formulations as:

$$Inter(P_i, P_j) = \frac{1}{|\Gamma|} \sum_{(o,s)\in\Gamma} cos\left(z_i^{o,g}, z_j^{o,g}\right).$$

Finally, to visualize the topology of the latent space, we utilize UMAP (McInnes et al., 2018) to generate a 2D projection of the hidden state space and visualize its structure.

• **Relevance of Parts of Prompt Information:** We study the relationship between inputs and predictions using gradients attribution methods to generate saliency scores for each element of the input (Madsen et al., 2021). For each scenario σ and each prompt formulation P_i, we compute the averaged saliency of prompt tokens for every observation-goal pair (o, g) from Γ. Saliency scores are computed using the `Inseq` toolkit (Sarti et al., 2023), as the Integrated Gradient of the output probability of the policy $\pi(a^*|p_i^{o,g})$ with respect to the tokens of the prompt $p_i^{o,g}$, with a^* the most likely action regarding the observation-goal o, g. To enhance interpretability, saliency scores for each prompt tokens are finally aggregated depending on the part of the prompt they belong to, among {possible actions, goal, observation, inventory}. We do so after filtering out the top 5% most significant tokens from each section to eliminate perturbations caused by irrelevant tokens.

[2]In our experiments, Γ is built from rollouts of σ_0 to infer actions. Similar results were observed with other distributions.

• **Knowledge acquired by the LLM about the environment:** To analyze what the LLM learned about the environment, we measure the accuracy in question-answering (QA) tasks related to TWC-Medium that include object counting (TWC OC) and task-related questions (TWC QA). We follow the (Xiang et al., 2023) methodology for constructing the set of questions based on the optimal successful trajectory of TWC-Medium. For task-related questions, we generate multiple-choice questions regarding the best object from a given set to accomplish a task. For the object counting task, we present to the LLM a trajectory in the environment and prompt it to count the number of objects in a specific location.

5 Quantifying overfitting

We now focus on the impact of fine-tuning LLMs in interactive environments according to our three main research questions. We leave the impact of the contrastive learning scenario on task achievement and question answering for Section 6.

5.1 Q1: Prompt sensitivity

We evaluate the LLM performance in solving tasks with different prompt formulations in both environments. Figure 3 shows the SR values for Flan-T5 78M, 780M, and Gpt-Neo 1.3B obtained according to the different evaluation scenarios: σ_{zs}, σ_0, and $\sigma_{0:3}$. Results of training with other formulations and models are available in Appendix B.1.

First, we see that LLM performance in σ_{zs} is 50% lower than that of models fine-tuned via PPO. This underscores the necessity for LLM specialization in interactive environments to efficiently achieve goals. Conversely, while the 78M model and GPT-Neo 1.3B exhibit heterogeneous results across both environments in σ_{zs}, the Flan T5 780M model demonstrates more consistent performance. Second, fine-tuning the LLM with a single prompt formulation, notably P_0, enhances the SR value for the training prompt, revealing that the LLM learns to effectively achieve tasks and capture the dynamics of interactive environments. For more details, the training curves are provided in Appendix A.4. For instance, in the TWC-Medium environment, both the Flan T5 78M and 780M models achieve success rates exceeding 90% (compared to a maximum of 45% with σ_{zs}) and approach the 85% success rate of the GPT-Neo 1.3B (compared to at most 20% with σ_{zs}). However, a notable decline

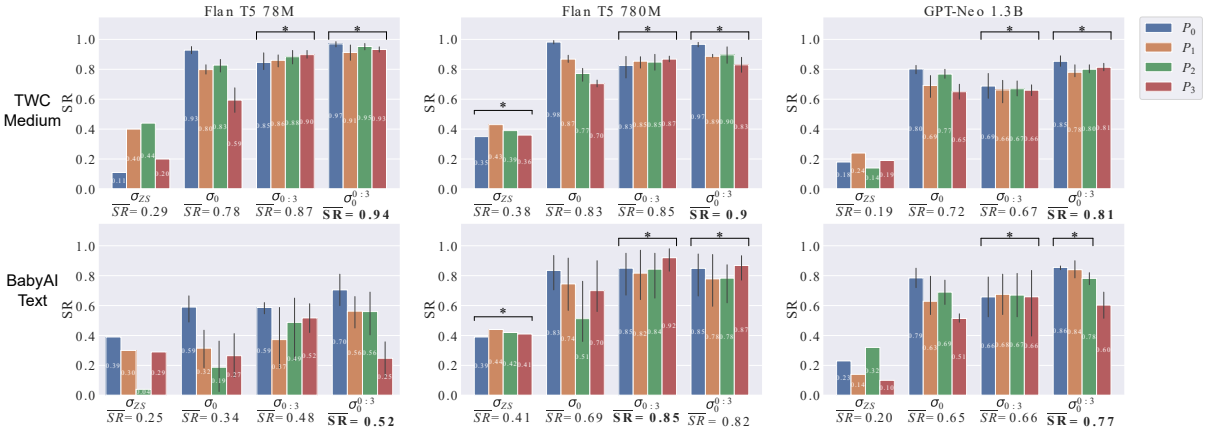

Figure 3: **Success Rate (SR)** in BabyAI-Text and TWC-Medium. The x-axis indicates the training scenario, while colors represent the prompt formulation used to format inputs during rollouts. The asterisk (*) indicates instances where the chi-squared test exceeds the homogeneity threshold (p-value > 0.05). Bolded values in $\overline{\mathbf{SR}}$ represent the best results in the evaluation. Results show that the LLM exhibits heterogeneous performance when the prompt formulation used during training is changed, with a drop in success rate of up to 30% in certain scenarios.

in performance is observed when using another prompt formulation at test time, with a decrease of over 30% from using the original P_0 to the P_3 variation in TWC-Medium and more than 30% from P_0 to P_2 in BabyAI-Text. This drop outlines prompt overfitting in LLMs. The trend is similar for larger models (Flan T5 2.7B and LLama 7B) as detailed in Appendix B.1. We also observe that, for both σ_{zs} and fine-tuned models, encoder-decoder models outperform decoder-only architectures, even when the decoder-only models have a larger parameter size. Finally, results obtained with $\sigma_{0:3}$ indicate that training with all prompts maintains the LLM's performance more consistently across prompt variations. However, the LLM does not achieve the peak of performance observed when trained on a single prompt formulation. This is likely due to the higher difficulty for the LLM to adapt to multiple formulations than to a single formulation. By comparing environments, we observe that the Flan T5 78M model struggles in learning appropriate policies in BabyAI-Text. This might be attributed to its exploratory nature and reduced reliance on common sense knowledge, due to the small portion of exploited vocabulary.

5.2 Q2: State representation

To further analyze prompt overfitting, we investigate the latent representation of states formatted with different prompt formulations. We compare in Table 1 the intra-prompt and inter-prompt similarities to measure the topology of the latent representations of states according to our different scenarios (σ_{zs}, σ_0, and $\sigma_{0:3}$), averaged over any pair of prompt formulations used to format inputs. The most striking result is that LLMs tend to cluster prompts formulated with the same prompt formulation ($Intra \simeq 1$), even when they correspond to different goal-observation pairs. In contrast, the same pair formulated with different prompt formulations exhibits low similarity ($Inter < 0.5$).

Models		σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
$78M$	$Intra$	0.992 ± 0.003	0.991 ± 0.003	0.991 ± 0.003	0.907 ± 0.017
	$Inter$	0.376 ± 0.019	0.382 ± 0.020	0.371 ± 0.020	0.806 ± 0.029
$780M$	$Intra$	0.998 ± 0.001	0.997 ± 0.001	0.998 ± 0.001	0.939 ± 0.017
	$Inter$	0.469 ± 0.462	0.458 ± 0.449	0.47 ± 0.461	0.812 ± 0.06
$1.3B$	$Intra$	0.995 ± 0.001	0.994 ± 0.001	0.997 ± 0.001	0.994 ± 0.017
	$Inter$	0.552 ± 0.03	0.539 ± 0.01	0.501 ± 0.05	0.94 ± 0.009

Table 1: **Inter and intra-similarity** comparison for Flan T5 78M, 780M and GPT-Neo 1.3B models on TWC-Medium on our different scenarios. For all models, we observe that $Intra$ approaches 1, while $Inter$ is consistently below 0.5. This indicates that the LLMs tend to cluster prompts based on their formulation rather than on content. A similar trend is observed for both $\sigma_{0:3}$ and σ_0 scenarios.

The low similarity between prompts and the high similarity within prompts suggest that LLMs capture more about the prompt formulation than about the content itself. This reinforces the previous observation about Q1 highlighting prompt overfitting. Figure 4 depicts the embedding of prompts using

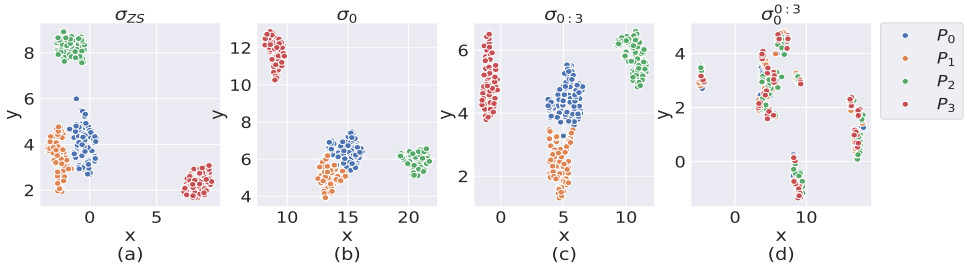

Figure 4: **UMAP visualization of hidden representations in GPT-Neo 1.3B across 100 states of TWC-Medium using four prompt formulations** demonstrating clustering based on prompt formulation over semantic state similarity in both σ_{zs} and fine-tuned models σ_0 and $\sigma_{0:3}$. Additional results for fine-tuning on other prompts can be found in Appendix B.3.

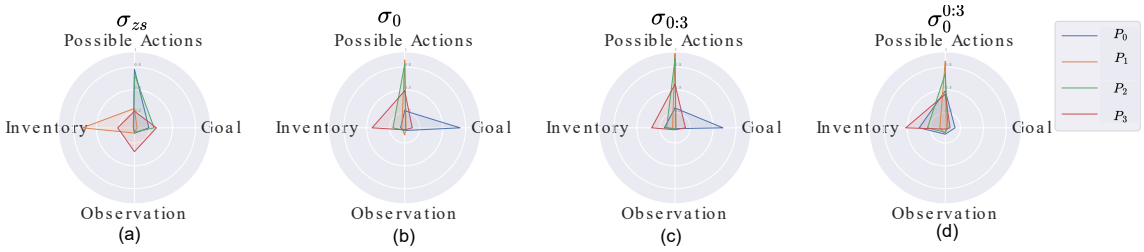

Figure 5: **Saliency maps of the Flan-T5 78M model across scenarios on TWC-Medium**, highlighting various parts of the prompts. The maps show that the LLM focuses on different sections of the prompt depending on the prompt formulation. This variation in focus is linked with performance changes when the prompt formulation is altered. The phenomenon is observed in σ_{zs}, σ_0 and $\sigma_{0:3}$ scenarios.

the UMAP visualization and also corroborates this hypothesis. We see that there is no overlap between clusters of prompt formulations in σ_{zs}, and fine-tuning with σ_0 does not change this. Interestingly, using $\sigma_{0:3}$ does not mitigate this, highlighting the need for methods tackling overfitting in both task efficiency (Q1) and state representation (Q2).

5.3 Q3: Usefulness of Prompt Information

We then analyze which parts of the prompts (goal, possible actions, observation, and inventory) are used by LLMs to predict the best action. Figure 5 depicts the saliency scores of prompt parts obtained using the Integrated Gradients algorithm (Madsen et al., 2021). We observe that the importance of the parts of the prompt varies regarding the different prompt formulations and scenarios. For instance, in σ_{zs}, LLMs prioritize Inventory when P_1 is used, whereas switching to P_0 and P_2 shifts the focus to Possible Actions, with quite similar saliency scores for both. This can be explained by the fact that P_0 and P_2 follow the same order of information (see Appendix A.3), so LLMs find possible actions in the same relative position. Fine-tuning LLMs alters the saliency scores compared to σ_{zs}, with LLMs prioritizing the goal over possible actions when using P_0. However, LLMs still focus on different parts of the prompt when changing prompt formulation to the others. Using $\sigma_{0:3}$ also results in heterogeneous saliency maps across prompt formulations, showing that LLMs continue to process the prompts differently despite being fine-tuned with multiple prompt formulations. This strengthens the findings of Q2, highlighting the influence of prompt formulation on the behavior of LLMs. Additional results for fine-tuning on other prompts can be found in Appendix B.3.

6 Mitigating Overfitting with Contrastive Learning Regularization

We now evaluate the impact of the contrastive loss proposed in Section 3.3 with the aim to help the LLMs focus more on the content in prompts so as to mitigate prompt overfitting.

Performance and prompt overfitting: Figure 3 reports the SR and prompt homogeneity across both environments and three model sizes. In most cases, the $\overline{\textbf{SR}}$ of $\sigma_0^{0:3}$ surpasses the σ_{zs}, σ_0 and $\sigma_{0:3}$ (except for the 780M model on BabyAi-Text, where $\sigma_0^{0:3}$ underperforms by 3%). This is noticeable since $\sigma_0^{0:3}$ learns the matching between prompt formulations while $\sigma_{0:3}$ learns each prompt indepen-

	$\sigma_{0:3}$	$\sigma_0^{0:3}$
78 M	0.77 ± 0.11 (3%)	0.92 ± 0.02 (97%)
780 M	0.80 ± 0.06 (4.7%)	0.86 ± 0.05 (91%)
1.3 B	0.66 ± 0.02 (99%)	0.76 ± 0.03 (98%)

Table 2: **Success Rate (SR) in TWC-Medium** using prompt formulation P_4 unseen during training. The values in parentheses represent the χ^2 homogeneity test results, red indicates heterogeneity (p-value < 5%) and green indicates homogeneity (p-value > 5%). $\sigma_0^{0:3}$ scenario demonstrates superior performance in terms of SR and homogeneity compared to $\sigma_{0:3}$ across all models.

dently. In terms of homogeneity, both $\sigma_0^{0:3}$ and $\sigma_{0:3}$ scenarios yield consistent results (marked with an asterisk (*) in Figure 3) for all TWC-medium evaluations. However, as mentioned in Section 5.1, the 78M model struggles in BabyAI-Text with both task-solving and achieving homogeneity across prompts for $\sigma_0^{0:3}$ and $\sigma_{0:3}$. For the GPT-Neo 1.3B model, $\sigma_0^{0:3}$ achieves homogeneity across prompts P_0 to P_2 but its performance drops with P_3. This is likely because the environment involves extensive exploration, making it difficult to maintain semantic consistency with paraphrased text. Nevertheless, the $\overline{\textbf{SR}}$ for $\sigma_0^{0:3}$ remains higher than that of $\sigma_{0:3}$.

We also evaluate in Table 2 the generalization capabilities of $\sigma_0^{0:3}$ compared to $\sigma_{0:3}$ using the prompt formulation P_4, which was not seen during training in either scenario. The results indicate that the mean success rate on P_4 is higher for $\sigma_0^{0:3}$ than for $\sigma_{0:3}$, indicating superior generalization capabilities across all three model sizes. To validate these findings, we conduct a χ^2 test to verify the homogeneity of P_4 results compared to the mean results obtained from the training prompts. The values in parentheses summarize the results, showing that for all $\sigma_0^{0:3}$ models, the success rates on the unseen prompt P_4 follow the same distribution as the training success rates. This indicates robust generalization to unseen prompts, in contrast to $\sigma_{0:3}$, where the 78M and 780M models do not exhibit the same distribution of results as during training. This finding supports the conclusion that regularization helps mitigate overfitting.

State representation: The last column of Table 1 shows a significant increase in the inter-prompt proximity for the 78M, 780M and 1.3B models, confirming our intuition that applying the contrastive loss can help disregard the prompt formulation at the benefit of its content. The scatter plot in Figure 4(d) indicates that the topology of the latent space changed. The latent vectors are no longer clustered by formulation and now overlap,

	TWC QA	**TWC OC**
σ_{zs}	0.4866 *	0.0876 ***
σ_0	0.4901 *	0.1340 ***
$\sigma_{0:3}$	0.5019 *	0.2526 *
$\sigma_0^{0:3}$	**0.6322**	**0.5155**

Table 3: **Impact on knowledge of the LLM about the environment** on TWC QA and TWC OC datasets. * and *** correspond to the p-value (resp. < 0.05 and < 0.001) of Welch's t-test to compare the performance between $\sigma_0^{0:3}$ and other scenarios. We observe a significant improvement with $\sigma_0^{0:3}$ scanerio compared to σ_{zs}, σ_0, and $\sigma_{0:3}$ scenarios across both datasets.

suggesting that the model has learned to match between states. Further details regarding distances and visualizations can be found in Appendix B.2.

Saliency of prompt information: Figure 5(d) shows that contrastive loss homogenizes saliency maps across prompts, as it impacts their performance. Indeed, we see more homogeneous results in the salient prompt parts across prompt formulations, unlike when training on multiple prompts.

Knowledge acquired by the LLM about the environment. We measure the accuracy of the LLM in TWC QA and TWC OC datasets before and after fine-tuning over all training scenarios in Table 3. Before fine-tuning, the LLM struggles to answer environment-related questions, exhibiting poor performance across both QA datasets. Following fine-tuning with σ_0 or $\sigma_{0:3}$, the accuracy shows only superficial improvement for the TWC QA, and minor enhancements are observed in the TWC OC compared to the σ_{zs} setting. This indicates that fine-tuning in these scenarios leads to superficial updates. However, fine-tuning with the $\sigma_0^{0:3}$ scenario results in a more substantial improvement compared to other scenarios, with at least a 13% increase in TWC QA accuracy, and respective gains of 25%, 35%, and 43% for the $\sigma_{0:3}$, σ_0, and σ_{zs} scenarios. These findings highlight that not only enhances robustness to prompt variations but also improves the LLM's understanding of the environment.

Altogether, these results highlight the effectiveness of our contrastive method to mitigate prompt overfitting according to different criteria: 1) the impact of prompt sensitivity on performance, 2) leveraging state description rather than prompt formatting, 3) giving importance to the same information in the prompt and 4) have better knowledge about the environment. It is worth noting that this is achieved with the benefit of better generalization, without

sacrificing performance compared to a scenario that leverages multiple prompts simultaneously during fine-tuning.

7 Conclusion and future work

In this work, we studied the sensitivity of LLMs to prompt formulation during RL fine-tuning. We introduced an evaluation protocol to assess prompt overfitting, considering Success Rate, and the role of internal mechanisms such as embeddings and saliency. We evaluated three models of varying sizes (Flan T5 78M and 780M, and GPT-Neo 1.3B) across two environments (BabyAI-text and TWC-medium). The results revealed the impact of prompt overfitting, which increased the model's sensitivity to variations in prompt formulation. To address this, we proposed a contrastive regularization method and demonstrated its effectiveness. However, this study has limitations. We focus on solutions where LLM interacts with the world solely through text, and requires detailed descriptions at each step. Evaluating other modalities (like images) will be addressed in future work.

Acknowledgments

Experiments presented in this paper were carried out using the HPC resources of IDRIS under the allocation 2024-[A0151013011] made by GENCI. This work was supported by the European Commission's Horizon Europe Framework Programme under grant No 101070381 (PILLAR-robots) and by PEPR Sharp (ANR-23-PEIA-0008, ANR, FRANCE 2030).

Limitations

Our evaluation relies on fine-tuning LLMs, which is computationally intensive and time-consuming. Furthermore, using RL further slows down the training process due to the need for interactions with the environment and RL optimization. For instance, training a Flan T5 78M model requires four NVIDIA V100-32GB GPUs, while a 780M model necessitates four NVIDIA A100-80GB GPUs and GPT-Neo 1.3B necessitates eight NVIDIA A100-80GB. Consequently, fine-tuning larger models is challenging given our available computational resources.

Ethical Considerations

In our research, we investigate the sensitivity of LLMs to prompts and propose solutions to mitigate this issue. We believe that our efforts to reduce sensitivity and align LLM outputs with human intentions will be beneficial for the application of LLMs in real-world tasks, and represent a step forward for the implementation of LLMs in robotics tasks and beyond.

References

Marwa Abdulhai, Isadora White, Charlie Victor Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. 2023. LMRL gym: Benchmarks for multi-turn reinforcement learning with language models. Technical report, Berkeley's AI Research lab.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *ArXiv preprint*, abs/2204.01691.

Sid Black, Stella Biderman, Eric Hallahan, Quentin G. Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Martin Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Benqi Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model. *ArXiv*, abs/2204.06745.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. PALM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. 2023.

Grounded decoding: Guiding text generation with grounded models for robot control. *ArXiv preprint*, abs/2303.00855.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2022. Vima: General robot manipulation with multimodal prompts. *arXiv*.

Jiachen Li, Qiaozi Gao, Michael Johnston, Xiaofeng Gao, Xuehai He, Suhaila Shakiah, Hangjie Shi, Reza Ghanadan, and William Yang Wang. 2023. Mastering robot manipulation with multimodal prompts through pretraining and multi-task fine-tuning. *arXiv preprint arXiv:2310.09676*.

Xiang Lorraine Li, Adhiguna Kuncoro, Jordan Hoffmann, Cyprien de Masson d'Autume, Phil Blunsom, and Aida Nematzadeh. 2021. A systematic investigation of commonsense knowledge in large language models. *arXiv preprint arXiv:2111.00607*.

Manikanta Loya, Divya Sinha, and Richard Futrell. 2023. Exploring the sensitivity of llms' decision-making capabilities: Insights from prompt variations and hyperparameters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Andreas Madsen, Siva Reddy, and A. P. Sarath Chandar. 2021. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55:1 – 42.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9018–9027.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Liu Pengfei, Yuan Weizhe, Fu Jinlan, Jiang Zhengbao, Hayashi Hiroaki, and Neubig Graham. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Abel Salinas and Fred Morstatter. 2024. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. *arXiv preprint arXiv:2401.03729*.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, Oskar van der Wal, Malvina Nissim, and Arianna Bisazza. 2023. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 21–435, Toronto, Canada. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. 2019. Observational overfitting in reinforcement learning. *arXiv preprint arXiv:1912.02975*.

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. 2024. Large language models as generalizable policies for embodied tasks. *arXiv:2310.17722*.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. 2024. True knowledge comes from practice: Aligning llms with embodied environments via reinforcement learning. *arXiv preprint arXiv:2401.14151*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv: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. 2023b. JARVIS-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Muning Wen, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Entropy-regularized token-level policy optimization for large language models. *arXiv preprint arXiv:2402.06700*.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. 2023. Language models meet world models: Embodied experiences enhance language models. *ArXiv preprint*, abs/2305.10626.

Xue Yan, Yan Song, Xinyu Cui, Filippos Christianos, Haifeng Zhang, David Henry Mguni, and Jun Wang. 2023. Ask more, know better: Reinforce-learned prompt questions for decision making with large language models. *arXiv preprint arXiv:2310.18127*.

Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. 2023. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. *arXiv preprint arXiv:2309.12311*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. 2023. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*.

Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. 2023. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. ArCHer: Training language model agents via hierarchical multi-turn RL. *arXiv*.

Appendix

A Experimental setup

A.1 Environments

The BabyAI-Text environment (Carta et al., 2023) encapsulates BabyAI, a simple mini-grid environment where an agent navigates and interacts with objects through a limited action space of six commands: *turn left, turn right, go forward, pick up, drop, and toggle*. BabyAI-Text describes each observation with sentences instead of using a symbolic representation. A textual description consists of a list of template descriptions with the following structure:

- "You see a <object> <location>" if the object is a key, a ball, a box or a wall
- "You see a(n) open/closed door <location>", if the agent sees a door.
- "You carry a <object>", if the agent carries an object.

The TWC environment (Murugesan et al., 2021) is notably more complex than BabyAI-Text regarding objects and actions and more amenable to common sense knowledge. TWC agents should achieve a series of household tasks, such as "picking up an apple and placing it in an appropriate location". The agent receives a description of a scene and a list of plausible actions. It must then decide which action to take given the current game state. Successful actions are rewarded with points.

TWC games are categorized into easy, medium, and hard difficulties. As difficulty increases, the number of target objects and rooms to clean up also increases, as detailed in Table 4.

	Objects	**Targets**	**Rooms**
Easy	1	1	1
Medium	2-3	1-3	1
Hard	6-7	5-7	1-2

Table 4: Number of objects, target objects and rooms in TWC games per difficulty level.

To choose a difficulty level, we first conducted a σ_{zs} evaluation on TWC-Easy using different prompt formulations. The results, summarized in Table 5, indicate that the LLM does not encounter significant difficulties in solving tasks in σ_{zs}. This is why we also performed training and evaluation on TWC-Medium, where the LLM struggles in σ_{zs}, to better analyze its performance and adaptation.

	P_0	P_1	P_2	P_3
78M	0.73	0.81	0.75	0.83
780M	0.83	0.9	0.86	0.88

Table 5: LLM evaluation in zero-shot on TWC-Easy

A.2 Grounding Evaluation

In this section, we provide details on the dataset used for grounding evaluation, following the methodology of (Xiang et al., 2023). The evaluation consists of two tasks:

(1) QA task, where the model is asked to identify the relevant object needed to complete a household activity. Example: *"Question: To wash clothes, a possibly related item could be. Possible answer: ["highlighter", "crackers", "laundry detergent", "cupcake"] Answer:*

(2) Object counting task, where the model must determine the number of objects in a specific location. Example: *"you opened a cooking pot and grabbed an apple. Next, you pulled out a dish bowl and scrubbed another apple. He found a bookshelf and put the first apple on it. Then, you opened a clothes pile and washed it before putting it on the same bookshelf. He grabbed another dish bowl and plate and put the cutlets on the bookshelf. He moved the clothes pile, grabbed the first apple and moved it back to its original spot on the bookshelf. Finally, you put the second dish bowl on the bookshelf. How many items are there on the bookshelf?"*

A.3 Models and Prompt Variations

In this section, we detail the prompt formulations and provide examples of trajectories performed by the agent.

Figure 7 shows an example of the initial state s^1 in TWC-Medium, formatted with different prompt formulations. Similarly, Figure 8 presents the same for BabyAI-Text.

A.4 Training

In this section, we present success rate curves for models trained on σ_0, $\sigma_{0:3}$, and $\sigma_0^{0:3}$ (see Figure 9). All training scanrios converge, exceeding a 90% success rate. This indicates that the policy effectively learned to solve the required tasks. The models are trained for an identical number of steps to ensure a fair performance comparison.

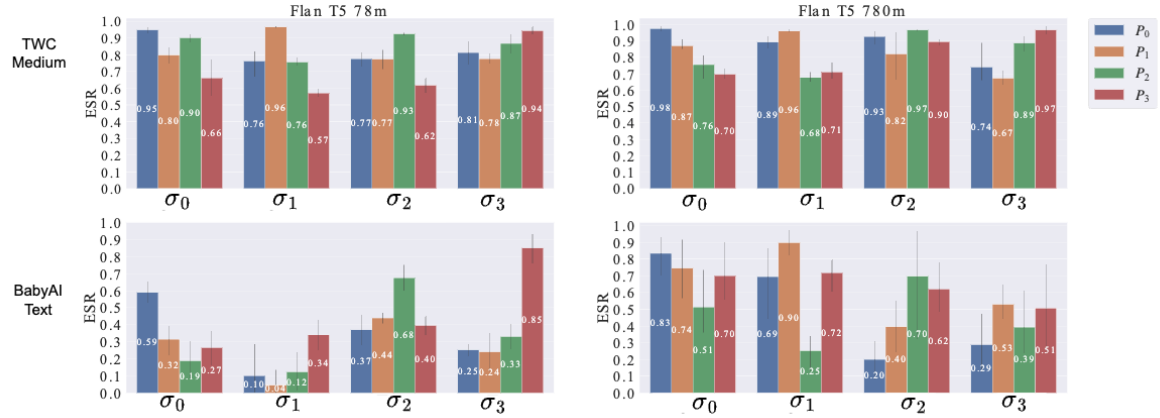

Figure 6: **Complementary results of success rates** for both the 78M and 780M models in two environments across prompt formulations P_0 to P_3.

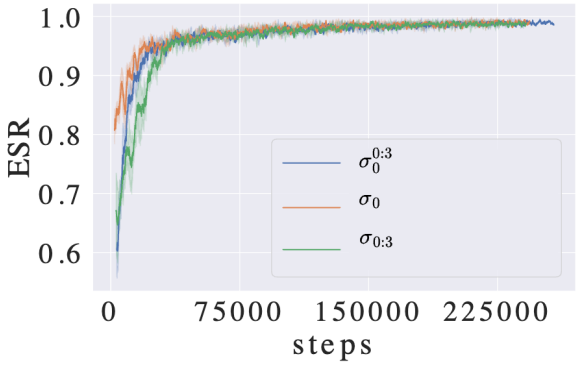

Figure 9: **Evolution of the Success Rate (SR)** during training in TWC-Medium.

B Quantifying overfitting

In this section, we present additional results on the three questions studied in the main paper.

B.1 Q1: Prompt sensitivity

First, we perform the same success rates analysis for LLMs trained with different prompt formulations in two environments. Figure 6 displays these results. Prompt overfitting is also evident when fine-tuning with P_1, P_2, and P_3, further supporting our findings. Additionally, we note that the 78M model struggles with BabyAI-Text but shows excellent performance when trained on P_3, indicating that the LLM can better adapt to certain prompt formulations.

We also evaluate larger models like Flan T5 2.7B (Chung et al., 2022) and LLama 7B (Touvron et al., 2023). Table 6 summarizes the results, demonstrating that prompt overfitting is also present in larger models. Due to the computational cost of fine-tuning such large LLMs, this evaluation was performed exclusively on the TWC environment with σ_0 scenario.

Models	P_0	P_1	P_2	P_3
Llama 7B fine-tuned with σ_0	0.9	0.72	0.75	0.79
Flan T5 XL 2.7B fine-tuned on σ_0	0.93	0.89	0.84	0.74

Table 6: Success Rate (SR) for LLaMA and T5-XL fine-tuned on a single prompt in the TWC environment shows a similar trend of prompt overfitting, even in larger models.

P_0:
Possible actions of the agent: close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table
Goal: clean the Kitchen
Observation: You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.
Inventory: You are carrying nothing.
Next action of the agent:

P_1:
Goal: clean the Kitchen
Inventory: You are carrying nothing.
Observation: You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.
Possible actions of the agent:'close fridge', 'close kitchen cupboard', 'close oven', 'take bottle of cold water from kitchen cupboard', 'take clean mug from dining table'
Next action of the agent:

P_2:
<Begin Possible actions> close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table **<End Possible actions>**
<Begin Goal>clean the Kitchen **<End Goal>**
<Begin Observation> You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.**<End Observation>**
<Begin Inventory> You are carrying nothing. **<End Inventory>**
Next action :

P_3 :
Welcome to TextWorld! You find yourself in a messy house. Many things are not in their usual location. Let's clean up this place. After you'll have done, this little house is going to be spick and span! Look for anything that is out of place and put it away in its proper location. What you can do is to close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table. Your goal is to clean the Kitchen. You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.. Now, You are carrying nothing., and your next action is to

P_4:
{Context: 'Welcome to TextWorld! You find yourself in a messy house. Many things are not in their usual location. Let's clean up this place. After you'll have done, this little house is going to be spick and span! Look for anything that is out of place and put it away in its proper location.'
What you need to do: clean the Kitchen
What you see: 'You can see a fridge. Empty! You can see an opened kitchen cupboard. The kitchen cupboard contains a bottle of cold water. Oh, great. Here's an oven. The oven is empty, You lean against the wall, inadvertently pressing a secret button. The wall opens up to reveal a dining table. On the dining table you see a clean mug.'
What are you carrying:
What you can do: "close fridge, close kitchen cupboard, close oven, take bottle of cold water from kitchen cupboard, take clean mug from dining table"
Next action :}

Figure 7: Example of a state described using different prompt formulations in TWC-Medium.

P_0:
Possible actions of the agent: turn left, turn right, go forward, pick up, drop,toggle
Goal of the agent: go to the purple box
Observation: You see a wall 2 steps forward, You see a purple box 2 steps left, You see a purple ball 1 step right and 1 step forward, You
see a grey key 2 steps right
Next action :

P_1:
Goal of the agent: go to the purple box
Possible actions of the agent: turn left, turn right, go forward, pick up, drop, toggle
Observation: You see a wall 2 steps forward, You see a purple box 2 steps left, You see a purple ball 1 step right and 1 step forward, You
see a grey key 2 steps right
Next action :

P_2:
<Begin.Possible actions>turn left, turn right, go forward, pick up, drop, toggle **<End Possible actions>**
<Begin Goal> go to a grey box**<End Goal>**
<Begin Current Observation>
Observation: You see a wall 3 steps forward, You see a wall 2 steps left, You see a grey ball 1 step right and 1 step forward, You see a grey box 2 steps right and 1 step forward, You see a grey box 3 steps right and 1 step forward**<End Current Observation>**
Next action :

P_3 :
you are on a maze and you have to solve a task, what you can do is: turn left, turn right, go forward, pick up, drop, toggle your task is to go to a grey box, what you see now: You see a wall 3 steps forward, You see a wall 2 steps left, You see a grey ball 1 step right and 1 step forward, You see a grey box 2 steps right and 1 step forward, You see a grey box 3 steps right and 1 step forward and you next action is to

Figure 8: Example of a state described using different prompt formulations in BabyAi-Text.

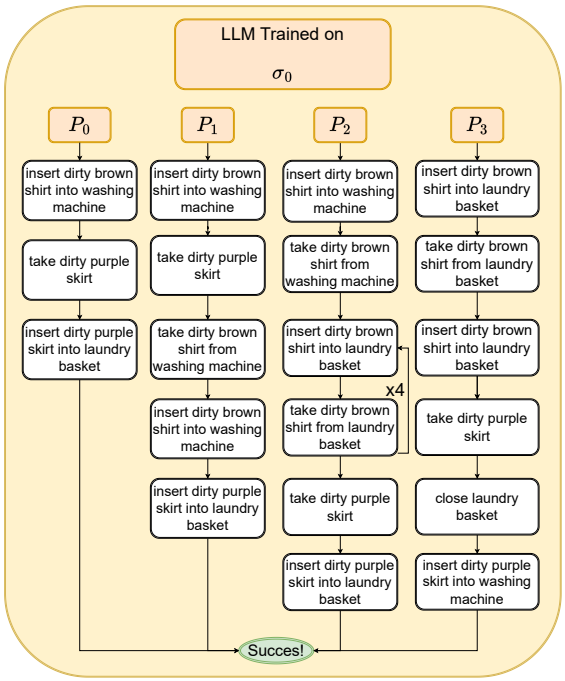

Figure 10: **Trajectories of the 780M Agent Trained with σ_0** and queried using prompts P_0 to P_0: each vertical line represents a trajectory with the prompt formatted using a specific formulation P, and each step denotes the successive actions taken by the LLM until the goal is achieved.

Indeed, fine-tuning with a single prompt formulation yields good results when evaluated on that same prompt, but there is a significant drop in per-

formance when evaluated on different prompt formulations. Another key point is the episode length of successful episodes when changing prompt formulations. Figure 10 shows an example of four trajectories of the 780 model trained with σ_0 and evaluated across P_0, P_1, P_2 andP_3. The minimal number of actions is observed when formatted with the formulation used during training, i.e. P_0. By contrast, changing the prompt formulation results in irrelevant actions. For instance, when formatted with P_2, the LLM loops four times between two actions before moving to the correct actions to achieve the goal. This further corroborates the impact of prompt overfitting.

B.2 Q2: State representation

While Table 1 displays the mean intra and inter-prompt similarities for LLMs in σ_{zs}, σ_0, $\sigma_{0:3}$, and with $\sigma_0^{0:3}$, Table 7 provides an individual breakdown of these similarities.

Models	similarity	σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
78M	$Intra(P_0)$	0.988	0.987	0.987	0.938
	$Intra(P_1)$	0.989	0.988	0.988	0.834
	$Intra(P_2)$	0.999	0.999	0.999	0.999
	$Intra(P_3)$	0.993	0.992	0.993	0.858
	$Inter(P_0, P_1)$	0.384	0.388	0.390	0.853
	$Inter(P_0, P_2)$	0.319	0.310	0.295	0.726
	$Inter(P_0, P_3)$	0.427	0.448	0.429	0.823
780M	$Intra(P_0)$	0.998	0.987	0.998	0.938
	$Intra(P_1)$	0.998	0.988	0.997	0.834
	$Intra(P_2)$	0.999	0.999	0.999	0.999
	$Intra(P_3)$	0.999	0.992	0.999	0.858
	$Inter(P_0.P_1)$	0.623	0.388	0.624	0.853
	$Inter(P_0.P_2)$	0.165	0.310	0.164	0.726
	$Inter(P_0.P_3)$	0.619	0.448	0.622	0.823

Table 7: **Detailed inter and intra-similarity for Flan T5 78M and 780M models** on TWC-Medium. We observe the same clustering behavior depending on the prompt formulation when analyzing each prompt independently.

Besides, we further examine the UMAP visualization of models trained with σ_0, σ_1, σ_2 and σ_3, for TWC-Medium to observe potential cluster separation based on prompt formulation rather than semantic content. Results in Figure 11 reveal distinct clusters corresponding to different prompt formulations, aligning with the findings presented in Table 7. Similarly, UMAP visualization with the 78M model in BabyAI-Text (see Figure 12) underscores the persistence of prompt overfitting across different model sizes and environments. Notably, the clustering tendency appears more pronounced in BabyAI-Text, which may elucidate the challenges encountered in implementing the contrastive solution in the 78M model variant on BabyAI-Text.

B.3 Q3: Relevance of Prompt Information in Decision-Making

For evaluating the relevance of subparts information of the input prompt, we assess models fine-tuned with σ_0, σ_1, σ_2 and σ_3. Results in Figure 13, show a variability in the importance of prompt parts across different prompt formulations, even after fine-tuning the LLM.

Figure 14 summarizes outcomes based on two types of prompt formulations: when the formulation P_0 used during fine-tuning is equal to the evaluation formulation P_j ($P_i = P_j$) and when σ_0 differs from P_j.

Notably, the contrastive regularization approach is more homogeneous, indicating consistent behavior of the LLM across various prompt formulations.

C Mitigating Overfitting with Contrastive Learning Regularization

In this section, we clarify the rationale behind the choice of the regularization token for applying the contrastive method and its adaptation to both encoder-decoder and decoder-only architectures. Two pooling methods can be employed: mean pooling of token embeddings or using the first token embedding. We conducted an evaluation (see Table 8) on both methods and observed that applying contrastive regularization to the mean embeddings does not significantly affect performance, whereas using the first token embedding yields better regularization. We interpret this as the contrastive regularization applied to the first token influencing the embeddings of other tokens through attention mechanisms.

	P_0	P_1	P_2	P_3
Contrastive Mean Token	0.85	0.71	0.82	0.66
Contrastive First Token	**0.97**	**0.91**	**0.95**	**0.93**

Table 8: Comparison of the effect of contrastive regularization applied to the first token versus mean token embeddings shows that using the first token provides better regularization compared to mean token embeddings.

For encoder-decoder architectures, we apply regularization to the encoding of the first token in the output of the encoders, following the work of (Ni

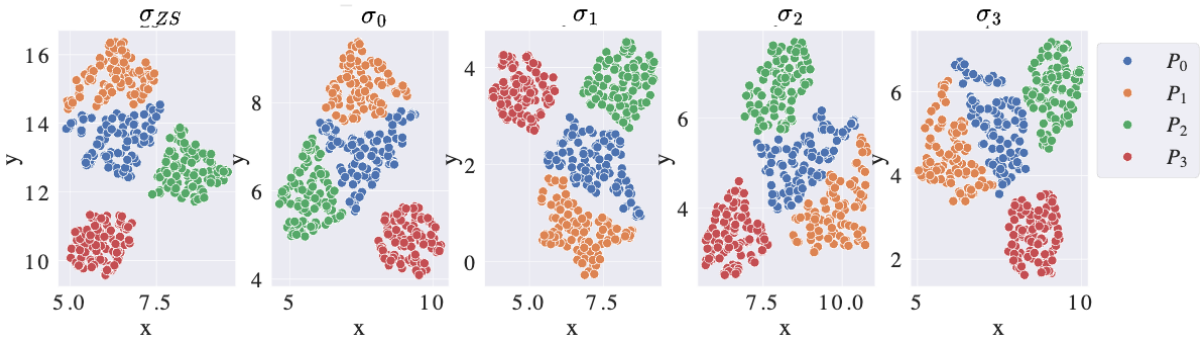

Figure 11: **UMAP visualization of the hidden representations of Flan T5 780M** with σ_{zs} and fine-tuning with σ_0, σ_1, σ_2 and σ_3 on TWC-Medium.

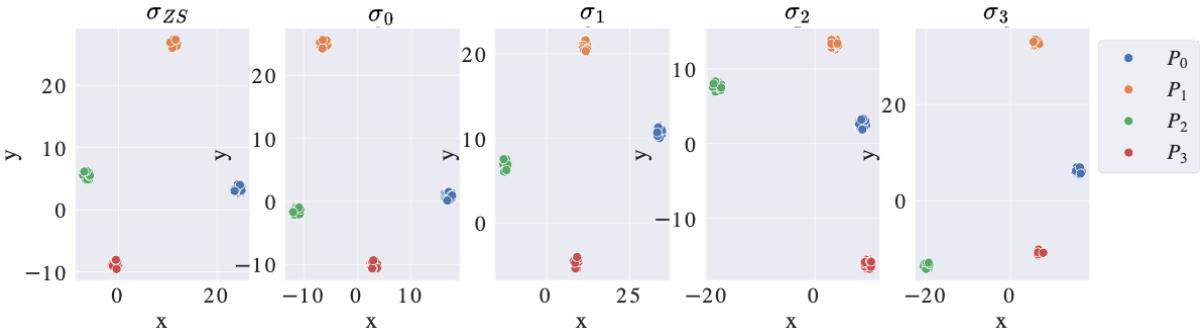

Figure 12: **UMAP visualization of the hidden representations of Flan T5 78M** with σ_{zs} and fine-tuning with P_0, P_1, P_2 and P_3 on BabyAI-Text.

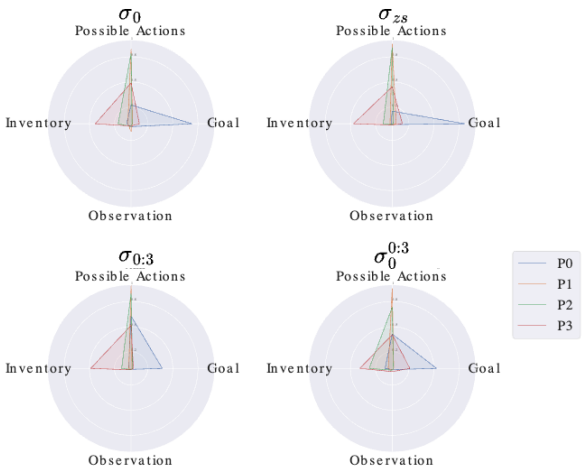

Figure 13: **Complementary results of Saliency maps** of different parts of prompts for fine-tuned T5 78M models on σ_0 to σ_3.

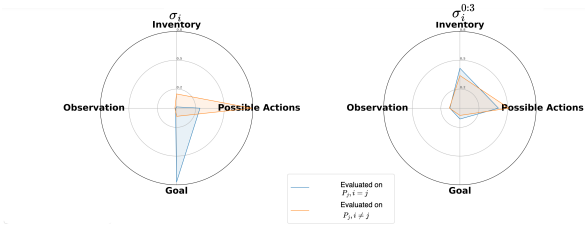

Figure 14: **Saliency maps** when $P_i = P_j$ and when $P_i \neq P_j$.

et al., 2022). In contrast, for decoder-only architectures, the causal nature of these networks means that directly applying contrastive regularization to the first token can diminish the contrastive loss, as the first token does not have access to future tokens. To address this issue, we introduce a new token, <contrastive>, positioned at the beginning of the network, which allows bidirectional attention exclusively for this token. This approach enables the <contrastive> token to access the entire prompt, thereby influencing the embeddings of future tokens. Figure 16 summarizes the method. Additionally, we examined which layers are most effective for applying contrastive regularization. Empirical tests conducted on the first five layers of the

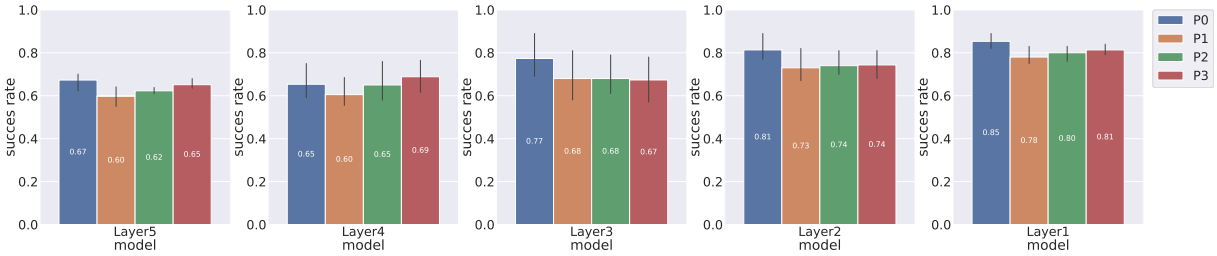

Figure 15: Comparison of different layer choices for applying contrastive regularization on GPT-Neo 1.3B reveals that earlier layers provide better performance. In contrast, performance declines as one moves toward the later layers of the network.

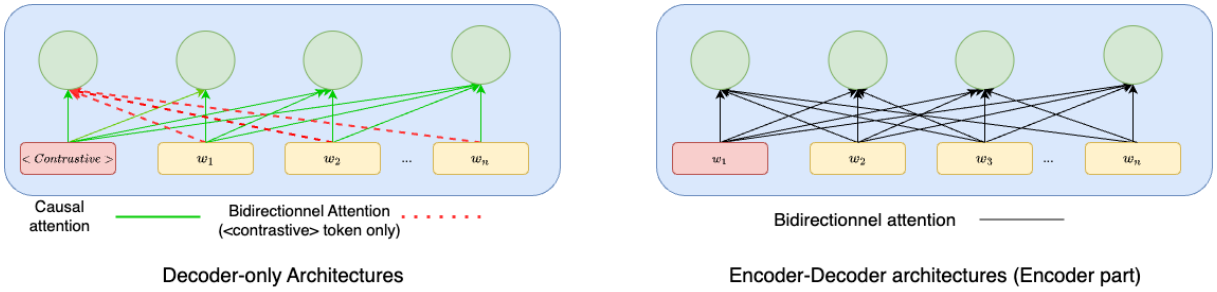

Figure 16: Selection of a regularization token for encoder-decoder and decoder-only architectures. The regularization token is highlighted in red. In decoder-only architectures, bidirectional attention is permitted for the regularization token, enabling it to access the entire prompt and encode the semantics.

networks evaluated the impact of contrastive regularization on performance. The results, presented in Figure 15, indicate that applying the regularization at the beginning of the network yields superior performance, while subsequent layers can still be utilized for task learning with the regularization applied. Additionally, we found in our experiment, that setting $\alpha = 0.5$ in Equation 2 provides a better balance between the PPO loss and the contrastive loss.

Table 9 provides complementary results on the differences between training on a single prompt formulation, all prompts, and the contrastive learning scenario, by displaying the mean SR and chi-squared results for the different models and training scenarios.

ENV	Model	Metrics	σ_{zs}	σ_0	$\sigma_{0:3}$	$\sigma_0^{0:3}$
TWC	78 M	SR	0.28 $\pm$0.13	0.80 $\pm$ 0.12	0.88 $\pm$ 0.04	0.94 $\pm$0.03
		χ^2	7x10^{-}4 $\pm$6,6x10^{-4}	1,49x10^{-}4 $\pm$2x10^{-4}	0.99 $\pm$0.01	0.99 $\pm$0.01
	780 M	SR	0.38 $\pm$ 0.03	0.83 $\pm$ 0.11	0.87 $\pm$ 0.03	0.89 $\pm$ 0.05
		χ^2	0.99 $\pm$0.01	4,5x10^{-2} $\pm$6$\times$10^{-3}	0.99 $\pm$0.01	0.98 $\pm$0.01
BabyAI	78 M	SR	0.25 $\pm$ 0.13	0.38 $\pm$ 0.23	0.49 $\pm$ 0.17	0.52 $\pm$ 0.21
		χ^2	1x10^{-4} $\pm$10^{-4}	3,09x10^{-4} $\pm$2x10^{-4}	0.531 $\pm$0.21	3,1x10^{-3} $\pm$4x10^{-3}
	780 M	SR	0.41 $\pm$ 0.01	0.63 $\pm$ 0.24	0.85 $\pm$ 0.12	0.82 $\pm$ 0.12
		χ^2	0.99 $\pm$0.01	1,5x10^{-3} $\pm$2,1x10^{-4}	0.99 $\pm$0.01	0.97 $\pm$0.02

Table 9: **Mean success rate and chi-squared (χ^2) p-value for zero-shot models**, models trained on a single prompt σ_0, models trained on all prompts ($\sigma_{0:3}$, and models trained with contrastive regularization.

D Training costs

We trained the Flan T5 780M and 2.7B, the GPT-Neo 1.3B and the Llama, with eight NVIDIA A100 80GB GPUs, with each LLM instance distributed on one GPU. For the 78M models, we employed four NVIDIA V100 32GB GPUs. Training was conducted with five different seeds in each scenario and environment. In total, our experiments required 10800 GPU hours on A100 80GB and 6400 GPU hours on V100 32GB.