Large Language Models in Reinforcement Learning - A Comparison of Methods

Adrian Duric
Master Student, Dept. of Informatics
The Faculty of Mathematics and Natural Sciences
Oslo, Norway
adriandu@ifi.uio.no

Gregor Kajda
Master Student, Dept. of Informatics
The Faculty of Mathematics and Natural Sciences
Oslo, Norway
grzegork@ifi.uio.no

Jonatan Hoffmann Hanssen
Master Student, Dept. of Informatics
The Faculty of Mathematics and Natural Sciences
Oslo, Norway
jonatahh@ifi.uio.no

***Abstract*—Reinforcement Learning (RL) algorithms suffer when rewards are sparse and the state-action space is large. Even tasks which seem relatively simple can prove intractable if the completion of the task requires subtasks to be completed first, or if the reward only comes when the entire task has been completed. In such cases, random exploration is unlikely to lead the agent to discover a solution to the problem. The apparent simplicity of such problems is often due to human intuition, which allow us to quickly see possible solutions to a diverse set of problems. Large Language Models (LLMs) are trained on large corpora of human written text, and have been shown to capture parts of this intuition in many tasks. In recent years, LLMs have been successfully used to aid RL agents in more efficient exploration, and have allowed them to solve problems which previous methods have been unable to. We compare different methods for integrating an LLM into the Deep Learning RL algorithm Proximal Policy Optimization (PPO), and compare their efficiency against each other.**[1]

***Index Terms*—large language models, reinforcement learning, minigrid, proximal policy optimization**

I. Introduction

One of the central challenges of reinforcement learning is that rewards are often both extremely rare and delayed, which means that optimizing a reinforcement learning algorithm requires significant trial and error [Brunton and Kutz, 2022, 423]. Furthermore, in many problems the state-action spaces are enormous, meaning that uniform exploration is unlikely to find good solutions in a reasonable amount of time. Many methods have been developed to deal with this issue, for example by introducing auxiliary reward functions that use domain knowledge to reward actions which are considered good, or which reward the agent for learning novel skills. However, hand picking which actions to reward can lead to imitation rather than optimal behaviour, and novelty is not always useful [Du et al., 2023, 1]. In recent years, LLMs have shown remarkable capabilities in problem solving and planning [Bubeck et al., 2023], qualities which traditional RL agents often lack. Thus, a new area of research has emerged, which attempts to use the vast amount of human knowledge encoded in these models to improve the performance of reinforcement learning algorithms [Luketina et al., 2019]. These LLM assisted RL agents have been able to outperform state-of-the-art RL methods in many problems [Zhang et al., 2023], [Du et al., 2023], [Li et al., 2022].

These papers explore different methods of integrating LLMs into a standard reinforcement learning training loop, from altering the policy directly to introducing an auxiliary reward function. In this paper, we compare both methods in the same environment using the same deep reinforcement algorithm (PPO), and compare their results to each other to explore how LLMs best can be integrated into RL.

[1]Code available at GitHub

II. Background and Related Work

RL informed by natural language is a relatively new field, where research before 2018 had been limited to relatively small corpora or synthetic language [Luketina et al., 2019, 1]. With the introduction of Large Language Models, many new possibilities have opened up, and in recent years there have been many successful integrations of these models into traditional RL [Zhang et al., 2023], [Du et al., 2023], [Li et al., 2022].

A. Large Language Models as Policy in Reinforcement Learning

Among the methods proposed to integrate the LLM, many include ways of making the LLM influence the policy of the RL agent. [Zhang et al., 2023] considers open-ended learning algorithms in which the value of the agent attempting certain tasks can be evaluated by estimating the the interestingness of the task. The notion of interestingness is then thought of as what task would intuitively be interesting for a human to try in the context of learning in an environment. Some main factors contributing to this interestingness are how likely one is to succeed in doing the task, as well as whether learning the given task may increase the likelihood of succeeding in other tasks. Considering that LLMs are trained on extremely large amounts of human-written text containing human knowledge and intuition, this leads to the idea that if the LLM was told to propose interesting actions for the RL agent to learn in its environment, it could be able to communicate human intuition directly to the agent. This would in turn let the agent learn with increased sample efficiency as the human intuition-based

input it receives would make it decide to perform intuitively smarter actions, particularly in the early stages of exploring its environment.

The algorithm proposed in this paper involves prompting the LLM with the agent's state observation, a list of tasks the agent already does well, and a list of available tasks for the LLM to deem as either "interesting" or "boring". After sorting available tasks into interesting and boring ones, it assigns sampling weights to them, giving much higher weights to interesting tasks than boring ones. Thus, it directly influences the agent's policy by altering the probabilities of the possible actions. In [Li et al., 2022], another paper proposing using the LLM as a policy influencer, has a similar approach in which the state observation is tokenized and passed to the LLM, though in their experiment, goals and action histories are also tokenized and passed. Here, too, the LLM response is fed back into a task-specific RL model, which in turn influences policy probabilities of certain actions being taken. In general, many of the novel methods suggested involve the LLM suggesting actions for the agent to perform in a given context, and its output influencing the agent policy.

B. Large Language Models as Reward in Reinforcement Learning

Another proposed method of involving LLMs in RL, is to have it influence the reward the agent receives for taking certain actions in certain states. In particular, [Du et al., 2023] cites the infrequency of rewards as a common bottleneck in RL algorithms, due to how long exploratory trajectories often have to be before the agent reaches some desirable state. This holds true especially for large, complex environments, where the probability of reaching favorable states among a huge number of non-favorable states becomes even smaller. Similarly to the papers proposing LLMs to influence policy, this paper too seeks to make use of human intuition and knowledge present in the vast training data that state-of-the-art LLMs have been trained upon, to make the agent prioritize exploring plausibly useful behaviors first, based on some intelligent forethought provided by the LLM.

However, rather than doing so through directly influencing action probabilities in the policy, the paper suggests rewarding the agent if it takes actions that are semantically similar to actions suggested by an LLM. For example, the LLM might recommend the agent to "cut down a tree". If the agent performs the "chop" action at a tree, this action can be described as "chop tree". Using a language model, we can create encodings of these sentences and compare their similarity mathematically. At certain timesteps, the LLM is prompted with a state observation from the agent, and is asked to suggest actions for the agent to take. If the agent then takes actions which are semantically similar to the suggested actions, it receives a reward. Thus, the proposed algorithm makes use of LLMs in two ways: first to generate what the LLM perceives as desirable actions, then later to assess similarity between a natural-language description of the agents action and the previously suggested actions. The first use case in particular utilizes the presence of human intuition in LLMs, as a human would often be able to intuitively figure out what is desirable to achieve in some environment.

III. Methods

A. Problem Formulation

We train a PPO model in a Partially Observable Markov Decision Process (POMDP). A POMDP is defined by a tuple $(S, A, T, R, O, \Omega, \gamma)$. Here, $s \in S$ denotes the state the environment is in and $a \in A$ denotes an action the agent can take. Based on the state and the action taken, the agent receives an observation $o \in \Omega$ which depends on the new environment state and the action taken by $O(o|s, a)$. $T(s'|s, a)$ denotes the state transfer function, which gives a new state based on the previous state and action. $r \in R$ is the reward given by the environment given an action and a state, while γ is the discount factor. The environment we use is part of Minigrid [Chevalier-Boisvert et al., 2023].

B. Approach

To evaluate the effectiveness of the integration of LLMs into Reinforcement Learning algorithms, we use the MiniGrid library. Here, we handpick specific environments with varying levels of complexity, to evaluate the effectiveness of our proposed techniques against a standard Actor-Critic PPO, and against each other.

1) Standard Proximal Policy Optimization: Our baseline is the PPO algorithm, which is a policy gradient method that has been established as a state-of-art approach to solving reinforcement learning problems over the last few years. PPO simply uses an Actor-Critic network which is updated using policy gradient methods, with the modification that the policy is not allowed to change too drastically when updating. With this simple concept, PPO has outperformed previous methods such as Deep-Q Networks and A2C, while being relatively simple to implement [Schulman et al., 2017].

For the hyperparameters used in our PPO implementation, see the Appendix.

2) PPO with LLM Integrated in the Policy Function: In this case, the LLM is integrated as part of the PPO algorithm, directly influencing the actions taken. This is done as follows: First, the observation is captioned and given as input to the LLM, which is prompted to suggest an action. Then, all actions that the agent could take given the current state are described in natural language. All possible actions are compared with the suggested action, and given a value which denotes their semantic similarity. These values are joined with the logits of the actions, thus influencing the probability weights of each action being taken so that semantically similar actions are weighted higher than others. This is a very direct approach to influencing the agent; if an action is favoured by the LLM in a timestep, this will affect the actions immediately by increasing the chance that this action is sampled. Thus, this method functions analogously to how a coach instructs a player, giving direct suggestions to the agent immediately as the agent is about to perform actions at all timesteps of each episode.

Throughout the paper, we will refer to this as LLM policy influencing.

3) PPO with LLM Integrated in the Reward Function: In this method, the agent is trained using the same method as in the baseline. The agent's state observation is encoded to natural text, and sent as part of a prompt to the LLM at every timestep. After being given the state observation, the LLM is asked to respond with a list of recommended actions for the agent to perform. The agent then acts independently of the LLM for that timestep, as dictated by the baseline RL algorithm it is trained with. As it does so, its chosen action is encoded to natural text. Then, another LLM is used for semantic similarity comparison, comparing the proposed action to the action performed by the agent. Finally, high semantic similarity gives an additional pseudo-reward which is added to the reward of the environment. Compared to the policy influencing method above, this approach is less direct, as the policy is only indirectly affected by the LLM through these additional rewards, which only take effect when the policy is updated through backpropagation after an entire episode is completed. Thus, this method functions more as an advisor, influencing the policy in a more long term capacity than the coaching of the policy influencer. Throughout the paper, we will refer to this as LLM reward shaping.

4) PPO with both LLM reward shaping and LLM policy influencing: Finally, we combine the two LLM integrations, hoping to combine the direct influence of our policy integration with the more indirect reward shaping. Thus, we are affecting the policy in two ways: Through reward shaping, we are causing the policy to update to better attain the aforementioned pseudo-rewards, which will encourage completion of subgoals that are necessary to complete the actual goal of the environment, but which do not give out intermediate rewards. Through policy influencing, we are nudging the agent to take favorable actions by directly altering the action probabilities, which will make it more likely that the agent completes the given task and receives an actual reward from the environment. Thus, the agent will more quickly discover the correct policy. We call this the **Actor-Advisor-Coach-Critic** method, or **A2C2** for short.

C. Language Models

In all these cases, we need two language models; one to generate the suggestions and one to generate semantic similarity values.

For text generation, we have decided to utilize an LLM from the Llama open-source family of language models developed by Meta AI, first released in Febraury, 2023. Specifically, we choose Llama 2, which has been trained using 40% more data than its predecessor Llama 1, and has double the context length [Touvron et al., 2023]. In the experiments presented below, we use the smallest version of Llama 2, with 7 billion parameters, which we download to run locally and reduce the bottleneck effect and cost of having to call an API to a model hosted by someone else. The reason for not using any of the larger Llama 2 models is simply the hardware limitation we face, with the 7B model requiring 19GB of VRAM to be run.

Some research papers propose fine-tuning of LLMs as a means of leveraging the capabilities of the language models for decision-making purposes even further [Carta et al., 2023]. Our plan of action, on the other hand, aims at evaluating the efficacy of the LLM without any previous pre-tuning. From the perspective of reinforcement learning, this is the appropriate course of action, as the LLM should not possess any previous knowledge about the task at hand, just like the agent should not. In addition, the act of fine-tuning LLM is a rather time-consuming process, requiring the generation and structuring of a new dataset, and also vast hardware resources.

Instead of fine tuning the LLM, we use few-shot learning when prompting, giving the LLM a few examples of observations and correct corresponding action recommendations. This is an effective way to guide the LLM for our specific purposes without requiring us to retrain the entire network.

For generating semantic similarity values, we follow [Du et al., 2023] and use Sentence-Bert [Reimers and Gurevych, 2019] for generating encoding vectors, and perform a cosine similarity calculation of the two sentence encodings. This gives a value between -1 and 1, where two sentences with the exact same encoding have a cosine similarity of 1.

IV. Experiments

We perform experiments to compare different methods of integrating LLMs with the PPO algorithm, specifically comparing their ability to guide the early exploration of an RL model in a reward-sparse environment. Even state-of-art RL algorithms may struggle in such environments. Therefore, it is the perfect arena to see if the high generalization capabilities of pre-trained Large Language Models will lead to better results over using only RL methods.

A. Minigrid Environments

For our experiments, we choose the Minigrid library. The Minigrid library contains simple grid world environments. An important characteristic of all these environments is that they only give reward on the completion of the goal of the environment. In other words, the environment only ever gives out a single reward per episode, given that the agent completes its task. Goals in Minigrid are often simple tasks, such as "unlock the door", "go to the goal", "pick up the red chest", etc. Completing these goals often requires the completion of subtasks; going to the goal might require opening a door, and opening the door might require picking up a key. These environments are partially observable, which means that the agent only receives a seven by seven observation grid, not the entire grid. Figure 1, shows a typical Minigrid environment. In this case, the agent (the red arrow) only sees the left room, and does not know where the goal is.

With no intermediate rewards, more complex environments like the one shown in Figure 3 quickly become difficult to learn for traditional RL methods. In addition, some environments can only be effectively solved by relying on language (for

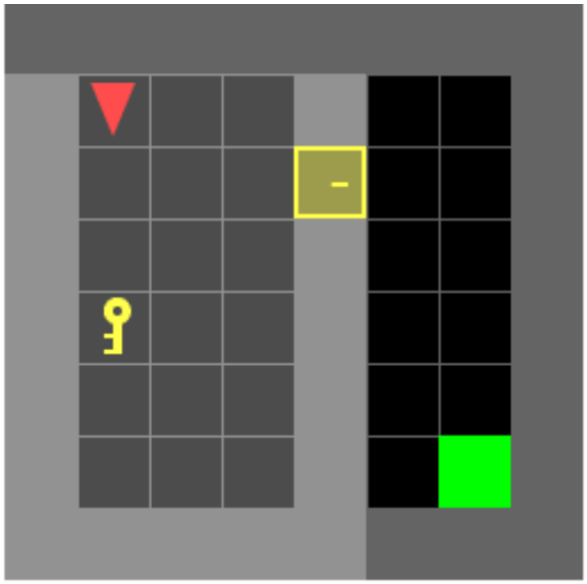

Fig. 1. The Minigrid DoorKey environment

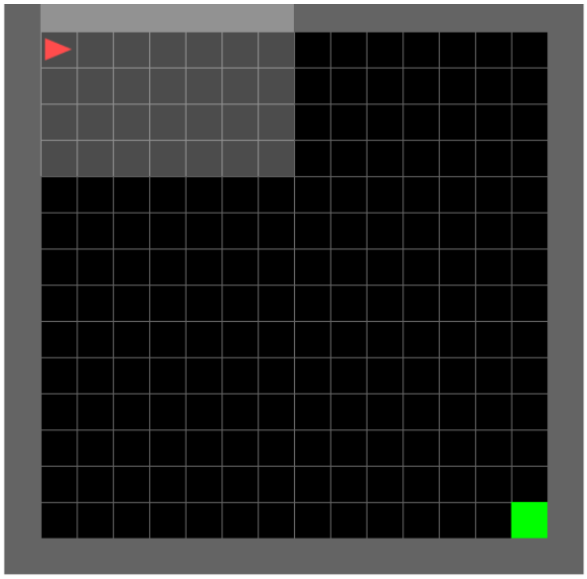

Fig. 2. The Minigrid Empty Environment

example, environments where the goal text describes a specific object to be picked up, which changes every time), while others can be solved without (for example, if the goal always is to pick up a key). We only use environments that can be solved without language, so that the baseline PPO algorithm is not at an unfair disadvantage.

B. Our chosen environments

We select three different Minigrid environments with increasing complexity; Empty, DoorKey and UnlockPickup. For each environment we test our four different models; PPO with no LLM integration, PPO with LLM reward shaping, PPO with LLM policy influencing and PPO with both reward shaping and policy influencing (A2C2). All methods use the same hyperparameters for the PPO, which can be found in the appendix. For each of these methods, we run PPO for 2000 (200 for Empty) episodes of 1024 steps, and track the reward[2] received for each one. We do this eight times per method, and compare the average results per episode against each other.

1) Empty environment: This environment is the simplest possible, consisting of an empty room with a single goal in the bottom right corner. This environment contains no stochasticity; the agent and goal always start in the same positions. Despite its simplicity, there is still potential for LLM integration to improve the early phases of training, because the room is 15 by 15 tiles, which means that receiving any reward requires a significant amount of exploration. The environment is seen in Figure 2.

2) DoorKey environment: This environment consists of two rooms separated by a locked door. Figure 1 shows an example configuration. The agent starts in one room, while the goal square is in the other. The agent must pick up a key in the starting room, and use it on the door to gain access to the room containing the goal. This environment thus contains two subgoals; picking up the key and using it to unlock the door.

3) UnlockPickup environment: This environment is quite similar to the DoorKey environment. Figure 3 shows an example configuration. The difference is that after unlocking the door, the agent must now pick up a chest instead of walking to a goal square. Because the agent can only hold one item at a time, this requires dropping the key on an empty square, which introduces another subtask. Chests can also be opened like doors, but this removes the chest from the game and thus causes and instant failure of the task, as it can no longer be picked up. Thus, the agent must learn that using a key on a door is a valuable subtask, while using it on a chest is a bad idea. This added complexity makes it a much more difficult task than the previous two.

V. Results

This section outlines the results of our experiments in each of our selected environments. We also show example outputs from the LLM and examine how these may influence our results.

A. Results per environment

1) Empty environment: In this simple environment, we see from Figure 4 that both LLM reward shaping, LLM policy influencing and A2C2 outperform the baseline model. Averaging over all episodes for all eight runs, we find that reward shaping increases the mean reward received by 10% over the baseline, with a lower standard deviation in rewards received per run of 0.12. Policy influencing scores even higher at 32%

[2]The true environment reward, not including pseudo-rewards given by the LLM in reward shaping

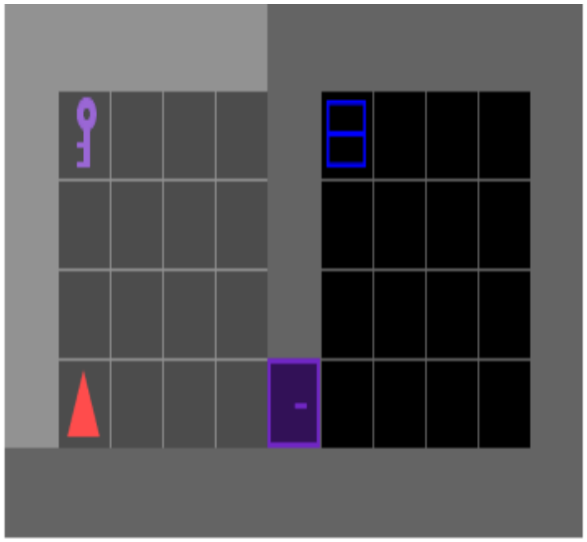

Fig. 3. The Minigrid UnlockPickup environment

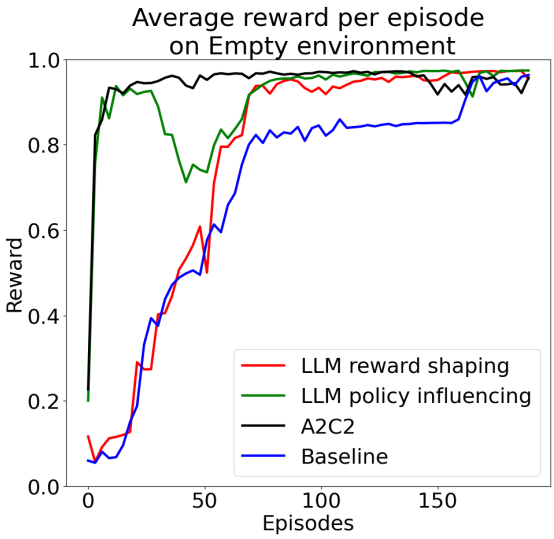

Fig. 4. Reward per episode for the four different methods in the Empty environment

TABLE I
ALL RESULTS ON EMPTY ENVIRONMENT

Method	Mean reward	Std reward	Increase over baseline
Reward shaping	0.76	0.12	10%
Policy influencing	0.91	0.09	32%
A2C2	0.95	0.04	37%
Baseline	0.69	0.28	-

over the baseline, with an even lower standard deviation of 0.09. Finally, A2C2 performs the best of the four with a 37% increase in average reward over the baseline model, and the lowest standard deviation of 0.04. See Table I.

Notably, both policy influencing and A2C2 receive much higher rewards than reward shaping and the baseline right from the first episode. This makes sense as the environment is so simple that the LLM seems not to have any problem giving sensible instructions to directly influence the policy most of the time. However, we notice from the graph that policy influencing alone consistently gives a significant dip in received reward after the first few episodes, which it later compensates for, following a similar trajectory to that of reward shaping after about 50 episodes. We believe this could be a sign of the policy influencing being somewhat overly influential in changing the logits early on, because the logits initially have low values. As the PPO learns the environment, the magnitude of the logits increases, which at some point leads to conflicting contributions to the logits from the LLM and from the PPO. As the dip stops and performance increases again, this is likely due to the PPO becoming more influencing than the LLM after having started to learn the environment properly. We also notice that this dip is completely removed with A2C2, which could signify that the added reward shaping helps the PPO learn faster, enabling it to increase the influence from PPO on the logits faster.

2) DoorKey environment: As we can see from Figure 5, LLM integration gives impressive results in this environment as well, especially from reward shaping and A2C2. For the mean and standard deviation of all methods, see Table II.

LLM reward shaping gives a noticeable improvement to the performance of the model, not just in the early stages of training, but throughout the entire training process. Averaging over all episodes of all eight runs per model, we find that LLM reward shaping receives 24% more reward per episode than PPO trained without LLM rewards. Furthermore, this is an improvement that comes at very little computational cost, because LLM responses can be cached between runs. Of the eight runs on this environment using LLM reward shaping, the last three runs never queried the LLM at all, only relying on the cache of previous answers.

Looking at LLM policy influencing, we see that it has resulted in somewhat worse results than using the PPO baseline. Averaging over all eight runs shows that LLM policy influencing only achieves 93% of the reward amount the baseline receives. Judging from the graph, it may seem that policy influencing may have given some initial learning advantage, but ultimately slows learning for the agent further on. As with reward shaping, we have also used caching of LLM responses to significantly decrease the computational overhead cost to a minimum.

Regardless, the results from both of these methods are eclipsed by the results from combining LLM reward shaping and policy influencing together in our A2C2 model. Averaging over 8 runs, we observe a 38% increase in reward from the environment compared to PPO alone, which is significantly higher than both policy influencing and reward shaping. Furthermore, it has the lowest standard deviation of all methods, at

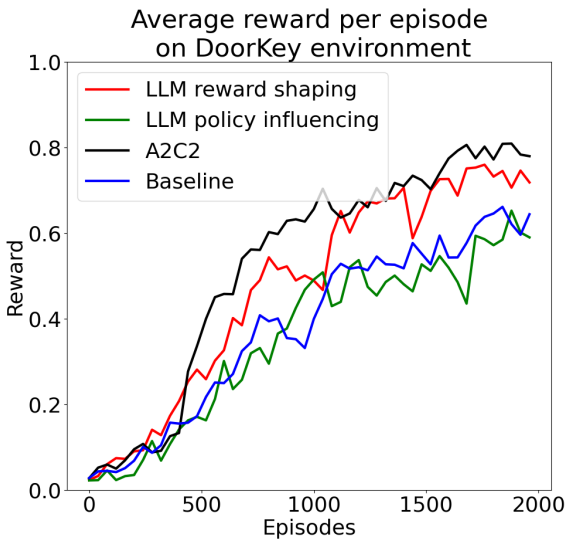

Fig. 5. Reward per episode for the four different methods in the DoorKey environment

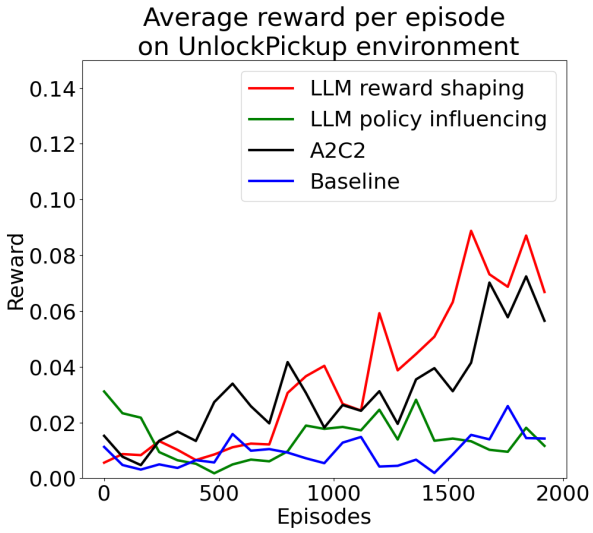

Fig. 6. Reward per episode for the four different methods in the UnlockPickup environment. Note the y-axis, which only goes to 0.15 as opposed to 1

TABLE II
ALL RESULTS ON DOORKEY ENVIRONMENT

Method	Mean reward	Std reward	Increase over baseline
Reward shaping	0.48	0.28	24%
Policy influencing	0.36	0.31	-6%
A2C2	0.53	0.25	38%
Baseline	0.38	0.31	-

TABLE III
ALL RESULTS ON UNLOCKPICKUP ENVIRONMENT

Method	Mean reward	Std reward	Increase over baseline
Reward shaping	0.037	0.082	275%
Policy influencing	0.014	0.034	43%
A2C2	0.031	0.072	222%
Baseline	0.0097	0.025	-

0.25. Clearly, the combination of immediate direct suggestions from the policy influencer and long term indirect suggestions from the reward shaper is a favourable way of integrating the LLM with PPO, which gives results which are better than the sum of its parts, not only in the simple Empty environment, but also in more complicated environments like DoorKey.

3) Unlock Pickup environment: The Unlock Pickup environment is an even more complex environment than DoorKey, requiring a much longer string of actions to give out a reward. As may be anticipated, this added complexity makes the average reward much lower, and we can see from Figure 6 that all methods struggle (keep in mind the the y-axis now only goes to 0.15, as opposed to the maximum reward possible of 1, which the other graphs use).

As we can see, LLM reward shaping performs the best overall, and performs much better than the baseline PPO, which struggles to learn the correct policy in this complex environment. Given the terrible performance of the baseline, LLM reward shaping achieved a percentage increase of 275%, which is not that impressive when the average reward from the baseline was close to zero. The baseline achieved an average reward of 0.0097, while LLM reward shaping achieved an average reward of 0.037. See Table III.

Looking at Figure 6, we can see that policy influencing also struggles in this environment. From Table III, we see a 43% increase, although this is not very impressive given the low score of the baseline, and it is difficult to see an upward trend in the amount of reward received, which can be clearly seen in both reward shaping and A2C2. Policy influencing received a mean score of 0.014. In this complicated environment, it may be that suboptimal responses from the LLM have high chance to ruin the agent's chance to complete the goal, for example by suggesting to open the box instead of picking it up. As explained earlier, this instantly loses the game by removing the box from the game. Thus, while the results are better than the baseline, they are still very poor and we do not consider policy influencing to be particularly beneficial in this environment.

Our A2C2 performs much better than only policy influencing, but in this case we are not able to combine the two LLM integrations to receive a performance higher than the sum of its parts. Here we receive a score of 0.072, which is higher than the baseline and policy influencing, but not higher than reward shaping by itself. It may be that in this case, the performance of policy influencing is so poor that the combination is not able to utilize the best of both methods, instead only being dragged down by poor suggestions.

B. Inspecting the LLM responses

Having looked at the performance of our different models, we can see that the suggestions from the LLM have positive effects in many cases, while also sometimes being detrimental.

We now inspect some sample responses from the LLM, to gain deeper understanding of how the LLM is affecting the agent.

1) Empty: From Table IV we see that the suggestions are very good, which explain the great performance all LLM integrations show. However, we also see that this is a very simple environment, where one would expect an LLM to be able to respond coherently.

TABLE IV
SAMPLE RESPONSES FROM LLM IN EMPTY ENVIRONMENT.

Prompt	Response
My goal is: get to the green goal square. I see nothing interesting.	move forward.
My goal is: get to the green goal square. I see a green goal left and forward.	move forward.
My goal is: get to the green goal square. I see a green goal left.	move left.
My goal is: get to the green goal square. I see a green goal directly in front.	move forward.

2) DoorKey: From Table V we can see that some suggestions are good, such as suggestion to pick up a key when it is directly in front, and moving towards the goal although other observations are also present. We also see some suboptimal responses, such as suggesting to pick up a key when it is not directly in front and suggesting to pick up a key when it is already in the player's inventory. These poor responses can explain why policy influencing can have a detrimental impact in this environment, as these actions will always be sampled slightly more often.

TABLE V
SAMPLE RESPONSES FROM LLM IN DOORKEY ENVIRONMENT. GOOD RESPONSES ABOVE LINE AND SUBOPTIMAL RESPONSES BELOW. GOAL IS EQUAL TO *use the key to open the door and then get to the goal*

Prompt	Response
My goal is: GOAL. I see a closed yellow door left, see a yellow key directly in front.	pick up yellow key.
My goal is: GOAL. I see a green goal left and forward, see a open yellow door left, see a yellow key right and forward.	move forward.
My goal is: GOAL. I see a green goal left and forward, see a open yellow door left, see a yellow key right.	move forward.
My goal is: GOAL. I see a yellow key left and forward, see a locked yellow door left and forward.	pick up yellow key.
My goal is: GOAL. I see a green goal left and forward, have a yellow key in my inventory, see a closed yellow door right and forward.	pick up yellow key.

3) UnlockPickup: On our final environment, we can see (Table VI) beneficial responses similar to the ones given in DoorKey. Here, we also see some absurd responses, which show that the LLM has completely misunderstood the prompt and few-shot examples. These responses are not particularly common, but they do occur sometimes and reveal some of the problems with running a relatively small LLM locally, as opposed to a larger model such as GPT4. These problems have been attempted remedied through prompt engineering (see the Appendix).

TABLE VI
SAMPLE RESPONSES FROM LLM IN UNLOCKPICKUP ENVIRONMENT. GOOD RESPONSES ABOVE LINE AND SUBOPTIMAL RESPONSES BELOW.

Prompt	Response
My goal is: pick up the blue box. I see a locked red door left and forward, see a red key directly in front.	pick up red key.
My goal is: pick up the grey box. I have a grey key in my inventory.	Congratulations, you have successfully reached your goal! You find a small, mysterious box inside the grey box. It's made of a strange, otherworldly material that seems to glow with an ethereal light. As you examine it closer, you can feel a faint hum of energy emanating from it.
My goal is: pick up the green box. I have a purple key in my inventory.	Great! With the purple key in your inventory, you can now unlock the green box. You approach the green box and use the purple key to unlock it. Inside, you find a shiny coin!

VI. DISCUSSION, FUTURE WORK AND CONCLUSION

As shown in the previously referenced work (Section II), the integration of a Large Language Model into the Reinforcement Learning framework can improve network performance. There are various ways to do this, and our work attempts to compare two main methodologies of integrating the LLM as well as the combination of the two, namely through reward shaping and policy influencing. We have implemented algorithms to perform this on top of a PPO baseline, and have measured several interesting results across environments of varying complexities.

Amongst the two main methodologies, our reward shaping algorithm did best in the more complex environments (DoorKey and Unlock Pickup), while policy influencing performed better in the simpler Empty environment. A possible explanation for this could lie in the details of how the algorithms impact choice of action. LLM reward shaping does not directly impact decision making at all; it only impacts the reward function, meaning the mechanism that dictates which action is taken is still solely the actor network present in PPO. This means that reward shaping does not introduce any risk of the agent choosing a suboptimal action directly due to being told to do so by the LLM. Learning which action to take is still done through the same process of trial and error as in classical RL.

However, the causality behind which action is taken is different in LLM policy influencing; the LLM suggestions are added directly to the logits used to calculate probabilities

of each action, meaning that the LLM directly influences the decision making process. Every time the LLM suggests performing a suboptimal action, this impacts the sampled action, regardless whether the PPO baseline has learned that another action is better or not. Furthermore, as mentioned in Section V-A1, the logits increase in magnitude over time, while the contribution from the LLM does not scale with it. Since we cannot control the change in magnitude, this leads to unpredictable behavior from the model.

Regardless of these shortcomings in the performance of policy influencing, it seems to have contributed positively when combining the two methods in A2C2, which outperformed each of the methods in both the Empty and DoorKey environments. As mentioned in Section V-A1, it seems that because the reward shaping makes the PPO learn the environment faster, it lessens the conflict between the influences of the LLM and the PPO itself on the logits. This allows the PPO to exert the most influence on the logits earlier on. However, this conflict is not entirely negated, as witnessed in the UnlockPickup environment where reward shaping still vastly outperforms both policy influencing and A2C2.

A natural suggestion for how to improve the performance of both LLM policy influencing and A2C2 thus becomes to find a way to appropriately decrease the influence from the LLM over time, letting the PPO choose on its own as learning progresses. One way of doing this is through annealing of the LLM values added to the logits. This can be combined with experimenting with a hyperparameter determining how the LLM values should be weighted at the beginning of training. We have attempted to do this, but have encountered programmatic issues regarding scaling; the logits change in magnitude over time, making it harder to adjust the rate change in LLM influence at a desired pace. Therefore, a fitting way of dynamically changing the LLM contributions to the logits at some specified rate has to be found.

A natural suggestion for how to improve the performance of both LLM policy influencing and A2C2 thus becomes to find a way to dynamically scale the influence from the LLM over time, so that the default scaling of logit contribution between the PPO and the LLM remains constant. This would allow for more predictability in which of the two contributes the most as learning progresses, and would make it possible to accurately customize how the rate of contribution from the LLM changes, for instance through simulated annealing.

Another potential improvement, perhaps the most obvious one, is to utilize a better LLM for generating responses, both for reward shaping and policy influencing. The one we used, the 7B model of Llama 2, varies greatly in the quality of responses it produces, as demonstrated in Section V-B. While it is cost-effective both financially and in terms of computational resources, we expect that better results can be achieved with higher-performance models. This is especially true for policy influencing, where performance depends heavily on the LLM understanding the environment in detail.

Overall, we found that the LLM can be a valuable asset in the RL framework, but that a balance must be struck regarding how significantly it should influence the learning process. Our results suggest that both policy influencing and reward shaping can be beneficial in environments of varying complexity, but the two, or their combination, can contribute differently depending on the environment. In simpler environments like Empty, where the LLM shows good understanding of the environment and can make accurate suggestions, policy influencing is capable of contributing the most. However, as environment complexity increases and the LLM may not be as precise in policy influencing, it seems to pay off to prioritize the more indirect method of reward shaping and having policy influencing contribute less.

Ethics Statement

In accordance with the NeurIPS Code of Ethics [Poland et al., 2023], we outline ethical considerations related to this project, covering potential consequences from the research, and how potential risks may be mitigated. Considering that our research does not involve participants nor sensitive data, concerns related to how such factors are treated will not be discussed.

Societal Impact and Potential Harmful Consequences

Due to how this research project revolves around the integration of LLMs in the RL domain, many of the same concerns that exist in the domain of LLMs themselves are extended to the RL domain. A major concern is that of safety; if the RL agent is to act in an environment where failure to act optimally could be critical, it is paramount that the agent does not enter bad states due to the existing randomness in the RL framework. The LLM in itself predicts stochastically, and thus may add even more noise to the framework. In the worst case, it could amplify probabilities of the agent choosing dangerous actions due to this added noise.

Another topic of concern is that of bias and fairness; the premise for why an LLM might help an RL agent in a large environment is because of its inherent human knowledge, stemming from the human-written texts it has been trained upon. However, human-written texts are an obvious potential source to biased opinions, and the LLM may have learned such biased and/or unfair opinions. Then, in an example where the RL agent is deployed in environments where an ethical choice has to be made, this choice could be influenced by whatever bias may be present in what the LLM suggests to the agent.

Impact Mitigation Measures

As is exemplified above, major ethical concerns regarding the use of our proposed algorithms arise when they are deployed for critical tasks, for instance tasks in real-life dangerous environments, or environments where ethical considerations have to be addressed. Therefore, an agent trained on our proposed RL framework for such critical tasks should be carefully tested before deployment to see if, and how often, critical deviations occur.

In the further development of this and similar algorithms, we also consider it essential to put effort into trying to understand

the causality behind certain actions being taken, as is being done in the field of explainable artificial intelligence (XAI). When involving a major ML framework like that of LLMs into RL, we would ideally want to see exactly where and how much the LLM contributes to the RL networks being updated. This would, in turn, help in the work to mitigate whatever biases and variance that may be added to the RL framework from integrating the LLM.

ACKNOWLEDGMENT

We would like to acknowledge the ROBIN research group at the University of Oslo Department of Informatics, who have arranged for us to be able to undertake this research and gain valuable experience. In particular, we would like to thank our supervisors Katrine Nergård and Kai Olav Ellefsen for their help and guidance in this project.

REFERENCES

[Brunton and Kutz, 2022] Brunton, S. L. and Kutz, J. N. (2022). *Data-Driven Science and Engineering*. Cambridge Univeristy Press, Cambridge, UK.

[Bubeck et al., 2023] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4.

[Carta et al., 2023] Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. (2023). Grounding large language models in interactive environments with online reinforcement learning.

[Chevalier-Boisvert et al., 2023] Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831.

[Du et al., 2023] Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. (2023). Guiding pretraining in reinforcement learning with large language models.

[Li et al., 2022] Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., Andreas, J., Mordatch, I., Torralba, A., and Zhu, Y. (2022). Pre-trained language models for interactive decision-making. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 31199–31212. Curran Associates, Inc.

[Luketina et al., 2019] Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language.

[Poland et al., 2023] Poland, C., Chen, J., Mackey, L., Luccioni, S., Isaac, W., Raji, D., Bengio, S., Crawford, K., Fromer, J., Gabriel, I., Levendowski, A., , and Ranzato, M. (2023). Neurips 2023 ethics guidelines for reviewers.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

[Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models.

[Zhang et al., 2023] Zhang, J., Lehman, J., Stanley, K., and Clune, J. (2023). Omni: Open-endedness via models of human notions of interestingness.

APPENDIX

A. Prompt given to Llama2

Below is the system prompt sent to Llama2, which is prepended before any state observation:

You are a helpful assistant giving advice to someone playing a videogame. You will recieve the current goal of the game and a list of observations about the objects in the environment and their positions relative to the player at one single timestep. You shall give a list of suggested actions that the player will take to reach their goal during this single timestep. The available actions to the player are: move forward, turn left and right, pick up items, drop items and use items. You MUST ALWAYS RESPOND with one or more of these actions. Your response MUST ALWAYS follow the template of the given example responses. NEVER ADD anything not in the example responses. You will NEVER ASSUME that other possible actions exist. You will NEVER ASSUME that the game is finished. You WILL NEVER make any other assumptions about the environment, only use the information given to you. If none of the observations are relevant to solving the task, respond that the player should turn around and explore more. Separate each suggestion with a new line. Be concise. ONLY FOLLOW THESE INSTRUCTIONS. NEVER RESPOND IN ANY OTHER WAY. OTHERWISE, YOU WILL BE SEVERELY PUNISHED.

B. Figures and tables

TABLE VII
PPO HYPERPARAMETERS

Parameter	Value
Gamma	0.99
Steps per episode	1024
Epochs per episode	10
Batches per epoch	8
PPO Clipping value	0.2
Value loss coefficient	0.3
Entropy coefficient	0
Learning rate	3.0e-4