Safe Reinforcement Learning with Free-form Natural Language Constraints and Pre-Trained Language Models

Xingzhou Lou
School of Artificial Intelligence, UCAS
Institute of Automation, CAS
Beijing, China
louxingzhou2020@ia.ac.cn

Junge Zhang*
School of Artificial Intelligence, UCAS
Institute of Automation, CAS
Beijing, China
jgzhang@nlpr.ia.ac.cn

Ziyan Wang
King's College London
London, the United Kingdom
ziyan.wang@kcl.ac.uk

Kaiqi Huang
School of Artificial Intelligence, UCAS
Institute of Automation, CAS
Beijing, China
kqhuang@nlpr.ia.ac.cn

Yali Du*
King's College London
London, the United Kingdom
yali.du@kcl.ac.uk

ABSTRACT

Safe reinforcement learning (RL) agents accomplish given tasks while adhering to specific constraints. Employing constraints expressed via easily-understandable human language offers considerable potential for real-world applications due to its accessibility and non-reliance on domain expertise. Previous safe RL methods with natural language constraints typically adopt a recurrent neural network, which leads to limited capabilities when dealing with various forms of human language input. Furthermore, these methods often require a ground-truth cost function, necessitating domain expertise for the conversion of language constraints into a well-defined cost function that determines constraint violation. To address these issues, we proposes to use pre-trained language models (LM) to facilitate RL agents' comprehension of natural language constraints and allow them to infer costs for safe policy learning. Through the use of pre-trained LMs and the elimination of the need for a ground-truth cost, our method enhances safe policy learning under a diverse set of human-derived free-form natural language constraints. Experiments on grid-world navigation and robot control show that the proposed method can achieve strong performance while adhering to given constraints. The usage of pre-trained LMs allows our method to comprehend complicated constraints and learn safe policies without the need for ground-truth cost at any stage of training or evaluation. Extensive ablation studies are conducted to demonstrate the efficacy of each part of our method.

KEYWORDS

Safe Reinforcement Learning, Natural Language Constraints, Language Models

*Correspondence.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Xingzhou Lou, Junge Zhang, Ziyan Wang, Kaiqi Huang, and Yali Du. 2024. Safe Reinforcement Learning with Free-form Natural Language Constraints and Pre-Trained Language Models. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

In recent years, great success has been achieved with reinforcement learning (RL) algorithms on multiple domains such as robotic control [20], resource allocation [43] and games [9, 26, 35]. In real-world settings such as autonomous driving [14], RL agents are not given complete freedom for safety, fairness or other concerns and must adhere to specific constraints provided by humans. To this end, safe RL is widely studied, following the constrained Markov Decision Process (CMDP) [2] framework, where agents must satisfy the constraints given by auxiliary costs when optimizing their policies to maximize the objective function.

To learn a safe policy, safe RL algorithms [1, 13, 28, 38, 40] utilize ground-truth cost value given by a well-defined cost function in CMDP. Because the conditions for violating constraints are different, cost functions need to be defined for all constraints case by case and require task-specific domain knowledge, which can be very expensive and inefficient. For example, in a navigation task, the agent should not get close to objects that may cause damage to it. To define such a cost function, one must be aware of *all* the harmful objects in the environment, otherwise the agent may be harmed by some unexpected hazards. This severely limits safe RL's application to general and complex tasks where transforming constraints into cost functions can be expensive and requires specific domain knowledge.

One intuitive and easily-accessible approach to regulate safe RL agents is to provide constraints via human language [19, 21, 27, 39]. By providing natural language constraints, potential end users can easily regularize and interact with agents. However, without a ground-truth cost function, it is difficult to determine whether natural language constraints are violated. To avoid using ground-truth costs, previous method [39] learns to identify entities relevant to the natural language constraints in the environment so that agents can predict whether the constraints are violated. However,

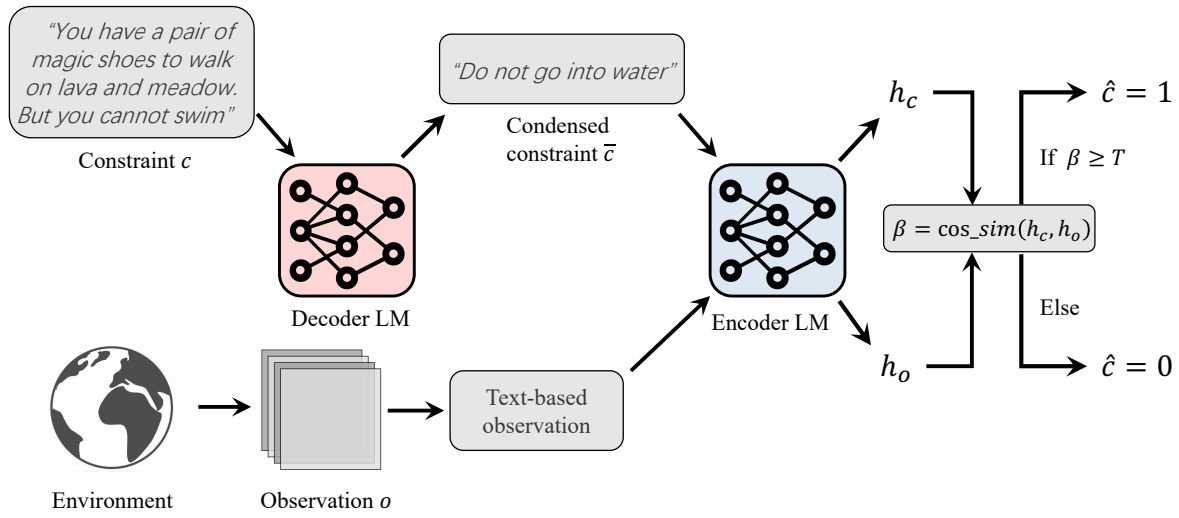


Figure 1: Cost prediction in the proposed method. The decoder LM condenses the semantic meaning of the constraint to eliminate ambiguity and redundancy. The encoder LM encodes the condensed constraints and text-based observations into embeddings according to their semantic meaning. If cosine similarity between the embeddings is greater than threshold T, the model will predict the constraint is violated (predicted cost $\hat{c} = 1$), otherwise $\hat{c} = 0$. Embedding h_c is also used later as input to condition the policy network.

it requires additional computation to model the entities and may result in inaccurate predictions in complex environments. And since the model is trained in a supervised fashion, it is incapable of handling free-form or new natural language constraints from humans.

In this paper, to deal with the problem of constraint violation prediction in safe RL with natural language constraints, we propose to adopt pre-trained language models (LM) [4, 8] to comprehend and encode the constraints and predict costs for policy learning. Specifically, a decoder-based LM (such as GPT [4]) is adopted to eliminate ambiguity and redundancy and extract semantic meaning from the constraints. As decoder LMs are pre-trained on a very large corpus of data and align with human values, they are able to generate high-quality constraints with condensed semantic meaning if we prompt them properly. To condition the policy which is parameterized by neural networks with natural language constraints, an encoder-based LM (such as BERT [8]) encodes constraints into embeddings according to their semantic meaning. We also use the encoder LM to encode the text-based observations. By computing the semantic similarities between constraint embeddings and description embeddings, we can determine whether the constraint is violated. To distinct constraint embeddings from each other, we adopt a contrastive loss to fine-tune the encoder LM so that constraints have similar embeddings only when they have similar semantic meaning.

We empirically validate the effectiveness of our method in two environments: Hazard-World-Grid [6] and SafetyGoal from Safety-Gymnasium [18]. Hazard-World-Grid is a grid-world navigation task, and SafetyGoal is aimed at safe robot control. In both environments, agents must navigate the world to achieve the goal and avoid potential hazards according to the constraints given by free-form natural language constraints from humans. Compared to constraints given by structured language in previous works [27, 39], pre-trained LMs' strong capabilities in natural language processing allow our method to deal with much more complicated forms of constraints. Our method achieves strong performance regarding both reward and cost without the need for ground-truth costs at any stage of training or evaluation. Experimental results demonstrate that the proposed method is able to constrain the agents effectively. Extensive ablation studies show the decoder LM and encoder LM are both necessary in order to predict costs accurately.

Contributions of this paper are three-fold:

(1) We introduce pre-trained LMs to safe RL with natural language constraints to replace the recurrent neural network in previous works, which can only handle structured or fixed-form natural language constraints.
(2) An encoder LM and a decoder LM are adopted for accurate cost prediction, which is essential for safe policy learning. The LM-based cost prediction enables agents to learn safe policy without the need for ground-truth cost at any stage of training.
(3) Experiments on grid-world navigation and robot control both show the proposed method can make agent learn safe policies to follow the constraints. Extensive ablation studies confirm the efficacy of our LM-based cost prediction module.

2 RELATED WORK

2.1 Safe Reinforcement Learning

Safe RL predominantly aims to train policies that maximize rewards while obeying specific constraints[11, 14]. Safety is a significant practical problem in RL's real-world applications, especially

in safety-critical scenarios such as autonomous driving [22] and power allocation [23]. The most straightforward safe RL algorithm is PPO-Lagrangian [28], which uses adaptive penalty coefficients to enforce constraints with a Lagrangian multiplier to the reward sum objective and solve the problem by solving the equivalent max-min optimization method. Constrained Policy Optimization (CPO) [1] is a policy search algorithm that derives policy improvement steps that guarantee both an increase in reward and satisfaction of constraints on costs. Projection-Based Constrained Policy Optimization (PCPO) [40] first performs local reward improvement update and then projects the updated policy to the constraint set to make sure the updated policy is always safe. However, typical safe RL algorithms require ground-truth costs to do policy updates, which are often unavailable in tasks where constraints are only given by natural language.

2.2 Reinforcement Learning with Natural Language

Multiple previous studies have combined reinforcement learning with natural language. RL is widely used in text-based tasks such as machine translation [37] and text games [7] to achieve strong performance. Similarly, He et al. [15] studied RL agents with a natural language action space. Natural language can be used to instruct or inform agents to get higher rewards. Kaplan et al. [19] used natural language instructions from an expert to aid RL agents in accomplishing the tasks. Agents receive additional rewards if they follow the expert's natural language instructions. Goyal et al. [12] proposed the use of natural language instructions to perform reward shaping to improve sample efficiency of RL algorithms. Previous works also used natural language to constrain agents to behave safely. Prakash et al. [27] trained a constraint checker in a supervised fashion to predict whether the natural language constraints are violated and guide RL agents to learn safe policies. However, they required ground-truth cost to train the constraint checker, which may be unavailable in many tasks. Yang et al .[39] trained a constraint interpreter to predict which entities in the environment may be relevant to the constraint and used the interpreter to predict costs. Although their method did not require ground-truth cost, the interpreter had to model and predict all entities in the environment, which required additional computation and may lead to inaccurate results in complex tasks. For comparison, our method does not need ground-truth costs and adopts pre-trained LMs to predict constraint violations, which avoids training extra modules and can harness the knowledge in large pre-trained LMs.

2.3 Pre-Trained Language Models

By utilizing the power of Transformer [34], pre-trained LMs have drawn enormous attention in recent years. Based on the encoder part of Transformer, encoder-based LMs such as Bidirectional Encoder Representations from Transformers (BERT) [8] extract semantic meaning and learn representations for text inputs by joint conditioning on their context and can be easily fine-tuned for downstream tasks. Based on the decoder part of Transformer, decoder-based LMs such as Generative Pre-trained Transformer (GPT) [25] and Llama [33] are trained to generate texts given previous contexts, and thus good at text generation tasks. Both encoder-based and decoder-based LMs are pre-trained on a very large corpus of text and thus possess strong natural language processing ability. Previous studies have tried to introduce pre-trained LMs to RL agents. Du et al. [10] utilized decoder LMs to generate novel goals for agents and encoder LMs to provide extrinsic rewards to encourage agents to achieve the generated goal and explore the environment. Hu et al. [17] used pre-trained LMs to generate policies conditioned on human instructions and help RL agents converge to equilibria aligned with human preferences, which significantly boosted their human-AI coordination performance. Nottingham et al. [24] proposed to utilize pre-trained LMs to hypothesize a world model to guide RL agents' exploration and improve sample efficiency. The hypothesized world model is also verified by world experience of RL agents from interactions. But to the best of our knowledge, our work is the first to apply pre-trained LMs to the field of safe RL.

3 PRELIMINARIES

Safe RL mostly models problems as Constrained Markov Decision Process (CMDP) [2]. CMDP consists of a tuple $< S, A, T, R, \gamma, C >$, where S is the state space, A is the action space, T is the state transition function, R is the reward function, C is some given cost function and γ is the discount factor. The objective for the agent is to maximize accumulated reward $J = \mathbb{E}_\pi[\sum_{t=0} \gamma^t R_t]$ and minimize the accumulated cost $J_C = \mathbb{E}_\pi[\sum_{t=0} \gamma^t c_t]$ to satisfy the constraints at the same time. The learning objective is given by

$$\max_\pi J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t R_t \right] \quad s.t. \;\; J_C(\pi) \quad = \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t c_t \right] \leq H, \tag{1}$$

where H is the constraint violation budget.

However, we consider the problem where the cost function C is not known but instead implied by some free-form natural language. Thus, instead of CMDP, we model the problem by another tuple $< S, A, T, R, \gamma, M, X >$, which is an MDP augmented by a constraint transformation function M and natural language constraint space X. $M : X \rightarrow C^x$ maps some natural language constraint $x \in X$ to a cost function C^x, where $C^x : S \times A \rightarrow \{0, 1\}$ decides whether the agent has violated the natural language constraints. The agent only knows the constraint x and has no knowledge of ground-truth cost $C^x(s_t, a_t)$ under any circumstances. The constraint x is sampled at the beginning of each episode and remains consistent within the episode. As each natural language constraint x corresponds to a cost function C^x, we will refer natural language constraints with variable c in what follows.

4 METHODOLOGY

In this section, we introduce our proposed method. We first introduce our LM-based cost prediction module and then show how the policy is trained with the predicted costs.

4.1 Cost Prediction Module

We give the structure of the cost prediction module in Fig. 1. In general, to predict whether the constraint is violated, we adopt a decoder LM (GPT) to eliminate ambiguity and condense semantic meaning in free-form natural language constraint c and an encoder

LM (BERT) to extract semantic information from the condensed natural language constraint $\bar{c}$ and transform the constraints into embedding h_c. Condensed constraint $\bar{c}$ aligns with the original constraint c on which entity or behaviour the agents are prohibited from. This is necessary as free-form constraint c from humans may be verbose or semantically vague, which will severely affect the cost prediction result. Since decoder LM GPT aligns with human values [44], it can eliminate ambiguity within the human natural language constraints and better summarize the constraints. We empirically show in the experiment section that without the decoder LM, the performance of the cost prediction module will drop significantly. The prompt we use for the decoder LM is provided in the supplementary material.

To ensure that constraints with the same semantic meaning have similar embeddings, we use a contrastive loss to fine-tune the encoder LM. For a batch of natural language constraint pairs, the contrastive loss is given by

$$\mathcal{L}_c = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left(Y^i - cosine\_sim(h^i_{c_1}, h^i_{c_2})\right)^2\right], \tag{2}$$

where $h^i_{c_1}, h^i_{c_2}$ are embeddings of ith natural language constraint pair (c^i_1, c^i_2). The constraints are first processed by the decoder LM and then encoded by the encoder LM. Target $Y^i = 1$ when constraints c^i_1, c^i_2 prohibit the agent from the same entities or behaviours, otherwise $Y^i = 0$. Cosine similarity $cosine\_sim(h_{c_1}, h_{c_2}) = \frac{h_{c_1} \cdot h_{c_2}}{\|h_{c_1}\| \|h_{c_2}\|}$.

The contrastive loss enables the encoder LM to recognise constraints' semantic similarity [29]. As a result, constraints concerning the same entities and behaviours will have embeddings with high cosine similarity and vice versa.

The constraint embedding h_c is later used together with the observation to determine whether the constraint is violated. As h_c is semantic embedding, we have to pre-process observation o into text-based observation, so that it can be used for cost prediction with the semantic embedding of constraints. Text-based observations are widely used as they provide easily accessible information for LMs to process [10, 36, 41, 42]. The transformation can be done via template matching [5] or caption models with supervised learning [16]. In our experiments, a descriptor will automatically analyze observation-action pairs and give natural language descriptions based on predefined templates. After obtaining the text-based observation, the encoder LM will encode it into observation embedding h_o, whose semantic meaning contains information about the current situation of the agent. Finally, the predicted cost $\hat{c}$ is given by

$$\hat{c} = \begin{cases} 1 & \text{if } \ cosine\_sim(h_c, h_o) > T \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where T is a hyperparameter defining the threshold and sensitivity of the cost prediction module, h_c is the constraint embedding, h_o is the observation embedding, and $cosine\_sim$ is cosine similarity. $\hat{c} = 1$ means that the cost prediction module predicts the constraint has been violated. The insight in devising such a cost function is that If the constraint and observation are semantically similar to each other, it is very likely that the agent has taken undesired actions or encountered undesired situations, resulting in an observation similar to the constraints.

Algorithm 1 PPO with LM-based Cost Prediction (PPO-CP)

Initialize value function network ϕ, cost value function network ϕ_c, policy network θ, decoder LM M_d and encoder LM M_e, Lagrange Multiplier update stepsize η
for each episode **do**
  Sample a natural language constraint x
  Condense and extract the semantic meaning of x with M_d
  Encode the condensed constraint with M_e to get constraint embedding h_c
  Rollout the policy with constraint h_c and get trajectory $\{o_t, a_t, o'_t, r_t\}_{t=1,..,n}$
  for t **in** {1,..,n} **do**
    Transform o_t into text-based observation and encode it with M_e to get observation embedding h_o
    Predict cost $\hat{c}_t$ according to Eq. 3
  end for
  Calculate loss L^{VF} and L^{CVF} for value function ϕ and cost value function ϕ_c
  Calculate policy loss L^{CLIP} and L^{CLIP}_C with value function ϕ and cost value function ϕ_c
  Update value function ϕ, cost value function ϕ_c and policy network θ according to Eq. 8
  Update α by stepsize η to maximize L_α in Eq. 9.
end for

4.2 Policy Training

With the predicted cost $\hat{c}$, we are ready to train our safe RL agents. It is worth noting that our method does not require the ground-truth cost under any circumstances during training or evaluation, which is distinct compared to other existing safe RL algorithms. We integrate the cost prediction module to proximal policy optimization (PPO) [32] with Lagrange multiplier [28], so that the agents can maximize rewards while adhering to specific constraints at the same time.

We define the cost return, cost value function and cost advantage function as J_C, V^π_C and A^π_C in analogy to return J, value function V^π and advantage function A^π.

The policy π is obtained by

$$\pi = \arg\max_\pi J(\pi) - \alpha J_C(\pi),$$

where $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{T}\gamma^t r_t\right]$ is the expected reward sum under policy π and $J_C(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{T}\gamma^t \hat{c}_t\right]$ is the expected cost sum, where $\hat{c}_t$ are predicted cost at timestep t, and α is the Lagrange multiplier.

The training of value function V^π and cost value function V^π_C is updated by minimizing the corresponding mean squared TD-error

$$L^{VF} = \mathbb{E}_\pi\left[\left(r^t_{(i)} + \gamma V(s^{t+1}_{(i)}) - V(s^t_{(i)})\right)^2\right], \tag{4}$$

$$L^{CVF} = \mathbb{E}_\pi\left[\frac{1}{2}\left(\hat{c}^t_{(i)} + \gamma V_C(s^{t+1}_{(i)}, h_c) - V_C(s^t_{(i)}, h_c)\right)^2\right], \tag{5}$$

where h_c is the constraint embedding from the encoder LM, n is batch size and $\hat{c}^t$ is the predicted cost.

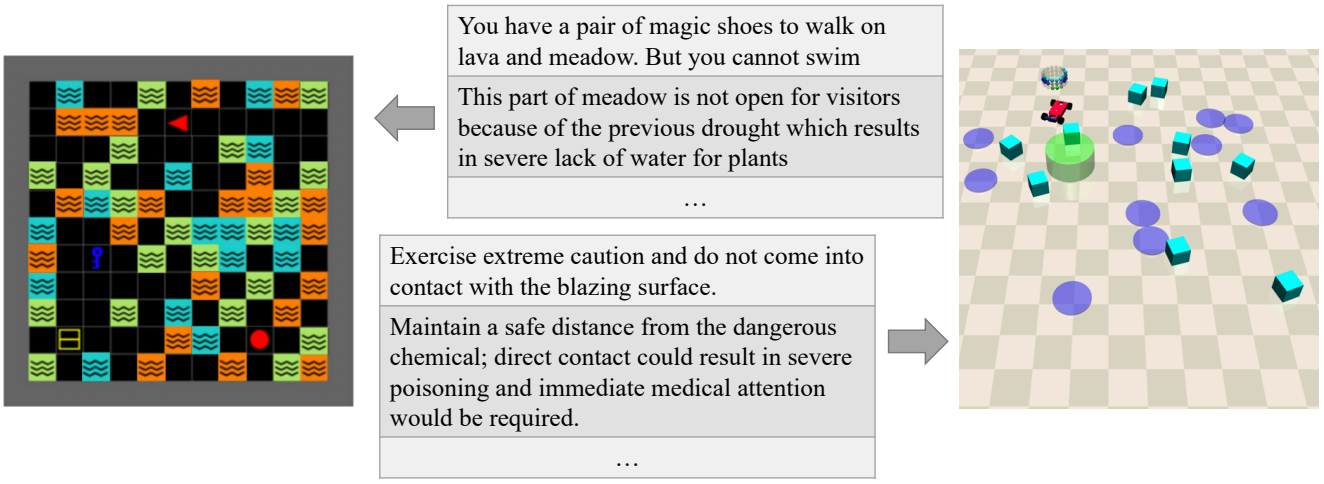


Figure 2: (a) One layout in Hazard-World-Grid, where orange tiles are lava, blue tiles are water and green tiles are grass. (c) Robot navigation task SafetyGoal built-in Safety-Gymnasium [18], where there are multiple types of objects in the environment. In both environments (a) and (c), agents have to reach goals while avoiding some type of terrain or objects specified in the natural language constraints. (b) Constraint examples for two environments in our experiments. Compared to constraints by structured language in previous works, constraints in our experiments are much more free-form and less intuitive.

To maximize return J and minimize cost return J_C, two clipped policy losses are adopted:

$$L^{CLIP} = \mathbb{E}_t\Bigg[\min\Bigg(\frac{\pi(a^t|s^t,h_c)}{\pi_{old}(a^t|s^t,h_c)}A^t, clip\Big(\frac{\pi(a^t|s^t,h_c)}{\pi_{old}(a^t|s^t,h_c)}, 1-\epsilon, 1+\epsilon\Big)A^t\Bigg)\Bigg], \tag{6}$$

$$L_C^{CLIP} = \mathbb{E}_t\Bigg[\min\Bigg(\frac{\pi(a^t|s^t,h_c)}{\pi_{old}(a^t|s^t,h_c)}A_C^t, clip\Big(\frac{\pi(a^t|s^t,h_c)}{\pi_{old}(a^t|s^t,h_c)}, 1-\epsilon, 1+\epsilon\Big)A_C^t\Bigg)\Bigg], \tag{7}$$

where advantage A^t is obtained by generalized advantage estimator [31] $A^t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_V^{t+l}$, $\delta_t^V = r + \gamma V(s^{t+1}) - V(s^t)$ is the TD-error. A_C^t is obtained by replacing V with cost value function V_C.

Together, the value networks and policy network is trained by maximizing the total loss

$$L = L^{CLIP} + \alpha L_C^{CLIP} - c_1 L^{VF} - c_2 L^{CVF}, \tag{8}$$

where c_1 and c_2 are coefficients for value network updates.

Lagrange multiplier α is updated after each training iteration. Loss for updating α is given by

$$L_\alpha = \alpha(\mathbb{E}_\pi[\sum_t \hat{c}_t] - H), \tag{9}$$

where H is the hyperparameter of cost budget from Eq. 1. By updating α with a small stepsize to maximize L_α, penalty coefficient α will increase if the expected cost sum is larger than H, and otherwise decrease (but will always be non-negative). This will enable agents to learn policies that satisfy the constraint quickly and achieve rewards as high as possible.

By doing the update in Eq. 8 iteratively, the agent will learn to maximize the reward and minimize constraint violations simultaneously. The agent will learn a safe policy and accomplish the given task while trying to follow the natural language constraints. The pseudo-code of the proposed method is given in Algorithm 1.

5 EXPERIMENT

5.1 Experiment Settings

We evaluate the proposed method on two tasks: Hazard-World-Grid [39] and SafetyGoal [18] shown in Fig. 2. Each task is accompanied by 20 different constraints to regularize the agents. A full list of detailed constraints for each environment is given in the supplementary material. The pre-trained decoder LM and encoder LM we use are *gpt-3.5-turbo* [4] and *all-MiniLM-L12-v2* from sentence BERT [30]. So, in this section, we also refer to the encoder LM as BERT and the decoder LM as GPT. The detailed constraints, prompts for the decoder LMs to condense constraints and how we transform observation into text-based observations are provided in the supplementary material. On each task, all methods are run for 4 times with different random seeds. The baseline methods we compare here are PPO [32], PPO-Lagrangian (PPO-Lag) [28] and CPO [1]. PPO only optimizes policies to maximize reward and does not consider the constraints. PPO-Lag enforces constraints with a Lagrangian multiplier to the reward objective and optimizes the policy by solving the equivalent min-max problem. CPO derives policy improvement steps that guarantee both an increase in reward and

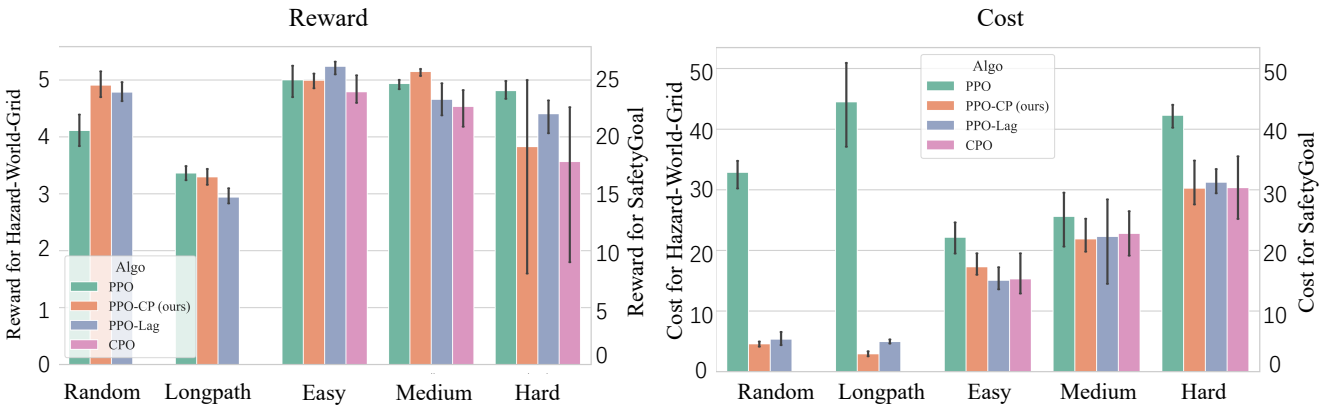


Figure 3: Experiment results on Hazard-World-Grid and SafetyGoal. *random* and *longpath* are two layouts in Hazard-World-Grid. *Easy, Medium* and *Hard* are three levels in SafetyGoal. There are two objects of each hazard type in *Easy*, four in *Medium* and six in *Hard*. Thus, in total, there are 8 hazard objects in *Easy*, 16 in *Medium* and 24 in *Hard*. PPO-Lag refers to the baseline method PPO-Lagrangian, and PPO-CP refers to the proposed method, where the suffix CP means cost prediction. From the results, the proposed method successfully learns a safer policy with the predicted cost. CPO is proposed for robot locomotion tasks. Thus, we do not include it in the first two tasks from Hazard-World-Grid. It is worth noting that in some tasks, the proposed method learns safer policies than the baselines using ground-truth costs. This is because the cost prediction module may mistakenly think of some safe but risky actions as violating the constraints, resulting in a more conservative policy.

satisfaction of constraints on costs and is a strong baseline for robot locomotion tasks with constraints. For these two methods, we input the natural language constraints encoded by our encoder LM to the policy network and provide them ground-truth costs to do policy learning, while our proposed method it does not have access to the ground-truth costs at all. Implementation and hyperparameters of the baseline methods can be found here [1]. Our method keeps the default hyperparameter and network details of PPO-Lag but use the predicted costs to replace the ground-truth costs.

Hazard-World-Grid In Hazard-World-Grid, agents navigate in a grid-world to find goal objects $\{ball, box, key\}$. Reward for reaching *ball* is 1, 2 for *box* and 3 for *key*. The reward will linearly decay to 0.1 times its original value w.r.t. timesteps within an episode, which means the sooner an object is found, the higher the reward. $\{lava, water, grass\}$ are hazards in the grid-world. In each episode, the agent will be told to avoid a specific kind of hazard by a natural language constraint. At the end of an episode, the ground-truth cost will be the total number of times that the constraint is violated. An episode terminates when all three goal objects are found, or maximum timesteps (300 in our experiments) are reached. Two layouts in this environment are used in our experiment: (1) *random*, where lava, water and grass tiles are randomly scattered in the grid-world as in Fig. 2 (a); (2) *longpath*, where the grid-world is filled with lava, except for a single safe path with many turns and no hazards on it. The difficulty of *longpath* is that the agent can only walk on the path if the constraint is to avoid lava. while in any other case, it can behave freely. Threshold T for cost prediction is 0.4 in this environment.

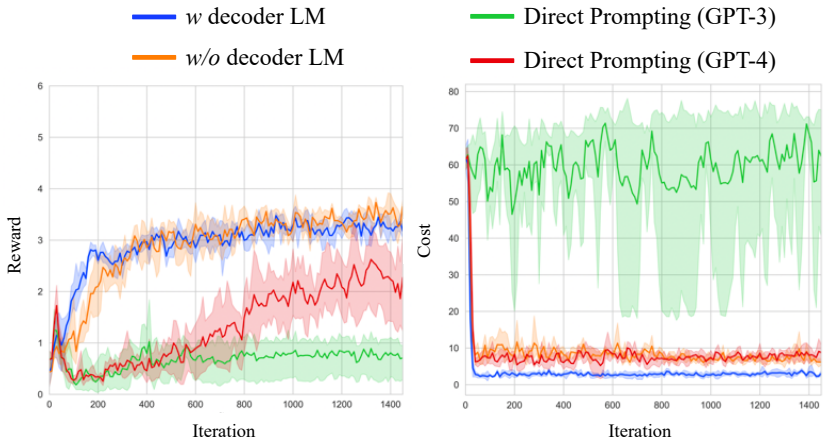
SafetyGoal In SafetyGoal, agents control a robot to navigate a 2D plane to reach the Goal's location while circumventing a given type of object. The objects are categorized into {poisonous-hazard,burning-hazard,radioactive-hazard,bio-hazard}. The goal will be randomly relocated when the robot reaches its location. Similar to Hazard-World-Grid, at the beginning of each episode, a natural language constraint will be given to tell the agent which type of object to avoid. The cost will be the total number of constraint violations within the episode. An episode terminates when maximum timesteps are reached (1000 in our experiments). According to the number of hazard objects of each type in the environment, we name the tasks *Easy*, *Medium* and *Hard*, where in *Easy*, there are 8 hazard objects in total, 16 in *Medium* and 24 in *Hard*. Threshold T for cost prediction is 0.55.

5.2 General Results

General results on all five tasks are given in Fig. 3. We report episode reward sum and episode cost sum, respectively to show how the methods perform. As we use cost prediction (CP) to replace ground-truth costs, we refer to our method as PPO-CP.

From the results, we can see that compared with PPO that does not consider constraints, PPO-CP can learn safe policies like the baseline methods that require ground-truth costs. This indicates that our cost prediction module is very effective and helpful for learning safe policies. In some tasks, PPO-CP even learns safer policies than the baseline methods, which may be because the cost prediction module mistakenly thinks of some risky but potentially safe actions as violating the constraints (like walking on the edge of the cliff). Thus, the policy is trained to avoid these risky actions, resulting in fewer constraint violations and lower costs. PPO-CP and PPO-Lag achieve even higher rewards than PPO on *Random* in Hazard-World-Grid. This is because they always tend to find the fastest way to obtain all the goal objects to finish the episode, as they will violate the constraints less if the episode ends fast. And

[1] https://github.com/PKU-Alignment/Safe-Policy-Optimization



(a) Reward (b) Cost

	Precision	*Recall*	*F1-score*
with decoder	78.8%	98.9%	87.7%
w/o decoder	72.9%	46.3%	56.6%
GPT-3	34.9%	53.7%	42.3%
GPT-4	50.7%	86.0%	63.8%

(c) Cost prediction

Figure 4: Experiment results for ablating decoder LM and directly prompting GPT for cost prediction. *w* decoder LM is our proposed method, *w/o* decoder LM removes the decoder LM and keeps the other modules such as cost prediction, Direct Prompting GPT-3 removes our cost prediction module and query GPT-3 with the constraint and text-based observation for cost prediction, and Direct Prompting GPT-4 replace GPT-3 with GPT-4 in Direct Prompting GPT-3. (a) gives the results of the episode reward. (b) is the results of episode cost and (c) gives the cost prediction results. The proposed method achieves the best performance on all three metrics while Directly prompting GPT-3 to perform the worst.

the reward will decay within an episode. Therefore, PPO-CP and PPO-Lag can achieve even higher rewards than PPO.

5.3 Ablation Study

5.3.1 Ablation on Contrastive Loss. To see the effectiveness of fine-tuning BERT (encoder LM) for cost prediction, we run the proposed method and compare the cost prediction results on layout *random* in Hazard-World-Grid with and without fine-tuning. The encoder LM is fine-tuned for 10 iterations. In each iteration, the encoder LM is trained with 128 randomly sampled condensed constraint pairs to minimize the contrastive loss in Eq. 2.

The cost prediction results are given in Table 1. From the results, we can see that our cost prediction module can predict costs accurately. Especially when the encoder LM is fine-tuned, the *F1-score* of cost prediction is improved by 6.7%. And it is worth noting that the *Recall* (true positive rate) is improved by a large margin. As a result, the cost prediction module is much more sensitive to constraint violations, and the learned policy will follow the constraints better.

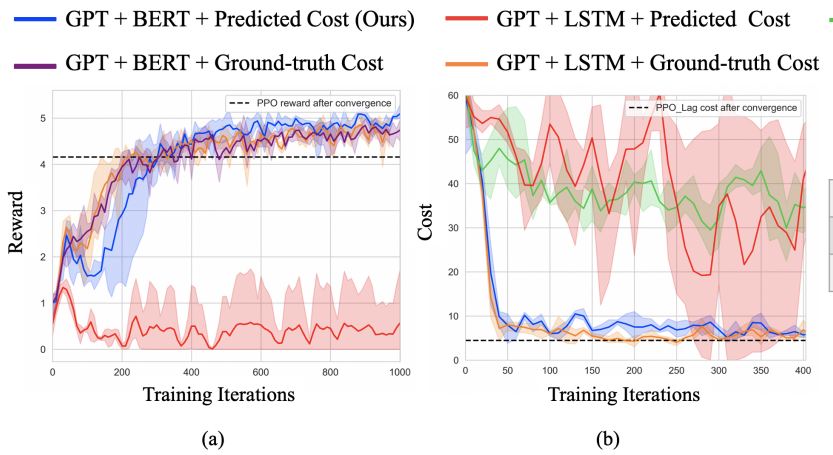
Table 1: Ablation study on contrastive loss in Eq. 2 to fine-tune encoder LM. Results show that contrastive loss can effectively improve cost prediction results, especially recall (true positive rate).

	Precision	*Recall*	*F1-score*
w contrastive	89.8%	98.0%	93.7%
w/o contrastive	87.4%	86.6%	87.0%

5.3.2 Ablation on Encoder LM. BERT is pre-trained on large text corpora so that it can encode natural language input according to their semantic meaning. This is essential for our cost prediction because we need to compare the similarity between the natural language constraint and text-based observation. In this section, we ablate the encoder LM BERT from the proposed method and replace it with a long-short-term-memory (LSTM) network. We remain the other modules of the proposed method, such as GPT as decoder LM, and run an ablation study on layout *random* in Hazard-World-Grid. As the randomly initialized LSTM cannot tell the difference among constraints, we also pre-train it with the contrastive loss for 10 iterations to make a fair comparison. The ablation results are given in Fig. 5. Labels are in the format of '*decoder model + encoder model + cost type*'. For example, '*GPT + BERT + Predicted Cost*' stands for using GPT as decoder LM, BERT as encoder LM and predicted cost from our cost prediction module for policy learning.

From Fig. 5(c), we can see the cost prediction module performs poorly with LSTM as an encoder. Consequently, training of '*GPT + LSTM + Predicted Cost*' collapses in Fig. 5(a) and (b). But when the ground-truth cost is available, '*GPT + LSTM + Ground-truth Cost*' has very strong performance, which means the LSTM network is able to encode natural language constraint to constrain the policy, but unable to extract semantic meaning of constraints and observations to do cost prediction.

5.3.3 Ablation on Decoder LM. Our proposed method adopts GPT as a decoder LM to condense semantic meaning and eliminate ambiguity within the natural language constraints. To see the efficacy of the decoder LM, we ablate the decoder LM and use the encoder LM to directly encode original natural language constraints (such as the ones given in Fig. 2(b)). Experiments are run on layout *longpath* in Hazard-World-Grid. We compare episode reward, episode cost and cost prediction results between the original method and the ablation.



(a) (b)

	Precision	*Recall*	*F1-score*
LSTM	12.3%	99.3%	21.9%
BERT	89.8%	98.0%	93.7%

(c)

Figure 5: (a) gives the result of episode reward and (b) gives the result of episode cost as training goes on. The black dashed line in (a) is the converged reward of PPO, and in (b) is the converged cost of *'GPT + BERT + Ground-truth Cost'*. The labels are in the format of *'decoder model + encoder model + cost type'*. For example, *'GPT + BERT + Predicted Cost'* stands for using GPT as decoder LM, BERT as encoder LM and predicted cost from our cost prediction module for policy learning. (c) gives the cost prediction results when using BERT and LSTM as the encoder model. The cost prediction with LSTM as encoder has very poor performance, which leads to the collapsed training of *'GPT + LSTM + Predicted Cost'* in (a) and (b).

From the results in Fig. 4, we can see after ablating decoder LM from the cost prediction modules, our method (*w* decoder LM) and *w/o* decoder LM achieve similar episodes reward, but our method violates the constraints much less than the ablation. The average constraint violation for *w* decoder LM is 3.2 per episode, while *w/o* decoder LM violates the constraints 6.8 times per episode on average. And we can see from Fig. 4(c) that the cost prediction results deteriorate dramatically, which demonstrates that it is necessary to adopt decoder LM to condense semantic meaning and eliminate ambiguity in the natural language constraints.

5.3.4 Direct Prompting. Through the acquisition of vast amounts of data, pre-trained large LMs, such as GPT [25], Llama 2 [33], and PaLM 2 [3], have demonstrated remarkable potential in achieving human-level intelligence. These models have strong capabilities in text generation tasks such as question answering. So, we directly prompt the large LMs with the text-based observation and natural language constraints and ask them to determine whether the constraints are violated. Then, we use the predicted costs by large LMs to train policies on layout *longpath* and compare experiment results with the proposed method. The prompts we use are given in the supplementary material. The pre-trained models we consider in this study are GPT-3.5 [4] and GPT-4 [25], which are state-of-the-art pre-trained large LMs.

The results for direct prompting are given in Fig. 4. Compared to other methods, the results show that Directly prompting GPT-3 performs poorly on all metrics, achieving the lowest reward and highest cost and struggling with cost prediction. As a successor of GPT-3, GPT-4 has a much stronger ability in causality. In our experiments, Directly prompting GPT-4 also results in better performance than GPT-3. With GPT-4 predicted costs, the policy can gradually learn to get high rewards, but it is still worse than using predicted cost by our cost prediction module. It is worth noting that directly prompting GPT-4 leads to a large improvement in the Recall rate, which means the number of False Negative predictions is dramatically reduced. As a result, directly prompting GPT-4 can also learn a safe policy compared to directly prompting GPT-3.

6 CONCLUSION AND FUTURE WORK

In this paper, we study the problem of safe RL with free-form natural language constraints and propose to use pre-trained LMs for cost prediction. A decoder LM and an encoder LM are adopted to process, comprehend and encode the constraints, which is later used to predict costs. Compared to previous methods dealing with safe RL with natural language constraints, our proposed method does not require ground-truth costs, can handle much more free-form constraints and does not need to learn an extra module to model entities within the environments. Empirically, we evaluate the proposed method on grid-world navigation tasks and robot control tasks against baseline methods that require ground-truth costs. Experiment results demonstrate that our method can accurately predict costs and learn safe policies that enable agents to accomplish given tasks while obeying the constraints. Extensive ablation studies are also conducted to show the efficacy of each module in our method.

In the future, besides learning safe policies with pre-trained LMs, we will also explore how to make agents' decisions more interpretable through pre-trained LMs, which is crucial for safety-critical tasks such as autonomous driving.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China (Grant No.2022ZD0116403) and Strategic Priority Research Program of Chinese Academy of Sciences (XDA27010300).

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.

[2] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC press.

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403* (2023).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Roberto Brunelli. 2009. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons.

[6] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic gridworld environment for openai gym. (2018).

[7] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*. Springer, 41–75.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[10] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692* (2023).

[11] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

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

[13] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. 2023. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence* 319 (2023), 103905.

[14] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. 2022. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330* (2022).

[15] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636* (2015).

[16] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CsUR)* 51, 6 (2019), 1–36.

[17] Hengyuan Hu and Dorsa Sadigh. 2023. Language instructed reinforcement learning for human-ai coordination. *arXiv preprint arXiv:2304.07297* (2023).

[18] Jiaming Ji, Borong Zhang, Xuehai Pan, Jiayi Zhou, Juntao Dai, and Yaodong Yang. 2023. Safety-Gymnasium. https://github.com/PKU-Alignment/safety-gymnasium. *GitHub repository* (2023).

[19] Russell Kaplan, Christopher Sauer, and Alexander Sosa. 2017. Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539* (2017).

[20] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2023. Champion-level drone racing using deep reinforcement learning. *Nature* 620, 7976 (2023), 982–987.

[21] Ilias Kazantzidis, Timothy J Norman, Yali Du, and Christopher T Freeman. 2022. How to Train Your Agent: Active Learning from Human Preferences and Justifications in Safety-critical Environments. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1654–1656.

[22] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 6 (2021), 4909–4926.

[23] Yasar Sinan Nasir and Dongning Guo. 2019. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications* 37, 10 (2019), 2239–2250.

[24] Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050* (2023).

[25] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[26] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. 2022. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (2022), 990–996.

[27] Bharat Prakash, Nicholas Waytowich, Ashwinkumar Ganesan, Tim Oates, and Tinoosh Mohsenin. 2020. Guiding safe reinforcement learning policies using structured language constraints. *UMBC Student Collection* (2020).

[28] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708* 7, 1 (2019), 2.

[29] Nils Reimers and Iryna Gurevych. 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. https://arxiv.org/abs/1908.10084

[30] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[31] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[33] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[35] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[36] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[37] Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866* (2018).

[38] Long Yang, Jiaming Ji, Juntao Dai, Linrui Zhang, Binbin Zhou, Pengfei Li, Yaodong Yang, and Gang Pan. 2022. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems* 35 (2022), 9111–9124.

[39] Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. 2021. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems* 34 (2021), 13794–13808.

[40] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2020. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152* (2020).

[41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).

[42] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2023. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485* (2023).

[43] Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C Parkes, and Richard Socher. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances* 8, 18 (2022), eabk2607.

[44] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and c Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).