Learning Reasoning Strategies in End-to-End Differentiable Proving

Pasquale Minervini [1] **Sebastian Riedel** [1 2] **Pontus Stenetorp** [1] **Edward Grefenstette** [1 2] **Tim Rocktäschel** [1 2]

Abstract

Attempts to render deep learning models interpretable, data-efficient, and robust have seen some success through hybridisation with rule-based systems like Neural Theorem Provers (NTPs). These neuro-symbolic reasoning models can induce interpretable rules and learn representations from data via back-propagation, while providing logical explanations for their predictions. However, they are restricted by their computational complexity, as they need to consider all possible proof paths for explaining a goal, thus rendering them unfit for large-scale applications. We present Conditional Theorem Provers (CTPs), an extension to NTPs that learns an optimal rule selection strategy via gradient-based optimisation. We show that CTPs are scalable and yield state-of-the-art results on the CLUTRR dataset, which tests *systematic generalisation* of neural models by learning to reason over smaller graphs, and evaluating on larger ones. Finally, CTPs show better link prediction results on standard benchmarks in comparison with other neuro-symbolic reasoning models, while retaining their explainability properties.

1. Introduction

Neural Natural Language Understanding (NLU) systems—wherein a deep neural network is used as a function approximator (LeCun et al., 2015; Goodfellow et al., 2016)—have been extremely successful at various natural language tasks tasks, such as Question Answering (QA) and Natural Language Inference (NLI) (Goldberg, 2017), achieving strong generalisation results on the datasets available for these tasks (Seo et al., 2017; Hu et al., 2018; Shen et al., 2016; Huang et al., 2018). Such groundbreaking results were even more evident with the advent of large models pre-trained via self-supervision, such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), and RoBERTa (Liu et al., 2019).

Generalisation in Neural Models However, there are growing concerns about the ability of NLU systems – and neural networks more generally – to generalise in a systematic and robust way (Bahdanau et al., 2019; Lake & Baroni, 2018; Johnson et al., 2017; Sinha et al., 2019). For instance, Jia & Liang (2017) highlight the brittleness of NLU systems to adversarial examples, while Gururangan et al. (2018); Kaushik & Lipton (2018) show that neural NLU models tend to leverage annotation artefacts and spurious correlations in the data. Furthermore, analysing and supervising the inner workings of such models is not trivial, due to their inherent black-box character (Doshi-Velez & Kim, 2017; Lipton, 2018). More generally, Garnelo & Shanahan (2019) emphasise several limitations of neural models, in terms of *i*) data inefficiency and high sample complexity – the need of high volumes of training data in order to be effective, *ii*) poor generalisation – modern neural models may not produce the correct predictions when exposed to data outside the training distribution, and *iii*) lack of interpretability – such models are *black boxes* where internal representations and computations are hardly interpretable by humans.

In this respect, Sinha et al. (2019) systematically measured and compared the systematic generalisation abilities of several neural models (including very strong baselines such as BERT (Devlin et al., 2019) and Graph Attention Networks (GATs) (Velickovic et al., 2018)) on the task of answering questions about family relationship graphs, by evaluating on held-out combinations of reasoning patterns and by adding curated distracting noise facts. Interestingly they found that, for every model in their pool, performance degrades monotonically as they increase the complexity of the relational graph – highlighting the challenge of systematic generalisation (Lake & Baroni, 2018; Sodhani et al., 2018).

Neuro-Symbolic Reasoning A promising strategy for overcoming these issues consists in combining *neural models* and *symbolic reasoning* – given their complementary strengths and weaknesses (d'Avila Garcez et al., 2015; Evans & Grefenstette, 2018; Garnelo & Shanahan, 2019). We focus on NTPs (Rocktäschel & Riedel, 2017), a family of neuro-symbolic reasoning models. NTPs have interesting properties: they can jointly learn representations and interpretable rules from data via backpropagation, and can potentially combine such rules in ways that may have not been observed during training. A major limitation in NTPs is that, during inference, they need to consider *all rules* for explaining a given goal or sub-goal. This quickly renders them ineffective in settings requiring a large number of rules or reasoning steps.

Conditional Theorem Provers For solving such limitations, we propose CTPs, an extension of NTPs that is able to learn a strategy for selecting subsets of rules to consider at each step of the reasoning process. This is achieved by a `select` module that, given a goal, produce the rules needed for proving it. Predicates and constants in the produced rules lie in a continuous embedding space – for such a reason, the `select` module is end-to-end differentiable, and can be trained jointly with the other modules via gradient-based optimisation.

2. End-to-End Differentiable Proving

NTPs (Rocktäschel & Riedel, 2017) are a continuous relaxation of the *backward chaining algorithm* (Russell & Norvig, 2010). This algorithm works backward from the goal, chaining through rules to find known facts supporting the proof. Given a *query* (or *goal*) G, backward chaining first attempts at unifying it with the facts available in the Knowledge Base (KB). If no matching fact is available, it considers all rules $\mathrm{H} \coloneq \mathrm{B}$, where H denotes the *head* (or *consequence*) and B the *body* (or *premise*), where H can be unified with the query G, resulting in a substitution for the variables contained in H. Then, the backward chaining algorithm applies the substitution to the body B, and recursively attempts at proving the atoms contained therein. Backward chaining can be seen as a type of `and`/`or` search: `or` because the goal can be proven by any rule in the KB, and `and` because all the conjuncts in the premise of a rule must be proven.

Example 2.1 (Backward Chaining)**.** Consider a KB composed by the facts $\mathtt{p}(\text{RICK}, \text{BETH})$ and $\mathtt{p}(\text{BETH}, \text{MORTY})$, and by the rule $\mathtt{g}(\mathrm{X}, \mathrm{Y}) \coloneq \mathtt{p}(\mathrm{X}, \mathrm{Z}), \mathtt{p}(\mathrm{Z}, \mathrm{Y})$, where $\mathtt{p}$ and $\mathtt{g}$ denote the relationships *parent* and *grandparent*, respectively. The goal $G = \mathtt{g}(\text{RICK}, \text{MORTY})$ can be proven by unifying G with the head of the rule $\mathtt{g}(\mathrm{X}, \mathrm{Y})$, with the substitution $\{\mathrm{X}/\text{RICK}, \mathrm{Y}/\text{MORTY}\}$, and then by recursively proving the subgoals $\mathtt{p}(\text{RICK}, \mathrm{Z}), \mathtt{p}(\mathrm{Z}, \text{MORTY})$, which hold true for the substitution $\{\mathrm{Z}/\text{BETH}\}$. $\triangle$

NTPs make this reasoning process more flexible, by replacing the comparison between symbols with an end-to-end differentiable comparison between their embedding representations. NTPs recursively build a neural network enumerating all the possible proof paths for proving a query on a given KB, and aggregate all their proof scores via max pooling. They do so by relying on three modules – an *unification module*, which compares sub-symbolic representations of logic atoms, and mutually recursive `or` and `and` modules, which jointly enumerate all possible proof paths, before the final aggregation selects the highest scoring one. The whole process is outlined in Algorithm 1.

Example 2.2 (Neural Theorem Provers)**.** Consider a variant of Example 2.1, where each predicate and constant lives in a continuous embedding space, *i.e.* $\boldsymbol{\theta}_{\mathtt{p}:}, \boldsymbol{\theta}_{\mathtt{q}:}, \boldsymbol{\theta}_{\text{RICK}:}, \boldsymbol{\theta}_{\text{BETH}:}, \boldsymbol{\theta}_{\text{MORTY}:} \in \mathbb{R}^k$. Rocktäschel & Riedel (2017) propose replacing comparisons between symbols with a differentiable similarity measure $K : \mathbb{R}^k \times \mathbb{R}^k \to [0, 1]$, such as a Gaussian kernel, between their embeddings. Their model enumerates all possible proofs for a goal G, and generates a *proof score* for each of them, given by the *minimum* of all embedding similarities. For instance, if $G = \mathtt{grandPa}(\text{RICK}, \text{MORTY}) = [\mathtt{grandPa}, \text{RICK}, \text{MORTY}]$, one candidate proof consists in using the facts $F = \mathtt{p}(\text{RICK}, \text{BETH})$ and $F' = \mathtt{p}(\text{BETH}, \text{MORTY})$ and the rule $\mathtt{g}(\mathrm{X}, \mathrm{Y}) \coloneq \mathtt{p}(\mathrm{X}, \mathrm{Z}), \mathtt{p}(\mathrm{Z}, \mathrm{Y})$ from the KB, yielding the score $K(\boldsymbol{\theta}_{\mathtt{grandPa}}, \boldsymbol{\theta}_{\mathtt{g}})$. It is important to mention that NTPs allow unifying symbols like `grandPa` and `g` (which, in this case, share the same semantics), even though they are lexically different. The score for G is given by the *maximum* of all proof scores. $\triangle$

3. Conditional Proving Strategies

The NTPs proposed by Rocktäschel & Riedel (2017) use a fixed set of rules – either specified by the user or learned from data. In this model, given a goal, there is no *hard* decision mechanism for deciding which rules can be used for reformulating this goal into subgoals: all rules in the KB need to be considered when proving each goal. For this reason, NTPs were shown not to scale to large datasets (Rocktäschel & Riedel, 2017). Furthermore, selecting which rules to consider for each goal is a discrete decision step, which cannot be trivially solved using gradient-based optimisation.

Differentiable Goal Reformulation In order to *learn* which rule to consider at each step, we propose the following solution. Rather than relying on a fixed, potentially very large set of rules, we propose dynamically *generating* a minimal set of rules via a neural network architecture conditioned on the goal to prove.

Example 3.1 (Conditional Theorem Proving)**.** Assume that the goal to prove is $G = \mathtt{g}(\text{RICK}, \text{MORTY})$, we want the model to be able to only consider the best rules for proving G, such as $\mathtt{g}(\mathrm{X}, \mathrm{Y}) \coloneq \mathtt{p}(\mathrm{X}, \mathrm{Z}), \mathtt{p}(\mathrm{Z}, \mathrm{Y})$, rather than all rules in the KB. Remember that, in NTPs, relations and constants in the KB are represented by their embedding vectors, and the aforementioned rule selection can be implemented via a mapping from $\boldsymbol{\theta}_{\mathtt{g}:}$ to $[\boldsymbol{\theta}_{\mathtt{p}:}, \boldsymbol{\theta}_{\mathtt{p}:}]$. $\triangle$

Consider the `or` module in Algorithm 1. Selecting which rule to use during the proving process for a given goal G can be implemented by rewriting the `or` module as in Algorithm 2, where the set of clauses is produced by a module that, given the goal, generates a set of rules for proving G.

The core difference between NTPs and CTPs is that, when proving a goal G, rather than iterating through a possibly

Algorithm 1 Overview of the neural backward chaining algorithm proposed by Rocktäschel & Riedel (2017) – intuitively, it recursively proves each goal with all rules in the KB (OR module) and, for each rule, it proves its premise (AND module), up to d recursion steps

```
1: function or($G$, $d$, $S$)
2:   for H :– $\mathbb{B} \in \mathcal{K}$ do
3:     for $S \in$ and($\mathbb{B}, d,$ unify(H, G, $S$)) do
4:       yield $S$

1: function and($\mathbb{B}$, $d$, $S$)
2:   if $\mathbb{B} = []$ or $d = 0$ then yield $S$ else
3:   for $S' \in$ or(sub($\mathbb{B}_0, S_\psi$), $d-1, S$) do
4:     for $S'' \in$ and($\mathbb{B}_{1:}, d, S'$) do
5:       yield $S''$

1: function unify(H, G, $S = (S_\psi, S_\rho)$)
2:   $S'_\psi = S_\psi \bigcup_i T_i$
       with $T_i = \begin{cases} \{H_i/G_i\} & \text{if } H_i \in \mathcal{V} \\ \{G_i/H_i\} & \text{if } G_i \in \mathcal{V}, H_i \notin \mathcal{V} \\ \varnothing & \text{otherwise} \end{cases}$
3:   $S'_\rho = \min \{S_\rho\} \bigcup_{H_i, G_i \notin \mathcal{V}} \{K(\boldsymbol{\theta}_{H_i}, \boldsymbol{\theta}_{G_i})\}$
4:   return $(S'_\psi, S'_\rho)$
```

Algorithm 2 In Conditional Theorem Provers, the set of rules conditioned on the goal G.

```
1: function or($G$, $d$, $S$)
2:   for H :– $\mathbb{B} \in$ select$_\theta(G)$ do
3:     for $S \in$ and($\mathbb{B}, d,$ unify(H, G, $S$)) do
4:       yield $S$
```

very large set of clauses in the KB $\mathcal{K}$ (see Line 2 in the `or` module definition in Algorithm 1), the conditional `or` module in Algorithm 2 only iterates through a small set of generated clauses, whose generation is conditioned on G (see Line 2 in Algorithm 2). The `select` module is an end-to-end differentiable module with parameters θ that, given the goal G, produces a set of clauses.

Note that selecting the sub-goals to prove given a goal can be implemented by an end-to-end differentiable parametrised function $f(\cdot)$ that, given a goal G, produces a finite sequence of corresponding subgoals:

$$\texttt{select}_\theta(G) : \mathcal{A} \to [\mathcal{A} :\!\!- \mathcal{A}^*], \tag{1}$$

where $\mathcal{V}$ is a set of variables, and $\mathcal{A} \triangleq \mathbb{R}^k \times (\mathbb{R}^k \cup \mathcal{V}) \times (\mathbb{R}^k \cup \mathcal{V})$ denotes the embedding representation of a goal, such as $\texttt{g}(\textsc{rick}, \textsc{morty})$.

For instance, the `select` module in Eq. 1 can be implemented by a neural network that, given a goal such as $G = [\boldsymbol{\theta}_{\texttt{g}:}, \boldsymbol{\theta}_{\textsc{rick}:}, \boldsymbol{\theta}_{\textsc{morty}:}]$, generates H :– B with $\text{H} = [\boldsymbol{\theta}_{g:}, \text{X}, \text{Y}]$ and $\text{B} = [[\boldsymbol{\theta}_{p:}, \text{X}, \text{Z}], [\boldsymbol{\theta}_{p:}, \text{Z}, \text{Y}]]$, corresponding to the symbolic rule $\texttt{g}(\text{X}, \text{Y}) :\!\!- \texttt{p}(\text{X}, \text{Z}), \texttt{p}(\text{Z}, \text{Y})$. If the positions of the variables in the rule are fixed, the whole module is end-to-end differentiable with respect to its parameters θ.

Neural Goal Reformulation Here we define $\texttt{select}_\theta$ as a function of the goal predicate:

$$\texttt{select}_\theta(G) \triangleq [F_{\text{H}}(G) :\!\!- F_{\text{B}_1}(G), F_{\text{B}_2}(G)], \tag{2}$$

where the head and body of the resulting rule are given by $F_{\text{H}}(G) = [f_{\text{H}}(\boldsymbol{\theta}_{G_1}), \text{X}, \text{Y}]$, $F_{\text{B}_1}(G) = [f_{\text{B}_1}(\boldsymbol{\theta}_{G_1}), \text{X}, \text{Z}]$, and $F_{\text{B}_2}(G) = [f_{\text{B}_2}(\boldsymbol{\theta}_{G_1}), \text{Z}, \text{Y}]$. Here, each $f_i : \mathbb{R}^k \to \mathbb{R}^k$ is a differentiable function, such as the linear projection $f_i(\mathbf{x}) = \mathbf{W}_i \mathbf{x} + \mathbf{b}$, with $\mathbf{W}_i \in \mathbb{R}^{k \times k}$ and $\mathbf{b} \in \mathbb{R}^k$. In this way, instead of iterating through a possibly very large set of rules in the KB $\mathcal{K}$, we can generate a significantly smaller set of rules, whose generation is conditioned on the goal G and can be trained end-to-end on downstream reasoning tasks.

Attentive Goal Reformulation We can incorporate a useful prior in the `select` module architecture—namely that predicate symbols in the rule already exist in the KB, among the available relations $\mathcal{R}$. A method for incorporating this prior consists in using the given goal G for generating a distribution over the set of relations $\mathcal{R}$:

$$\begin{aligned} f_i(\mathbf{x}) &= \boldsymbol{\alpha}\, \mathbf{E}_{\mathcal{R}} \\ \boldsymbol{\alpha} &= \text{softmax}(\mathbf{W}_i \mathbf{x}) \in \Delta^{|\mathcal{R}|-1}, \end{aligned} \tag{3}$$

where $\mathbf{E}_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times k}$ denotes the predicate embedding matrix, $\mathbf{W}_i \in \mathbb{R}^{k \times |\mathcal{R}|}$, and $\boldsymbol{\alpha}$ is an attention distribution $\boldsymbol{\alpha} \in \Delta^{|\mathcal{R}|-1}$ over the predicates in $\mathcal{R}$, where Δ^n denotes the standard n-simplex [1]. This is especially helpful if the embedding size is larger than the number of relationships, *i.e.* $k \gg |\mathcal{R}|$.

Memory-Based Goal Reformulation A problem with a black-box neural network for reformulating the goal into sub-goals is that it can be difficult to inspect the rules by analysing the model parameters, as in Rocktäschel & Riedel (2017). For such a reason, we propose an alternative goal reformulation module, where rules are stored in a differentiable memory. More precisely, n rules are stored as memory matrices $\mathbf{M}_\cdot$, where each $\mathbf{M}_i \in \mathbb{R}^{n \times k}$ denotes the k-dimensional embeddings of the i-th predicates in the n rules. Then, the goal G is used to compute an attention distribution $\boldsymbol{\alpha} \in \Delta^{n-1}$, where each $\boldsymbol{\alpha}_i$ denotes the attention

[1]The standard n-simplex Δ^n is defined as $\Delta^n = \{(\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} \boldsymbol{\alpha}_i = 1 \land \forall i : \boldsymbol{\alpha}_i \geq 0\}$

weight on the i-th rule. This model can be formalised as follows:

$$\begin{aligned} f_i(\mathbf{x}) &= \boldsymbol{\alpha}\,\mathbf{M}_i \\ \boldsymbol{\alpha} &= \mathrm{softmax}\left(\mathbf{W}\mathbf{x}\right) \in \Delta^{n-1}, \end{aligned} \tag{4}$$

where $f : \mathbb{R}^k \rightarrow \mathbb{R}^k$ is a differentiable function that, given the goal, is used for producing an attention distribution $\boldsymbol{\alpha} \in \Delta^{n-1}$ over the rules and uses it for indexing a memory $\mathbf{M}$, analogous to a key-value memory network (Miller et al., 2016).

4. Related Work

Memory-Augmented Networks Memory-augmented neural architectures aim at improving the generalisation and reasoning abilities in neural networks by disentangling representations from computations. By enriching neural networks with a differentiable *external memory*, these models were able to multi-hop reason over texts (Sukhbaatar et al., 2015), induce algorithmic behaviours (Graves et al., 2014; Joulin & Mikolov, 2015; Grefenstette et al., 2015; Kaiser & Sutskever, 2016), and rapidly assimilate new data (Santoro et al., 2016).

Neuro-Symbolic Models *Differentiable interpreters* enable translating declarative or procedural knowledge into neural networks exhibiting strong inductive biases of that knowledge (Bošnjak et al., 2017; Rocktäschel & Riedel, 2017; Evans & Grefenstette, 2018). Bošnjak et al. (2017) propose $\partial 4$, a differentiable abstract machine for the Forth programming language.

Rocktäschel & Riedel (2017) propose a differentiable implementation for the backward chaining algorithm, effectively implementing a differentiable Datalog interpreter. Evans & Grefenstette (2018) propose a differentiable forward-chaining reasoning process, while Donadello et al. (2017) propose a continuous generalisation of the semantics of first-order logic. Yang et al. (2017) and Sadeghian et al. (2019) propose an approach for learning function-free Datalog clauses from KBs by means of a differentiable graph traversal operator, while Das et al. (2018) propose learning policies for navigating a KB via reinforcement learning.

A major problem with these approaches is their computational complexity, which renders them unusable for larger-scale learning problems. In order to address this issue, Minervini et al. (2019) propose Greedy NTPs (GNTPs), an extension to NTPs where, for each goal, only the top-k facts and rules are considered during the differentiable reasoning process.

Neural Module Networks Andreas et al. (2016) introduce Neural Module Networks (NMNs), an end-to-end differentiable composition of jointly trained neural modules. Analogously, NTPs can be seen as a recursive differentiable composition of `or` and `and` modules, jointly trained on downstream reasoning tasks. NMNs allow defining and training end-to-end differentiable composable models, and interpret and execute their compositions as simple programs. This is especially useful when dealing with reasoning tasks from visual and natural language inputs, such as reasoning over text with arithmetic modules (Gupta et al., 2019) or visual question answering (Andreas et al., 2016). Interestingly, in this work the structure of the composition is statically drawn from the data, while Jiang & Bansal (2019) propose a way of learning the model composition via a coordination model (Jiang & Bansal, 2019).

Incorporating Knowledge via Regularisation Another branch of works uses symbolic background knowledge to learn better representations for entities and relationships in a KB. An early work in this space is Rocktäschel et al. (2015), which regularise a relation extraction model by penalising inconsistency with respect to a set of logical constraints and sampled entities. Minervini et al. (2017a) regularise relation representations to incorporate equivalence and inversion axioms for a set of neural link prediction models, while Demeester et al. (2016) focus on simple implication axioms.

Minervini et al. (2017b) propose adversarially regularising neural models by identifying, during training, inputs that violate a given set of constraints, and regularising the model to decrease the degree of such violations. Xu et al. (2018) propose a similar idea, by using a *semantic loss* measuring to which extent a model matches a set of given constraints.

5. Experiments

In our experiments, we evaluated CTPs on two datasets, namely systematic generalisation on the CLUTRRs dataset, and link prediction in Knowledge Graphs (KGs). Datasets are introduced in Section 5.1, while baselines are described in Section 5.2.

5.1. Datasets and Tasks

Systematic Generalisation CLUTRR – Compositional Language Understanding and Text-based Relational Reasoning (Sinha et al., 2019) – contains a large set of graphs modelling hypothetical family relationships. Given a set of family relations, encoded as a graph with a variable number of nodes and edges, the goal is to infer the relationship between two family members, whose relationship is not explicitly mentioned. To solve this task, a learning agent should be able to induce the logical rules governing the kinship relationships, such as the *parent* of a *parent* is a *grandparent*, and use these rules to infer the relationship between a given pair of entities.

CLUTRR allows testing a learning agent's ability for *systematic generalisation*, by testing on graphs containing com-

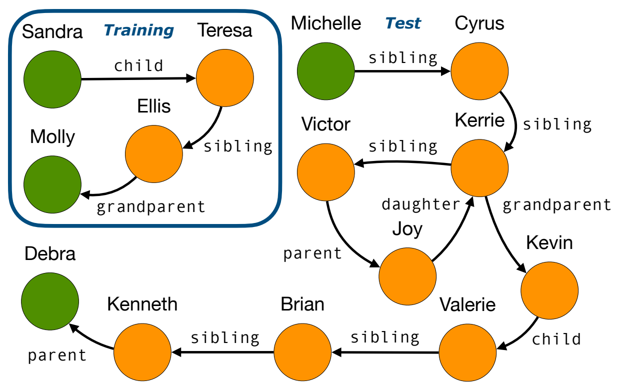


Figure 1. Example of a train and test instance in CLUTRR – the training instance (upper left) is composed by a graph with three edges, while the test instance is composed by ten edges; the task consists in identifying the relationships between the green nodes.

CLUTRR – Sample of Learned Rules
$\texttt{child}(X, Y) \Leftarrow \texttt{child}(X, Z), \texttt{sibling}(Z, Y)$ $\texttt{child}(X, Y) \Leftarrow \texttt{SO}(X, Z), \texttt{child}(Z, Y)$
$\texttt{grand}(X, Y) \Leftarrow \texttt{child}(X, Z), \texttt{child}(Z, Y)$ $\texttt{grand}(X, Y) \Leftarrow \texttt{grand}(X, Z), \texttt{sibling}(Z, Y)$ $\texttt{grand}(X, Y) \Leftarrow \texttt{grand}(X, Z), \texttt{sibling}(Z, Y)$
$\texttt{in-law}(X, Y) \Leftarrow \texttt{child}(X, Z), \texttt{SO}(Z, Y)$ $\texttt{in-law}(X, Y) \Leftarrow \texttt{sibling-in-law}(X, Z), \texttt{child}(Z, Y)$
$\texttt{sibling}(X, Y) \Leftarrow \texttt{sibling}(X, Z), \texttt{sibling}(Z, Y)$ $\texttt{sibling}(X, Y) \Leftarrow \texttt{child}(X, Z), \texttt{uncle}(Y, Z)$ $\texttt{sibling}(X, Y) \Leftarrow \texttt{child}(X, Z), \texttt{child}(Y, Z)$

Figure 2. Rules learned on CLUTRR by CTPs – symbols were obtained by decoding the goal reformulations with the nearest predicate in embedding space.

binations of logical rules that were not seen during training.

Fig. 1 shows an example of a training instance and a test instance in CLUTRR: the training instances consists in a graph modelling a set of family relations of only three edges, while the test instance is composed by a graph with ten edges. In both cases, the task consists in inferring the relationships between two of the nodes in the graph.

Link Prediction Furthermore, we evaluate CTPs on neural link prediction tasks following the same evaluation protocols as Rocktäschel & Riedel (2017), on the Countries (Bouchard et al., 2015), Nations, UMLS, and Kinship (Kemp et al., 2006) datasets. The Countries dataset contains countries, regions, and sub-regions as entities, and it is carefully designed to test the logical reasoning and learning capabilities of neural link prediction models: queries have the form $\texttt{locatedIn}(\texttt{c}, \cdot)$, where the answers are regions. This dataset comes with three tasks – $S1$, $S2$, and $S3$ – each requiring reasoning skills of increasing complexity; we refer to Rocktäschel & Riedel (2017) for more details about this dataset. The Unified Medical Language System (UMLS) dataset is from bio-medicine: entities are biomedical concepts , and relations include treats and diagnoses. The Kinship dataset contains kinship relationships among members of the Alyawarra tribe from Central Australia.

5.2. Models and Baselines

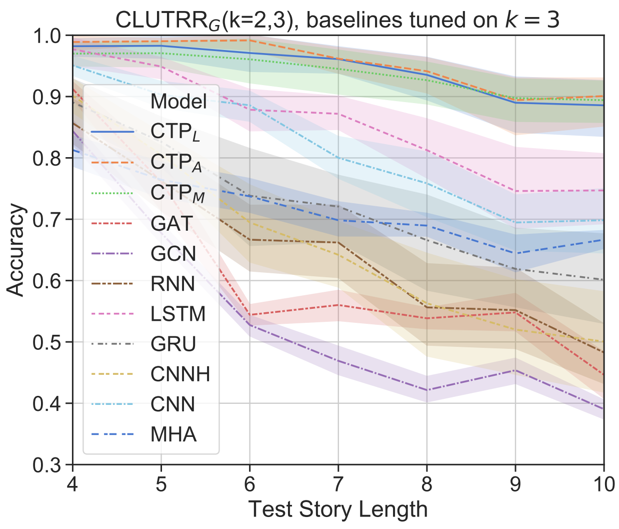
Models We consider the following three CTP model variants: *i*) CTP$_L$, where the mappings f_i from goal predicates to rule predicates are implemented via a linear projection, *ii*) CTP$_A$, where it is implemented via *attentive goal reformulation*, as described in Section 3, and *iii*) CTP$_M$, where it is implemented via *memory-based goal reformulation*, also described in Section 3.

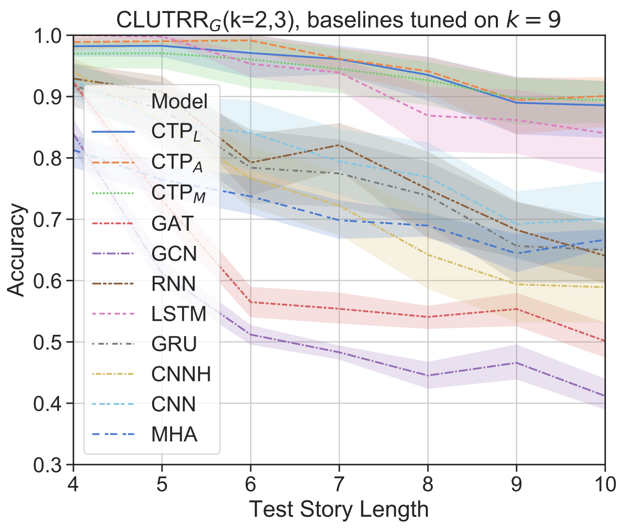
Baselines In our experiments, we consider two classes of baselines – *graph-based* and *sequence-based*. Graph-based baselines consist in neural architectures specifically designed for graph data and: for our experiments, we considered GNTPs, a neuro-symbolic reasoning model, and two Graph Neural Network (GNN) architectures, namely GATs (Velickovic et al., 2018) and Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017).

Sequence-based baselines are neural architectures originally conceived for encoding sequences: by linearising the relational graphs in sequences of subject-predicate-object triples, we can use such models for encoding graphs. We consider several sequence encoding models, namely Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs) (Hochreiter & Schmidhuber, 1997), Gated Recurrent Units (GRUs) (Cho et al., 2014), Convolutional Neural Networks (CNNs) (Kim, 2014), CNN with Highway Encoders (CNNHs) (Kim et al., 2016), Multi-Headed Attention Networks (MHAs) (Vaswani et al., 2017).

Encoding For both graph-based and sequence-based baselines, we considered two approaches: *i*) encoding the the KB and goal independently, as in Sinha et al. (2019), and *ii*) conditioning the KB encoding on the goal. Let $\text{enc}_{\boldsymbol{\theta}_e}$ denote an encoder that, given a set of ground facts (such as a KB or a goal), produces a continuous k-dimensional representation, and $\hat{y}$ denote a conditional distribution over the candidate relationship types. The encoder in item (i), where the goal G and the KB $\mathcal{K}$ are encoded independently, can be summarised as $\hat{y} = \text{softmax}(\mathbf{W}\,[\text{enc}_{\boldsymbol{\theta}_e}(\mathcal{K}); \text{enc}_{\boldsymbol{\theta}_e}(G)])$. The encoder in item (ii), where G and $\mathcal{K}$ are encoded jointly, can be summarised as $\hat{y} = \text{softmax}(\mathbf{W}\,\text{enc}_{\boldsymbol{\theta}_e}([G; \mathcal{K}]))$.

For model selection, we generated a CLUTRR-like dataset using the code published by Sinha et al. (2019) composed by a training set of graphs with $\{2, 3\}$ edges, and two validation sets, one with graphs with three edges, and another with





	4 Hops	5 Hops	6 Hops	7 Hops	8 Hops	9 Hops	10 Hops
CTP$_L$	$.98 \pm .02$	$.98 \pm .03$	$.97 \pm .05$	$.96 \pm .04$	$.94 \pm .05$	$.89 \pm .07$	$.89 \pm .07$
CTP$_A$	$\mathbf{.99 \pm .02}$	$\mathbf{.99 \pm .01}$	$\mathbf{.99 \pm .02}$	$\mathbf{.96 \pm .04}$	$\mathbf{.94 \pm .05}$	$.89 \pm .08$	$\mathbf{.90 \pm .07}$
CTP$_M$	$.97 \pm .03$	$.97 \pm .03$	$.96 \pm .06$	$.95 \pm .06$	$.93 \pm .06$	$\mathbf{.90 \pm .06}$	$.89 \pm .06$
GNTP	$.49 \pm .18$ ▽	$.45 \pm .21$ ▽	$.38 \pm .23$ ▽	$.37 \pm .21$ ▽	$.32 \pm .20$ ▽	$.31 \pm .19$ ▽	$.31 \pm .22$ ▽
GAT	$.91 \pm .02$ ▼	$.76 \pm .06$ ▼	$.54 \pm .03$ ▼	$.56 \pm .04$ ▼	$.54 \pm .03$ ▼	$.55 \pm .05$ ▼	$.45 \pm .06$ ▼
GCN	$.84 \pm .03$ ▼	$.68 \pm .02$ ▼	$.53 \pm .03$ ▼	$.47 \pm .04$ ▼	$.42 \pm .03$ ▼	$.45 \pm .03$ ▼	$.39 \pm .02$ ▼
RNN	$.86 \pm .06$ ▽	$.76 \pm .08$ ▼	$.67 \pm .08$ ▼	$.66 \pm .08$ ▼	$.56 \pm .10$ ▼	$.55 \pm .10$ ▼	$.48 \pm .07$ ▼
LSTM	$.98 \pm .04$	$.95 \pm .03$	$.88 \pm .05$ ▽	$.87 \pm .04$ ▽	$.81 \pm .07$ ▽	$.75 \pm .10$ ▽	$.75 \pm .09$ ▽
GRU	$.89 \pm .05$ ▽	$.83 \pm .06$ ▼	$.74 \pm .12$ ▽	$.72 \pm .09$ ▼	$.67 \pm .12$ ▽	$.62 \pm .10$ ▼	$.60 \pm .12$ ▼
CNNH	$.90 \pm .04$ ▽	$.81 \pm .05$ ▼	$.69 \pm .10$ ▼	$.64 \pm .08$ ▼	$.56 \pm .13$ ▼	$.52 \pm .12$ ▼	$.50 \pm .12$ ▼
CNN	$.95 \pm .02$	$.90 \pm .03$ ▼	$.89 \pm .04$ ▽	$.80 \pm .05$ ▼	$.76 \pm .08$ ▽	$.69 \pm .07$ ▼	$.70 \pm .08$ ▽
MHA	$.81 \pm .04$ ▼	$.76 \pm .04$ ▼	$.74 \pm .05$ ▼	$.70 \pm .04$ ▼	$.69 \pm .03$ ▼	$.64 \pm .05$ ▼	$.67 \pm .02$ ▼

	4 Hops	5 Hops	6 Hops	7 Hops	8 Hops	9 Hops	10 Hops
CTP$_L$	$.98 \pm .02$	$.98 \pm .03$	$.97 \pm .05$	$.96 \pm .04$	$.94 \pm .05$	$.89 \pm .07$	$.89 \pm .07$
CTP$_A$	$.99 \pm .02$	$.99 \pm .01$	$\mathbf{.99 \pm .02}$	$\mathbf{.96 \pm .04}$	$\mathbf{.94 \pm .05}$	$.89 \pm .08$	$\mathbf{.90 \pm .07}$
CTP$_M$	$.97 \pm .03$	$.97 \pm .03$	$.96 \pm .06$	$.95 \pm .06$	$.93 \pm .06$	$\mathbf{.90 \pm .06}$	$.89 \pm .06$
GNTP	$.49 \pm .18$ ▽	$.45 \pm .21$ ▽	$.38 \pm .23$ ▽	$.37 \pm .21$ ▽	$.32 \pm .20$ ▽	$.31 \pm .19$ ▽	$.31 \pm .22$ ▽
GAT	$.92 \pm .01$ ▼	$.73 \pm .04$ ▼	$.56 \pm .04$ ▼	$.55 \pm .04$ ▼	$.54 \pm .03$ ▼	$.55 \pm .04$ ▼	$.50 \pm .04$ ▼
GCN	$.84 \pm .04$ ▼	$.61 \pm .03$ ▼	$.51 \pm .02$ ▼	$.48 \pm .02$ ▼	$.45 \pm .03$ ▼	$.47 \pm .05$ ▼	$.41 \pm .04$ ▼
RNN	$.93 \pm .03$ ▽	$.91 \pm .03$ ▽	$.79 \pm .08$ ▼	$.82 \pm .06$ ▼	$.75 \pm .11$ ▽	$.68 \pm .07$ ▼	$.64 \pm .07$ ▼
LSTM	$\mathbf{1.0 \pm .00}$	$\mathbf{1.0 \pm .00}$	$.95 \pm .03$	$.94 \pm .04$	$.87 \pm .08$	$.86 \pm .08$	$.84 \pm .09$
GRU	$.92 \pm .05$ ▽	$.88 \pm .06$ ▽	$.78 \pm .09$ ▽	$.77 \pm .09$ ▽	$.74 \pm .08$ ▼	$.66 \pm .10$ ▼	$.65 \pm .08$ ▼
CNNH	$.94 \pm .03$ ▽	$.86 \pm .06$ ▽	$.77 \pm .08$ ▼	$.72 \pm .08$ ▼	$.64 \pm .09$ ▼	$.59 \pm .10$ ▼	$.59 \pm .09$ ▼
CNN	$.93 \pm .04$ ▽	$.86 \pm .07$ ▽	$.84 \pm .09$ ▽	$.79 \pm .08$ ▽	$.77 \pm .10$ ▽	$.69 \pm .09$ ▽	$.70 \pm .11$ ▽
MHA	$.81 \pm .04$ ▼	$.76 \pm .04$ ▼	$.74 \pm .05$ ▼	$.70 \pm .04$ ▼	$.69 \pm .03$ ▼	$.64 \pm .05$ ▼	$.67 \pm .02$ ▼

Figure 3. Results on the CLUTRR dataset after training on stories of lengths $\{2, 3\}$ and evaluating on stories of length $\{4, 5, \ldots, 10\}$ – hyperparameters were fine-tuned on either short stories (left) and long stories (right). Significance testing was assessed via a unequal variances t-test in comparison with CTP$_L$: ▼ (resp. ▽) represents a p-value lower than 10^{-4} (resp. 10^{-2}).

graphs with nine edges. We then selected two sets of hyperparameters for each of the baselines: one that maximises the validation accuracy on graphs with three edges, and another that maximises the test accuracy on graphs with nine edges. All details on the hyperparameter selection process can be found in Appendix A.

To assess the statistical significance of our results, we ran each of the experiments 10 times, each time with a different seed, and compared the resulting accuracy values using an *unequal variances t-test*, or Welch's t-test [2].

5.3. Results

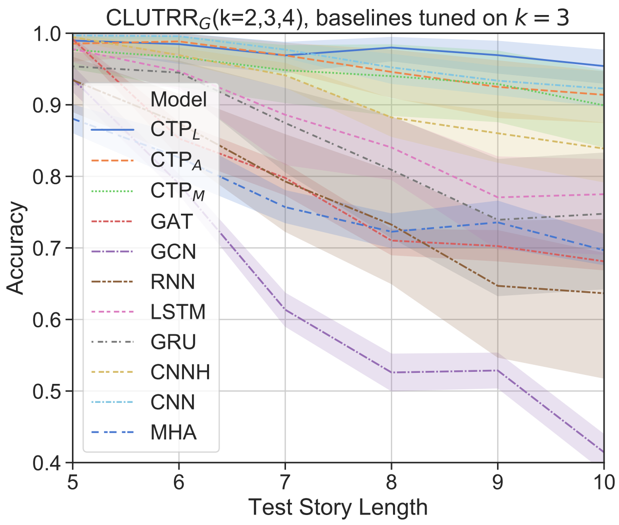
CLUTRR We evaluated three CTPs variants and all considered baselines on two datasets published by Sinha et al. (2019) under the identifiers `089907f8` and `db9b8f04` – we refer to such datasets as CLUTRR$_{\text{G}}(k = 2, 3)$ and CLUTRR$_{\text{G}}(k = 2, 3, 4)$, where k denotes the number of edges in the training graphs. Results for CLUTRR$_{\text{G}}(k = 2, 3)$ are summarised in Fig. 3, while results for CLUTRR$_{\text{G}}(k = 2, 3, 4)$ are summarised in Fig. 4.

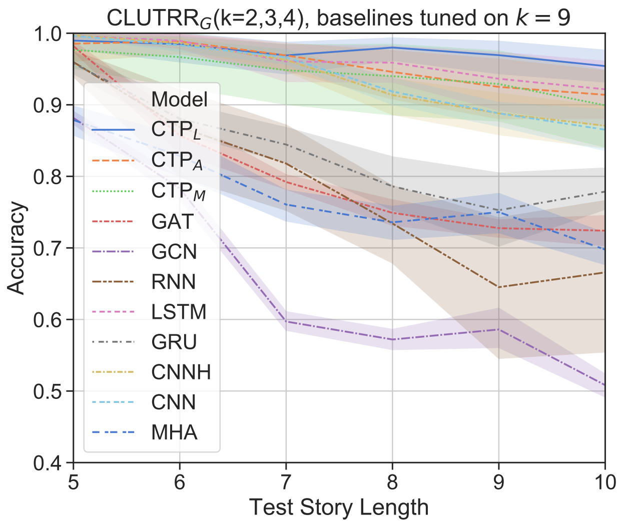
In Fig. 3 we observe that, after training on graphs with two and three edges, baseline models tend to be able to generalise correctly to slightly longer stories – such as graphs with four and five edges – but that predictive accuracy quickly decreases with increasing graph sizes and this phenomenon happens when tuning hyper-parameters either on graphs with three edges or on graph with nine edges.

In our experiments, LSTMs had a strikingly different behaviour in comparison with other baselines: for graphs with nine edges, the accuracy decrease caused by using the LSTMs baseline is only significant with $p \leq 10^{-2}$ (for all other baselines this change is significant with $p \leq 10^{-4}$), with a drop in significance for smaller graphs.

This phenomenon – LSTMs yielding surprisingly accurate results on the CLUTRR dataset – was observed across every

[2]We assume accuracy values to be Gaussian-distributed, as they approach a normal distribution for large numbers of re-runs, due to the Central Limit Theorem.





	5 Hops	6 Hops	7 Hops	8 Hops	9 Hops	10 Hops
CTP$_L$	$.99 \pm .02$	$.98 \pm .04$	$.97 \pm .04$	$\mathbf{.98 \pm .03}$	$\mathbf{.97 \pm .04}$	$\mathbf{.95 \pm .04}$
CTP$_A$	$.99 \pm .04$	$.99 \pm .03$	$.97 \pm .03$	$.95 \pm .06$	$.93 \pm .07$	$.91 \pm .05$
CTP$_M$	$.98 \pm .04$	$.97 \pm .06$	$.95 \pm .06$	$.94 \pm .08$	$.93 \pm .08$	$.90 \pm .09$
GNTP	$.68 \pm .28$	$.63 \pm .34$	$.62 \pm .31$	$.59 \pm .32$	$.57 \pm .34$	$.52 \pm .32$
GAT	$.99 \pm .00$	$.85 \pm .04$ ▼	$.80 \pm .03$ ▼	$.71 \pm .03$ ▼	$.70 \pm .03$ ▼	$.68 \pm .02$ ▼
GCN	$.94 \pm .03$ ▽	$.79 \pm .02$ ▼	$.61 \pm .03$ ▼	$.53 \pm .04$ ▼	$.53 \pm .04$ ▼	$.41 \pm .04$ ▼
RNN	$.93 \pm .06$	$.87 \pm .07$ ▽	$.79 \pm .11$ ▽	$.73 \pm .12$ ▽	$.65 \pm .16$ ▽	$.64 \pm .16$ ▽
LSTM	$.98 \pm .03$	$.95 \pm .04$	$.89 \pm .10$	$.84 \pm .07$ ▽	$.77 \pm .11$ ▽	$.78 \pm .11$ ▽
GRU	$.95 \pm .04$	$.94 \pm .03$	$.87 \pm .08$	$.81 \pm .13$ ▽	$.74 \pm .15$ ▽	$.75 \pm .15$ ▽
CNNH	$.99 \pm .01$	$.97 \pm .02$	$.94 \pm .03$	$.88 \pm .04$ ▼	$.86 \pm .05$ ▽	$.84 \pm .06$ ▽
CNN	$\mathbf{1.0 \pm .00}$	$\mathbf{1.0 \pm .01}$	$\mathbf{.98 \pm .01}$	$.95 \pm .03$	$.93 \pm .03$	$.92 \pm .04$
MHA	$.88 \pm .03$ ▼	$.83 \pm .05$ ▼	$.76 \pm .04$ ▼	$.72 \pm .04$ ▼	$.74 \pm .05$ ▼	$.70 \pm .03$ ▼

	5 Hops	6 Hops	7 Hops	8 Hops	9 Hops	10 Hops
CTP$_L$	$.99 \pm .02$	$.98 \pm .04$	$.97 \pm .04$	$\mathbf{.98 \pm .03}$	$\mathbf{.97 \pm .04}$	$\mathbf{.95 \pm .04}$
CTP$_A$	$.99 \pm .04$	$.99 \pm .03$	$\mathbf{.97 \pm .03}$	$.95 \pm .06$	$.93 \pm .07$	$.91 \pm .05$
CTP$_M$	$.98 \pm .04$	$.97 \pm .06$	$.95 \pm .06$	$.94 \pm .08$	$.93 \pm .08$	$.90 \pm .09$
GNTP	$.68 \pm .28$	$.63 \pm .34$	$.62 \pm .31$	$.59 \pm .32$	$.57 \pm .34$	$.52 \pm .32$
GAT	$.98 \pm .01$	$.86 \pm .04$ ▼	$.79 \pm .02$ ▼	$.75 \pm .03$ ▼	$.73 \pm .02$ ▼	$.72 \pm .03$ ▼
GCN	$.88 \pm .01$ ▼	$.78 \pm .02$ ▼	$.60 \pm .02$ ▼	$.57 \pm .02$ ▼	$.59 \pm .04$ ▼	$.51 \pm .02$ ▼
RNN	$.96 \pm .03$	$.87 \pm .09$ ▽	$.82 \pm .09$ ▽	$.73 \pm .09$ ▼	$.65 \pm .15$ ▽	$.67 \pm .16$ ▽
LSTM	$1.0 \pm .01$	$\mathbf{.99 \pm .02}$	$.96 \pm .04$	$.96 \pm .04$	$.94 \pm .06$	$.92 \pm .07$
GRU	$.96 \pm .02$ ▽	$.88 \pm .03$ ▼	$.84 \pm .04$ ▼	$.79 \pm .06$ ▼	$.75 \pm .08$ ▼	$.78 \pm .04$ ▼
CNNH	$\mathbf{1.0 \pm .00}$	$.99 \pm .01$	$.96 \pm .02$	$.91 \pm .04$ ▽	$.89 \pm .04$ ▽	$.87 \pm .04$ ▽
CNN	$1.0 \pm .00$	$.98 \pm .01$	$.97 \pm .02$	$.92 \pm .03$ ▽	$.89 \pm .03$ ▼	$.87 \pm .04$ ▽
MHA	$.88 \pm .03$ ▼	$.83 \pm .05$ ▼	$.76 \pm .03$ ▼	$.74 \pm .04$ ▼	$.75 \pm .04$ ▼	$.70 \pm .03$ ▼

Figure 4. Results on the CLUTRR dataset after training on stories of lengths $\{2, 3, 4\}$ and evaluating on stories of length $\{5, \ldots, 10\}$ – hyperparameters were fine-tuned on either short stories (left) and long stories (right). Significance testing was assessed via a unequal variances t-test in comparison with CTP$_L$: ▼ (resp. ▽) represents a p-value lower than 10^{-4} (resp. 10^{-2}).

experiment in our empirical evaluation, while other recurrent neural architectures such as RNNs and GRUs do not benefit from this property. A possible explanation for this phenomenon is that LSTMs encode an inductive bias that allows these architectures to better generalise to larger graphs, but explaining the behaviour of recurrent neural models is still an open problem.

Model Analysis A great feature of CTPs is that we can analyse the goal reformulation process to understand the reasoning process underlying a given prediction, and extract explainable rules. In Fig. 2 we show a sample of the rules and common-sense reasoning patterns learned by CTPs on the CLUTRR dataset.

We can see that, for example, CTPs successfully identify that *e.g. the child of a child is a grandchild*, *the child of one's significant other is also one's child*, and *the parent of a significant other is an in-law*.

Training Dynamics

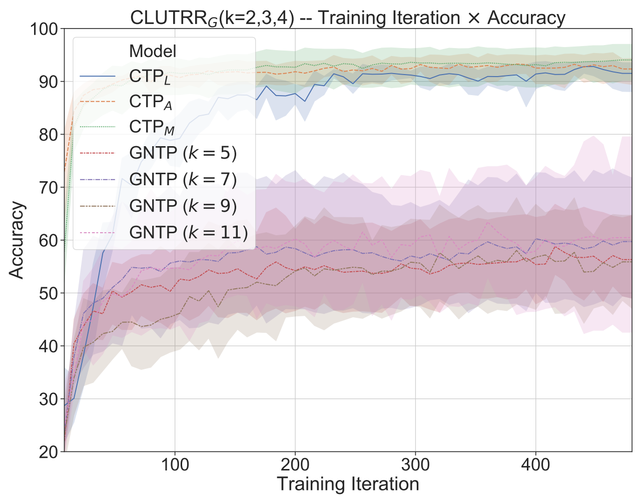


Figure 5. Training dynamics of CTPs and GNTPs.

Table 1. Comparison of CTPs, with GNTPs (Minervini et al., 2019), NeuralLP (Yang et al., 2017) and MINERVA (Das et al., 2018) (from Minervini et al. (2019)) on benchmark datasets: hyperparameters were selected based on the validation MRR, and we report the mean and standard deviation over 10 random seeds.

Datasets	Metrics	CTP	GNTP Standard	GNTP Attention	NeuralLP	MINERVA	Learned Rules
Countries $S1$	AUC-PR	**100.0 ± 0.00**	99.98 ± 0.05	**100.0 ± 0.00**	**100.0 ± 0.0**	**100.0 ± 0.0**	`locIn`(X,Y) :– `locIn`(X,Z), `locIn`(Z,Y)
Countries $S2$	AUC-PR	91.81 ± 1.07	90.82 ± 0.88	**93.48 ± 3.29**	75.1 ± 0.3	92.36 ± 2.41	`neighOf`(X,Y) :– `neighOf`(X,Z), `locIn`(Z,Y)
Countries $S3$	AUC-PR	94.78 ± 0.00	87.70 ± 4.79	91.27 ± 4.02	92.20 ± 0.2	**95.10 ± 1.20**	`neighOf`(X,Y) :– `neighOf`(Y,X)
Kinship	MRR	**0.764 ± 0.00**	0.719	0.759	0.619	0.720	`term0`(X, Y) :– `term22`(Y, X)
Kinship	Hits@1	**0.646 ± 0.01**	0.586	0.642	0.475	0.605	`term4`(X, Y) :– `term4`(Y, X)
Kinship	Hits@3	**0.859 ± 0.01**	0.815	0.850	0.707	0.812	`term20`(X,Y) :– `term24`(X, Z), `term6`(Z, Y)
Kinship	Hits@10	0.958 ± 0.00	0.958	**0.959**	0.912	0.924	`term2`(X,Y) :– `term4`(X, Z), `term7`(Z, Y)
Nations	MRR	**0.709 ± 0.03**	0.658	0.645	—	—	`tourism3`(X, Y) :– `eemigrants`(Y, X)
Nations	Hits@1	**0.562 ± 0.05**	0.493	0.490	—	—	`independence`(X,Y) :– `commonbloc0`(Y,X)
Nations	Hits@3	**0.813 ± 0.03**	0.781	0.736	—	—	`relngo`(X,Y) :– `timesinceally`(Y,X)
Nations	Hits@10	**0.995 ± 0.00**	0.985	0.975	—	—	`relstudents`(X, Y) :– `relexportbooks`(X, Y)
UMLS	MRR	0.852 ± 0.01	0.841	**0.857**	0.778	0.825	`isa`(X,Y) :– `isa`(X,Z), `isa`(Z,Y)
UMLS	Hits@1	0.752 ± 0.01	0.732	**0.761**	0.643	0.728	`resultOf`(X,Y) :– `resultOf`(X,Z), `resultOf`(Z,Y)
UMLS	Hits@3	**0.947** ± 0.01	0.941	**0.947**	0.869	0.900	`treats`(X, Y) :– `prevents`(X, Z), `resultOf`(Z, Y)
UMLS	Hits@10	0.984 ± 0.00	**0.986**	0.983	0.962	0.968	`uses`(X,Y) :– `produces`(X,Y)

We analyse the training dynamics of CTP$_L$, CTP$_A$, CTP$_M$, and GNTPs (Minervini et al., 2019) on CLUTRR$_G$(k=2,3,4). The CTP variants consisted of 5 goal reformulators, each implemented by an independent `select` module, while GNTP has a KB of 32 rules and $k \in \{5, 7, 9, 11\}$. For all models, the embedding size for entities and relations was set to 50.

In Fig. 5, we can see how the training accuracy of such models evolves during training. We can see that, while the three CTP variants get to a nearly-perfect training set accuracy in less than 300 iterations, GNTPs is unable to match this result, even after careful hyperparameter tuning. A possible explanation is that, in order to scale to large rule sets, GNTPs only considers the top-k rules, based on the similarity between the goal and the head of the rules. This is equivalent to an hard attention mask, which is known to be problematic to train via gradient-based optimisation (Luong et al., 2015).

Link Prediction In Table 1 we show link prediction results in comparison with three other neuro-symbolic reasoning methods, namely GNTPs (Minervini et al., 2019), NeuralLP (Yang et al., 2017) and MINERVA (Das et al., 2018). GNTPs are an extension of NTPs where rules are heuristically selected by search for the rules where the head predicate is closest to the sub-goal predicate in embedding space.

Our experiments show that CTPs produce significantly more accurate or very competitive link prediction results, while controlling the complexity of the reasoning process via the goal-conditioned rule selection. For instance, in the Nations dataset, only two rules were generated by CTPs for each goal, while in Rocktäschel & Riedel (2017) NTPs were required to iterate over sixty rules. Also in this case, CTPs were able to produce explanations for each of their predictions. For instance, in the Nations dataset, CTPs successfully extracted logical patterns such as $\texttt{commonbloc1}(X, Y) :\!- \texttt{relngo}(Y, X)$, $\texttt{timesincewar}(X, Y) :\!- \texttt{independence}(X, Y)$, $\texttt{unweightedunvote}(X, Y) :\!- \texttt{relngo}(X, Y)$, and $\texttt{ngo}(X, Y) :\!- \texttt{independence}(Y, X)$.

6. Conclusions

We introduced CTPs, an extension to NTPs for learning the optimal rule selection strategy via gradient-based optimisation: for each sub-goal, a `select` module produces a smaller set of rules, which is then used during the proving mechanism. Furthermore, we propose three variants of the rule selection mechanism, where the sub-goal reformulations are obtained by linear projections of the sub-goal predicate, attention distributions over predicate embeddings, and a key-value memory lookup over a set of rules.

We show that CTPs are scalable and yield state-of-the-art results on the CLUTRR dataset, which explicitly tests the systematic generalisation of neural models, in comparison with a wide set of neural baselines. Finally, we show that CTPs yield competitive results in standard link prediction benchmark in comparison with other neuro-symbolic approaches.

Acknowledgements This work was supported by the EU

Horizon 2020 Research and Innovation Programme under the grant 875160. We thank NVIDIA for GPU donations.

References

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *CVPR*, pp. 39–48. IEEE Computer Society, 2016.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. C. Systematic generalization: What is required and can it be learned? In *ICLR (Poster)*. OpenReview.net, 2019.

Bouchard, G., Singh, S., and Trouillon, T. On approximate reasoning capabilities of low-rank vector spaces. In *AAAI Spring Symposia*. AAAI Press, 2015.

Bošnjak, M., Rocktäschel, T., Naradowsky, J., and Riedel, S. Programming with a Differentiable Forth Interpreter. In *ICML*, volume 70, pp. 547–556. PMLR, 2017.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pp. 1724–1734. ACL, 2014.

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR (Poster)*. OpenReview.net, 2018.

d'Avila Garcez, A. S., Besold, T. R., Raedt, L. D., Földiák, P., Hitzler, P., Icard, T., Kühnberger, K., Lamb, L. C., Miikkulainen, R., and Silver, D. L. Neural-symbolic learning and reasoning: Contributions and challenges. In *AAAI Spring Symposia*. AAAI Press, 2015.

Demeester, T., Rocktäschel, T., and Riedel, S. Lifted rule injection for relation embeddings. In *EMNLP*, pp. 1389–1399. The Association for Computational Linguistics, 2016.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Donadello, I., Serafini, L., and d'Avila Garcez, A. S. Logic tensor networks for semantic image interpretation. In *IJCAI*, pp. 1596–1602. ijcai.org, 2017.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *CoRR*, abs/1702.08608, 2017.

Evans, R. and Grefenstette, E. Learning Explanatory Rules from Noisy Data. *JAIR*, 61:1–64, 2018.

Garnelo, M. and Shanahan, M. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17 – 23, 2019. ISSN 2352-1546. SI: 29: Artificial Intelligence (2019).

Goldberg, Y. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017.

Goodfellow, I. J., Bengio, Y., and Courville, A. C. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

Graves, A., Wayne, G., and Danihelka, I. Neural Turing Machines. *CoRR*, abs/1410.5401, 2014.

Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. Learning to Transduce with Unbounded Memory. In *NIPS*, pp. 1828–1836, 2015.

Gupta, N., Lin, K., Roth, D., Singh, S., and Gardner, M. Neural module networks for reasoning over text. *CoRR*, abs/1912.04971, 2019.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S. R., and Smith, N. A. Annotation artifacts in natural language inference data. In *NAACL-HLT (2)*, pp. 107–112. Association for Computational Linguistics, 2018.

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

Hu, M., Peng, Y., Huang, Z., Qiu, X., Wei, F., and Zhou, M. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*, pp. 4099–4106. ijcai.org, 2018.

Huang, H., Zhu, C., Shen, Y., and Chen, W. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *ICLR (Poster)*. OpenReview.net, 2018.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, pp. 2021–2031. Association for Computational Linguistics, 2017.

Jiang, Y. and Bansal, M. Self-assembling modular networks for interpretable multi-hop reasoning. In *EMNLP/IJCNLP (1)*, pp. 4473–4483. Association for Computational Linguistics, 2019.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, pp. 1988–1997. IEEE Computer Society, 2017.

Joulin, A. and Mikolov, T. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In *NIPS*, pp. 190–198, 2015.

Kaiser, L. and Sutskever, I. Neural GPUs Learn Algorithms. In *ICLR*, 2016.

Kaushik, D. and Lipton, Z. C. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *EMNLP*, pp. 5010–5015. Association for Computational Linguistics, 2018.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning Systems of Concepts with an Infinite Relational Model. In *AAAI*, pp. 381–388, 2006.

Kim, Y. Convolutional neural networks for sentence classification. In *EMNLP*, pp. 1746–1751. ACL, 2014.

Kim, Y., Jernite, Y., Sontag, D. A., and Rush, A. M. Character-aware neural language models. In *AAAI*, pp. 2741–2749. AAAI Press, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2879–2888. PMLR, 2018.

LeCun, Y., Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521(7553):436–444, 2015.

Lipton, Z. C. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *EMNLP*, pp. 1412–1421. The Association for Computational Linguistics, 2015.

Miller, A. H., Fisch, A., Dodge, J., Karimi, A., Bordes, A., and Weston, J. Key-value memory networks for directly reading documents. In *EMNLP*, pp. 1400–1409. The Association for Computational Linguistics, 2016.

Minervini, P., Costabello, L., Muñoz, E., Novácek, V., and Vandenbussche, P. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In *ECML/PKDD (1)*, volume 10534 of *Lecture Notes in Computer Science*, pp. 668–683. Springer, 2017a.

Minervini, P., Demeester, T., Rocktäschel, T., and Riedel, S. Adversarial sets for regularising neural link predictors. In *UAI*. AUAI Press, 2017b.

Minervini, P., Bosnjak, M., Rocktäschel, T., Riedel, S., and Grefenstette, E. Differentiable reasoning on large knowledge bases and natural language. *CoRR*, abs/1912.10824, 2019.

Rocktäschel, T. and Riedel, S. End-to-end differentiable proving. In *NIPS*, pp. 3788–3800, 2017.

Rocktäschel, T., Singh, S., and Riedel, S. Injecting logical background knowledge into embeddings for relation extraction. In *HLT-NAACL*, pp. 1119–1129. The Association for Computational Linguistics, 2015.

Russell, S. J. and Norvig, P. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010.

Sadeghian, A., Armandpour, M., Ding, P., and Wang, D. Z. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *NeurIPS*, pp. 15321–15331, 2019.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850, 2016.

Seo, M. J., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. In *ICLR (Poster)*. OpenReview.net, 2017.

Shen, Y., Huang, P., Gao, J., and Chen, W. Reasonet: Learning to stop reading in machine comprehension. *CoRR*, abs/1609.05284, 2016.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *EMNLP/IJCNLP (1)*, pp. 4505–4514. Association for Computational Linguistics, 2019.

Sodhani, S., Chandar, S., and Bengio, Y. On training recurrent neural networks for lifelong learning. *CoRR*, abs/1811.07017, 2018.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. End-To-End Memory Networks. In Cortes, C. et al. (eds.), *NIPS*, pp. 2440–2448, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR (Poster)*. OpenReview.net, 2018.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and den Broeck, G. V. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5498–5507. PMLR, 2018.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In Guyon, I. et al. (eds.), *NIPS*, pp. 2316–2325, 2017.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

A. Hyperparameter Selection

For model selection in baseline models, we generated a CLUTRR-like dataset using the code published by Sinha et al. (2019) composed by a training set of graphs with $\{2, 3\}$ edges, and two test sets, one with graphs with three edges, and another with graphs with nine edges. We then selected two sets of hyperparameters for each of the baselines: one that maximises the validation accuracy on graphs with three edges, and another that maximises the test accuracy on graphs with nine edges. For each of the baselines, we considered a wide range of hyperparameters: the dimensionalities of node and edge embeddings were varied in $\{10, 50, 100, 200, 500\}$, the number of attention heads in attention-based architectures in $\{1, 2, \ldots, 10\}$, the number of filters in convolutional architectures in $\{1, 2, \ldots, 10\}$, and the number of hidden units in recurrent architectures in $\{32, 64, 128, 256, 512\}$.

To assess the statistical significance of our results, we ran each of the experiments 10 times, each time with a different seed, and compared the resulting accuracy values using an unequal variances t-test, or Welch's t-test. This is motivated by the observation that accuracy values to be Gaussian-distributed, as they approach a normal distribution for large numbers of re-runs, due to the Central Limit Theorem.

A.1. Optimal Hyperparameters

Graph Attention Networks: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $h = 10$ for the number of attention heads, $k = 50$ for the dimensionality of node embeddings, $k_e = 200$ for the dimensionality of edge embeddings. For validation graphs with 9 edges, $k = 50$, $k_e = 500$, and $h = 10$.

Graph Convolutional Networks: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 50$ for the dimensionality of node embeddings, $k_e = 500$ for the dimensionality of edge embeddings. For validation graphs with 9 edges, $k = 50$, and $k_e = 50$.

Convolutional Neural Networks: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 50$ for the dimensionality of node and edge embeddings, $f = 8$ convolutional filters, and conditional encoding. For validation graphs with 9 edges, $k = 200$, $f = 4$, and conditional encoding.

Recurrent Neural Networks: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 50$ for the dimensionality of node and edge embeddings, $h = 64$ for the size of the hidden state representations, and conditional encoding. For validation graphs with 9 edges, $k = 500$, $h = 512$, and conditional encoding.

Long Short-Memory Networks: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 50$ for the dimensionality of node and edge embeddings, $h = 64$ for the size of the hidden state representations, and conditional encoding. For validations graphs with 9 edges, $k = 100$, $h = 512$, and independent encoding.

Gated Recurrent Units: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 50$ for the dimensionality of node and edge embeddings, $h = 64$ for the size of the hidden state representations, and conditional encoding. For validation graphs with 9 edges, $k = 200$, $h = 512$, and conditional encoding.

CNN with Highway Encoder: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 200$ for the dimensionality of node and edge embeddings, $h = 2$ highway layers, and conditional encoding. For validation graphs with 9 edges, $k = 200$, $h = 1$, and conditional encoding.

Multi-Head Attention: the hyperparameters that maximise the accuracy on validation graphs with 3 edges are $k = 500$ for the dimensionality of node and edge embeddings, $h = 10$ for the number of attention heads, h_k for the size of the hidden state representation of the top LSTM layer, and conditional encoding. For validation graphs with 9 edges, $k = 500$, $h = 10$, $h_k = 128$, and conditional encoding.