Using Natural Language for Reward Shaping in Reinforcement Learning - Reproduction and Possible Enhancements

Alexandre Amiredjibi

***Abstract*— This research aims to replicate and improve upon the results of 2019 the paper "Using Natural Language for Reward Shaping in Reinforcement Learning" by Prasoon Goyal, Scott Niekum, and Raymond J. Mooney by mirroring their methodology and suggesting enhancements.**

The research of Goyal *et al.* implements a Reinforcement Learning (RL) model whose training process is significantly augmented by the use of natural language commands. They introduce the LanguagE-Action Reward Network (LEARN), a framework that maps free-form natural language instructions to intermediate rewards based on actions taken by the agent.

This paper introduces TransLEARNer: a transformer-based model that shares the same functionality as LEARN, while simultaneously performing worse.

Results of evaluating TransLEARNer show that, while not introducing meaningful gains in some tasks, the model is able to significantly improve performance on some especially sparse-reward tasks that are difficult or impossible to complete without intermediate rewards. Still, these gains come with serious drawbacks, and effectively flatten out after some number of time steps as the agent begins to prefer language rewards to the external rewards. All project code is available in the GitHub repository linked in the footnote below. [1]

I. Introduction

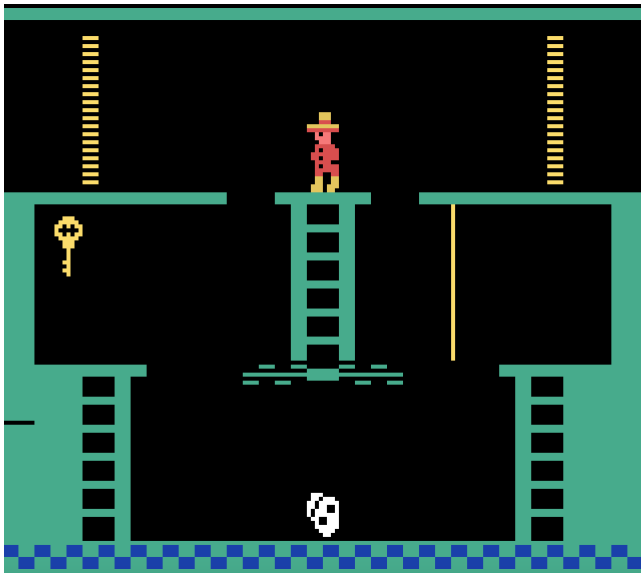

Fig. 1. Starting position in the Montezuma's Revenge environment

The training process of a Reinforcement Learning (RL) model often requires a large amount of interaction with the environment before arriving at tangible results. To reduce interaction time, researchers have resorted to designing reward functions that offer additional feedback to the model based on intermediate states in the training environment (referred to as "reward shaping"), separate from the external feedback offered by completing the tasks inherent to the environment. However, reward shaping (even referred to as "RewArt", i.e "reward art") is a complex tool limited to experts.

The novel technique of Goyal et al. uses natural language instructions (*e.g.* "jump over the skull while going to the left") to shape intermediate rewards. This significantly streamlines the task of designing intermediate rewards for RL training, and effectively extends the capability to non-experts of reward shaping. Their experiments demonstrate an improvement in task completion of 60% for the same number of interactions compared to learning without language. These results, paired with the substantially improved accessibility of reward shaping offered by natural language instruction, show promise in streamlining RL research capabilities.

To achieve effective instruction using natural language, Goyal et al. present their *LanguagE-Action Reward Network (LEARN)* : a neural network that takes paired (trajectory, language) data from the environment and predicts if the language describes the actions within the trajectory. In their work, trajectories are passed to LEARN as "action-frequency" vectors, which describe the relative frequency of each available action in the given trajectory; as a result, the process loses temporal information about the sequence of actions.

This paper presents *TransLEARNer*, a transformer-based [Vaswani et al. 2017] classification model that combines action-frequency vectors with shortened, simple verbal descriptions of the given trajectory, and predicts whether the said trajectory relates to the specified natural language instruction.

Reinforcement Learning. Reinforcement learning is a type of machine learning that involves an agent learning through trial and error in an environment in order to maximize a reward. It can be formalized using the Markov Decision Process (MDP), which is a mathematical framework that consists of a set of states, actions, and a reward function. The agent follows a policy, which is a set of rules that determine what action to take in a given state. The goal of the agent is to learn the optimal policy, which is the policy that maximizes the expected reward over time.

[1]All project code is available at `https://github.com/alexamirejibi/rlnlp`

II. Related work

There have been numerous recent approaches to using NLP to specify rewards in RL settings. The approach of Goyal et al. and this research for using natural language processing (NLP) to aid reinforcement learning (RL) differs from previous methods in that it maps language to a reward function directly from annotated data, potentially reducing human effort, but requires pairs of trajectories and instructions for training. This is in contrast to methods by Williams et al. (2017) and Arumugam et al. (2017), which map natural language to a reward function using predefined objects and simple features, which may not scale to more complex environments. Other approaches, such as those by Misra et al. (2017) and Kuhlmann et al. (2004), use natural language for reward shaping or to affect action probabilities, respectively. Branavan et al. (2012b) and Bahdanau et al. (2018) incorporate linguistic features into the action-value function or use them to improve a policy through adversarial learning. The authors' approach is related to the method by Wang et al. (2018), which also uses intermediate language-based rewards in RL, but focuses on using RL to improve natural language instruction-following while the authors focus on using instructions to improve RL performance.

Kaplan *et al.* have proposed a similar technique, also involving natural language commands for shaping intermediate rewards. Their approach starts out with mapping visual frames of the environment and their natural language descriptions (e.g. "The ball is to the right of the paddle") into a single embedding space, which the agent learns in the initial phase of training. In this phase, the agent is evaluated on its ability to satisfy the natural language description of a desired state.

There are key differences between this approach and the implementation of Goyal *et al.* While Kaplan *et al.* first learns an embedding between the environment and its natural language descriptions, and then evaluates the agent on its ability to achieve the desired state, the latter assigns (a) appropriate and (b) random language descriptions to a set of trajectories (actions), and uses those as positive examples and negative examples respectively. Alignment with language descriptions is then implemented in the reward function as an extension of the Markov Decision Process (MDP). The authors refer to the new framework as MDP+L, where L represents the language command. Because the only change to the original model is in the reward function, the authors point out that their language-aided RL implementation is agnostic to the particular choice of RL algorithm, which, in my opinion, is a notable strength of the approach.

Other notable work in the area of natural language assisted RL comes from Mu *et al*, who propose another approach for leveraging intrinsic rewards in sparse reward environments. L-AMIGo, which extends AMIGo (Campero et al., 2021), involves separate adversarially motivated student and teacher agents, where the student is rewarded for completing the goals provided by the teacher, and the teacher is rewarded for proposing goals that are increasingly difficult but achievable. Goals here are presented in natural language, from a running set of goals that is "updated as the student encounters new state descriptions" (p3). Importantly, while the previous two papers evaluate their results compared to RL approaches not aided by intrinsic exploration (which all of the aforementioned papers here employ), Mu et al. evaluate their results directly against state-of-the-art non-linguistic exploration baselines (AMIGo, Campero et al., 2021) and NovelD (Zhang et al., 2021), and find that these are significantly outperformed by their linguistic approach, further strengthening the case for language-assisted RL.

III. Method: Goyal et al.

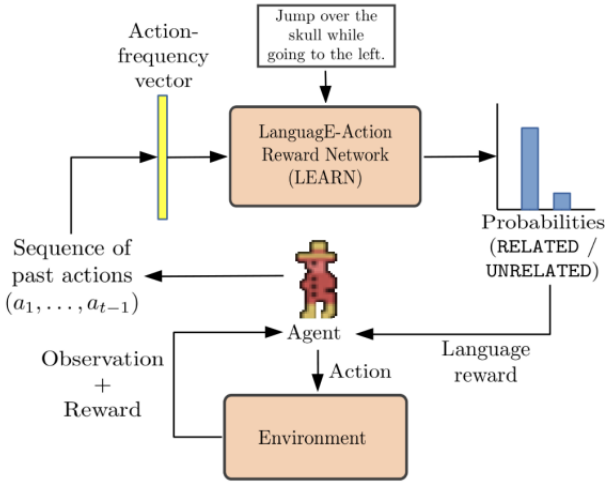

Figure 2: Our framework consists of the standard RL module containing the agent-environment loop, augmented with a LanguagE-Action Reward Network (LEARN) module.

Fig. 2. The above figure and description is a screenshot from Goyal et al.

As mentioned earlier, the paper uses an extension of the MDP framework that they refer to as MDP+L. This highlights the fact that the only modifications to the model itself that enable language assistance are in the reward function (MDP). The MDP framework is used for the extrinsic rewards offered by the environment (completing a task), while the MDP+L is used for intrinsic rewards, i.e the intermediate rewards specified with natural language. In order to find the optimal policy for MDP+L, the authors employ a two-phase approach. First, they train a neural network (LEARN) that takes paired (trajectory, language) data from the environment and predicts if the language describes the actions within the trajectory. In practice, this involves (a) sampling two time steps i and j to define time sequence from between the two time steps, and creating "action-frequency vectors", which are vectors of a size equal to the number of actions in the MDP+L, where the k component is the fraction of timesteps action k appears in the time sequence. This allows the authors to create a dataset of (*action-frequency vector, language command*) pairs from a given set of (*time sequence, language command*) pairs - in other words, pairing

actions with language commands. Actions are paired with relevant language commands for positive examples, and with random language commands for negative examples. This allows the LEARN network to predict whether or not a given action-frequency vector is related to a given natural languag description.

In basic terms, the LEARN network predicts whether the actions being performed by the agent at any given time step are relevant to the language description. In training, the agent is rewarded in proportion to the related-ness of its action to the language instruction, as determined by the *softmax* (related/unrelated) layer of the neural network.

IV. Method: This research

A. *Transformer-based Language-Action Reward Network (TransLEARNer)*

TransLEARNer is an adaptation of Goyal et al.'s LEARN model, using Transformer architecture [Vaswani et al. 2017]. Like LEARN, TransLEARNer predicts the probability that the sequence of actions being executed by the agent is relevant to the assigned natural-language instruction. TransLEARNer employs a multi-modal transformer model, adapted from DistilBERT [Sanh et al. 2020], to predict the relevance of the agent's actions to the provided natural language instruction. In addition to the action-frequency vectors introduced by Goyal et al., TransLEARNer takes in a shortened sequence of the agent's actions, preserving meaningful temporal information. The sequence of actions is translated to natural language using the action-meanings provided by the Atari environments, manually adapted to the actions present in Montezuma's Revenge; this allows the model to make use the rich semantic representations already available to it in the base DistilBERT model, making fine-tuning more straightforward.

In order to implement numerical data in the form of action-frequency vectors, I use the Multimodal Transformer library [Gu and Budhkar 2021]. Here, the natural language instruction is passed to DistilBERT together with the transcribed action sequence as a pair of sentences, separated by a [SEP] token and preceded by a [CLS] token, as is common practice for sequence-pair classification tasks. The embedding generated by DistilBERT is combined with the action-frequency vector, which is a vector of size equal to the number of available actions in the Montezuma's Revenge environment and contains the relative frequencies of each action, calculated by taking the absolute frequency of the action in a trajectory and dividing it by the length of the trajectory. Finally, the combined embedding passes through a series of fully connected layers followed by a softmax function, which gives us the probabilities of the language-trajectory pair belonging to each of the 5 labels defined in the dataset (explained in the Data section). In order to arrive at a final classification, we simply take the *argmax* of the probabilities.

B. *The combine module*

. To combine DistilBERT embeddings with action-frequency vectors, I use the gated summation combine method provided by the Multimodal Toolkit library, based on research by Rahman et al.

In order to use the TransLEARNer network while training in the Gym environment, we simply create a Gym wrapper environment: an object that can take in any Gym environment and adopt it to the user's needs. In this environment, we are able to intercept the actions and the rewards passing between the agent and the environment, store the actions as trajectories, calculate language rewards with TransLEARner, and return the rewards alongside the external rewards passed to us by the environment. This makes the implementation highly adaptable, requiring no tinkering with the RL algorithm used. We can simply pass the wrapped environment to the algorithm and train it as usual.

Following in the footsteps of Goyal et al., I chose the use a Proximal Policy Optimization (PPO) algorithm. For each task, the algorithm was trained for 500,000 timesteps, with and without language rewards.

V. Data

All raw data used for training the TransLEARNer network was made publicly available by Goyal et al., and can be found in their GitHub repository. [2]

A. *Data collection*

Goyal et al. collected data for their LEARN model by generating trajectories in an environment and obtaining natural language annotations for these trajectories from human annotators. They used 20 trajectories from the Atari Grand Challenge dataset, which contains hundreds of crowd-sourced trajectories of human gameplays on 5 Atari games, including Montezuma's Revenge. These trajectories contain a total of about 183,000 frames, from which Goyal et al. extracted 2,708 equally-spaced clips that were three seconds long. To obtain language descriptions for these clips, they used Amazon Mechanical Turk, where workers were shown clips from the game and asked to provide corresponding language instructions. Each annotator was asked to provide descriptions for 6 distinct clips, while each clip was annotated by 3 people. After filtering out bad annotations, Goyal et al. obtained a total of 6,870 language descriptions. They note that the resulting dataset may be noisy, as their filtering process did not explicitly check if the language instructions were related to the corresponding clips and did not correct for spelling or grammatical errors.

B. *Data augmentation*

The following augmentations were performed by the author of this paper, rather than Goyal et al.

[2] https://github.com/prasoongoyal/rl-learn

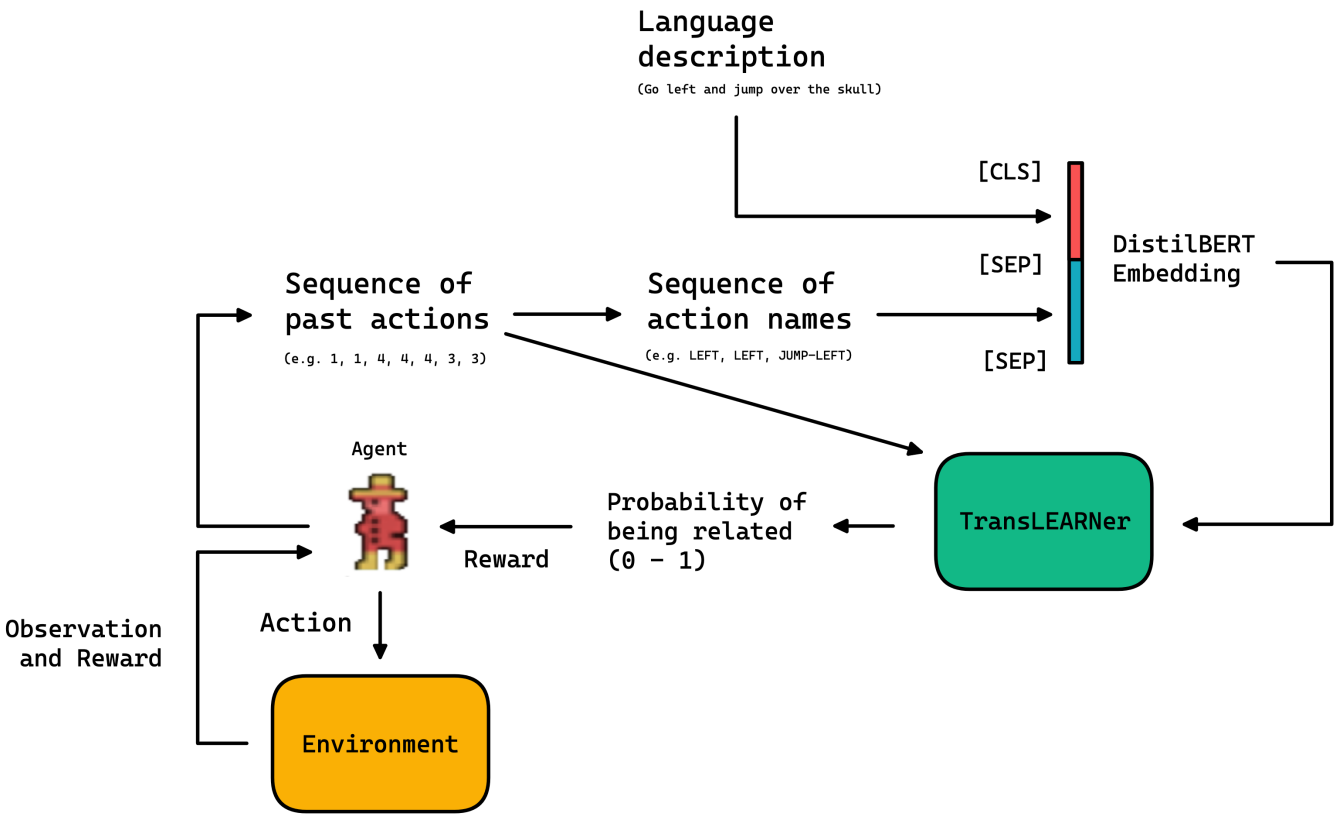

Fig. 3. The training loop.

1) Omitting non-action: All trajectories containing only non-action, i.e the action marked by ”0”, were removed from the dataset. This left our dataset with 4,480 pairs of trajectories and descriptions. Furthermore, all instances of non-action were removed from each trajectory, leaving us with no examples with the agent simply standing in place. The reason for this filtering was the overwhelming dominance of the ”0” action in the dataset, which may have provided a significant incentive for the agent to remain still. Thus, the TransLEARNer network is unable to helpfully process instructions for ”staying still” and otherwise waiting.

2) Diversifying examples: In order to create negative examples, Goyal et al. paired language descriptions with random trajectories from the dataset. Experience with training TransLEARNer showed that such examples were not sufficient to account for the chaotic, random motions of an RL agent, which resulted in inaccurate reward predictions. To mitigate this, additional data labels were added to describe data pairs with various levels of noise, taking the binary classification task to a multi-label classification with 5 labels: negative data, 75 percent noise, 50 percent noise, 25 percent noise, and gold standard data from the original dataset. Noise was added to the corresponding data by arbitrarily replacing actions with random actions, better imitating the actions of a newly-initialized RL agent.

Label	Augmentation	Number of examples
0	Negative data	24228
1	75% noise	6057
2	50% noise	6057
3	25% noise	6057
4	Original data	6057

TABLE I
AUGMENTED DATA DISTRIBUTION

3) Hindsight comments on data: Hindsight has revealed a fatal downfall of these augmentations: the invariability of noise for each label may have served as a significant bottleneck for generating correct predictions in the environment setting. I speculate that the rewards in the Montezuma’s Revenge environment were limited to trajectories that vaguely resembled the noise levels specified above. When using TransLEARner to generate rewards, a trajectories was only assessed as belonging to the positive labels if they were a correct trajectory with roughly 25, 50, or 75 percent random actions. This might have significantly reduced the amount of language rewards generated.

4) Negative data: In order to aid the model in learning the most important features of our language-trajectory pairs, five methods were employed to create diverse negative data:

- Language description paired with random trajectory from data.
- Language description paired with fully random trajec-

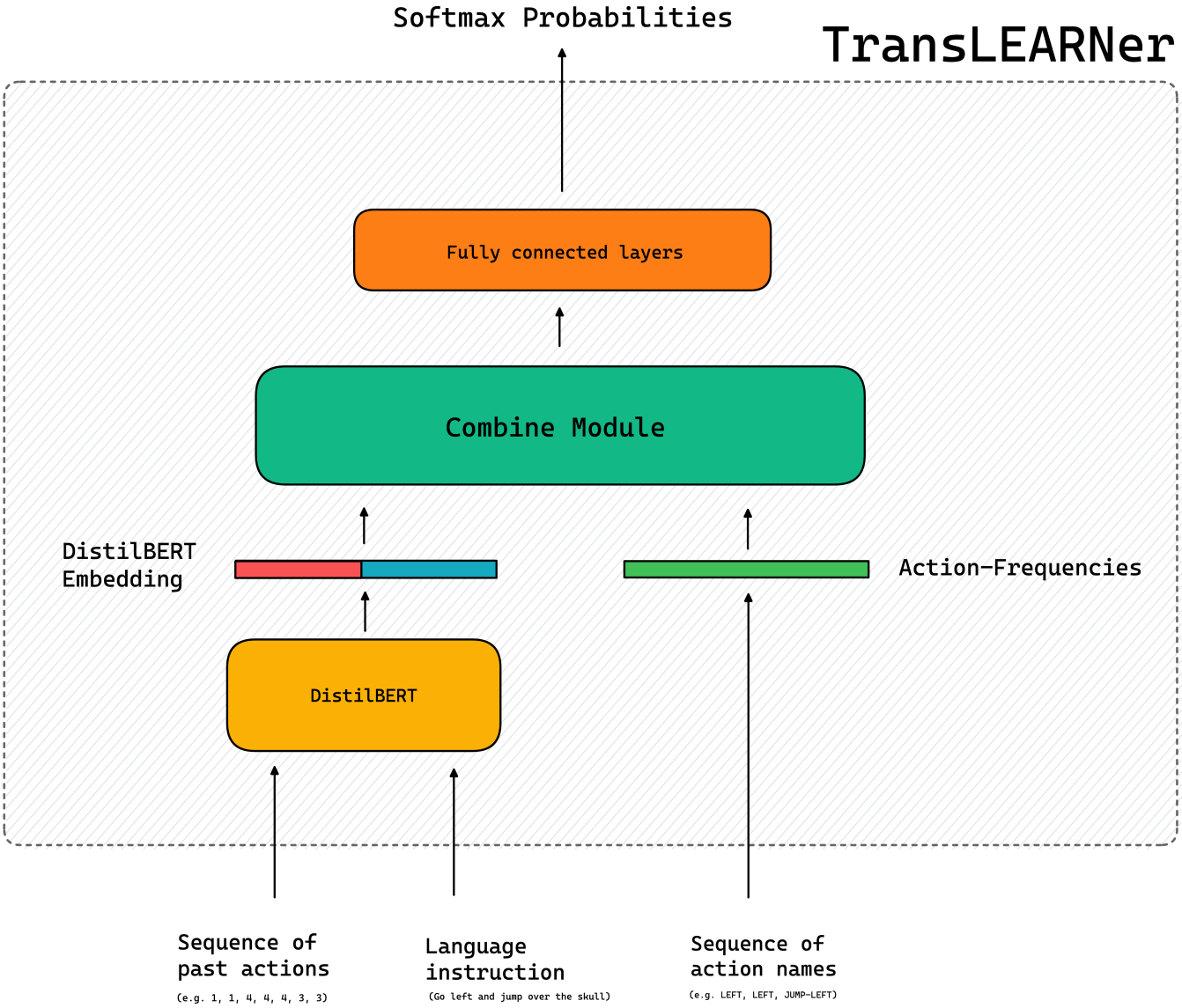

Fig. 4. The TransLEARNer model.

tory.

- Language description paired with random trajectory from data with 75% noise.
- Language description paired with random trajectory from data with 50% noise.
- Language description paired with random trajectory from data with 25% noise.

VI. Evaluation

A. Overview

In order to evaluate the effectiveness of natural language-based rewards in the Montezuma's Revenge environment, I evaluate both policies on a series of tasks over 500,000 time steps. For each task, I count successful episodes over n time steps, where task success is defined manually for each task. Figure 4 displays one example of a task. This evaluation method is identical to Goyal et al.

Each policy was tested on each experiment at least twice, and the best results were selected. It's worth noting that the results didn't differ much from each other.

VII. Limitations of the Evaluation

The number of experiments conducted is generally insufficient to conclude that the language-based policy outperforms the non-language policy. While the repetition of each task establishes a slightly higher confidence, the number of tasks themselves are rather low. This stems from the fact that these tasks were conducted on numerous iterations of the implementation, and for each iteration, training consumed a significant amount of time.

Moreover, the evaluation here is absent of meaningful statistical analyses, which would help us establish whether the results are at all meaningful.

Regardless, the slight bump in performance and the significant proportional increase displayed in task 3 (fig 6) hints

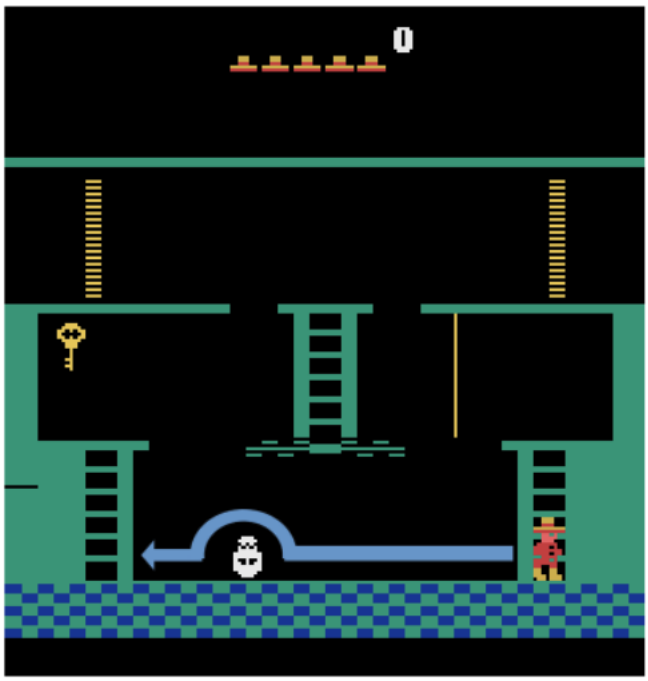

Fig. 5. Task 3: In order to complete the task, the agent must run left, jump over the skull, and reach the ladder.

that the approach is promising.

VIII. Results

TABLE II
RESULTS

Task completion over 500,000 episodes		
Task	Ext. Only	Lang+Ext
2	13347	13427
3	0	14
4	36906	39053

The results show that for some tasks, the TransLEARNer model is able to increase the learning performance of the RL agent. Figures 5, 6, and 7 display the performance of both policies (with language rewards and without language rewards).

Task 3 (fig. 6) is both a positive example of the language policy outperforming the non-language policy, as well as an unfortunate failure case. Here, we see that the language policy is sometimes able to complete the task, but only a handful of times (specifically, 14 times), and completely stops being able to complete the task around the 200,000th time step mark. There is a simple explanation for this: the model learns to optimize for the language reward rather than the environmental reward (finishing the task), which helps it finish the task a few times, but eventually causes the model to ”forget” the rewards of the task as it is flooded with the rewards from the TransLEARNer model. This serves to illustrate an important point: in order for the language rewards - or any sort of intermediate rewards - to be effective, they must be balanced carefully with the rewards of the actual task. In this case, each language reward was multiplied by 0.1 (the language reward coefficient can be specified in the argument parser), which put the average reward at approximately 0.03 (the task reward being 1). However, as this task is rather long and difficult, the agent was able to generate an overwhelmingly larger amount of language rewards than task-based rewards, and proceeded to optimize accordingly.

Still, the language-based policy being able to complete task 3 while its non-language counterpart being wholly unable to complete the task is promising evidence that language-shaped rewards can augment training in sparse-reward environments. Of all the environments, task 3 has, indeed, the sparsest rewards.

Task 2 doesn't show a meaningful improvement from using language rewards, but it is worth noting that the language policy approached its peak completion rate sooner. Task 4 shows minor improvement.

Goyal et al.'s model achieves much larger gains in performance - 60% more completions per task on average. Therefore, it's reasonable to assume that the mediocre results presented in this paper are a result of the researcher's inferior know-how in the field of Machine Learning.

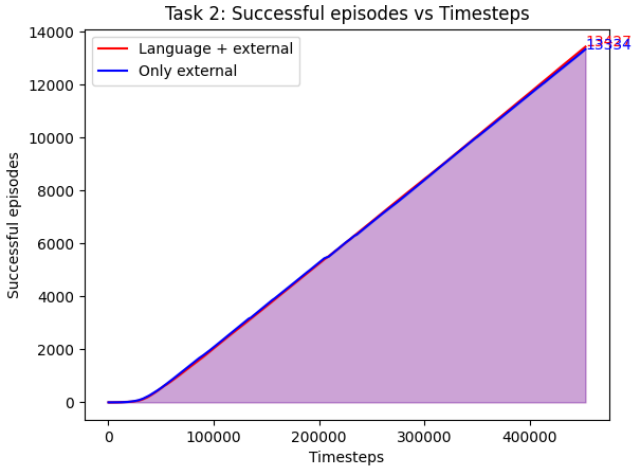

Fig. 6. Task 2: In order to complete the task, the agent must run left, jump over the skull, and reach the ladder.

IX. Conclusion

In this paper, I present TransLEARNer - a model for specifying intermediate rewards in sparse-reward RL environments. With performance on task 3, we see that TransLEARNer is able to nudge the agent to complete a task is difficult to complete in the environment without language rewards. However, the results achieved by TransLEARNer are largely underwhelming, and it's safe to say that it does not live up to its dare I say epic name.

There are several important improvements to be made. First, using shortened natural language commands to capture temporal relations of the trajectory is a questionable approach. The rationale for this design choice, as mentioned

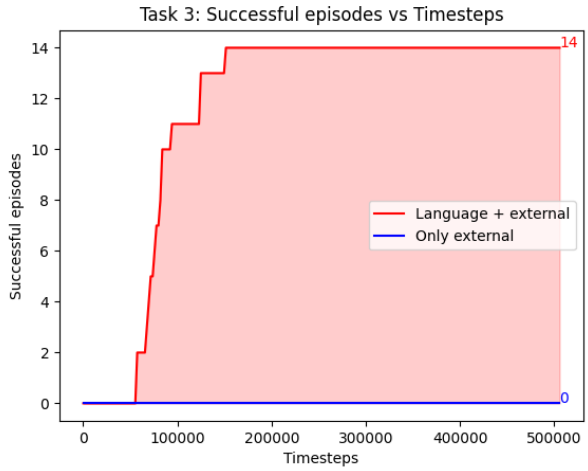

Fig. 7. Task 3: In order to complete the task, the agent must run left, jump over the skull, climb up the ladder, and get the key. It is the most difficult of all tasks.

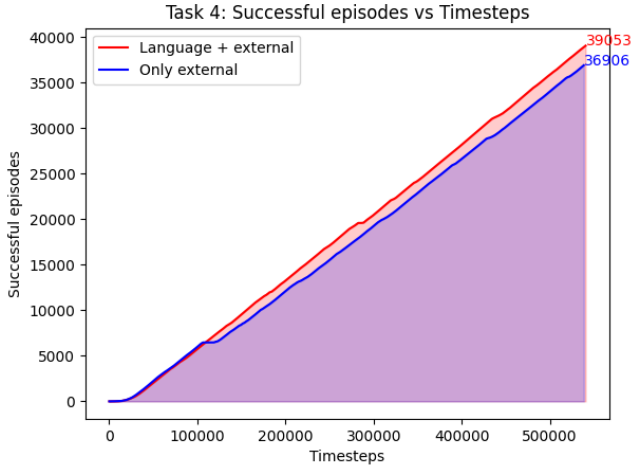

Fig. 8. Task 4: In order to complete the task, the agent must climb down the ladder on the left side of the room, go to the right side of the room, and climb up the ladder. There is no skull in this task - it is dead.

earlier, was to take advantage of the semantic features available to us through transformer language models. With the privilege of a fresh start, I would avoid this approach altogether, and simply design a non-linguistic way of capturing temporal information. Second, the data augmentation failed to capture the range of data generated by a newly initialized RL agent; I suspect that with more arbitrary amounts of noise (rather pre-specifed amounts for each label), TransLEARNer would be more effective at classifying trajectories out on the field. Third, the language rewards tend to overwhelm the rewards presented by the environment, and therefore the agent begins to prefer these rewards. Adjusting reward coefficients did not yield improvements, and did not mitigate the agent's vanishing ability to complete the task.

X. Discussion

We would not need to confront this last point if the language commands did, indeed, lead the agent to finishing the task every time. Unfortunately, it seems like the path of optimizing for language rewards consistently diverges from the optimal path, despite diverse sets of instructions (three similar instructions are provided to the agent, and the label predictions are averaged before generating the rewards). It should be noted, however, that the purpose of language rewards should not be to lead the agent all the way to completion; as intermediate rewards, they serve to reward the agent for being vaguely on the way towards completing the task. It remains unclear to the author of this paper how to mitigate the risk of our RL agent succumbing to TransLEARNer reward addiction.

It seems likely that there are certain "loophole" trajectories that the agent learns to abuse to generate language rewards without arriving at task completion. As language rewards are specified at the start of training, it is impossible, in this implementation, to foresee or mitigate these loopholes during training.

XI. Future work

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] Goyal, P., Niekum, S., Mooney, R.J. (2019). Using Natural Language for Reward Shaping in Reinforcement Learning. IJCAI.

[2] Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N.D., Rocktaschel, T., Grefenstette, E. (2022). Improving Intrinsic Exploration with Language Abstractions. ArXiv, abs/2202.08938.

[3] Kaplan, R., Sauer, C., Sosa, A. (2017). Beating Atari with Natural Language Guided Reinforcement Learning. ArXiv, abs/1704.05539.

[4] Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J.B., Rocktäschel, T., Grefenstette, E. (2021). Learning with AMIGo: Adversarially Motivated Intrinsic Goals. ArXiv, abs/2006.12122.

[5] Zhang, Tianjun et al. "NovelD: A Simple yet Effective Exploration Criterion." NeurIPS (2021).

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[7] Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[8] Arumugam, D., Karamcheti, S., Gopalan, N., Wong, L. L., Tellex, S. (2017). Accurately and efficiently interpreting human-robot instructions of varying granularities. arXiv preprint arXiv:1704.06616.

[9] Bahdanau, D., Hill, F., Leike, J., Hughes, E., Kohli, P., Grefenstette, E. (2018). Learning to follow language instructions with adversarial reward induction. arXiv preprint arXiv:1806.01946, 6-9.

[10] Branavan, S. R. K., Kushman, N., Lei, T., Barzilay, R. (2012a). Learning high-level planning from text. The Association for Computational Linguistics.

[11] Branavan, S. R. K., Silver, D., Barzilay, R. (2012b). Learning to win by reading manuals in a monte-carlo framework. Journal of Artificial Intelligence Research, 43, 661-704.

[12] Kaplan, R., Sauer, C., Sosa, A. (2017). Beating atari with natural language guided reinforcement learning. arXiv preprint arXiv:1704.05539.

[13] Kuhlmann, G., Stone, P., Mooney, R., Shavlik, J. (2004, July). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In The AAAI-2004 workshop on supervisory control of learning and adaptive systems.

[14] Misra, D., Langford, J., Artzi, Y. (2017). Mapping instructions and visual observations to actions with reinforcement learning. arXiv preprint arXiv:1704.08795.

[15] Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y. F., ... Zhang, L. (2019). Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6629-6638).

[16] Williams, E. C., Gopalan, N., Rhee, M., Tellex, S. (2017). Learning to parse natural language to grounded reward functions with weak supervision. In AAAI Fall Symposium on Natural Communication for Human-Robot Collaboration, 2017.