TaxoCritic: Exploring Credit Assignment in Taxonomy Induction with Multi-Critic Reinforcement Learning

Injy Sarhan[1,2], Bendegúz Toth[2], Pablo Mosteiro[2], Shihan Wang[2]
[1]Shell Global Solutions International B.V., Amsterdam, The Netherlands
[2]Utrecht University, Utrecht, The Netherlands
injy.sarhan@shell.com, toth.bendeguz@outlook.com, {p.mosteiro, s.wang2}@uu.nl

Abstract

Taxonomies can serve as a vital foundation for several downstream tasks such as information retrieval and question answering, yet manual construction limits coverage and full potential. Automatic taxonomy induction, particularly using deep Reinforcement Learning (RL), is underexplored in Natural Language Processing (NLP). To address this gap, we present TaxoCritic, a novel approach that leverages deep multi-critic RL agents for taxonomy induction while incorporating credit assignment mechanisms. Our system uniquely assesses different sub-actions within the induction process, providing a granular analysis that aids in the precise attribution of credit and blame. We evaluate the effectiveness of multi-critic algorithms in experiments regarding both accuracy and robustness performance in edge identification. By providing a detailed comparison with state-of-the-art models and highlighting the strengths and limitations of our method, we aim to contribute to the ongoing development of automatic taxonomy induction while exploring the usage of deep RL techniques in this field.

Keywords: Taxonomy Induction, Reinforcement Learning, Credit Assignment, Actor-Critic

1. Introduction

A domain's taxonomy categorizes concepts based on "is-a" relationships (Brachman, 1983), forming acyclic graphs. Nodes represent terms, and directed edges signify relationships. In this context, *"P is-a Q"* implies that term *P* (hyponym) is a subclass or more specific instance of term *Q* (hypernym). Taxonomies provide a hierarchical organization of concepts that enables more efficient data categorization and retrieval. Recent advances aim to automatically create faceted taxonomies to support more nuanced classifications and facilitate easier search and navigation within linked data applications (Zong et al., 2017). Additionally, several NLP methods utilize term taxonomies to support knowledge-rich applications, such as information extraction (Demeester et al., 2016) and question answering (Harabagiu et al., 2003), demonstrating the importance of structured knowledge that is embodied in taxonomies. Integrating taxonomies into linked datasets can significantly enhance interoperability and semantic depth, contributing to improved understanding, reasoning, and performance on complex NLP tasks.

Manual taxonomy construction is a resource-intensive and time-consuming task that requires domain knowledge. There have been efforts to handcraft large taxonomies, such as WordNet (Miller, 1995), yet ensuring comprehensive coverage remains a challenge. Automatically constructing a high-quality taxonomy is non-trivial. The goal is to infer a taxonomy graph from a set of background resources. This involves two subtasks (Wang et al., 2017): **(a) Hierarchy detection:** Identifying *"is-a"* relations between terms. Various combinations of candidate words are tested with the aid of a background corpus to uncover domain-specific relations. **(b) Hierarchy construction:** Organizing extracted pairs from (a) in a tree-like structure presents challenges, including representing transitive relations[1], and ensuring the taxonomy remains an acyclic graph with a single root node, to which all other nodes can trace a path.

The asymmetrical nature of the hypernym relation leads to two possibilities: (1) The parent node (hypernym) exists, enabling the addition of its pair as a child node (hyponym); (2) The child node is already in the taxonomy, requiring the addition of its parent, which is more complex due to the taxonomy's graph structure. Since the taxonomy is a tree with a single *root*, all nodes inherently have a parent, making it non-trivial to add a new parent node for an arbitrary child node. Consequently, most methods allow the insertion of a child node into an existing parent, and not the reverse.

Unlike conventional approaches, deep Reinforcement Learning (RL) allows for simultaneous optimization of both hierarchy detection and organization tasks, minimizing error propagation (Mao et al., 2018). Despite its potential, deep RL's application in taxonomy induction is limited (Mao et al., 2018; Han et al., 2021). In taxonomy induction, an RL action involves selecting a term (child node) from the remaining set and adding it to another term (parent node) in the taxonomy. Previous work unified these

[1] A transitive relation is defined as: if a "is-a" b and b "is-a" c then also a "is-a" c. Entity ambiguity complicates these relations in automated taxonomies (Liang et al., 2017).

DLnLD: Deep Learning and Linked Data @LREC-COLING-2024, pages 14–30

actions (Mao et al., 2018; Han et al., 2021). We posit that both components (chosen child and parent nodes) must be correct for meaningful learning of a single action. Actions, rather than nodes, are deemed correct or incorrect as a whole. However, in certain cases, one of the sub-actions[2] might be contextually accurate for the taxonomy being constructed. This leads us to the problem of credit assignment which involves identifying the cause of a certain outcome (Minsky, 1961). Proper credit assignment is crucial for pinpointing the component in the action that originates the error. Without it, the model's learning process and performance can be hindered. In this paper, we delve into the crucial aspect of credit assignment, and explore how credit assignment along with multi-critic can better attribute blame to specific sub-actions.

We introduce TaxoCritic, a novel deep RL method for automatic taxonomy induction[3]. Our goal is to enhance this task using multi-critic RL, emphasizing improved credit assignment. Our contributions are: 1) Introduce a novel RL formalization that considers parent and child nodes of the action in taxonomy induction simultaneously, in contrast to prior methods. 2) Conduct a thorough experimental evaluation of credit assignment in taxonomy induction. 3) Propose a multi-critic approach to highlight the effectiveness of credit assignment in taxonomy induction, leading to improved robustness.

The paper is structured as follows. In Section 2 we present an overview of previous work upon which we build. Section 3 describes our methodology. Section 4 describes our dataset and presents the results of our experiments. We draw our conclusions in Section 5, and discuss our limitations in Section 6.

2. Related Work

Taxonomy induction methods can broadly be categorized into traditional approaches and RL-based techniques. In this section, we briefly overview traditional approaches before focusing on advances using RL. Traditional methods for hierarchical detection are pattern-based (Hearst, 1992), offering high precision but low recall, or statistical, using background text statistics for identifying relations without manual syntax specification. For example, Fu et al. (2014) uses the spatial properties of embeddings like GloVe (Pennington et al., 2014) or Word2vec (Mikolov et al., 2013) to detect hypernym-hyponym pairs. For more information on traditional methods, please refer to Page 49 of Weikum et al. (2021).

Limited research exists on RL in taxonomy induction. Mao et al. (2018) argue that the two-phase taxonomy-induction setup, i.e. hierarchy detection and construction, is inherently suboptimal due to one-directional information flow. Their system, TaxoRL, unifies both phases, training a REINFORCE (Williams, 1992) agent to select and append a child node to a pre-existing parent in the taxonomy, which is also chosen by the agent. DTaxa (Han et al., 2021) builds on TaxoRL with an actor-critic approach, using a variant of the DDPG agent instead of REINFORCE, for faster learning and better performance. TaxoRL and DTaxa achieve competitive performance on taxonomy induction benchmarks.

Both TaxoRL and DTaxa face a common drawback in their action representations. They treat the selection of a term and its position in the taxonomy as a single action, missing the ability to discern different types of errors. For example, choosing a child node without a parent in the tree could lead to multiple incorrect parent choices. We argue that adjusting action handling to better align with the problem's structure and semantics could enhance taxonomy induction.

3. Methodology

We formulate taxonomy induction as an RL problem. The goal is to create a taxonomy that accurately organizes a given set of terms, aligning with the golden taxonomy. To achieve this, the model is provided with a large background corpus, allowing it to incorporate information about the relations of words as features in its action representation.

3.1. Problem formulation

Taxonomy induction is formulated as a finite and discrete Markov Decision Process (MDP) (Bellman, 1957). At the beginning of every episode, we start with a taxonomy tree containing only a single word (also known as a term). We expand the tree at each time step by appending a word from the term set as a child to one of the nodes in the tree until all the terms are added (i.e. the end of an episode). At each time step t, there is a set of words that are nodes in the taxonomy tree U_t, a set of remaining terms that are not yet part of the tree V_t and a set of edges E_t, each of which connects two nodes in the taxonomy: $E_t \subseteq \{(U_t \times U_t)\}$. Furthermore, a root node $\text{ROOT}_t \in U_t$ serves as the root of the taxonomy tree. To give a concrete example of the notion of a state, we refer to Figure 1. We follow the standard definition in Sutton and Barto (2018)

[2] We refer to the choice of either one of the two terms as a "sub-action", while the complete action refers to the choice of both the child and parent terms.

[3] Our implementation of TaxoCritic and the Appendix file are publicly available at `https://github.com/BendeguzToth/taxonomy-construction`.

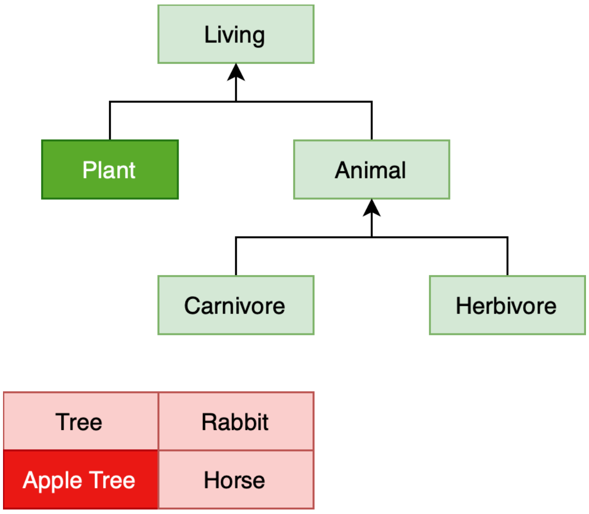

Figure 1: Example of an action a_t = *(Apple Tree, Plant)*, where $root_t$ = {*Living*}, U_t = {*Living, Plant, Animal, Carnivore, Herbivore*}, V_t = {*Tree, Rabbit, Apple Tree, Horse*}, and E_t = {*(Plant, Living), (Animal, Living), (Carnivore, Animal), (Herbivore, Animal)*}. After the execution of this action, $root_{t+1}$ = {*Living*}, U_{t+1} = {*Living, Plant, Animal, Carnivore, Herbivore, Apple Tree*}, V_{t+1} = {*Tree, Rabbit, Horse*}, and E_{t+1} = {*(Plant, Living), (Animal, Living), (Carnivore, Animal), (Herbivore, Animal), (Apple Tree, Plant)*}.

and define the key elements as follows:

State The MDP contains a set of observed *states* S. The state $s_t \in S$ at any time step t represents the taxonomy at time t, consisting of a collection of edges E_t, as well as the remaining term set V_t. Notably, there is no need to explicitly include the terms that are already part of the tree (the nodes) as they are implicitly represented by the edges. The state is formally denoted as $s_t = (E_t, V_t)$.

Action There is a set of *actions* A. An action a_t ($a_t \in A$) can fall into one of two types:

- **Adding a new node as a child** In this case, the action a_t takes the form $(v, u) \in (V_t \times U_t)$. The new term v is added to the taxonomy as a child node to u. The update to the taxonomy at time step $t+1$ is as follows:

$$U_{t+1} = U_t \cup \{v\}, \quad V_{t+1} = V_t \setminus \{v\}$$
$$E_{t+1} = E_t \cup \{(v, u)\}, \quad \text{ROOT}_{t+1} = \text{ROOT}_t$$

- **Adding a new node as root** Alternatively, the current root is the child, and a new term is appended as its parent (resulting in the new term becoming the new root). This action a_t is represented as (ROOT_t, v) where $v \in V_t$. The common updates to set U and V at time step $t+1$ are the same as in the previous action. The specific updates to E and ROOT are:

$$E_{t+1} = E_t \cup \{(\text{ROOT}_t, v)\}, \quad \text{ROOT}_{t+1} = v$$

By combining those two action possibilities, an action takes the following form:

$$a_t \in (V_t \times U_t) \cup (\{\text{ROOT}_t\} \times V_t) \qquad (1)$$

Transition The transition from one state to another is deterministic, i.e. $Pr(s_{t+1}|s_t, a_t) = 1$. Thus, following the two action possibilities, the next state is determined by the updated taxonomy:

$$s_{t+1} = (E_{t+1}, V_{t+1})$$

Reward Similar to (Mao et al., 2018) and (Han et al., 2021), we utilize the difference in *Edge-F1* at each time step as the deterministic reward signal. *Edge-F1* is defined in Equation 2, where E^* is the set of edges present in the golden taxonomy and $\overline{E}$ is the set of edges predicted by the model. The reward at time step t is then $F_1^t - F_1^{t-1}$.

$$P = \frac{|\overline{E} \cap E^*|}{|\overline{E}|}, \quad R = \frac{|\overline{E} \cap E^*|}{|E^*|}$$
$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \qquad (2)$$

3.2. Design Architecture

To address the issue of proper credit assignment in previous methods, we propose TaxoCritic, a single-actor and multi-critic RL algorithm that individually evaluates both sub-actions. This approach allows for assigning rewards (either positive or negative) to the two sub-actions independently, leading to better credit assignments. Inspired by existing multi-critic RL techniques (Martinez-Piazuelo et al., 2020; Mysore et al., 2021), we integrate the idea of multiple critics into the domain of taxonomy induction for the first time. Instead of relying on a single critic to estimate the value of an action, our algorithm incorporates two distinct critics, each dedicated to one of the sub-actions and their outputs are combined to produce the final estimate. More precisely, one of the critics assesses the choice of the parent node, while the other evaluates the choice of the child node. This design allows the sub-critics to be independent and simplifies model optimization by backpropagating only once from the combined action value. The actor, on the other hand, remains undivided and determines the best joint action to take. An overview of the TaxoCiritc framework is illustrated in Figure 2.

3.2.1. Actor

In our method, the actor is a fully connected 2-layer feed-forward neural network. The design of the actor architecture poses a unique challenge due to

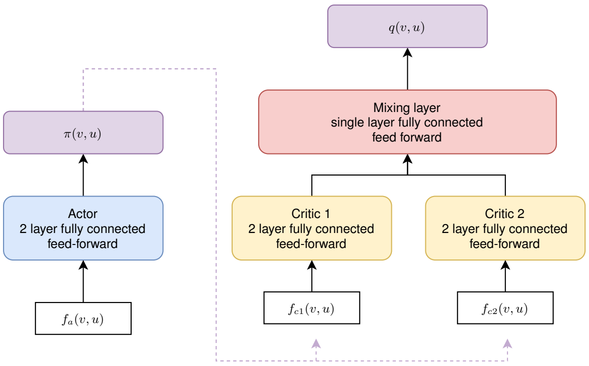

Figure 2: An overview of the TaxoCritic method. f_a, f_{c_1} and f_{c_2} represent the feature representations (vectors) of the inputs for the actor and two critics respectively. The **actor** (a two-layer fully connected feed-forward neural network) takes the encoding of a state as the input and outputs the policy π. Following this policy, the environment executes a sampled action which contains two sub-actions. As shown on the right side, the **critic** network features two sub-critics and a mixing layer. Considering the state and two sub-actions, one critic evaluates the child's choice, while the other evaluates the parent's choice. The mixing layer combines those results from both critics and produces the action value q.

the dynamic nature of our action space. Unlike agents trained for tasks like playing Atari games or controlling robotic arms, where the number of actions remains constant throughout the task (Mnih et al., 2015; Franceschetti et al., 2021), taxonomy induction demands a more flexible approach. As described in Section 3.1, the actions are defined by the number of terms left to be added to the tree, as well as the current nodes that are present in the taxonomy. These quantities are dynamic and change at each time step, making it impractical to adopt a standard architecture where the neural network takes only the state representation as input and outputs a probability distribution over a fixed number of possible actions. Thus, in our method, the policy network takes the features of a possible state-action pair (s_t, a_t) as the input, generates the probability of taking that specific action in the given state (i.e. $Pr(a_t|s_t)$). This design allows accommodating an arbitrary number of actions. During the construction of a taxonomy, all possible action pairs at the current state are fed through the network, outputting a probability distribution over the valid action space through a Softmax function.

Moreover, a challenge arises from the variability in action semantics across different episodes. For instance, when constructing a taxonomy for the animal kingdom, and subsequently another one for different kinds of mining equipment, all the actions would have entirely different semantic meanings, despite the action space size remaining constant. In other words, the action corresponding to the first output value will probably have an entirely different interpretation in taxonomy a than in taxonomy b. Thus, it is crucial to explicitly encode the actions themselves, as relying on constant positions is no longer sufficient. By using a network that incorporates action embeddings, the semantic meaning of each action can be communicated in the TaxoCritic.

3.2.2. Critic

In taxonomy induction, an action is expressed as an edge (v, u) to be added to the taxonomy graph. An action can be split naturally into two sub-actions. One component of the action, v, denotes the new term that shall be added to the taxonomy as a child node, while u denotes the parent node to connect the child node. This clear division between the two parts of the action allows us to employ distinct critics for assessing each sub-action independently. Therefore, in our model, the critic is divided into two distinct sub-critics. Each of the two sub-critics can only observe a part of the feature space (depending on which part of the action they focus on) and, as such, are responsible for rating different components of the action. The outcomes of these sub-critics are then merged with a single feed-forward layer neural network to obtain the final q value estimate.

3.3. Feature Representation

In taxonomy induction, both states and actions are tuples of words. To capture the semantic features of the words, we use the embedding of state and action as inputs of neural networks. We generate the feature representations as in TaxoRL (Mao et al., 2018), adapted by DTaxa (Han et al., 2021). A syntax-level feature vector is constructed for every possible action, consisting of eight features: Capitalization, Endswith, Contains, LCS, LD, Normalized frequency difference, and Generality difference; see Appendix A for detailed information. The final vector is the concatenation of the embeddings for the vectors v and u corresponding to the action, the dependency path, and the syntax-level feature vector. Figure 3 depicts an overview of a feature vector for action $a_t = (v, u)$.

As previously outlined in Section 3.1, a state of the MDP has the form $s_t = (E, V)$. However, incorporating the remaining term set V as part of the state feature is redundant since the action encoding already encapsulates this data. Therefore, we simplify the state representation to only E. The edges of a taxonomy at a given time t correspond to the taken actions since each action effectively

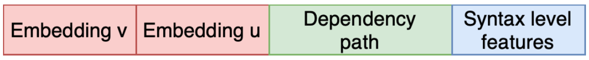

Figure 3: The action feature vector $a_t = (v, u)$ concatenates the word embeddings —using GloVe (Pennington et al., 2014)— for terms v and u, their dependency path from the corpus, and syntactic features into one vector.

adds a new edge to the tree. The state s_t is represented as the sequence of actions taken up to t:

$$s_t = (a_1, a_t, ..., a_{t-1}) \tag{3}$$

To represent this, all action embeddings $(a_1, a_2, ..., a_{t-1})$ are input to a single-layer Long short-term memory (LSTM) network that combines those values into one vector. As the model is fully differentiable, backpropagation into the LSTM parameters is straightforward.

For all the possible actions at time step t, the **actor** takes the feature representations of each pair of state and action and concatenates them as one feature vector input. On the other side, both **sub-critics** utilize different feature vectors based on the state and action representations. These two vectors are similar, essentially mirroring each other. The rationale behind this design is to ensure that when evaluating one sub-action, no assumptions are to be made about the other part of the action. This leads to two changes in feature representation compared to the one employed by the actor:

Word embeddings: Each sub-critic includes the word vector for only the term it evaluates. Specifically, one sub-critic uses the embedding of v, and the other uses the embedding of u.

Relational features: The dependency path and syntax level features can no longer be used, as they rely on knowing both words of the action. Instead of leaving those features out, they are modified in a way that requires knowledge of only one term. This is achieved by summarizing the relations between the known word and all its potential pairs. For example, if the chosen action by the policy network is the tuple (v_i, u_j), then the critic responsible for the choice of the child node would take the relations of v_i with every possible u and average them to obtain an approximation. This averaged feature is called the *average shared feature* of critic 1. The average shared feature of critic 2 is constructed in a similar way, except the choice of u_j is known, and the average is taken over all possible choices of v. A comprehensive formal definition of the shared features for the sub-critics is provided in Appendix B.

3.4. Training

The training of our model is done simultaneously by training the two sub-critic networks and the policy network. Both critics are trained jointly, with the gradient being distributed by the mixing function[4]. The loss is computed using the output of the mixing function, which aggregates the output values of both sub-critics. We refer to the entire value network (both sub-critics and the mixing layer) as *combined critic*. A comprehensive outline of the joint training algorithm is in Appendix C. All the experiments were run on a Linux virtual machine powered by an Intel Xeon Platinum CPU with 2 cores and 32 GB RAM. With this setup, running a single epoch took 50-60 minutes on average. Running the full experiment up to 300 epochs took over 11 days.

4. Experimental Results

To evaluate our model's performance and compare it with previous methods, we conducted a series of experiments. Our goal is to gain insights into the characteristics, strengths, and weaknesses of the algorithms by not only examining the final performance metrics but also by conducting qualitative analyses of the resulting taxonomies. We conducted three analyses:

1. **Ablation analysis.** We conducted an ablation analysis to evaluate specific features in our model, focusing on their individual contributions to overall performance.
2. **Performance assessment.** In this experiment, we evaluated the performance of our final model, as well as two of the baseline models —TaxoRL and DTaxa—, on a set of taxonomy induction tasks. We compared their results based on two evaluation metrics, examining the accuracy (i.e. edge F1 score) in individual runs as well as the robustness (i.e. consistency score) across different runs.
3. **Credit assignment analysis.** One of the motivations for choosing a multi-critic approach for our model was to improve credit assignment in the critic. In this qualitative analysis, we showcase how our critics have effectively learned the behavior patterns we outlined above.

4.1. Experimental Setup

Experimental Environment In our experiments, the dataset was split into training, validation, and

[4] Multiple constructions of the mixing layer have been experimented with e.g. QMIX-like architecture (Rashid et al., 2020). Here we only present the selected architecture with superior performance.

test sets with a distribution of 70/15/15, corresponding to 533/114/114 taxonomies respectively. Each model was trained for 300 epochs, and the resulting weights were saved for subsequent qualitative analysis. During an epoch, a single training episode is executed for every taxonomy in the training set, amounting to 533 episodes per epoch. An episode involves constructing a single taxonomy. At the start of each episode, a set of terms is provided, and the goal is to build up a taxonomy from said terms that match the target *golden taxonomy* as closely as possible. The agent's action interface only allows for the extension of an already existing taxonomy, requiring at least one node (*root*) to begin construction. To address this, like TaxoRL, we chose to start each episode with a randomly selected root node. The agent can then attach a new root node on top of it by selecting the node to be added as a parent and the current node as a child. This approach is intended to improve the model's robustness by reducing the potential for overfitting to a specific construction sequence for each taxonomy as it aims to allow the model to adapt to various starting points. All results are averaged over multiple turns.

Hyperparameters We explored various layer sizes and learning rates for both the actor and the critic networks. A similar trend was observed in the results, we therefore selected the optimal ones for the following results. Both the critic networks and the actor network consist of a single-neuron two-layer and a multi-neuron first layer (with 64 and 60 hidden nodes respectively). We set a learning rate of 5×10^{-4} for the actor and 1×10^{-4} for the critic. Moreover, we employed a ReLU activation function and the Adam optimizer (Kingma and Ba, 2014) for training. The discount factor is set to 0.95. For more information about parameter optimization, please refer to Appendix D.

4.2. Dataset

We used the WordNet taxonomy (Bansal et al., 2014), also utilized by TaxoRL and DTaxa. It encompasses a set of 761 taxonomies sampled from WordNet (Miller, 1995), each with a depth of three, built up from 10-50 nodes. While this dataset provides word sets and their corresponding target taxonomies, it does not specify the underlying background corpus. The agent's performance on the benchmark is heavily influenced by this background corpus, which is essential for extracting statistical relations among terms forming a crucial aspect of the feature representation during training. To ensure meaningful comparisons with prior methodologies, we opted to utilize the same background text as TaxoRL, which is an aggregation of Wikipedia dump, the UMBC web-based corpus (Han et al., 2013) and the One Billion Language Modelling Benchmark (Chelba et al., 2013).

4.3. Evaluation Metrics

Edge-F1 score: Similar to Mao et al. (2018), we first evaluate the Edge-F1 score. At the end of the episode, once all terms are incorporated into the taxonomy tree, the final construction is evaluated against the gold taxonomy. A detailed explanation of the edge score follows:

- Edge set in the constructed taxonomy: E_{pred}
- Edge set in the gold taxonomy: E_{gold}
- Edge precision: $P_e = |E_{pred} \cap E_{gold}|/|E_{pred}|$
- Edge recall: $R_e = |E_{pred} \cap E_{gold}|/|E_{gold}|$
- Edge-F1: $F1_e = 2 \cdot P_e \cdot R_e/(P_e + R_e)$

Consistency scores: To assess the model's robustness we introduce a consistency score, denoted as C_{root}, which measures the model's ability to converge consistently across different runs. It is calculated as a ratio of the number of consistent convergences where the model consistently identified the correct root $R_{\text{consistent}}$ to the total number of experimental runs R_{total}.

$$C_{\text{root}} = \frac{R_{\text{consistent}}}{R_{\text{total}}} \tag{4}$$

In addition, we also introduce a 'Consistency in Edge Score' C_{edge}, as the ratio of the number of correct edges identified across all runs to the total number of edges in the experiment E_{total}.

$$C_{\text{edge}} = \frac{\sum_{i=1}^{totalRun} |E_{pred_i} \cap E_{gold}|}{E_{total}} \tag{5}$$

4.4. Results

Experiment #1 Ablation Analysis In this experiment we assess the impact of two features: sibling embeddings and history inclusion. Sibling nodes m and n share the same parent node, and their embeddings are averaged and added to the feature vector of action (v, u) if sibling embeddings are employed. The history feature encompasses a summary of past actions in the feature vector.

Surprisingly, omitting the history representation led to enhanced performance. We observed a consistent pattern when testing using TaxoRL algorithm. Making use of sibling embeddings, on the other hand, positively impacts the performance. Based on this analysis, we decided to leave out the history representation from both our model and

TaxoRL for the main experiment [5]. The results of this analysis can be seen in Table 1.

History Usage	Sibling Usage	F1 after 150 epochs	F1 after 200 epochs
No	No	0.3233	0.3328
No	**Yes**	**0.3301**	**0.3434**
Yes	No	0.1649	0.1724
Yes	Yes	0.2506	0.2596

Table 1: The result table showcases the Edge-F1 score of TaxoCritic model when certain features are omitted.

Experiment #2 Accuracy Performance We trained our model, TaxoRL, and DTaxa* [6] on the dataset described in Section 4.2. We average the results of each algorithm over three runs. The training results are illustrated in Figure 4. Despite its initial slow start, our method eventually outperforms TaxoRL in the experiment. The slower convergence at the beginning shows a characteristic difference between the two algorithms.TaxoRL trains its policy network based on sampled returns at each transition, yielding a noisy but unbiased return estimate. In contrast, our agent updates its policy network using the critic's output, which begins as random due to the critic's initial random initialization. However, once the critic's estimates stabilize, our policy's convergence accelerates and surpasses TaxoRL's speed. The graph further illustrates that DTaxa* significantly outperforms both other methods. A potential reason for DTaxa*'s better performance is attributed to the use of an efficient actor-critic algorithm DDPG. Table 2 shows the results of all methods after a specific number of epochs. For additional evaluation results, see Appendix E.

Experiment #3 Robustness Performance We assess our model's consistent convergence across various runs on a randomly selected taxonomy sample. We conducted five runs with five different initial root words. In TaxoCritic, the correct root was correctly identified in 3 out of 5 cases. In the remaining two instances, other terms were chosen as roots. Among the 55 total edges from the 5 runs, 31 were correctly identified. Among the incorrect edges, a pattern emerged: 11 out of the 24 erroneous edges had *guestroom* as their parent, indicating a systematic bias in the model. This bias is more manageable for practical use, as domain experts can focus on potentially flawed parts of the final taxonomy. This is especially beneficial for larger and more intricate trees. While we demonstrate this with a smaller example for clarity, similar principles can apply to more complex domains. The selected taxonomy sample and all the resulting structures are depicted in Appendix F.

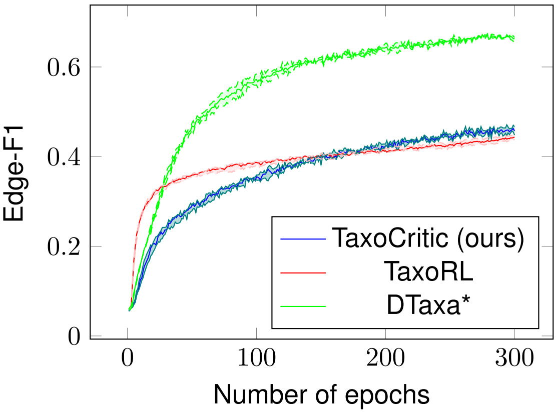

Figure 4: The training performance comparison graph of TaxoCritic (ours), TaxoRL, and DTaxa. The central darker line represents the average performance, while the lighter lines above and below indicate the range of minimum and maximum values across the runs.

Model	Epochs		
	100	**200**	**300**
TaxoRL	0.386	0.413	0.443
DTaxa*	**0.571**	**0.643**	**0.664**
TaxoCritic	0.349	0.421	0.459

Table 2: Edge-F1 scores on the training set performance of the algorithms at different epochs.

We repeated the experiment using the same taxonomy with the two benchmark models. DTaxa* displayed a similar overall correctness, with 29 correct edges. However, it consistently struggled to identify the correct root node across all 5 runs, exhibiting a bias towards the term *connecting room*, frequently assigning it numerous children despite it being a leaf node in the golden taxonomy, resulting in 11 incorrect edges featuring it as their parent. TaxoRL achieved the lowest overall performance by correctly identifying 26 edges. However, it consistently identified the correct root in all 5 cases. The results are summarized in Table 3. Refer to Appendix F for the generated trees by DTaxa* and TaxoRL.

[5] Note that this analysis was run with an earlier version of the model before it was fully optimized, therefore the results are slightly lower than in the final experiment.

[6] In our effort to access the code associated with the DTaxa paper, we made attempts to contact the authors, but unfortunately, we did not receive a response. To conduct our experiments, we undertook the task of recreating their model to the best of our abilities, based on the limited information provided in the paper. We refer to this recreated model as DTaxa*. The code of DTaxa* is also provided in our project repository.

Model	C_{root}	C_{edge}
TaxoRL	**1**	0.47
DTaxa*	0	0.53
TaxoCritic	0.6	**0.56**

Table 3: Robustness Scores for the three Models.

Experiment #4 Credit Assignment The multi-critic architecture was adopted to enhance convergence speed and effective credit assignment by the critic. This concept was demonstrated through a hypothetical scenario where one sub-action could be held responsible for an incorrect action (p, c) while the other sub-action could be a suitable choice. We examined our model's feasibility to exhibit this attribute by analyzing the construction of a small example taxonomy.

Figure 5 depicts the state of the tree at the specific point of interest. In this analysis, we will look at the sub-critic output values for potential actions within the partial taxonomy. With $V = \{$*nursery, day nursery, connecting room, adjoining room*$\}$, allowable actions are: Each term within V can serve as a child node, either linked to an existing taxonomy node (denoted in yellow and blue), or any term from V can function as a new node, becoming the parent of the current root node *bedroom*.

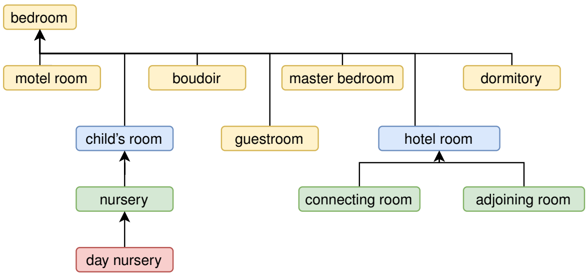

Figure 5: A simple partially constructed bedroom hierarchy. Yellow and Blue nodes (top three rows) denote correctly placed terms. Green and Red nodes (bottom two rows) are yet to be placed, while indicating their intended positions.

To analyze our approach's ability to learn credit assignment for sub-actions, we assess intermediate output values from two critics. To demonstrate the credit assignment of an individual sub-action, we focus on the average output of the child critic, responsible for estimating the value of selecting each potential node v in V as the child node.

In this example, selecting any of the blue terms as a child node is beneficial due to the presence of correct parent nodes for each of them in the tree. However, the red node *day nursery* lacks a suitable parent. We intuitively expect that the sub-critic for choosing the child node should assign lower values to actions involving *day nursery* compared to others. We therefore investigate the intermediate values assigned by the term-choice critic for each possible action. There are 8 potential parents for each child candidate term. Table 4 displays the average action values for every possible child candidate. Notably, *day nursery* has the lowest value of -5.14, while the average for green terms (with valid parents) is -4.32. This observation indicates that our critic prioritizes the green nodes as potential children, aligning with expectations. Meanwhile, since *day nursery* cannot be properly attached to any existing nodes, the critic assigns it a lower rating. For a similar analysis of the parent node selection, please refer to Appendix G, where we observe a similar trend.

This analysis demonstrates that our multi-critic algorithm is correctly assigning the blame when one sub-action is primarily responsible for the choice of incorrect action, without penalizing sub-actions that are conceptually correct but fail due to the wrong choice of the other sub-action. This property contributes to maintaining consistent action value estimates during training.

Terms	Action value
nursery	-3.72
connecting room	-4.76
adjoining room	-4.47
day nursery	-5.14

Table 4: Inverted action values show the critic's rating of each node's suitability as a child.

5. Conclusion

In this paper, we introduce TaxoCritic, a deep Reinforcement Learning-based approach for taxonomy induction that utilizes a multi-critic algorithm. Unlike previous methods treating all actions as independent, TaxoCritic divides actions into two distinct sub-parts, each assigned to its own critic. This framework enhances credit assignment by accurately attributing blame to the responsible action component in case of errors. While our approach did not surpass all baselines in learning performance, the enhanced credit assignment analysis and overall robustness performance highlight the potential of multi-critic strategies for taxonomy induction. In conclusion, we believe that our method can serve as a good foundation for further research on applying deep RL techniques to taxonomy induction in a promising direction. Further, we encourage the research community to explore integrating taxonomy induction methods into the linked data ecosystem to improve knowledge representation. This effort should ensure that the resulting structures are interoperable, semantically rich, and could be easily integrated into existing datasets.

6. Limitations

We anticipated that the critics in our methodology would easily adapt to their more constrained role and would be able to work together efficiently, and our credit assignment analysis confirmed this expectation as the critics effectively identified correct and incorrect sub-actions, a promising outcome. However, this success did not translate proportionally to overall performance. Despite outperforming TaxoRL, our model's performance was notably inferior to DTaxa*, a single-critic method. This is an unexpected outcome considering that both sub-critics performed as intended. Trying to pinpoint the reasons behind this performance deficit might be an interesting follow-up research. We also propose exploring the impact of employing an alternative mixing function to effectively merge insights from the sub-critics for the final value. In addition, although we follow the existing works to use the WordNet taxonomy for evaluation which has a depth limited to three, exploring our method's generalizability on datasets with varied depth levels would be an intriguing direction.

Ethics Statement

Our data are taken from publicly available sources. For this reason, we do not expect that there are ethical issues or conflicts of interest in our work.

7. Bibliographical References

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1041–1051.

Richard Bellman. 1957. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Ronald J. Brachman. 1983. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer*, 16(10):30–36.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1389–1399, Austin, Texas. Association for Computational Linguistics.

Andrea Franceschetti, Elisa Tosello, Nicola Castaman, and Stefano Ghidoni. 2021. Robotic arm control and task training through deep reinforcement learning. In *International Conference on Intelligent Autonomous Systems*, pages 532–550. Springer.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209.

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc_ebiquity-core: Semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 44–52.

Yongming Han, Yanwei Lang, Minjie Cheng, Zhiqiang Geng, Guofei Chen, and Tao Xia. 2021. Dtaxa: An actor–critic for automatic taxonomy induction. *Engineering Applications of Artificial Intelligence*, 106:104501.

Sanda M Harabagiu, Steven J Maiorano, and Marius A Paşca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiaqing Liang, Yi Zhang, Yanghua Xiao, Haixun Wang, Wei Wang, and Pinpin Zhu. 2017. On the transitivity of hypernym-hyponym relations in data-driven lexical taxonomies. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1185–1191. AAAI Press.

Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-end reinforcement learning for automatic taxonomy induction. *arXiv preprint arXiv:1805.04044*.

Juan Martinez-Piazuelo, Daniel E Ochoa, Nicanor Quijano, and Luis Felipe Giraldo. 2020. A multi-critic reinforcement learning method: An application to multi-tank water systems. *IEEE Access*, 8:173227–173238.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Siddharth Mysore, George Cheng, Yunqi Zhao, Kate Saenko, and Meng Wu. 2021. Multi-critic actor learning: Teaching rl policies to act with style. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2017. A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1190–1203.

Gerhard Weikum, Xin Luna Dong, Simon Razniewski, and Fabian Suchanek. 2021. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Nansu Zong, Hong-Gee Kim, and Sejin Nam. 2017. Constructing faceted taxonomy for heterogeneous entities based on object properties in linked data. *Data & Knowledge Engineering*, 112:79–93.

A. Actors's Features

In Table 5, we list a detailed description of the features utilized by the Actor.

Table 5: Features and Descriptions

Features	**Description**
Capitalization	Whether any (or both) of the words are capitalized.
Endswith	If the second word ends with the first word (for example, for the pair (bear, polar bear), this would fire.)
Contains	If the second word contains the first word.
Suffix match	The number of matching trailing letters.
LCS	The length of the longest continuous substring contained by both words.
LD	Length difference between the words. $10 * \frac{|w_1|-|w_2|}{|w_1|+|w_2|}$
Normalized frequency difference	The ratio between the frequency of pair (v, u) and the most frequent parent of v, u': $\frac{\text{freq}(v,u)}{\max_{u'} \text{freq}(v,u')}$.
Generality difference	The generality $g(v)$ of term v is the logarithm of the number of its distinct hyponyms. The generality difference of the pair (v, u) is defined as $g(u) - g(v)$.

B. Sub-critics' Features

In this appendix section, Figures 6 and 7 illustrates how the features outlined in Equation 6 are used by the two sub-critics. The shared features for the sub-critics as mentioned in Section 3.2.2 are defined as:

$$\begin{aligned}
&f(v,u) : \text{dependency path and syntax features of the term pair } (v,u)\\
&f_{c1}(v) : \text{The average shared feature of critic 1, where the child is } v.\\
&\text{The mean is taken of all feature vectors with } v \text{ as child.}\\
&f_{c2}(u) : \text{The average shared feature of critic 2, where the parent is } u.\\
&\text{The mean is taken of all feature vectors with } u \text{ as parent.}\\
&f_{c1}(v) = \frac{\sum_{u \in U} f(v,u)}{|U|}\\
&f_{c2}(u) = \frac{\sum_{v \in V} f(v,u)}{|V|}
\end{aligned} \tag{6}$$

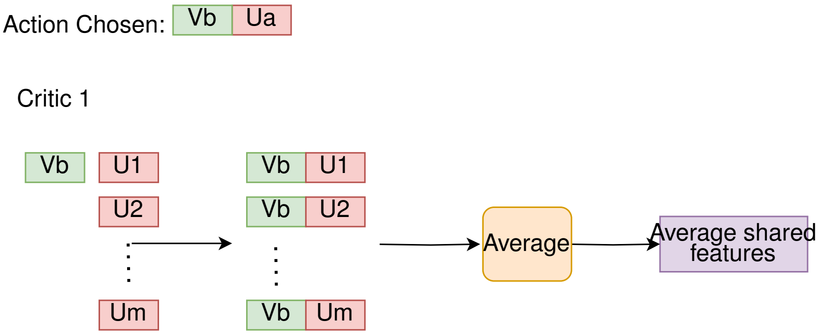

Figure 6: Shared feature summary of sub-critic 1. This sub-critic is only aware of the choice of the child term. To obtain the dependency path and syntax level features, it takes the features with all possible parent terms, then averages them.

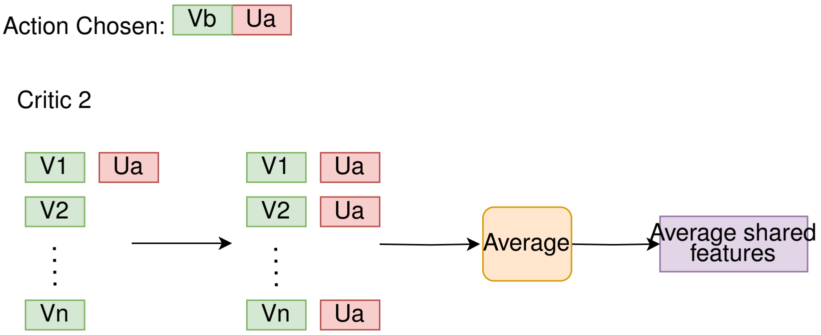

Figure 7: Shared feature summary of sub-critic 2. This sub-critic is only aware of the choice of the parent term. To obtain the dependency path and syntax level features, it takes the features with all possible child terms, then averages them.

B.1. Network architecture

The value network is built up of three distinct parts. This is illustrated in Figure 8. There is one network for both critics. Those share the same architecture, with two fully connected layers and a ReLU in between. The input vector contains a word embedding (the embedding of v in the case of critic1 and the embedding of u in the case of critic2), the appropriate average shared features, and the state representation. The input size is 140. The first fully connected layer consists of 64 neurons, while the second layer is just a single neuron. The output is interpreted as the value of the sub-action. The last part of the critic is the mixing layer. It is a simple, single-layer feed-forward neural network that takes the two sub-action values and combines them into the final action value. We experimented with different mixing functions, most notably a QMIX-like architecture (Rashid et al., 2020), but we found a simple fully-connected layer to be more performant.

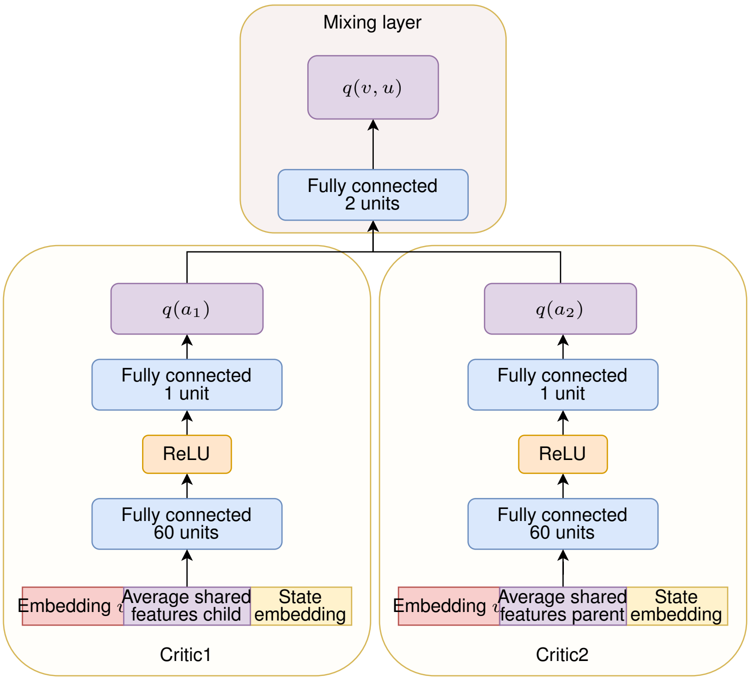

Figure 8: The architecture of the critic. $q(v, u)$ is the action-value of the action (v, u), and $q(a_i)$ is the value of sub-action i.

C. TaxoCritic Training Algorithm

Algorithm 1 describes the joint training process of the agent in detail. The two critics are trained jointly, with the gradient being distributed by the mixing function. The loss is calculated based on the output of the mixing function, that combines the output values of both sub-critics. In the pseudo-code below we refer to the entire value network (both sub-critics and the mixing layer) as *combined critic*.

```
DEFINITIONS;
$D$ : Training dataset;
$\alpha_c, \alpha_a$ : Critic and actor learning rate;
$\gamma$ : Rewards discount rate;
$\tau$ : Target update rate;
$\mu$ : Actor parameters;
$\theta, \theta'$ : Combined critic and combined critic target parameters;
$\pi, q$ : Policy and value functions;
$s, r$ : State and reward representations;
buff: Replay buffer;
INITIALIZATION;
buff $\leftarrow \emptyset$;
Initialize $\theta, \mu$ randomly;
$\theta' \leftarrow \theta$;
for $(V, U, E) \in D$;                          // For each taxonomy in the training set.
do
    while $|V| > 0$;                            // Repeat until the remaining term set is empty
    do
        $A = (V \times U) \cup (\{\text{ROOT}\} \times V)$;          // 'A' is the set of all actions.
        LP $\leftarrow \{\pi_\mu(s, a)$ for all $a \in A\}$;   // 'LP' is the vector of log probabilities of all
         actions.
        $(v, u) \leftarrow$ sample(Softmax(LP));                      // Sample action
        $V \leftarrow V \setminus \{v\}$;           // Update the taxonomy with the selected action
        $U \leftarrow U \cup \{v\}$;
        $E \leftarrow E \cup \{(v, u)\}$;
        buff.add$(s, (v, u), r, s')$;     // Add (state, action, reward, next state) to buffer
    end
    ;                                        // After the episode ends train on all transitions.
    for $(s, a, r, s') \in$ buff;                        // For each transition in buffer
    do
        target $= r + \gamma q_{\theta'}(s', a)$;               // Calculate the critic target
        $L_c = (\text{target} - q_\theta(s, a))^2$;                   // Combined critic loss
        $L_a = \ln(\pi_\mu(s, a)) \cdot q_\theta(s, a)$;                        // Actor loss
        $\theta \leftarrow \theta + \alpha_c * \nabla_\theta L_c$;                // Updating the parameters
        $\mu \leftarrow \mu + \alpha_a * \nabla_\mu L_a$;
    end
    $\theta' \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta'$;                   // Updating target params
    buff $\leftarrow \emptyset$;
end
```

Algorithm 1: Joint training of the actor and the combined critic

D. Hyperparameters

D.1. Learning Rate

Specifically, we conducted experiments to determine the optimal learning rate pairs for both the actor and critic networks. The outcomes of this analysis are detailed in Table 6.

D.2. Path LSTM Dimensions

The path LSTM is an important part of the model, as it is responsible for summarizing the information about the relation of two words into a fixed-size representation.The dimension of this LSTM layer significantly affects the final performance. To determine the optimal dimensionality, we conducted an experiment, and the results are presented in Table 7.

Learning Rate		Edge-F1
Actor	Critic	(150 epochs)
1×10^{-4}	1×10^{-4}	0.2816
5×10^{-4}	1×10^{-4}	**0.3301**
1×10^{-3}	1×10^{-4}	0.2041

Table 6: Results of learning rate analysis for the TaxoCritic model

Path LSTM Dimension	Edge-F1 at 150 epochs	Edge-F1 at 200 epochs
60	0.3301	0.3434
128	0.3353	0.3354
256	0.3208	0.3303

Table 7: Performance analysis of the TaxoCritic model with different path LSTM dimensions.

E. Additional Experimental Results

Results on the evaluation set are reported in Table 8, while the edge training results are reported in Table 9.

Model	Epochs				
	100	150	200	250	300
TaxoRL	0.063	**0.066**	**0.069**	**0.068**	**0.065**
DTaxa*	**0.065**	0.063	0.067	**0.068**	0.053
TaxoCritic	0.063	0.062	0.062	0.067	0.058

Table 8: The Edge-F1 score on the evaluation set performance of the of all three models, TaxoRL, DTaxa*, and TaxoCritic after given number of epochs.

F. Robustness Results

F.1. TaxoCritic Robustness

Figure 9 illustrates the example taxonomy. While Figure 10 shows the tree generated by TaxoCritic to evaluate the robustness.

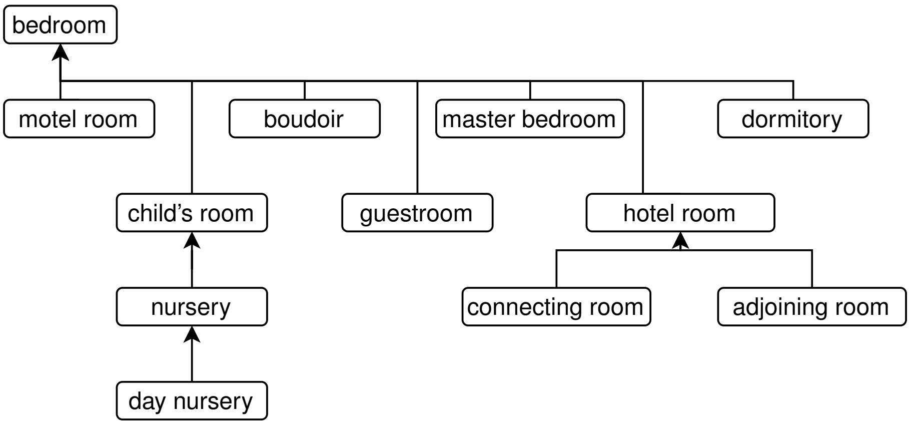

Figure 9: The example taxonomy.

Model	Edge Precision	Edge Recall	Edge F1
TaxoRL	0.23	0.427	0.299
DTaxa*	**0.55**	**0.662**	**0.601**
TaxoCritic	0.321	0.443	0.372

Table 9: Final precision, recall, and F1 scores after training of all three models, TaxoRL, DTaxa*, and TaxoCritic.

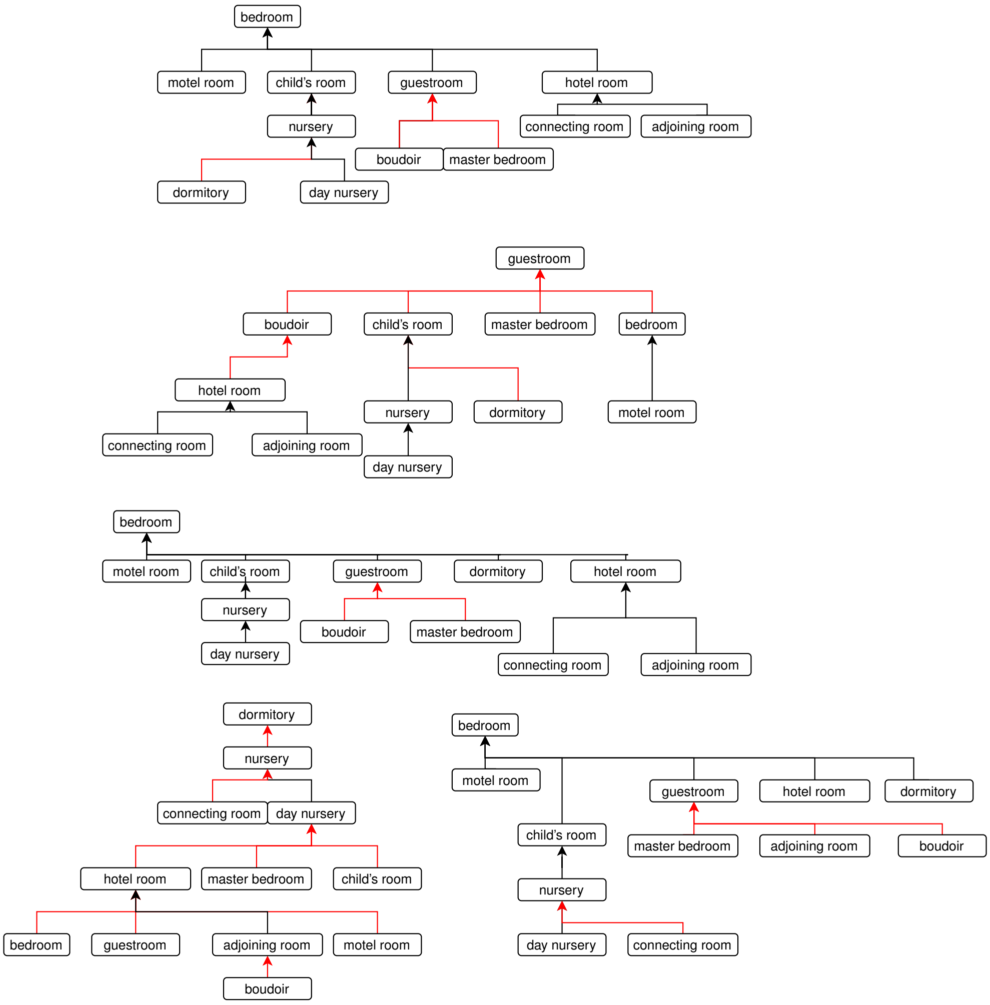

Figure 10: Trees generated by our model to evaluate TaxoCritic's robustness. Red arrows indicate incorrect edges, while black edges represent correct ones.

F.2. DTaxa* Robustness

Figure 11 shows the taxonomy trees generated by DTaxa* during the robustness analysis. Only three different trees were generated, two of the trees occurring twice each.

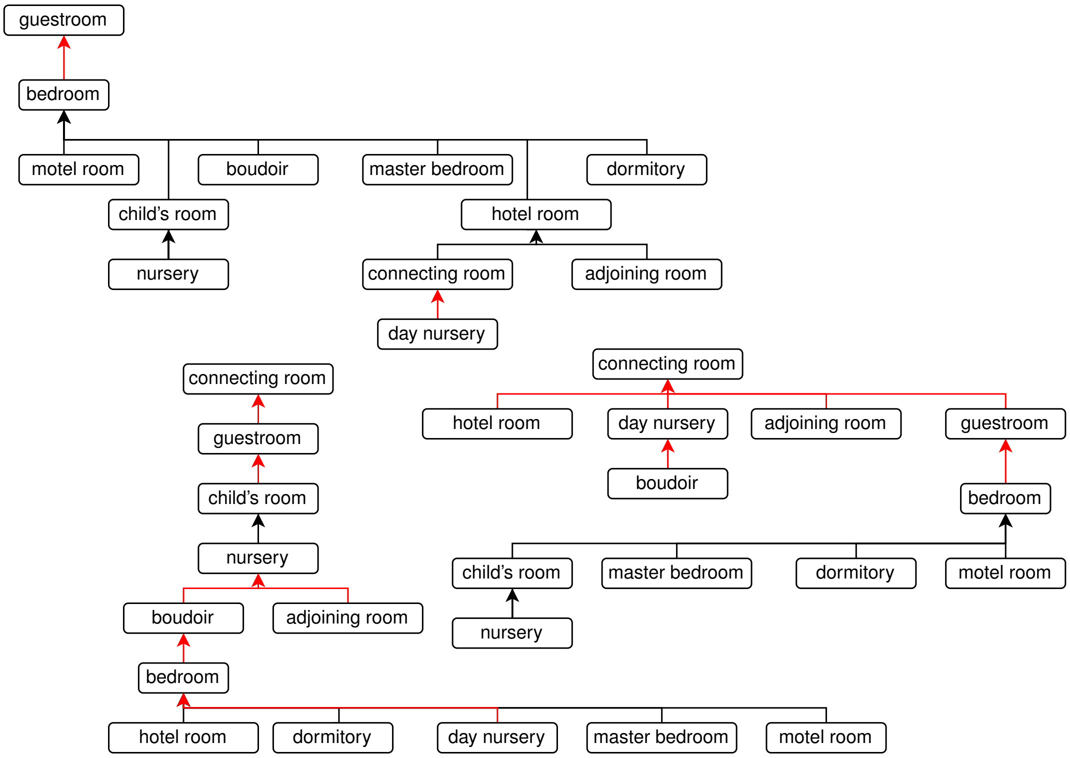

Figure 11: The trees generated by DTaxa* in five attempts, with two trees produced twice. Incorrect edges are indicated by red arrows, while the black edges represent correct identifications.

F.3. TaxoRL Robustness

Figure 12 shows the taxonomy trees generated by TaxoRL during the robustness analysis. Only three different trees were generated, with the first one occurring three times.

G. Credit Assignment Analysis

Referring to Figure 5, we notice that only selecting one of the blue nodes (*child's room, hotel room*) as parents leads to a correct action, as none of the yellow terms can be a parent to any of the potential children. Thus, we anticipate the blue terms to have a higher action values than the yellow ones. Table 10 displays the values of the parent candidates. The average value of the incorrect parent candidates (yellow) is -6.92, while the average value of the correct parent choices (blue) is -4.04. Once again, we observe the same phenomenon as earlier: the sub-critic effectively prioritizes choices that are meaningful independently, even without specific information about the other component of the action (i.e., the choice of child in this case).

	bedroom	boudoir	motel room	guestroom
Action value	-5.76	-4.87	-5.87	-5.24

	master bedroom	dormetry	child's room	hotel room
Action value	-9.00	-10.76	-4.71	-3.37

Table 10: This table shows the inverse actions values of choosing each of the nodes as the parent

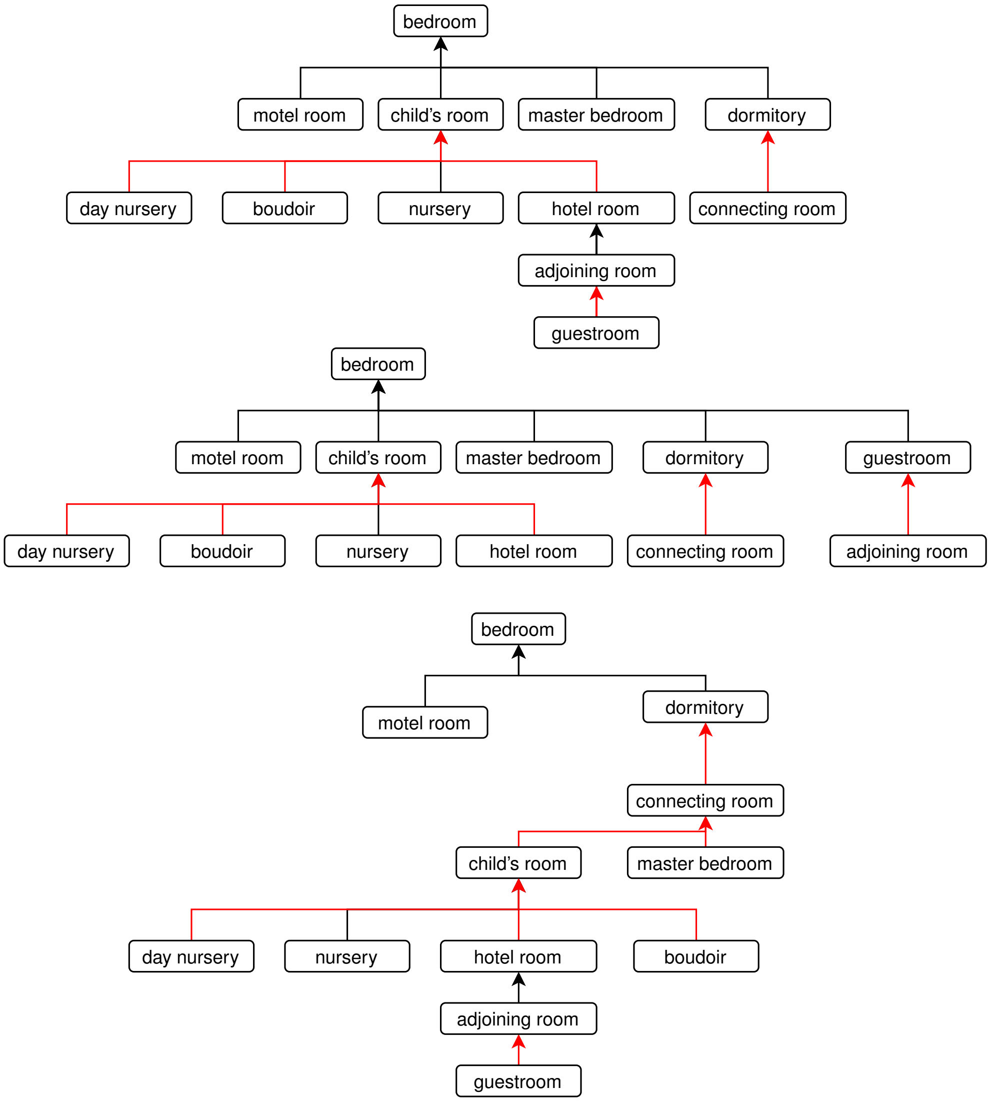

Figure 12: The trees generated by TaxoRL in five attempts, with the first tree occurring three times. Incorrect edges are highlighted by red arrows, while the black edges indicate correct identifications.