Guiding Pretraining in Reinforcement Learning with Large Language Models

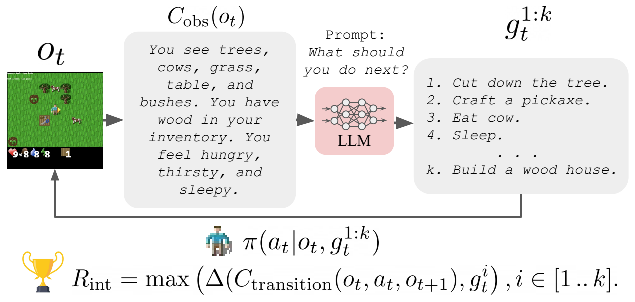
Yuqing Du [*1] Olivia Watkins [*1] Zihan Wang [2] Cédric Colas [3 4] Trevor Darrell [1] Pieter Abbeel [1]
Abhishek Gupta [2] Jacob Andreas [3]

Abstract

Reinforcement learning algorithms typically struggle in the absence of a dense, well-shaped reward function. Intrinsically motivated exploration methods address this limitation by rewarding agents for visiting novel states or transitions, but these methods offer limited benefits in large environments where most discovered novelty is irrelevant for downstream tasks. We describe a method that uses background knowledge from text corpora to shape exploration. This method, called ELLM (**E**xploring with **LLM**s) rewards an agent for achieving goals suggested by a language model prompted with a description of the agent's current state. By leveraging large-scale language model pretraining, ELLM guides agents toward human-meaningful and plausibly useful behaviors without requiring a human in the loop. We evaluate ELLM in the *Crafter* game environment and the *Housekeep* robotic simulator, showing that ELLM-trained agents have better coverage of common-sense behaviors during pretraining and usually match or improve performance on a range of downstream tasks.

1. Introduction

Reinforcement learning algorithms work well when learners receive frequent rewards that incentivize progress toward target behaviors. But hand-defining such reward functions requires significant engineering efforts in all but the simplest cases (Amodei et al., 2016; Lehman et al., 2020). To master complex tasks in practice, RL agents may therefore need to



You see trees, cows, grass, table, and bushes. You have wood in your inventory. You feel hungry, thirsty, and sleepy.

1. Cut down the tree.
2. Craft a pickaxe.
3. Eat cow.
4. Sleep.
. . .
k. Build a wood house.

$$R_{\text{int}} = \max\left(\Delta(C_{\text{transition}}(o_t, a_t, o_{t+1}), g_t^i\right), i \in [1 \,..\, k].$$

Figure 1: **ELLM** uses a pretrained large language model (LLM) to suggest plausibly useful goals in a task-agnostic way. Building on LLM capabilities such as context-sensitivity and common-sense, ELLM trains RL agents to pursue goals that are likely meaningful without requiring direct human intervention. Prompt is illustrative; see full prompt and goal format in Appendix D.

learn some behaviors in the absence of externally-defined rewards. What should they learn?

Intrinsically motivated RL methods answer this question by augmenting rewards with auxiliary objectives based on novelty, surprise, uncertainty, or prediction errors (Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2019; Zhang et al., 2021; Liu & Abbeel, 2021; Yarats et al., 2021). But not everything novel or unpredictable is useful: noisy TVs and the movements of leaves on a tree may provide an infinite amount of novelty, but do not lead to meaningful behaviors (Burda et al., 2019). More recent approaches compute novelty with higher-level representations like language (Tam et al., 2022; Mu et al., 2022), but can continue driving the agent to explore behaviors that are unlikely to correspond to any human-meaningful goal—like enumerating unique configurations of furniture in a household. It is not sufficient for extrinsic-reward-free RL agents to optimize for novelty alone: learned behaviors must also be useful.

In this paper, we describe a method for using not just language-based representations but **pretrained language models** (LLMs) as a source of information about useful behavior. LLMs are probabilistic models of text trained on large text corpora; their predictions encode rich information about human common-sense knowledge and cultural conven-

[*]Equal contribution [1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA [2]University of Washington, Seattle [3]Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory [4]Inria, Flowers Laboratory. Correspondence to: Yuqing Du <yuqing_du@berkeley.edu>, Olivia Watkins <oliviawatkins@berkeley.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

tions. Our method, **E**xploring with **LLM**s (ELLM), queries LMs for possible goals given an agent's current context and rewards agents for accomplishing those suggestions. As a result, exploration is biased towards completion of goals that are diverse, context-sensitive, and human-meaningful. ELLM-trained agents exhibit better coverage of useful behaviors during pretraining, and outperform or match baselines when fine-tuned on downstream tasks.

2. Background and Related Work

Intrinsically Motivated RL. When reward functions are sparse, agents often need to carry out a long, specific sequence of actions to achieve target tasks. As action spaces or target behaviors grow more complex, the space of alternative action sequences agents can explore grows combinatorially. In such scenarios, undirected exploration that randomly perturbs actions or policy parameters has little chance of succeeding (Ten et al., 2022; Ladosz et al., 2022).

Many distinct action sequences can lead to similar outcomes (Baranes & Oudeyer, 2013)—for example, most action sequences cause a humanoid agent to fall, while very few make it walk. Building on this observation, **intrinsically motivated** RL algorithms (IM-RL) choose to explore *outcomes* rather than actions (Oudeyer & Kaplan, 2009; Ten et al., 2022; Ladosz et al., 2022). **Knowledge-based** IMs (KB-IMs) focus on maximising the diversity of states (reviews in Aubret et al., 2019; Linke et al., 2020). **Competence-based IMs** (CB-IMs) maximise the diversity of *skills* mastered by the agent (review in Colas et al., 2022). Because most action sequences lead to a very restricted part of the outcome space (e.g. all different ways of *falling on the floor* likely correspond to a single outcome), these methods lead to a greater diversity of outcomes than undirected exploration (Lehman et al., 2008; Colas et al., 2018).

However, maximizing diversity of outcomes may not always be enough. Complex environments can contain sources of infinite novelty. In such environments, seeking ever-more-novel states might drive learning towards behaviors that have little relevance to the true task reward. Humans do not explore outcome spaces uniformly, but instead rely on their physical and social common-sense to explore *plausibly-useful* behaviors first. In video games, they know that keys should be used to open doors, ladders should be climbed, and snakes might be enemies. If this semantic information is removed, their exploration becomes severely impacted (Dubey et al., 2018). The approach we introduce in this paper, ELLM, may be interpreted as a CB-IM algorithm that seeks to explore the space of possible and plausibly-useful skills informed by human prior knowledge.

Linguistic Goals and Pretrained Language Models. One way of representing a diverse outcome space for exploration is through language. Training agents to achieve language goals brings several advantages: (1) goals are easy to express for non-expert users; (2) they can be more abstract than standard state-based goals (Colas et al., 2022); and (3) agents can generalize better thanks to the partial compositionality and recursivity of language (Hermann et al., 2017; Hill et al., 2019; Colas et al., 2020). Such linguistic goals can be used as instructions for language-conditioned imitation learning or RL. In RL, agents typically receive language instructions corresponding to the relevant reward functions (Luketina et al., 2019) and are only rarely intrinsically motivated (with the exception of Mu et al., 2022; Colas et al., 2020; Tam et al., 2022), where language is also used as a more general compact state abstraction for task-agnostic exploration.

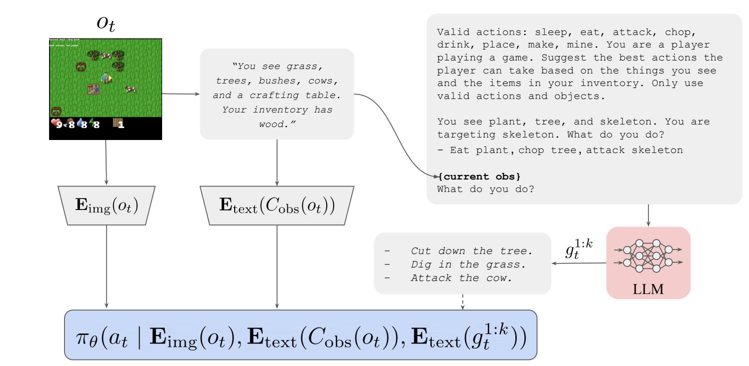
Representing goals in language unlocks the possibility of using text representations and generative models of text (large language models, or LLMs) trained on large corpora. In imitation learning, text pretraining can help learners automatically recognize sub-goals and learn modular sub-policies from unlabelled demonstrations (Lynch & Sermanet, 2020; Sharma et al., 2021), or chain pre-trained goal-oriented policies together to accomplish high-level tasks (Yao et al., 2020; Huang et al., 2022a; Ahn et al., 2022; Huang et al., 2022b). In RL, LM-encoded goal descriptions greatly improve the generalization of instruction-following agents across instructions (Chan et al., 2019) and from synthetic to natural goals (Hill et al., 2020). LLMs have also been used as proxy reward functions when prompted with desired behaviors (Kwon et al., 2023). Unlike these approaches, ELLM uses pretrained LLMs to constrain exploration towards plausibly-useful goals in a task-agnostic manner. It does not assume a pretrained low-level policy, demonstrations, or task-specific prompts. Most similar to our work, Choi et al. (2022) also prompt LLMs for priors. However, they use LM priors to classify safe and unsafe states to reward, which is a subset of common-sense exploratory behaviors ELLM should generate. Also similar to our work, Kant et al. (2022) query LLMs for zero-shot commonsense priors in the Housekeep environment, but they apply these to a planning task rather than as rewards for reinforcement learning.

3. Structuring Exploration with LLM Priors

3.1. Problem Description

We consider partially observed Markov decision processes defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \Omega, \mathcal{T}, \gamma, \mathcal{R})$, in which observations $o \in \Omega$ derive from environment states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ via $\mathcal{O}(o \mid s, a)$. $\mathcal{T}(s' \mid s, a)$ describes the dynamics of the environment while $\mathcal{R}$ and γ are the environment's reward function and discount factor.

IM agents optimize for an intrinsic reward $\mathcal{R}_{\text{int}}$ alongside


o_t
"You see grass, trees, bushes, cows, and a crafting table. Your inventory has wood."
$\mathbf{E}_{\text{img}}(o_t)$
$\mathbf{E}_{\text{text}}(C_{\text{obs}}(o_t))$
$g_t^{1:k}$
LLM
$\pi_\theta(a_t \mid \mathbf{E}_{\text{img}}(o_t), \mathbf{E}_{\text{text}}(C_{\text{obs}}(o_t)), \mathbf{E}_{\text{text}}(g_t^{1:k}))$

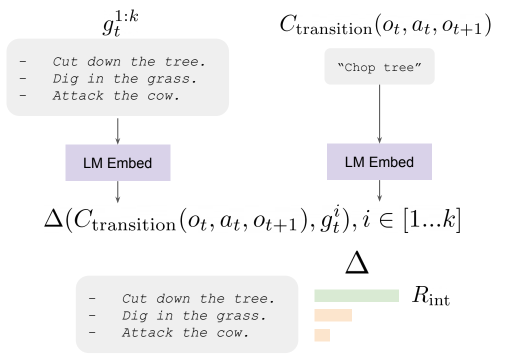
```
Valid actions: sleep, eat, attack, chop,
drink, place, make, mine. You are a player
playing a game. Suggest the best actions the
player can take based on the things you see
and the items in your inventory. Only use
valid actions and objects.

You see plant, tree, and skeleton. You are
targeting skeleton. What do you do?
- Eat plant, chop tree, attack skeleton

{current obs}
What do you do?
```

```
-   Cut down the tree.
-   Dig in the grass.
-   Attack the cow.
```

(a) Policy parametrization for ELLM. We optionally condition on embeddings of the goals $E_{\text{text}}(g_t^{1:k})$ and state $E_{\text{text}}(C_{\text{obs}}(o_t))$.


$g_t^{1:k}$
$C_{\text{transition}}(o_t, a_t, o_{t+1})$
"Chop tree"
LM Embed
LM Embed
$\Delta(C_{\text{transition}}(o_t, a_t, o_{t+1}), g_t^i), i \in [1...k]$
Δ
R_{int}

```
-   Cut down the tree.
-   Dig in the grass.
-   Attack the cow.
```

(b) LLM reward scheme. We reward the agent for the similarity between the captioned transition and the goals.

Figure 2: ELLM uses GPT-3 to suggest adequate exploratory goals and SentenceBERT embeddings to compute the similarity between suggested goals and demonstrated behaviors as a form of intrinsically-motivated reward.

or in place of $\mathcal{R}$. CB-IM methods, in particular, define $\mathcal{R}_{\text{int}}$ via a family of goal-conditioned reward functions:

$$\mathcal{R}_{\text{int}}(o,\, a,\, o') \;=\; \mathbb{E}_{g\sim\mathcal{G}}\left[\mathcal{R}_{\text{int}}(o,\, a,\, o' \mid g)\right]. \qquad (1)$$

A CB-IM agent is expected to perform well with respect to the original $\mathcal{R}$ when the intrinsic reward $\mathcal{R}_{\text{int}}$ is both easier to optimize and well aligned with $\mathcal{R}$, such that behaviors maximizing $\mathcal{R}_{\text{int}}$ also maximize $\mathcal{R}$. Every CB-IM algorithm must define two elements in Equation 1: (1) the distribution of goals to sample from, i.e. $\mathcal{G}$, and (2) the goal-conditioned reward functions $\mathcal{R}_{\text{int}}(o, a, o' \mid g)$. Given these, A CB-IM algorithm trains a goal-conditioned policy $\pi(a \mid o, g)$ to maximize R_{int}. For some intrinsic reward functions, agents may achieve high reward under the original reward function $\mathcal{R}$ immediately; for others, additional fine-tuning with $\mathcal{R}$ may be required. In Equation (1), the space of goals $\mathcal{G}$ is determined by the goal-conditioned reward function $R_{\text{int}}(\cdot \mid g)$: every choice of g induces a corresponding distribution over optimal behaviors.

3.2. Goal-based Exploration Desiderata

How should we choose $\mathcal{G}$ and $\mathcal{R}_{\text{int}}(\cdot \mid g)$ to help agents make progress toward general reward functions $\mathcal{R}$? Goals targeted during exploration should satisfy three properties:

- **Diverse**: targeting diverse goals increases the chance that the target behavior is similar to one of them.
- **Common-sense sensitive**: learning should focus on feasible goals (`chop a tree` > `drink a tree`) which are likely under the distribution of goals humans care about (`drink water` > `walk into lava`).
- **Context sensitive**: learning should focus on goals that are feasible in the current environment configuration (e.g. `chop a tree` only if a tree is in view).

Most CB-IM algorithms hand-define the reward functions R_{int} (2) and the support of the goal distribution (1) in alignment with the original task $\mathcal{R}$, but use various intrinsic motivations to guide goal sampling (1): e.g. novelty, learning progress, intermediate difficulty (see a review in Colas et al., 2022). In **E**xploring with **L**arge **L**anguage **M**odels (ELLM), we propose to leverage language-based goal representations and language-model-based goal generation to alleviate the need for environment-specific hand-coded definitions of (1) and (2). We hypothesize that world knowledge captured in LLMs will enable the automatic generation of goals that are diverse, human-meaningful and context sensitive.

3.3. Goal Generation with LLMs ($\mathcal{G}$)

Pretrained large language models broadly fall into three categories: autoregressive, masked, or encoder-decoder models (Min et al., 2021). Autoregressive models (e.g. GPT; Radford et al., 2018), are trained to maximize the log-likelihood of the next word given all previous words, and are thus capable of language generation. Encoder-only models (e.g. BERT; Devlin et al., 2018), are trained with a masked objective, enabling effective encoding of sentence semantics. Pretraining LMs on large text corpora yields impressive zero- or few-shot on diverse language understanding and generation tasks, including tasks requiring not just linguistic knowledge but world knowledge (Brown et al., 2020).

ELLM uses autoregressive LMs to generate goals and masked LMs to build vector representations of goals. When LLMs generate goals, the support of the goal distribution becomes as large as the space of natural language strings. While querying LLMs unconditionally for goals can offer diversity and common-sense sensitivity, context-sensitivity requires knowledge of agent state. Thus, at each timestep we acquire goals by prompting the LLM with a list of the agent's available actions and a text description of the current observation via a *state captioner* $C_{\text{obs}} : \Omega \to \Sigma^*$, where Σ^* is the set of all strings (see Figure 2).

We investigate two concrete strategies for extracting goals from LLMs: (1) open-ended generation, in which the LLM outputs text descriptions of suggested goals (e.g. `next you should...`), and (2) closed-form, in which a possible goal is given to the LLM as a QA task (e.g. `Should the agent do X? (Yes/No)`). Here the LLM goal suggestion is only accepted when the log-probability of `Yes` is greater than `No`. The former is more suited for open-ended exploration and the latter is more suited for environments with large but delimitable goal spaces. While the LLM does not have prior knowledge of all possible goals, we can provide some guidance towards desirable suggestions through few-shot prompting. See Appendix D for the full prompt.

3.4. Rewarding LLM Goals ($\mathbf{R}_{\text{int}}$)

Next we consider the goal-conditioned reward (2). We compute rewards for a given goal g ($\mathcal{R}_{\text{int}}$ in Eq. 1) by measuring the semantic similarity between the LLM-generated goal and the description of the agent's transition in the environment as computed by a *transition captioner* $C_{\text{transition}} : \Omega \times \mathcal{A} \times \Omega \to \Sigma$:

$$\mathcal{R}_{\text{int}}(o, a, o' \mid g) = \begin{cases} \Delta(C_{\text{transition}}(o, a, o'), g) & \text{if } > T \\ 0 & \text{otherwise.} \end{cases}$$

Here, the semantic similarity function $\Delta(\cdot\,,\,\cdot)$ is defined as the cosine similarity between *representations* from an LM encoder $E(\cdot)$ of captions and goals:

$$\Delta(C_{\text{transition}}(o, a, o'), g) = \frac{E(C_{\text{transition}}(o, a, o')) \cdot E(g)}{\|E(C_{\text{transition}}(o, a, o'))\| \|E(g)\|}.$$

In practice, we use a pretrained SentenceBERT model (Reimers & Gurevych, 2019) for $E(\cdot)$. We choose cosine similarity to measure alignment between atomic agent actions and freeform LLM generations, as done in prior work (Huang et al., 2022a). When the caption of a transition is sufficiently close to the goal description ($\Delta > T$), where T is a similarity threshold hyperparameter, the agent is rewarded proportionally to their similarity. Finally, since there can be multiple goals suggested, we reward the agent for achieving any of the k suggestions by taking the maximum of the goal-specific rewards:

$$\Delta^{\max} = \max_{i=1\ldots k} \Delta\left(C_{\text{transition}}(o_t, a_t, o_{t+1}), g_t^i\right).$$

As a result, the general reward function of CB-IM methods from Equation 1 can be rewritten:

$$\mathcal{R}_{\text{int}}(o, a, o') = \mathbb{E}_{\mathbf{LLM}(g^{1\,..\,k} \mid C_{\text{obs}}(o))}\left[\Delta^{\max}\right]. \qquad (2)$$

3.5. Implementation Details

The full ELLM algorithm is summarized in Algorithm 1. See Figure 1 for the high-level pipeline. To impose a novelty bias, we also filter out LM suggestions that the agent has already achieved earlier in the same episode. This prevents the agent from exploring the same goal repeatedly. In Appendix L we show this step is essential to the method.

We consider two forms of agent training: (1) a **goal-conditioned** setting where the agent is given a sentence embedding of the list of suggested goals, $\pi(a \mid o, E(g^{1:k}))$, and (2) a **goal-free** setting where the agent does not have access to the suggested goals, $\pi(a \mid o)$. While R_{int} remains the same in either case, training a goal-conditioned agent introduces both challenges and benefits: it can take time for the agent to learn the meaning of the different goals and connect it to the reward, but having a language-goal conditioned policy can be more amenable to downstream tasks than an agent just trained on an exploration reward. We also consider two types of policy inputs– (1) just the partially observed pixel observations, or (2) the pixel observations combined with the embedded language-state captions $E(C_{\text{obs}}(o))$. Since (2) performs better (see analysis in Appendix A), we use (2) for all paper experiments unless otherwise specified. All variants are trained with the DQN algorithm (Mnih et al., 2013), with implementation details in Appendix H.

This paper focuses on the benefits of LLM priors for RL exploration and mostly assumes a pre-existing captioning function. In simulation, this can be acquired for free with the ground truth simulator state. For real world applications, one can use object-detection (Zaidi et al., 2022), captioning models (Stefanini et al., 2022), or action recognition models (Kong & Fu, 2022). Alternatively, one could use multi-modal vision-language models with a similar LM component (Alayrac et al., 2022). To test the robustness of our method under varying captioning quality, Section 4.1 studies a relaxation of these assumptions by looking at a variant of ELLM using a learned captioner trained on human descriptions.

4. Experiments

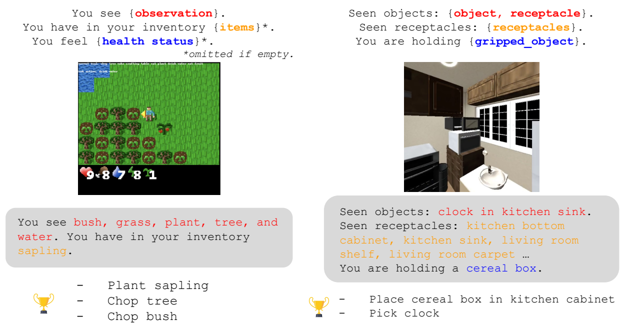
Our experiments test the following hypotheses:

- **(H1)** Prompted pretrained LLMs can generate plausibly-useful exploratory goals satisfying the desiderata listed in Section 3.2: diversity, common-sense and context sensitivity.
- **(H2)** Training an ELLM agent on these exploratory goals improves performance on downstream tasks compared to methods that do not leverage LLM-priors.

We evaluate ELLM in two complex environments: (1) *Crafter*, an open-ended environment in which exploration is required to discover long-term survival strategies

Algorithm 1 ELLM Algorithm

```
Initialize untrained policy $\pi$
$t \leftarrow 0$
$o_t \leftarrow$ env.RESET()
while $t <$ max_env_steps do
    # Generate $k$ suggestions, filtering achieved ones
    $g_t^{1:k} \leftarrow$PREV_ACHIEVED(LLM($C_{\text{obs}}(o_t)$))
    # Interact with the environment
    $a_t \sim \pi(a_t | o_t, \text{E}(C_{\text{obs}}(o_t))), \text{E}(g_t^{1:k}))$
    $s_{t+1} \leftarrow$ env.STEP($a_t$)
    # Compute suggestion achievement reward
    $r_t \leftarrow 0$
    $\Delta^{max} \leftarrow \max_{i=1 \dots k} \Delta(C_{\text{transition}}(o_t, a_t, o_{t+1}), g_t^i)$
    if $\Delta^{max} >$ threshold then
        $r_t = \Delta^{max}$
    end if
    # Update agent using any RL algorithm
    $\text{Buffer}_{t+1} \leftarrow \text{Buffer}_t \cup (o_t, a_t, g_t^{1:k}, r_t, o_{t+1})$
    $\pi \leftarrow$UPDATE($\pi$, $\text{Buffer}_{t+1}$)
end while
```



```
You see {observation}.
You have in your inventory {items}*.
You feel {health status}*.
*omitted if empty.
```

You see bush, grass, plant, tree, and water. You have in your inventory sapling.

- Plant sapling
- Chop tree
- Chop bush

```
Seen objects: {object, receptacle}.
Seen receptacles: {receptacles}.
You are holding {gripped_object}.
```

Seen objects: clock in kitchen sink. Seen receptacles: kitchen bottom cabinet, kitchen sink, living room shelf, living room carpet … You are holding a cereal box.

- Place cereal box in kitchen cabinet
- Pick clock

Figure 3: Sample templated captions and suggested goals.

(Hafner, 2021), and (2) *Housekeep*, an embodied robotics environment that requires common-sense to restrict the exploration of possible rearrangements of household objects (Kant et al., 2022). Besides environment affordances, these environments also differ in viewpoint (3rd vs 1st person) and action space (large high-level vs low-level). In each environment, we compare ELLM with existing IM-RL methods (Liu & Abbeel, 2021; Burda et al., 2019), an oracle with ground-truth rewards, and ablations of ELLM; see Table 1.

4.1. Crafter

Environment description. We first test ELLM in the Crafter environment, a 2D version of Minecraft (Hafner, 2021). Like Minecraft, Crafter is a procedurally generated and partially observable world that enables collecting and creating a set of artifacts organized along an achievement tree which lists all possible achievements and their respective prerequisites (see Figure 4 in Hafner, 2021). Although Crafter does not come with a single main task to solve, we can track agent progress along the achievement tree.

We modify the original game in two ways. Crafter's original action space already incorporates a great deal of human domain knowledge: a single do action is interpreted in different ways based on the agent's context, each of which would correspond to a very different low-level action in a real environment ('do' means '`attack`' in front of a zombie but '`eat`' in front of a plant). We remove this assistance by augmenting the action space with more specific verb + noun pairs that are not guaranteed to be useful (e.g. '`eat zombie`'). This makes it possible in Crafter to attempt a wide range of irrelevant/nonsensical tasks, providing an opportunity for an LM narrow the goal space down to reasonable goals. See Appendix C for details. Second, to make RL training easier across all conditions, we increase the damage the agent does against enemies and reduce the amount of wood required to craft a table from 2 to 1; see Appendix Figure 10 for comparisons.

We use Codex (Chen et al., 2021) as our LLM with the open-ended suggestion generation variant of ELLM, where we directly take the generated text from the LLM as the set of suggested goals to reward. Each query prompt consists of a list of possible verbs the agent can use (but not a list of all possible nouns), a description of the agent's current state, and the question '`What do you do?`'. We add two examples of similar queries to the start of the prompt in order to guide the language model to format suggestions in a consistent way; see the full prompt in Appendix D.

Goals suggested by the LLM. To answer **H1**, we study the goals suggested by the LLM in Table 2: are they diverse, context-sensitive and common-sensical? The majority of suggested goals (64.9%) are context-sensitive, sensible, and achievable in the game. Most of the 5% of goals not allowed by Crafter's physics (e.g. `build a house`) are context- and common-sensitive as well. The last third of the goals violate either context-sensitivity (13.6%) or common-sense (16.4%). See Appendix K for details.

Pretraining exploration performance. A perfect exploration method would unlock all Crafter achievements in every episode, even without prior knowledge of the set of possible achievements. Thus, we measure exploration quality as the average number of unique achievements per episode across pretraining (Figure 4). Although it is not given access to Crafter's achievement tree, ELLM learns to unlock about 6 achievements every episode, against 9 for the ground-truth-reward Oracle (Figure 4). It outperforms all exploration methods that only focus on generating novel behaviors (APT, RND, Novelty) — all limited to less than 3 achievements in average. As shown in Table 2, ELLM does not only focus on novelty but also generates

Method	Description
ELLM (ours)	Rewards the agent for achieving any goal suggested by the LLM using the similarity-based reward functions R_{int} defined in Eq. 2. It only rewards the agent for achieving a given goal once per episode (novelty bias).
Oracle (Crafter only)	The upper bound: it suggests all context-sensitive goals at any step, only common-sensical ones (from the list of valid goals) and uses the same novelty bias as ELLM. Rewards are computed exactly with a hard-coded R_{int}.
Novelty	This baseline removes the common-sense sensitivity assumption of the *Oracle* and rewards the agent for achieving any of the goals expressible in the environment including invalid ones (e.g. `drink tree`) as long as the agent performs the goal-reaching action in the right context (e.g. while facing a tree). Uses a hard-coded R_{int} and a novelty bias like the *Oracle*.
Uniform	This variant removes the novelty bias from *Novelty* and samples uniformly from the set of expressible goals.
APT (Liu & Abbeel, 2021)	State-of-the-art KB-IM algorithm that maximizes state entropy computed as the distance between the current state's embedding e_s and its K nearest neighbors $e_{s[1..K]}$ within a minibatch uniformly sampled from memory. There is no goal involved and $R_{\text{int}} = \log \|e_s - e_{s[1..K]}\|$.
RND (Burda et al., 2019)	State-of-the-art KB-IM algorithm that rewards the agent for maximizing a form of novelty estimated by the prediction error of a model h trained to predict the output of a random network $\tilde{h}$. $R_{\text{int}} = \|h(s,a) - \tilde{h}(s,a)\|$.

Table 1: Descriptions of the compared algorithms. (Additional comparisons in Appendix N).

	Suggested	Rewarded
Context-Insensitive	13.6%	1.1%
Common-Sense Insensitive	16.4%	32.4%
Good	64.9%	66.5%
Impossible	5.0%	0%

Table 2: Fractions of suggested and rewarded goals that fail to satisfy context-sensitivity or common-sense sensitivity; that satisfy these properties and are achievable in Crafter (Good); or that are not allowed by Crafter's physics. See Appendix K for examples of each.

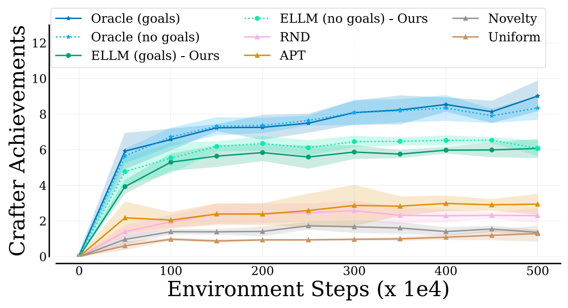


Figure 4: Ground truth achievements unlocked per episode across pretraining, mean±std across 5 seeds.

common-sensical goals. This boosts exploration in Crafter, supporting **H1**.

As discussed in Section 3.5, we also test variants of each method (with / without goal conditioning, with / without text observations) where applicable. We do not find goal conditioning to bring a significant advantage in performance during pretraining. The non-conditioned agent might infer the goals (and thus the rewarded behaviors) from context alone. Similarly to Mu et al. (2022) and Tam et al. (2022), we find that agents trained on visual + textual observations (as computed by $E(C_{\text{obs}}(o))$) outperform agents trained on visual observations only for all the tested variants (opaque vs semi-transparent bars in Appendix Figure 8). That said, optimizing for novelty alone, whether in visual or semantic spaces, seems to be insufficient to fully solve Crafter.

The naïve approach of finetuning a pretrained policy on the downstream task performs poorly across all pretraining algorithms. We hypothesize this is because relevant features and Q-values change significantly between pretraining and finetuning, especially when the density of rewards changes. Instead, we find it is more effective to use the pretrained policy for guided exploration. We initialize and train a new agent, but replace 50% of the algorithm's randomly-sampled ϵ-greedy exploration actions with actions sampled from the pretrained policy. In Appendix M we include the poor finetuning results discuss why we think guided exploration does better.

Figure 5 compares the downstream performance of ELLM to the performance of the two strongest baselines RND and APT using both transfer methods. (full comparisons with all baselines shown in Appendix B). For the goal-conditioned version of ELLM, we provide the agent with the sequence of subgoals required to achieve the task. Even though not all subgoals were mastered during pretraining, we still observe that the goal-conditioned pretrained agents outperform the unconditioned ones.

Performance of the different methods varies widely task-to-task and even seed-to-seed since each task requires a different set of skills, and any given agent may or may not have learned a particular skill during pretraining. For instance, ELLM agents typically learn to place crafting tables and attack cows during pretraining, leading to low-variance learning curves. They typically do not learn to make wood swords, so we see a high-variance learning curve which depends on how quickly each agent stumbles across the goal during finetuning. Despite the variance, we see that goal-conditioned ELLM stands out as the best-performing method on average. Notably, ELLM (both goal-conditioned and goal-free) is the only method with nonzero performance across all tasks.

ELLM with imperfect transition captioner. Perfect captioners might not be easy to obtain in some environments. However, trained captioners might generate more linguistic diversity and make mistakes. To test the robustness of ELLM to diverse and imperfect captions, we replace the oracle transition captioner $C_{\text{transition}}$ with a captioner trained on a mixture of human and synthetic data (847+900 labels) using the ClipCap algorithm (Mokady et al., 2021b). Synthetic data removes some of the human labor while still providing a diversity of captions for any single transition (3 to 8). Appendix J presents implementation details and analyzes how the trained captioner might cause errors in generated rewards. Although its false negative rate is low (it detects goal achievements well), its false positive rate is rather high. This means it might generate rewards for achievements that were not unlocked due to a high similarity between the generated caption and goal description generated by the LLM. In ELLM pretraining, we use the learned captioner to caption transitions where an action is successful and use that caption to compute the reward via the similarity metric (see Section 3). Figure 6 shows that ELLM performance is overall robust to this imperfect captioner.

4.2. Housekeep

Environment description. Housekeep is an embodied robotics environment where the agent is tasked with cleaning up a house by rearranging misplaced objects (Kant et al., 2022). The agent must successfully match the environment's ground truth correct mapping of objects to receptacles without direct instructions specifying how objects need to be rearranged. This mapping was determined via crowd-sourcing common-sense object-receptacle combinations. An example layout of the task can be found in Figure 1 in Kant et al. (2022). Common-sense priors are necessary for learning to rearrange misplaced objects into reasonable configurations.

	Task 1	**Task 2**	**Task 3**	**Task 4**
Match Acc.	85.7%	87.5%	50%	66.7%
Mismatch Acc.	93.8%	90.1%	94.0%	87.6%

Table 3: Classification accuracy of LLM for each Housekeep task (top row is true positives, bottom row is true negatives).

We focus on a simplified subset of Housekeep consisting of 4 different scenes with one room each, each with 5 different misplaced objects and a suite of different possible receptacles; see Appendix F for details. Because the agent does not have access to the ground truth target locations, we use the game reward's rearrangement success rate as a measure of exploration quality: common-sensical exploration should perform better. A success rate of 100% means the agent has picked and placed all 5 misplaced objects in correct locations. Note that we intentionally focus on a domain where the downstream application benefits strongly from exploring reasonable goals during pretraining. Rather than designing reward functions that correspond to all correct rearrangements for all possible objects, we investigate whether ELLM can be a general purpose method that guides learning human-meaningful behaviors.

Unlike Crafter's combinatorial and high-level action space, Housekeep operates with low-level actions: moving forward, turning, looking up or down, and picking or placing an object. This allows us to investigate whether ELLM enables high-level exploration despite using lower-level control. We assume access to an egocentric instance segmentation sensor to generate captions of in-view objects and receptacles, and use the `text-davinci-002` InstructGPT model (Ouyang et al., 2022) as our LLM. Given a description of visible objects, the receptacles the objects are currently in, and all previously seen receptacles, we create a list of all possible object-receptacle mappings. We use the closed-form variant of ELLM and query the LLM for whether each object should be placed in each receptacle as a `yes/no` question. By querying for each object-receptacle combination individually, we are able to cache and efficiently reuse LLM queries. The agent can be given two types of goals: (1) picking an object if it is not already in a suggested receptacle, and (2) placing a gripped object in a suggested receptacle.

Goals suggested by LLM. In Housekeep, we assess LLM goals by looking at the classification accuracy of correct and incorrect arrangements (Table 3). We find that the LLM accuracy at identifying mismatches (e.g. `vase in kitchen sink`) are all above 87%, however, accuracy of identifying matches varies greatly depending on the available objects and receptacles (ranging from 50-90%). Since there are only a few correct positions, each false negative hurts accuracy greatly. Taking a closer look, we find that some

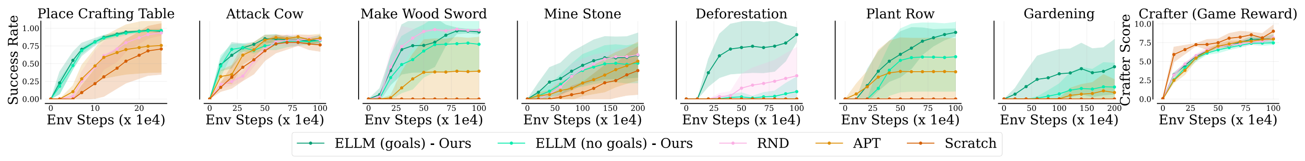


Figure 5: Success rates across training for each of the seven downstream tasks in the Crafter environment. Each run trains an agent from scratch while leveraging a pretrained policy for exploration. Plots show mean ± std for 5 seeds. Some plots have multiple overlapping curves at 0.

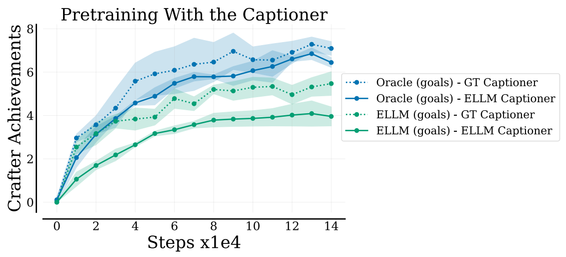


Figure 6: Pretraining with a learned captioner vs a ground truth captioner. We see performance drops, especially for ELLM, but still relatively good performance. (3 seeds, mean± std.)

LLM labels are reasonable despite disagreeing with the environment mapping: e.g. suggesting `vase in living room table`, and not suggesting `pan in living room cabinet`. This suggests that there are ambiguities in the ground truth mappings, likely due to human disagreement.

Pretraining and downstream performance. To investigate **H1**, we compare ELLM against the strongest baselines (RND, APT, Novelty) described in Table 1. In Housekeep the novelty baseline rewards the agent for novel instances of pick or place actions in an episode, allowing us to differentiate between success attributable solely to the captioner and the pick/place prior, and success attributable to any LLM common-sense priors. For brevity, we focus only on the pixel + text-observation variant of all methods. Sample efficiency curves measuring the ground truth rearrangement success during both pretraining and finetuning are shown in Figure 7a. In three of the four tasks, we find that the ELLM bias leads to higher success rates during pretraining, suggesting coverage better aligned with the downstream task compared to the baselines. We also find much higher pretraining success rates in the first two tasks. Since Table 3 shows higher LLM accuracy for these two tasks, this difference shows the impact of LLM inaccuracies on pretraining.

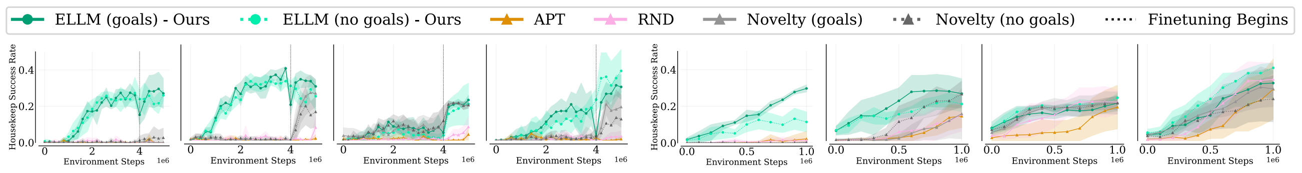
For **H2**, we test two different ways of using the pretrained models in the downstream rearrangement task. First, we directly finetune the pretrained model on the ground truth correct rearrangement; shown after the dashed vertical line in Figure 7a. Here, the success rates for finetuned ELLM matches or outperform the baselines, especially if pretraining has already led to high success rates. Interestingly, we also find that the goal-conditioned ELLM variant consistently suffers a drop in performance when finetuning starts. We hypothesize this is due to the treatment of all suggested goals as a single string, so if any single goal changes between pretraining and finetuning the agent must relearn the goal embedding changes. Second, in Figure 7b we present results for directly training a new agent on the downstream task, using the frozen pretrained model as an exploratory actor during ϵ-greedy exploration. Once again, we observe that ELLM consistently matches or outperforms all baselines. We also see here that the KB-IM baselines are more competitive, suggesting that this training scheme is better suited for pretrained exploration agents that are not well-aligned to the downstream task.

5. Conclusions and Discussion

We have presented ELLM, an intrinsic motivation method that aims to bias exploration towards common-sense and plausibly useful behaviors via a pretrained LLM. We have shown that such priors are useful for pretraining agents in extrinsic-reward-free settings that require common-sense behaviors that other exploration methods fail to capture.

ELLM goes beyond standard novelty search approaches by concentrating exploration on common-sensical goals. This is helpful in environments offering a wide array of possible behaviors among which very few can said to be *plausibly useful*. It is less helpful in environments with little room for goal-based exploration, when human common-sense is irrelevant or cannot be expressed in language (e.g. fine-grained manipulation), or where state information is not naturally encoded as a natural language string.

LLM performance is sensitive to prompt choice. Even with a well-chosen prompt, LLMs sometimes make errors, often due to missing domain-specific knowledge. False negatives can permanently prevent the agent from learning a key skill: in Crafter, for example, the LLM never suggests creating wood pickaxes. There are multiple avenues to address this limitation: (1) combining ELLM rewards with other KB-IM rewards like RND, (2) prompting LLMs with descriptions of past achievements (or other feedback about



(a) *Pretraining and finetuning:* pretraining for 4M steps then fine-tuning for 1M steps on the ground truth correct arrangement.

(b) *Downstream evaluation:* Using the frozen pretrained exploration policies only for ϵ-greedy-style action selection for 1M steps.

Figure 7: *Housekeep:* Correct arrangement success rates on 4 object-receptacle task sets. Mean ± std over 5 seeds.

environment dynamics) so that LLMs can learn about the space of achievable goals, (3) injecting domain knowledge into LLM prompts, or (4) fine-tuning LLMs on task-specific data. While ELLM does not rely on this domain knowledge, when this information exists it is easy to incorporate.

ELLM requires states and transition captions. Our learned captioner experiments Figure 6 suggest we can learn these from human-labeled samples, but in some environments training this captioner might be less efficient than collecting demonstrations or hard-coding a reward function. Still, we are optimistic that as progress in general-purpose captioning models continues, off-the-shelf captioners will become feasible for more tasks. Lastly, suggestion quality improves considerably with model size. Querying massive LLMs regularly may be time- and cost-prohibitive in some RL environments.

As general-purpose generative models become available in domains other than text, ELLM-like approaches might also be used to suggest plausible visual goals, or goals in other state representations. ELLM may thus serve as a platform for future work that develops even more general and flexible strategies for incorporating human background knowledge into reinforcement learning.

6. Acknowledgements

YD and OW are funded by the Center for Human-Compatible Artificial Intelligence. CC received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101065949. This material is based upon work supported by the National Science Foundation under Grant No. 2212310 to AG and JA. OpenAI credits for GPT-3 access were provided through OpenAI's Researcher Access Program. We thank Sam Toyer and the members of the RLL for feedback on early iterations of this project.

References

Abid, A., Farooqi, M., and Zou, J. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 298–306, 2021.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., and Yan, M. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL `https://arxiv.org/abs/2204.01691`.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Aubret, A., Matignon, L., and Hassas, S. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

Baranes, A. and Oudeyer, P.-Y. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, pp. 1–17, 2019.

Chan, H., Wu, Y., Kiros, J., Fidler, S., and Ba, J. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546*, 2019.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Choi, K., Cundy, C., Srivastava, S., and Ermon, S. LMPriors: Pre-trained language models as task-specific priors. *arXiv preprint arXiv:2210.12530*, 2022.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *International conference on machine learning*, pp. 1039–1048. PMLR, 2018.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P., and Oudeyer, P.-Y. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33:3761–3774, 2020.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., and Efros, A. A. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

Hafner, D. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*, 2019.

Hill, F., Mokra, S., Wong, N., and Harley, T. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022a.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022b.

Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., and Agrawal, H. Housekeep: Tidying virtual households using commonsense reasoning. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 355–373, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19842-7.

Kong, Y. and Fu, Y. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, 130(5):1366–1401, 2022.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=10uNUgI5Kl.

Ladosz, P., Weng, L., Kim, M., and Oh, H. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

Lehman, J., Stanley, K. O., et al. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pp. 329–336, 2008.

Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P. J., Bernard, S., Beslon, G., Bryson, D. M., Cheney, N., Chrabaszcz, P., Cully, A., Doncieux, S., Dyer, F. C., Ellefsen, K. O., Feldt, R., Fischer, S., Forrest, S., Fŕenoy, A., Gagńe, C., Le Goff, L., Grabowski, L. M., Hodjat, B., Hutter, F., Keller, L., Knibbe, C., Krcah, P., Lenski, R. E., Lipson, H., MacCurdy, R., Maestre, C., Miikkulainen, R., Mitri, S., Moriarty, D. E., Mouret, J.-B., Nguyen, A., Ofria, C., Parizeau, M., Parsons, D., Pennock, R. T., Punch, W. F., Ray, T. S., Schoenauer, M., Schulte, E., Sims, K., Stanley, K. O., Taddei, F., Tarapore, D., Thibault, S., Watson, R., Weimer, W., and Yosinski, J. The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. *Artificial Life*, 26(2):274–306, 05 2020. ISSN 1064-5462. doi: 10.1162/artl_a_00319. URL https://doi.org/10.1162/artl_a_00319.

Linke, C., Ady, N. M., White, M., Degris, T., and White, A. Adapting behavior via intrinsic reward: A survey and empirical study. *Journal of Artificial Intelligence Research*, 69:1287–1332, 2020.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34, 2021.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

Lynch, C. and Sermanet, P. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heinz, I., and Roth, D. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv preprint arXiv:2111.01243*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Mokady, R., Hertz, A., and Bermano, A. H. Clipcap: Clip prefix for image captioning, 2021a. URL https://arxiv.org/abs/2111.09734.

Mokady, R., Hertz, A., and Bermano, A. H. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021b.

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N., Rocktäschel, T., and Grefenstette, E. Improving intrinsic exploration with language abstractions. *arXiv preprint arXiv:2202.08938*, 2022.

Nadeem, M., Bethke, A., and Reddy, S. Stereoset: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*, 2020.

Oudeyer, P.-Y. and Kaplan, F. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, pp. 6, 2009.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. 11 2019. URL http://arxiv.org/abs/1908.10084.

Sharma, P., Torralba, A., and Andreas, J. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

Stanić, A., Tang, Y., Ha, D., and Schmidhuber, J. Learning to generalize with object-centric agents in the open world survival game crafter. *arXiv preprint arXiv:2208.03374*, 2022.

Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G., and Cucchiara, R. From show to tell: a survey on deep learning-based image captioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Tam, A. C., Rabinowitz, N. C., Lampinen, A. K., Roy, N. A., Chan, S. C., Strouse, D., Wang, J. X., Banino, A., and Hill, F. Semantic exploration from language abstractions and pretrained representations. *arXiv preprint arXiv:2204.05080*, 2022.

Ten, A., Oudeyer, P.-Y., and Moulin-Frier, C. Curiosity-driven exploration. *The Drive for Knowledge: The Science of Human Information Seeking*, pp. 53, 2022.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Yao, S., Rao, R., Hausknecht, M., and Narasimhan, K. Keep calm and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*, 2020.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., and Lee, B. A survey of modern deep learning based object detection models. *Digital Signal Processing*, pp. 103514, 2022.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Crafter Pretraining Ablation

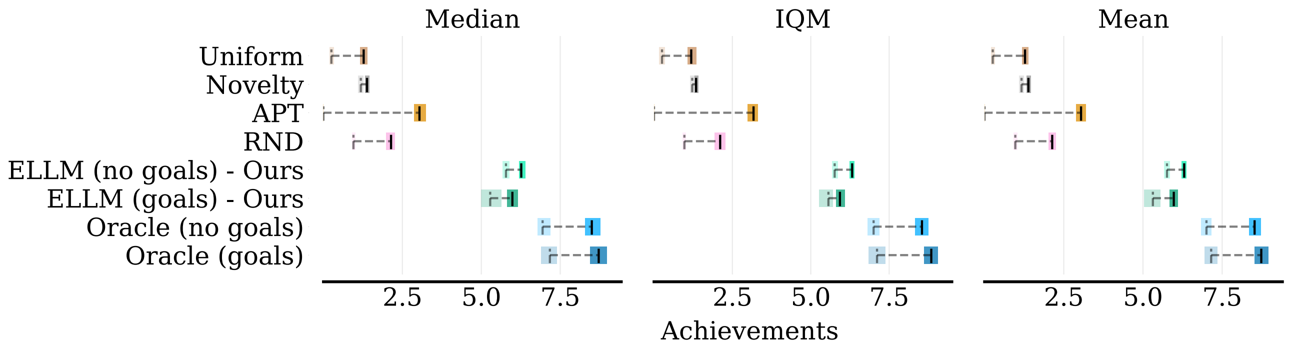


Figure 8: Number of ground truth achievements unlocked per episode at the end of pretraining. We show the median, interquartile mean (IQM) and mean of the achievements measured in 10 evaluation trials, each averaged over 10 episodes and 5 seeds (50 points) (Agarwal et al., 2021). Opaque bars represent variants leveraging textual observations in addition of visual ones and dashed lines represent the gap with vision-only variants (less opaque). We report results for each method described in Table 1. Results show that providing textual observations increases performance across all conditions.

B. Crafter Downstream Training

We finetune on seven downstream Crafter tasks plus the Crafter game reward:

- **Place Crafting Table** - agent must chop a tree and then create a crafting table. This is an easy task most agents will have seen during pretraining.
- **Attack Cow** - agent must chase and attack a cow. This is also an easy task often seen during pretraining in most methods.
- **Make Wood Sword** - agent must chop a tree, use it to make a crafting table, chop a second tree, use the wood at the crafting table to make a wood sword. This task could be achieved during the pretraining env, but many agents rarely or never achieved it because of the sheer number of prerequisites.
- **Mine Stone** - agent must chop a tree, use it to make a crafting table, chop a second tree, use the wood at the crafting table to make a wood pickaxe, seek out stone, and then mine stone. This task is so challenging that we replaced the fully sparse reward (where all pretraining methods fail) with a semi-sparse reward for achieving each subtask.
- **Deforestation** - agent must chop 4 trees in a row. This task tests whether having goal conditioning improves performance by directing the agent. During pretraining most agents will have chopped a tree, but novelty bias should deter agents from regularly chopping 4 trees in a row.
- **Gardening** Like above, this task tests the value of goal conditioning. The agent must first collect water and then chop the grass. Both skills maybe have been learned during pretraining, but never in sequence.
- **Plant Row** - agent must plant two plants in a row. This task is challenging because even a highly skilled ELLM agent cannot have learned this task 0-shot because the state captioner has no concept of a “row”.

C. Crafter Env Modifications

The default Crafter action space contains an all purpose “do” action which takes different actions depending on what object the agent is facing - for instance attacking a skeleton, chopping a tree, or drinking water.

We modify the action space to increase the exploration problem by turning the general ‘do’ action into more precise combinations of action verbs + noun arguments. Whereas ‘do’ previously was an all purpose action that could attack a

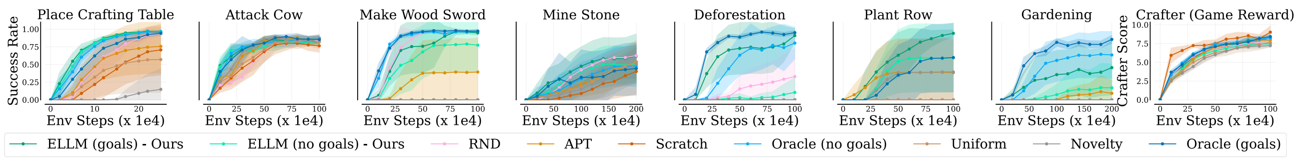


Figure 9: Goal completion success rate for different tasks in the Crafter environment. RL training uses sparse rewards. Each method trains an agent from scratch while using a pretrained policy for exploration. Each line shows the mean across 5 seeds with shaded stds.

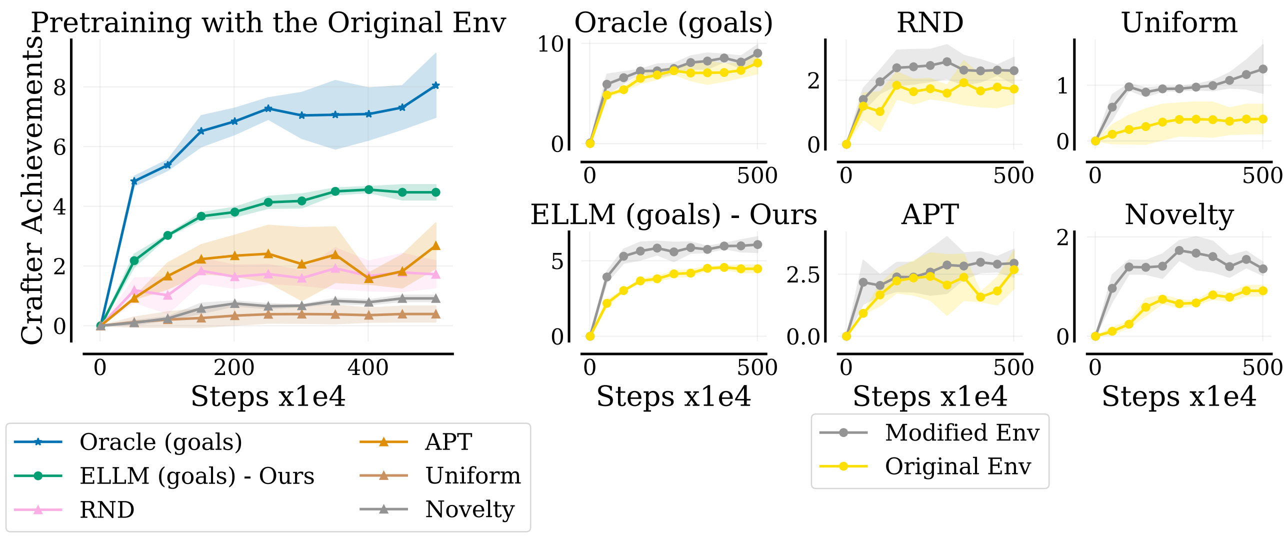


Figure 10: Training without the environment simplifications described in Section 4.1. Left: pretraining results (comparable to Figure 4). Right: original vs modified env performance. Curves average over 3 seeds with std shading. We see minor performance changes across most algorithms but no change in the rank-order of methods.

skeleton, chop a tree, or drink water, the agent must now learn to choose between the actions as arbitrary verb + noun combinations, ‘`attack skeleton`’, ‘`chop tree`’, ‘`drink water`.’ The exploration problem becomes more difficult as this larger combinatorial action space is not restricted to admissible actions and the agent could try to `drink skeleton` or `attack water`. Whereas the old action space was 17-dimensional, our new combinatorial one contains 260 possible actions. One way to impose human priors is to design the agent’s action space explicitly to disallow invalid combinations (e.g. `’drink’ + ’furnace’`). However, manually designing and imposing such constraints is also unlikely to be scalable. We hypothesize that our method, guided by common-sense knowledge from LLMs, will focus on learning to use only meaningful action combinations. For the purposes of the Novelty and Uniform baselines, which reward agents for achieving even nonsensical goals, we consider a goal “achieved” if the agent takes an action in front of the appropriate target object (e.g taking “drink furnace” in front of a furnace).

D. Crafter Prompt

```
Valid actions: sleep, eat, attack, chop, drink, place, make, mine

You are a player playing a game. Suggest the best actions the player can take based on the things
you see and the items in your inventory. Only use valid actions and objects.

You see plant, tree, and skeleton. You are targeting skeleton. What do you do?

- Eat plant

- Chop tree
```

`- Attack skeleton`

`You see water, grass, cow, and diamond. You are targeting grass. You have in your inventory plant. What do you do?`

`- Drink water`

`- Chop grass`

`- Attack cow`

`- Place plant`

In total, the actions present in the prompt make up:

- 6 / 10 (60%) of the good actions the ELLM agent receives.
- 6 / 21 (28.6%) of all rewarded actions the agent receives.
- 7 / 15 (50%) of all good action suggested.
- 7 / 51 (13.7%) of all actions suggested.

In future work, it would be interesting to explore how performance changes with fewer actions included in the prompt. As a preliminary experiment, we have found that pretraining performance is maintained if you provide a prompt with only one example of a list of valid goals. The list only contains two goals. Instead, we use more extensive instructions to tell the agent what good suggestions look like. See the prompt below and pretraining comparison in Figure 11. This new prompt comes with a decrease in the fraction of "Good" suggestions (shown in Table 4, showing that suggestion accuracy is not perfectly correlated with success.

New prompt: `Valid actions: sleep, eat, attack, chop, drink, place, make, mine`

`You are a player playing a Minecraft-like game. Suggest the best actions the player can take according to the following instructions.`

`1. Make suggestions based on the things you see and the items in your inventory.`

`2. Each scene is independent. Only make suggestions based on the visible objects, status, and inventory in the current scene.`

`3. Each suggestion should either be a single valid action, or a phrase consisting of an action and an object. (example: "Eat plant").`

`4. Do not make suggestions which are not possible or not desirable, such as ‘‘Eat skeleton’’.`

`5. Only make suggestions which are reasonable given the current scene (e.g. only ‘‘Eat plant’’ if a plant is visible).`

`6. You may suggest multiple actions with the same object, but do not duplicate list items.`

`7. Use your knowledge of Minecraft to make suggestions.`

`8. Prioritize actions which involve the object you are facing or which the agent hasn’t achieved before.`

`9. Each scene will include a minimum and maximum number of suggestions. Stick within this range.`

`New scene: You see plant, cow, and skeleton. You are facing skeleton. What do you do (include 1-2 suggestions)?`

`- Eat plant`

`- Attack skeleton`

`New scene: You see [INSERT CURRENT SCENE DESCRIPTION.] What do you do (include 2-7 suggestions)?`

	Suggested	Rewarded
Context-Insensitive	21.0%	0.8%
Common-Sense Insensitive	20.5%	54.8%
Good	34.1%	44.4%
Impossible	24.5%	0%

Table 4: Fractions of suggested and rewarded goals which are good, generated with the modified two-example prompt.

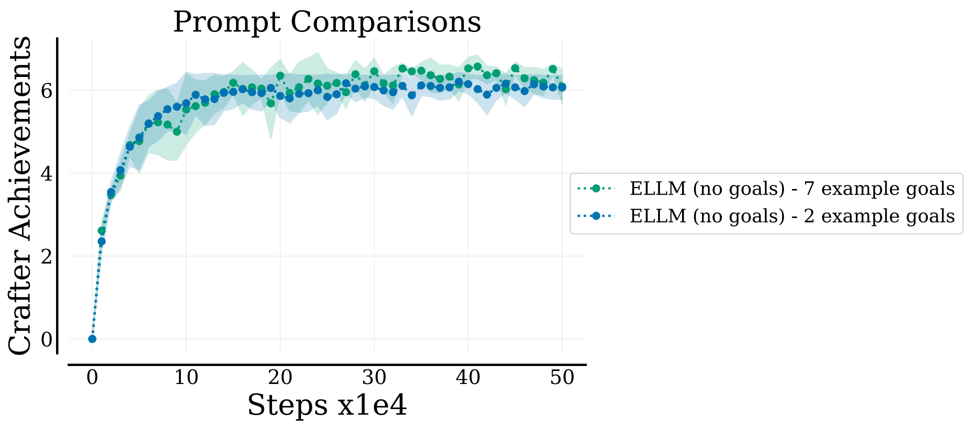


Figure 11: Comparison between performance of the prompt containing 7 suggested goals (used throughout the paper) and a modified prompt which only includes 2 examples.

E. Crafter Action Space

We expand the action space of Crafter to increase exploration difficulty and study if ELLM can learn to avoid nonsensical or infeasible actions. The full action space consists of just verbs (for actions that do not act on anything, such as `sleep`) or verb + noun combinations as follows:

- Verbs: `do nothing` (no noun), `move left` (no noun), `move right` (no noun), `move up` (no noun), `move down` (no noun), `sleep` (no noun), `mine`, `eat`, `attack`, `chop`, `drink`, `place`, `make`
- Nouns: `zombie, skeleton, cow, tree, stone, coal, iron, diamond, water, grass, crafting table, furnace, plant, wood pickaxe, stone pickaxe, iron pickaxe, wood sword, stone sword, iron sword`

For example, an action can be `drink water` or `drink grass`.

F. Housekeep Tasks

The original Housekeep benchmark features a large set of different household scenes and episodes with different objects and receptacles possibly instantiated. The ground truth correct object-receptacle placements were determined by crowdsourcing humans. However, since our focus is on RL pretraining, we do not make use of the mapping and planning methods from the original benchmark. To scope the problem for RL, we focus on the first 4 tasks with 5 different misplaced objects per task.

	Misplaced Objects
Task 1	peppermint, lamp, lantern, herring fillets, vase
Task 2	lamp, sparkling water, plant, candle holder, mustard bottle
Task 3	pepsi can pack, electric heater, helmet, golf ball, fruit snack
Task 4	chocolate, ramekin, pan, shredder, knife

Table 5: Objects per task

Name	Value (Crafter)	Value (Housekeep)
Frame Stack	4	4
γ	.99	.99
Seed Frames	5000	5000
n-step	3	3
batch size	64	256
lr	6.25e-5	1e-4
target update τ	1.0	1.0
ϵ-min	0.01	0.1
update frequency	4	4

Table 6: DQN Hyperparameters

G. Housekeep Prompt

```
You are a robot in a house. You have the ability to pick up objects and place them in new locations.
For each example, state if the item should be stored in/on the receptacle.

Should you store a dirty spoon in/on the chair: No.

Should you store a mixing bowl in/on the dishwasher: Yes.

Should you store a clean sock in/on the drawer: Yes.
```

H. Algorithmic Details

We make use of DQN (Mnih et al., 2013), with double Q-learning (Van Hasselt et al., 2016), dueling networks (Wang et al., 2016), and multi-step learning (Sutton et al., 1998).

For both environments, policies take in 84×84 images which are encoded using the standard Nature Atari CNN (Mnih et al., 2015). The image is then passed through a linear layer to output a 512 dimensional vector. If the policy is text-conditioned, we compute the language embedding of the state caption using `paraphrase-MiniLM-L3-v2` SBERT model (Reimers & Gurevych, 2019), and if the policy is goal-conditioned we similarly compute the language embedding of the goals $g_{1:k}$ using `paraphrase-MiniLM-L3-v2`. We encode all goals as a single text sequence as we did not see any improvement from encoding them each separately and summing or concatenating the embeddings. The image and text embeddings are then concatenated together before being passed to the Q-networks. Each of the value and advantage streams of the Q-function are parametrized as 3-layer MLPs, with hidden dimensions of 512 and ReLU nonlinearities.

In the Crafter environment, we swept over the following hyperparameters for the Oracle and Scratch (no-pretraining) conditions: learning rate, exploration decay schedule, and network update frequency. We then applied these hyperparameters to all conditions, after confirming that the hyperparameters were broadly successful in each case.

For Housekeep pretraining, we swept lr $\in [1e-3, 1e-4, 1e-5]$, ϵ-min $\in [0.1, 0.01]$, and batch size $\in [64, 256]$.

I. Hard-coded Captioner Details

Crafter The state captioner is based on the template shown in Figure 3 (left). This consists of three components: the observation, the items, and the agent status.

- Observation: We take the underlying semantic representation of the current image from the simulator. Essentially this maps each visible grid cell to a text description (e.g. each tree graphic is mapped to "tree"). We then take this set of descriptions (i.e. not accounting for the number of each object) and populate the "observation" cell of the template.
- Items: We convert each of the inventory items to the corresponding text descriptor, and use this set of descriptions to populate the "item" cell of the template.
- Health status: We check if any of the health statuses are below maximum, and if so, convert each to a corresponding language description (e.g. if the hunger status is < 9, we say the agent is "hungry").

The transition captioner uses the action labels. Each action maps to a predefined verb + noun pairing directly (e.g. "eat cow").

Housekeep The state captioner is based on the template shown in Figure 3 (right). We use the simulator's semantic sensor to get a list of all visible objects, receptacles, and the currently held object. The transition captioner is also based on the simulator's semantic sensor, which indicates which receptacles the visible objects are currently in.

J. Learned Crafter Captioner

The captioner is trained with a slightly modified ClipCap algorithm (Mokady et al., 2021a) on a dataset of trajectories generated by a trained policy using the PPO implementation from Stanić et al. (2022). Visual observations at timestep t and $t+1$ are embedded with a pretrained and frozen CLIP ViT-B-32 model (Radford et al., 2021) and concatenated together with the difference in semantic embeddings between the two corresponding states. Semantic embeddings include the inventory and a multi-hot embedding of the set of objects present in the local view of the agent. This concatenated representation of the transition is then mapped through a learned mapping function to a sequence of 10 tokens. Finally, we use these 10 tokens as a prefix and pursue decoding using a pretrained and frozen GPT-2 to generate the caption (Radford et al., 2019). We train the mapping from transition representation to GPT tokens on a dataset of 847 human labels and 900 synthetic labels obtained by sampling from a set of between 3 and 8 different captions for each each distinct type of transitions. Instead of the programmatic "`chop tree`" and "`attack zombie`," labeled captions involve fully-formed sentences: "`You collected a sapling from the ground`," "`You built a sword out of wood`," or "`You just stared at the sea`." Because of this additional linguistic diversity, we compare captions to goals with a lower cosine similarity threshold of .5.

Imperfect captioners can cause learning issues in two different ways: (1) they can generate wrong captions all together and (2) they can generate a valid caption that still lead to faulty reward computations. If the caption is linguistically too different from the achievement it captions, the similarity-based reward might not be able to pick it up (false negative reward). This same linguistic variability might cause the reward function to detect the achievement of another achievement that was not achieved (false positive reward). Figure 12 measures all these issues at once. For each row, it answers: what is the probability that the reward function would detect a positive reward for each of the column achievements when the true achievement is the row label? The false negative rate is 11% on average (1 - the diagonal values), with a much higher false negative rate for *chop grass* (100%). Indeed, human caption mentioned the outcome of that action instead of the action itself (*collect sapling*); which the similarity-based reward fails to capture. The false positive rate (all non diagonal values) is significant here: the agent can get rewarded for several achievements it did not unlock. This often occurs when achievements share words (e.g. wood, stone, collect). This indicates a difficulty of the semantic similarity to differentiate between achievements involving these words.

K. Crafter LLM Analysis

Table 2 shows that the actions agents are rewarded for are dominated by good actions (66.5%) and bad actions (32.4%). This makes sense; impossible actions can never be achieved. Most context-insensitive cannot be achieved (e.g. "drink water" suggested when no water is present). We consider an action a "success" by checking whether the agent attempted a particular action in front of the right object, so the agent occasionally is rewarded when it takes a context-insensitive action in the appropriate physical location but without the necessary prerequisites (e.g. mining stone without a pickaxe).

Table 7 gives examples of LLM suggestions in Crafter.

Suggestion Type	**Examples**
Good	chop tree, attack skeleton, place plant
Context-Insensitive	make crafting table (without wood), mine stone (without a pickaxe or not by stone)
Common-Sense-Insensitive	mine grass, make diamond, attack plant
Impossible	make path, make wood, place lava

Table 7: Classification accuracy of LLM for each Housekeep task (left column is true positives, right column is true negatives).

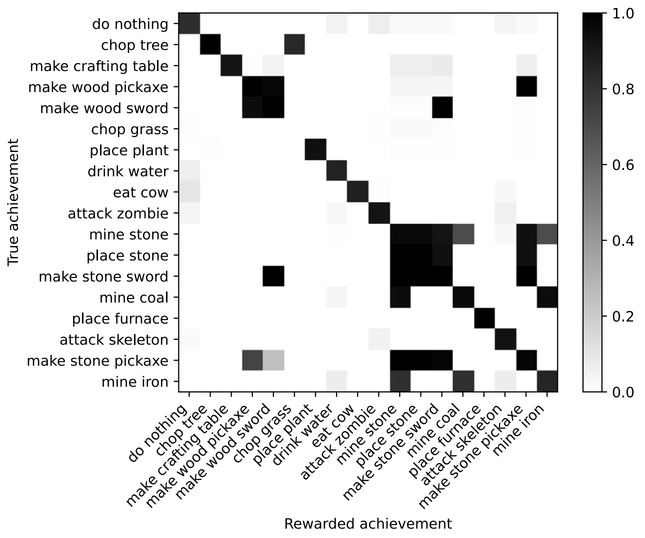


Figure 12: Reward confusion matrix. Each row gives the probability that any of the column achievement is detected when the row achievement is truly unlocked. For instance, in row 2, when the agent chops a tree, with high probability the agent will be rewarded for the "chop tree" and "chop grass" actions. Tested on trajectories collected from an expert PPO policy, each row estimates probabilities using between 27 and 100 datapoints (27 for *mine iron*, the rarest achievements). Rows do not sum to one, as a given achievement, depending on its particular caption, could potentially trigger several rewards.

L. Novelty Bonus Ablation

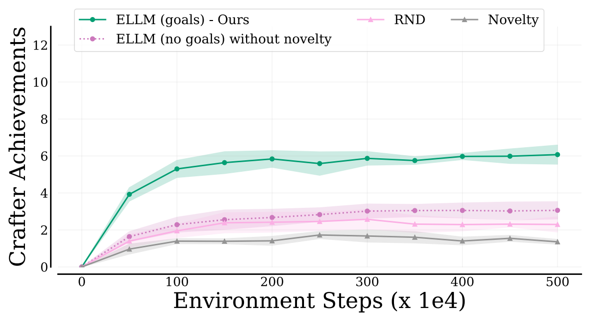
We ablate the importance of ELLM's novelty bias in Figure 13 by allowing the agent to be rewarded repeatedly for achieving the same goal. We see that without the novelty bonus the agent only learns to repeat a small set of easy goals and fails to explore diversely.

M. Analysis of Downstream Training Approaches

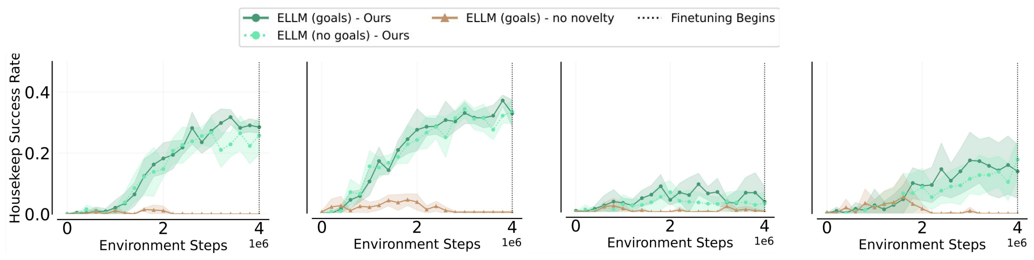
We explored two methods for using exploratory policies: *finetuning*, where the weights of the exploration policy are finetuned and the *guided exploration* method, where a new policy is trained from scratch and the pretrained policy is used for ϵ-greedy exploration.

We found that in Housekeep both methods are effective for ELLM (Figure 7a and Figure 7b). However, in Crafter we found that the finetuning method performed poorly across all methods (ELLM, baselines, and oracles). Often, we observed that early in finetuning, the agent would unlearn all of its previous useful behaviors, including moving around the environment to interact with objects. We hypothesize that this due to a mismatch in the density and magnitude of rewards between pretraining and finetuning. When the finetuning agent finds it is achieving much lower than the expected return for taking its typical actions, it down-weights the likelihood of taking those actions and unlearns its previous skills. We found that decreasing the learning rate, freezing early layers of the network, manually adjusting finetuning rewards to be at the same scale as pretraining rewards, and decreasing the initial exploration rate partially mitigated this problem. However, these also decrease the sample efficiency and/or performance at convergence of the finetuned policy compared to a training-from-scratch baseline. In Figure 14), across all methods this method is less reliable than the guided exploration method (Figure 5).

These findings are consistent with our Housekeep findings. In that environment, the ELLM pretraining task (achieving object placements suggested by a LLM) and the finetuning task (achieving object placements suggested by humans) are similar enough we only see minor dips in performance when finetuning starts. However, the RND and APT baselines have a greater pretrain-finetune mismatch, and we observe those methods did comparatively better with the guided exploration method.



(a) Crafter pretraining runs (similar to Figure 4), including the "ELLM without novelty" ablation where ELLM's novelty bias is removed.



(b) Housekeep pretraining runs (similar to Figure 7a), including the "ELLM without novelty" ablation where ELLM's novelty bias is removed.

Figure 13

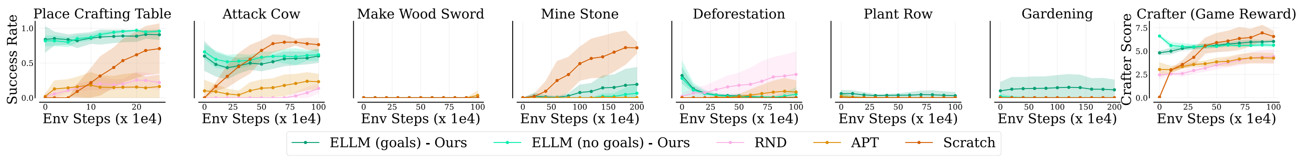


Figure 14: Success rates across training for each of the seven downstream tasks in the Crafter environment. Each run finetunes the pretrained agent using a lower learning rate than used during pretraining ($2e-5$). Plots show mean $\pm$ std for 5 seeds

N. Additional Baselines

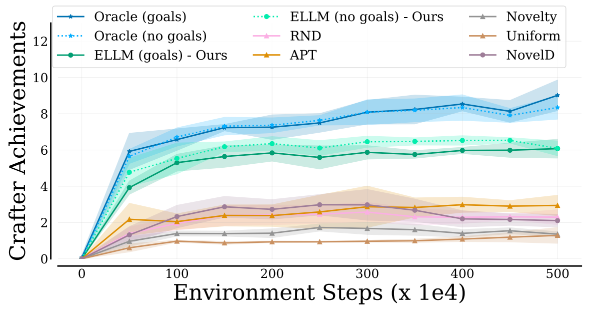
We also include experiments with NovelD (Zhang et al., 2021) in Figure 15, a state-of-the-art exploration method which uses an estimate of state novelty to reward the agent for moving to more novel states. During pretraining, we find it performs similarly to the other prior-free intrinsic motivation methods.

O. Code and Compute

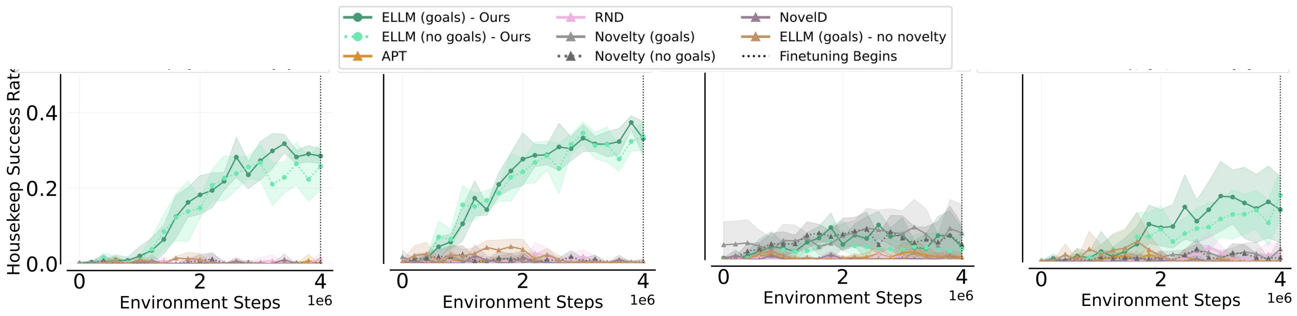
All code will be released soon, licensed under the MIT license (with Crafter, Housekeep licensed under their respective licenses).

For LLM access, we use OpenAI's APIs. Initial experiments with the smaller GPT-3 models led to degraded performance, hence choosing Codex and Davinci for our experiments. Codex is free to use and Davinci is priced at \$0.02/1000 tokens. We find caching to be significantly helpful in reducing the number of queries made to the API. Each API query takes .02 seconds, so without caching a single 5-million step training run would spend 27 hours querying the API (and far more once we hit the OpenAI rate limit) and cost thousands of dollars. Since we cache heavily and reuse the cache across runs, by the end of our experimentation, were make almost no API queries per run.

We use NVIDIA TITAN Xps and NVIDIA GeForce RTX 2080 Tis, with 2-3 seeds per GPU and running at roughtly



(a) Crafter pretraining curve as in Figure 4, including NovelD baseline



(b) Housekeep pretraining curves as in Figure 7a, including NovelD baseline

Figure 15: Additional pretraining curves including NovelD.

100ksteps/hour. Across all the ablations, this amounts to approximately 100 GPUs for pretraining.

P. Societal Impact

While LLMs priors have been shown to exhibit impressive common-sense capabilities, it is also well-known that such models are highly prone to harmful social biases and stereotypes (Bender et al., 2021; Abid et al., 2021; Nadeem et al., 2020). When using such models as reward functions for RL, as in ELLM, it is necessary to fully understand and mitigate any possible negative behaviors that can be learned as a result of such biases. While we focus on simulated environments and tasks in this work, we emphasize that more careful study is necessary if such a system is deployed to more open-ended learning in the real world. Potential mitigations with ELLM specifically can be: actively filtering LLM generations for harmful content before using them as suggested goals, prompting the LM with guidelines about what kinds of prompts to output, and/or using only the closed-form ELLM variant with more carefully constrained goal spaces.