Reasoning on Knowledge Graphs with Debate Dynamics

Marcel Hildebrandt,[1,2,*] **Jorge Andres Quintero Serna,**[1,2,*] **Yunpu Ma,**[1,2]
Martin Ringsquandl,[1] **Mitchell Joblin,**[1] **Volker Tresp**[1,2]
[1]Siemens Corporate Technology, [2]Ludwig Maximilian University
first_name.last_name@siemens.com

Abstract

We propose a novel method for automatic reasoning on knowledge graphs based on debate dynamics. The main idea is to frame the task of triple classification as a debate game between two reinforcement learning agents which extract arguments – paths in the knowledge graph – with the goal to promote the fact being true (thesis) or the fact being false (antithesis), respectively. Based on these arguments, a binary classifier, called the judge, decides whether the fact is true or false. The two agents can be considered as sparse, adversarial feature generators that present interpretable evidence for either the thesis or the antithesis. In contrast to other black-box methods, the arguments allow users to get an understanding of the decision of the judge. Since the focus of this work is to create an explainable method that maintains a competitive predictive accuracy, we benchmark our method on the triple classification and link prediction task. Thereby, we find that our method outperforms several baselines on the benchmark datasets FB15k-237, WN18RR, and Hetionet. We also conduct a survey and find that the extracted arguments are informative for users.

1 Introduction

A large variety of information about the real world can be expressed in terms of entities and their relations. Knowledge graphs (KGs) store facts about the world in terms of triples (s, p, o), where s (subject) and o (object) correspond to nodes in the graph and p (predicate) denotes the edge type connecting both. The nodes in the KG represent entities of the real world and predicates describe relations between pairs of entities.

KGs are useful for various artificial intelligence (AI) tasks in different fields such as named entity disambiguation in natural language processing (Han and Zhao 2010), visual relation detection (Baier, Ma, and Tresp 2017), or collaborative filtering (Hildebrandt et al. 2019). Examples of large-size KGs include Freebase (Bollacker et al. 2008) and YAGO (Suchanek, Kasneci, and Weikum 2007). In particular, the Google Knowledge Graph (Singhal 2012) is a well-known example of a comprehensive KG with more than 18 billion facts, used in search, question answering, and various NLP tasks. One major issue, however, is that most real-world KGs are incomplete (i.e., true facts are missing) or contain false facts. Machine learning algorithms designed to address this problem try to infer missing triples or detect false facts based on observed connectivity patterns. Moreover, many tasks such as question answering or collaborative filtering can be formulated in terms of predicting new links in a KG (e.g., (Lukovnikov et al. 2017), (Hildebrandt et al. 2018)). Most machine learning approaches for reasoning on KGs embed both entities and predicates into low dimensional vector spaces. A score for the plausibility of a triple can then be computed based on these embeddings. Common to most embedding-based methods is their black-box nature. This lack of transparency constitutes a potential limitation when it comes to deploying KGs in real world settings. Explainability in the machine learning community has recently gained attention; in many countries laws that require explainable algorithms have been put in place (Goodman and Flaxman 2017). Additionally, in contrast to one-way black-box configurations, comprehensible machine learning methods enable the construction of systems where both machines and users interact and influence each other.

Most explainable AI approaches can be roughly categorized into two groups: Post-hoc interpretability and integrated transparency (Došilović, Brčić, and Hlupić 2018). While post-hoc interpretability aims to explain the outcome of an already trained black-box model (e.g., via layer-wise relevance propagation (Montavon et al. 2017)), integrated transparency-based methods either employ internal explanation mechanisms or are naturally explainable due to low model complexity (e.g., linear models). Since low complexity and prediction accuracy are often conflicting objectives, there is typically a trade off between performance and explainability. The goal of this work is to design a KG reasoning method with integrated transparency that does not sacrifice performance while also allowing a human-in-the-loop.

In this paper we introduce R2D2 (**R**eveal **R**elations using **D**ebate **D**ynamics), a novel method for triple classification based on reinforcement learning. Inspired by the concept outlined in (Irving, Christiano, and Amodei 2018) to increase AI safety via debates, we model the task of triple classification as a debate between two agents, each presenting arguments either in favor of the thesis (the triple is true) or the antithesis (the triple is false). Based on these arguments, a binary

[*]These authors contributed equally to this work.

classifier, called the judge, decides whether the fact is true or false. As opposed to most methods based on representation learning, the arguments can be displayed to users such that they can trace back the classification of the judge and potentially overrule the decision or request additional arguments. Hence, the integrated transparency mechanism of R2D2 is not based on low complexity components, but rather on the automatic extraction of interpretable features. While deep learning made manual feature engineering to great extends redundant, this advantage came at the cost of producing results that are difficult to interpret. Our work is an attempt to close the circle by employing deep learning techniques to automatically select sparse, interpretable features. The major contributions of this work are as follows.

- To the best of our knowledge, R2D2 constitutes the first model based on debate dynamics for reasoning on KGs.
- We benchmark R2D2 with respect to triple classification on the datasets FB15k-237 and WN18RR. Our findings show that R2D2 outperforms all baseline methods with respect to the accuracy, the PR AUC, and the ROC AUC, while being more interpretable.
- To demonstrate that R2D2 can in principle be employed for KG completion, we also evaluate its link prediction performance on a subset of FB15k-237. To include a real world task, we employ R2D2 on Hetionet for finding gene-disease associations and new target diseases for drugs. R2D2 either outperforms or keeps up with the performance of all baseline methods on both datasets with respect to standard measures such as the MRR, the mean rank, and hits@k, for k = 3, 10.
- We conduct a survey where respondents take the role of the judge classifying the truthfulness of statements solely based on the extracted arguments. Based on a majority vote, we find that nine out of ten statements are classified correctly and that for each statement the classification of the respondents agrees with the decision of R2D2's judge. These findings indicate that the arguments of R2D2 are informative and the judge is aligned with human intuition.

This paper is organized as follows. We briefly review KGs and the related literature in the next section. Section 3 describes the methodology of R2D2. Section 4 details an experimental study on the benchmark datasets FB15k-237, WN18RR, and Hetionet. In particular, we compare R2D2 with various methods from the literature and describes the findings of our survey. In Section 5 the quality of the arguments and future works are discussed. We conclude in Section 6.

2 Background and Related Work

In this section we provide a brief introduction to KGs in a formal setting and review the most relevant related work. Let $\mathcal{E}$ denote the set of entities and consider the set of binary relations $\mathcal{R}$. A knowledge graph $\mathcal{KG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a collection of facts stored as triples of the form (s, p, o) – subject, predicate, and object. To indicate whether a triple is true or false, we consider the binary characteristic function $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \{0, 1\}$. For all $(s, p, o) \in \mathcal{KG}$ we assume

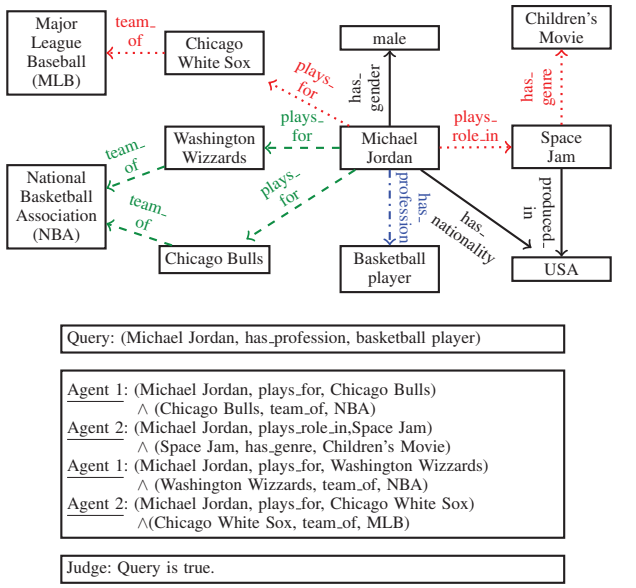

Figure 1: The agents debate whether Michael Jordan is a professional basketball player. While agent 1 extracts arguments from the KG supporting the thesis that the fact is true (green), agent 2 argues that it is false (red). Based on the arguments the judge decides that Michael Jordan is a professional basketball player.

$\phi(s, p, o) = 1$ (i.e., a KG is a collection of true facts). However, in case a triple is not contained in $\mathcal{KG}$, it does not imply that the corresponding fact is false but rather unknown (open world assumption). Since most KGs that are currently in use are incomplete in the sense that they do not contain all true triples or they actually contain false facts, many canonical machine learning tasks are related to KG reasoning. KG reasoning can be roughly categorized according to the following two tasks: first, inference of missing triples (KG completion or link prediction), and second, predicting the truth value of triples (triple classification). While different formulations of these tasks are typically found in the literature (e.g., the completion task may involve predicting either subject or object entities as well as relations between a pair of entities), we employ the following definitions throughout this work.

Definition 1 (Triple Classification and KG completion)**.** Given a triple $(s, p, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, triple classification is concerned with predicting the truth value $\phi(s, p, o)$. KG completion is the task to rank object entities $o \in \mathcal{E}$ by their likelihood to form a true triple together with a given subject-predicate-pair $(s, p) \in \mathcal{E} \times \mathcal{R}$. [1]

Many machine learning methods for KGs can be trained to operate in both settings. For example, a triple classifier of the form $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to [0, 1]$ with $f(s, p, o) \approx \phi(s, p, o)$,

[1] Throughout this work, we assume the existence of inverse relations. That means for any relation $p \in \mathcal{R}$ there exists a relation $p^{-1} \in \mathcal{R}$ such that $(s, p, o) \in \mathcal{KG}$ if and only if $(o, p^{-1}, s) \in \mathcal{KG}$. Hence, the restriction to rank object entities does not lead to a loss of generality.

induces a completion method given by $f(s, p, \cdot) : \mathcal{E} \to [0, 1]$, where function values for different object entities can be used to produce a ranking. While the architecture of R2D2 is designed for triple classification, we demonstrate that it can in principle also work in the KG completion setting. The performance on both tasks is reported in Section 4.

Representation learning is an effective and popular technique underlying many KG refinement methods. The basic idea is to project both entities and relations into a low dimensional vector space. Then the likelihood of triples is modelled as a functional on the embedding spaces. Popular completion methods based on representation learning include the translational embedding methods TransE (Bordes et al. 2013) and TransR (Lin et al. 2015) as well as the factorization approaches RESCAL (Nickel, Tresp, and Kriegel 2011), DistMult (Yang et al. 2015), ComplEx (Trouillon et al. 2016), and SimplE (Kazemi and Poole 2018). Path-based reasoning methods follow a different philosophy. For instance, the Path-Ranking Algorithm (PRA) proposed in (Lao, Mitchell, and Cohen 2011) uses for inference a combination of weighted random walks through the graph. In (Xiong, Hoang, and Wang 2017) the reinforcement learning based path searching approach called DeepPath was proposed, where an agent picks relational paths between entity pairs. Recently, and more related to our work, the multi-hop reasoning method MINERVA was proposed in (Das et al. 2018). The basic idea in that paper is to display the query subject and predicate to the agents and let them perform a policy guided walk to the correct object entity. The paths that MINERVA produces also lead to some degree of explainability. However, we find that only actively mining arguments for the thesis and the antithesis, thus exposing both sides of a debate, allows users to make a well-informed decision. Mining evidence for both positions can also be considered as adversarial feature generation, making the classifier (judge) robust towards contradictory evidence or corrupted data.

3 Our Method

We formulate the task of triple classification in terms of a debate between two opposing agents. Thereby, a query triple corresponds to the statement around which the debate is centered. The agents proceed by mining paths on the KG that serve as evidence for the thesis or the antithesis. More precisely, they traverse the graph sequentially and select the next hop based on a policy that takes past transitions and the query triple into account. The transitions are added to the current path, extending the argument. All paths are processed by a binary classifier called the judge that attempts to distinguish between true and false triples based on the arguments provided by the agents. Figure 1 shows an exemplary debate. The main steps of a debate can be summarized as follows:

1. A query triple around which the debate is centered is presented to both agents.
2. The two agents take turns extracting paths from the KG that serve as arguments for the thesis and the antithesis.
3. The judge processes the arguments along with the query triple and estimates the truth value of the query triple.

While the parameters of the judge are fitted in a supervised fashion, both agents are trained to navigate through the graph using reinforcement learning. Based on the formal framework presented in (Das et al. 2018), the agents' learning tasks are modelled via the fixed horizon decision processes outlined below.

States The fully observable state space $\mathcal{S}$ for each agent is given by $\mathcal{E}^2 \times \mathcal{R} \times \mathcal{E}$. Intuitively, we want the state to encode the location of exploration $e_t^{(i)}$ (i.e., the current location) of agent $i \in \{1, 2\}$ at time t and the query triple $q = (s_q, p_q, o_q)$. Thus, a state $S_t^{(i)} \in \mathcal{S}$ for time $t \in \mathbb{N}$ is represented by $S_t^{(i)} = \left(e_t^{(i)}, q\right)$.

Actions The set of possible actions for agent i from a state $S_t^{(i)} = \left(e_t^{(i)}, q\right)$ is denoted by $\mathcal{A}_{S_t^{(i)}}$. It consists of all outgoing edges from the node $e_t^{(i)}$ and the corresponding target nodes. More formally, $\mathcal{A}_{S_t^{(i)}} = \left\{(r, e) \in \mathcal{R} \times \mathcal{E} : S_t^{(i)} = \left(e_t^{(i)}, q\right) \wedge \left(e_t^{(i)}, r, e\right) \in \mathcal{KG}\right\}$. Moreover, we denote with $A_t^{(i)} \in \mathcal{A}_{S_t^{(i)}}$ the action that agent i performed at time t. We include self-loops for each node such that the agent can stay at the current node.

Environments The environments evolve deterministically by updating the state according to the agents' actions (i.e., by changing the agents' locations), whereby the query fact remains the same. Formally, the transition function of agent i at time t is given by $\delta_t^{(i)}(S_t^{(i)}, A_t^{(i)}) := \left(e_{t+1}^{(i)}, q\right)$ with $S_t^{(i)} = \left(e_t^{(i)}, q\right)$ and $A_t^{(i)} = \left(r, e_{t+1}^{(i)}\right)$.

Policies We denote the history of agent i up to time t with the tuple $H_t^{(i)} = \left(H_{t-1}^{(i)}, A_{t-1}^{(i)}\right)$ for $t \geq 1$ and $H_0^{(i)} = (s_q, p_q, o_q)$ along with $A_0^{(i)} = \emptyset$ for $t = 0$. The agents encode their histories via LSTMs (Hochreiter and Schmidhuber 1997)

$$\boldsymbol{h}_t^{(i)} = \text{LSTM}^{(i)}\left(\left[\boldsymbol{a}_{t-1}^{(i)}, \boldsymbol{q}^{(i)}\right]\right), \quad (1)$$

where $\boldsymbol{a}_{t-1}^{(i)} = \left[\boldsymbol{r}_{t-1}^{(i)}, \boldsymbol{e}_t^{(i)}\right] \in \mathbb{R}^{2d}$ corresponds to the vector space embedding of the previous action (or the zero vector for at time $t = 0$) with $\boldsymbol{r}_{t-1}^{(i)}$ and $\boldsymbol{e}_t^{(i)}$ denoting the embeddings of the relation and the target entity into $\mathbb{R}^d$, respectively. Moreover, $\boldsymbol{q}^{(i)} = \left[\boldsymbol{e}_s^{(i)}, \boldsymbol{r}_p^{(i)}, \boldsymbol{e}_o^{(i)}\right] \in \mathbb{R}^{3d}$ encodes the query triple for agent i. Both entity and relation embeddings are specific for each agent and learned in the debate process during training. Note that expanding the state space definitions with the histories leads to a Markov decision processes.

The history-dependent action distribution of each agent is given by

$$\boldsymbol{d}_t^{(i)} = \text{softmax}\left(\boldsymbol{A}_t^{(i)}\left(\boldsymbol{W}_2^{(i)}\text{ReLU}\left(\boldsymbol{W}_1^{(i)}\boldsymbol{h}_t^{(i)}\right)\right)\right), \quad (2)$$

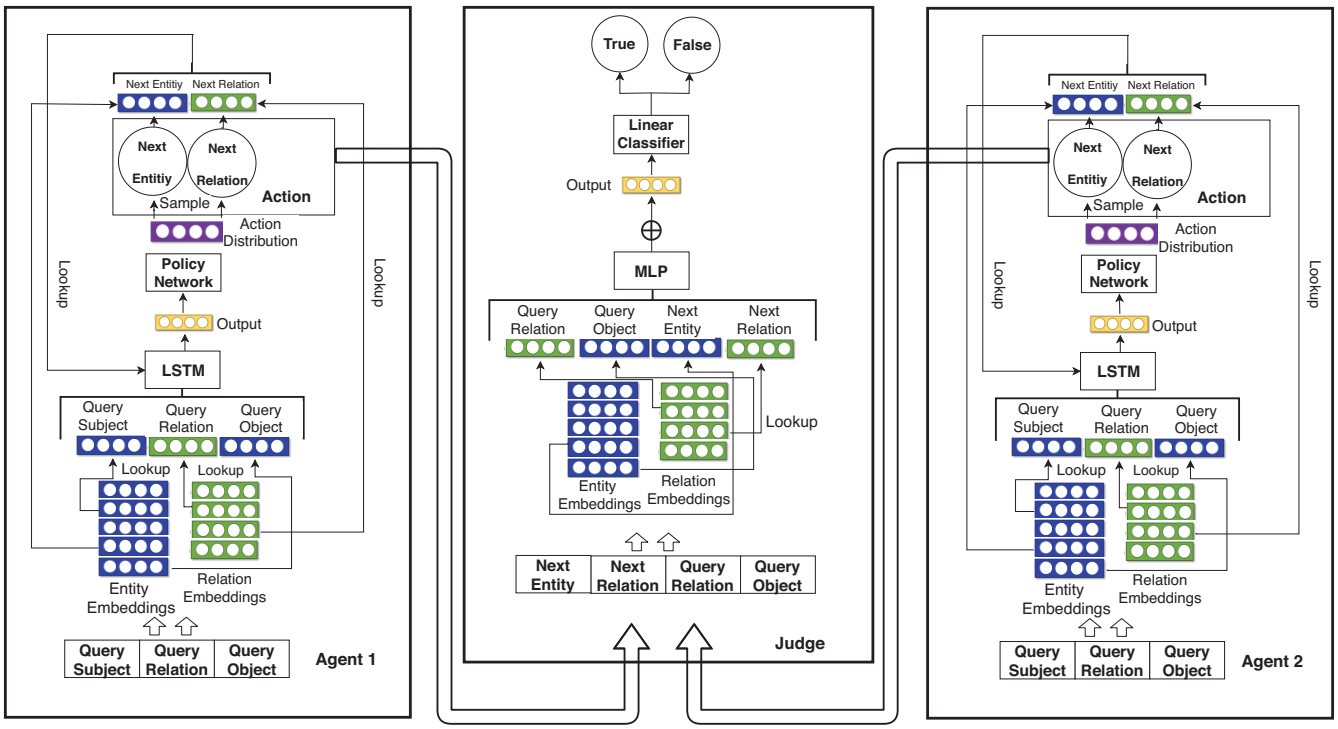

Figure 2: The overall architecture of R2D2; the two agents extract arguments from the KG. Along with the query relation and the query object, these arguments are processed by the judge who classifies whether the query is true or false.

where the rows of $\boldsymbol{A}_t^{(i)} \in \mathbb{R}^{|\mathcal{A}_{S_t^{(i)}}| \times d}$ contain latent representations of all admissible actions from $S_t^{(i)}$. The action $A_t^{(i)} = (r, e) \in \mathcal{A}_{S_t^{(i)}}$ is drawn according to

$$A_t^{(i)} \sim \text{Categorical}\left(\boldsymbol{d}_t^{(i)}\right) . \tag{3}$$

Equations (1) and (2) define a mapping from the space of histories to the space of distribution over all admissible actions, thus inducing a policy $\pi_{\theta^{(i)}}$, where $\theta^{(i)}$ denotes the set of all trainable parameters in Equations (1) and (2).

Debate Dynamics In a first step, the query triple $q = (s_q, p_q, o_q)$ with truth value $\phi(q) \in \{0, 1\}$ is presented to both agents. Agent 1 argues that the fact is true, while agent 2 argues that it is false. Similar to most formal debates, we consider a fixed number of rounds $N \in \mathbb{N}$. In every round $n = 1, 2, \ldots, N$, the agents start graph traversals with fixed length $T \in \mathbb{N}$ from the subject node of the query s_q. The judge observes the paths of the agents and predicts the truth value of the triple. Agent 1 starts the game generating a sequence of length T consisting of states and actions according to Equations (1 - 3). Then agent 2 proceeds by producing a similar sequence starting from s_q. Algorithm 1 contains a pseudocode of R2D2 at inference time.

To ease the notation we have enumerated all actions consecutively and dropped the superscripts that indicate which agent performs the action. Then the sequence corresponding to the n-th argument of agent i is given by

$$\tau_n^{(i)} := \left(A_{\tilde{n}(i,T)+1}, A_{\tilde{n}(i,T)+2}, \ldots, A_{\tilde{n}(i,T)+T}\right) , \tag{4}$$

where we used the reindexing $\tilde{n}(i, T) := (2(n-1)+i-1)T$. The sequence of all arguments is denoted by

$$\tau := \left(\tau_1^{(1)}, \tau_1^{(2)}, \tau_2^{(1)}, \tau_2^{(2)}, \ldots, \tau_N^{(1)}, \tau_N^{(2)}\right) . \tag{5}$$

The Judge The role of the judge in R2D2 is twofold: First, the judge is a binary classifier that tries to distinguish between true and false facts. Second, the judge also evaluates the quality of the arguments extracted by the agents and assigns rewards to them. Thus, the judge also acts as a critic, teaching the agents to produce meaningful arguments. The judge processes each argument together with the query individually by a feed forward neural network $f : \mathbb{R}^{2(T+1)d} \to \mathbb{R}^d$, sums the output for each argument up and processes the resulting sum by a binary classifier. More concretely, after processing each argument individually, the judge produces a representation according to

$$\boldsymbol{y}_n^{(i)} = f\left(\left[\boldsymbol{\tau}_n^{(i)}, \boldsymbol{q}^J\right]\right) \tag{6}$$

with

$$\boldsymbol{\tau}_n^{(i)} := \left[\boldsymbol{a}^J_{\tilde{n}(i,T)+1}, \boldsymbol{a}^J_{\tilde{n}(i,T)+2}, \ldots, \boldsymbol{a}^J_{\tilde{n}(i,T)+T}\right] \tag{7}$$

where $\boldsymbol{a}_t^J = \left[\boldsymbol{r}_t^J, \boldsymbol{e}_t^J\right] \in \mathbb{R}^{2d}$ denotes the judge's embedding for the action A_t and $\boldsymbol{q}^J = \left[\boldsymbol{r}_p^J, \boldsymbol{e}_o^J\right] \in \mathbb{R}^{2d}$ encodes the query predicate and the query object. Note that the query

Algorithm 1: R2D2 Inference

input : Triple query $q = (s_q, p_q, o_q)$
output : Classification score $t_\tau \in [0, 1]$ of the judge along with the list of arguments τ

$\tau \leftarrow [\,]$ // Initialize the list of arguments with an empty list
// Loop over the debate rounds
for $n = 1$ *to* N **do**
  // Loop over the two agents
  for $i = 1$ *to* 2 **do**
    $e_1^{(i)} \leftarrow s_q$ // Initialize the position of the agent
    $\tau_n^{(i)} \leftarrow [\,]$ // Initialize the argument with an empty list
    // Loop over the path
    for $t = 1$ *to* T **do**
      Sample a transition (r, e) from $e_t^{(i)}$ according to $\pi_{\theta^{(i)}}$ // See Equations (1-3)
      $\tau_n^{(i)}$.append(r, e) // Extend the argument
      $e_{t+1}^{(i)} \leftarrow e$ // Update the position of the agent
    end
    τ.append$(\tau_n^{(i)})$ // Extend the list of all argument
  end
end
Process τ via the judge and retrieve the classification score t_τ // See Equation (6-8)
return t_τ *and* τ

subject is not revealed to the judge because we want the judge to base its decisions solely on the agents' actions rather than on the embedding of the query subject. After processing all arguments in τ, the debate is terminated and the judge scores the query triple q with $t_\tau \in (0, 1)$ according to

$$t_\tau = \sigma\left(\boldsymbol{w}^\intercal \mathrm{ReLU}\left(\boldsymbol{W} \sum_{i=1}^{2} \sum_{n=1}^{N} \boldsymbol{y}_n^{(i)}\right)\right), \tag{8}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{w} \in \mathbb{R}^d$ denote the trainable parameters of the classifier and $\sigma(\cdot)$ denotes the sigmoid activation function. We also experimented with more complex architectures where the judge processes each argument in τ via a recurrent neural network. However, we found that both the classification performance and the quality of the arguments suffered.

The objective function of the judge for a single query q is given by the cross-entropy loss

$$\mathcal{L}_q = \phi(q) \log t_\tau + (1 - \phi(q)) (1 - \log t_\tau). \tag{9}$$

Hence, during training, we aim to minimize the overall loss given by

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{q \in \mathcal{T}} \mathcal{L}_q, \tag{10}$$

where $\mathcal{T}$ denotes the set of training triples. To prevent overfitting, an additional L_2-penalization term with strength $\lambda \in \mathbb{R}_{\geq 0}$ on the parameters of the judge is added to Equation (10).

An overview of the overall architecture of R2D2 is depicted in Figure 2.

Rewards In order to generate feedback for the agents, the judge also processes each argument $\tau_n^{(i)}$ individually and produces a score according to

$$t_n^{(i)} = \boldsymbol{w}^\intercal \mathrm{ReLU}\left(\boldsymbol{W} f\left(\left[\boldsymbol{\tau}_n^{(i)}, \boldsymbol{q}^J\right]\right)\right), \tag{11}$$

where both the neural network f as well as the linear weights vector $\boldsymbol{w}$ correspond to the definitions given in the previous paragraph. Thus, $t_n^{(i)}$ corresponds to the classification logits of q solely based on the n-th argument of agent i. Since agent 1 argues for the thesis and agent 2 for the antithesis, the rewards are given by

$$R_n^{(i)} = \begin{cases} t_n^{(i)} & \text{if } i = 1 \\ -t_n^{(i)} & \text{otherwise.} \end{cases} \tag{12}$$

Intuitively speaking, this means that the agents receive high rewards whenever they extract an argument that is considered by judge as strong evidence for their position

Reward Maximization and Training Scheme We employ REINFORCE (Williams 1992) to maximize the expected cumulative reward of the agents given by

$$G^{(i)} := \sum_{n=1}^{N} R_n^{(i)}. \tag{13}$$

Thus, the agents' maximization problems are given by

$$\arg\max_{\theta^{(i)}} \mathbb{E}_{q \sim \mathcal{KG}_+} \mathbb{E}_{\tau_1^{(i)}, \tau_2^{(i)}, \ldots, \tau_N^{(i)} \sim \pi_{\theta^{(i)}}} \left[G^{(i)} \,\middle|\, q \right], \tag{14}$$

where $\mathcal{KG}_+$ is the set of training triples that contain in addition to observed triples in $\mathcal{KG}$ also unobserved triples. The rationale is as follows: As KGs only contain true facts, sampling queries from $\mathcal{KG}$ would create a dataset without negative labels. Therefore it is common to create corrupted triples that are constructed from correct triples (s, p, o) by replacing the object with an entity $\tilde{o}$ to create a false triple $(s, p, \tilde{o}) \notin \mathcal{KG}$ (see (Bordes et al. 2013)). Rather than creating any kind of corrupt triples, we generate a set of plausible but false triples. More concretely, for each $(s, p, o) \in \mathcal{KG}$ we generate one triple $(s, p, \tilde{o}) \notin \mathcal{KG}$ with the constraint that $\tilde{o}$ appears in the database as the object with respect to the relation p. More formally, we denote the set of corrupted triples with $\mathcal{KG}_C := \{(s, p, \tilde{o}) \,|\, (s, p, \tilde{o}) \notin \mathcal{KG}, \exists \tilde{s} : (\tilde{s}, p, \tilde{o}) \in \mathcal{KG}\}$. Then the set of training triples $\mathcal{T}$ is contained in $\mathcal{KG}_+ :=$

Dataset	Entities	Relations	Triples
FB15k-237	14,541	237	310,116
WN18RR	40,943	11	93,003
Hetionet	47,031	24	2,250,197

Table 1: Statistics of the datasets used in the experiments.

$\mathcal{KG} \cup \mathcal{KG}_C$. The underlying rationale for working with plausible but false facts is that we do not waste resources on triples that break implicit type-constrains. Since this heuristic only needs to be computed once and filters out triples that could easily be discarded by a type-checker, we can focus on the prediction of facts that present more of a challenge.

During training the first expectation in Equation (14) is substituted with the empirical average over the training set. The second expectation is approximated by the empirical average over multiple rollouts. We also employ a moving average baseline to reduce the variance. Further, we use entropy regularization with parameter $\beta \in \mathbb{R}_{\geq 0}$ to enforce exploration.

In order to address the problem that the agents require a trained judge to obtain meaningful reward signals, we freeze the weights of the agents for the first episodes of the training. The rationale is that training the judge does not necessarily rely on the agents being perfectly aligned with their actual goals. For example, even if the agents do not extract arguments that correspond to their position, they can still provide useful features that the judge learns to exploit. After the initial training phase, where we only fit the parameters of the judge, we employ an alternating training scheme where we either train the judge or the agents.

4 Experiments

Datasets We measure the performance of R2D2 with respect to the triple classification and the KG completion task on the benchmark datasets FB15k-237 (Toutanova et al. 2015) and WN18RR (Dettmers et al. 2018). To test R2D2 on a real world task we also consider Hetionet (Himmelstein and Baranzini 2015), a large scale, heterogeneous graph encoding information about chemical compounds, diseases, genes, and molecular functions. We employ R2D2 for detecting gene-disease associations and finding new target diseases for drugs, two tasks of high practical relevance in the biomedical domain (see (Himmelstein and Baranzini 2015)). The statistics of all datasets are given in Table 1.

Metrics and Evaluation Scheme As outlined in Section 2, triple classification aims to decide whether a query triple (s_q, p_q, o_q) is true or false. Hence, it is a binary classification task. For each method we set a threshold δ obtained by maximizing the accuracies on the validation set. That means, for a given query triple (s_q, p_q, o_q), if its score (e.g., given by Equation (8) for R2D2) is larger than δ, the triple will be classified as true, otherwise as false. Since most KGs do not contain facts that are labeled as false, we have generated a set of negative triples: For each observed triple in the validation and test set we create a false but plausible fact (see Section 3).[2] We report the accuracy, the PR AUC, and ROC AUC for all methods. Since R2D2 is a stochastic classifier, we can produce multiple rollouts of the same query at inference time and average the resulting classification scores to lower the variance.

Even though the purpose of R2D2 is triple classification, one can turn it into a KG completion method as follows: We consider a range of object entities each producing a different classification score t_τ given by Equation (8). Since t_τ can be interpreted as a measure for the plausibility of a triple, we use the classification scores to produce a ranking. More concretely, we rank each correct triple in the test set against all plausible but false triples (see Section 3). Since this procedure is computational expensive during training (one needs to run multiple debates per training triple to produce a ranking), we select the following relations for training and testing purposes: For FB15k-237 we follow (Socher et al. 2013) and consider the relations 'profession', 'nationality', 'ethnicity', and 'religion'. Following (Himmelstein and Baranzini 2015) and (Himmelstein et al. 2017), the relations 'gene_associated_with_disease' and 'compound_treats_disease' are considered for Hetionet. We report the mean rank of the correct entity, the mean reciprocal rank (MRR), as well as Hits@k for $k = 1, 3, 10$ - the percentage of test triples where the correct entity is ranked in the top k.

In order to find the most suitable set of hyperparameters for all considered methods, we perform cross-validations. Thereby the canonical splits of the datasets into a training, validation, and test set are used. In particular, we ensured that triples that are assigned to the validation or test set (and their respective inverse relations) are not included in the KG during training. The results on the test set of all methods are reported based on the hyperparameters that showed the best performance (based on the highest accuracy for triple classification and the the highest MRR for link prediction) on the validation set. We considered the following hyperparameter ranges for R2D2: The number of latent dimensions d for the embeddings is chosen from the range $\{32, 64, 128\}$. The number of LSTM layers for the agents is chosen from $\{1, 2, 3\}$. The the number of layers in the MLP for the judge is tuned in the range $\{1, 2, 3, 4, 5\}$. β was chosen from $\{0.02, 0.05, 0.1\}$. The length of each argument T was tuned in the range $\{1, 2, 3\}$ and the number of debate rounds N was set to 3. Moreover, the L_2-regularization strength λ is set to 0.02. Furthermore, the number of rollouts during training is given by 20 and 50 (triple classification) or 100 (KG completion) at test time. The loss of the judge and the rewards of the agents were optimized using Adam with learning rate given 10^{-4}. The best hyperparameter are reported in Table 3.

All experiments were conducted on a machine with 48 CPU cores and 96 GB RAM. Training R2D2 on either dataset takes at most 4 hour. Testing takes about 1-2 hours depending on the dataset.

[2]The datasets along with the code of R2D2 are available at https://github.com/m-hildebrandt/R2D2.

Dataset	FB15k-237			WN18RR		
Method	Acc	PR AUC	ROC AUC	Acc	PR AUC	ROC AUC
DistMult	0.739	0.78	0.803	**0.804**	**0.901**	**0.872**
ComplEx	0.738	0.789	0.796	0.802	0.887	0.860
TransE	0.673	0.727	0.736	0.69	0.794	0.732
TransR	0.612	0.655	0.651	0.721	0.724	0.792
SimplE	0.703	0.733	0.756	0.722	0.812	0.742
R2D2	**0.751**	**0.86**	**0.848**	0.726	0.821	0.808
R2D2$_+$	**0.764**	**0.865**	**0.857**	**0.804**	**0.909**	**0.893**

Table 2: The performance on the triple classification task.

Parameter	FB15k-237	WN18RR	FB15k-237 (subset)	Hetionet
Embedding size (d)	64	64	64	32
# stacked LSTM cells (agents)	2	1	2	2
# layers MLP (judge)	1	1	3	2
# rounds in a debate (N)	3	3	3	3
Argument/path length (T)	2	2	2	2
Entropy regularization (β)	0.02	0.02	0.1	0.1

Table 3: The best hyperparameters for R2D2 found via cross-validation.

Results

Triple Classification We compare the performance of R2D2 on the triple classifications task against DistMult, ComplEx, TransE, TransR, and SimplE. The results are displayed in Table 2. On FB15k-237, R2D2 outperforms all baselines with respect to the accuracy, the PR AUC, and the ROC AUC. However, on WN18RR the performance of R2D2 is dominated by the factorization methods ComplEx and DistMult by a significant margin. We conjecture that this is due to the sparsity in the dataset. As a remedy we employ pretrained embeddings from TransE that are fixed during training.[3] We denote the resulting method with R2D2$_+$ and find that it outperforms all other methods with respect to the PR AUC and ROC AUC on WN18RR. We also test R2D2$_+$ on FB15k-237 and find that it improves the results of R2D2 by only a small margin. This is expected since FB15k-237 is not as sparse as WN18RR.

KG completion Next to the baselines used for triple classification we also employ the path based link prediction method MINERVA. Note that it is not possible to compute a fair mean rank for MINERVA, since it does not produce a complete ranking of all candidate objects. Table 4 displays the results on the completion task for all methods under consideration on FB15k-237 and Hetionet (subsets; see above). R2D2 outperforms all other methods on FB15k-237 with respect to all metrics but Hits@10. However, the performance of MINERVA is almost on par. Moreover, R2D2 outperforms all baselines on Hetionet with respect to the MRR, the mean rank, Hits@3, and Hits@10. While MINERVA exhibits the best performance with respect to Hits@1, R2D2 yields significantly better results with respect to all other metrics.

[3]We choose TransE embeddings due to the simple functional relations between entities. These can be easily exploited by R2D2.

Survey To asses whether the arguments are informative for users in an objective setting, we conducted a survey where respondents take the role of the judge making a classification decision based on the agents' arguments. More concretely, we set up an online quiz consisting of ten rounds. Each round is centered around a query (with masked subject) sampled from the test set of FB15k-237 (KG completion). Along with the query statement we present the users six arguments extracted by the agents in randomized order. Based on these arguments the respondents are supposed to judge whether the statement is true or false. In addition, we asked the respondents to rate their confidence in each round.

Based on 44 participants (109 invitations were sent) we find that the overall accuracy of the respondents' classifications was 81.8%. Moreover, based on a majority vote (i.e., classification based on the majority of respondents) nine out of ten questions were classified correctly indicating that humans are approximately on par with the performance of the automated judge. Further, the statement where the majority of respondents was wrong corresponds to the only query that was also misclassified by the judge. In this round the participants were supposed to decide whether a person has the religion Methodism. It is hard to answer this question correctly because the person at hand is Margaret Thatcher who had two different religions over her lifetime: Methodism and the Church of England. The fact that the majority of respondents and the judge agree in all rounds indicates that the judge is aligned with human intuition and that the arguments are informative. Moreover, we found that when users assigned a high confidence score to their decision ('rather certain' or 'absolutely certain') the overall accuracy of their classification was 89%. The accuracy dropped to 68.4% when users assigned a low confidence score ('rather uncertain' or 'absolutely uncertain').

Dataset	FB15k-237 (subset)					Hetionet (subset)				
Metrics	MRR	Mean Rank	Hits@1	Hits@3	Hits@10	MRR	Mean Rank	Hits@1	Hits@3	Hits@10
DistMult	0.502	8.607	0.363	0.572	0.779	0.134	31.190	0.054	0.121	0.291
ComplEx	0.521	7.477	0.383	0.587	0.806	0.148	31.439	0.061	0.141	0.325
TransE	0.473	10.2112	0.345	0.522	0.745	0.110	35.559	0.033	0.097	0.267
TransR	0.543	8.737	0.391	0.635	0.816	0.144	27.841	0.049	0.136	0.340
SimplE	0.429	11.760	0.275	0.506	0.736	0.177	31.965	0.091	0.174	0.354
MINERVA	0.580	–	0.448	0.657	**0.857**	0.174	–	**0.097**	0.18	0.364
R2D2	**0.589**	**6.332**	**0.459**	**0.665**	0.853	**0.206**	**23.486**	0.090	**0.219**	**0.455**

Table 4: The performance on the KG completion task.

Query:	Richard Feynman $\xrightarrow{nationality}$ USA?	Nelson Mandela $\xrightarrow{hasProfession}$ Actor?
Agent 1:	Richard Feynman $\xrightarrow{livedInLocation}$ Queens	Nelson Mandela $\xrightarrow{hasFriend}$ Naomi Campbell
	$\wedge$ Queens $\xrightarrow{locatedIn}$ USA	Naomi Campbell $\xrightarrow{hasDated}$ Leonardo DiCaprio
Agent 2:	Richard Feynman $\xrightarrow{hasEthnicity}$ Russian people	Nelson Mandela $\xrightarrow{hasProfession}$ Lawyer
	$\wedge$ Russian people $\xrightarrow{geographicDistribution}$ Republic of Tajikistan	$\wedge$ Lawyer $\xrightarrow{specializationOf^{-1}}$ Barrister

Table 5: Two example debates generated by R2D2: While agent 1 argues that the query is true and agent 2 argues that it is false.

5 Discussion and Future Works

We examined the quality of the extracted paths manually and typically found reasonable arguments, but quite often also arguments that do not make intuitive sense. We conjecture that one reason for that is that agents often have difficulties finding meaningful evidence if they are arguing for the false position. Moreover, for many arguments, most of the relevant information is already contained in the first step of the agents and later transitions often contain seemingly irrelevant information. This phenomenon might be due to the fact that the judge ignores later transitions and agents do not receive meaningful rewards. Further, relevant information about the neighborhood of entities can be encoded in the embeddings of entities. While the judge has access to this information through the training process, it remains hidden to users. For example, when arguing that Nelson Mandela was an actor (see Table 5), the argument of agent 1 requires the user to know that Naomi Campbell and Leonardo DiCaprio are actors (which is encoded in FB15k-237). Then this argument serves as evidence that Nelson Mandela was also an actor since people tend to have friends that share their profession (social homophily). However, without this context information it is not intuitively clear why this is a reasonable argument.

To examine the interplay between the agents and the judge, we consider a setting where we train R2D2 with two agents, but neglect the arguments of one agent during testing. This should lead to a biased outcome in favor of the agent whose arguments are considered by the judge. We test this setting on FB15k-237 and find that when we consider only the arguments of agent 1, the number of positive predictions increases by 18.8%. In contrast, when we only consider agent 2, the number positive predictions drops by 70.2%. This result shows that the debate dynamics are functioning as intended and that the agents learn to extract arguments that the judge considers as evidence for their respective position.

While the results of the survey are encouraging, we plan to develop variants of R2D2 that improve the quality of the arguments and conduct a large scale experimental study that also includes other baselines in a controlled setting. Moreover, we plan to discuss fairness and responsibility considerations. In that regard, (Nickel et al. 2015) stress that when applying statistical methods to incomplete KGs the results are likely to be affected by biases in the data generating process and should be interpreted accordingly. Otherwise, blindly following these predictions can strengthen the bias. While the judge in our method also exploits skews in the data, the arguments can help to identify these biases and potentially exclude problematic arguments from the decision.

6 Conclusion

We proposed R2D2, a new approach for KG reasoning based on a debate game between two opposing reinforcement learning agents. The agents search the KG for arguments that convince a binary classifier (judge) of their position. Thereby, they act as sparse, adversarial feature generators. Since the judge bases its decision solely on mined arguments, R2D2 is more interpretable than other baseline methods. Our experiments showed that R2D2 outperforms all baselines in the triple classification setting with respect to all metrics on the benchmark datasets WN18RR and FB15k-237. Moreover, we demonstrated that R2D2 can in principle operate in the KG completion setting. We found that R2D2 has competitive performance compared to all baselines on a subset of FB15k-237 and Hetionet. Furthermore, the results of our survey indicate that the arguments are informative and that the judge is aligned with human intuition.

References

Baier, S.; Ma, Y.; and Tresp, V. 2017. Improving visual relationship detection using semantic modeling of scene descriptions. In *International Semantic Web Conference*, 53–68. Springer.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database

for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. ACM.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Došilović, F. K.; Brčić, M.; and Hlupić, N. 2018. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 0210–0215. IEEE.

Goodman, B., and Flaxman, S. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine* 38(3):50–57.

Han, X., and Zhao, J. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the ACL*, 50–59. ACL.

Hildebrandt, M.; Sunder, S. S.; Mogoreanu, S.; Thon, I.; Tresp, V.; and Runkler, T. 2018. Configuration of industrial automation solutions using multi-relational recommender systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 271–287. Springer.

Hildebrandt, M.; Sunder, S. S.; Mogoreanu, S.; Joblin, M.; Mehta, A.; Thon, I.; and Tresp, V. 2019. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *European Semantic Web Conference*, 179–193. Springer.

Himmelstein, D. S., and Baranzini, S. E. 2015. Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS computational biology* 11(7):e1004259.

Himmelstein, D. S.; Lizee, A.; Hessler, C.; Brueggeman, L.; Chen, S. L.; Hadley, D.; Green, A.; Khankhanian, P.; and Baranzini, S. E. 2017. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife* 6:e26726.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Irving, G.; Christiano, P.; and Amodei, D. 2018. Ai safety via debate. *arXiv preprint arXiv:1805.00899*.

Kazemi, S. M., and Poole, D. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, 4284–4295.

Lao, N.; Mitchell, T.; and Cohen, W. W. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 529–539. ACL.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Lukovnikov, D.; Fischer, A.; Lehmann, J.; and Auer, S. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, 1211–1220. International World Wide Web Conferences Steering Committee.

Montavon, G.; Lapuschkin, S.; Binder, A.; Samek, W.; and Müller, K.-R. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition* 65:211–222.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, volume 11, 809–816.

Singhal, A. 2012. Introducing the knowledge graph: things, not strings. [Online; accessed 28-March-2019].

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, 926–934.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706. ACM.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1499–1509.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, volume 48, 2071–2080.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. ACL.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.