Assessing the Zero-Shot Capabilities of LLMs for Action Evaluation in RL

Eduardo Pignatelli
University College London
e.pignatelli@ucl.ac.uk

Johan Ferret
Google DeepMind

Tim Rocktäschel
University College London
& Google DeepMind

Edward Grefenstette
University College London

Davide Paglieri
University College London

Samuel Coward
University of Oxford

Laura Toni
University College London

Abstract

The temporal credit assignment problem is a central challenge in Reinforcement Learning (RL), concerned with attributing the appropriate influence to each actions in a trajectory for their ability to achieve a goal. However, when feedback is delayed and sparse, the learning signal is poor, and action evaluation becomes harder. Canonical solutions, such as *reward shaping* and *options*, require extensive domain knowledge and manual intervention, limiting their scalability and applicability. In this work, we lay the foundations for Credit Assignment with Language Models (CALM), a novel approach that leverages Large Language Models (LLMs) to automate credit assignment via reward shaping and options discovery. CALM uses LLMs to decompose a task into elementary subgoals and assess the achievement of these subgoals in state-action transitions. Every time an option terminates, a subgoal is achieved, and CALM provides an auxiliary reward. This additional reward signal can enhance the learning process when the task reward is sparse and delayed without the need for human-designed rewards. We provide a preliminary evaluation of CALM using a dataset of human-annotated demonstrations from MiniHack, suggesting that LLMs can be effective in assigning credit in zero-shot settings, without examples or LLM fine-tuning. Our preliminary results indicate that the knowledge of LLMs is a promising prior for credit assignment in RL, facilitating the transfer of human knowledge into value functions.

1 Introduction

The Credit Assignment Problem (CAP) [Minsky, 1961, Sutton, 1984, Pignatelli et al., 2024] is a fundamental challenge in RL. It typically involves determining the contribution of each action to the final outcome, a process crucial for accurate policy evaluation. Effective Credit Assignment (CA) enables agents to learn useful associations between actions and outcomes, and provides useful directions to improve the policy.

However, when rewards are dispensed only at the end of a task [Efroni et al., 2021], as it is often the case, the feedback becomes sparse and delayed, making CA particularly challenging. In such scenarios, rewarding events are rare, and Deep Reinforcement Learning (Deep RL) agents often struggle to convert occasional successes into a robust decision-making process. To exacerbate the issue, RL agents typically begin with no prior knowledge (*tabula rasa*) and must learn the nuances

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

and intricacies of complex tasks from scratch. The lack of controlled experimental conditions, such as the ability to observe counterfactuals, makes it difficult for them to distinguish between correlation and causation. As a result, tasks that are usually easy to solve for humans become hard to address for an RL agent.

To address these challenges, many methods incorporate prior human knowledge into RL systems. Two techniques are canon: reward shaping [Ng et al., 1999, Gupta et al., 2022] and Hierarchical Reinforcement Learning (HRL) [Al-Emran, 2015, Sutton et al., 1999] via options [Sutton et al., 1999]. Reward shaping involves providing an additional synthetic reward to guide the agent's actions when natural rewards are uninformative. HRL decomposes complex tasks into simpler ones (*options*), training agents to achieve intermediate objectives that provide a signal while the Markov Decision Process (MDP) would not. Despite their effectiveness, these methods require extensive human input, making them costly and difficult to scale across different environments.

Recently, LLMs have emerged as a useful tool to transfer human knowledge into computational agents, either through planning [Dalal et al., 2024], expressing preferences [Klissarov et al., 2023], or grounding their abstract knowledge into practical solutions [Huang et al., 2023, Carta et al., 2023]. Notably, these models have produced strong results in causal reasoning tasks [Jin et al., 2023] with performances comparable to humans [Kıcıman et al., 2023]. These results suggest that LLMs could be an effective, supplementary tool to distinguish between correlation and causation more effectively than traditional methods used in early stages of RL training.

With these results, a natural question arises: "*Can the knowledge encoded in LLMs serve as a useful prior for CA in RL?*" Inspired by the successes of LLMs, we introduce CALM, a general method to perform CA with LLMs using reward shaping. We hypothesize that the prior knowledge of a LLM can provide valuable signals that improve CA in RL, and propose a way to transfer these priors into the agent's value function. On this assumption, CALM leverages a pretrained LLM to break down tasks into smaller, composable subgoals and determine if a state-action-state transition achieves a subgoal. This provides an additional reward signal to enhance RL algorithms, and effectively automates reward shaping by substantially reducing the involvement of humans in the training loop.

We present a preliminary evaluation of the efficacy of CALM in zero-shot settings, with no examples and no finetuning. We collect a dataset of demonstrations from MiniHack [Samvelyan et al., 2021] and use it to compare the performance of LLMs against human annotations. Our results indicate that LLMs are a viable means to transfer common human knowledge into value functions, and can be effective in automating reward shaping. This bodes well for the prospect to improve CA in the full RL problem.

2 Related work

LLMs for RL. Recent advancements have shown the potential of pretrained LLMs in enhancing RL agents. Paischer et al. [2022, 2024] used CLIP encodings to improve the state representations of Partially-observable MDPs (POMDPs). Yao et al. [2020], Du et al. [2023] investigated the ability of pretrained LLMs to improve exploration. Huang et al. [2023], Carta et al. [2023] grounded the abstract knowledge of these models and their capabilities into practical RL tasks. LLMs have been used for planning, either directly as world models [Huang et al., 2022, Wang et al., 2023, Singh et al., 2023, Brohan et al., 2023, Dasgupta et al., 2023, Shah et al., 2023, Zhong et al., 2020, 2022] or by writing code [Liang et al., 2022]. Unlike these methods we use pretrained LLMs as a critic: the LLM provides an *evaluation* of an action for how useful it is to achieve a goal in the future. Among the methods above, Du et al. [2023] is the only method to use subgoals, but these are used to condition a goal-oriented policy, rather than as a critic.

LLMs for reward shaping. Carta et al. [2022], Goyal et al. [2019] explore the advantages of using pure language abstractions for reward shaping, but do not use a pretrained LLMs and its prior knowledge. Kwon et al. [2023] use the responses of LLMs as a reward signal, but the investigation is limited to conversational environments.

LLMs for knowledge transfer. Another set of studies used intrinsic rewards to transfer the prior knowledge of an LLM to a value function. Wu et al. [2024] used LLMs to provide an auxiliary reward signal in Atari [Bellemare et al., 2013], based on the information contained in a game manual.

Unlike this study, we use subgoals to extract the reward signal, and we do not focus on incorporating external knowledge material, but rely on the LLM's prior knowledge to solve the task. Klissarov et al. [2023] constructed a reward function from the LLM's preferences over NetHack [Küttler et al., 2020] in-game messages only. Instead, our method incorporates the full observation, does not use preferences, and does not require a separate stage to fit the preference set, but uses the LLM's output directly.

In short, none of these methods proposes to generalise reward shaping with hierarchical skills using pretrained LLMs. Unlike the methods above, we use pretrained LLMs as a critic: we aim to uncover cause-effect relationships between actions and goals by both breaking down a task into valuable subgoals and then acting as a reward function for them. This provides an intermediate signal to shape the agent's behaviour when rewards are sparse and delayed.

3 Preliminaries

We consider the problem of learning to solve POMDPs. A POMDP is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \mu, \mathcal{O}, O, \gamma)$. Here $\mathcal{S}$ is the state space with elements s. $\mathcal{A}$ is the action space of elements a. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a deterministic, bounded reward function. $\mu : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function. $\mathcal{O}$ is the space of all observations, and $O : \mathcal{S} \to \mathcal{O}$ is an observation function, mapping a state s to a partial observation o. $\gamma \in [0, 1]$ is the discount factor.

To best isolate the CAP from other problems, we focus only on environments with a discrete action space, and deterministic state transitions. To evaluate the capabilities of LLMs in environments where the CAP is hard, we only consider tasks where the reward signal is delayed. Here, the reward function is 0 everywhere, and 1 when a goal state is reached.

To start the investigation, we evaluate the LLM only in language settings, and do not consider multimodal (text, image, audio, video) settings. For this reason, we consider only environments with an observation kernel that maps to a textual codomain, $O : \mathcal{S} \to \mathcal{T}$, where $\mathcal{T}$ is a set of sequences of characters.

Finally, we consider a black box, pretrained LLM, that takes an input text and maps it to a finite set of output characters. We consider only open-weights models that can fit an NVIDIA A100 80Gb in either 16 bits floating point or 4 bits integer mode. We assume that the LLM has enough preliminary knowledge of the MiniHack environment to recognise valuable actions that progress towards a win.

4 Methods

We set out to design a general method to assign credit in RL using LLMs that can generalise to multiple tasks with little human input. Next, we formalise the method, discuss its assumptions and provide details on the protocols we use to evaluate it.

4.1 Reward shaping

Among the available CA techniques, we focus on *reward shaping* [Ng et al., 1999], due to both its effectiveness in assigning credit and its limitations to generalisation related to the costs of human involvement in the training loop. Reward shaping aims to address the scarcity of learning signals by introducing an auxiliary reward function, the *shaping function*:

$$\tilde{r}_{t+1} = \tilde{R}(s_t, a_t, s_{t+1}). \tag{1}$$

Here, s_t is the state at time t, a_t is the action taken in that state, s_{t+1} is the resulting state, and $\tilde{r}_{t+1}$ is the auxiliary reward collected after taking a_t in s_t. This reward is added to the original reward signal $R(s_t, a_t, s_{t+1})$ to obtain the new, shaped reward

$$r_{t+1} = R(s_t, a_t, s_{t+1}) + \tilde{R}(s_t, a_t, s_{t+1}). \tag{2}$$

If there exist a function $\phi : \mathcal{S} \to \mathbb{R}$ such that $\tilde{R}(s_t, a_t, s_{t+1}) = \phi(s_{t+1}) - \phi(s_t)$, then the set of optimal policies is preserved, and the shaping function is also a *potential function* [Ng et al., 1999]. In the following, we consider the more general case of non-optimality preserving functions.

For example, in key-door environments, a common testbed for CA methods [Hung et al., 2019, Mesnard et al., 2021], the agent must reach a reward behind a locked door, which can only be opened if the agent possesses a key. Here, the agent has clear subgoals: (*i*) go to the key, (*ii*) pick it up, (*iii*) go to the door, (*iv*) unlock it, (*v*) go to the reward. Achieving these subgoals sequentially leads to optimal behaviour. However, the agent struggles to recognise this hierarchical pattern due to the lack of immediate feedback from the environment. This is particularly true in the early stages of training, when behaviour is erratic, and two optimal actions can be separated by a long sequence of random ones. Providing intermediate feedback for each achievement often improves the agent's performance [Gupta et al., 2022], and the ability of $\tilde{R}$ to produce an instantaneous signal indicating progress is crucial for better CA. Thus, reward shaping can significantly accelerate the learning process in environments with sparse or delayed rewards.

However, designing an effective shaping function is challenging. The function should be carefully designed to provide useful guidance without leading to unintended behaviours. This often calls for incorporating domain knowledge or heuristic information about the task, and requires deep task and environment knowledge. Such knowledge may not be readily available or easily codifiable, limiting the applicability of reward shaping in diverse or unknown environments. This process is complex and time-consuming, and it might not always be possible to devise a reward function that incentivizes learning, is computationally cheap, and general enough to adapt to various tasks. Improving this limitation could enable broader use of reward shaping and enhance CA in deep RL.

4.2 LLMs as shaping functions

Encouraged by the recent successes of LLMs in RL [Klissarov et al., 2023] and of using language to abstract skills [Jiang et al., 2019, Jacob et al., 2021, Sharma et al., 2021, Mu et al., 2022], we explore whether these models can offer a valid alternative to humans in the reward shaping process. Our goal is to produce a function that, given a description of the task and a state-action-state transition, produces a binary signal indicating whether the action makes progress towards solving the task or not:

$$LLM : desc(\mathcal{M}) \times desc(\mathcal{S} \times \mathcal{A} \times \mathcal{S}) \to \mathbb{B}. \tag{3}$$

Here, LLM is a pretrained LLM; $desc(\mathcal{M})$ is a natural language description of the POMDP (the task); $desc(\mathcal{S} \times \mathcal{A} \times \mathcal{S})$ is a textual representation of the transition, not necessarily in natural language (for example, a grid-arranged text), and $\mathbb{B} = \{0, 1\}$ is the Boolean domain. In this scenario, the LLM acts as a critic: its role is to *evaluate* the action a_t in the transition (s_t, a_t, s_{t+1}) based on the heuristics that we describe next.

We operationalise the idea using the notion of *options* [Sutton et al., 1999]. An *option* is a temporally extended action and consists of two elements: an intra-option policy $\pi_i : \mathcal{S} \to \Delta(\mathcal{A})$, and a termination condition $\beta : \mathcal{S} \to \mathbb{B}$.[1]

To develop an intuition of options, it is useful to visualise one as a macro-action: a set of actions that, taken together, have precise semantics. For example, in our key-to-door example, one useful option to consider is to *pick up the key*. This macro action includes a set of primitive actions – the set of actions to navigate to the key and the action *pickup* – and a termination condition – whether the key is picked up. For the purpose of our analysis, this termination is crucial, as it signals that the subtask has been successfully achieved.

We exploit this idea to build our shaping function, set up a single-turn conversation, and prompt the model to perform two subtasks:

(i) To identify a set of useful options in the environment, by breaking down the task into a sequence of shorter subgoals. These options, and more specifically their termination, effectively constitutes our set of subgoals, since a subgoal is achieved when the option terminates (a key is picked up).

(ii) Determine whether an option terminated (thus, if a subgoal is achieved) in the transition (s_t, a_t, s_{t+1}).

Every time an option terminates, we augment the task reward with the subtask reward as according to our reward shaping rule, $\tilde{R}(s_t, a_t, s_{t+1}) = \beta(s_{t+1})$.

[1] We consider $\mathcal{S}^+ = \mathcal{S}$ and omit the initiation set $\mathcal{S}^+$

In essence, Equation (3) aims to mimic a human supervising an RL agent's decisions, acting as an auxiliary critic. Decomposing the task into multiple subgoals allows each sub-achievement to correspond to a small step towards success, and composing all the subgoals sequentially results in successful behaviour. Since achieving a subgoal is contingent on achieving all the preceding ones, the number of subgoals achieved quantifies the agent's progresses. To develop an intuition of the idea, subgoals can be thought of as levels; gaining a level at the current time indicates progress in achieving a specific goal in the future. This process of *actualisation*, where an action is evaluated for its future potential to achieve a goal, characterises the function as a CA method [Pignatelli et al., 2024].

4.3 Experimental protocol

The viability of CALM in online RL settings depends on the quality of the assignments provided by the LLM. Good quality assignments – signals that reinforce optimal actions – can improve the performance of an RL algorithm. Thus, we provide a preliminary evaluation of CALM on an offline dataset of demonstrations.

Environment. We focus on the KeyRoom environment, a canonical testbed for CA methods [Hung et al., 2019, Mesnard et al., 2021, 2023] originally proposed in Minigrid [Chevalier-Boisvert et al., 2018]. We choose its MiniHack version, for it provides a textual representation of the observations that can be fed to a language system. The game presents a top-down view of a grid-like environment with two rooms. The agent starts in the first room, where a key is located. It must pick up the key and use it to unlock the door to the second room, where a reward is located. We consider two types of observations:

1. **Cropped observations.** a top-down, north-facing, 9x9 crop around the agent, which is known to improve the performance in standard RL benchmarks on Nethack [Küttler et al., 2020].
2. **Game screens.** A top-down, north-facing, 21x79 grid showing the entire game scene, including an in-game message and a set of statistics of the current state. We also refer to these as *human* observations, since they reproduce the conditions of human game play.

Both observations are partial, despite containing different amounts of information. We consider a discrete action set: *go north*, *go east*, *go south*, *go west*, *pickup*, *apply*. The reward function is deterministic, providing a reward of 1 if the agent reaches the goal tile and 0 otherwise. Transitions are also deterministic.

Dataset. We collect 256 one-step transitions $d_t = (s_t, a_t, s_{t+1})$ using a random policy. Given a set of subgoals $\mathcal{G} \subset (\mathcal{S} \times \mathcal{A} \times \mathcal{S})$, a transition d_t can then be classified as either achieving a subgoal $g \in \mathcal{G}$ or not. This produces categories $\mathcal{C} = \{c_i \,:\, 0 \leq i \leq |\mathcal{G}| + 1\}$, one for each subgoal, and an additional one when no subgoal is achieved. To characterise the abilities of an LLM to assign credit accurately, we produce a balanced dataset where each goal appears with equal probability.

Composing the prompt. For each transition we then compose a prompt using the following structure:

1. `<ROLE>` specifies the role we ask the LLM to simulate.
2. `<ENVIRONMENT-DESCRIPTION>` describes the RL environment, the source of the gameplay.
3. `<SYMSET>` is a list reporting Nethack wiki entries[2] of what each symbols in the grid represents.
4. `<TASK-DESCRIPTION>` specifies the overall goal of the agent, and does not contain information about subgoals.
5. `<SUBGOALS>` contains either a list of subgoals to achieve, or asks the LLM to produce one.
6. `<INSTRUCTIONS>` tasks the agent to determine whether a subgoal is achieved in the trajectory presented in `<TRANSITION>`.

[2] https://nethackwiki.com/wiki/Symset

Example prompt

```
The environment is MiniHack.

I will present you with a short extract of a gameplay. At each timestep, symbols represent the following items:
- "." represents a floor tile.
- "|" can represent either a wall, a vertical wall, an open door.
- "-" can represent either the bottom left corner (of a room), bottom right corner (of a room), wall, horizontal wall,
wall, top left corner (of a room), op right corner (of a room).
- "+" represents a closed door. Doors can be locked, and require a key to open.
- "(" represents a useful item (pick-axe, key, lamp...)
- "<" represents a ladder or staircase up.
- ">" represents a ladder or staircase down.

The task of the agent is to win the game.

First, based on your knowledge of NetHack, break down the task of the agent into subgoals.
Then, consider the following game transition, which might or might not contain these subgoals.
Determine if any of the subgoals is achieved at Time: 1 or not.


Report your response in a dictionary containing the name of the subgoals as keys and booleans as value. For example:
‘‘‘python
{
    <name of goal>: <bool>,
}

Observation Sequence:

<gameplay>
Time: 0
Current message:

      - - - -
      | . . |
      | . . |
- - + - . < |
| . . . @ . |
| . ( . . . |
- - - - - - -


Time: 1
Current message:

      | . . |
      | . . |
- - + - . < |
| . . . . . |
| . ( . @ . |
- - - - - - -



</gameplay>

I will not consider anything that is not in the dictionary.
You have only one shot at this, and you cannot ask for clarifications.
```

Prompt 1: Example of a prompt for instruction verification. Here, goals are provided externally from a human.

7. Finally, `<OUTPUT-FORMAT-REQUEST>` requests the output in a format that can be easily parsed, for example, a python dictionary.

Prompt 1 shows a concrete instantiation of this structure, where goals are provided as part of the input. Here, the role is not specified, exhorting the LLM to play a generic role, and the environment description (*The environment is MiniHack*) is minimal. In the symset – the list of symbols with their meaning – the descriptions are extracted from the wiki (`https://nethackwiki.com/wiki/Symset`). The task is as generic as possible (*to win the game*), and it is followed by the set of predetermined subgoals (*pick up the key* and *unlock the door*). The instructions and the request for an appropriate output format follow on that. Finally, we enclose the transition within a `<gameplay>` tag, and remark that this is a single-turn conversation to avoid the model asking additional clarifications. Notice that we separate each cell in the observation with a whitespace to ensure that each cell (plus their whitespace) corresponds to a separate token. We discuss this more in depth in Appendix D.1, and provide more details and variations of prompts in Appendix A. To develop an intuition of the role covered by the model, we encourage the reader to scan over them before proceeding.

Models. We use pretrained, open-weights large language models that can fit a 80Gb A100 Nvidia GPU in either 16 bits brain floating point [Dean et al., 2012] or 4 bits integer weights representations. When models cannot fit in memory, we use their NF4 [Dettmers et al., 2023] quantised equivalent. These models are marked with an asterisk (*) in the tables below. All the models are finetuned for instructions following, and tokens are deterministically sampled using a greedy policy.

Annotations. For each transition a human annotator produces a term of reference for comparison. The annotator is presented with each prompt in the dataset, without any further instructions. We then record the annotator's answer, and use it as a term of reference for the LLM's responses. Since the prompt has a correct answer, these are not subjective evaluations, but rather direct verification, with little room for interpretation.

Metrics. We then compare the LLM's annotations with the human ones. The response is a true positive if both the LLM and the human annotator identify that a subgoal is achieved. It is a false positive (a *hallucination*) if the LLM identified it, but the human has not; a false negative (a *miss*) if the human identified one, but the agent has not. This effectively casts the problem as classification, with the set of classes $\mathcal{C}$, as described in the dataset description. We then compare the LLM's hypotheses with the human responses as ground truth, and report accuracy, F1 score, precision and recall.

5 Experiments, results, and discussion

To evaluate the effectiveness of LLMs in CA for RL, we consider environments with textual representations. We assume that the LLM has sufficient knowledge of the game to evaluate actions. While this assumption might be strong for NetHack, it is reasonable for MiniHack, where tasks are simplified yet challenging models of common NetHack scenarios, requiring only partial knowledge.

Based on the set of experimental conditions described above, we then consider a spectrum of settings requiring progressively less input from humans. We start by providing the LLM with: *(a)* cropped observations focused around the agent; *(b)* an effective, predetermined set of subgoals; We then proceed to progressively relax these conditions to: *(a)* gamescreen observations; *(b)* allowing the LLM to discover useful subgoals autonomously. These conditions are set to replicate the conditions of a human playing the game.

5.1 Can LLMs understand goal specifications and verify option termination?

This experiment aims to assess whether a pretrained LLM can function as a reward function when subgoals are provided externally. We provide the LLM with the environment name, `MiniHack`, and a list of two subgoals: `pick up the key` and `unlock the door`. We specify that the goal of the agent is simply to *win the game* [Jeurissen et al., 2024], and ask it to determine if each subgoal has been achieved in the transition. Prompt 4 shows an example prompt for this experiment.

We present results for multiple pretrained LLMs, using both cropped observations and full game screens. The purpose of the comparison is not to determine a winning model. It is, instead, to understand whether the ability to assign credit to single transitions is in the spectrum of capabilities of existing open-weights LLMs. This will lay the foundation for applying the method in full RL settings.

We report results in Tables 1 and 2, and draw the following two insights. First, LLMs, except *gemma-1.1-2b-it*, probably due to its small size, are generally effective in recognising when an instruction has been successfully completed in a state-action-state transition. This shows their ability to understand goal specifications and to recognise when an option terminates due to completion. We also noticed that *c4ai-command-r-plus* degenerates into outputting *false* for most transitions, most probably due to quantisation.

Second, restricting the field of view of the observation helps improve performance. This is most likely due to observations being more concise, and avoiding the information to drown among a high number of tokens. This also seems to increase the lower bound, and the performance of models drastically failing with *human* observations greatly improves, especially *gemma-1.1-2b-it*.

Annotator	F1 ↑	Accuracy ↑	Precision ↑	Recall ↑	TP ↑	TN ↑	FP ↓	FN ↓
Human	**1.00**	**1.00**	**1.00**	**1.00**	**171**	**85**	**0**	**0**
Mixtral-8x7B-Instruct-v0.1*	**0.74**	0.67	0.77	**0.73**	**124**	47	38	**47**
gemma-1.1-7b-it	0.73	**0.70**	0.91	0.61	105	75	10	66
Meta-Llama-3-70B-Instruct	0.66	0.65	**0.97**	0.50	85	82	3	86
Meta-Llama-3-8B-Instruct	0.64	0.64	0.95	0.49	83	81	4	88
c4ai-command-r-v01*	0.60	0.57	0.80	0.49	83	64	21	88
Mistral-7B-Instruct-v0.2	0.48	0.54	0.96	0.32	55	83	2	116
gemma-1.1-2b-it	0.00	0.33	0.00	0.00	0	**85**	**0**	171
Random	0.33	0.33	0.33	0.33				

Table 1: Performance of LLM annotations against human annotations with **game screen** observations and with the subgoals **provided** in the prompt. Models marked with an asterisk (*) are quantised to NF4 format. TP stands for *true positives*, TN for *true negatives*, FP for *false positives*, and FN for *false negatives*. Rows sorted by F1 score.

Annotator	F1 ↑	Accuracy ↑	Precision ↑	Recall ↑	TP ↑	TN ↑	FP ↓	FN ↓
Human	**1.00**	**1.00**	**1.00**	**1.00**	**171**	**85**	**0**	**0**
Mixtral-8x7B-Instruct-v0.1*	**0.78**	**0.70**	0.78	**0.77**	**132**	48	37	**39**
gemma-1.1-7b-it	0.76	0.69	0.79	0.73	124	52	33	47
gemma-1.1-2b-it	0.76	0.68	0.76	**0.77**	131	43	42	40
c4ai-command-r-v01*	0.75	0.69	0.81	0.70	120	57	28	51
Meta-Llama-3-70B-Instruct	0.63	0.58	0.76	0.54	92	56	29	79
Meta-Llama-3-8B-Instruct	0.61	0.61	0.92	0.46	79	78	7	92
Mistral-7B-Instruct-v0.2	0.61	0.62	**0.96**	0.45	77	**82**	**3**	94
Random	0.33	0.33	0.33	0.33				

Table 2: Performance with **cropped** observations and with the subgoals **provided** in the prompt.

5.1.1 Can LLMs suggest effective options?

In this experiment, we evaluate whether LLMs can autonomously suggest effective options. Instead of providing a predetermined list, we ask the LLM to break down the task into subgoals and verify whether these subgoals have been achieved. Despite only a small change on the surface, removing some key information from the prompt intensively tests the LLM's knowledge of NetHack. More importantly, it stresses the ability of the models to come up with a viable and effective hierarchy of subgoals such that, if reinforced, produces useful signals for progress.

This setting is more complex but also more general, as it replicates the amount of information typically available to a human player. Prompt 5 shows an example prompt for this experiment. As for the previous experiment, we evaluate the performance of different models using both cropped and human observations.

Annotator	F1 ↑	Accuracy ↑	Precision ↑	Recall ↑	TP ↑	TN ↑	FP ↓	FN ↓
Human	**1.00**	**1.00**	**1.00**	**1.00**	**171**	**85**	**0**	**0**
Meta-Llama-3-70B-Instruct	**0.82**	**0.72**	0.71	**0.96**	**165**	19	66	**6**
Meta-Llama-3-8B-Instruct	0.80	0.70	0.72	0.89	153	26	59	18
gemma-1.1-7b-it	0.77	0.66	0.71	0.85	145	25	60	26
Mixtral-8x7B-Instruct-v0.1*	0.74	0.64	0.71	0.76	130	33	52	41
Mistral-7B-Instruct-v0.2	0.57	0.48	0.63	0.53	90	32	53	81
c4ai-command-r-v01*	0.56	0.52	0.71	0.47	80	52	33	91
gemma-1.1-2b-it	0.00	0.33	0.00	0.00	0	**85**	**0**	171
Random	0.33	0.33	0.33	0.33				

Table 3: Performance with **game screen** observations and with **autonomously discovered** subgoals.

Results in Table 3 indicate that LLMs can effectively suggest subgoals when presented with game screen observations, and that these subgoals align with those identified by humans. Models like *Meta-Llama-3-70B-Instruct* and *Meta-Llama-3-8B-Instruct* come close to human performance, suggesting that LLMs can effectively use the additional information to suggest and validate subgoals.

Annotator	F1 ↑	Accuracy ↑	Precision ↑	Recall ↑	TP ↑	TN ↑	FP ↓	FN ↓
Human	**1.00**	**1.00**	**1.00**	**1.00**	**171**	**85**	**0**	**0**
Meta-Llama-3-70B-Instruct	**0.83**	**0.75**	0.75	0.93	159	33	52	12
gemma-1.1-7b-it	0.81	0.70	0.71	**0.95**	**163**	17	68	**8**
Mixtral-8x7B-Instruct-v0.1*	0.72	0.62	0.71	0.74	127	32	53	44
Mistral-7B-Instruct-v0.2	0.65	0.54	0.66	0.64	109	28	57	62
c4ai-command-r-v01*	0.60	0.52	0.68	0.54	92	41	44	79
gemma-1.1-2b-it	0.47	0.52	**0.89**	0.32	55	**78**	**7**	116
Meta-Llama-3-8B-Instruct	0.45	0.39	0.57	0.37	63	38	47	108
Random	0.33	0.33	0.33	0.33				

Table 4: Performance with **cropped** observations and with **autonomously discovered** subgoals.

These results bode well for applications of CALM where human input, while still considerably smaller than in canonical reward shaping, is still expensive to collect.

When transitioning to cropped observations (Table 4) LLMs perform worse. This is most likely due to a misalignment between the subgoals proposed by the models and the ones of the ground truth. We did not observe any substantial difference in how different models propose subgoals and in the types of subgoals they suggest. Most models correctly identify *item collection* and *locating objects*, such as stairs, monsters and keys. They often include "going to <object>" instructions as subgoals. We provide examples of such prompts in Appendix B.

While this evaluation can be unfair, since we compare the LLM's response with the set of subgoals the human identified, it still tells whether the LLM way of reasoning about a task align with the human one. These elements, together with the ability of LLMs to verify if a subgoal is achieved, suggest that LLMs can be an effective means to transfer human knowledge into value functions.

5.2 Conclusions, limitations, and future work

In this study, we explored whether LLMs can be a useful means to transfer human knowledge into the value function of RL agents. By focusing on reward shaping, we highlighted its limitations in scalability due to the cost of human involvement. To mitigate these costs, we proposed replacing humans with LLMs, leveraging their ability to decompose tasks into shorter subgoals. Preliminary results from an offline dataset of MiniHack demonstrations suggest that LLMs are effective in verifying subgoal achievement and align with those proposed by humans. This suggests the potential of using LLMs to enhance CA in RL.

Limitations. While preliminary results are promising, they are limited by the scope of the current evidence. We did not conduct RL experiments to validate the method in online RL settings. The dynamic nature of online RL could pose unique challenges not present in offline settings. Additionally, despite KeyRoom being representative of the CA challenges, and a common testbed for CA, evaluating the method in a broader range of environments would provide more comprehensive evidence of its robustness and applicability.

The method also has inherent limitations. Environments must provide observations in the form of text. The LLM must hold enough knowledge of the game to evaluate actions. While this can be a mild assumption for MiniHack, it can be an obstacle for environments requiring more specialised knowledge, such as Nethack [Küttler et al., 2020] or Crafter [Hafner, 2021, Matthews et al., 2024]. Finally, the LLM relies solely on their prior knowledge and does not incorporate new knowledge while assigning credit, limiting their adaptability and accuracy over time.

Future work. Future work should focus on addressing these limitations. Validating the approach in online RL settings and exploring its applicability to a broader range of environments can tell if CALM can enhance the learning process of RL agents in practice. A natural extension of this work is to generalise the method beyond text-only observations. Baumli et al. [2023] follows this line of research, testing the capability of Vision Language Models (LLMs) to evaluate the completion of an instruction from pixels alone. The instruction completion question corresponds to ours in the LLMs domain. Finally, a closed feedback loop where CALM helps improve the policy, the policy provides new information to the LLM, and the LLM incorporates this information to improve its CA ability could help scale to more complex problems requiring specialistic knowledge.

References

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961. ISSN 00968390. doi: 10.1109/JRPROC.1961.287775.

Richard S Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts, 1984.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=bNtr6SLgZf. Survey Certification.

Yonathan Efroni, Nadav Merlis, and Shie Mannor. Reinforcement learning with trajectory feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7288–7295, 2021.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

Mostafa Al-Emran. Hierarchical reinforcement learning: a survey. *International journal of computing and digital systems*, 4(02), 2015.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Murtaza Dalal, Tarun Chiruvolu, Devendra Singh Chaplot, and Ruslan Salakhutdinov. Plan-seq-learn: Language model guided RL for solving long horizon robotics tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hQVCCxQrYN.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. 2023.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. Grounded decoding: Guiding text generation with grounded models for robot control. *arXiv preprint arXiv:2303.00855*, 2023.

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

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. CLadder: Assessing causal reasoning in language models. In *NeurIPS*, 2023. URL https://openreview.net/forum?id=e2wtjx0Yqu.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*, 2023.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=skFwlyefkWJ.

Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *International Conference on Machine Learning*, pages 17156–17185. PMLR, 2022.

Fabian Paischer, Thomas Adler, Markus Hofmarcher, and Sepp Hochreiter. Semantic helm: A human-readable memory for reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep calm and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*, 2020.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

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

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

Ishita Dasgupta, Christine Kaeser-Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. Collaborating with language models for embodied reasoning. *arXiv preprint arXiv:2302.00763*, 2023.

Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. Rtfm: Generalising to new environment dynamics via reading. In *ICLR*, pages 1–17. ICLR, 2020.

Victor Zhong, Jesse Mu, Luke Zettlemoyer, Edward Grefenstette, and Tim Rocktäschel. Improving policy learning via language dynamics distillation. *Advances in Neural Information Processing Systems*, 35:12504–12515, 2022.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2022. *URL https://arxiv. org/abs/2209.07753*, 3, 2022.

Thomas Carta, Pierre-Yves Oudeyer, Olivier Sigaud, and Sylvain Lamprier. Eager: Asking and answering questions for automatic reward shaping in language-guided rl. *Advances in Neural Information Processing Systems*, 35:12478–12490, 2022.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*, 2019.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models, 2023.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M Mitchell. Read and reap the rewards: Learning to play atari with the help of instruction manuals. *Advances in Neural Information Processing Systems*, 36, 2024.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10(1):5223, 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-13073-w. URL `https://doi.org/10.1038/s41467-019-13073-w`.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, pages 7654–7664. Proceedings of Machine Learning Research, 2021.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Athul Paul Jacob, Mike Lewis, and Jacob Andreas. Multitasking inhibits semantic drift. *arXiv preprint arXiv:2104.07219*, 2021.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

Jesse Mu, Victor Zhong, Roberta Raileanu, Minqi Jiang, Noah Goodman, Tim Rocktäschel, and Edward Grefenstette. Improving intrinsic exploration with language abstractions. *Advances in Neural Information Processing Systems*, 35:33947–33960, 2022.

Thomas Mesnard, Wenqi Chen, Alaa Saade, Yunhao Tang, Mark Rowland, Theophane Weber, Clare Lyle, Audrunas Gruslys, Michal Valko, Will Dabney, Georg Ostrovski, Eric Moulines, and Remi Munos. Quantile credit assignment. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24517–24531. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/mesnard23a.html`.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. arxiv 2023. *arXiv preprint arXiv:2305.14314*, 2023.

Dominik Jeurissen, Diego Perez-Liebana, Jeremy Gow, Duygu Cakmak, and James Kwan. Playing nethack with llms: Potential & limitations as zero-shot agents. *arXiv preprint arXiv:2403.00690*, 2024.

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

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*, 2024.

Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, et al. Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*, 2023.