Path Ranking with Attention to Type Hierarchies

Weiyu Liu, Angel Daruna, Zsolt Kira, Sonia Chernova
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology, Atlanta, Georgia
{wliu88, adaruna3, zkira}@gatech.edu, chernova@cc.gatech.edu

Abstract

The objective of the knowledge base completion problem is to infer missing information from existing facts in a knowledge base. Prior work has demonstrated the effectiveness of path-ranking based methods, which solve the problem by discovering observable patterns in knowledge graphs, consisting of nodes representing entities and edges representing relations. However, these patterns either lack accuracy because they rely solely on relations or cannot easily generalize due to the direct use of specific entity information. We introduce Attentive Path Ranking, a novel path pattern representation that leverages type hierarchies of entities to both avoid ambiguity and maintain generalization. Then, we present an end-to-end trained attention-based RNN model to discover the new path patterns from data. Experiments conducted on benchmark knowledge base completion datasets WN18RR and FB15k-237 demonstrate that the proposed model outperforms existing methods on the fact prediction task by statistically significant margins of 26% and 10%, respectively. Furthermore, quantitative and qualitative analyses show that the path patterns balance between generalization and discrimination.

Introduction

Knowledge bases (KBs), such as WordNet (Miller 1995) and Freebase (Bollacker et al. 2008), have been used to provide background knowledge for tasks such as recommendation (Wang et al. 2018), and visual question answering (Aditya, Yang, and Baral 2018). Such KBs typically contain facts stored in the form of $(source\ entity, relation, target\ entity)$ triples, such as $(Fork, in, Kitchen)$. Combined, the dataset of triples is often represented as a graph, consisting of nodes representing entities and edges representing relations. Many KBs also contain type information for entities, which can be represented as type hierarchies by ordering each entity's types based on levels of abstraction.

Despite containing millions of facts, existing KBs still have a large amount of missing information (Min et al. 2013). As a result, robustly reasoning about missing information is important not only for improving the quality of KBs, but also for providing more reliable information for applications relying on the contained data. The objective of the *knowledge base completion problem* is to infer missing information from existing facts in KBs. More specifically, *fact prediction* is the problem of predicting whether a missing triple is true.

Prior work on fact prediction has demonstrated the effectiveness of path-ranking based methods (Lao, Mitchell, and Cohen 2011; Gardner and Mitchell 2015; Neelakantan, Roth, and McCallum 2015; Das et al. 2016), which solve the knowledge base completion problem by discovering observable patterns in knowledge graphs. A foundational approach in this area is the Path Ranking Algorithm (PRA) (Lao, Mitchell, and Cohen 2011), which

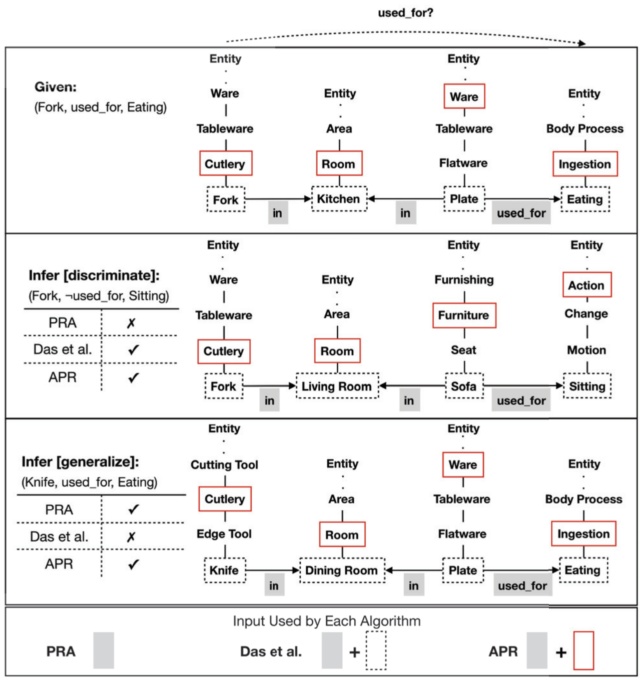


Figure 1: Paths and type hierarchies of a known triple and two missing triples. Our proposed method, Attentive Path Ranking, can correctly predict both missing triples by using attention to select types at appropriate levels of abstraction.

extracts patterns based on sequences of relations. However, relying on relation-only patterns often leads to over-generalization, as shown in the example in Figure 1. In this example, the top triple ($\langle Fork, used\_for, Eating\rangle$) and its associated information is assumed to be known, and the second ($\langle Fork, \neg used\_for, Sitting\rangle$) and third ($\langle Knife, used\_for, Eating\rangle$) triples must be inferred from available information. PRA is limited to using only relation patterns, and as such uses $\langle in, -in, used\_for\rangle$[1], observed in the path of the existing triple, to make its predictions. As the example shows, this representation enables PRA to correctly generalize to similar scenarios, such as the third triple, but the lack of context leads to over-generalization and failure to discriminate the second triple.

To improve the discriminativeness of PRA, (Das et al. 2016) extend the above approach to incorporate entity information in addition to relations within the learned paths. Paths are expanded to include entities directly, or by aggregating all types for each entity. As an example, the path pattern of the known triple in Figure 1 becomes $\langle Fork, in, Kitchen, -in, Plate, used\_for, Eating\rangle$. This new approach successfully discriminates the second triple, however, this comes at the cost of losing generalizability, resulting in misclassification of the third triple.

In our work, we introduce a novel path-ranking approach, **Attentive Path Ranking (APR)** [2], that seeks to achieve discrimination and generalization simultaneously. Our work is motivated by the distributional informativeness hypothesis (Santus et al. 2014), according to which the generality of a term can be inferred from the informativeness of its most typical linguistic contexts. Similar to (Das et al. 2016), we utilize a path pattern consisting of both entity types and relations. However, for each entity we use an *attention mechanism* (Bahdanau, Cho, and Bengio 2014) to select a single type representation for that entity at the appropriate level of abstraction, thereby achieving both generalization and discrimination. As shown in Figure 1, the APR path pattern of the known triple becomes $\langle Cutlery, in, Room, -in, Ware, used\_for, Ingestion\rangle$, which successfully discriminates from the second triple, and correctly generalizes to the third.

Our work makes the following contributions:

- We introduce an end-to-end trained attention-based RNN model for entity type selection. Our approach builds on prior work on attention mechanisms, while contributing a novel approach for learning the appropriate levels of abstraction of entities from data.
- Based on the above, we introduce Attentive Path Ranking, a novel path pattern representation that leverages type hierarchies of entities to both avoid ambiguity and maintain generalization.
- Applicable to the above, and other path-ranking based methods more broadly, we introduce a novel pooling method based on attention to jointly reason about contextually important information from multiple paths.

[1]the negative sign indicates reversed direction for the relation

[2]The code and data are available at: https://github.com/wliu88/AttentivePathRanking.git

We quantitatively validate our approach against five prior methods on two benchmark datasets: WN18RR (Dettmers et al. 2018) and FB15k-237 (Toutanova et al. 2015). Our results show statistically significant improvement of 26% on WN18RR and 10% on FB15k-237 over state-of-the-art path-ranking techniques. Additionally, we demonstrate that our attention method is more effective than fixed levels of type abstraction, and that attention-based pooling improves performance over other standard pooling techniques. Finally, we show that APR significantly outperforms knowledge graph embedding methods on this task.

Related Work

In this section, we present a summary of prior work.

Knowledge Based Reasoning

Multiple approaches to knowledge based reasoning have been proposed. Knowledge graph embedding (KGE) methods map relations and entities to vector representations in continuous spaces (Nickel et al. 2015). Methods based on inductive logic programming discover general rules from examples (Quinlan and Cameron-Jones 1993). Statistical relational learning (SRL) methods combine logics and graphical models to probabilistically reason about entities and relations (Getoor and Taskar 2007). Reinforcement learning based models treat link prediction (predicting target entities given a source entity and a relation) as Markov decision processes (Xiong, Hoang, and Wang 2017; Das et al. 2018). Path-ranking based models use supervised learning to discover generalizable path patterns from graphs (Lao, Mitchell, and Cohen 2011; Gardner and Mitchell 2015; Neelakantan, Roth, and McCallum 2015; Das et al. 2016).

In the context of knowledge base completion, we focus on path-ranking based models. The Path Ranking Algorithm (PRA) (Lao, Mitchell, and Cohen 2011) is the first proposed use of patterns based on sequences of relations. By using patterns as features of entity pairs, the fact prediction problem is solved as a classification problem. However, the algorithm is computationally intensive because it uses random walks to discover patterns and calculate feature weights. Subgraph Feature Extraction (SFE) (Gardner and Mitchell 2015) reduces the computational complexity by treating patterns as binary features, as feature weights provide no discernible benefit to the performance, and using more efficient bi-directional breadth-first search to exhaustively search for sequences of relations and additional patterns in graphs. To generalize semantically similar patterns, (Neelakantan, Roth, and McCallum 2015) use recurrent neural networks (RNNs) to create vector representations of patterns, which are then used as multidimensional features for prediction. (Das et al. 2016) improve the accuracy of the RNN method by making use of the additional information in entities and entity types. We also use entity types but we focus on using types at different levels of abstraction for different entities.

Representing Hierarchical Structures

Learning representations of hierarchical structures in natural data such as text and images has been shown to be ef-

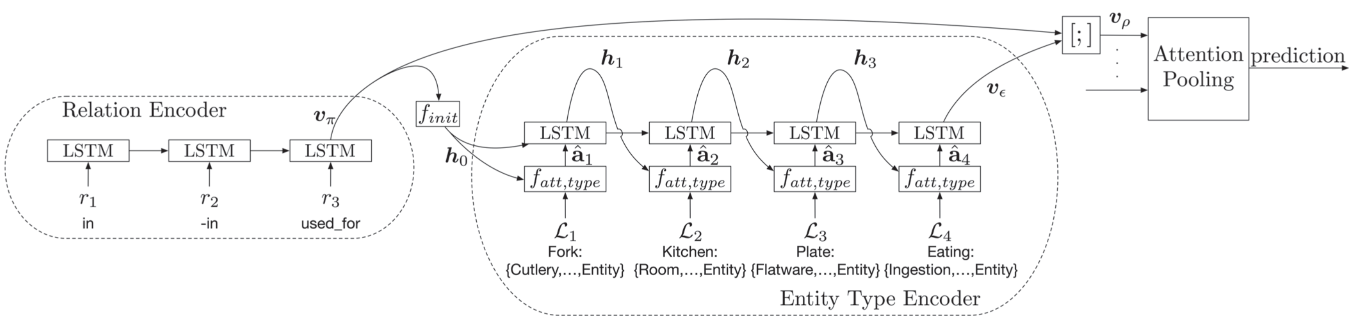


Figure 2: Relation encoder and entity type encoder with attention to type hierarchies.

fective for tasks such as hypernym classification and textual entailment. (Vendrov et al. 2016) order text and images by mapping them to a non-negative space in which entities that are closer to the origin are more general than entities that are further away. (Athiwaratkun and Wilson 2018) use density order embeddings where more specific entities have smaller, concentrated probabilistic distributions and are encapsulated in broader distributions of general entities. In this work, we do not explicitly learn the representation of hierarchical types. Instead, we leverage the fact that types in type hierarchies have different levels of abstraction to create path patterns that balance generalization and discrimination.

Attention

Attention was first introduced in (Bahdanau, Cho, and Bengio 2014) for machine translation, where it was used to enable the encoder-decoder model to condition generation of translated words to different parts of the original sentence. Later, cross-modality attention was shown to be effective at image captioning (Xu et al. 2015) and speech recognition (Chan et al. 2016). Our approach uses attention to focus on contextually important information from multiple paths, much like the above methods. More importantly, we use attention in a novel way to efficiently discover the correct levels of abstraction for entities from a large search space.

Problem Definition

A KB is formally defined as a set of triples, also called relation instances, $\mathcal{X} = \{(e_i, r_j, e_k) | e_i, e_k \in \mathcal{E} \wedge r_j \in \mathcal{R}\}$, where $\mathcal{E}$ denotes the entity set, and $\mathcal{R}$ denotes the relation set. The KB can be represented as a multi-relational graph $\mathcal{G}$, where nodes are entities and edges are relations. A directed edge from e_i to e_k with label r_j exists for each triple (e_i, r_j, e_k) in $\mathcal{X}$. A path between e_i and e_k in $\mathcal{G}$ is denoted by $p = \langle e_1, r_1, ..., r_M, e_{M+1} \rangle$, where $e_1 = e_i$ and $e_{M+1} = e_k$. The length of a path is defined as the number of relations in the path, M in this case. For all pairs of entities e_i and e_k in the graph $\mathcal{G}$, we can discover a set of N paths up to a fixed length, $\mathcal{P}_{ik} = \{p_1, ..., p_N\}$.

Our objective is, given an incomplete KB and the path set $\mathcal{P}_{ik}$ extracted from the KB, to predict whether the missing triple (e_i, r_j, e_k) is true, or equivalently whether the entity pair e_i and e_k can be linked by r_j.

Attentive Path Ranking

In this section, we present our proposed Attentive Path Ranking model, which takes as input the set of paths between entities e_i and e_k, $\mathcal{P}_{ik}$, and outputs $P(r_j | e_i, e_k)$, the probability that r_j connects e_i and e_k. Our model, shown in Figure 2, consists of three components: a relation encoder, an entity type encoder, and an attention-based pooling method. Given a path $p = \langle e_1, r_1, ..., r_M, e_{M+1} \rangle$ in $\mathcal{P}_{ik}$, the **relation encoder** encodes the sequence of relations $\langle r_1, ..., r_M \rangle$ (e.g., $\langle in, -in, used\_for \rangle$ in Figure 2). The **entity type encoder** both selects a type l_t for each entity e_t with a certain level of abstraction, and encodes the selected types $\langle l_1, ..., l_{M+1} \rangle$ (e.g., $\langle Cutlery, Room, Ware, Ingestion \rangle$). We then combine the relation and entity type encodings to form the path pattern $\hat{p} = \langle l_1, r_1, ..., r_M, l_{M+1} \rangle$. The above process is repeated for each path in $\mathcal{P}_{ik}$. **Attention-based pooling** is then used to combine all path patterns to predict the truth value of (e_i, r_j, e_k). In the following sections we present details of the three core model components.

Relation Encoder

The relation encoder uses a LSTM (Hochreiter and Schmidhuber 1997) to sequentially encode vector embeddings of relations $\boldsymbol{v}_\delta(r_t)$ for all relations in the path p. Here, the trainable vector embeddings help generalize semantically similar relations by representing them as similar vectors. The last state of the LSTM is used as the vector representation of the sequence of relations, denoted by $\boldsymbol{v}_\pi(\hat{p})$. We use a LSTM instead of a simple RNN for its ability to model long-term dependencies, which aids in modeling longer paths.

Entity Type Encoder

The entity type encoder consists of attention modules applied to entity types and a second LSTM. Together, these models are responsible for selecting a type l_t for each entity e_t from its type hierarchy $\mathcal{L}_t = \{l_{t,1}, ..., l_{t,C}\}$, where the lowest level $l_{t,1}$ represents the most specific type and the highest level $l_{t,C}$ represents the most abstract type in a hierarchy of height C.

As shown by the distributional informativeness hypothesis (Santus et al. 2014), selecting l_t from more specific levels of the type hierarchy increases the discriminativeness of the path pattern $\hat{p}$ while selecting from more abstract levels makes the path pattern easier to generalize. Choosing l_t at the appropriate level helps create a path pattern that is both discriminative and generalizable, leading to greater prediction accuracy. However, the substantial number of combinations when considering possible types for entities in all path patterns makes exhaustively searching across all l_t's impossible.

To select l_t, we use the deterministic "soft" attention introduced in (Bahdanau, Cho, and Bengio 2014) to create an approximated vector representation of l_t from the set of type vectors $\{\boldsymbol{v}_\tau(l_{t,1}), ..., \boldsymbol{v}_\tau(l_{t,C})\}$, where $\boldsymbol{v}_\tau(l_{t,i})$ can be obtained by learning vector embeddings of entity types. We name this approximated vector representation of l_t as the type context vector $\hat{\mathbf{a}}_t$. For each type vector $\boldsymbol{v}_\tau(l_{t,i})$ of entity e_t, a weight $\alpha_{t,i}$ is computed by a feed-forward network $f_{att,type}$ conditioned on an evolving **context** $\hat{\mathbf{c}}_t$. This weight can be interpreted as the probability that $l_{t,i}$ is the right level of abstraction or the relative importance for level i to combine $\boldsymbol{v}_\tau(l_{t,i})$'s together. Formally, $\hat{\mathbf{a}}_t$ can be calculated as:

$$e_{t,i} = f_{att,type}(\boldsymbol{v}_\tau(l_{t,i}), \hat{\mathbf{c}}_t) \tag{1}$$

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{k=1}^{C} exp(e_{t,k})} \tag{2}$$

$$\hat{\mathbf{a}}_t = \sum_{i=1}^{C} \alpha_{t,i} \boldsymbol{v}_\tau(l_{t,i}) \tag{3}$$

We model the context of the current step as the previous hidden state of the LSTM, i.e., $\hat{\mathbf{c}}_t = \boldsymbol{h}_{t-1}$. We also use $\boldsymbol{v}_\pi(\hat{p})$, the last state of the relation encoder, to compute the initial memory state and hidden state of the LSTM:

$$\boldsymbol{c}_0 = f_{init,c}(\boldsymbol{v}_\pi(\hat{p})) \tag{4}$$

$$\boldsymbol{h}_0 = f_{init,h}(\boldsymbol{v}_\pi(\hat{p})) \tag{5}$$

where $f_{init,c}$ and $f_{init,h}$ are two separate feed-forward networks. Illustrated in Figure 2, as the LSTM stores information from the relation encoder and previously approximated entity types, both the sequence of relations $\langle in, -in, used\_for \rangle$ and the type context vector of the previous entity $Fork$ can affect the approximated type for $Kitchen$.

With the type context vector $\hat{\mathbf{a}}_t$ computed for each entity e_t, the LSTM sequentially encodes these vectors. The last hidden state of the LSTM is used as a vector representation of all selected entities types, denoted $\boldsymbol{v}_\epsilon(\hat{p})$. We then concatenate $\boldsymbol{v}_\pi(\hat{p})$ and $\boldsymbol{v}_\epsilon(\hat{p})$ together to get the final representation of our proposed path pattern $\boldsymbol{v}_\rho(\hat{p}) = [\boldsymbol{v}_\pi(\hat{p}); \boldsymbol{v}_\epsilon(\hat{p})]$.

Attention Pooling

After representations of all paths in $\mathcal{P}_{ik}$ are obtained using the above models, we then reason over all of the resulting information, jointly, to make the prediction of whether (e_i, r_j, e_k) is true.

Prior neural network models (Neelakantan, Roth, and McCallum 2015) and (Das et al. 2016) use a feed-forward network to condense the vector representation for each path p_i to a single value s_i. One of the pooling methods f_{pool} – Max, Average, Top-K, or LogSumExp – is then applied to all s_i values, combining them and then passing the result through a sigmoid function σ to make the final prediction

$$P(r_j|e_i, e_k) = \sigma(f_{pool}(s_i)) \ , \ \forall s_i \tag{6}$$

Compressing vector representations of paths to single values, as described above, hinders the model's ability to collectively reason about the rich contextual information in the vectors. As a result, in our approach we introduce the use of an attention mechanism for integrating information from all paths, similar to that used to compute the type context vector. We use a trainable relation-dependent vector $\boldsymbol{u}$ to represent the relation r_j we are trying to predict. Following similar steps as when computing $\hat{\mathbf{a}}_t$ from $\{\boldsymbol{v}_\tau(l_{t,1}), ..., \boldsymbol{v}_\tau(l_{t,C})\}$, here we compute a vector representation of all path patterns $\hat{\mathbf{p}}$ from $\{\boldsymbol{v}_\rho(\hat{p}_1), ..., \boldsymbol{v}_\rho(\hat{p}_N)\}$ conditioned on the relation-dependent vector $\boldsymbol{u}$:

$$e_i = f_{att,path}(\boldsymbol{v}_\rho(\hat{p}_i), \boldsymbol{u}) \tag{7}$$

$$\alpha_i = \frac{exp(e_i)}{\sum_{k=1}^{N} exp(e_k)} \tag{8}$$

$$\hat{\mathbf{p}} = \sum_{i=1}^{N} \alpha_i \boldsymbol{v}_\rho(\hat{p}_i) \tag{9}$$

Since $\hat{\mathbf{p}}$ represents all the paths carrying information from both relations and entity types with correct levels of abstraction, the probability that r_j exists between e_i and e_k can be accurately predicted using a feed-forward network f_{pred} along with a sigmoid function σ:

$$P(r_j|e_i, e_k) = \sigma(f_{pred}(\hat{\mathbf{p}})) \tag{10}$$

Training Procedure

The relation encoder, entity type encoder, and attention pooling can be trained end to end as one complete model. For predicting each relation r_j, we train the model using true triples and false triples in the training set as positive and negative examples, denoted $\mathcal{D}_{r_j}^{Train^+}$ and $\mathcal{D}_{r_j}^{Train^-}$ respectively, requiring all triples having relation r_j. Our training objective is to minimize the negative log-likelihood:

$$L = -\sum_{(e_i,r_j,e_k)\in\mathcal{D}_{r_j}^{Train^+}} logP(r_j|e_i,e_k) + \sum_{(e_i,r_j,e_k)\in\mathcal{D}_{r_j}^{Train^-}} logP(r_j|e_i,e_k) \tag{11}$$

We use backpropagation to update the learnable model parameters, which are the relation embedding $\boldsymbol{v}_\delta$ (dim=50), entity type embedding $\boldsymbol{v}_\tau$ (dim=50), trainable relation-dependent vector $\boldsymbol{u}$ (dim=50), two LSTMs in the relation encoder and entity type encoder (dim=150), and feedforward networks $f_{init,c}$, $f_{init,h}$, $f_{att,type}$, $f_{att,path}$, and f_{pred}. We used Adam (Kingma and Ba 2014) for optimization with default parameters (learning rate=$1e^{-3}$, β_1=0.9, β_2=0.999, ϵ=$1e^{-8}$). We trained the models fully to 50 epochs. Then we used early stopping on mean average precision as regularization.

	WN18RR	FB15k-237
# Relations	11	235
# Entities	40,943	13,545
# Relation instances	134,720	254,290
# Relations tested	11	10
Avg. # train inst/relation	38,039	16,696
Avg. # testing inst/relation	11,888	5,219
Avg. path length	5.3	3.5
Max path length	6	4
Avg. # paths per instance	88.6	154.8
# Types	8,092	1,029
Max height of type hierarchy	14	7
Avg. height of type hierarchy	4.6	6.4

Table 1: Statistics of the datasets, the training/testing examples, the extracted paths, and the type hierarchies.

Experimental Setup

This section describes the datasets and baselines used for evaluating our proposed method on the fact prediction task.

Datasets

We evaluated our method and baseline methods on two standard datasets for knowledge base completion: FB15k-237, a subset of the commonsense knowledge graph Freebase (Bollacker et al. 2008), and WN18RR, a subset of the English lexical database WordNet (Miller 1995). The first section of Table 1 shows the statistics of these two datasets.

From all true triples in each KB $\mathcal{X}$, we built up a complete dataset for experiments $\mathcal{D} = \{(e_i, r_j, e_k, y) | e_i, e_k \in \mathcal{E} \wedge r_j \in \mathcal{R} \wedge y \in \{0, 1\}\}$, where y indicates whether a triple is true or false. This dataset contains additional false triples that are sampled from $\mathcal{X}$ using the method based on personalized page rank[3] (released code from (Gardner and Mitchell 2015)). Negative sampling is necessary because both standard datasets only contain true triples. As the number of negative examples has a significant effect on algorithms' performance (Kadlec, Bajgar, and Kleindienst 2017) and evaluation, we consistently sampled 10 negative examples for each positive one. We split the dataset into 80% training and 20% testing. Because path-ranking based methods typically model each relation separately, the data was further divided based on relations.

To extract paths between entities, we first constructed graphs from true triples in the datasets. We augmented the graphs with reverse relations following existing methods (Lao, Mitchell, and Cohen 2011). We extracted paths using bi-directional BFS. We set the maximum length of paths to 6 for WN18RR and 4 for more the densely connected FB15k-237. We randomly sampled 200 paths for each pair if there were more paths. Limiting the length of paths and sub-sampling paths are both due to computational concerns.

To create type hierarchies, we extracted inherited hypernyms available in WordNet (Miller 1995) for WN18RR entities and used type data released in (Xie, Liu, and Sun 2016) for FB15K-237 entities. The number of type levels was automatically determined by the number of inherited hypernyms for each entity in WN18RR and the number of types for each entity in FB15K-237. We further ordered Freebase types based on their frequency of occurrence because types for Freebase entities are not strictly hierarchical. We then followed (Das et al. 2016) to select up to 7 most frequently occurring types for each entity. As more specific types apply to fewer entities and therefore appear in fewer entities' types, ordering types by frequencies allowed us to construct hierarchies of types with specific types generally in the bottom levels and abstract types generally in the top levels. Finally, for WN18RR, we mapped types to their vector representations using a pre-trained Google News word2vec model. Because we did not find a suitable pre-trained embedding for types of FB15K-237 entities, we trained an embedding with the whole model end-to-end.

Baselines

We compared the performance of APR to the following methods[4,5]: ***PRA*** from (Lao, Mitchell, and Cohen 2011), ***SFE*** from (Gardner and Mitchell 2015), ***Path-RNN^A*** from (Neelakantan, Roth, and McCallum 2015), and the two models ***Path-RNN^B*** and ***Path-RNN^C*** from (Das et al. 2016). ***Path-RNN^B*** and ***Path-RNN^C*** are different in the data types they use, as shown in Table 2.

Results

In this section, we report results comparing performance to the above prior methods, as well as independent validation of attention-based pooling. We additionally present insights into the APR model and its modeling of abstraction.

Comparison to Existing Methods

We compared the performance of APR, paired with various pooling methods, against the baselines; Table 2 summarizes the results. All ***APR*** variants outperformed the prior state of the art on both datasets. Directly comparing all methods that utilize LogSumExp pooling (***APR^C*** and all three ***Path-RNN*** models), our results show statistically significant[6] improvement of ***APR^C***. This result indicates that adding types, with the right balance between being generalizable and discriminative, helps create path patterns that allow for more accurate prediction. Our best model ***APR^D***, which further

[3]Gardner and Mitchell show that this negative sampling method has no statistically significant effect on algorithms' performance compared to sampling with PRA, another established but less efficient method used in previous works (Lao, Mitchell, and Cohen 2011; Xiong, Hoang, and Wang 2017).

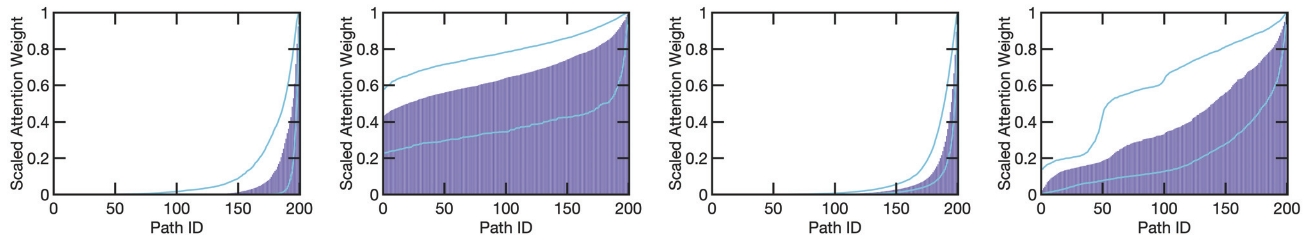
[4]The first two methods are tested with code released by Gardner and Mitchell, and the other three with code by Das et al.. All baselines have been tuned on the validation data and the result from the best hyperparameters is reported for each model.

[5]The first two baselines implement path extraction themselves, other methods and our models use the paths we extracted.

[6]As determined by a paired t-test, with each relation treated as paired data. ($p < 0.05$)

	Data Types	Pooling Method	WN18RR	FB15k-237
PRA	Relation	N/A	38.85	34.33
SFE	Relation	N/A	30.75	36.79
Path-RNN^{A}	Relation	LogSumExp	67.16	45.64
Path-RNN^{B}	Relation, Type	LogSumExp	50.82	51.23
Path-RNN^{C}	Relation, Type, Entity	LogSumExp	51.08	52.17
APR^{A}	Relation, Type	Max	73.23	52.40
APR^{B}	Relation, Type	Average	80.30	54.72
APR^{C}	Relation, Type	LogSumExp	74.51	54.09
APR^{D}	Relation, Type	Attention	**84.91**	**57.35**

Table 2: MAP% of APR (bottom) and baseline methods (top).



(a) *also_see* (b) *verb_group* (c) *music/record_label/artist* (d) *sports/sports_team_roster*

Figure 3: Visualization of path weight for four relations. To generate the visualization, for each relation instance, the path weights are first sorted in ascending order and normalized. Due to the varying numbers of paths between entity pairs, path weights for each relation instance are then interpolated to 200 data points. Median, 25 percentile, and 75 percentile of path weights across relation instances of each relation are plotted.

leverages attention pooling to reason about the rich contextual information in paths, is able to improve state-of-the-art performance by 26% on WN18RR and 10% on FB15k-237 with ($p < 0.005$).

One surprising result is that ***Path-RNN^{B}*** and ***Path-RNN^{C}***, even using entity and type information, still perform worse than ***Path-RNN^{A}*** on WN18RR. We suspect that the extremely large number of entities and types in WN18RR, and the simple feature aggregation method used by these models, cause learning to not generalize even for highly adaptable neural network models. The use of abstraction helps our model generalize information from individual entities and types, and achieve more robust prediction.

Pooling Methods

We compared our proposed attention pooling to three existing pooling methods. As shown in Table 2, ***APR^{D}*** with attention pooling performs the best on both datasets. The superior performance is likely due to the pooling method's ability to collectively reason about the rich contextual information in paths. The other three methods lose information when they compress path representations to single values.

To gain insight about the behavior of attention pooling, we visualized the path weights α_i's computed by the attention module. Figure 3 shows visualizations of four representative relations from the two datasets. Figure 3a and Figure 3c show that for some relations, attention pooling focuses on small numbers of highly predictive paths. In other cases, as in Figures 3b and 3d, attention pooling incorporates data across a broad collection of paths. This ability to dynamically adjust attention based on context highlights the adaptability of attention pooling.

	WN18RR	FB15k-237
Abstract	81.17	49.71
Specific	82.66	54.55
Attention (APR^{D})	**84.91**	**57.35**

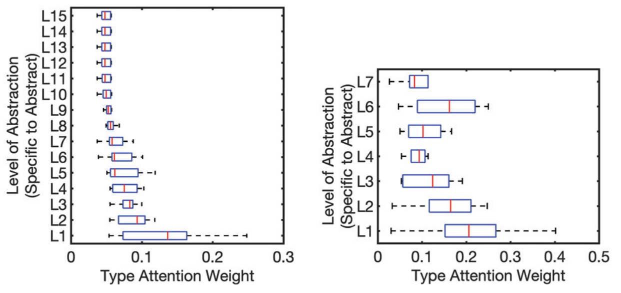
Table 3: MAP% of three models using different levels of abstraction for entities.

Correct Level of Abstraction

We further investigated whether our model learns the proposed path patterns that balance between discrimination and generalization. For comparison, we modified the best model ***APR^{D}*** by fixing the selection of types to either the most specific level or the most abstract level. Table 3 shows the effect of different levels of abstraction on performance. Using a fixed level of abstraction leads to worse performance compared to using attention. This result not only confirms that balancing between generalization and discrimination makes prediction more accurate, but also that our proposed model achieves this balance by learning the correct levels of ab-

Query Relation: has_profession							
Interpretation: Actors graduated from the same school are likely to share the same profession.							
True:	[Actor] John Rhys-Davis	$\xleftarrow{\text{has\_graduates}}$	[Thing] RADA	$\xrightarrow{\text{has\_graduates}}$	**[Actor]** Micheal Gambon	$\xrightarrow{\text{has\_profession}}$	[Thing] Actor
False:	[Actor] Peter Ustinov	$\xleftarrow{\text{has\_graduates}}$	[Thing] Westminster School	$\xrightarrow{\text{has\_graduates}}$	**[Author]** Christopher Wren	$\xrightarrow{\text{has\_profession}}$	[Thing] Scientist
Query Relation: has_genre							
Interpretation: Films focusing on the same subject are likely to share the same genre. However, films depicting the same profession can have different genres.							
True:	[Topic] Life Is Beautiful	$\xleftarrow{\text{has\_film}}$	**[Subject]** World War 2	$\xrightarrow{\text{has\_film}}$	[Topic] Valkyrie	$\xrightarrow{\text{has\_genre}}$	[Genre] War Film
False:	[Topic] Air Force One	$\xleftarrow{\text{has\_film}}$	**[Profession]** Aviation	$\xrightarrow{\text{has\_film}}$	[Topic] Team America	$\xrightarrow{\text{has\_genre}}$	[Genre] Political Satire
Query Relation: nominee_work							
Interpretation: The best actors are likely to perform great in all movies. The same company may not always produce award winning works.							
True:	**[Thing]** Academy Actor	$\xrightarrow{\text{nominee\_work}}$	**[Award Topic]** Network	$\xleftarrow{\text{nominated\_for}}$	**[Actor]** Sidney Lumet	$\xrightarrow{\text{nominated\_for}}$	**[Award Item]** Orient Exp
False:	**[Award Category]** Emmy Comedy	$\xrightarrow{\text{nominee\_work}}$	**[Tv Program]** Entourage	$\xleftarrow{\text{nominated\_for}}$	**[Employer]** HBO	$\xrightarrow{\text{nominated\_for}}$	**[Tv Program]** Gia

Table 4: Correctly predicted examples by the proposed model on test data of FB15k-237. Despite the fact that each pair of true and false examples share the same sequence of relations, the uses of entity types at the right level of abstraction (shown in bold) help distinguish between them. Abbreviated entity and relation names are shown for clarity.



(a) Average type weights of WN18RR (b) Average type weights of FB15k-237

Figure 4: Box plot of average attention weights at each level. For each relation, the average attention weight of each level is computed by averaging over the weights predicted by the model at this level for all entities in all paths.

straction for entities.

We also examined the distribution of attention weight at different levels of type hierarchies. Figure 4 shows that the model leveraged all levels for representing various entities. More specifically, level 1, 2, and 3 are most commonly emphasized for entities in WN18RR; level 1,2, and 6 are most often used for entities in FB15k-237. The emphasis on lower levels is expected since using the most specific level is more beneficial than using the most abstract level with evidence in Table 3. However, the diversity of levels used by the model proves that not all entities should use the most specific level: different entities require different levels of abstraction.

Finally, we visualized examples of paths from prediction along with the top weighted types the model selected for entities. In Table 4, we are able to see examples that correspond strongly with human intuition. This qualitative result again verifies that the model learns a new class of rules (path patterns) that is more specific yet still generalizable.

	WN18RR	FB15k-237
TuckER	38.16	40.12
ComplEx-N3	40.89	40.35
APR^{D}	**84.91**	**57.35**

Table 5: MAP% comparison to two SOTA embedding methods.

Comparison with Knowledge Graph Embedding Methods

As a final point of comparison, we validated our best model ***APR^{D}*** against two state-of-the-art KGE methods[7]: ***TuckER*** (Balazevic, Allen, and Hospedales 2019) and ***ComplEx-N3*** (Lacroix, Usunier, and Obozinski 2018). As shown in Table 5, our model performs significantly better on both datasets. In fact, all neural-based path ranking methods (***APR*** and ***Path-RNN***) outperform the KGE methods. One possible reason that path ranking methods outperform KGE is that KGE methods require both the source e_i and target entities e_k in the missing triple (e_i, r_j, e_k) to be known, thus failing to perform well when test entities are not present in the training set. However, only 17.6039% of FB15k-237's test data consists of new entities, and only 0.9412% in WN18RR. As a result, new entities do not alone account for the difference in performance. This comparison additionally affirms the improved state-of-the-art performance in fact prediction achieved by our method.

[7]We followed (Xiong, Hoang, and Wang 2017) to evaluate KGE methods on the fact prediction task. Test triples with the same relation are ranked.

Conclusion

This work addresses the problem of knowledge base completion. We introduced Attentive Path Ranking, a novel class of generalizable path patterns leveraging type hierarchies of entities, and developed an attention-based RNN model to discover the new path patterns from data. Our approach results in statistically significant improvement over state-of-the-art path-ranking based methods and knowledge graph embedding methods on two benchmark datasets WN18RR and FB15k-237. Quantitative and qualitative analyses of the discovered path patterns provided insights into how APR achieves a balance between generalization and discrimination.

Acknowledgments

This work is supported in part by NSF IIS 1564080, NSF GRFP DGE-1650044, and ONR N000141612835.

References

Aditya, S.; Yang, Y.; and Baral, C. 2018. Explicit reasoning over end-to-end neural architectures for visual question answering. In *AAAI*.

Athiwaratkun, B., and Wilson, A. G. 2018. Hierarchical density order embeddings. In *ICLR*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Balazevic, I.; Allen, C.; and Hospedales, T. M. 2019. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP*.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Chan, W.; Jaitly, N.; Le, Q.; and Vinyals, O. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*.

Das, R.; Neelakantan, A.; Belanger, D.; and McCallum, A. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*.

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

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Gardner, M., and Mitchell, T. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*.

Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Kadlec, R.; Bajgar, O.; and Kleindienst, J. 2017. Knowledge base completion: Baselines strike back. In *Rep4NLP@ACL*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Lacroix, T.; Usunier, N.; and Obozinski, G. 2018. Canonical tensor decomposition for knowledge base completion. In *ICML*.

Lao, N.; Mitchell, T.; and Cohen, W. W. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*. ACL.

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

Min, B.; Grishman, R.; Wan, L.; Wang, C.; and Gondek, D. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL*, 777–782.

Neelakantan, A.; Roth, B.; and McCallum, A. 2015. Compositional vector space models for knowledge base inference. In *2015 AAAI Spring Symposium Series*.

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

Quinlan, J. R., and Cameron-Jones, R. M. 1993. Foil: A midterm report. In *European conference on machine learning*. Springer.

Santus, E.; Lenci, A.; Lu, Q.; and Schulte im Walde, S. 2014. Chasing hypernyms in vector spaces with entropy. In *EACL*. ACL.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Vendrov, I.; Kiros, R.; Fidler, S.; and Urtasun, R. 2016. Order-embeddings of images and language. In *ICLR*.

Wang, H.; Zhang, F.; Xie, X.; and Guo, M. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *World Wide Web Conference*, 1835–1844. International World Wide Web Conferences Steering Committee.

Xie, R.; Liu, Z.; and Sun, M. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, 564–573. Association for Computational Linguistics.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.