SCALABLE NEURAL THEOREM PROVING ON KNOWLEDGE BASES AND NATURAL LANGUAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning over text and Knowledge Bases (KBs) is a major challenge for Artificial Intelligence, with applications in machine reading, dialogue, and question answering. Transducing text to logical forms which can be operated on is a brittle and error-prone process. Operating directly on text by jointly learning representations and transformations thereof by means of neural architectures that lack the ability to learn and exploit general rules can be very data-inefficient and not generalise correctly. These issues are addressed by Neural Theorem Provers (NTPs) (Rocktäschel & Riedel, 2017), neuro-symbolic systems based on a continuous relaxation of Prolog's backward chaining algorithm, where symbolic unification between atoms is replaced by a differentiable operator computing the similarity between their embedding representations. In this paper, we first propose Neighbourhood-approximated Neural Theorem Provers (NaNTPs) consisting of two extensions to NTPs, namely *a)* a method for drastically reducing the previously prohibitive time and space complexity during inference and learning, and *b)* an *attention mechanism* for improving the rule learning process, deeming them usable on real-world datasets. Then, we propose a novel approach for jointly reasoning over KB facts and textual mentions, by jointly embedding them in a shared embedding space. The proposed method is able to *extract rules* and *provide explanations*—involving both textual patterns and KB relations—from large KBs and text corpora. We show that NaNTPs perform on par with NTPs at a fraction of a cost, and can achieve competitive link prediction results on challenging large-scale datasets, including `WN18`, `WN18RR`, and `FB15k-237` (with and without textual mentions) while being able to provide explanations for each prediction and extract interpretable rules.

1 INTRODUCTION

The main focus in Artificial Intelligence is building systems that exhibit intelligent behaviour (Levesque, 2014). In particular, Natural Language Understanding (NLU) and Machine Reading (MR) in particular, aim at building models and systems with the ability to read text, extract meaningful knowledge, and actively reason with it (Etzioni et al., 2006; Hermann et al., 2015; Weston et al., 2015; McCallum et al., 2017a).This ability enables both the synthesis of new knowledge and the possibility to verify and update a given assertion. For example, given the following statement:

The River Thames is in the United Kingdom.

and the following supporting text:

London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia.

a reader can verify that the statement is consistent since *London is standing on the River Thames* and *London is in the United Kingdom*. Automated reasoning applied on text requires Natural Language Processing (NLP) tools capable of extracting meaningful knowledge from free-form text and compiling it into KBs (Niklaus et al., 2018). However, the compiled KBs tend to be incomplete , ambiguous, and noisy, impairing the application of standard deductive reasoners (Huang et al., 2005).

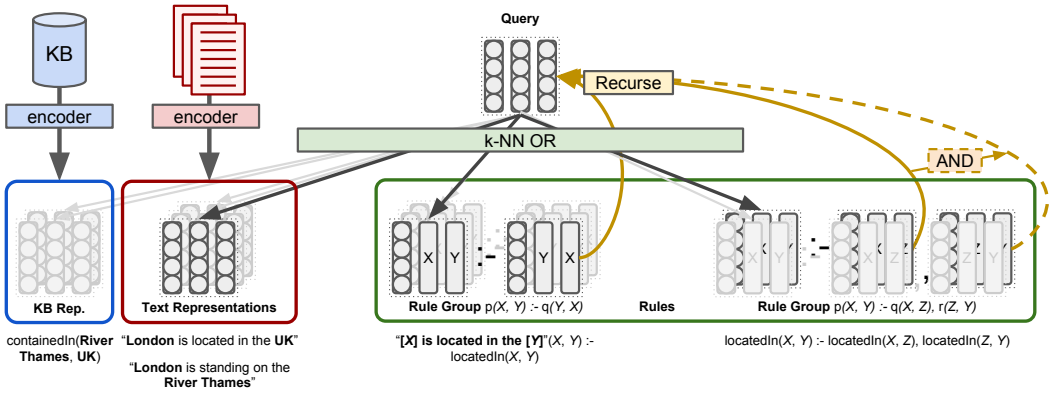

Figure 1: Overall architecture of NaNTPs: the two main contributions consist in a faster inference mechanism (represented by the K-NN OR component, discussed in Section 3) and two dedicated encoders, one for KB facts and rules, and another for text (discussed in Section 4).

A rich and broad literature in MR has approached this problem within a variety of frameworks, including Natural Logic (MacCartney & Manning, 2007; Angeli & Manning, 2014) and Semantic Parsing (Dong & Lapata, 2016; Bos, 2008), and by framing the problem as Natural Language Inference—also referred to as Recognising Textual Entailment (Fyodorov et al., 2000; Condoravdi et al., 2003; Dagan et al., 2005; Bowman et al., 2015; Rocktäschel et al., 2015) —and Question Answering (Hermann et al., 2015). However, such methods suffer from several limitations. For instance, they rely on significant amounts of annotated data to suitably approximate the implicit distribution from which training and test data are drawn, and thus are often unable to generalise correctly in the absence of a sufficient quantity of training data or appropriate priors on model paramaters (e.g. via regularisation). Orthogonally, even if accurate, such methods also cannot provide explanations for a given prediction (Evans & Grefenstette, 2018; Marcus, 2018).

A promising strategy for overcoming these issues consists of combining *neural models* and *symbolic reasoning*, given their complementary strengths and weaknesses (Rocktäschel & Riedel, 2017; Evans & Grefenstette, 2018). While symbolic models can generalise well from a small number of examples when the problem domain fits the inductive biases presented by the symbolic system at hand, they are brittle and prone to failure when the observations are noisy and ambiguous, or when the domain's properties are not known or formalisable, all of which being the case for natural language (Raedt & Kersting, 2008). On the other hand, neural models are robust to noise and ambiguity but prone to overfitting (Marcus, 2018) and not easily interpretable (Lipton, 2018), making them incapable of providing explanations or incorporating background knowledge.

Recent work in neuro-symbolic systems (Garcez et al., 2015) has made progress in learning neural representations that allow for comparison of symbols not on the basis of identity, but of their semantics (as learned in continuous representations of said symbols), while maintaining interpretability and generalisation, thereby inheriting the best of both worlds. Among such systems, NTPs (Rocktäschel & Riedel, 2017) are end-to-end differentiable deductive reasoners based on Prolog's backward chaining algorithm, where unification between atoms is replaced by a differentiable operator computing their similarity between their embedding representations. NTPs are especially interesting since they allow learning *interpretable rules* from data, by back-propagating a KB reconstruction error to the rule representations. Furthermore, NTPs are *explainable*: by looking at the proof tree associated with the highest proof score, it is possible to know which rules are activated during the reasoning process—this enables providing explanations for a given reasoning outcome, performing error analysis, and driving modelling choices. So far, due to their computational complexity, NTPs have only been successfully applied to learning tasks involving very small KBs. However, most human knowledge is still stored in large KBs and natural language corpora, which are difficult to reason over automatically.

With this paper we aim at addressing these issues, by proposing:*a)* An efficient method for significantly reducing the time and space complexity required by NTPs by reducing the number of candidate proof scores and by using an attention mechanism for reducing the number of parameters required for learning new rules (Section 3), and *b)* An extension of NTPs towards text, by jointly embedding

predicates and textual surface patterns in a shared embedding space by means of an efficient reading component (Section 4).

2 NEURAL THEOREM PROVING

In NTPs, the neural network structure is built recursively, and its construction is defined in terms of *modules* similarly to dynamic neural module networks (Andreas et al., 2016). Given a goal, a KB, and a current proof state as inputs, each module produces a list of new proof states, *i.e.*, neural networks representing partial proof success scores and variable substitutions. In the following we briefly overview the modules constituting NTPs.

Unification Module. In backward chaining, unification between two logic atoms is used for checking whether they can represent the same structure. In discrete unification, non-variable symbols are checked for equality, and the proof fails if two symbols differ. Rather than comparing symbols, NTPs compare their *embedding representations* by means of a differentiable similarity function, such as a Radial Basis Function (RBF) kernel. This allows matching different symbols with similar semantics, such as relations like `locatedIn` and `situatedIn`.

Unification is carried out by the `unify` operator, which updates a substitution set S, and creates a neural network for comparing the vector representations of non-variable symbols in two sequences of terms. More details on the `unify` module are available in Appendix A.1.

For example, given two atoms $\mathrm{H} = [\texttt{locatedIn}, \textsc{London}, \textsc{uk}]$ and $\mathrm{G} = [\texttt{situatedIn}, \mathrm{X}, \mathrm{Y}]$, the result of $\texttt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, (\varnothing, 1))$ with $S = (\varnothing, 1)$ will be a new substitution set $S' = (S'_\psi, S'_\rho)$ where $S'_\psi = \{\mathrm{X}/\textsc{London}, \mathrm{Y}/\textsc{uk}\}$ and $S'_\rho = \min(1, k(\boldsymbol{\theta}_{\texttt{locatedIn:}}, \boldsymbol{\theta}_{\texttt{situatedIn:}}))$, where $k(\boldsymbol{\theta}_{\texttt{locatedIn:}}, \boldsymbol{\theta}_{\texttt{situatedIn:}})$ computes the similarity between the embedding representations of the relation types `locatedIn` and `situatedIn`.

OR Module. Given a goal G, the `or` module unifies G with all facts and rules in a KB: for each rule $\mathrm{H} :\!- \mathrm{B} \in \mathfrak{K}$, after unifying G with the head H, it also attempts to prove the atoms in the body H by invoking the `and` module. The signature of `or` is $\mathcal{L} \times \mathbb{N} \times \mathcal{S} \rightarrow \mathcal{S}^N$, where $\mathcal{L}$ is the domain of goal atoms, the second argument specifies the maximum proof depth, and N denotes the number of possible output proof states. This operator is implemented as follows:

$$\texttt{or}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d, S) = [S' \mid S' \in \texttt{and}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{B}, d, \texttt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, S)), \mathrm{H} :\!- \mathrm{B} \in \mathfrak{K}] \tag{1}$$

where $\mathrm{H} :\!- \mathrm{B}$ denotes a rule in a given KB $\mathfrak{K}$ with a head atom H and a list of body atoms B.

For example, given a goal $\mathrm{G} = [\texttt{situatedIn}, \mathrm{Q}, \textsc{uk}]$ and a rule $\mathrm{H} :\!- \mathrm{B}$ with $\mathrm{H} = [\texttt{locatedIn}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\texttt{locatedIn}, \mathrm{X}, \mathrm{Z}], [\texttt{locatedIn}, \mathrm{Z}, \mathrm{Y}]]$, the model would instantiate an `and` sub-module as follows:

$$\texttt{or}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d, S) = [S' | S' \in \texttt{and}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{B}, d, (\{\mathrm{X}/\mathrm{Q}, \mathrm{Y}/\textsc{uk}\}, \hat{S}_\rho)), \ldots]$$

by first unifying the goal G with the head of the rule H, and then proving the sub-goals in the body of the rule by using `and`.

AND Module. The `and` module recursively tries to prove a list of sub-goals, by invoking the `or` module. More details on this module are available in Appendix A.3.

Proof Aggregation. Finally, NTPs define the overall success score of proving a goal G using a KB $\mathfrak{K}$ with parameters $\boldsymbol{\theta}$ as:

$$\texttt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, m) = \underset{\substack{S \in \texttt{or}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d, (\varnothing, 1)) \\ S \neq \texttt{FAIL}}}{\arg\max} S_\rho$$

where d is a predefined maximum proof depth, and the initial proof state is set to $(\varnothing, 1)$, denoting an empty substitution set and a proof success score of 1.

Training. The model is then trained using a Leave-One-Out cross-entropy loss—by removing one fact from the KB, and predicting its score. Since this procedure only generates positive examples, negative examples are generated by corrupting positive examples, by randomly changing the entities (Nickel et al., 2016). We refer to Rocktäschel & Riedel (2017) for more information on the training procedure.

Rule Learning. NTPs can be used for learning *interpretable rules* from data, getting a deeper understanding of the domain. Rocktäschel & Riedel (2017) show that it is possible to learn rules from data by specifying *rule templates* $\mathrm{H} :\!- \mathrm{B}$, with $\mathrm{H} = [\boldsymbol{\theta}_{p:}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\boldsymbol{\theta}_{q:}, \mathrm{X}, \mathrm{Z}], [\boldsymbol{\theta}_{r:}, \mathrm{Z}, \mathrm{Y}]]$, where $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:} \in \mathbb{R}^k$. The parameters $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:}$ can be *learned from data*, and decoded by searching the closest representation of known predicates.

3 Neural Theorem Proving at Scale

Scaling up Inference. The algorithm in Section 2 is capable of deductive reasoning, and the proof paths with the highest score can provide human-readable explanations for a given prediction. However, a significant computational bottleneck lies in the `or` operator in Eq. 1

For instance, assume a KB $\mathfrak{K}$, composed of $|\mathfrak{K}|$ facts and no rules. The number of facts in a real-world KB can be quite large—for instance, Freebase contains 637 million facts (Dong et al., 2014), while the Google Knowledge Graph contains 18 billion facts (Nickel et al., 2016). Given a query G, in the absence of rules, NTP reduces to solving the following optimisation problem:

$$\mathtt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, 1) = \underset{\mathrm{F} \in \mathfrak{K},\ S \neq \mathtt{FAIL}}{\arg\max} S_\rho \ \text{ with } S = \mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{F}, \mathrm{G}, (\varnothing, 1)) \tag{2}$$

that is, it finds the fact $\mathrm{F} \in \mathfrak{K}$ in the KB $\mathfrak{K}$ that, unified with the goal G, yields the maximum unification score. Recall from Section 2 that the unification score between a fact $\mathrm{F} = [\mathrm{F}_p, \mathrm{F}_s, \mathrm{F}_o]$ and a goal $\mathrm{G} = [\mathrm{G}_p, \mathrm{G}_s, \mathrm{G}_o]$ is given by the similarity of their representations in a Euclidean space:

$$\mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{G}, \mathrm{F}, (\varnothing, 1)) = \min\left\{1, k(\boldsymbol{\theta}_{\mathrm{G}_p:}, \boldsymbol{\theta}_{\mathrm{F}_p:}), k(\boldsymbol{\theta}_{\mathrm{G}_s:}, \boldsymbol{\theta}_{\mathrm{F}_s:}), k(\boldsymbol{\theta}_{\mathrm{G}_o:}, \boldsymbol{\theta}_{\mathrm{F}_o:})\right\} \tag{3}$$

where k denotes a RBF kernel, and $\boldsymbol{\theta}_{\mathrm{G}_p:}, \boldsymbol{\theta}_{\mathrm{G}_s:}, \boldsymbol{\theta}_{\mathrm{G}_o:} \in \mathbb{R}^k$ (resp. $\boldsymbol{\theta}_{\mathrm{F}_p:}, \boldsymbol{\theta}_{\mathrm{F}_s:}, \boldsymbol{\theta}_{\mathrm{F}_o:} \in \mathbb{R}^k$) denote the embedding representation of the predicate, first and second argument of the goal G (resp. fact F).

Given a goal G, the NTPs proposed by (Rocktäschel & Riedel, 2017) will compute the unification score in Eq. 3 between G and every fact $\mathrm{F} \in \mathfrak{K}$ in the KB. This is problematic, since computing the similarity between the representations of the goal G and every fact $\mathrm{F} \in \mathfrak{K}$ is computationally prohibitive—the number of comparisons is $\mathcal{O}(|\mathfrak{K}|n)$, where n is the number of goals and sub-goals in the proving process. However, $\mathtt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d)$ only returns the single largest proof score, implying that every lower scoring proof is discarded during both inference and training.

One of the core contributions in this paper is to exactly compute $\mathtt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, m)$ by only considering a subset of proof scores that contains the largest one. Specifically, we make the following observation: given a goal G, if we know the most similar fact $\mathrm{F} \in \mathfrak{K}$ in embedding space as measured by $\mathtt{unify}_{\boldsymbol{\theta}}$, the number of comparisons needed for computing the final proof score $\mathtt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, 1)$ is reduced from $\mathcal{O}(|\mathfrak{K}|)$ to $\mathcal{O}(1)$. The same reasoning can be extended to rules as well.

We argue that, given G, we can restrict the search of the closest fact $\mathrm{F} \in \mathfrak{K}$ to a Euclidean *local neighbourhood* of size n of G, $\mathcal{N}_{\mathfrak{K}}(\mathrm{G}) \subseteq \mathfrak{K}$ such that $|\mathcal{N}_{\mathfrak{K}}(\mathrm{G})| = n$, defined as follows: [1]

$$\mathcal{N}_{\mathfrak{K}}(\mathrm{G}) = \text{k-}\underset{\mathrm{F} \in \mathfrak{K}}{\arg\min} \left\|\boldsymbol{\theta}_{\mathrm{F}:} - \boldsymbol{\theta}_{\mathrm{G}:}\right\|_2 \tag{4}$$

Then, the matching fact F will be very likely to be contained across the n most similar facts:

$$\mathtt{ntp}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, 1) \approx \underset{\substack{S = \mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{F}, \mathrm{G}, (\varnothing, 1)) \\ \mathrm{F} \in \mathcal{N}_{\mathfrak{K}}(\mathrm{G}),\ S \neq \mathtt{FAIL}}}{\arg\max} S_\rho \tag{5}$$

where the size of the neighbourhood is much lower than the size of the whole KB, *i.e.*, $|\mathcal{N}_{\mathfrak{K}}(\mathrm{G})| \ll |\mathfrak{K}|$. The same idea can be extended from facts to rules, by selecting only the rules $\mathrm{H} :\!- \mathrm{B} \in \mathfrak{K}$ such that

[1] We approximate the neighbourhood with respect to the minimum of component distances with the neighbourhood with respect to a distance of concatenated representations.

their head H is closer to the goal. However, finding the *exact* neighbourhood of a point in a Euclidean space is very costly, due to the *curse of dimensionality* (Indyk & Motwani, 1998). Experiments showed that methods for identifying the exact neighbourhood can rarely outperform brute-force linear scan methods when dimensionality is high (Weber et al., 1998).

A practical solution consists of Approximate Nearest Neighbour Search (ANNS) algorithms, which focus on finding an *approximate* solution to the k-nearest neighbour search problem outlined in Eq. 4 on high dimensional data. Several families of ANNS algorithms exist, such as Locally-Sensitive Hashing (Andoni et al., 2015), Product Quantisation (Jégou et al., 2011; Johnson et al., 2017), and Proximity Graphs (Malkov et al., 2014).

In this work, we use Hierarchical Navigable Small World (HNSW) (Malkov & Yashunin, 2016), a graph-based incremental ANNS structure which can offer significantly better logarithmic complexity scaling during neighbourhood search than other approaches (Li et al., 2016). Specifically, given a subset of the KB $\mathcal{P} \subseteq \mathfrak{K}$—for instance, containing all facts in $\mathfrak{K}$—we construct a HNSW graph for all elements in $\mathcal{P}$, which has a $\mathcal{O}(|\mathcal{P}| \log |\mathcal{P}|)$ time complexity. Then, given a goal G, the HNSW graph is used for identifying its neighbourhood $\mathcal{N}_{\mathcal{P}}(\mathrm{G})$ within $\mathcal{P}$, which has a $\mathcal{O}(\log |\mathcal{P}|)$ time complexity. In our implementation, we construct the HNSW graph-based indexing structure when instantiating the model and, during training, we update the index every b batches.

Specifically, in our implementation, we generate a partitioning $\mathfrak{P} \in 2^{\mathfrak{K}}$ of the KB $\mathfrak{K}$, where each element in $\mathfrak{P}$ groups all facts and rules in $\mathfrak{K}$ sharing the same signature. Then, we redefine the `or` operator as follows:

$$\mathtt{or}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathrm{G}, d, S) = [S' \mid S' \in \mathtt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathrm{B}, d, \mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, S)), \mathrm{H} \mathrel{:-} \mathrm{B} \in \mathcal{N}_{\mathcal{P}}(\mathrm{G}), \mathcal{P} \in \mathfrak{P}] \tag{6}$$

where, instead of trying to unify a goal or sub-goal G with all facts and rule heads in the KB, we constrain the unification with ANNS to only facts and rule heads in its local neighbourhood $\mathcal{N}_{\mathfrak{K}}(\mathrm{G})$.

Improving Rule Learning via Attention. Although NTPs can be used for *learning interpretable rules* from data, the solution proposed by Rocktäschel & Riedel (2017) can be quite data-inefficient, as the number of parameters associated to a rule can be quite large. For instance, assume the rule H :– B, with $\mathrm{H} = [\boldsymbol{\theta}_{p:}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\boldsymbol{\theta}_{q:}, \mathrm{X}, \mathrm{Z}], [\boldsymbol{\theta}_{r:}, \mathrm{Z}, \mathrm{Y}]]$ discussed in Section 2, where $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:} \in \mathbb{R}^k$. Such a rule introduces $3k$ parameters in the model, and it may be computationally inefficient to learn each of the embedding vectors.

We propose using an *attention mechanism* (Bahdanau et al., 2015) for attending over known predicates for defining the predicate embeddings $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:}$. Let $\mathcal{R}$ be the set of known predicates, and let $R \in \mathbb{R}^{|\mathcal{R}| \times k}$ be a matrix representing the embeddings for the predicates in $\mathcal{R}$. We define the $\boldsymbol{\theta}_{p:}$ as:

$$\boldsymbol{\theta}_{p:} = \mathrm{softmax}(\mathbf{a}_{p:})^{\intercal} R \tag{7}$$

where $\mathbf{a}_{p:} \in \mathbb{R}^{|\mathcal{R}|}$ is a set of trainable *attention weights* associated with the predicate p. This sensibly improves the parameter efficiency of the model in cases where the number of known predicates is low, *i.e.* $|\mathcal{R}| \ll k$, by introducing $c|\mathcal{R}|$ parameters for each rule rather than ck, where c is the number of trainable predicate embeddings in the rule.

4 Jointly Reasoning on Knowledge Bases and Text

In this section, we show we can use NaNTPs for jointly reasoning over KBs and natural language corpora. In the following, we assume that our KB $\mathfrak{K}$ is composed by facts, rules, and *mentions*. A fact is composed by a predicate symbol and a sequence of arguments, *e.g.* [`locationOf`, LONDON, UK]. On the other hand, a *mention* is a textual pattern between two co-occurring entities in the KB (Gabrilovich et al., 2013; Toutanova et al., 2015), such as "LONDON is located in the UK".

We represent mentions jointly with facts and rules in $\mathfrak{K}$ by considering each textual surface pattern linking two entities as a new predicate, and embedding it in a d-dimensional space by means of an end-to-end differentiable reading component. For instance, the sentence "United Kingdom borders with Ireland" is translated into the following mention in $\mathfrak{K}$: [[`[arg1]`, `borders`, `with`, `[arg2]`], UK, IRELAND] by first identifying sentences or paragraphs containing KB entities, and then considering the textual surface pattern connecting such entities as an extra relation type.

While predicates in $\mathcal{R}$ are encoded by a look-up operation to a predicate embedding matrix $R \in \mathbb{R}^{|\mathcal{R}| \times k}$, textual surface patterns are encoded by an $\texttt{encode}_{\boldsymbol{\theta}}$ module. The signature of $\texttt{encode}_{\boldsymbol{\theta}}$ is $\mathcal{V}^* \rightarrow \mathbb{R}^k$, where $\mathcal{V}$ is the vocabulary of words and symbols occurring in textual surface patterns: it takes a sequence of tokens, and maps it to a k-dimensional embedding space.

More formally, given a textual surface pattern $t \in \mathcal{V}^*$—for instance, $t = [\texttt{[arg1]}, \texttt{borders}, \texttt{with}, \texttt{[arg2]}]$—the $\texttt{encode}_{\boldsymbol{\theta}}$ module first encodes each token w in t by means of a token embedding matrix $V \in \mathbb{R}^{|\mathcal{V}| \times k'}$, resulting in a pattern matrix $W_t \in \mathbb{R}^{|t| \times k'}$. Then, the module produces a textual surface pattern embedding vector $\boldsymbol{\theta}_{t:} \in \mathbb{R}^k$ from W_t by means of an end-to-end differentiable encoder.For the sake of simplicity and efficiency, in this paper, we use a simple $\texttt{encode}_{\boldsymbol{\theta}}$ module that computes the average of the token embedding vectors composing a textual surface pattern:

$$\texttt{encode}_{\boldsymbol{\theta}}(t \in \mathcal{V}^*) = \frac{1}{|t|} \sum_{w \in t} V_{w\cdot} \in \mathbb{R}^k$$

albeit the encoder $\texttt{encode}$ can be implemented by using other differentiable architectures, such as Recurrent Neural Networks (RNNs).

5 Related Work

A significant corpus of literature aims at addressing the limitations of neural architectures in terms of generalisation and reasoning abilities. A line of research consists of enriching neural network architectures with a differentiable *external memory* (Sukhbaatar et al., 2015; Graves et al., 2014; Joulin & Mikolov, 2015; Grefenstette et al., 2015; Kaiser & Sutskever, 2016; Miller et al., 2016; Graves et al., 2016). The underlying idea is that a neural network can learn to represent and manipulate complex data structures, thus disentangling the algorithmic part of the process from the representation of the inputs.

Another way of improving the generalisation and extrapolation abilities of neural networks consists of designing architectures capable of learning general, reusable *programs*—atomic primitives that can be reused across a variety of environments and tasks (Reed & de Freitas, 2016; Neelakantan et al., 2016; Parisotto et al., 2016). By doing so, it becomes also possible to train such models from enriched supervision signals, such as from *program traces* rather than simple input-output pairs.

Yet another line of work is *differentiable interpreters*—program interpreters where declarative or procedural knowledge, *e.g.*, a sorting program, is compiled into a neural network architecture (Bošnjak et al., 2017; Gaunt et al., 2016; Rocktäschel & Riedel, 2017; Evans & Grefenstette, 2018)—NTPs fall in this category. This family of models allows imposing strong inductive biases on the models by partially defining the program structure used for constructing the network, *e.g.*, in terms of instruction sets or rules. A major problem with differentiable interpreters, however, is their computational complexity, that so far deemed them unusable except for smaller-scale learning problems.

This work is also related to Rae et al. (2016), which use an approximate nearest neighbour data structure for sparsifying read operations in memory networks. Furthermore, Riedel et al. (2013) pioneered the idea of jointly embedding KB facts and textual mentions in a shared embedding space, by considering mentions as additional relations in a KB factorisation setting. This idea was later extended to more elaborate mention encoders by McCallum et al. (2017b). Our work is also related to path encoding models (Das et al., 2016) and random walk approaches (Lao et al., 2011; Gardner et al., 2014), which both lack rule induction mechanisms. Lastly, our work is related to Yang et al. (2017) which is a scalable rule induction approach for knowledge base completion, but has not been applied to textual surface patterns.

6 Experiments

6.1 Datasets and Evaluation Protocols

We report the results of experiments on benchmark datasets — Countries (Bouchard et al., 2015), Nations, UMLS, and Kinship (Kemp et al., 2006) — following the same evaluation protocols as Rocktäschel & Riedel (2017). Furthermore, since our scalability improvements described in Section 3

Table 1: Link prediction results for WN18, WN18RR, and FB15k-237—ComplEx and DistMult were trained with embedding size $d = 100$.

	WN18				**WN18RR**				**FB15k-237**			
		Hits				Hits				Hits		
	MRR	@10	@3	@1	MRR	@10	@3	@1	MRR	@10	@3	@1
DistMult	0.782	0.931	0.910	0.658	0.411	0.463	0.423	0.382	**0.214**	**0.402**	**0.240**	**0.127**
ComplEx	**0.923**	**0.948**	**0.941**	**0.904**	**0.420**	**0.469**	**0.431**	**0.395**	0.206	0.373	0.222	0.126
NaNTP	0.769	0.937	0.884	0.649	0.398	0.432	0.402	0.377	0.176	0.310	0.185	0.110

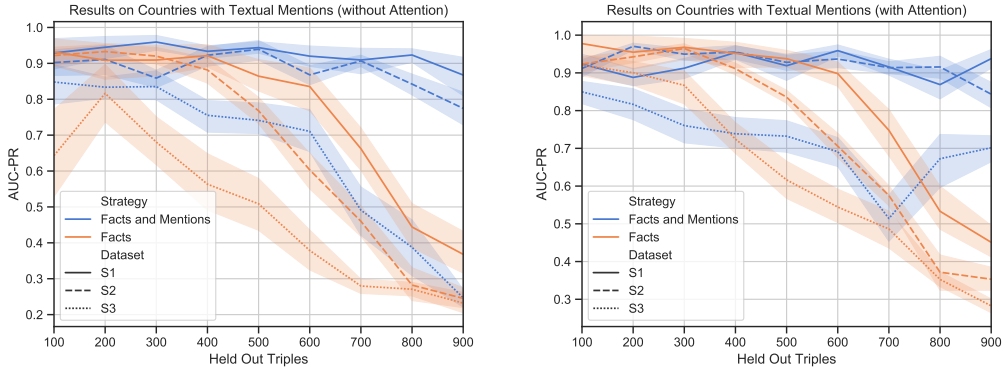

Figure 2: Given the Countries dataset, we replaced a varying number of training triples with mentions (see Appendix B.1.2 for details) and integrated the mentions using two different strategies: by encoding the mentions using the encoder introduced in Section 4 (*Facts and Mentions*) and by simply adding them to the KB (*Facts*).Experiments were conducted with the attention mechanism proposed in Section 3 (right) and the standard rule-learning procedure (left), each with 10 different random seeds. We can see that, on each of the datasets, using the encoder yields consistently better AUC-PR values than simply adding the mentions to the KB.

allow us to experiment on significantly larger datasets, we also report results on the WN18 (Bordes et al., 2013), WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015) datasets—whose characteristics are outlined in Table 5. For evaluating our natural language reading component proposed in Section 4, we use FB15k-237.E (Toutanova et al., 2015)—a set of textual mentions for all entity pairs derived from ClueWeb12 with Freebase entity mention annotations Gabrilovich et al. (2013)—and a set of manually generated mentions for Countries. Results are reported in terms of Area Under the Precision-Recall Curve (Davis & Goadrich, 2006) (AUC-PR), Mean Reciprocal Rank (MRR), and HITS@m (Bordes et al., 2013). All datasets are described in detail in Appendix B.

Baselines. We compare NaNTPs with NTPs on benchmark datasets, and with DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016), two state-of-the-art Neural Link Predictors used for identifying missing facts in potentially very large KBs, on WordNet and FreeBase. For computing likelihoods of facts, DistMult and ComplEx embed each entity and relation type in a d-dimensional embedding space and use a differentiable scoring function based on the embeddings corresponding to the entity and relation of a fact. Embedding representations and scoring function parameters are learned jointly by minimising a KB reconstruction error.

Note that, while the complexity of scoring a triple in DistMult and ComplEx is $\mathcal{O}(1)$—they only need the embeddings of the symbols within a fact for computing the ranking score—instead in our model, it is $\mathcal{O}(\log |\hat{\mathfrak{K}}|)$. For such a reason, instead of performing a full hyperparameter search, we fix some of the hyperparameters—such as the embedding size $d = 100$—and report results using these hyperparameters for both NaNTPs and baselines.

6.2 Scalability Experiments

Table 2: Explanations, in terms of rules and supporting facts, for the queries in the validation set of WN18 and WN18RR provided by NaNTPs by looking at the proof paths yielding the largest proof scores.

	Query	Score S_ρ	Proofs / Explanations
WN18	part_of(CONGO.N.03, AFRICA.N.01)	0.995	part_of(X, Y) :– has_part(Y, X) has_part(AFRICA.N.01, CONGO.N.03)
		0.787	part_of(X, Y) :– instance_hyponym(Y, X) instance_hyponym(AFRICAN_COUNTRY.N.01, CONGO.N.03)
	hyponym(EXTINGUISH.V.04, DECOUPLE.V.03)	0.987	hyponym(X, Y) :– hypernym(Y, X) hypernym(DECOUPLE.V.03, EXTINGUISH.V.04)
		0.920	hypernym(SNUFF_OUT.V.01, EXTINGUISH.V.04)
	part_of(PITUITARY.N.01, DIENCEPHALON.N.01)	0.995	has_part(DIENCEPHALON.N.01, PITUITARY.N.01)
	has_part(TEXAS.N.01, ODESSA.N.02)	0.961	has_part(X, Y) :– part_of(Y, X) part_of(ODESSA.N.02, TEXAS.N.01)
	hyponym(SKELETAL_MUSCLE, ARTICULAR_MUSCLE)	0.987	hypernym(ARTICULAR_MUSCLE, SKELETAL_MUSCLE)
	deriv_related_form(REWRITE, REWRITING)	0.809	deriv_related_form(X, Y) :– hypernym(Y, X) hypernym(REVISE, REWRITE)
WN18RR	also_see(TRUE.A.01, FAITHFUL.A.01)	0.962	also_see(X, Y) :– also_see(Y, X) also_see(FAITHFUL.A.01, TRUE.A.01)
		0.590	also_see(CONSTANT.A.02, FAITHFUL.A.01)
	also_see(GOOD.A.03, VIRTUOUS.A.01)	0.962	also_see(VIRTUOUS.A.01, GOOD.A.03)
		0.702	also_see(RIGHTEOUS.A.01, VIRTUOUS.A.01)
	instance_hypernym(CHAPLIN, FILM_MAKER)	0.812	instance_hypernym(CHAPLIN, COMEDIAN)

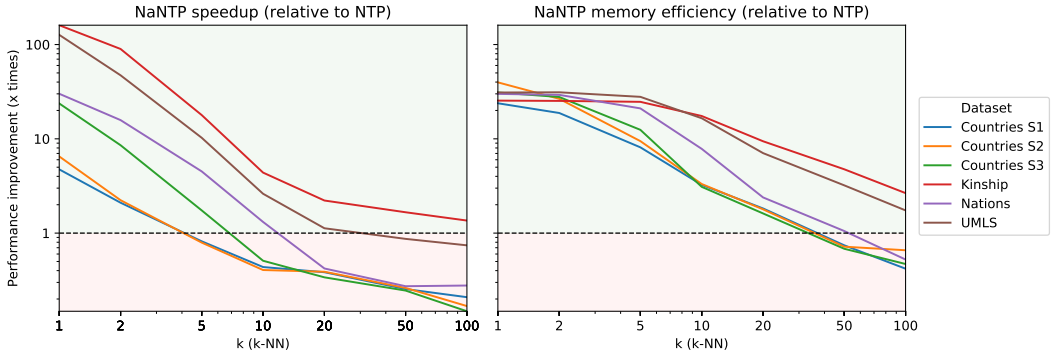

Figure 3: Run-time and memory performance of NaNTP in comparison with NTP. Run-time speedup calculated as the ratio of examples per second of NaNTP and NTP. Memory efficiency calculated as a ratio of the memory use of NTP and NaNTP. Dashed line denotes equal performance – above it (green) NaNTP performs better, below it (red) performs worse.

Evaluation Performance Comparison In order to verify the correctness of our approximation, we compare the evaluation performance of NaNTP and NTP on the set of benchmark datasets presented in Rocktäschel & Riedel (2017). Results, presented in Table 3 show that NaNTP achieves on par or better results than NTP, consistently through benchmark datsets (see footnote[0] for a comment on the overly optimistic evaluation used in Rocktäschel & Riedel (2017)). On Countries S1, S2 and S3 dataset, NaNTP achieves both better performance and lower standard deviations (on 10 runs). On Kinship, Nations, and UMLS, NaNTP achieves better performance on Kinship, but (significantly) worse on Nations and UMLS. Please note that due to the late discovery of the optimistic evaluation issue[0] in Rocktäschel & Riedel (2017), we were unable to re-evaluate results for NTP. When running the same evaluation used in Rocktäschel & Riedel (2017), we achieve significantly better performance on Kinship, better performance on Nations, and are on par for UMLS. However, in the interest of correctness, we do not report these results in the main body of the paper but leave them in C.1), and will update the paper with the standardised results in the next version.

Run-Time Performance comparison To assess the run-time gains of NaNTP, we compare it to NTP with respect to time and memory performance during training. In our experiments, we vary the n of the ANNS approximation to assess the computational demands by increasing n. First, we compare the average number of examples (queries) per second by running 10 training batches with a

Table 3: Comparison of NaNTPs and NTPs on benchmark datasets.[0]

Datasets		Metrics	Models			
			NTP[0]	NaNTP		
				k=1	k=2	k=5
Countries	S1	AUC-PR	90.83 ± 15.4	**99.2 ± 1.19**	97.34 ± 3.86	96.37 ± 4.23
	S2		87.40 ± 11.7	**93.48 ± 3.29**	88.48 ± 5.87	83.56 ± 7.78
	S3		56.68 ± 17.6	**85.24 ± 5.83**	82.18 ± 9.11	72.68 ± 13.4
Kinship		MRR	0.6[0]	0.51 ± 0.04	0.66 ± 0.03	**0.67 ± 0.02**
		HITS@1	0.48[0]	0.37 ± 0.04	0.51 ± 0.04	**0.52 ± 0.02**
		HITS@3	0.7[0]	0.57 ± 0.05	0.78 ± 0.03	**0.78 ± 0.02**
		HITS@10	0.78[0]	0.82 ± 0.04	0.94 ± 0.02	**0.95 ± 0.0**
Nation		MRR	0.75[0]	**0.66 ± 0.03**	0.59 ± 0.02	0.54 ± 0.05
		HITS@1	0.62[0]	**0.51 ± 0.05**	0.41 ± 0.03	0.36 ± 0.07
		HITS@3	0.86[0]	**0.77 ± 0.03**	0.71 ± 0.02	0.65 ± 0.06
		HITS@10	0.99[0]	**0.99 ± 0.0**	0.98 ± 0.01	0.98 ± 0.01
UMLS		MRR	0.88[0]	0.63 ± 0.04	**0.8 ± 0.02**	0.8 ± 0.02
		HITS@1	0.82[0]	0.49 ± 0.05	**0.68 ± 0.02**	0.68 ± 0.03
		HITS@3	0.92[0]	0.73 ± 0.04	**0.91 ± 0.01**	0.9 ± 0.02
		HITS@10	0.97[0]	0.89 ± 0.02	**0.98 ± 0.01**	0.97 ± 0.01

maximum batch to fit the memory of NVIDIA GeForce GTX 1080 Ti, for all models. Second, we compare the maximum memory usage of both models on a CPU, over 10 training batches with same batch sizes. The comparison is done on a CPU to ensure that we include the size of the ANNS index in NaNTP measures and as a fail-safe, in case the model doesn't fit on the memory of GPU.

The results, presented in Figure 3, demonstrate that, compared to NTP, NaNTP is considerably mor time and memory efficiency. In particular, we observe that NaNTP yields significant speedups of an order of magnitude for smaller datasets (Countries S1 and S2), and two orders of magnitude for larger datasets (Kinship and Nations). Interestingly, with the increased size of the dataset, NaNTP consistently achieves higher speedups, when compared to NTP. Similarly, NaNTP is more memory efficient, with savings bigger than an order of magnitude, making them readily applicable to larger datasets, even when augmented with textual surface forms.

6.3 Results on Countries, UMLS, and Nations

Experiments with Generated Mentions. For evaluating different strategies of integrating textual surface patterns, in the form of mentions, in NTPs, we proceeded as follows. We replaced a varying number of training set triples from each of the Countries datasets (S1, S2, S3) with textual mentions generated by humans. For instance, the fact `neighbourOf`(UK, Ireland) may be replaced by the mention "UK `is neighbouring with` Ireland".

Then, we evaluate two ways of integrating textual mentions in NaNTPs, *i.e.*, either by adding them as facts to the KB, or by parsing the mention by means of an encoder, as described in Section 4. We show results in Fig. 2. We can see that, especially if the number of held-out facts is higher, using the proposed encoding module yields better AUC-PR values on all datasets.

Explanations Involving Mentions. NaNTPs is extremely efficient at learning rules involving both *logic atoms and textual mentions*. For instance, by analysing the learned models and their

[0]The results for NTP reported in Table 3 are based on those reported by Rocktäschel & Riedel (2017). We discovered hours before submission that to obtain these results, the authors use the ordering of the gold-standard facts for ranking facts with the same score, which requires privileged information not available to the model. We use lexical ordering instead, which is a fairer comparison metric. Time did not permit to re-run the original NTP code the authors provide, to provide properly standardised results against which to compare, but these will be presented in an updated version of the paper.

explanations, we can see that NaNTPs learn patterns such as:

$$\begin{aligned}
\texttt{neighborOf}(X, Y) &:\!- \quad \texttt{neighborOf}(Y, X) \\
\texttt{neighborOf}(X, Y) &:\!- \quad \text{X:was:a:neighbor:of:Y}(Y, X) \\
\texttt{neighborOf}(X, Y) &:\!- \quad \text{X:is:a:neighboring:state:to:Y}(Y, X) \\
\texttt{locatedIn}(X, Y) &:\!- \quad \text{X:was:a:neighboring:state:to:Y}(X, Z), \text{X:was:located:in:Y}(Z, Y) \\
\texttt{locatedIn}(X, Y) &:\!- \quad \text{X:can:be:found:in:Y}(X, Z), \text{X:can:be:found:in:Y}(Z, Y)
\end{aligned}$$

and leverage them during their reasoning process, providing human-readable explanations for a given prediction.

6.4 Results on WordNet and Freebase

Results are summarised in Table 1, while Table 2 shows a sample of explanations for the facts in the validation set of WN18 and WN18RR provided by NaNTPs by analysing the proof paths yielding the largest proof scores. We can see that NaNTPs is capable of learning rules, such as $\texttt{has_part}(X, Y) :\!- \texttt{part_of}(Y, X)$, and $\texttt{hyponym}(X, Y) :\!- \texttt{hypernym}(Y, X)$.

Interestingly, it is also able to find an alternative, non-trivial explanations for a given fact, based on the similarity between entity representations. For instance, it can explain that CONGO is part of AFRICA by leveraging the similarity between AFRICA and AFRICAN_ COUNTRY, and the fact that the latter is a hyponym of CONGO. It is also able to explain that CHAPLIN is a FILM_MAKER by leveraging the prior knowledge that CHAPLIN is a COMEDIAN, and the similarity between FILM_MAKER and COMEDIAN.

7 Conclusion

NTPs combine the strengths of rule-based and neural models but, so far, they were unable to reason over large KBs, and therefore over natural language. In this paper, we proposed NaNTPs that utilise ANNS and attention over rules as a solution to scaling issues of NTP. By dynamically considering only the highest, approximately scoring proofs, NaNTPs yield drastic speedups and memory efficiency, while performing on par or better than NTPs. This then enables application of NaNTPs to mixed KB and natural language data by embedding logic atoms and textual mentions in a joint embedding space. Albeit underperforming when compared to state-of-the-art models for Neural Link Prediction on large datasets, NaNTPs is interpretable and is able to provide explanations of its reasoning at scale.

References

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In Corinna Cortes et al. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pp. 1225–1233, 2015.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In Kevin Knight et al. (eds.), *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1545–1554. The Association for Computational Linguistics, 2016.

Gabor Angeli and Christopher D Manning. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 534–545, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges et al.

(eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pp. 2787–2795, 2013.

Johan Bos. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pp. 277–286. Association for Computational Linguistics, 2008.

Guillaume Bouchard, Sameer Singh, and Theo Trouillon. On approximate reasoning capabilities of low-rank vector spaces. In *Proceedings of the 2015 AAAI Spring Symposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, 2015.

Matko Bošnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel. Programming with a differentiable forth interpreter. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, pp. 547–556. PMLR, 2017.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez et al. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 632–642. The Association for Computational Linguistics, 2015. ISBN 978-1-941643-32-7.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. Entailment, intensionality and text understanding. In Graeme Hirst and Sergei Nirenburg (eds.), *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pp. 38–45, 2003.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In Joaquin Quiñonero Candela et al. (eds.), *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005*, volume 3944 of *LNCS*, pp. 177–190. Springer, 2005. ISBN 3-540-33427-0.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In William W. Cohen et al. (eds.), *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, volume 148 of *ACM International Conference Proceeding Series*, pp. 233–240. ACM, 2006.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith et al. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In Sofus A. Macskassy et al. (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pp. 601–610. ACM, 2014.

Oren Etzioni, Michele Banko, and Michael J Cafarella. Machine reading. In *AAAI*, volume 6, pp. 1517–1519, 2006.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Yaroslav Fyodorov, Yoad Winter, and Nissim Francez. A natural logic inference system. In *Proceedings of the of the 2nd Workshop on Inference in Computational Semantics*, 2000.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June 2013.

Ad Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, Stanford*, 2015.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In Alessandro Moschitti et al. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 397–406. ACL, 2014. ISBN 978-1-937284-96-1.

Alexander L Gaunt, Marc Brockschmidt, Rishabh Singh, Nate Kushman, Pushmeet Kohli, Jonathan Taylor, and Daniel Tarlow. Terpret: A probabilistic programming language for program induction. *arXiv preprint arXiv:1608.04428*, 2016.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pp. 1828–1836, 2015.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In Leslie Pack Kaelbling et al. (eds.), *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 454–459. Professional Book Center, 2005.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Jeffrey Scott Vitter (ed.), *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pp. 604–613. ACM, 1998.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pp. 190–198, 2015.

Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In *Proceedings of the International Conference on Learning Representations*, 2016.

Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, pp. 5, 2006.

Stanley Kok and Pedro M. Domingos. Statistical predicate invention. In Zoubin Ghahramani (ed.), *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, volume 227 of *ACM International Conference Proceeding Series*, pp. 433–440. ACM, 2007.

Ni Lao, Tom M. Mitchell, and William W. Cohen. Random walk inference and learning in A large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pp. 529–539. ACL, 2011. ISBN 978-1-937284-11-4.

Hector J. Levesque. On our best behaviour. *Artif. Intell.*, 212:27–35, 2014.

Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). *CoRR*, abs/1610.02455, 2016.

Zachary C. Lipton. The mythos of model interpretability. *ACM Queue*, 16(3):30, 2018.

Bill MacCartney and Christopher D Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 193–200. Association for Computational Linguistics, 2007.

Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.

Yury A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016.

Gary Marcus. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631, 2018.

Andrew McCallum, Arvind Neelakantan, Rajarshi Das, and David Belanger. Chains of reasoning over entities, relations, and text using recurrent neural networks. In Mirella Lapata et al. (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pp. 132–141. Association for Computational Linguistics, 2017a.

Andrew McCallum, Arvind Neelakantan, and Patrick Verga. Generalizing to unseen entities and entity pairs with row-less universal schema. In Mirella Lapata et al. (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pp. 613–622. Association for Computational Linguistics, 2017b.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In Jian Su et al. (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pp. 1400–1409. The Association for Computational Linguistics, 2016.

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

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of the International Conference on Learning Representations*, 2016.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. A survey on open information extraction. *CoRR*, abs/1806.05599, 2018. URL `http://arxiv.org/abs/1806.05599`.

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis. *arXiv preprint arXiv:1611.01855*, 2016.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*, pp. 3621–3629, 2016.

Luc De Raedt and Kristian Kersting. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *LNCS*, pp. 1–27. Springer, 2008.

Scott E. Reed and Nando de Freitas. Neural programmer-interpreters. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In Lucy Vanderwende et al. (eds.), *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*, pp. 74–84. The Association for Computational Linguistics, 2013.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In Isabelle Guyon et al. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 3791–3803, 2017.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In Corinna Cortes et al. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pp. 2440–2448, 2015.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In Lluís Màrquez et al. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 1499–1509. The Association for Computational Linguistics, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan et al. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2071–2080. JMLR.org, 2016.

Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Ashish Gupta et al. (eds.), *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases*, pp. 194–205. Morgan Kaufmann, 1998.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base completion. *CoRR*, abs/1702.08367, 2017. URL `http://arxiv.org/abs/1702.08367`.

A NEURAL THEOREM PROVING

In NTPs, the neural network structure is built recursively, and its construction is defined in terms of *modules* similarly to dynamic neural module networks (Andreas et al., 2016). Given a goal, a KB, and a current proof state as inputs, each module produces a list of new proof states, *i.e.*, neural networks representing partial proof success scores and variable substitutions. In the following we briefly overview the modules constituting NTPs.

A.1 UNIFICATION MODULE

In backward chaining, unification between two logic atoms is used for checking whether they can represent the same structure. In discrete unification, non-variable symbols are checked for equality, and the proof fails if two symbols differ. Rather than comparing symbols, NTPs compare their *embedding representations* by means of an end-to-end differentiable similarity function, such as a RBF kernel. This allows matching different symbols with similar semantics, such as relations like `locatedIn` and `situatedIn`.

Unification is carried out by the `unify` operator, which updates a substitution set S, and creates a neural network for comparing the vector representations of non-variable symbols in two sequences of terms. The signature of `unify` is $\mathcal{L} \times \mathcal{L} \times \mathcal{S} \rightarrow \mathcal{S}$, where $\mathcal{L}$ is the domain of lists of terms: it takes two atoms, represented as lists of terms, and an upstream proof state, and returns a new proof state S'.

More formally, let H and G denote two lists of terms, each denoting a logical atom, such as $[\mathtt{p}, \mathtt{A}, \mathtt{B}]$ and $[\mathtt{q}, \mathtt{X}, \mathtt{Y}]$ for respectively denoting the atoms $\mathtt{p}(\mathtt{A}, \mathtt{B})$ and $\mathtt{q}(\mathtt{X}, \mathtt{Y})$. Given a *proof state* $S = (S_\psi, S_\rho)$, where S_ψ and S_ρ respectively denote a *substitution set* and a *proof score*, unification is computed as follows:

1. $\mathtt{unify}_{\boldsymbol{\theta}}([\,], [\,], S) = S$
2. $\mathtt{unify}_{\boldsymbol{\theta}}([\,], \mathrm{G}, S) = \mathtt{FAIL}$
3. $\mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, [\,], S) = \mathtt{FAIL}$
4. $\mathtt{unify}_{\boldsymbol{\theta}}(h :: \mathrm{H}, g :: \mathrm{G}, S) = \mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, S')$ with $S' = (S'_\psi, S'_\rho)$ where:

$$S'_\psi = S_\psi \cup \left\{ \begin{array}{ll} \{h/g\} & \text{if } h \in \mathcal{V} \\ \{g/h\} & \text{if } g \in \mathcal{V}, h \notin \mathcal{V} \\ \varnothing & \text{otherwise} \end{array} \right\}, \quad S'_\rho = \min\left(S_\rho, \left\{ \begin{array}{ll} k\left(\boldsymbol{\theta}_{h:}, \boldsymbol{\theta}_{g:}\right) & \text{if } h \notin \mathcal{V}, g \notin \mathcal{V} \\ 1 & \text{otherwise} \end{array} \right\}\right)$$

Here, S' refers to the new proof state, $\mathcal{V}$ refers to the set of variable symbols, h/g is a substitution from the variable symbol h to the symbol g, and $\boldsymbol{\theta}_{g:}$ denotes the embedding look-up of the non-variable symbol with index g.

For example, given two atoms $\mathrm{H} = [\mathtt{locatedIn}, \mathrm{LONDON}, \mathrm{UK}]$ and $\mathrm{G} = [\mathtt{situatedIn}, \mathrm{X}, \mathrm{Y}]$, the result of $\mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, (\varnothing, 1))$ with $S = (\varnothing, 1)$ will be a new substitution set $S' = (S'_\psi, S'_\rho)$ where $S'_\psi = \{\mathrm{X}/\mathrm{LONDON}, \mathrm{Y}/\mathrm{UK}\}$ and $S'_\rho = \min\left(1, k\left(\boldsymbol{\theta}_{\mathtt{locatedIn}:}, \boldsymbol{\theta}_{\mathtt{situatedIn}:}\right)\right)$

A.2 OR MODULE

Given a goal G, the `or` module unifies G with all facts and rules in a KB: for each rule $\mathrm{H} :\!- \mathrm{B} \in \mathfrak{K}$, after unifying G with the head H, it also attempts to prove the atoms in the body H by invoking the `and` module. The signature of `or` is $\mathcal{L} \times \mathbb{N} \times \mathcal{S} \rightarrow \mathcal{S}^N$, where $\mathcal{L}$ is the domain of goal atoms, the second argument specifies the maximum proof depth, and N denotes the number of possible output proof states. This operator is implemented as follows:

$$\mathtt{or}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d, S) = [S' \mid S' \in \mathtt{and}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{B}, d, \mathtt{unify}_{\boldsymbol{\theta}}(\mathrm{H}, \mathrm{G}, S)), \mathrm{H} :\!- \mathrm{B} \in \mathfrak{K}] \tag{8}$$

where $\mathrm{H} :\!- \mathrm{B}$ denotes a rule in a given KB $\mathfrak{K}$ with a head atom H and a list of body atoms B.

For example, given a goal $\mathrm{G} = [\mathtt{situatedIn}, \mathrm{Q}, \mathrm{UK}]$ and a rule $\mathrm{H} :\!- \mathrm{B}$ with $\mathrm{H} = [\mathtt{locatedIn}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\mathtt{locatedIn}, \mathrm{X}, \mathrm{Z}], [\mathtt{locatedIn}, \mathrm{Z}, \mathrm{Y}]]$, the model would instantiate an `and` sub-module as follows:

$$\mathtt{or}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{G}, d, S) = [S' | S' \in \mathtt{and}^{\mathfrak{K}}_{\boldsymbol{\theta}}(\mathrm{B}, d, (\{\mathrm{X}/\mathrm{Q}, \mathrm{Y}/\mathrm{UK}\}, \hat{S}_\rho)), \ldots]$$

by first unifying the goal G with the head of the rule H, and then proving the sub-goals in the body of the rule by using `and`.

A.3 AND MODULE

The `and` module recursively tries to prove a list of sub-goals, by invoking the `or` module. Its signature is $\mathcal{L} \times \mathbb{N} \times \mathcal{S} \rightarrow \mathcal{S}^N$, where $\mathcal{L}$ is the domain of lists of atoms, and N is the number of possible output proof states for a list of atoms with a known structure and a provided KB. This module is implemented as:

1. $\texttt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(_, _, \texttt{FAIL}) = \texttt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(_, 0, _) = \texttt{FAIL}$
2. $\texttt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}([\,], _, S) = S$
3. $\texttt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathrm{G} : \mathbb{G}, d, S) = [S'' \mid S'' \in \texttt{and}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathbb{G}, d, S'), S' \in \texttt{or}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\text{substitute}(\mathrm{G}, S_\psi), d-1, S)]$

where substitute is an auxiliary function that applies substitutions to variables in an atom whenever possible. Line 3 defines the recursion—the first sub-goal is proven by instantiating an `or` module after substitutions are applied, and every resulting proof state is used for proving the remaining sub-goals by instantiating an `and` module.

A.4 PROOF AGGREGATION

Finally, NTPs define the overall success score of proving a goal G using a KB $\mathfrak{K}$ with parameters $\boldsymbol{\theta}$ as:

$$\texttt{ntp}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathrm{G}, m) = \underset{\substack{S \in \texttt{or}_{\boldsymbol{\theta}}^{\mathfrak{K}}(\mathrm{G}, d, (\varnothing, 1)) \\ S \neq \texttt{FAIL}}}{\arg\max} S_\rho$$

where d is a predefined maximum proof depth, and the initial proof state is set to $(\varnothing, 1)$, denoting an empty substitution set and a proof success score of 1.

A.5 TRAINING

The model is then trained using a Leave-One-Out cross-entropy loss—by removing one fact from the KB, and predicting its score. Since this procedure only generates positive examples, negative examples are generated by corrupting positive examples, by randomly changing the entities (Nickel et al., 2016). We refer to Rocktäschel & Riedel (2017) for more information on the training procedure.

A.6 RULE LEARNING

NTPs can be used for learning *interpretable rules* from data, getting a deeper understanding of the domain. For example, consider the rule H :– B, with $\mathrm{H} = [\texttt{grandfatherOf}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\texttt{fatherOf}, \mathrm{X}, \mathrm{Z}], [\texttt{parentOf}, \mathrm{Z}, \mathrm{Y}]]$. Rocktäschel & Riedel (2017) show that it is possible to learn this rule from data by specifying a *rule template* H :– B, with $\mathrm{H} = [\boldsymbol{\theta}_{p:}, \mathrm{X}, \mathrm{Y}]$ and $\mathrm{B} = [[\boldsymbol{\theta}_{q:}, \mathrm{X}, \mathrm{Z}], [\boldsymbol{\theta}_{r:}, \mathrm{Z}, \mathrm{Y}]]$, where $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:} \in \mathbb{R}^k$. The parameters $\boldsymbol{\theta}_{p:}, \boldsymbol{\theta}_{q:}, \boldsymbol{\theta}_{r:}$ can be *learned from data*, and decoded by searching the closest representation of known predicates.

B DATASETS

We run experiments on the following datasets—also outlined in Table 5—and report results in terms of Area Under the Precision-Recall Curve (Davis & Goadrich, 2006) (AUC-PR), MRR, and HITS@m (Bordes et al., 2013).

B.1 COUNTRIES, UMLS, NATIONS

B.1.1 COUNTRIES

Countries is a dataset introduced by Bouchard et al. (2015) for testing reasoning capabilities of neural link prediction models. It consists of 244 countries, 5 regions (*e.g.* EUROPE), 23 sub-regions (*e.g.* WESTERN EUROPE, NORTH AMERICA), and 1158 facts about the neighbourhood of countries, and the location of countries and sub-regions. As in Rocktäschel & Riedel (2017), we randomly split

Predicate Name	Mentions
$\texttt{locatedIn}(a, b)$	a is located in b, a is situated in b, a is placed in b, a is positioned in b, a is sited in b, a is currently in b, a can be found in b, a is still in b, a is localized in b, a is present in b, a is contained in b, a is found in b, a was located in b, a was situated in b, a was placed in b, a was positioned in b, a was sited in b, a was currently in b, a used to be found in b, a was still in b, a was localized in b, a was present in b, a was contained in b, a was found in b
$\texttt{neighborOf}(a, b)$	a is adjacent to b, a borders with b, a is butted against b, a neighbours b, a is a neighbor of b, a is a neighboring country of b, a is a neighboring state to b, a was adjacent to b, a borders b, a was butted against b, a neighbours with b, a was a neighbor of b, a was a neighboring country of b, a was a neighboring state to b

Table 4: Mentions used for replacing a varying number of training triples in the Countries S1, S2, and S3 datasets.

countries into a training set of 204 countries (train), a development set of 20 countries (validation), and a test set of 20 countries (test), such that every validation and test country has at least one neighbour in the training set. Subsequently, three different task datasets are created, namely **S1**, **S2**, and **S3**. For all tasks, the goal is to predict $\texttt{locatedIn}(c, r)$ for every test country c and all five regions r, but the access to training atoms in the KB varies.

S1: All ground atoms $\texttt{locatedIn}(c, r)$, where c is a test country and r is a region, are removed from the KB. Since information about the sub-region of test countries is still contained in the KB, this task can be solved by using the transitivity rule $\texttt{locatedIn}(\mathrm{X}, \mathrm{Y}) \text{:–} \texttt{locatedIn}(\mathrm{X}, \mathrm{Z}), \texttt{locatedIn}(\mathrm{Z}, \mathrm{Y})$.

S2: In addition to **S1**, all ground atoms $\texttt{locatedIn}(c, s)$ are removed where c is a test country and s is a sub-region. The location of countries in the test set needs to be inferred from the location of its neighbouring countries: $\texttt{locatedIn}(\mathrm{X}, \mathrm{Y}) \text{:–} \texttt{neighborOf}(\mathrm{X}, \mathrm{Z}), \texttt{locatedIn}(\mathrm{Z}, \mathrm{Y})$. This task is more difficult than **S1**, as neighbouring countries might not be in the same region, so the rule above will not always hold.

S3: In addition to **S2**, also all ground atoms $\texttt{locatedIn}(c, r)$ are removed where r is a region and c is a country from the training set training that has a country from the validation or test sets as a neighbour. The location of test countries can for instance be inferred using the rule $\texttt{locatedIn}(\mathrm{X}, \mathrm{Y}) \text{:–} \texttt{neighborOf}(\mathrm{X}, \mathrm{Z}), \texttt{neighborOf}(\mathrm{Z}, \mathrm{W}), \texttt{locatedIn}(\mathrm{W}, \mathrm{Y})$.

B.1.2 Countries with Mentions

We generated a set of variants of Countries S1, S2, and S3, by randomly replacing a varying number of training set triples with mentions. The employed mentions are outlined in Table 4.

B.1.3 Nations and UMLS

Furthermore, we consider the Nations, and the Unified Medical Language System (UMLS) datasets (Kok & Domingos, 2007). UMLS contains 49 predicates, 135 constants and 6529 true facts, while Nations contains 56 binary predicates, 111 unary predicates, 14 constants and 2565 true facts. We follow the protocol used by Rocktäschel & Riedel (2017) and split every dataset into training, development, and test facts, with a $80\%/10\%/10\%$ ratio. For evaluation, we take a test fact and corrupt its first and second argument in all possible ways such that the corrupted fact is not in the original KB. Subsequently, we predict a ranking of the test fact and its corruptions to calculate MRR and HITS@m.

Note that neither Countries nor UMLS and Nations have mentions. For such a reason, for each of these datasets, we generated one equivalent mention—using a natural language sentence rather than a predicate name—and replaced a varying amount of training set triples with equivalent mentions.

Table 5: Dataset statistics.

Dataset Name	#Rel.	#Entities	#Train	#Validation	#Test	#Mentions
FB15k-237.E (Toutanova et al., 2015)	237	27,395	272,115	17,535	20,466	3,978,014
WN18 (Bordes et al., 2013)	18	40,943	141,442	5,000	5,000	—
WN18RR (Dettmers et al., 2018)	11	40,943	86,835	3,034	3,134	—

B.2 WORDNET

WN18 (Bordes et al., 2013) is a subset of WordNet (Miller, 1995), a lexical KB for the English language, where entities correspond to word senses and relationships define lexical relations between them. We also consider WN18RR (Dettmers et al., 2018), a dataset derived from WN18 where predicting missing links is sensibly harder.

B.3 FREEBASE

For evaluating the impact of also using mentions, we use the FB15k-237 dataset (Toutanova & Chen, 2015; Toutanova et al., 2015), a subset of FB15k (Bordes et al., 2013) that excludes redundant relations and direct training links for held-out triples, with the goal of making the task more realistic. Textual relations for FB15k-237 are extracted from 200 million sentences in the ClueWeb12 corpus, coupled with Freebase mention annotations (Gabrilovich et al., 2013), and include textual links of all co-occurring entities from the KB set. After pruning [2], there are 2.7 million unique textual relations that are added to the KB. The number of relations and triples in the training, validation, and test portions of the data are given in Table 5. In particular, FB15k-237.E denotes the FB15k-237 dataset augmented with textual relations proposed by Toutanova et al. (2015).

B.4 WORDNET AND FREEBASE STATISTICS

C ADDITIONAL RESULTS

C.1 EVALUATION PERFORMANCE WITH EVALUATION FROM ROCKTÄSCHEL & RIEDEL (2017)

[2]The full set of 37 million textual patterns connecting the entity pairs of interest was pruned based on the count of patterns and their tri-grams, and their precision in indicating that entity pairs have KB relations.

[3]The results for NTP reported in Table 3 are based on those reported by Rocktäschel & Riedel (2017). We discovered hours before submission that to obtain these results, the authors use the ordering of the gold-standard facts for ranking facts with the same score, which requires privileged information not available to the model. We use lexical ordering instead, which is a fairer comparison metric. Time did not permit to re-run the original NTP code the authors provide, to provide properly standardised results against which to compare, but these will be presented in an updated version of the paper.

Table 6: Results for NaNTP from Table 3 using the optimistic evaluation measure from Rocktäschel & Riedel (2017). Please note that these results are not based on a fair metric and are here just to testify to our statement that the evaluation in Rocktäschel & Riedel (2017) is too optimistic.

Datasets		**Metrics**	**Models**			
			NTP^3	**NaNTP**		
				k=1	k=2	k=5
Countries	S1	AUC-PR	90.83 ± 15.4	99.2 ± 1.19	97.34 ± 3.86	96.37 ± 4.23
	S2		87.40 ± 11.7	93.48 ± 3.29	88.48 ± 5.87	83.56 ± 7.78
	S3		56.68 ± 17.6	85.24 ± 5.83	82.18 ± 9.11	72.68 ± 13.4
Kinship		MRR	0.6^0	0.68 ± 0.04	0.73 ± 0.03	0.73 ± 0.01
		HITS@1	0.48^0	0.6 ± 0.04	0.61 ± 0.04	0.61 ± 0.02
		HITS@3	0.7^0	0.7 ± 0.05	0.83 ± 0.03	0.83 ± 0.01
		HITS@10	0.78^0	0.86 ± 0.03	0.95 ± 0.01	0.96 ± 0.0
Nation		MRR	0.75^0	0.78 ± 0.02	0.72 ± 0.03	0.59 ± 0.07
		HITS@1	0.62^0	0.69 ± 0.03	0.59 ± 0.04	0.42 ± 0.09
		HITS@3	0.86^0	0.84 ± 0.02	0.81 ± 0.02	0.71 ± 0.06
		HITS@10	0.99^0	0.99 ± 0.0	0.99 ± 0.01	0.98 ± 0.01
UMLS		MRR	0.88^0	0.73 ± 0.06	0.88 ± 0.02	0.85 ± 0.05
		HITS@1	0.82^0	0.63 ± 0.08	0.81 ± 0.02	0.77 ± 0.06
		HITS@3	0.92^0	0.81 ± 0.04	0.95 ± 0.01	0.92 ± 0.03
		HITS@10	0.97^0	0.92 ± 0.02	0.98 ± 0.01	0.98 ± 0.01