machine learning & knowledge extraction

MDPI

Article

Large Language Models for Structured Task Decomposition in Reinforcement Learning Problems with Sparse Rewards

Unai Ruiz-Gonzalez [1,2,*], Alain Andres [2,3] and Javier Del Ser [2,4,*]

[1] Department of Communications Engineering, University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
[2] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain; alain.andres@tecnalia.com
[3] Faculty of Engineering, University of Deusto, 20012 Donostia, Spain
[4] Department of Mathematics, University of the Basque Country (UPV/EHU), 48940 Leioa, Spain
* Correspondence: unai.ruiz@ehu.eus (U.R.-G.); javier.delser@ehu.eus (J.D.S.)

Abstract

Reinforcement learning (RL) agents face significant challenges in sparse-reward environments, as insufficient exploration of the state space can result in inefficient training or incomplete policy learning. To address this challenge, this work proposes a teacher–student framework for RL that leverages the inherent knowledge of large language models (LLMs) to decompose complex tasks into manageable subgoals. The capabilities of LLMs to comprehend problem structure and objectives, based on textual descriptions, can be harnessed to generate subgoals, similar to the guidance a human supervisor would provide. For this purpose, we introduce the following three subgoal types: positional, representation-based, and language-based. Moreover, we propose an LLM surrogate model to reduce computational overhead and demonstrate that the supervisor can be decoupled once the policy has been learned, further lowering computational costs. Under this framework, we evaluate the performance of three open-source LLMs (namely, Llama, DeepSeek, and Qwen). Furthermore, we assess our teacher–student framework on the MiniGrid benchmark—a collection of procedurally generated environments that demand generalization to previously unseen tasks. Experimental results indicate that our teacher–student framework facilitates more efficient learning and encourages enhanced exploration in complex tasks, resulting in faster training convergence and outperforming recent teacher–student methods designed for sparse-reward environments.

Keywords: goal-oriented reinforcement learning; teacher–student; sparse-reward environments

Academic Editor: Danial Javaheri

Received: 9 September 2025
Revised: 8 October 2025
Accepted: 17 October 2025
Published: 22 October 2025

Citation: Ruiz-Gonzalez, U.; Andres, A.; Del Ser, J. Large Language Models for Structured Task Decomposition in Reinforcement Learning Problems with Sparse Rewards. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 126. https://doi.org/10.3390/make7040126

1. Introduction

Imagine trying to master a complex new skill without any guidance or feedback until the moment the effort is fully completed. No indication of progress, no suggestions for improvement; only endless trial and error. This is the challenge of reinforcement learning (RL) agents in sparse-reward environments. The lack of intermediate guidance makes learning inefficient and exploration highly resource-intensive [1,2]. In such settings, traditional RL approaches relying on random exploration often fail to scale, especially when tasks involve multiple stages or dependencies.

To address these challenges, researchers have explored structured learning paradigms, such as curriculum learning (CL) [3], imitation learning [4], hierarchical reinforcement learning (HRL) [5], and teacher–student frameworks [6]. While CL provides a progressive

learning path and HRL decomposes tasks into subtasks, both approaches face limitations in adaptability and dynamic feedback generation. The teacher–student paradigm, in contrast, introduces a guiding entity capable of tailoring subgoals to the agent's current capabilities.

Recent advances in large language models (LLMs) offer a promising avenue for implementing such dynamic teachers [7]. LLMs possess the ability to interpret textual descriptions, develop task structures, and generate context-aware subgoals. This capability aligns with findings in natural language processing, where LLMs improve inferential reasoning when guided by rationales [8]. Similarly, LLMs can enhance exploration and generalization in RL by providing structured, human-like guidance.

Motivated by these capabilities, this study proposes a novel teacher–student RL framework that leverages LLMs as expert teachers, guiding agents in sparse-reward environments by decomposing tasks into different types of subgoals. While prior work has explored the potential of integrating LLMs into RL, existing approaches often fall short because they rely heavily on repeated LLM queries, which are computationally expensive and introduce significant latency. Moreover, prior work often focuses on generating subgoals that broaden the agent's knowledge or facilitate exploration, rather than explicitly subdividing a single task into ordered steps that drive the agent towards successful completion (Figure 1). To address these limitations, we build upon our preliminary findings presented in [9] to introduce a scalable surrogate model that approximates the LLM's behavior while avoiding overfitting to oracle-like supervision. In addition, we study how subgoals can be gradually decoupled, allowing the agent to benefit from them during training but ultimately operating at deployment without any reliance on explicit subgoals or LLM queries.

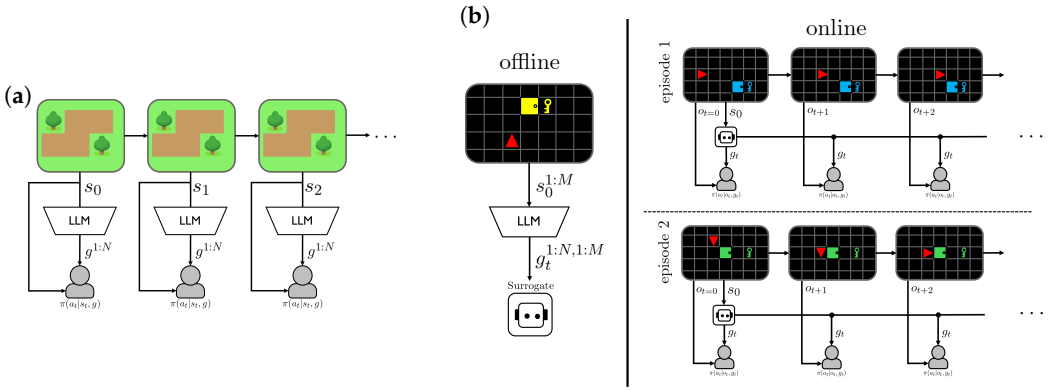


Figure 1. Comparison of RL frameworks guided by LLMs. (**a**) An LLM-based RL approach in which, at each timestep, the agent receives N non-sequential subgoals directly from the LLM. (**b**) Our approach is as follows: first (**left side**), we collect data over M episodes, where the LLM predicts N subgoals per episode. These LLM-generated subgoal sequences are used to train a surrogate model that approximates the LLM's behavior. Then (**right side**), during agent training, we use the surrogate to predict the entire sequence of task-relevant subgoals at the start of each episode.

We evaluate our framework using three open-source LLMs (Llama, DeepSeek, and Qwen) across procedurally generated environments from the MiniGrid benchmark [10]. Our results demonstrate that LLM-guided subgoal generation accelerates learning, improves exploration, and enhances generalization to unseen tasks. The results show that LLM decomposition facilitates faster convergence, enabling our method to solve the evaluated environments in a fraction of the training steps required by other approaches [11,12], achieving over 90% improvement.

In summary, our contributions are as follows:

- We perform a detailed evaluation of different LLMs, identifying the most effective models for the teacher role in our proposed framework.

- We identify the most effective subgoal encoding that enables agents to perceive and act on them.
- We develop a scalable method for LLM integration, reducing computational costs and ensuring practical applicability in real-world settings.

The rest of the manuscript is structured as follows. In Section 2, we examine the related literature, providing an overview of existing approaches in the field. Section 3 outlines the proposed framework and methodology, including details of our approach. Section 4 introduces the experimental setup and Section 5 presents the obtained results, followed by a discussion of our findings. Section 6 concludes the paper and outlines future research directions.

2. Related Work

In this section, we review existing research and methodologies relevant to our study. We begin by exploring CL (Section 2.1), emphasizing the benefits of a well-structured learning process. Next, we discuss HRL (Section 2.2), which focuses on decomposing complex RL problems into more manageable subtasks. Finally, we analyze different strategies for integrating LLMs into the RL training process (Section 2.3).

2.1. Curriculum Learning

Organizing learning tasks in a structured sequence has emerged as an effective strategy in training agents. Originally conceived as "training from easy to hard", CL has attracted attention due to its potential to expedite learning processes [11,13]. However, its efficacy is not universally consistent across implementations, where the curriculum—i.e., the structured sequence of tasks, environments, or goals presented to the agent in increasing order of difficulty to facilitate its learning—must be well-defined in order to be effective [14–16], suggesting that its application requires careful considerations.

Implementing CL in RL, therefore, requires deliberate design choices, often relying on teachers to guide the learning process. Randomized task sequences offered initial improvements over unguided training, but dynamically chosen tasks by a teacher helped maintain previously acquired skills [14,17]. Beyond these, to make curricula more adaptive, a buffer of high-potential learning tasks can be used, or tasks can be monitored for changes in success rates, ensuring training is concentrated where learning is most active [18,19].

Introducing subgoals allowed complex tasks to be decomposed into manageable objectives, while the teacher adaptively adjusted subgoal difficulty over time, effectively creating a curriculum tailored to the agent's knowledge [20]. Adversarial frameworks complemented this process by evaluating the effectiveness of these subgoals without relying on environment rewards [21,22]. Subsequent work extended this idea by expressing subgoals in natural language, which not only provided richer guidance but also made the learning process interpretable to human observers [12].

2.2. Hierarchical Reinforcement Learning

HRL leverages structured abstractions to decompose complex tasks into manageable subgoals, improving generalization and sample efficiency [5,23]. Among these abstractions, language has proven to be especially powerful: high-level policies can use linguistic instructions to guide low-level controllers, enabling both flexible skill reuse across diverse tasks and improved interpretability [24]. Policies for these subgoals are typically trained from demonstrations, feedback, or language, allowing high-level planners to orchestrate them with structured commands [25,26]. Building on this paradigm, recent work integrates LLMs into HRL, where a high-level RL policy leverages LLMs to generate intermediate

subgoals for the low-level agent, further enhancing the adaptability and scalability of its learning process [27,28].

2.3. LLM-Enhanced RL

Leveraging the power of LLMs has become an emerging strategy in the field of RL, exploiting the capabilities of natural language processing for RL frameworks. Models like the proprietary GPT series [29–31] or open-source alternatives such as Llama [32], Qwen [33], and DeepSeek [34] are nowadays being harnessed to guide RL agents.

Recent surveys in the field [7,35] have provided substantial insights into the diverse capabilities of LLMs, highlighting various studies that integrate them into RL frameworks. One prominent approach involves preprocessing RL environments into textual representations, including goals, histories, and observations, allowing LLMs to predict actions either in a zero-shot manner [36] or through fine-tuning on these textual environments [37]. Beyond acting as decision-makers, LLMs can play a central role in shaping reward functions, leveraging their ability to reason. By interpreting user-provided task descriptions, they can generate reward signals that reflect desired behaviors [38,39]. They can also elicit preferences over candidate behaviors to iteratively refine reward functions, guiding agents toward the acquisition of complex skills [40,41]. In addition, LLMs can serve as powerful information processors, offering new ways to simplify tasks. They can decompose tasks into smaller steps, suggest meaningful goals, and update plans based on outcomes, enabling agents to act effectively in diverse environments [42–44]. LLMs can also evaluate potential actions, combining natural language descriptions with value estimates to select the most promising behaviors [45], or leverage multimodal inputs and memory to generate and execute comprehensive plans, achieving state-of-the-art performance [46].

2.4. Contribution

Our approach leverages a pretrained teacher to generate subgoals, avoiding the need for the teacher to learn subgoal generation from scratch [12,21,27]. In contrast to methods that focus on diverse, non-sequential skills [43], we concentrate on sequential, task-specific subgoals that guide the agent through a structured learning sequence. Beyond simply assessing whether the LLM can generate subgoals [44], we examine their impact on RL training, including how the integration of these subgoals into the agent's observations influences learning performance. Table 1 summarizes prior subgoal generation methods and highlights key differences in sequentiality, task-specificity, LLM usage, and RL integration. As shown in this table, our work differs from previous approaches in that we employ a LLM to train a surrogate model that acts as a teacher to decompose episodic, time-sensitive tasks, thereby enabling more efficient agent training and guiding learning through structured, sequential subgoals.

Table 1. Comparison of prior subgoal generation methods and our approach. Columns indicate (1) whether subgoals are sequential and task-specific; (2) whether an LLM is used as a teacher; (3) whether the method trains an RL agent; and (4) the main contribution.

Method	(1)	(2)	(3)	(4)
AMIGO [21]	✗	✗	✓	Trains an adversarial teacher to generate subgoals that guide the agent to explore the state space efficiently.
LAMIGO [12]	✗	✗	✓	Extends AMIGO by representing subgoals as natural language instructions, enabling text-based guidance.
LLMxHRL [27]	✓	✗	✓	Leverages an LLM to provide commonsense priors, which guide a hierarchical agent in generating and interpreting subgoals.

Table 1. *Cont.*

Method	(1)	(2)	(3)	(4)
ELLM [43]	✗	✓	✓	Uses LLM to propose non-sequential, diverse skills, without task-specific sequencing.
CALM [44]	✓	✓	✗	LLM to decompose tasks into subgoals, evaluating only the accuracy of proposed subgoals without assessing RL performance.
Ours	✓	✓	✓	Generates sequential, task-specific subgoals with surrogate models from LLMs, guiding agents while reducing computational cost.

3. Proposed Framework and Methodology

Our framework integrates an LLM as an omniscient teacher that supervises an RL agent through sequences of subgoals. At the start of each episode, the LLM, with full access to the initial environment state $s_0 \in \mathcal{S}$ and the global objective g^{obj}, generates a fixed sequence of N subgoals $g_0, g_1, \ldots, g_{N-1}$. These subgoals are designed to be unique ($g_{i+1} \neq g_i$) and progression-dependent, ensuring the agent advances toward the global goal g^{obj}. During interaction, the RL agent only observes partial states o_t, and at each time step it receives the triplet (o_t, r_t, g_i), where r_t is the environment reward and g_i is the current subgoal provided by the teacher. Once a subgoal is achieved, such as reaching or interacting with the specified target, the agent immediately receives the next subgoal g_{i+1}. The agent thus learns a student policy $\pi_{student}$ that maps (o_t, g_i) to actions, while the LLM provides the teacher policy $\pi_{teacher}$ by generating subgoals. The teacher remains static throughout training, serving as a source of prior knowledge, while the student adapts by learning how to effectively use the subgoals to achieve the global objective.

In this section, we first outline the methodology for generating subgoals (Section 3.1) and defining their representations (Section 3.2). Then, we describe our approach to prompt engineering, optimizing LLM performance for effective subgoal decomposition. Finally, we introduce a surrogate model that approximates LLM-generated subgoals, enabling efficient training without the need for repeated LLM queries (Section 3.3).

3.1. Subgoal Generation

Subgoal generation takes place at the beginning of each episode, where the LLM, with the complete environment state s_0, creates a sequence of subgoals $\{g_0, g_1, \ldots, g_N\}$, which progressively guide the agent toward the final goal of the RL task. These subgoals reduce the reward horizon by decomposing the problem into intermediate steps, thereby facilitating and shaping the learning process. In doing so, our approach constitutes an intermediate setup between sparse-reward cases (where feedback is limited to task completion) and dense-reward cases (where feedback is provided at a high frequency, e.g., at every step).

To incentivize both subgoal achievement and task completion, we augment the extrinsic reward r_t^e with an auxiliary subgoal reward r_t^g. The total reward at each time step is defined as $r_t = r_t^e + \alpha \cdot r_t^g$, where $\alpha \in \mathbb{R}^+$ balances the emphasis on subgoal completion versus overall task success. The subgoal reward r^g is structured to encourage efficient progress. It is computed as $r^g = 1 - \frac{t^g}{t_{\max}}$, where t^g is the time step at which subgoal g is achieved, and $t_{\max}$, which depends on the environment, is the maximum number of steps allowed in an episode and must be chosen carefully as it can strongly impact learning. Using this formulation, we penalize excessive time spent on individual subgoals while rewarding timely progression.

3.2. Subgoal Representation

The form in which subgoals are presented plays a crucial role in how effectively the agent can use them. In our framework, we consider the following three distinct encoding strategies to represent teacher-provided subgoals:

1. Position-based subgoals are defined in terms of coordinates provided by the environment (e.g., $\{x, y\}$ in 2D or $\{x, y, z\}$ in 3D). In discrete domains, these subgoals represent a point in space, whereas in continuous domains they denote a target region. While intuitive, this approach assumes the agent can infer directional relationships and may suffer from misalignment when the LLM inaccurately predicts object positions.
2. Representation-based subgoals are defined in terms of identifiable components or features of the environment's state, which the agent perceives through its observations o_t. In object-based domains, this could correspond to the representation of a specific object, while in image-based domains, it could correspond to an element detected through perception modules. This approach can be limited in novel or unseen scenarios since subgoals are tightly linked to predefined state representations.
3. Language-based subgoals are derived from LLM-generated text, which is especially effective when the task cannot be directly expressed as an element of the environment. The text is then converted into embeddings using a pretrained language encoder, allowing the subgoals to be represented in a continuous vector space. To reduce the complexity of handling diverse natural language instructions, we first construct a fixed pool of environment-relevant subgoals. This pool is generated by enumerating all objects and their attributes and combining them with a set of predefined actions. Each resulting subgoal phrase is then converted into a vector embedding. When a new instruction is received, it is similarly converted into an embedding and matched to the closest canonical subgoal in the pool using cosine similarity.

Visual examples of these subgoal representations and the language subgoal pool for the RL environments considered in this work can be found in Appendix D.

3.3. Modeling LLM–Surrogate Subgoal Generation

While LLMs excel at decomposing tasks into subgoals, their direct integration into RL training poses scalability challenges. Many RL problems require training across hundreds of thousands of episodes, making real-time subgoal generation with LLMs computationally and economically impractical. For instance, modern environments can run at 1000–3000 frames per second, making a 1000-step task complete at roughly 1–3 Hz [47]. In contrast, end-to-end LLM queries might run at 0.2–0.5 queries per second [48]. This gap means that using an LLM inside the training loop would slow data collection by one to two orders of magnitude. Moreover, continuously running LLMs is often impractical due to resource constraints, making RL training both costly and unstable.

Beyond scalability and high resource demands, another key challenge arises when agents are trained with perfect subgoals but must be deployed under imperfect guidance. Training under flawless supervision, whether from a hypothetical perfect LLM or manually specified subgoals, creates a distribution mismatch: at deployment, agents encounter noisy or incomplete subgoals, rather than the ideal guidance seen during training. This can lead to overfitting, where agents rely on unrealistic guidance and fail to generalize properly when deployed. Moreover, generating perfect supervision through LLM prompting for every training step is costly and infeasible at the scale required for many RL tasks.

To address this issue, we introduce an offline surrogate framework (Algorithm 1) that approximates LLM-guided subgoal generation while avoiding repeated LLM queries during training. The surrogate is designed to mimic the variability of real LLM outputs, allowing agents to train under deployment-like conditions and improving robustness when

oracle supervision is no longer available. Results in Appendix B confirm the effectiveness of this strategy for the RL environment later described in the experiments of this manuscript.

To take this idea to practice, we sample M levels from the environment's level distribution and collect a set of subgoal sequences by prompting the LLM. Using an oracle, we label the optimal subgoal sequences as ground truth—the only supervision needed—and if no oracle is available, a human could provide the annotations. Finally, we statistically characterize the deviations of the LLM proposals from the oracle decompositions to construct a surrogate model, focusing on two error types:

1. *Positional errors*: For position-based subgoals, we measure the distance between the LLMs' proposed coordinates G^m and the oracle's optimal coordinates O^m.
2. *Semantic errors*: For representation-based and language-based subgoals, we evaluate mismatches in object type, state, or sequence.

Algorithm 1 Surrogate LLM framework

M: The number of levels considered in the evaluation.
$\mathcal{G}^m$: The set of subgoal proposals at iteration m.
Ω^m: The set of optimal subgoals at iteration m.
$E = \{e_0, e_1, \ldots, e_N\}$: The set of all errors accumulated over all levels.
ϵ: The accuracy for the LLM, computed based on E.
$\widehat{S}$: The surrogate model, created based on the accuracies of the subgoals.
// Accuracy evaluation
for $m = 1$ to M **do**
    Prompt the LLM for subgoal proposals $\mathcal{G}^m \leftarrow \text{LLM}(L^m, s_0)$
    Label the optimal subgoals using the oracle $\Omega^m \leftarrow \{\omega_0^m, \omega_1^m, \ldots, \omega_N^m\}$
    for $i = 0$ to N **do**
        $(g_i^m, \omega_i^m) \leftarrow (\mathcal{G}^m[i], \Omega^m[i])$
        $$e_i^m = \begin{cases} \text{Manhattan distance}(g_i^m, \omega_i^m), & \text{if position-based,} \\ \mathbb{1}_{\text{match}}(g_i^m, \omega_i^m), & \text{if semantic-based.} \end{cases}$$
        Append e_i^m to E
    end for
end for
$\epsilon \leftarrow \text{ComputeMean}(E)$
// Surrogate model application
for each episode **do**
    Sample level from environment
    for each $\omega_i \in \Omega$ **do**
        $e_i' \sim \mathcal{U}(0, 1)$
        $$g_i^m = \begin{cases} \widehat{S}(\omega_i^m, \text{modify}), & e_i' < \epsilon, \\ \widehat{S}(\omega_i^m, \text{retain}), & e_i' \geq \epsilon. \end{cases}$$
    end for
    Train RL agent with subgoals from $\widehat{S}$
end for

Once trained, this surrogate can generate imperfect but realistic subgoals across the full distribution of levels, not just the M sampled ones. In doing so, it removes the scalability bottleneck of repeated LLM queries while preserving the benefits of LLM-guided task decomposition.

4. Experimental Setup

To evaluate the effectiveness of our proposed method, we conduct experiments across several challenging environments from the MiniGrid framework [10], a widely used benchmark for RL research. MiniGrid, part of the Farama Foundation ecosystem, provides

lightweight, grid-based environments that simulate navigation, object interaction, and task completion tasks. MiniGrid environments are procedurally generated, meaning that each episode presents a unique level layout while maintaining consistent task structures (e.g., retrieving keys, unlocking doors, and navigating mazes). This procedural generation ensures that agents must learn to generalize across diverse instances rather than memorizing specific layouts, making MiniGrid an ideal testbed for evaluating the robustness and adaptability of our approach. We focus on six environments of varying complexity:

- *MultiRoomN2S4* and *MultiRoomN4S5*: Navigation tasks requiring the agent to traverse multiple rooms.
- *KeyCorridorS3R3* and *KeyCorridorS6R3*: Tasks involving key retrieval and door unlocking in increasingly complex layouts.
- *ObstructedMaze1Dlh* and *ObstructedMaze1Dlhb*: Challenging maze navigation tasks with obstructed paths and objects.

We first investigate how different LLMs affect subgoal generation and, consequently, agent training. To this end, we evaluate three open-source models (Llama3-70b [32], Qwen1.5-72b-chat [33], and DeepSeek-coder-33b [34]), using zero-shot chain-of-thought reasoning [30,49,50] to generate position-, representation- and language-based subgoals (for details on prompt design, see Appendix A). For language-based subgoals, embeddings are generated using all-MiniLM-L6-v2 [51]. We also examine the effect of reward balance on agent performance, as it can influence the agent's ability to prioritize certain tasks over others. Finally, we conduct an ablation study to evaluate the impact of each subgoal component, specifically the subgoal representation in the agent's observation and the associated reward, on overall performance.

4.1. Algorithm and Model Architecture

We train agents using the PPO [52] implementation by [53] with an actor–critic architecture tailored to the grid-based observation space. The model begins with three convolutional layers, each comprising 32 filters, a 3×3 kernel, a stride of 2×2, and padding of 1, followed by ELU activations to extract spatial features. The resulting 32-dimensional representation is mapped through a shared, fully connected layer to a 256-dimensional latent vector. From this shared representation, two separate fully connected networks parameterize the policy (actor) and the value function (critic).

4.2. Hyperparameter Configuration

Table 2 summarizes the hyperparameters used in our PPO implementation. These values are based on extensive tuning for MiniGrid tasks [53], ensuring optimal performance across diverse environments.

Table 2. Hyperparameter values used for PPO in our experiments.

Hyperparameter	Description	Value
α	Subgoal magnitude	0.2
γ	Discount factor	0.99
λ_{GAE}	Generalized Advantage Estimation (GAE) factor	0.95
PPO rollout length	Number of steps to collect in each rollout per worker	256
PPO epochs	Number of epochs for each PPO update	4
Adam learning rate	Learning rate for the Adam optimizer	0.0001
Adam ϵ	Epsilon parameter for numerical stability	1×10^{-8}
PPO max gradient norm	Maximum gradient norm for clipping	0.5
PPO clip value	Clip value for PPO policy updates	0.2
Value loss coefficient	Coefficient for the value function loss	0.5
Entropy coefficient	Coefficient for the entropy term in the loss function	0.0005

4.3. Metrics

We evaluate our model's performance using distinct metrics for LLM-based subgoal generation and policy execution. On one hand, to assess the accuracy of subgoal prediction, we consider the following four metrics:

1. *Accuracy*: The ratio of correct subgoal predictions to total predicted subgoals:

$$\text{Accuracy} = \frac{\#\text{correct subgoal predictions}}{\#\text{predicted subgoals}} \tag{1}$$

2. *Correct SGs/Total*: The number of correct subgoals identified by the LLM compared to the total number of subgoals.
3. *Correct EPs/Total*: The number of complete episodes where all subgoals were correctly predicted.
4. *Manhattan distance*: Average distance between predicted and actual subgoal positions:

$$\text{Manhattan distance} = \frac{1}{M}\sum_{i=1}^{M}|x_i^{\text{pred}} - x_i^{\text{true}}| + |y_i^{\text{pred}} - y_i^{\text{true}}| \tag{2}$$

with standard deviation:

$$\sigma_{\text{Manhattan}} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(|x_i^{\text{pred}} - x_i^{\text{true}}| + |y_i^{\text{pred}} - y_i^{\text{true}}| - \text{mean}\right)^2} \tag{3}$$

A position-based subgoal is correct if the predicted position matches the object's true location; otherwise, error is measured by Manhattan distance. Representation-based subgoals are correct if the object (and color, if relevant) is accurately described. Metric computation, including data cleaning and validation, is detailed in Appendix C.

For position-based subgoals, training errors are simulated by sampling alternative grid points, including infeasible ones, to reflect LLM inaccuracies (Figure 2—right). For representation- and language-based subgoals, training errors are approximated by selecting incorrect objects to mimic typical LLM mistakes (Figure 2—left).

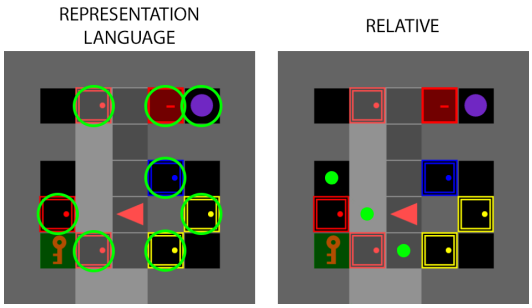


Figure 2. An episode belonging to the environment KeyCorridorS3R3 from MiniGrid, where the possible subgoals subject to the error are shown. Here, the first subgoal is assumed to be the key located at the bottom left position of the grid. (**left**) Possible subgoals for representation- or language-based subgoals, which are circled with a green circle; (**right**) possible position-based subgoals according to a Manhattan distance error of 2, indicated with a big green dot.

With respect to the policy trained via RL with LLM subgoal proposals, we measure the expected return over the following episode:

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\right] = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k (r_{t+k}^e + \alpha \cdot r_{t+k}^g)\right]. \tag{4}$$

5. Results

In this section, we present our results based on the experimentation introduced in Section 4. We begin by analyzing the performance of the surrogate LLM (Section 5.1), then evaluate RL training outcomes under this framework (Section 5.2), and conclude with an ablation study that isolates the contributions of individual subgoal components (Section 5.3).

5.1. Surrogate LLM Evaluation

We begin our experimental analysis by conducting a comprehensive evaluation by querying the aforementioned three LLMs across $M = 1000$ distinct levels in each of the six environments under consideration. These levels are sampled uniformly at random from the full environment-level distribution. Prior work [4,47] has demonstrated that $M = 1000$ levels are sufficient to capture the variability of the environment.

As summarized in Table 3, Llama consistently outperforms the other models, achieving the highest accuracy in five out of six position-based tasks and three out of six semantic-based tasks. On average, Llama's accuracy in position-based tasks exceeds Qwen's by approximately 10% and DeepSeek's by roughly 50%. For semantic-based tasks, Llama maintains a similar advantage, outperforming Qwen by about 10% and DeepSeek by around 25%. Across all models, performance tends to be higher on semantic tasks than on position-based tasks, with DeepSeek showing the largest discrepancy, losing nearly 25% accuracy on position tasks. Additionally, accuracy generally decreases in larger, more complex environments, but Llama's performance declines more gradually, indicating greater robustness and scalability.

Table 3. Performance comparison of Llama, DeepSeek, and Qwen based on position (white background) and semantic (gray background) accuracy across six environments, in terms of accuracy; the number of correct subgoals (SGs) out of the total SGs; the number of correct episodes (EPs) out of the total EPs; and the average Manhattan distance with standard deviation. The model with the highest average performance in each environment is highlighted in bold.

Environment	LLM	Accuracy	Correct SGs /Total	Correct EPs /Total	Manhattan Distance
MultiRoomN2S4 (Position)	**Llama**	**0.9210**	**1842/2000**	**859/1000**	0.26 ± 1.01
	DeepSeek	0.3310	662/2000	125/1000	2.91 ± 2.54
	Qwen	0.7135	1427/2000	581/1000	1.12 ± 2.05
MultiRoomN2S4 (Semantic)	**Llama**	**0.9295**	**1859/2000**	**871/1000**	-
	DeepSeek	0.2570	514/2000	21/1000	-
	Qwen	0.5585	1117/2000	286/1000	-
MultiRoomN4S5 (Position)	**Llama**	**0.5397**	**2159/4000**	**179/1000**	3.69 ± 4.65
	DeepSeek	0.1200	480/4000	3/1000	7.92 ± 4.64
	Qwen	0.3600	1440/4000	56/1000	5.23 ± 4.94
MultiRoomN4S5 (Semantic)	**Llama**	**0.5480**	**2192/4000**	**166/1000**	-
	DeepSeek	0.1625	650/4000	7/1000	-
	Qwen	0.3517	1407/4000	38/1000	-
KeyCorridorS3R3 (Position)	Llama	0.9743	2923/3000	**962/1000**	0.18 ± 1.20
	DeepSeek	0.3337	1001/3000	18/1000	2.57 ± 2.54
	Qwen	**0.9810**	**2943/3000**	958/1000	0.07 ± 0.63
KeyCorridorS3R3 (Semantic)	Llama	0.9847	2954/3000	963/1000	-
	DeepSeek	0.9063	2719/3000	770/1000	-
	Qwen	**0.9957**	**2987/3000**	**993/1000**	-

Table 3. *Cont.*

Environment	LLM	Accuracy	Correct SGs / Total	Correct EPs / Total	Manhattan Distance
KeyCorridorS6R3 (Position)	**Llama**	**0.9137**	**2741/3000**	**802/1000**	1.47 ± 5.02
	DeepSeek	0.3650	1095/3000	44/1000	7.24 ± 7.47
	Qwen	0.8810	2643/3000	732/1000	1.34 ± 4.25
KeyCorridorS6R3 (Semantic)	Llama	0.9323	2797/3000	806/1000	-
	DeepSeek	0.7617	2285/3000	536/1000	-
	Qwen	**0.9587**	**2876/3000**	**895/1000**	-
ObstructedMaze1Dlh (Position)	**Llama**	**0.9870**	**2961/3000**	**984/1000**	0.08 ± 0.82
	DeepSeek	0.4300	1290/3000	155/1000	2.98 ± 3.41
	Qwen	0.6687	2006/3000	486/1000	2.18 ± 3.53
ObstructedMaze1Dlh (Semantic)	**Llama**	**0.9877**	**2963/3000**	**986/1000**	-
	DeepSeek	0.7390	2217/3000	452/1000	-
	Qwen	0.7417	2225/3000	561/1000	-
ObstructedMaze1Dlhb (Position)	**Llama**	**0.4485**	**1794/4000**	0/1000	4.33 ± 4.21
	DeepSeek	0.2288	915/4000	**2/1000**	4.49 ± 3.39
	Qwen	0.4050	1620/4000	1/1000	3.85 ± 3.71
ObstructedMaze1Dlhb (Semantic)	Llama	0.4858	1943/4000	0/1000	-
	DeepSeek	**0.5182**	**2073/4000**	**14/1000**	-
	Qwen	0.4820	1928/4000	1/1000	-

5.2. Policy Training Evaluation

Building on the surrogate LLM, we next evaluate the performance of agents trained using these subgoal proposals across the six MiniGrid environments.

As shown in Figure 3, Llama consistently outperforms the other models across most environments. Agents trained with Llama-generated subgoals reach task completion significantly faster, achieving reductions in required training steps of up to $\tilde{4}5\%$ relative to DeepSeek in position-based subgoals. On the other hand, in representation-based and language-based subgoals, Llama increases the sample efficiency by 20%. Moreover, in most of the considered tasks, Llama also surpasses Qwen by 10–30% sample efficiency, depending on the environment and subgoal type. The gains are even more pronounced compared to LAMIGO and AMIGO: Llama required 90–99% fewer training steps to reach equivalent performance. This extreme difference occurs because in several environments, AMIGO and LAMIGO do not achieve task completion even after 1,000,000,000 steps, which highlights the improved learning when following the subgoal proposed by the LLMs. These results align with the findings in Table 3; higher subgoal proposal accuracy translates into more sample-efficient agent training.

Although DeepSeek generally exhibits the lowest subgoal accuracy across environments, as shown in Table 3, an intriguing exception arises in KeyCorridorS6R3 for representation- and language-based subgoals. Based on the data, DeepSeek does not always select the optimal subgoal, but in KeyCorridorS6R3, the environment's larger size means that incorrect subgoals often lie closer to the agent's current position. We hypothesize that this behavior can actually benefit learning: by offering intermediate waypoints that are easier for the agent to reach, DeepSeek effectively facilitates early-stage subgoal-following behavior. Furthermore, this mechanism is moderated by the fact that DeepSeek still proposes optimal subgoals with a non-negligible probability (approximately 75% for this environment and subgoal types, see Table 3), creating a balance between easier-to-reach subgoals and task-aligned subgoals. This combination may explain why, despite overall

lower subgoal accuracy, DeepSeek achieves the highest task performance in these specific cases. Experimental results supporting this hypothesis are detailed in Appendix E.

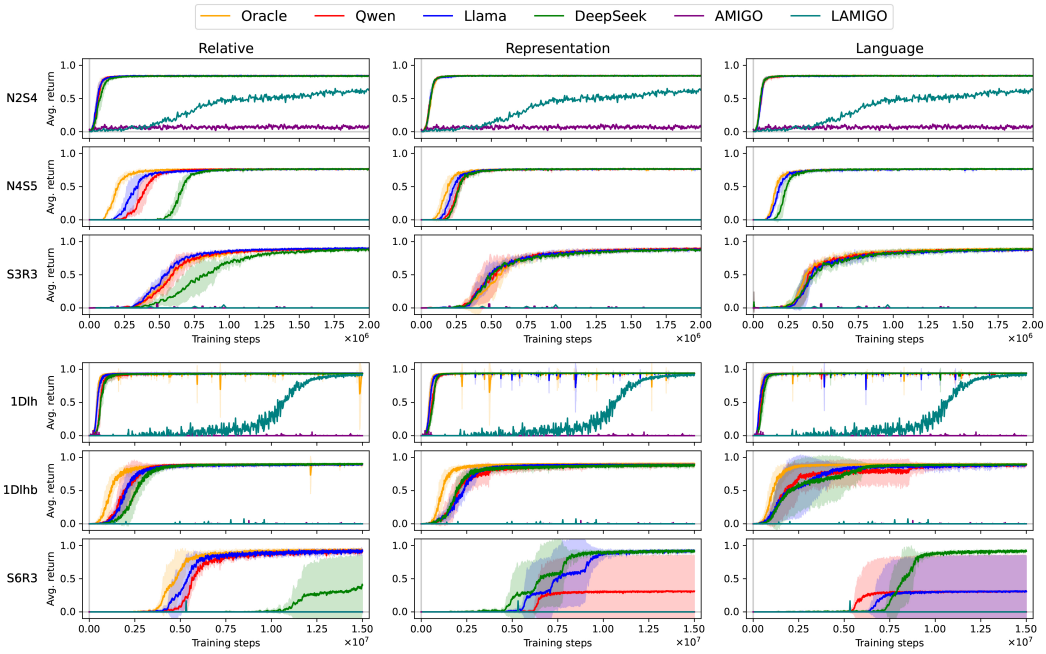


Figure 3. Training curves for the proposed LLM-assisted RL training framework, showing the average return over training steps for the six environments using three methodologies: position relative to the agent's position (**first column**), representation-based (**second column**), and language-based (**third column**). The analysis includes the three LLMs under consideration (Qwen, Llama, and DeepSeek) against optimal (oracle-provided) subgoals. Additionally, we compare our framework to AMIGO and LAMIGO to evaluate its performance against existing approaches. The shaded area represents the variability in the average return across 3 runs of the agent's training process.

To assess whether the observed differences in accuracy across LLMs reported in Table 3 are statistically significant, we conduct pairwise comparisons using a Wilcoxon signed-rank test. This non-parametric test evaluates whether two related samples differ significantly in their distributions, making it suitable for comparing algorithmic performance across multiple environments. Given the small number of environments considered (six), we report *exact* p-values rather than relying on asymptotic approximations. Additionally, we do not apply any correction for multiple comparisons, as the limited sample size would render such corrections overly conservative and potentially obscure meaningful differences.

The results of this test are summarized in Table 4 for both *position-based* and *semantic* encoding strategies. Each row compares two algorithms, reporting their average ranks (lower values indicate better performance) and the exact p-value of the test. A p-value below a significance value of 0.05 suggests a statistically meaningful difference in performance. The average ranks provide a useful summary of overall performance across environments, while the p-values quantify the confidence in these differences.

As can be concluded from this table, Llama significantly outperforms DeepSeek in position-based subgoals ($p = 0.0312$), and shows a trend toward outperforming Qwen ($p = 0.0625$). Qwen significantly outperforms DeepSeek in position-based subgoals ($p = 0.0312$). For semantic-based subgoals, none of the comparisons reach statistical significance, although Llama tends to rank better on average. These findings support the conclusion that Llama generally performs best, especially in position-based subgoal tasks, and that DeepSeek consistently underperforms relative to the other models.

Table 4. Wilcoxon signed-rank test between the accuracy scores reported in Table 3 for every pair of LLMs.

LLM 1 AvgRank (pos/Semantic)	LLM 2 AvgRank (pos/Semantic)	Exact p-Value (Position)	Exact p-Value (Semantic)
Llama (1.33/1.83)	DeepSeek (2.66/2.00)	0.0312	0.0625
Llama (1.33/1.83)	Qwen (2.00/2.16)	0.0625	0.3125
DeepSeek (2.66/2.00)	Qwen (2.00/2.16)	0.0312	0.0938

5.2.1. LLMs' Lack of Spatial Reasoning

A key insight from Table 3 results is that LLMs excel at tasks involving logical sequences (e.g., "pick up key → unlock door → retrieve ball") but struggle with tasks requiring a deep understanding of spatial hierarchies and topological dependencies among subgoals. In the MultiRoom suite, the agent must traverse a series of interconnected rooms, where access to later rooms depends on resolving spatial relationships and progression constraints. According to our experiment (Table 3), LLMs seem to lack these capabilities, as they do not seem to infer geometric or topological relationships from static state descriptions. Nevertheless, as shown in Figure 3, agent training in environments requiring spatial reasoning appears to succeed despite these limitations. We hypothesize that training succeeds in these environments because their small size limits the impact of occasional subgoal errors. Moreover, subgoals almost always correspond to doors, the only interactive objects, so the task reduces to repeatedly opening doors to progress. This simplicity ensures that even imperfect subgoal proposals provide a sufficiently useful signal, allowing the agent to learn effective behavior despite the LLMs' limited spatial reasoning.

5.2.2. Analysis of Subgoal Strategies

Table 3 compares position- and semantic-based subgoal strategies across Llama, DeepSeek, and Qwen. Overall, semantic subgoals provide better accuracies for DeepSeek (+4.2% to +57%) and moderate accuracy gains for Llama (+0.07% to +1.86%) and Qwen (+7% to +8%).

To better understand how differences in accuracy translate into learning, we analyze the training performance of position- and semantic-based (representation- and language-based) subgoals. As shown in Figure 3, the effectiveness of subgoal types depends on the reward horizon, defined as the distance from the agent's starting position to the first subgoal and between subsequent subgoals. In smaller environments like MultiRoomN2S4 and N4S5, where targets lie within or near the agent's 7×7 observation window, semantic subgoals improve over position by 6–48%, enabling direct navigation using local information. By contrast, in long-horizon environments such as KeyCorridorS6R3, semantic subgoals achieve higher accuracy (Table 3) but are less effective during training due to limited observations. Position-based subgoals provide explicit stepwise guidance, improving navigation and sample efficiency—requiring 40% fewer samples than language-based and 7% fewer than representation-based subgoals, with Llama showing 84% and 27% gains, respectively. In the case of ObstructedMaze1Dlhb, representation-based subgoals outperform position- and language-based subgoals, while language performs worst (Figure 3). We hypothesize this is because overall accuracy in this environment is low (Table 3) and language-based subgoals, being higher-dimensional, make training more difficult for the agent.

The difference between the different subgoal strategies highlights a key trade-off: for local guidance, when subgoals are close or visible within the agent's observation window,

both representation- and language-based subgoals are highly effective. Representation-based subgoals allow the agent to immediately recognize and act upon the target object, accelerating task completion. Similarly, although more complex, language-based subgoals perform almost as well as representation-based ones and provide the added benefit of abstract, human-readable guidance. On the other hand, for global guidance, when subgoals are distant, position-based subgoals become more effective. By providing relative position information, these subgoals help the agent navigate toward far-off targets, even in situations where the agent cannot directly perceive the subgoal in its field of view.

Despite these advantages, all subgoal types face challenges under partial observability. While position-based subgoals mitigate long-horizon navigation, they risk leading agents into obstacles (e.g., walls) due to static instructions. Representation-based subgoals, on the other hand, fail to guide agents when targets are outside their perceptual field. In such cases, the agent must rely on its exploratory behavior and may reach the intended subgoal just by serendipity. Similarly, language-based subgoals share limitations with representation-based subgoals. While they offer an abstract form of guidance, they can still struggle when the agent cannot directly perceive the target object or when the subgoal requires more complex spatial reasoning beyond the provided language instructions.

5.2.3. Sensitivity Analysis of Subgoal Reward Balance

The balance between subgoal rewards and the overall task reward is critical for effective training. We evaluate this balance through the reward coefficient α, which scales the subgoal reward r_t^g relative to the extrinsic task reward r_t^e. Our experiments (Figure 4) reveal that a low α value (e.g., $\alpha = 0.2$) strikes an optimal balance, ensuring that subgoals guide the agent without overshadowing the primary task objective.

The effect of α varies substantially across task suites. In the MultiRoom environments, the task is simple—traverse a few rooms and open doors—so even with low subgoal accuracy, a high α does not severely hinder training. Nevertheless, as shown in Figure 4, Llama tends to struggle with higher alpha values. On the other hand, in ObstructedMaze environments, where tasks involve more intricate dependencies, high α values amplify the impact of incorrect subgoals. This issue is evident in ObstructedMaze1Dlhb, where Llama achieves 44–48% accuracy and DeepSeek 22–52% (Table 3), making high α values impractical. By contrast, in the KeyCorridor suite, subgoal accuracy is much higher—above 90% for most models except DeepSeek (Table 3)—which allows the agent to benefit from larger α values, despite reduced sample efficiency in low alpha values (about 40% lower in KeyCorridorS6R3 with $\alpha = 0.2$). Taken together, our experiments indicate that setting $\alpha = 0.2$ provides the best balance across environments, offering sufficient guidance from subgoals without allowing subgoal errors to dominate learning, and ensuring consistent task performance even in more complex or error-prone settings.

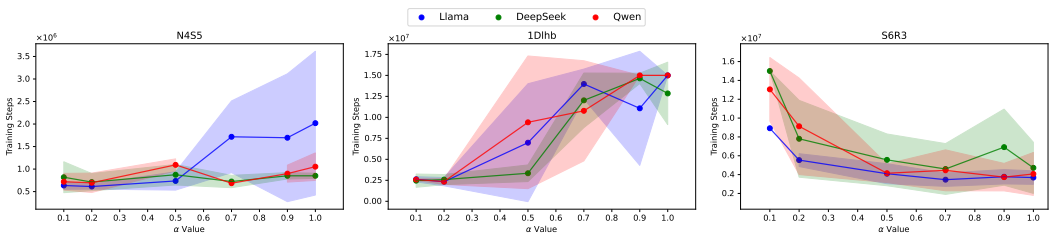


Figure 4. Average number of training steps required for agents to achieve a mean success rate exceeding 99%, aggregated across all subgoal types and environments. Each color represents the average performance of a specific LLM (Llama, DeepSeek, Qwen). Lower values indicate faster and more desirable learning.

5.3. Ablation Study: Impact of Subgoals and Rewards

To further understand the role of subgoals and rewards in agent training, we conduct an ablation study across all evaluated LLMs using representation-based subgoals (representation-based subgoals were chosen, as they demonstrated the most consistent performance across environments). We analyze two specific training conditions:

- No Reward: In this setup, agents receive representation-based subgoals in their observation, but no supplementary rewards for achieving them. Results show that agents fail to learn in environments with long reward horizons, namely, MultiRoomN4S5, KeyCorridorS3R3, KeyCorridorS6R3, and ObstructedMaze1Dlhb (Figure 5, first and third columns). This highlights the critical role of rewards in guiding exploration and reinforcing task completion, particularly in complex scenarios where subgoals alone are insufficient to bridge the reward gap. However, in environments with shorter reward horizons, such as MultiRoomN2S4 and ObstructedMaze1Dlh, agents successfully learn the task even without supplementary rewards, as the reduced complexity allows subgoals to provide sufficient guidance.
- No Subgoal: Agents receive supplementary rewards for subgoal completion but do not have access to subgoals in their observation space. Agents still perform well in this condition, achieving convergence comparable to setups where subgoals are explicitly provided (compare Figure 5, which shows training without subgoals, to Figure 3, which shows training with subgoals). For example, in ObstructedMaze1Dlh, access to subgoals leads to substantially faster convergence, with Qwen and Llama improving by roughly 20% and DeepSeek by about 12.5%. A similar trend is observed in KeyCorridorS6R3, where all models converge approximately 15% faster when subgoals are available. However, once the underlying policy has been fully acquired, subgoals in the observation space are no longer necessary for successful task completion, suggesting that incorporating subgoals into the agent's observation space primarily serves to accelerate training.

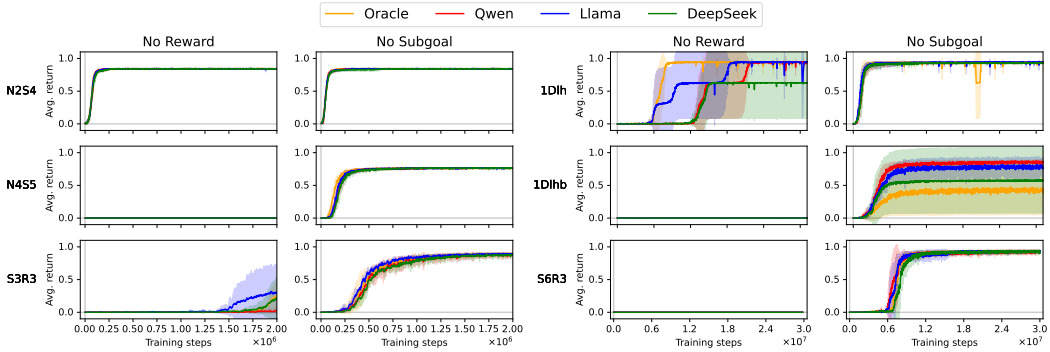


Figure 5. Performance comparison for different training scenarios using representation-based subgoals. The figure displays results for no-reward training (subplots in the first and third columns) and no-subgoal training (subplots in the second and fourth columns). The no-reward training scenarios depict the agent's performance when rewards are absent and representation subgoals are present, while the no-subgoal training condition shows the performance when subgoals are absent, but rewards are still present. Plots of each training scenario are split into two columns for the sake of visual readability.

These findings reveal that subgoals significantly reduce training time by providing intermediate milestones, particularly in environments with sparse rewards or long horizons. More importantly, supplementary rewards are essential for effective learning, especially in complex tasks where subgoals alone cannot bridge the reward gap. As a consequence,

an important observation is that agents can be decoupled from subgoals once training is complete, reducing computational overhead and operational costs during deployment.

6. Conclusions and Future Work

This work has shown that LLMs can be effectively leveraged as subgoal generators to help RL agents learn in sparse environments, without requiring any fine-tuning of the language models. By providing structured guidance, LLMs help agents overcome reward sparsity and improve learning efficiency, particularly in long-horizon tasks.

We have conducted an extensive analysis of three different LLMs, evaluating their effectiveness in generating useful subgoals for RL agents. Our results demonstrate superior performance over recent teacher–student baselines within the MiniGrid benchmark. Additionally, we have explored three distinct subgoal-generation strategies, each tailored to different task requirements.

Furthermore, we have shown that our method can override the need for an LLM in deployment when the subgoal is not considered in the agent's observation space. That is, we use the LLM to accelerate the training while maintaining scalability for real-world applications, where querying an LLM at inference time may not always be feasible. In addition, by not relying on perfect guidance, this approach improves the agent's robustness and generalization, particularly when the quality of subgoals is degraded, which is critical in real-world scenarios such as robotic exploration in novel environments, content recommendation with shifting user preferences, or vehicular traffic management under varying inflow rate [54].

Future Work

We aim to extend our framework to a broader set of benchmarks. While our current results demonstrate the benefits of LLM-generated subgoals, testing on a wider variety of domains would provide deeper insights into their general applicability. Additionally, incorporating real-world tasks, like robotic assemblies or autonomous drone flights, could further validate the robustness of our approach.

Another promising direction is to make the proposed framework more adaptive to the agent's knowledge progression. This can be approached in two ways. First, by dynamically adjusting the complexity of subgoals: breaking tasks into smaller, manageable steps and progressively increasing their difficulty as the agent improves. Second, by training agents to follow LLM-generated subgoals more closely, either by applying higher weighting factors (e.g., $\alpha \approx 1$) or by using dynamically adjusted weighting factors. These strategies could enhance the agent's ability to generalize to entirely new environments. Consequently, the following key question arises: can an agent leverage LLM guidance to solve novel tasks in completely unfamiliar settings? Investigating this question could provide insights into how well LLM-driven subgoal generation supports cross-domain transfer and rapid adaptation in open-world settings

Finally, our current framework includes an LLM surrogate component, which serves as an approximation of the reasoning capabilities of large models. While this surrogate approach is already effective, refining it could lead to even better subgoal quality and alignment with the agent's capabilities.

Author Contributions: Conceptualization, U.R.-G., A.A. and J.D.S.; methodology, U.R.-G.; software, U.R.-G. and A.A.; validation, J.D.S.; investigation, U.R.-G.; writing—original draft preparation, U.R.-G.; writing—review and editing, A.A. and J.D.S.; visualization, U.R.-G.; supervision, A.A. and J.D.S.; funding acquisition, A.A. and J.D.S. All authors have read and agreed to the published version of the manuscript.

Funding: Alain Andres and Javier Del Ser acknowledge funding support from the `Basque Government` through its ELKARTEK funding program (KK-2024/00064, IKUN). The work of Javier Del Ser is also supported by the consolidated research group MATHMODE (IT1866-26) funded by the same institution. Alain Andres also acknowledges funding support from IKASLAGUN project (ref. 2024-CIE2-000006-01), funded by *Diputación Foral de Gipuzcoa* under the program *Red Guipuzcoana de Ciencia, Tecnología e Innovación: GIPUZCOA NEXT*.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and analyzed during the current study were produced using large language models (LLMs) through the framework described in the manuscript. While the exact outputs of LLMs cannot be fully reproduced due to their stochastic nature, the complete framework and instructions required to generate equivalent datasets are provided. Researchers can replicate the data generation process using the framework to obtain comparable datasets for validation and further experimentation.

Acknowledgments: During the preparation of this work, the author(s) used ChatGPT in order to improve the readability of the original contents. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. Prompts Used for Subgoal Generation

In this appendix, we detail the prompts used to instruct the LLM for subgoal generation at the start of each episode. These prompts are provided sequentially, using a CoT approach to maintain context and build upon previous responses.

Figure A1 illustrates the initial prompt, where the LLM is asked about its knowledge of MiniGrid. This step establishes a foundational understanding of the environment, which is crucial for generating relevant subgoals.

Similarly, Figure A2 shows a prompt that provides detailed information about the MiniGrid environment, including the encoding of different objects and the representation of the agent's direction. This prompt ensures that the LLM has a clear understanding of the environment's state and object semantics. Finally, Figure A3 presents the final prompt, where the LLM is explicitly instructed to generate subgoals based on the current state. The LLM is guided to focus on specific objects of interest rather than movement patterns and is asked to provide the subgoals in a structured format.

These prompts are delivered sequentially at the first time step of each episode in state s_0, with the LLM's responses building on the information provided in the previous prompts. This method leverages Chain of Thought reasoning, allowing the LLM to maintain memory and coherence throughout the prompting process.

Subgoal Generation Prompting

User:
Tell me about your knowledge of MiniGrid, Farama-Foundation's grid world environments for reinforcement learning.

Figure A1. Initial prompt input to the LLM about its knowledge of MiniGrid, establishing a foundational understanding of the environment under consideration.

Subgoal Generation Prompting

User:
Each environment in MiniGrid is defined by a grid. I will provide you a matrix representing this grid.

Each tile in the grid is encoded with a number, and each number represent the following:

0: Wall, walls are not movable objects, that cannot be overpassed.
2: Key, the key is used to unlock locked doors.
x: Floor, the agent can walk of these.
3: Ball, the ball is the final goal of the environment.
7: Closed door, these doors can be opened without a key.
8: Locked door, these doors need a key to be opened.
9: Opened door, these doors can be overpassed.

Also the agent is represented by:
>: Looking right
V: Looking down
<: Looking left
^: Looking up

With all of this in mind, describe the semantics of the state, using only information from the state and your knowledge of MiniGrid.

Figure A2. Prompt providing detailed information about MiniGrid's grid encoding scheme and object representation, ensuring the LLM understands the state semantics.

Subgoal Generation Prompting

User:
Now, respond by explicity declaring which subgoals would be optimal to progress toward the goal. Remember that keys are needed to open locked doors.

Do not provide subgoals related to movement patterns. Instead, focus on subgoals involving specific objects of interest. Also, provide the final goal as subgoal.

For the subgoals, follow the following format:
<subgoals>
1. instruction (str) (x (int), y (int))
2. instruction (str) (x (int), y (int))
3. instruction (str) (x (int), y (int))
</subgoals>
The instruction must be text without any number. The x and y are the coordinates of the subgoal. Do not add any more information.

The state the agent is seeing is the following: {state}

Figure A3. Final prompt instructing the LLM to generate specific subgoals based on the current state, focusing on object-related goals and providing subgoals in a structured format.

Appendix B. Training Solely with Oracle Data Is Insufficient

In this appendix, we delve into the experimental setup and results that explore the benefits of using surrogate LLMs for training agents, as compared to training solely on oracle-generated data. We analyze the implications of different training–evaluation configurations and present a detailed comparison of agents trained with oracle data versus surrogate LLM data.

Appendix B.1. Experimental Setup

The primary objective of this experiment is to explore the benefits of surrogate LLMs. Specifically, we aim to determine whether training an agent on data generated by surrogate LLMs provides advantages over training exclusively on an oracle-generated data set. The hypothesis is that training an agent solely on oracle-generated data may result in poor performance when exposed to a real LLM with inherent errors and variability. This is because the agent becomes too reliant on perfect subgoal guidance, and the introduction of errors in a real LLM may disrupt its ability to succeed. By training the agent on data generated by a surrogate LLM, we avoid the need for continuous LLM queries during training, ensuring scalability. However, we hypothesize that this approach will lead to better generalization over time, with improved performance when the agent is eventually exposed to a real LLM that can handle errors more effectively. To achieve this, we train agents using the following two data sources:

- *Oracle-generated data*: The agent is trained on a perfectly curated data set.
- *Surrogate LLM data*: The agent is trained on data generated by a surrogate LLM (e.g., Llama or DeepSeek).

After training, the agents are evaluated on real-world LLMs to assess their performance in practical scenarios where no surrogate models are available. The experiments performed for this evaluation consist of the following two phases:

- *Training*: Agents are trained on either oracle-generated data or surrogate LLM data.
- *Evaluation*: The trained agents are evaluated on real-world LLMs to measure their effectiveness.

Performance is assessed based on three subgoal types: position, representation, and language. This results in twelve distinct configurations, producing the results summarized in Table A1 and Figure A4.

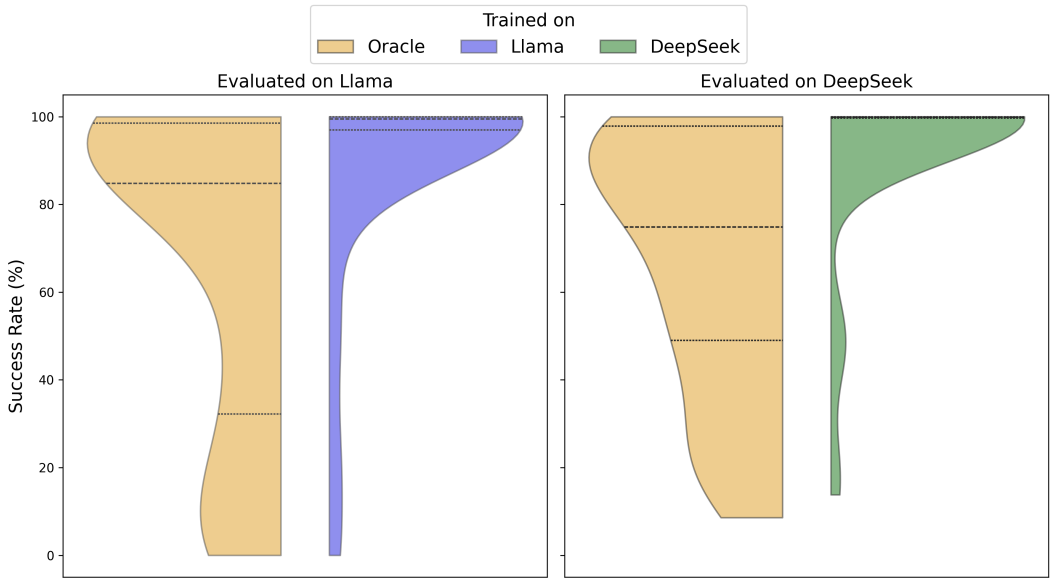


Figure A4. Violin plots comparing the success rates of agents trained on oracle-generated data versus surrogate LLM data (Llama or DeepSeek) and evaluated on real-world LLMs. The left plot shows results for agents evaluated on Llama, while the right plot presents results for agents evaluated on DeepSeek. The distribution of success rates highlights the performance gap between training exclusively on oracle data and incorporating surrogate LLM data, emphasizing the advantages of the latter in real-world generalization.

Table A1. Success rates (%) and percentiles for different subgoal types and training–evaluation configurations. The columns have four different configurations, where the first value in each configuration refers to the language model used for training (either oracle or a surrogate LLM) and the second value represents the LLM evaluated in the real-world scenario, without any surrogate approach.

Metric	Subgoal Type	Oracle → Llama	Oracle → DeepSeek	Llama → Llama	DeepSeek → DeepSeek
Mean	Position	72.77	55.15	97.87	99.79
	Representation	62.95	73.87	89.61	91.88
	Language	63.01	72.07	78.21	85.54
Median	Position	83.60	53.50	99.50	99.80
	Representation	81.35	84.60	98.70	99.80
	Language	93.30	87.00	99.70	100.00
IQR	Position	36.58	34.75	1.47	0.30
	Representation	66.93	43.55	3.60	1.75
	Language	90.32	50.20	38.15	0.30

Appendix B.2. Insights and Implications of Training with Surrogate LLM Data

The results reveal that agents trained on data from surrogate LLMs (Llama or DeepSeek) consistently outperform those trained solely on oracle-generated data and evaluated on LLMs. This suggests that while oracle data provides a high-quality, curated data set of subgoals, it may lack the diversity and complexity inherent in real-world data. Surrogate LLM data, on the other hand, better captures the nuances and variability of real-world scenarios, leading to improved generalization and performance.

A particularly notable observation is the significant performance gap in the DeepSeek surrogate model. The oracle, being a perfect data set, appears to overly influence the DeepSeek agent, which normally exhibits poor accuracy. This overfitting to the oracle's perfection results in suboptimal performance when the agent is evaluated on real-world DeepSeek data. In contrast, training directly on surrogate DeepSeek data yields significantly higher success rates, as the agent learns to adapt to the inherent characteristics and limitations of DeepSeek.

These findings underscore the importance of incorporating surrogate LLM data in our training pipeline to enhance the robustness and applicability of agents in practical settings.

Appendix C. LLM Accuracy Evaluation

In this appendix, we present the detailed algorithm (Algorithm A1) used to evaluate subgoal predictions generated by language models. The LLM evaluation is conducted on a data set containing entries with both oracle-provided ground truth subgoals and LLM-generated predictions, where each subgoal is represented by a prompt (descriptive text) and a position (coordinates in the environment). The process is designed to assess the accuracy of these predictions in two key aspects: *position correctness* and *semantic correctness*.

Position correctness assesses how closely the predicted coordinates match the ground truth locations. To measure this, we compute the Manhattan distance between each predicted and true position, which quantifies the step-wise spatial discrepancy along grid axes. The mean Manhattan distance (Equation (2)) captures the average alignment error across all subgoals, while the standard deviation (Equation (3)) reflects the variability of these errors.

Semantic correctness, by contrast, evaluates whether the predicted subgoal descriptions explicitly reference the key elements of the ground truth prompt. In practice, this requires that the predicted text includes the correct object name and, when relevant, its associated attributes such as color. A prediction is considered semantically correct if these essential components are present, ensuring that representation-based and language-based subgoals preserve the intended meaning and context.

Algorithm A1 LLM Accuracy Evaluation

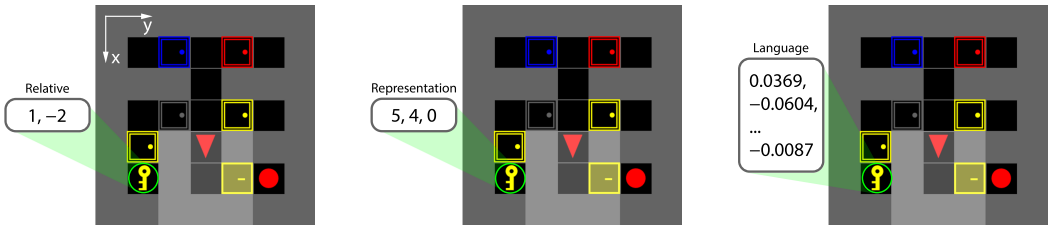
Input: Data set $\mathcal{D} = \{entry_1, \ldots, entry_N\}$, where each entry contains:
    ground_truth position and prompt
    predicted position and prompt
Output: Position accuracy, semantic accuracy, and distance metrics
Initialize: Initialize accuracy counters and distance metrics to 0.
for each $entry \in \mathcal{D}$ **do**
    Standardize predicted prompts (e.g., replace "goal" with "ball")
    Remove redundant subgoals and merge close-position ones
    Pad prediction list to match length of ground truth
    for each $(y, \hat{y})$ in $(\texttt{ground\_truth}, \texttt{prediction})$ **do**
        if $y\_pos == \hat{y}\_pos$ **then**
            $position\_correct \leftarrow position\_correct + 1$
            Append 0 to *manhattan_distances*
        else
            Compute and append Manhattan distance between y_pos and $\hat{y}\_pos$
        end if
        if y_prompt matches $\hat{y}\_prompt$ (including attributes like color) **then**
            $semantic\_correct \leftarrow semantic\_correct + 1$
        end if
        $total\_subgoals \leftarrow total\_subgoals + 1$
    end for
    if all subgoals are correct **then**
        $position\_correct\_episodes \leftarrow position\_correct\_episodes + 1$
        $semantic\_correct\_episodes \leftarrow semantic\_correct\_episodes + 1$
    end if
    $total\_entries \leftarrow total\_entries + 1$
end for
Compute accuracy, mean, and standard deviation of *manhattan_distances*
return Position and semantic evaluation metrics

Appendix D. Subgoal Types with Illustrative Examples

In this appendix, we present examples of the different types of subgoals used throughout our experiments, including position-based, representation-based, and language-based subgoals. In this study, we assume MiniGrid as our testbed for the different types of subgoals, although it can be extended to any other environment (not all environments can represent every type of subgoal considered; however, the vast majority can implement at least two of them).

Appendix D.1. Position-Based Subgoals

As shown in Figure A5a, relative position subgoals are represented as a two-dimensional vector. The first value corresponds to the distance along the x-axis, while the second value represents the distance in grid blocks along the y-axis. With this type of subgoals, the agent learns to interpret the direction it must move in to reach the subgoal, adjusting its navigation based on the relative position provided.



(**a**) (**b**) (**c**)

Figure A5. Subgoal representations in KeyCorridorS3R3 in MiniGrid. (**a**) Relative position: subgoal is the offset from the agent's current position (x, y). (**b**) Object representation: subgoal as a 3D vector (type, color, state). (**c**) Language embedding: subgoal as a 384D embedding from LM. The current subgoal (the key) is indicated with a green circle.

Appendix D.2. Representation-Based Subgoals

As shown in Figure A5b, representation subgoals are encoded as three-dimensional vectors. The first value denotes the type of object, the second represents the object's color, and the third indicates its state. There are 11 possible object types, including unseen (0), empty (1), wall (2), floor (3), door (4), key (5), ball (6), box (7), goal (8), lava (9), and agent (10). Since representation subgoals are limited to interactable objects within the environment, the subgoal space is restricted to doors, keys, balls, boxes, and goals. For colors, six options are available: red (0), green (1), blue (2), purple (3), yellow (4), and gray (5). The state value is particularly relevant for doors, with options for open (0), closed (1), and locked (2).

Appendix D.3. Language-Based Subgoals

As shown in Figure A5c, language-based subgoals are represented as embeddings retrieved from a language model based on the output text of the LLM. We use all-MiniLM-L6-v2 as the language model, which generates a 384-dimensional vector for each subgoal. For instance, in the subgoal illustrated in Figure A5c, the embeddings correspond to the output text "Go to the yellow key."

However, to reduce the complexity and variability of natural language, this subgoal (e.g., "Go to the yellow key.") is not generated freely but retrieved from a language subgoal pool. To construct this pool, we first enumerate all possible combinations of objects and attributes. For example, doors are combined with colors and states: "Red closed door."→"go to the closed red door", "Green locked door."→"go to the locked green door.". Other objects are paired with appropriate verbs, such as "yellow key"→"pick up the yellow key", "red ball"→"pick up the red ball.". Second, each phrase in the pool is converted into an embedding using the all-MiniLM-L6-v2 language model. Finally, when the LLM outputs a new instruction, we compute its embedding and retrieve the closest match in the subgoal pool using cosine similarity. The closest embedding from the pool is then used as a subgoal.

Appendix E. Adaptive Closer Subgoals

In environments where the first subgoal is too distant for the agent to reach by serendipity, we observe an intriguing phenomenon: less accurate LLMs can sometimes outperform more accurate ones during early training. This counterintuitive result arises because inaccurate LLMs tend to generate "random" subgoals that are closer to the agent's starting position. These closer subgoals provide denser rewards, making it easier for the agent to learn how to follow LLM-generated subgoals effectively. To investigate this effect, we implement a two-phase training process:

1. *Pretraining phase*: The RL agent is initially trained for 1 million steps using subgoals that are close to the agent's starting position, making them easily achievable. This

phase mimics the behavior of a less accurate LLM by providing subgoals that are relatively simple, ensuring the agent can achieve them with minimal effort.

2. *Training phase*: After pretraining, the agent transitions to standard training using subgoals generated by the surrogate LLM. By this stage, the agent has already developed an understanding of how to follow subgoals provided by the LLM, which enhances training efficiency.

Results shown in Figure A6 highlight an important consideration for LLM-generated subgoal decomposition. While breaking down a task into smaller steps is essential, it may not always be sufficient for effective learning. Our findings suggest that investigating methods to assess whether a decomposition provides reachable subgoals could be valuable.

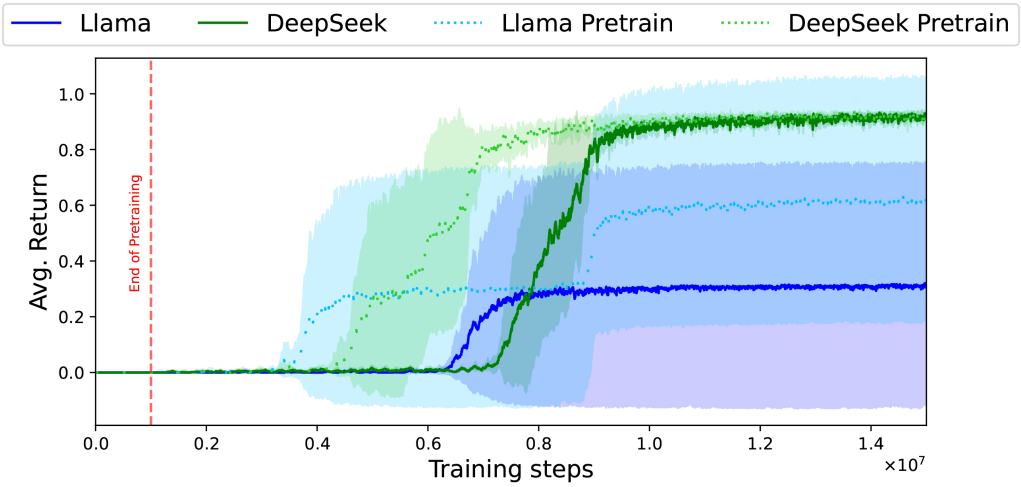


Figure A6. Training curves comparing runs *with* and *without* pretraining. The two dashed lines represent runs with pretraining, while the two solid lines correspond to runs without pretraining.

References

1. Pathak, D.; Agrawal, P.; Efros, A.A.; Darrell, T. Curiosity-driven exploration by self-supervised prediction. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017.
2. Rengarajan, D.; Vaidya, G.; Sarvesh, A.; Kalathil, D.; Shakkotta, S. Reinforcement Learning with Sparse Rewards using Guidance from Offline Demonstration. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022.
3. Klink, P.; D'Eramo, C.; Peters, J.R.; Pajarinen, J. Self-paced deep reinforcement learning. In Proceedings of the Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 6–12 December 2020; Volume 33, pp. 9216–9227.
4. Andres, A.; Schäfer, L.; Albrecht, S.V.; Del Ser, J. Using offline data to speed up reinforcement learning in procedurally generated environments. *Neurocomputing* **2025**, *618*, 129079. [CrossRef]
5. Vezhnevets, A.S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, NSW, Australia, 6–11 August 2017.
6. Anand, D.; Gupta, V.; Paruchuri, P.; Ravindran, B. An enhanced advising model in teacher-student framework using state categorization. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 6653–6660.
7. Hu, S.; Huang, T.; Liu, G.; Kompella, R.R.; Ilhan, F.; Tekin, S.F.; Xu, Y.; Yahn, Z.; Liu, L. A survey on large language model-based game agents. *arXiv* **2024**, arXiv:2404.02039. [CrossRef]
8. Pham, D.H.; Le, T.; Nguyen, H.T. How rationals boost textual entailment modeling: Insights from large language models. *Comput. Electr. Eng.* **2024**, *119*, 109517. [CrossRef]

9. Ruiz-Gonzalez, U.; Andres, A.; Bascoy, P.G.; Ser, J.D. Words as Beacons: Words as Beacons: Guiding RL Agents with High-Level Language Prompts. In Proceedings of the NeurIPS 2024 Workshop on Open-World Agents, Vancouver, BC, Canada, 15 December 2024.
10. Chevalier-Boisvert, M.; Dai, B.; Towers, M.; de Lazcano, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P.S.; Terry, J. Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In Proceedings of the 37th International Conference on Neural Information Processing System, Orleans, LA, USA, 10–16 December 2023; Volume 36, pp. 73383–73394.
11. Elman, J.L. Learning and development in neural networks: The importance of starting small. *Cognition* **1993**, *48*, 71–99. [CrossRef] [PubMed]
12. Mu, J.; Zhong, V.; Raileanu, R.; Jiang, M.; Goodman, N.; Rocktäschel, T.; Grefenstette, E. Improving intrinsic exploration with language abstractions. In Proceedings of the Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
13. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the International Conference on Machine Learning (ICML), Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
14. Graves, A.; Bellemare, M.G.; Menick, J.; Munos, R.; Kavukcuoglu, K. Automated curriculum learning for neural networks. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017; pp. 1311–1320.
15. Hacohen, G.; Weinshall, D. On the power of curriculum learning in training deep networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019; pp. 2535–2544.
16. Platanios, E.A.; Stretcu, O.; Neubig, G.; Poczos, B.; Mitchell, T.M. Competence-based curriculum learning for neural machine translation. In Proceedings of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL), Minneapolis, MN, USA, 2–7 June 2019; pp. 1151–1161.
17. Matiisen, T.; Oliver, A.; Cohen, T.; Schulman, J. Teacher–student curriculum learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3732–3740. [CrossRef] [PubMed]
18. Jiang, M.; Grefenstette, E.; Rocktaschel, T. Prioritized Level Replay. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; Volume 139, pp. 4940–4950.
19. Kanitscheider, I.; Huizinga, J.; Farhi, D.; Guss, W.H.; Houghton, B.; Sampedro, R.; Zhokhov, P.; Baker, B.; Ecoffet, A.; Tang, J.; et al. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *arXiv* **2021**, arXiv:2106.14876.
20. Racanière, S.; Lampinen, A.K.; Santoro, A.; Reichert, D.P.; Firoiu, V.; Lillicrap, T.P. Automated curricula through setter-solver interactions. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
21. Campero, A.; Raileanu, R.; Kuttler, H.; Tenenbaum J. B.; Rocktaschel T.; Grefenstette E. Learning with AMIGo: Adversarially motivated intrinsic goals. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
22. Florensa, C.; Held, D.; Geng, X.; Abbeel P. Automatic goal generation for reinforcement learning agents. In Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 1515–1528.
23. Dayan, P.; Hinton, G.E. Feudal Reinforcement Learning. In Proceedings of the Neural Information Processing Systems (NeurIPS), Denver, CO, USA, 30 November–3 December 1992; Volume 5, pp. 271–278.
24. Jiang, Y.; Gu, S.S.; Murphy, K.P.; Finn C. Language as an abstraction for hierarchical deep reinforcement learning. In Proceedings of the Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada 8–14 December 2019; Volume 32.
25. Goecks, V.G.; Waytowich, N.R.; Watkins-Valls, D.; Prakash B. Combining learning from human feedback and knowledge engineering to solve hierarchical tasks in minecraft. In Proceedings of the Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE), Stanford University, Palo Alto, CA, USA, 21–23 March 2022.
26. Prakash, R.; Pavlamuri, S.; Chernova, S. *Interactive Hierarchical Task Learning from Language Instructions and Demonstrations*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021.
27. Prakash, B.; Oates, T.; Mohsenin, T. Using LLMs for augmenting hierarchical agents with common sense priors. In Proceedings of the International FLAIRS Conference, Daytona Beach, FL, USA, 20–23 May 2024; Volume 37.
28. Shridhar, M.; Yuan, X.; Cote, M.A.; Bisk Y.; Trischler A.; Hausknecht M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
29. Radford, A.; Wu, J.; Child, R.; Luan D.; Amodei D.; Sutskever I. *Language Models Are Unsupervised Multitask Learners*; OpenAI Blog: San Francisco, CA, USA, 2019.
30. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah M.; Kaplan J.; Dhariwal P.; Neelakantan A.; Shyam P.; Sastry G.; Askell A.; et al. Language models are few-shot learners. In Proceedings of the Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020; Volume 33, pp. 1877–1901.

31. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad L.; Akkaya I.; Aleman F. L.; Almeida D.; Altenschmidt J.; Altman S.; Anadkat S.; et al. *GPT-4 Technical Report*; OpenAI Blog: San Francisco, CA, USA, 2023.
32. Touvron, H.; Lavril, T.; Izacard, G.; Martinet X.; Lachaux M.; Lacroix T.; Rozière B.; Goyal N.; Hambro E.; Azhar F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971. [CrossRef]
33. Bai, J.; Bai, S.; Chu, Y.; Cui Z.; Dang K.; Deng X.; Fan Y.; Ge W.; Han Y.; Huang F.; et al. Qwen Technical Report. *arXiv* **2023**, arXiv:2309.16609. [CrossRef]
34. Bi, X.; Chen, D.; Chen, G.; Chen S.; Dai D.; Deng C.; Ding H.; Dong K.; Du Q.; Fu Z.; et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv* **2024**, arXiv:2401.02954.
35. Cao, Y.; Zhao, H.; Cheng, Y.; Shu T.; Chen Y.; Liu G.; Liang G.; Zhao J.; Yan J.; Li Y.; et al. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *36*, 9737–9757. [CrossRef] [PubMed]
36. Li S.; Puig X.; Paxton C.; Du Y.; Wang C.; Fan L.; Chen T.; Huang D.; Akyürek E.; Anandkumar A.; et al. Pre-trained language models for interactive decision-making. In Proceedings of the Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 31199–31212.
37. Carta, T.; Romac, C.; Wolf, T.; Lamprier S.; Sigaud O.; Oudeyer P. Grounding large language models in interactive environments with online reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023; pp. 3676–3713.
38. Ma, Y.J.; Liang, W.; Wang, G.; Huang D.; Bastani O.; Jayaraman D.; Zhu Y.; Fan L.; Anandkumar A. Eureka: Human-level reward design via coding large language models. *arXiv* **2023**, arXiv:2310.12931
39. Kwon, M.; Xie, S.M.; Bullard, K.; Sadigh D. Reward design with language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
40. Klissarov, M.; D'Oro, P.; Sodhani, S.; Raileanu R.; Bacon P.; Vincent P.; Zhang A.; Henaff M. Motif: Intrinsic motivation from artificial intelligence feedback. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
41. Klissarov, M.; Henaff, M.; Raileanu, R.; Sodhani S.; Vincent P.; Zhang A.; Bacon P.; Precup D.; Machado M. C.; D'Oro P. MaestroMotif: Skill Design from Artificial Intelligence Feedback. In Proceedings of the International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025.
42. Wang, Z.; Cai, S.; Chen, G.; Liu A.; Ma X.; Liang Y. Describe, Explain, Plan and Select: Interactive planning with large language models enables open-world multi-task agents. In Proceedings of the Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10—16 December 2023.
43. Du, Y.; Watkins, O.; Wang, Z.; Colas C.; Darrell T.; Abbeel P.; Gupta A.; Andreas J. Guiding Pretraining in reinforcement learning with large language models. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23—29 July 2023; pp. 8657–8677.
44. Pignatelli, E.; Ferret, J.; Rockaschel, T.; Grefenstette E.; Paglieri D.; Coward S.; Toni L. Assessing the Zero-Shot Capabilities of LLMs for Action Evaluation in RL. *arXiv* **2024**, arXiv:2409.12798 .
45. Ahn, M.; Brohan, A.; Brown, N.; Chebotar Y.; Cortes O.; David B.; Finn C.; Fu C.; Gopalakrishnan K.; Hausman K.; et al. Do as i can, not as i say: Grounding language in robotic affordances. In Proceedings of the Conference on Robot Learning (CoRL), Auckland, New Zealand, 14–18 December 2022; pp. 287–318.
46. Wang, Z.; Cai, S.; Liu, A.; Jin Y.; Hou J.; Zhang B.; Lin H.; He Z.; Zheng Z.; Yang Y.; et al. JARVIS-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv* **2023**, arXiv:2311.05997. [CrossRef]
47. Cobbe, K.; Hesse, C.; Hilton, J.; Schulman, J. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; Volume 119, pp. 2048–2056.
48. Emani, M.; Foreman, S.; Sastry, V.; Xie, Z.; Raskar, S.; Arnold, W.; Thakur, R.; Vishwanath, V.; Papka, M.E. A Comprehensive Performance Study of Large Language Models on Novel AI Accelerators. *arXiv* **2023**, arXiv:2310.04607. [CrossRef]
49. Reynolds, L.; McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In Proceedings of the CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021.
50. Kojima, T.; Gu, S.S.; Reid, M.; Matsuo Y.; Iwasawa Y. Large language models are zero-shot reasoners. In Proceedings of the 36th International Conference on Neural Information Processing System, New Orleans, LA, USA, 28 November—9 December 2022; Volume 35, pp. 22199–22213.
51. Reimers, N.; Gurevych, I. all-MiniLM-L6-v2: Sentence Embeddings Using MiniLM-L6-v2. 2024. Available online: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 (accessed on 22 August 2025).
52. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford A.; Klimov O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]

53. Andres, A.; Villar-Rodriguez, E.; Del Ser, J. An evaluation study of intrinsic motivation techniques applied to reinforcement learning over hard exploration environments. In *Proceedings of the International Cross-Domain Conference for Machine Learning and Knowledge Extraction*; Springer: Vienna, Austria, 2022; pp. 201–220.
54. Padakandla, S. A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. *ACM Comput. Surv.* **2021**, *54*, 1–35. [CrossRef]