Learning Symbolic Rules over Abstract Meaning Representations for Textual Reinforcement Learning

Subhajit Chaudhury, Sarathkrishna Swaminathan, Daiki Kimura, Prithviraj Sen, Keerthiram Murugesan, Rosario Uceda-Sosa, Michiaki Tatsubori, Achille Fokoue, Pavan Kapanipathi, Asim Munawar and Alexander Gray
{subhajit, sarath.swaminathan, keerthiram.murugesan, asim, alexander.gray}@ibm.com
{daiki, mich}@jp.ibm.com, {rosariou, achille, kapanipa}@us.ibm.com

IBM Research

Abstract

Text-based reinforcement learning agents have predominantly been neural network-based models with embeddings-based representation, learning uninterpretable policies that often do not generalize well to unseen games. On the other hand, neuro-symbolic methods, specifically those that leverage an intermediate formal representation, are gaining significant attention in language understanding tasks. This is because of their advantages ranging from inherent interpretability, the lesser requirement of training data, and being generalizable in scenarios with unseen data. Therefore, in this paper, we propose a modular, NEuro-Symbolic Textual Agent (NESTA) that combines a generic semantic parser with a rule induction system to learn abstract interpretable rules as policies. Our experiments on established text-based game benchmarks show that the proposed NESTA method outperforms deep reinforcement learning-based techniques by achieving better generalization to unseen test games and learning from fewer training interactions.

1 Introduction

Text-based games (TBGs) (Côté et al., 2018) serve as popular sandbox environments for evaluating natural language-based reinforcement learning. The agent observes the state of the game in pure text and issues a textual command to interact with the environment. TBGs are partially observable where the full state of the world is hidden and action commands facilitate the agent to explore the unobserved parts of the environment. The reward signal from the environment is used to improve the agent's policy and make progress in the game.

Text-based games sit at the intersection of two research areas, i.e., language understanding and reinforcement learning. Existing RL agents for TBGs primarily use embeddings for observation as representations and are fed to an action scorer for predicting the next action (Narasimhan et al., 2015a; Yuan et al., 2019; He et al., 2016), ignoring the advances in language understanding. On the other hand, there has been a recent surge in neuro-symbolic techniques, particularly those that use symbolic representations, for better language understanding (Lu et al., 2021; Kapanipathi et al., 2021) through reasoning. In light of exploring such advances for text-based reinforcement learning, this work proposes a neuro-symbolic approach. Our approach, named NESTA (NEuro Symbolic Textual Agent) is a modular approach comprising a generic semantic parser in combination with a symbolic rule induction system as shown in Figure 1. The semantic parser translates text into the form of symbolic triples. NESTA uses Abstract Meaning Representation (Banarescu et al., 2013) as the initial parse which is then transformed into triples. This symbolic representation is used by an adaptation of the Inductive Logic Programming (ILP) sys-

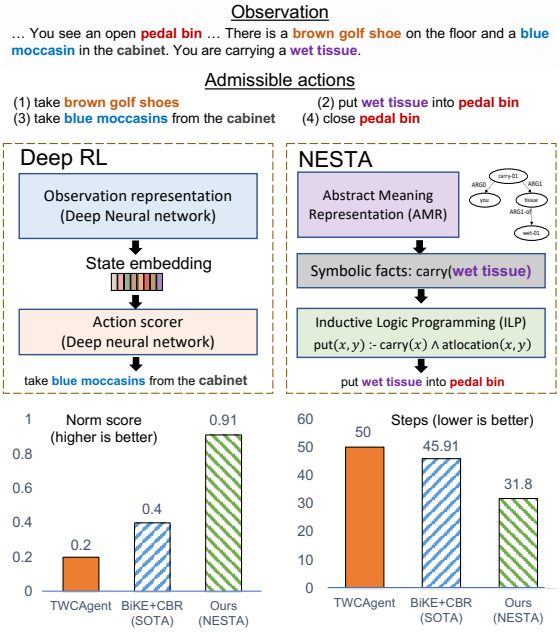

Figure 1: Our proposed NESTA model learns interpretable action rules as policy using ILP and outperforms SOTA deep RL methods on TBGs.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 6764–6776

tem using Logical Neural Networks (Riegel et al., 2020) for learning horn clauses as action rules.

NESTA, in comparison to other end-to-end learning approaches, has the following advantages: (a) modular language understanding using pre-trained large language models enabling our system to leverage the advances in semantic parsing. While such modular semantic parsing-based techniques have been around for other NLP tasks such as reading comprehension (Mitra and Baral, 2016; Galitsky, 2020), knowledge base question answering (Kapanipathi et al., 2021), and natural language inference (Lien and Kouylekov, 2015), this work is the first to demonstrate the application for TBGs ; (b) learning symbolic rules for model-free RL using a neuro-symbolic framework facilitates inherent interpretability and generalizability to unseen situations (Ma et al., 2021; Jiang and Luo, 2019; Dong et al., 2019). The rules learned by NESTA are abstract and not specific to entities in the training data. These abstract action rules in policies for TBGs enable reasoning over unseen entities during training.

Our main contributions in this work are: (1) We propose a novel and modular neuro-symbolic agent named NESTA. To the best of our knowledge, NESTA is the first to use a generic semantic parser with a rule learning system for TBGs, (2) Our empirical analysis of commonsense-aware textworld games shows that NESTA outperforms deep RL methods by a significant margin. We also show that NESTA has better sample efficiency compared to traditional text-based RL agents obtaining better test performance with up to $5\times$ lesser training interactions, and (3) Our method produces interpretable abstract rules from the rule induction system.

2 NEuro-Symbolic Textual Agent

Text-based RL agents for TBGs interact with the environment using text-only action commands and obtain feedback solely as textual observations. As the agent does not have access to global state information, it is modeled as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998) represented as $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \Omega, \mathcal{O})$, where $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$ represent a Markov Decision Process. Ω represents the finite set of all observations, and $\mathcal{O}$ represents the observation function representing the conditional distribution over observations for a given action and next state. The goal of the agent is to learn optimal action probabilities at each step such that the expected future reward is maximized.

We present **NE**uro-**S**ymbolic **T**extual **A**gent (**NESTA**), a modular approach for TBGs. Figure 1 illustrates the overview of NESTA which comprises of three primary components: (a) *Semantic Parser*, which extracts symbolic representation of the text using AMR as the generic semantic representation, (b) *Rule Learner*, an ILP-based rule induction module, which learns logical rules that abstract out the entities in the games, making these rules generally applicable to test games containing unseen entities, and (c) *Pruner*, that reduces the amount of branching factor at each step by pruning actions that do not contribute to the expected future reward. Below, we describe these components in detail.

2.1 Semantic Parser: Text to symbolic triples using AMR

The first step in NESTA is to translate the text into symbolic representation. To this end, inspired by works that address different NLP tasks (Kapanipathi et al., 2021; Galitsky, 2020; Mitra and Baral, 2016), we use an AMR parser as a generic semantic parser. The use of a generic semantic parse such as AMR allows the system to benefit from independent advances in AMR research. For example, the performance of AMR has improved in Smatch score (Cai and Knight, 2013) from 70.9 (Van Noord and Bos, 2017) to 86.7 (Lee et al., 2022) on LDC2017T10 in the last few years due to advances in large language models. The AMRs are subsequently transformed into a symbolic form using a deterministic AMR-to-triples approach.

Abstract Meaning Representation (AMR): AMR parsing produces rooted, directed acyclic graphs from the input sentences, where each node represents concepts from propbank frames (Kingsbury and Palmer, 2002) or entities from the text. The edges represent the arguments for the semantic frames. Fig. 2 shows the AMR graph generated from the sentence "*There is a brown golf shoe and a blue moccasin on the cabinet.*". The resultant AMR graph is rooted at the propbank frame `be-located-at-91` with ARG1 and ARG2 edges leading to its children. The other parts of the graph are used to describe the entities for "brown golf shoe" and "blue moccasin". We use StructBART (Zhou et al., 2021) for parsing a text to AMR.

AMR-to-triples: We design an AMR-to-triples

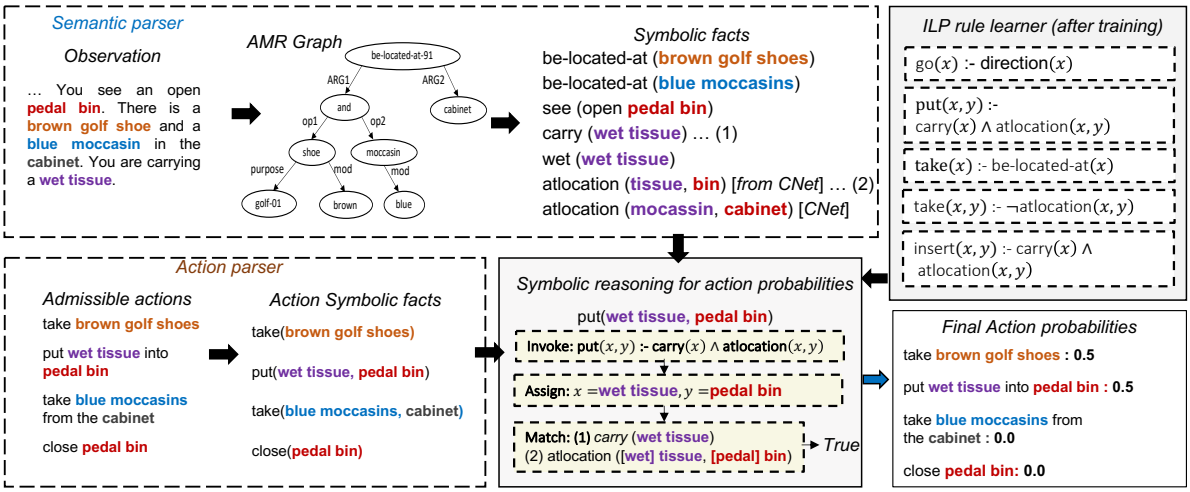

Figure 2: Overview of our method for neuro-symbolic reasoning. Our methods first extract symbolic facts from the surface text observations. During training, NESTA learns first-order lifted rules based on the reward signals. The learned rules are then used for obtaining the action probabilities.

module to extract a set of symbolic facts consisting of generic domain-agnostic predicates from the AMR semantic representation. Fig. 2 shows the extraction of facts from AMR. The AMR-to-triples module performs a set of graph operations to extract propbank nodes as the predicates and the children entities as the arguments. In the example, the two operands of the "and" node are converted into two symbolic facts with `be-located-at` predicate with two separate entities of "brown golf shoe" and "blue moccasins". We convert the symbolic facts to unary predicates. For example, we convert the `be-located-at` predicate with two arguments into single argument facts. These simplifications result in some loss of representational power but make the task of rule learning simpler. We also add the commonsense predicates from conceptnet subgraph (Speer et al., 2017) provided by the TWC environment (Murugesan et al., 2020).

2.2 Rule Learner: ILP from Rewards

In order to learn interpretable rules that can be debugged by humans, we use the symbolic representation obtained from the above step. Such symbolic rules are learned from reward signals by interacting with the environment. For this purpose, we use Inductive Logic Programming (ILP) in an RL setting with the objective of expected future reward maximization. We use Logical Neural Networks (LNN) as the differentiable rule learning engine.

Logical Neural Networks: LNN (Riegel et al., 2020) proposes a differentiable rule learning framework that retains the benefits of both neural networks and symbolic learners. It proposes a logical neuron that has the core properties of gradient-based learning similar to a standard neuron but adds logic-aware forward functions and constrained optimization making it suitable for logical operations. This can be illustrated on 2-input logical conjunction (AND) neuron with (x, y) as two logical inputs to the conjunction node. The LNN conjunction neuron generalizes the classical AND logic for real-valued logic by defining a noise threshold (α). The real-values in $[\alpha, 1]$ and $[0, 1-\alpha]$ signify a logical *high* and logical *low* respectively. To emulate an AND neuron, LNN uses the standard truth table of the conjunction (AND) gate to obtain the following constraints,

$$\begin{aligned} f(x,y) &\leq 1-\alpha, \quad \forall x, y \in [0, 1-\alpha] \\ f(x,y) &\leq 1-\alpha, \quad \forall x \in [0, 1-\alpha], y \in [\alpha, 1] \\ f(x,y) &\leq 1-\alpha, \quad \forall x \in [\alpha, 1], y \in [0, 1-\alpha] \\ f(x,y) &\geq \alpha, \quad \forall x, y \in [\alpha, 1] \end{aligned}$$

LNN uses the forward function as the weighted Łukasiewicz t-norm, $f(x, y; \beta, w_1, w_2) = \beta - w_1(1-x) - w_2(1-y)$, where β, w_1, w_2 are the bias and weights of the inputs. Given a target label, the weights and biases are tuned to learn the logical rule that best describes the data.

ILP-based reward maximization: Our ILP rule learner is based on the LNN rule learning implementation in Sen et al. (2022). However, our rule-learning model makes significant modifications to adapt the previous algorithm for model-free policy optimization suitable for text-based RL. Consider the state transition at time step t as (o_t, a_t, r_t, o_{t+1}), where o_t represents the textual observation, a_t is

the action command that yields the reward r_t and takes the agent to the next state with observation o_{t+1}. AMR-to-triples semantic parser is used to obtain the symbolic state s_t (list of symbolic facts) from o_t as shown in Figure 2. At each step, the agent has to choose from a set of admissible action commands which are also converted to their symbolic form. Starting from an initial random policy π, we sample trajectories $\tau \sim \pi$ and store the transitions (s_t, a_t, r_t, s_{t+1}) in a buffer $\mathcal{B}$. We also store the admissible actions set adm_t and the discounted future reward $g_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, for each step in the buffer, where γ is the discount factor.

From the buffer $\mathcal{B}$, we find a set of template predicates $\mathcal{P} = \{p \mid p \in s_t, \quad \text{for } s_t \in \mathcal{B}\}$, where $p \in s_t$ operation states whether facts with predicate p exist in the symbolic state s_t. We also obtain a set of action predicates $\mathcal{A} = \{a \mid a \in \text{adm}_t, \text{ for } \text{adm}_t \in \mathcal{B}\}$ finding all action predicates in the admissible action set. We initialize ILP rule learner $\pi_a(\theta)$ for each action predicate $a \in \mathcal{A}$. Action predicates for TBGs typically coincide with the action verbs. The LNN policy is formulated as a weighted conjunction operation over the template predicates $\mathcal{P}$. The likelihood of action a for abstract lifted variables x, y is given as a conjunction template over the predicate list as follows: unary action likelihood is given as $L(a(x)|s_t) = \bigwedge_k w_k p_k(x)$ and binary action likelihood is formulated as $L(a(x,y)|s_t) = \bigwedge_k w_k p_k(x) \bigwedge_m w_m q_m(x,y)$. The predicates p_k and q_m are 1 and 2 arity predicates in $\mathcal{P}$ respectively and $\bigwedge$ represents the LNN's logical conjunction operator. The weights w_k and w_m constitute the LNN parameters θ that are updated during training. At any given step, the likelihood of each action is normalized over all actions in the admissible action set to obtain the action probabilities.

For training the rule learning model $\pi_a(\theta)$ for a specific action a, we only extract transitions from the buffer containing the action a and store it in a sub-buffer $\mathcal{B}_a$. The model is updated following the policy gradient loss, $\mathcal{L} = \nabla_\theta \mathbb{E}_{(s_t,g_t)\sim\mathcal{B}_a} \log(\pi_a(a_t = a|s_t) g_t$, where the trajectories are sampled from $\mathcal{B}_a$. We assume that π_a gives normalized probabilities for this loss formulation. Therefore, this training procedure yields separate rules learned for each action predicate. Figure 2 shows the learned rules for each action.

Generalization under Distribution Shift: Having learned the action rules for each action predicate using dedicated ILP models, NESTA uses the rules for obtaining the action probabilities at each step. This process consists of three steps: (a) For each action in the admissible action list, **invoke** the learned rule for that action predicate, (b) **Assign** the abstract variables with the symbolic action arguments, and (c) **Match** the symbolic facts using entity alignment by root noun matching (instead of an exact match). The probabilities are then obtained by the LNN conjunction node feed-forward operation based on the current weights. This procedure is also used for sampling during training.

Figure 2 shows the reasoning steps for fixed weights after training is complete. Since the rules learned by NESTA abstract out the entities in the form of lifted variables, human interpretability and generalization to unseen entities is a natural advantage of our method. In addition to this, since we modularize language understanding and RL policy learning into separate modules, our LNN symbolic learner can solely focus on optimal reward performance leading to sample-efficient learning.

2.3 Pruner: Irrelevant Action Pruning by Look-Ahead

The third module in NESTA, tackles the large action space problem in TBGs by removing actions from the admissible commands that do contribute to future rewards in the games. A large number of possible actions at each step can increase the branching factor of the agent at each step during the training and testing leading to a combinatorially large search problem. We employ a look-ahead strategy to find out which actions do not contribute to future reward accumulation. For example, the action `examine`(x) returns the description of the entity x, but does not change the state of the game and does not contribute to future rewards. However, for the action `take`(x), although an immediate reward is not obtained on execution, it leads to a future reward when the object x is put in the correct container y using the `put`(x, y) command. Therefore, the action command of type `examine`(x) can be pruned but `take`(x) is essential and hence cannot be pruned. This can be computed by looking ahead from the current step and comparing the future reward if that particular action was removed from the trajectory.

Due to AMR error propagation and undesirable credit assignment (for example, `examine`(x) command issued just before a rewarded action), the rule

	Easy		Medium		Hard	
Methods	**Steps**	**Norm. Score**	**Steps**	**Norm. Score**	**Steps**	**Norm. Score**
Text	23.83 ± 2.16	0.88 ± 0.04	44.08 ± 0.93	0.60 ± 0.02	49.84 ± 0.38	0.30 ± 0.02
Text+CS	20.59 ± 5.01	0.89 ± 0.06	42.61 ± 0.65	0.62 ± 0.03	48.45 ± 1.13	0.32 ± 0.04
KG-A2C	22.10 ± 2.91	0.86 ± 0.06	41.61 ± 0.37	0.62 ± 0.03	48.00 ± 0.61	0.32 ± 0.00
BiKE	18.27 ± 1.13	0.94 ± 0.02	39.34 ± 0.72	0.64 ± 0.02	47.19 ± 0.64	0.34 ± 0.02
BiKE+ CBR	15.72 ± 1.15	0.95 ± 0.04	35.24 ± 1.22	0.67 ± 0.03	45.21 ± 0.87	0.42 ± 0.04
NESTA	$\mathbf{2.40 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	31.44 ± 2.08	0.80 ± 0.04	42.68 ± 6.01	$\mathbf{0.85 \pm 0.05}$
NESTA + OR	3.44 ± 2.08	$\mathbf{1.00 \pm 0.00}$	$\mathbf{11.76 \pm 1.78}$	$\mathbf{0.98 \pm 0.03}$	$\mathbf{35.84 \pm 7.88}$	$\mathbf{0.85 \pm 0.09}$
Human	2.12 ± 0.00	1.00 ± 0.00	5.33 ± 0.00	1.00 ± 0.00	15.00 ± 0.00	1.00 ± 0.00

Table 1: Proposed NESTA model shows better performance in terms of normalized score and steps to reach the goal compared to Deep RL methods on unseen TWC *in-distribution* games.

learner can assign high action probabilities to non-contributing actions. Therefore, the pruner module is desirable to remove such action predicates (a) by evaluating the total reward in action trajectories with and without the particular action predicate a. More specifically, for each episodic trajectory, we remove the action predicates a and re-evaluate the episodic reward obtained from the environment. If the average episodic reward in both cases, with and without removal of a is the same then the action predicate is not contributing to the future reward. Therefore, it can be removed from the original action predicate set, $\mathcal{A}$ to obtain the pruned action set $\mathcal{A}_{\text{pruned}}$ for which LNN models are learned.

3 Outlier Rejection in Policy Training

The training samples that NESTA collects from interacting with the environment can be noisy and this can affect learning a good policy. There exists two sources of noise: (a) **AMR noise**, where AMR incorrectly parses the surface text resulting in erroneous identification of entity extraction or relationships between entities, and (b) **RL credit assignment noise**, where discounted reward gives reward to a suboptimal action taken right before a correct action. Although symbolic reasoners have the advantages of learning from fewer data and better generalization, they are not robust to noise. We mitigate the effect of noise in LNN policy training by using a consensus-based noise rejection method.

Our noise rejection method trains the LNN Policy on multiple subsets of training data and selects the model with the smallest training error as the best model. The multiple subsets of training data are prepared as follows - for each training subset, a particular predicate p from the predicate list $\mathcal{P}$ is given priority. We only choose state transitions that contain the predicate p ensuring that this predicate will be part of the final learned rule, thus eliminating the source of AMR noise for this predicate (such a subset is rejected if the number of such transitions is less than some threshold percentage). Subsequently, the resulting transitions are sorted by the discounted reward g_t and we only retain the top first $k\%$ of this sorted data as training data. This encourages action transition with more immediate average reward gains to constitute the training data.

4 Experimental results

Our experiments are designed to answer these questions that analyze if NESTA can overcome the common drawbacks of deep RL methods: (i) Can NESTA enable better generalization in test environments? (ii) Does NESTA improve upon sample efficiency while still maintaining good reward performance, (iii) Are the rules learned by NESTA, human interpretable? For comparing the performance of various methods, we use the metrics of *normalized score* (total reward from the games normalized by maximum reward) and *number of steps* to reach the goal (lower is better). Our experiments were conducted on Ubuntu 18.04 operating system with NVidia V100 GPUs.

4.1 Environment

We use the textworld commonsense (TWC) environment (Murugesan et al., 2020) for empirical evaluation of our method. The goal here is to clean up a messy room by placing the objects in the correct containers. The game provides conceptnet sub-graphs relating the game entities which are used as commonsense graphs. TWC provides two splits of testing games: (i) *in-distribution* games that have the same entities as training games but unseen object-container configuration, and (ii) *out-of-distribution* games that use new objects not seen during training. This provides a systematic framework for measuring generalization in NESTA and

	Easy		Medium		Hard	
Methods	**Steps**	**Norm. Score**	**Steps**	**Norm. Score**	**Steps**	**Norm. Score**
Text	29.90 ± 2.92	0.78 ± 0.02	45.90 ± 0.22	0.55 ± 0.01	50.00 ± 0.00	0.20 ± 0.02
Text+CS	27.74 ± 4.46	0.78 ± 0.07	44.89 ± 1.52	0.58 ± 0.01	50.00 ± 0.00	0.19 ± 0.03
KG-A2C	28.34 ± 3.63	0.80 ± 0.07	43.05 ± 2.52	0.59 ± 0.01	50.00 ± 0.00	0.21 ± 0.00
BiKE	25.59 ± 1.92	0.83 ± 0.01	41.01 ± 1.61	0.61 ± 0.01	50.00 ± 0.00	0.23 ± 0.02
BiKE + CBR	17.15 ± 1.45	0.93 ± 0.03	35.45 ± 1.40	0.67 ± 0.03	45.91 ± 1.32	0.40 ± 0.03
NESTA	**2.40 ± 0.00**	**1.00 ± 0.00**	5.56 ± 0.53	1.00 ± 0.00	38.88 ± 3.24	**0.94 ± 0.04**
NESTA + OR	3.28 ± 1.76	**1.00 ± 0.00**	**3.60 ± 0.00**	**1.00 ± 0.00**	**31.40 ± 6.38**	0.91 ± 0.05
Human	2.24 ± 0.00	1.00 ± 0.00	4.40 ± 0.00	1.00 ± 0.00	17.67 ± 0.00	1.00 ± 0.00

Table 2: Normalized score and number of steps to reach the final goal for various methods on unseen TWC *out-of-distribution* games. NESTA shows large improvements over previous Deep RL methods, especially for *hard* games. **OR** is Outlier Rejection.

other baseline agents for both within-training distribution and out-of-training distributions. Since we are focusing on generalization aspects, we do not use other textworld games (Côté et al., 2018; Hausknecht et al., 2020) because these environments primarily focus on the agent's exploration strategies and are therefore not suitable to evaluate the agent's generalization ability.

4.2 Agents

For baseline agents, we report performance by these deep RL-based methods: (1) **Text**-based agent that uses a GRU network for observation representation and action scorer units, (2) **TWCAgent (Text + CS)** that uses combined textual and commonsense embeddings for action scoring, (3) **KG-A2C** (Ammanabrolu and Hausknecht, 2020) that uses extracted knowledge graphs as input, (4) **BiKE** (Murugesan et al., 2021) which leverages graph structures in both textual and commonsense information and (5) **CBR** (Atzeni et al., 2021) which is the SOTA method using case-based reasoning for improving generalization in text-based agents. We did not compare with previous neuro-symbolic methods (Kimura et al., 2021; Chaudhury et al., 2021) because they use a hand-crafted game-specific predicate design scheme that was not available for TWC.

4.3 Generalization to Test Games

We evaluate the generalization ability of NESTA on TWC *easy*, *medium* and *hard* games. Table 1 and Table 2 shows the performance of baseline and our agents on *in-distribution* and *out-of-distribution* games, including the human performance from Murugesan et al. (2020). For the baseline models, we report scores from Atzeni et al. (2021). For NESTA, we report the mean of 5 independent runs.

For easy games, NESTA gets a perfect score outperforming previous games with similar steps as human performance. For medium and hard games, NESTA greatly surpasses the SOTA agent and needs a lesser number of steps for both *in-distribution* and *out-of-distribution* games. For medium *out-of-distribution* games, NESTA outperforms humans in terms of the number of steps. This might be due to the fact that during human annotation, the subjects would take a larger number of steps for the initial few games due to trial-and-error, thus increasing the average number of steps.

While easy and medium games have a single-room setting, hard games present a two-room setting where the agent might require picking up an object in room 1 and putting it in a container in room 2. This requires learning a complex strategy especially for generalizing to unseen entities. Our method NESTA scores significantly higher compared to SOTA on hard games, thus exhibiting the ability of our method to generalize in complex settings while deep RL methods fail to generalize due to overfitting the training data. Furthermore, our outlier rejection model helps improve the number of steps to reach the goal for both *in-distribution* and *out-of-distribution* games.

4.4 Ablation Results with Action Pruning

To study the effect of our action pruning module on deep RL agents, we implemented action pruning on the publicly available TWCAgent code from Murugesan et al. (2020). We follow the exact same methodology for TWCAgent that we used for the NESTA agent. Using the look-ahead method, we obtain $\mathcal{A}_{\text{retain}}$, the list of action verbs to retain at a specific episode (episode num 10 for this result). For all subsequent training steps, only action verbs $a \in \mathcal{A}_{\text{retain}}$ were retained from the admissible list.

In-distribution		
	Steps	**Norm. Score**
TWCAgent	47.77 ± 1.50	0.49 ± 0.04
TWCAgent + AP	47.14 ± 0.85	0.61 ± 0.03
NESTA	43.44 ± 4.67	0.77 ± 0.08
NESTA + AP	$\mathbf{35.84 \pm 7.88}$	$\mathbf{0.85 \pm 0.09}$
Out-of-distribution		
	Steps	**Norm. Score**
TWCAgent	50.00 ± 0.00	0.21 ± 0.05
TWCAgent + AP	50.00 ± 0.00	0.37 ± 0.02
NESTA	47.52 ± 2.34	0.60 ± 0.15
NESTA + AP	$\mathbf{31.40 \pm 6.38}$	$\mathbf{0.91 \pm 0.05}$

Table 3: Ablation study showing the effect of our proposed symbolic action pruning (**AP**) on NESTA and TWCAgent for *hard* games. Proposed action pruning method shows better improvements on **NESTA** model when compared to improvements on TWCAgent.

Learned rules for *hard* games by NESTA
go(x) : − direction(x)
take(x) : − be-located-at(x)
take(x, y) : − be-located-at(y)
put(x, y) : − carry$(x) \wedge$ atlocation(x, y)
insert(x, y) : − carry$(x) \wedge$ atlocation(x, y)
In-distribution norm score: 0.71 (Steps: 46.4)
Out-distribution norm score: 0.85 (Steps: 37.4)
After rule correction by human
take(x, y) : − $\neg$**atlocation**(x, y)
In-distribution norm score: **0.88** (Steps: **42.4**)
Out-distribution norm score: **1.0** (Steps: **19.8**)

Table 4: Action rules learned for NESTA (seed=2) on *hard* games. The rules are human-interpretable making them easy to debug. We highlight that the rule learned for take(x, y) is sub-optimal and can be improved by human-in-the-loop correction of that single rule with large performance gains.

We also follow the same strategy for the test games.

Table 3 shows the results for action pruning for both TWCAgent and NESTA. Firstly, even without action pruning, NESTA outperforms the TWCAgent with action pruning. NESTA+AP shows a higher gain in performance compared to NESTA only, whereas TWCAgent did not exhibit such large improvements. We found that even without AP, TWCAgent learns to avoid sub-optimal actions. However, it suffers from overfitting and hence cannot generalize to unseen configurations and entities.

4.5 Human-in-the-loop Rule Debugging

NESTA enables the user to verify all the learned rules. It provides the facility to add new rules that might be missing or edit the rules if they are sub-optimal. The ability of human-in-the-loop debugging is what sets NESTA apart from other methods that tend to provide some level of explainability. Table 4 shows the human-interpretable learned rules for a particular training on hard games. The rule for take(x, y) can be identified as sub-optimal because it implies that the agent should take any object that is present in a container y present in the current room. The human-corrected rule implies the agent should only "take" objects that are not in their assigned location according to conceptnet facts. The human-corrected rule perfectly solves the *out-of-distribution* hard games in close to the optimal number of steps. This demonstrates that NESTA's human-in-the-loop rule debugging feature can be readily used to achieve favorable performance gains.

4.6 Sample efficient learning

We hypothesize that deep RL policies require a large number of training interactions because they learn both language understanding and action scoring from rewards ignoring external language pre-training. NESTA, on the other hand, decouples language understanding to AMR-based semantic representations while the LNN-ILP rule learner can focus on RL policy optimization resulting in learning from fewer samples. Figure 3 shows that the NESTA model obtains better scores for both *in-distribution* and *out-distribution* games at much fewer training interactions compared to the deep RL text agent. In fact, NESTA can outperform text agents even when it learns from $5\times$ lesser training interactions.

We also computed computational time for NESTA compared to neural agents. Average computational times (out-of-distribution) required for each step for NESTA compared to neural agents. For easy games, the average computation time for neural agents was 0.12 ± 0.06 s, and that for NESTA was 0.16 ± 0.05. The corresponding numbers for medium games were 0.17 ± 0.06 and 0.22 ± 0.06 respectively. NESTA requires extra time due to parsing. However, since it has a lower overall number of steps (almost 5 times lower for easy/medium games from Table 2), time per game would be lower or comparable.

5 Related Work

Text-only Agents: Early work on text-based reinforcement learning agents used an LSTM-

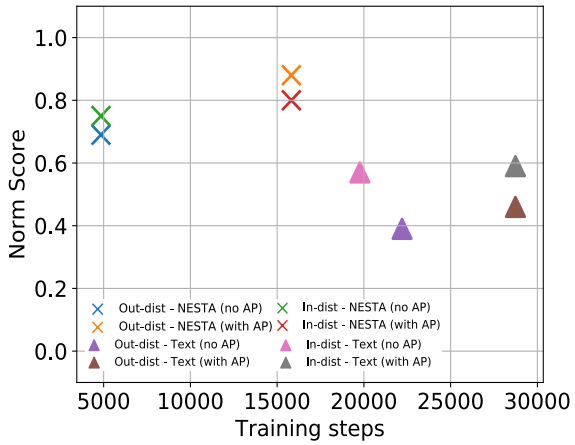

Figure 3: Normalized score obtained by NESTA and deep RL text agent on TWC games. NESTA achieves a higher score compared to the deep agent even when learning from 5x lesser training interactions. The top-left part represent better performance while the bottom-right part conveys worse performance.

based representation learning from textual observations (Narasimhan et al., 2015b), and Q-learning (Watkins and Dayan, 1992) in the action scorer of LSTM-DQN to assign probability scores to the possible actions. Yuan et al. (2018) used LSTM units in the action scorer of LSTM-DRQN to handle the better generalization. Chaudhury et al. (2020) further improved generalization and reduced overfitting by training a bootstrapped model, named CREST, on context-relevant observation text. Adolphs and Hofmann (2020) presented one of the winning strategies in the First-TextWorld Competition using the actor-critic algorithm (Mnih et al., 2016) for training the policy. Unlike these text-only models, NESTA uses symbolic reasoning over the lifted rules for better generalization and interpretability.

Graph-based Agents: Instead of relying on the neural models to capture the structure of the observed text , recent works considered the graph representation of the observed text to guide the agent for better exploration. Graph-based agents from (Ammanabrolu and Riedl, 2019; Ammanabrolu and Hausknecht, 2020) build a knowledge graph representation of the textual state for efficient exploration and handling large action space. Adhikari et al. (2020) learns a dynamic belief graph from raw observations using adversarial learning on the First Textworld Problems (FTWP). Atzeni et al. (2021) proposed a case-based reasoning approach that improves upon existing graph-based methods by reusing the past positive experiences stored in the agent's memory. Unlike NESTA, these graph-based methods suffer from noise in the observation as the graphs are generated from the observed text.

Reasoning-based Agents: Both text-only and graph-based methods use only the texts observed during the game interaction. Murugesan et al. (2020) introduced Textworld commonsense (TWC), text-based cleanup games that require commonsense reasoning-based knowledge about everyday household objects Recent works tried to enrich text-only agents with commonsense reasoning for exploiting readily-available external knowledge graphs (Murugesan et al., 2021) and images generated from the observed texts using pre-trained models (Murugesan et al., 2022). These methods suffer from noisy features extracted from the external knowledge thus hindering the learning ability of the text-based RL agents. Unlike the traditional deep RL agents, Chaudhury et al. (2021); Kimura et al. (2021); Basu et al. (2021) proposed neuro-symbolic agents for TBGs that show near-perfect performance. Related work from Li et al. (2021) uses the world model as a symbolic representation to capture the current state of the game. These approaches require hand-engineering of domain-specific symbolic state representation. On the other hand, NESTA presents a generic domain-independent symbolic logic representation with an automatic symbolic rule learner that handles large action spaces and noisy observation with ease.

In other symbolic methods, there are works (Petersen et al.; Costa et al., 2020) which employ deep learning for neuro-symbolic regression. Compared to these methods, NESTA aims to improve the generalization to unseen cases, whereas these methods train and test in the same setting. Additionally, neuro-symbolic regression methods have limited interaction with the environment in intermediate steps, and reward is obtained at the terminal state. However, for NESTA we use the symbolic representation from intermediate steps to learn action rules from partially-observable symbolic states.

6 Conclusion

In this paper, we present NESTA, a neuro-symbolic policy learning method that modularizes language understanding using an AMR-based semantic parsing module and RL policy optimization using an ILP rule learner. NESTA benefits from prior advances in AMR-based generic parsers for symbolic

fact extraction allowing the ILP symbolic learner to solely learn interpretable action rules. NESTA outperforms SOTA models on TBGs by showing better generalization while learning from a fewer number of training interactions. We believe our model is one of the first works combining advances in neural semantic parsing and efficient symbolic planning for text-based RL. We hope this work will encourage future research in this direction.

7 Limitations

The neuro-symbolic rule learning presented in the paper can handle most generic text-based games. Only in a few specific use cases, additional training of the AMR parser would be required. Since AMR is used for symbolic representation for text-based games, the vocabulary of the extracted triples is limited by the vocabulary of PropBank semantic roles. For applications in a very specific kind of domain where the predicates and entities do not match with this pre-defined vocabulary (for example, specific financial, legal domains, etc.), the AMR semantic parsing engine needs to be retrained first on such specific data before using it for rule learning. However, even in the cases where the testing environment requires additional rules, NESTA allows human-in-the-loop debugging to conveniently add them making it adaptable to generic environments.

8 Ethics Statement

Our method uses a constrained set of action samples to generate the textual actions in each step. Since this action set is generated from a controlled vocabulary of actions and entities, the produced actions cannot contain harmful content like hate speech and racial biases. Furthermore, our neuro-symbolic model produces human interpretable rules for the action policy thereby making the model transparent and easier to control. Due to these reasons, the ethical risk from this work is low.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems*, volume 33, pages 3045–3057. Curran Associates, Inc.

Leonard Adolphs and Thomas Hofmann. 2020. Ledeepchef deep reinforcement learning agent for families of text-based games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7342–7349.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Mattia Atzeni, Shehzaad Zuzar Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. 2021. Case-based reasoning for better generalization in textual reinforcement learning. In *International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Kinjal Basu, Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Tim Klinger, Murray Campbell, Mrinmaya Sachan, and Gopal Gupta. 2021. A hybrid neuro-symbolic approach for text-based games using inductive logic programming. In *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Subhajit Chaudhury, Daiki Kimura, Kartik Talamadupula, Michiaki Tatsubori, Asim Munawar, and Ryuki Tachibana. 2020. Bootstrapped Q-learning with context relevant observation pruning to generalize in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3002–3008.

Subhajit Chaudhury, Prithviraj Sen, Masaki Ono, Daiki Kimura, Michiaki Tatsubori, and Asim Munawar. 2021. Neuro-symbolic approaches for text-based policy learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3073–3078.

Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. 2020. Fast neural models for symbolic regression at scale. *arXiv preprint arXiv:2007.10784*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. *arXiv preprint arXiv:1904.11694*.

Boris Galitsky. 2020. Employing abstract meaning representation to lay the last-mile toward reading comprehension. In *Artificial Intelligence for Customer Relationship Management*, pages 57–86. Springer.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630.

Zhengyao Jiang and Shan Luo. 2019. Neural logic reinforcement learning. In *International conference on machine learning*, pages 3110–3119. PMLR.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue-Nkoutche, et al. 2021. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894.

Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander Gray. 2021. Neuro-symbolic reinforcement learning with first-order logic. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3505–3511.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Young-Suk Lee, Ramón Astudillo, Hoang Thanh Lam, Tahira Naseem, Radu Florian, and Salim Roukos. 2022. Maximum Bayes Smatch ensemble distillation for AMR parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5379–5392. Association for Computational Linguistics.

Belinda Z Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827.

Elisabeth Lien and Milen Kouylekov. 2015. Semantic parsing for textual entailment. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 40–49.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. In *NAACL-HLT*.

Zhihao Ma, Yuzheng Zhuang, Paul Weng, Hankz Hankui Zhuo, Dong Li, Wulong Liu, and Jianye Hao. 2021. Learning symbolic rules for interpretable deep reinforcement learning. *arXiv preprint arXiv:2103.08228*.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2020. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Efficient text-based reinforcement learning by jointly leveraging state and commonsense graph representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725.

Keerthiram Murugesan, Subhajit Chaudhury, and Kartik Talamadupula. 2022. Eye of the beholder: Improved relation generalization for text-based reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11094–11102.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015a. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015b. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Brenden K Petersen, Mikel Landajuela Larma, Terrell N Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. 2020. Logical neural networks. *arXiv preprint arXiv:2006.13155*.

Prithviraj Sen, Breno WSR de Carvalho, Ryan Riegel, and Alexander Gray. 2022. Neuro-symbolic inductive logic programming with logical neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8212–8219.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Rik Van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2019. Counting to explore and generalize in text-based games.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. Amr parsing with action-pointer transformer. *arXiv preprint arXiv:2104.14674*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7 is the limitations section

☑ A2. Did you discuss any potential risks of your work?
Section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☑ A4. Have you used AI writing assistants when working on this paper?
Grammar check assistant that corrected spellings on the paper

B ☑ Did you use or create scientific artifacts?

Section 2.1, Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 2.1, Section 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We will discuss the details of the license in the final code distribution

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 2.1, Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Partially in Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.