Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks

Rajarshi Das, Arvind Neelakantan, David Belanger, Andrew McCallum
{rajarshi,arvind,belanger,mccallum}@cs.umass.edu

Abstract

Our goal is to combine the rich multi-step inference of symbolic logical reasoning together with the generalization capabilities of vector embeddings and neural networks. We are particularly interested in complex reasoning about the entities and relations in knowledge bases. Recently Neelakantan et al. (2015) presented a compelling methodology using recurrent neural networks (RNNs) to compose the meaning of relations in a Horn clause consisting of a connected chain. However, this work has multiple weaknesses: it accounts for relations but not entities; it limits generalization by training many separate models; it does not combine evidence over multiple paths. In this paper we address all these weaknesses, making key strides towards our goal of rich logical reasoning with neural networks: our RNN leverages and jointly trains both relation and entity type embeddings, we train a single high-capacity RNN to compose Horn clause chains across all predicted relation types; we demonstrate that pooling evidence across multiple chains can dramatically improve both speed of training and final accuracy. We also explore multi-task training of entity and relation types. On a large dataset from ClueWeb and Freebase our approach provides a significant increase in mean average precision from 55.3% to 73.2%.

1 Introduction

As distributed representations and neural networks continue to provide dramatic new results, there is a rising interest in extending neural network to perform more complex reasoning, formerly addressed only by symbolic and logical reasoning systems. To achieve this goal, a variety of techniques have been proposed to perform neural reasoning on sentences and knowledge bases (KBs). Bowman et al. (2014 ;2015) propose methods for recognizing textual entailment, where the goal is to predict whether a pair of sentences are entailing, contradictory or neutral. Chains of Reasoning (Neelakantan et al., 2015), Neural Programmer (Neelakantan et al., 2016) and Neural Theorem Provers (Rocktäschel and Riedel, 2016) learn to perform reasoning on using neural networks with minimal supervision about the underlying logical structure of the problem.

Knowledge Bases (KBs) are structured knowledge sources that support a wide variety of queries about entities and their relations. We focus on inference through arbitrary length horn clauses forming. For example, we want to infer the relation LivesIn(Melinda Gates, Seattle) from the facts Chair(Melinda Gates, Gates Foundation) and HQIn(Gates Foundation, Seattle). In other words, we want a model that can learn to predict new relations between entity pairs using the paths connecting them. For example, LivesIn(A, B) is implied, with some probability, from Chair(A, X) and HQIn(X, B). This kind of reasoning would enable us to infer new or missing facts from the graph of connected relations and entities (KBs are usually incomplete (Min et al., 2013)). Early work extracts purely symbolic rules. For example, Schoenmackers et al. (2010) learn inference chain rules by exhaustively exploring relational paths of increasing length, and select those with high accuracy. (A “path” is a sequence of triples in which the second entity of each triple
matches the first entity of the next triple.) Since then, there have been a variety of machine learning approaches.

The Path Ranking Algorithm (PRA) (Lao et al., 2011) replaces exhaustive search by random walks. Observed paths are used as features for a per-target-relation binary classifier. Lao et al. (2012) extend PRA by augmenting KB-schema relations with observed text patterns (also known as OpenIE relations (Banko et al., 2007)). However, these methods do not generalize well to millions of distinct paths obtained from random exploration of the KB, since each unique path is treated as a singleton, where no commonalities between paths are modeled. In response, pre-trained vector representations have been used in PRA to tackle the feature explosion (Gardner et al., 2013, Gardner et al., 2014) but still rely on a classifier using atomic path features.

Neelakantan et al. (2015) propose composing the relations occurring in a path using a recurrent neural network (RNN-path). This enables reasoning about paths unseen at train time and shares statistical strength among similar paths. Its has many shortcomings, however: it ignores all information about entities on the path, it trains separate per-relation models, and it has simplistic methods for pooling evidence across paths.

This paper introduces a collection of modeling improvements that address the weaknesses of Neelakantan et al. (2015) and provides extensive experiments exploring the design space. These result in significant performance increases and represent fundamental progress in using neural networks and distributed representations for KB reasoning.

First, note that the RNN-path model (and other existing path-based approaches) consider only the relations in the path, ignoring the entities that it passes through. In Figure 1, reasoning using only information about relation edges would score both paths equally, although the latter path should receive a lesser score. In response, we incorporate entity information by having the RNN take as both entities and relations in the path. We consider two ways to represent the entity: (a) a separate embedding for every entity in the training set, and (b) as a function of its annotated types in the KB. Empirically, the latter approach works better, primarily because it generalizes to unseen entities at test time.

We also employ multi-task learning (Caruana, 1997) to help reduce the effect of data scarcity. We have millions of examples of entities annotated with their types in the KB. Relation extraction accuracy on infrequent relation types can be improved by sharing parameters between relation extraction and entity type prediction models. Additional parameter sharing is achieved by using a shared mapping from relation paths to vector embeddings for each target relation class, unlike Neelakantan et al. (2015) who employ completely independent models for each of the 46 target classes.

Finally, we note that there are many paths connecting two entities and previous work made predictions by selecting just a single path. We experiment with multiple pooling strategies that combines evidences of multiple paths and find that replacing max pooling with the log-sum-exp function provides both improved accuracy and faster training, since it yields dense gradients.

Overall, these techniques improve our mean average precision on 46 relation types from 55.3 to 73.26. The data and code for all our experiments are available at https://github.com/rajarshd/ChainsOfReasoning.
2 Background

The RNN-path model (Neelakantan et al., 2015) reasons on the path connecting an entity pair to infer new relations between them. Reasoning is performed non-atomically about conjunctions of relations in an arbitrary length path by composing them with a RNN. Figure 2 shows an example where the relations in a path between the entities Microsoft and USA are composed via a RNN. The representation of the path is given by the last hidden state of the RNN obtained after processing the entire path.

Let \((e_1, e_2)\) be an entity pair and \(S\) denote the set of all paths connecting them. To model the probability of \((e_1, e_2)\) participating in relation \(r\), first, the RNN-path model consumes only the relations in the path \(\pi = \{e_1, r_1, e_2, r_2, \ldots, r_N, e_{N+1}\} \in S\), forming an intermediate representation \(h_t\) after consuming \(r_t\) at step \(t = \{1, 2, \ldots, N\}\), given by

\[
h_t = f(W^r_{hh} h_{t-1} + W^r_{rh} v^r_{rel}(r_t))
\]

where \(v^r_{rel}(r) \in \mathbb{R}^d\) denotes the vector representation of relation \(r\), \(W^r_{hh}\) and \(W^r_{rh}\) are the parameters of the RNN and \(f\) is the sigmoid non-linear activation function. The vector representation of the path \(\pi (v_p(\pi) \in \mathbb{R}^d)\) is the last RNN hidden state \(h_N\).

The similarity of a path with the query relation is computed by taking the dot product between the vector representation of the path and the query relation.

\[
score(\pi, r) = (v_p(\pi) \cdot v^r_{rel}(r)), \forall \pi \in S
\]

Score Pooling

An entity pair has several paths connecting them in the knowledge graph. In RNN-path model, the probability that the entity pair \((e_1, e_2)\) participates in the query relation \(r\) is given by,

\[
P(r|e_1, e_2) = \max \sigma(score(v_p(\pi), v_r)), \forall \pi \in S
\]

where \(\sigma\) is the sigmoid function. Hence, during training only the parameters associated with the maximum scoring path receive a non-zero gradient update.

3 Modeling Approach

In this section, we propose several extensions to address the shortcomings of the RNN-path model.

3.1 Single Model

A knowledge graph such as Freebase has thousands of relation types and hence per-relation modeling approaches (Gardner et al., 2013; Gardner et al., 2014; Neelakantan et al., 2015) are impractical. Moreover, parameters are not shared across multiple target relation types leading to large number
of parameters to be learned from the training data. To alleviate these problems, the composition matrices of the recurrent neural network and representations of the relations are shared across all target relations enabling lesser number of parameters for the same training data. We refer to this model as Single Model.

The RNN hidden state (in equation (1)) is now given by,

$$ h_t = f(W_{hh}h_{t-1} + W_{rh}v_{rel}(r_t)) \quad (4) $$

Readers should take note that the parameters here are independent of each target relation $r$.

### 3.2 Model Training

We train the model along the same lines of RNN-path model, using existing observed facts (triples) in the KB as positive examples and unobserved facts as negative examples.

Let $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ denote the set of all relation types that we train for. Let $\Delta^+_r, \Delta^-_r$ denote the set of positive and negative triples for all the relation types in $\mathcal{R}$. The parameters of the model are trained to minimize the binary cross-entropy loss ($L$) between the predicted value and the true label.

$$ L(\Theta, \Delta^+_R, \Delta^-_R) = -\frac{1}{M} \sum_{e_1, e_2, r \in \Delta^+_R} \mathbb{P}(r|e_1, e_2) + \sum_{\hat{e}_1, \hat{e}_2, \hat{r} \in \Delta^-_R} (1 - \mathbb{P}(\hat{r}|\hat{e}_1, \hat{e}_2)) \quad (5) $$

Here $M$ is the total number of training examples and $\Theta$ denotes the set of all parameters of the model (lookup table of embeddings (shared) and parameters of the RNN (shared)). It should be noted that the RNN-path model had a separate loss function for each relation $r \in \mathcal{R}$ and each loss function would depend only on the subset of the data $\{\Delta^+_r, \Delta^-_r\}$ where,

$$ \Delta^+_r = \{(e_1, e_2, r) | (e_1, e_2, r) \in \Delta^+_R, \forall e_1, e_2\} $$

### 3.3 Score Pooling

The RNN-path model considers only the maximum scoring path between an entity pair. Not only this is a waste of computation (since we have to compute a forward pass for all the paths anyway), the relations in all other paths do not get any gradients updates during training. Also during initial stages of training, the maximum probable path will be random. To address these, we introduce new methods of pooling scores across paths that takes into account information from multiple paths. Let $\{s_1, s_2, \ldots, s_N\}$ be the similarity scores (Equation (2)) for $N$ paths connecting an entity pair $(e_1, e_2)$. The probability for entity pair $(e_1, e_2)$ to participate in relation $r$ (Equation (3)) is now given by,

1. **Top-(k):** A straightforward extension of the ‘max’ approach in which we consider the top $k$ scoring paths. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be the top-$k$ scores.

   $$ \mathbb{P}(r|e_1, e_2) = \sigma\left(\frac{1}{k} \sum_{i=1}^{k} \alpha_i\right) $$

2. **Average:** Here, the final score is the average of scores of all the paths.

   $$ \mathbb{P}(r|e_1, e_2) = \sigma\left(\frac{1}{N} \sum_{i=1}^{N} s_i\right) $$

3. **LogSumExp:** In this approach the pooling layer is a smooth approximation to the ‘max’ function - LogSumExp (LSE). Given a vector of scores, $s_1, s_2, \ldots, s_n$, the LSE is calculated as

   $$ LSE(s_1, s_2, \ldots, s_n) = \log(\sum_i \exp(s_i)) $$

   and hence the probability of the triple is,

   $$ \mathbb{P}(r|e_1, e_2) = \sigma(LSE(s_1, s_2, \ldots, s_n)) $$

### 3.4 Incorporating Entities

The RNN-path model and other multi-hop relation extraction approaches ignore the entities in the path. Figure [1] motivates the need for including entity information in relation extraction. We consider two ways to represent the entity: (a) learning a separate representation for every entity and (b) representing an entity using its annotated entity types. We incorporate entity information in equation(1) as follows
In other words we maximize the following objective, which seeks to rank the type of examples of the form of equation (5). Given a training set \( T \) of examples, we employ the Bayesian Personalized Ranking scheme. It was popularized for neural network approaches to NLP by Collobert et al. (2011). Recent works using multi-tasking include (Luong et al., 2011) and (Klerke et al., 2016).

\[
O(T, \Theta_{\text{type}}) = \sigma(score_{\text{type}}(e, t) - score_{\text{type}}(e, \hat{t}))
\]

\[\hat{t}\) is a randomly sampled type. \(score_{\text{type}}(e, t)\) is defined as the dot product of the entity and the embedding of the type, \(score_{\text{type}}(e, t) = v_e(e) \cdot v_t(t)\). In our experiments, we share the entity type parameters for this task along with the relation extraction task using Single Model + Type. During training, we shuffle all the available training example at each step the model performs gradient updates w.r.t one of equations (5) or (8). A batch of examples for entity type prediction is sampled w.r.t Bernoulli \(p\).

We initially set \(p\) to 0.5 and anneal it at a constant rate of \(2e^{-6}\), until it reaches 0.1 after \(200K\) steps.

## 4 Related Work

Many KB construction methods perform reasoning on individual relations (single-hop reasoning) to predict new relations (Mintz et al., 2009) [Bordes et al., 2013] [Riedel et al., 2013] [Socher et al., 2013a). For example, predicting \textit{Nationality}(X, Y) from \textit{BornIn}(X, Y). In some cases, such as Riedel et al. (2013) the graph is augmented with relations corresponding to observed surface patterns. The performance of relation extraction methods have been improved by incorporating \textit{selectional preferences}; i.e., relations enforce constraints on the allowed entity types for their candidate entities, both in sentence level (Roth and Yih, 2007) [Singh et al., 2013] and KB relation extraction (Chang et al., 2014), and in learning entailment rules (Berant et al., 2011).

Multi-hop reasoning can be performed by chaining together single-hop decisions. Alternatively, the PRA papers, discussed in the introduction, represent multi-hop KB paths as atomic units, but do not model compositionality along paths. Our use of LogSumExp pooling is motivated by the empirical improvements it provides in the work of Pathak et al. (2015).

Multi-task learning is a very useful tool in NLP, particularly when leveraging deep architectures, since they provide various parameter sharing schemes. It was popularized for neural network approaches to NLP by Collobert et al. (2011). Recent works using multi-tasking include (Luong et al., 2016) and (Klerke et al., 2016).

Recently, Guu et al. (2015) introduced a new pathwise training procedure for multi-hop KB reasoning. Every path introduces multiple training
The dimension of the relation type representations is \( d = 250 \) and the entity embeddings and entity type embeddings have \((m = 50)\) dimensions. The RNN-path model has sigmoid units as their activation function. However, we found rectifier units (ReLU) to work much better \( \text{[Le et al., 2015]} \). For each entity, we consider up to a maximum of \( l = 7 \) types. As our evaluation metric, we employ the average precision (AP) to score the ranked list of entity pairs (sorted in decreasing order of scores) for each relation. The %MAP score is the mean average precision across all \( Q \) query relations \( \text{[Manning et al., 2008]} \). We use Adam \( \text{[Kingma and Ba, 2014]} \) for optimization with the default hyperparameter settings.

### Table 1: Statistics of the dataset.

| Stats                      | #        |
|----------------------------|----------|
| # test relations           | 46       |
| # entities                 | 1.59M    |
| # entity pairs             | 3.22M    |
| # entity pairs (test)      | 2M       |
| # unique entity types      | 2218     |
| # Freebase relation types  | 27791    |
| # textual relation types   | 23599    |
| Avg. path length           | 4.7      |
| Max path length            | 16       |
| Total # paths              | 191M     |

### 5 Experimental Setup

We apply our models to the dataset released by Neelakantan et al. \( \text{[2015]} \) which is a subset of Freebase enriched with information from ClueWeb. The dataset comprise of a set of triples \((e_1, r, e_2)\) and also the set of paths connecting the entity pair \((e_1, e_2)\) in the knowledge graph. The triples extracted from ClueWeb consists of sentences that contained entities linked to Freebase \( \text{[Orr et al., 2013]} \). The phrase between the two entities in the sentence forms the relation type. For direct comparison with Neelakantan et al \( \text{[2015]} \), we also evaluate on their data for the same 46 Freebase relation types. However the paths in the dataset had the entity information missing from them and only contains the relation types occurring in them. For example, consider the path \( \text{SatyaNadella} \xrightarrow{\text{ceoOf}} \text{Microsoft} \xrightarrow{\text{locatedIn}} \text{Seattle} \). The original dataset had the entities in-between such as \( \text{Microsoft} \) and \( \text{Seattle} \) missing from it. We augment the dataset with the entities present in the paths. To gather the entities, we do a depth first traversal in the Freebase knowledge graph starting from the first entity of the entity pair and following the relation types until we reach the last entity of the pair. In cases of one-to-many relations we choose the next entity to be traversed at random. For the entities in our dataset, we also collect the entity type information from Freebase Table 1 summarizes some important statistics.

6.1 Path Pruning

Before discussing performance improvements resulting from the modeling techniques of Section 3 we describe a simple data processing step that improves performance from 55.30 \( \text{[Neelakantan et al., 2015]} \) to 68.77.

The maximum path length in the dataset is 16, but for preliminary models we found that high scoring paths had an average length of 4.52. In response, we only consider paths with length below a threshold \( \tau \). We experiment with values of \( \tau = \{3, 5, 8, 16\} \).

### 6.2 Effect of Pooling Techniques

Section 2 of Table 2 shows the effect of the various pooling techniques presented in section 3.3. It is encouraging to see that \( \text{LogSumExp} \) gives the best results. This demonstrates the importance of considering information from all the paths. However, Avg. pooling performs the worst, which shows that it is also important to weigh the paths scores according to their values. Figure 3 plots the training loss w.r.t gradient update step. Due to non-zero gradient updates for all the paths, the LogSumExp pooling strategy leads to faster training vs. max pooling, which
Table 2: Performance of a variety of proposed methods. The first section shows the result of restricting the maximum length of paths (§6.1). The maximum path lengths are in parentheses. All the subsequent results have the maximum path length limited to 8. The second subsection shows the effectiveness of LogSumExp as the score aggregation function (§6.2). The third section compares performance with existing multi-hop approaches (§6.3) and the last section shows the performance achieved using joint reasoning with entities and types (§6.4).

Figure 3: Comparison of the training loss w.r.t gradient update steps of various pooling methods. The loss of LogSumExp decreases the fastest among all pooling methods and hence leads to faster training.

6.3 Comparison with multi-hop models
We next compare the performance of the Single Model with two other multi-hop models - RNN-path and PRA (Lao et al., 2011). Both of these approaches train an individual model for each query relation. Neelakantan et al. (2015) also experiment with another extension of PRA that adds bigram features (PRA + Bigram). Additionally, we run an experiment replacing the max-pooling of RNN-path with LogSumExp. The results are shown in the third section of Table 2. It is not surprising to see that the Single Model, which leverages parameter sharing such that each relation effectively has access to more training data, improves performance. It is also encouraging to see that LogSumExp makes the RNN-path baseline stronger.

6.4 Effect of Incorporating Entities
Next, we provide quantitative results supporting our claim that modeling the entities along a KB path can improve reasoning performance. The last section of Table 2 lists the performance gain obtained by injecting information about entities. We achieve the best performance when we represent entities as a function of their annotated types in Freebase (Single Model + Types). In comparison, learning separate
representations of entities (Single Model + Entities) gives slightly worse performance. This is primarily because we encounter many new entities during test time, for which our model does not have a learned representation. However the relatively limited number of entity types helps us overcome the problem of representing unseen entities. We also extend PRA to include entity type information (PRA + Types), but this did not yield improvements.

6.5 Multitask Learning with Entity Type prediction

This section presents experiments designed to test whether our performance on infrequent relation types can be improved via parameter sharing and multi-tasking between relation extraction and entity type prediction tasks.

KBs are dynamic, with new relations being continuously added to the schema. Naturally, we have very little supervision for these new relations. Models like RNN-path would fail for such relations, since each relation-specific model needs to be trained on a reasonable amount of data. The Single Model can perform better because of parameter sharing with other relations for which we have significant amount of data.

We create a new dataset by randomly selecting 23 of the 46 relations and removing all but 1% of positive and negative triples for each relation (50 triples each). We compare the performance of two models - (a) Single Model + Type (§3.4). (b) Single model which additionally shares its entity type parameters for multitasking with entity type prediction (Single Model + MTL; §3.5). Table 3 shows the benefits of multitask learning for relation types for which we have very little data. Leveraging the entity type information present in the KB, our model helps alleviate data scarcity of infrequent relation types. We also find that the performance of entity type prediction doesn’t suffer at all. This is surprising, since we choose to weigh relation type prediction more than entity type prediction in our multi-task loss.

6.6 Zero-Shot Relation Extraction

In this experiment, we test whether our model can generalize to relations which it hasn’t been explicitly trained to predict. In other words, the model is trained without triples containing the query relation. The embeddings of the unseen target relations are updated during training when they occur in the paths connecting entity pairs. This is in contrast to the zero-shot setting of (Neelakantan et al., 2015), in which they use pre-trained relation embeddings and do not update them during training. In zero-shot learning, the goal is to learn to recognize new concepts by having a only description of them (Larochelle et al., 2008; Palatucci et al., 2009; Socher et al., 2013b; Romera-Paredes and Torr, 2015).

Table 4 shows the performance of different models in the zero shot setting. By sharing parameters across relations, we can see that our model can predict relations without being explicitly trained to predict them. The performance of the model that additionally uses entity type information is lower in this experiment.

6.7 Performance of RNN vs. LSTM

Long Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are common sequence models that have been shown to better model long term dependencies than simple RNNs. They can also alleviate the vanishing gradient problem (Bengio et al., 1994). However, for our task

| Model                  | Performance (%MAP) |
|------------------------|--------------------|
| RNN-path               | 22.06              |
| Single Model + Type    | 63.33              |
| Single Model + MTL     | 64.81              |

Table 3: The shared parameter architecture of Single Model drastically improves the performance for infrequent relation types by sharing relation embeddings with other query relations. The performance is further improved by multi-task learning with an entity type prediction task, using shared entity type representations.

| Model                  | Performance (%MAP) |
|------------------------|--------------------|
| Single Model           | 29.54              |
| Single Model + Type    | 26.79              |
| Random                 | 7.39               |

Table 4: Zero-shot results. In this setting, we hold-out training triples from 20 relations and evaluate performance on them.
Relation: /aviation/airport/serves (Does the airport serve the location?)

Baseline Path: (0.5174)

Sandy_Lake_Airport (location/contains)→Ontario and at the Toronto_Royal_Conservatory_Of_Music (including the)→Canada (geography/lake/basin/countries)→Big_Troul_Lake and to Sandy_Lake_First_Nation.

Our Model Path: (0.9502)

Sandy_Lake_Airport (location/contains)→Ontario (in northwestern)→Sandy_Lake_First_Nation.

Relation: /aviation/airport/serves

Baseline Score: (0.4348), Our Model Score: (0.9731) (Same path chosen by both models)

St._Mary’s_Airport (location/contains)→Wade_Hampton_Census_Area (location/us/county/hud/county/place)→St._Mary’s

Relation: /education/campus/institution (Is the educational institution located in this campus?)

Baseline Path: (0.4869)

Gray’s_Inn London (people/person/nationality)→Roger_Fry (people/deceased_person/place_of_death)→London /location/contains→City_Law_School

Our Model Path: (0.9676)

Gray’s_Inn (location/contains)→London_Borough_of_Camden /location/contains→City_Law_School

Relation: /geography/river/mouth (Does the river (tributary) flow into the other river?)

Baseline Path: (0.4578)

Gard_River /geography/river/basin/countries→Romania /geography/river/basin/countries→Jijia_River

Our Model Path: (0.9231)

Gard_River (location/contains)→Botosani_County /location/contains→Jijia_River

Table 5: Predictive paths chosen by the baseline and our model for four entity pair and relation triples. The relations are edge labels and the entities occur in between them and at the ends. The freebase relations starts with ‘/’, (location/contains, for e.g.). Inverse relations are denoted by $^{-1}$ i.e. $r(x,y) \implies r^{-1}(y,x), \forall (x,y) \in r$. The scores are given in parentheses (higher is better). Sometimes, both models find the same path (second example in /aviation/airport/serves), but we often find that our model correctly scores it higher. The reader can browse more examples at [http://people.cs.umass.edu/~rajarshi/paths.html](http://people.cs.umass.edu/~rajarshi/paths.html).

6.8 Predictive Paths

Table 5 shows maximum scoring paths for four entity pair and freebase relation triples chosen by the baseline and our model. We often find that the paths chosen by the baseline model have noisier textual relation, (like ‘London’, ‘and at the’) and have entities belonging to very different types than expected by the query relation. For example, in table 5 the path chosen by the baseline model for ‘/aviation/airport/serves’ goes to a music education school, and a water body and for ‘/education/campus/institution’, it goes to a country in which the institution is situated followed by a notable person in the country (unrelated to the query relation). We also find that sometimes both models finds the same max scoring path but our model assigns more confidence (higher scores) to them leading to better MAP scores.

7 Conclusion

This paper introduces a single high capacity RNN model which allows chains of reasoning across mul-

\textsuperscript{1}The freetext relation is different from the entity ‘London’ also occurring in the path

\textsuperscript{2}The reader can browse more examples at [http://people.cs.umass.edu/~rajarshi/paths.html](http://people.cs.umass.edu/~rajarshi/paths.html)
multiple relation types. It also leverages information from the intermediate entities present in the path between an entity pair and mitigates the problem of unseen entities by representing them as a function of their annotated types. We also demonstrate that pooling evidence across multiple paths improves both training speed and accuracy. Finally, we also address the problem of reasoning about infrequently occurring relations and show huge performance gains via multitasking with an auxiliary task of entity type prediction.

References

[Banko et al.2007] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In IJCAI.

[Bengio et al.1994] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. Trans. Neur. Netw.

[Berant et al.2011] Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In NAACL.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In NIPS.

[Caruana1997] Rich Caruana. 1997. Multitask learning. Machine Learning.

[Chang et al.2014] Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In EMNLP.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. J. Mach. Learn. Res.

[Gardner et al.2013] Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M. Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In EMNLP.

[Gardner et al.2014] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In EMNLP.

[Guu et al.2015] K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In EMNLP.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Comput.

[Kingma and Ba2014] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

[Klerke et al.2016] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. ACL.

[Lao et al.2011] Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In EMNLP, Stroudsburg, PA, USA.

[Lao et al.2012] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In EMNLP.
[Larochelle et al.2008] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In AAAI.

[Le et al.2015] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. CoRR.

[Luong et al.2016] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. ICLR.

[Manning et al.2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval.

[Min et al.2013] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In NAACL.

[Mintz et al.2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In ACL.

[Neelakantan et al.2015] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In ACL, Beijing, China.

[Neelakantan et al.2016] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural Programmer: Inducing latent programs with gradient descent. In ICLR.

[Orr et al.2013] Dave Orr, Amar Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 2013. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html.

[Palatucci et al.2009] Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In NIPS.

[Pathak et al.2015] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. 2015. Constrained convolutional neural networks for weakly supervised segmentation. In ICCV.

[Rendle et al.2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. UAI ’09.

[Riedel et al.2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In NAACL.

[Rocktäschel and Riedel2016] Tim Rocktäschel and Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In AKBC, NAACL.