Attentive Path Combination for Knowledge Graph Completion

Xiaotian Jiang [1,2] JIANGXIAOTIAN@IIE.AC.CN
Quan Wang [1,2,3] * WANGQUAN@IIE.AC.CN
Baoyuan Qi [1,2] QIBAOYUAN@IIE.AC.CN
Yongqin Qiu [1,2] QIUYONGQIN@IIE.AC.CN
Peng Li [1,2] LIPENG@IIE.AC.CN
Bin Wang [1,2] WANGBIN@IIE.AC.CN

[1] *Institute of Information Engineering, Chinese Academy of Sciences*
[2] *School of Cyber Security, Chinese Academy of Sciences*
[3] *State Key Laboratory of Information Security, Chinese Academy of Sciences*

Editors: Yung-Kyun Noh and Min-Ling Zhang

Abstract

Knowledge graphs (KGs) are often significantly incomplete, necessitating a demand for KG completion. Path-based relation inference is one of the most important approaches to this task. Traditional methods treat each path between entity pairs as an atomic feature, thus inducing sparsity. Recently, neural network models solve this problem by decomposing a path as the sequence of relations in the path, before modelling path representations with Recurrent Neural Network (RNN) architectures. In cases there are multiple paths between an entity pair, state-of-the-art neural models either select only one path, or make usage of simple score pooling methods like Top-K, Average, LogSumExp. Unfortunately, none of these methods can model the scenario where relations can only be inferred by considering multiple informative paths collectively. In this paper, we propose a novel path-based relation inference model that learns entity pair representations with attentive path combination. Given an entity pair and a set of paths connecting the pair, our model allows for integrating information from each informative path, and form a dynamic entity pair representation for each query relation. We empirically evaluate the proposed method on a real-world dataset. Experimental results show that the proposed model achieves better performance than state-of-the-art path-based relation inference methods.

Keywords: Knowledge graph completion, recurrent neural network, attention model

1. Introduction

Knowledge graphs (KGs), such as WordNet(Miller (1995)), YAGO(Suchanek et al. (2007)) and Freebase(Bollacker et al. (2008)), are graphs that take entities as nodes, and relations between entities as labelled-edges. KGs have played an very important role in many AI-related applications, like question answering(Hixon et al. (2015)), information retrieval(Richardson and Smeaton (1995)) and recommender systems(Catherine and Cohen (2016)), etc. KGs are usually stored in triples of the form (head entity, relation, tail en-

* Corresponding author.

tity), called facts. Typical KGs may contain millions of facts, but they are still far from complete. Knowledge graph completion tries to solve this incompleteness by inferring new triples based on the existing ones. There are three lines of work in this area: (1) models that use paths to infer relations between an entity pair. Related methods include Path Ranking Algorithm(Lao and Cohen (2010)), Path-RNN(Neelakantan et al. (2015)), Single-Model(Das et al. (2017)), etc. (2) models that directly use embeddings of entities and relation for inference (e.g., TransE(Bordes et al. (2013)) and TransH(Wang et al. (2014))) (3) graph models like Markov logic networks(Chen and Wang (2013)). In this paper, we focus on the first line of work for its interpretability and no need of logic rules, and call it path-based relation inference.

Path Ranking Algorithm (PRA) is the first work of this line. The key idea of PRA is to explicitly use paths connecting two entities to predict if potential relations exist between them. Given a specific query relation, random walks are first employed to find the set of paths between two entities. Here a path is a sequence of relations linking the two entities. These paths are then used as features in a binary classifier to predict if the entity pair have the given query relation.

In PRA method, each path is treated as an atomic feature. As a result, there used to be millions of distinct paths in a single classifier, not to mention that the size increases dramatically with the number of relations in a KG. To solve this problem, Neelakantan et al. (2015) proposed a neural method call Path-RNN. This model decomposes each path as a sequence of relations, and feed it into an RNN architecture to construct a vector representation for the path. After that, the predictability of the path to a query relation is calculated by dot-product on their representations. As is often the case that there are multiple paths connecting an entity pair, Path-RNN uses Max operator to select the path with the largest predictability at each training/testing iteration. To improve the performance of Path-RNN, Das et al. (2017) proposes several path combination operators, including Mean, Top-K, LogSumExp. In addition, they also use a single RNN model to deal with all query relations, as well as model the types of entities in the path.

However, each of these path combination operators works at score-level, and has its own deficiency:

- Max: Only one single path is used for inference, while all other informative paths are neglected.
- Average: The set of paths connecting an entity pair is usually very large, and only a few paths may contribute to the inference. Thus, this operator often makes model suffering from noise.
- Top-K: Different entity pairs may have different optimal K values. Besides, not all top K paths contribute equally to the inference.
- LogSumExp: This is a smooth approximation to the "Max" operator, which fails to effectively integrate evidences from multiple paths.

Actually, in path-based relation inference, it is often the case that accurate inference comes from integrating evidences in multiple informative paths and making inference collectively. See Figure 1 for an example. None of the four paths directly contains information

that the nationality of Steve Jobs is U.S., but if we consider these paths jointly, we will get much more evidences supporting the fact, profiting from the relevant information available in different paths. Unfortunately, the above-mentioned path combination operators fail to deal with this scenario. For instance, when using LogSumExp()[1] to combine scores of these paths, the result is only marginally larger than the max score of them. Thus, if a relation can not be inferred by any single path, meaning that even the max score stays low, the resulting score can still not be predictive to the relation.

.

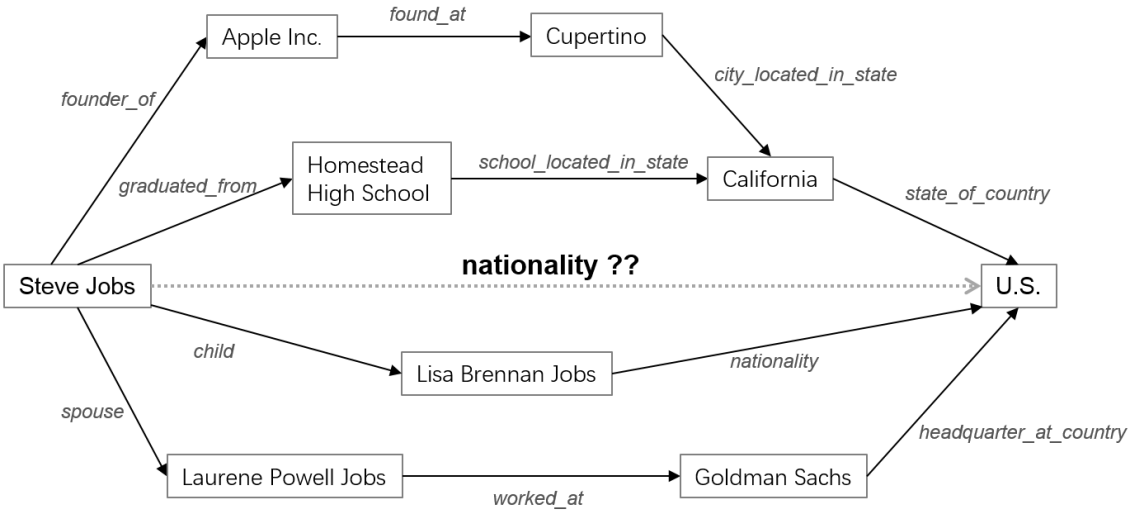

Figure 1: An illustration that relations between an entity pair can be inferred by considering information available in multiple paths collectively.

Based on this observation, in this paper we propose a novel attentive path-based model for knowledge graph completion. Given an entity pair, we first employ RNN to embed the semantic of each path connecting the pair. Afterwards, we represent each entity pair as a weighted combination of paths, enabling information aggregation from multiple path representations. With the usage of attention mechanism, the weights of combination are determined automatically and dynamically for each query relation. Finally, potential relations are extracted by calculating scores on the representation of entity-pair and query relation. In experiment section, we evaluate the proposed model on a real-world dataset. Experimental results show that the proposed model achieves better performance than state-of-the-art methods.

The main contributions of this paper can be summarized as follows:

- We point out that in path-based relation inference, it is often the case that only the aggregation of informative paths can be predictive. Existing path combination methods fail to deal with this situation.

1. This operator achieves the best performance in Das et al. (2017)

- We propose an attentive path-based relation inference model that aggregates evidences from multiple paths and constructs dynamic entity pair representation for each query relation automatically.
- In experiments, we show the effectiveness of the proposed model against state-of-the-art methods on a real-world dataset.

In the remainder of this paper, we first review related work in Section 2, and briefly introduce neural path-based relation inference models in Section 3. We then detail the proposed attentive model in Section 4. Experiments and results are reported in Section 5, followed by the conclusion and future work in Section 6.

2. Related Work

We will review three lines of work in this section: (i) Knowledge graph completion; (2)Path-based relation inference; (3) Attention-based models.

Knowledge graph completion. This task is to automatically infer missing facts from existing ones. Prior work can be roughly categorized into: (i) path-based relation inference models that use paths connecting two entities to predict potential relations between them (Lao and Cohen (2010); Lao et al. (2011)); (ii) models that use embedding of entities and relations for inference (Nickel et al. (2011); Bordes et al. (2013); Nickel et al. (2016);Wang et al. (2017)); (iii) probabilistic graphical models such as the Markov logic network (MLN) and its variants (Pujara et al. (2013); Jiang et al. (2012)). This paper focuses on path-based relation inference, since it is easily interpretable (as opposed to triple-based models) and requires no external logic rules (as opposed to MLN and its variants). Note that in triple-based methods, there are methods that use paths between entity pairs as well, like Lin et al. (2015) and Luo et al. (2015). These methods focus on learning better representation of entities and relations with the help of paths, and can be treated as path-enhanced versions of TransE(Bordes et al. (2013)). In contrast, path-based relation inference methods make inference mainly based on the paths themselves.

Path-based relation inference. Traditional models for path-based relation inference mainly include PRA and its variants. PRA, first proposed by Lao and Cohen (2010), is a random walk inference technique designed for predicting new relation instances in KGs. Recently, various extensions have been explored, ranging from incorporating a text corpus as additional evidence during inference (Gardner et al.; Gardner et al. (2014)), to introducing better schemes to generate more predictive paths (Gardner and Mitchell (2015)), or considering associations among certain relations (Wang et al. (2016)). These variants, with the attempts to reduce sparsity, still essentially treat each path as an atomic feature. As opposed to these methods, Neelakantan et al. (2015) adopts a different approaches by treating each path as a sequence of relations, and use an RNN architecture to generate its vector representation. The resulting path representations are then used for inference. Das et al. (2017) improves the performance of Path-RNN by proposing several path combination operators. Besides, they also use a single RNN model to deal with all query relations, as well as model the types of entities in the path.

Attention-based models. Attention mechanism enables models to focus on different context under different status. Recently, the attention-based model has attracted a lot

of interests in a wide range of research fields, including image classification (Mnih et al. (2014)), speech recognition (Chorowski et al. (2014)), image caption generation (Xu et al. (2015)) , machine translation (Bahdanau et al. (2014),Luong et al. (2015)), and sentence pair modelling (Yin et al. (2016)). To the best of our knowledge, this is the first attentive model used in path-based relation inference task.

3. Backgrounds

In this section, we will briefly introduce two neural models for path-based relation inference: Path-RNN, and its improved version, Single-Model.

3.1. Path-RNN

Given a pair of entities, Path-RNN(Neelakantan et al. (2015)) takes as input the set of paths connecting the pair and outputs new relations that may hold between the entities. For each of the paths, an RNN module is applied to get its vector representation, which is then used for inference. The set of candidate relations is called query relations. For each query relation, a binary classifier is built in the following process.

Let (e_s, e_t) denotes the entity pair, Π denotes the set of paths between the pair, $\pi = \{e_s, r_1, e_1, ..., r_k, e_t\} \in \Pi$ denotes a path in the set, δ denotes a query relation. For brevity, we use bold font to denote vector representation[2]. The length of a path is defined as the number of relations in the path. For each path, the RNN model consumes one relation at each step, and outputs a hidden state vector $\mathbf{h_t} \in \mathbb{R}^h$. At step t, the hidden state is defined as:

$$\mathbf{h_t} = f(\mathbf{W_1^\delta h_{t-1}} + \mathbf{W_2^\delta r^\delta}) \tag{1}$$

where $\mathbf{W_1^\delta} \in \mathbb{R}^{d\times d}$, $\mathbf{W_2^\delta} \in \mathbb{R}^{d\times d}$ are model parameters, f is Sigmoid() function.

When RNN has been applied on the entire path, the latest output is the hidden vector at the last step k, and is treated as the vector representation of the path.

Afterwards, the score of a path to infer query relation δ is calculated as:

$$score(\pi, \delta) = \boldsymbol{\pi} \cdot \boldsymbol{\delta} \tag{2}$$

Since there usually exists more than one path connecting an entity pair, Path-RNN applies a path selection process: the path with the largest score is selected to represent the entity pair. That is,

$$ep_{s,t} = \arg\max_{\pi\in\Pi} score(\pi, \delta) \tag{3}$$

Thus, the probability of a query relation δ holding between entity pair (e_s, e_t) is calculated as:

$$\mathbf{P}(\delta|e_s, e_t) = \sigma(\mathbf{ep_{s,t}} \cdot \boldsymbol{\delta}) \tag{4}$$

where σ is Sigmoid() function.

2. For example , $\mathbf{r_1^\delta} \in \mathbb{R}^d$ denotes the vector representation of relation r_1 when building the classifier for relation δ.

For each query relation, parameters of its classifier include relation embeddings, $\boldsymbol{W_1^\delta}$ and $\boldsymbol{W_2^\delta}$, which are trained to maximize the log likelihood on the training dataset. The loss function is optimized with AdaGrad (Duchi et al. (2011)), and L-2 regularization is employed to prevent over-fitting.

3.2. Single-Model

Compared with Path-RNN model, Single-Model(Das et al. (2017)) makes three improvements: parameter sharing, score pooling and entity type integration. In this section, we will detail these improvements.

Parameter sharing. Like PRA, Path-RNN builds a separate classifier for each query relation, thus keeping a separate set of model parameters for each query relation. For query relations with insufficient training instances, the model is hard to get well trained. In comparison, Single-Model allows all query relations to share the same model. Specifically, all query relations share the same model parameters: relation embeddings, $\mathbf{W_1}$, and $\mathbf{W_2}$. By doing this, Single-Model dramatically reduces the number of model parameters, and makes all query relations share the same underlying path representations. As a result of parameter sharing, the multiple binary classifiers are reduced to one single multi-instance multi-label classifier.

Score pooling. Path-RNN only uses one path for relation inference, neglecting the information carried by other paths. In contrast, Single-Model attempts to use several other path pooling method, including:

- Average: is the average score of all paths.
- Top-K: is the average score of the Top-K scored paths.
- LogSumExp: is a smooth approximation to the "Max" operator, and is calculated as

$$LogSumExp(s_1, s_2, ...s_n) = log(\sum_{i=1}^{n} exp(s_i)) \tag{5}$$

  where s_i is the score of the i-th path to the query relation.

Among these score pooling methods, LogSumExp is reported to have the best performance. This is because this operation shows emphasis on the highest scored path, as well as facilitating gradient propagation on all paths.

Incorporating entity types. Not only relations on each path can help inferring query relations, entities may also help building discriminative path representation. In consideration of the sparsity of entities, Single-Model uses entity types instead of entity itself. Many entities have multiple types. For example, Donald Trump has types like President, Businessman, television personality, American, etc. Single-Model uses 7 frequently occurring types (at most) for each entity, and averages the vector representation of the types. In our model, we follow this setting.

4. Attentive Path Combination Model

Given an entity pair (e_s, e_t) and the set of paths $\Pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ connecting the pair, we propose attentive path combination model that successively applies the following modules to infer the plausibility of each query relation δ holding between the pair:

1. Path representation generation: we employ an RNN architecture (for all query relations) to generate a vector representation for each path π_i.
2. Attentive path combination: we use attention mechanism to dynamically construct vector representation for the entity pair with respect to given query relation δ, which is then used for relation inference.

Details of the two components are demonstrated as follows.

4.1. Path Representation Generation

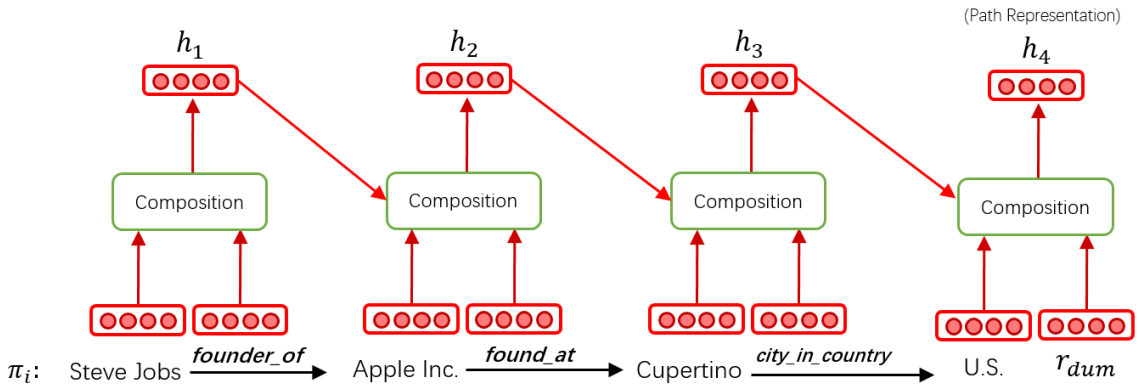

Figure 2: The RNN architecture used to extract path vector representation. r_{dum} is a dummy relation.

As shown in Figure 2, in this module, each relation and entity in path $\pi_i = \{e_s, r_1, e_1, r_2, ..., e_t\}$ is first mapped to a vector representation, then composed sequentially in an RNN fashion. At each RNN step t, the model consumes the representation of entity e_{t-1} (Let $e_0 = e_s$) and a relation r_t, and outputs a hidden state $\mathbf{h_t}$. To reduce model parameters, we map each entity to the averaged representation of its types. For simplicity, we still use $\mathbf{e_{t-1}} \in \mathbb{R}^k$ to denote the averaged type representation of entity e_{t-1}[4]. Let $\mathbf{r_t} \in \mathbb{R}^d$, $\mathbf{h_{t-1}} \in \mathbb{R}^d$, the RNN composition function is:

$$\mathbf{h_t} = f(\mathbf{W_1 h_{t-1}} + \mathbf{W_2 r} + \mathbf{W_3 e_{t-1}}) \tag{6}$$

where $\mathbf{W_1} \in \mathbb{R}^{d \times d}$, $\mathbf{W_2} \in \mathbb{R}^{d \times d}$, $\mathbf{W_3} \in \mathbb{R}^{d \times k}$ are RNN parameter matrices. f is a non-linear function. In the proposed method, we choose $f = ReLU()$.

4. If such representations are not supplied (when only relations in the path are used), the model just ignores term $\mathbf{W}_3\mathbf{e}_{t-1}$.

4.2. Attentive Path Combination

When all path representations are generated, we then construct an entity-pair level representation for each query relation. With the help of attention mechanism, the construction is implemented by automatically building discriminative weighted combinations of path representations for different query relations, as shown in Figure 3.

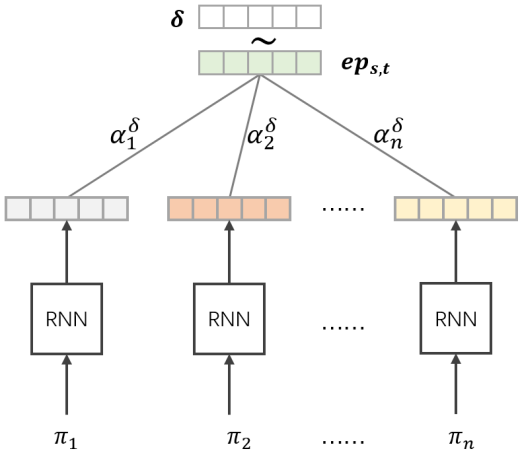

Figure 3: Architecture of the proposed method. $\pi_1, \pi_2, \cdots, \pi_n$ denote paths connecting an entity pair; $\boldsymbol{\delta}$ denotes the vector representation of query relation δ; $\sim$ represents dot-product operation. Each path is first converted to a vector representation by RNN, before integrated into entity pair representation $\mathbf{ep_{s,t}}$ with the proposed attentive path combination method.

The input of this module is a set of vector representations of the paths in Π: $\{\boldsymbol{\pi_1}, \boldsymbol{\pi_2}, \cdots, \boldsymbol{\pi_n}\}$.

We model the vector representation of an entity pair as a non-linear weighted representation of its paths using attention mechanism, which is

$$\mathbf{ep}^{\delta}_{\mathbf{s,t}} = f(\sum_{i=1}^{n} \alpha^{\delta}_{i} \boldsymbol{\pi_i}) \tag{7}$$

where α^{δ}_{i} is the weight of path i when modelling the entity pair representation for query relation δ. f is a non-linear function. In this paper, we use $f = Tanh()$.

The weight for each path α^{δ}_{i} is further defined as:

$$\alpha^{\delta}_{\mathbf{i}} = \frac{exp(z^{\delta}_{i})}{\sum_{j} exp(z^{\delta}_{j})} \tag{8}$$

where e^{δ}_{i} measures how well input path π_i and query relation δ matches, and is modeled as:

$$\mathbf{z}^{\delta}_{\mathbf{i}} = tanh(\boldsymbol{\pi_i}\mathbf{T})\boldsymbol{\delta} \tag{9}$$

where $\mathbf{T} \in \mathbb{R}^{d \times d}$ is the parameter matrix of attention module; $\boldsymbol{\delta}$ denotes the vector representation of query relation δ.

We have also tried several other attention forms , like $\boldsymbol{\pi_i T \delta}$, and find the form in (9) has better performance. This superiority can be attributed to its better modelling of non-linearity between the embeddings of path and query relation.

After getting the representation of the entity pair, its probability in participating relation δ is given by

$$\mathbf{P}(\delta|e_s, e_t) = \sigma(score(ep_{s,t}, \delta))$$

where σ is sigmoid function, and $score(ep_{s,t}, \delta) = \mathbf{ep_{s,t}} \cdot \boldsymbol{\delta}$.

Following Das et al. (2017), we train a single model for all query relations. Let Δ_R^+, Δ_R^- denote the set of positive and negative triples for all query relations, Θ denotes parameters of the model: $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{T}$, and the embeddings of relations and entity types. The model is trained to minimize the negative log-likelihood with L2 regularization, and its loss function is:

$$L(\Theta, \Delta_R^+, \Delta_R^-) = -\sum_{e_s, e_t, \delta \in \Delta_R^+} log\mathbf{P}(\delta|e_s, e_t) - \sum_{\hat{e_s}, \hat{e_t}, \hat{\delta} \in \Delta_R^-} log(1 - \mathbf{P}(\hat{\delta}|\hat{e_s}, \hat{e_t})) + \lambda||\Theta||^2 \quad (10)$$

All model parameters to be learned are initialized randomly, and the optimization method we use is Adam (Kingma and Ba (2014)).

5. Experiments

5.1. Dataset and Evaluation Metrics

Dataset. We evaluate the proposed method on a real-world dataset, whose previous versions have been used in earlier works of this line (Neelakantan et al. (2015), Das et al. (2017)). The dataset is continuously augmented, so we employ its latest released version in this paper. In our experiment, we call this version FC17 dataset. FC17 contains information from both Freebase and Clueweb, with the links of Freebase entity in ClueWeb dataset (Gabrilovich et al. (2013)) utilized to generated an enriched larger knowledge graph. Specifically, information from ClueWeb are merged by considering sentences that contain two entities linked to Freebase, and the phrase between the two entities are extracted and treated as relations. Note that for phrases that are of length greater than 4, only the first and last two words are kept. To reduce noise, only relations that occur at least 50 times are selected. 46 query relation in Freebase that have the most number of instances are selected as query relations. Models in comparison are all evaluated on a subset of facts hidden during training. In this dataset, the number of paths between an entity pair ranges drastically from 1 to 900 or more, so the robust of methods in comparison can be better evaluated with this dataset. Statistics of FC17 dataset is listed in Table 1. Compared with an older version used in Das et al. (2017), this dataset has far more ClueWeb triples.

Evaluation metrics. We use mean average precision (MAP) and mean reciprocal rank (MRR) as evaluation metrics, following recent works evaluating knowledge graph completion performance (Neelakantan et al. (2015), Das et al. (2017), West et al. (2014); Gardner and Mitchell (2015)). Both metrics evaluate some ranking process: if a method ranks the positive instances before the negative ones for each relation, it will get a high MAP or MRR.

Stats.	FC17 dataset
# Freebase triples	35M
# ClueWeb triples	104M
# Freebase relation types	2213
# textual relation types	23612
# query relation types	46
# entities	3.31M
# paths	146M
# path occurance	183M
Avg. unique paths/query relation	3.19M
Avg. path occurance/query relation	3.99M
Avg. path length	4.48
Max path length	7
Avg. training positive/query relation	6621
Avg. training negative/query relation	6622
Avg. positive test instances/query relation	3516
Avg. negative test instances/query relation	43777

Table 1: Statistics of FC17 dataset

5.2. Models in Comparison and Settings

On the above-mentioned real-world dataset, we compare the performance of the proposed method against the following baseline methods:

- **Path Ranking Algorithm (PRA)**: is a method that uses paths connecting an entity pair as atomic features for relation inference. An binary classifier is built for each query relation.
- **Path-RNN**: is a model that decomposes each path into a sequence of relations, and uses recurrent neural network to generate the representation of the path. For each query relation, an binary classifier is built.
- **Single-Model**: is a neural model that trains one model for all query relations. In cases there are multiple paths connecting an entity pair, the method uses several score pooling methods, among which LogSumExp[3] achieves the best performance. In this section, for simplicity, we use Single-Model to refer to the model with LogSumExp score pooling method.
- **Single-Model + Type**: is Single-Model with the types of each entity included at each step of RNN path modelling.
- **Att-Model**: refers to the proposed model that uses attention mechanism to combine paths and construct entity pair representations.
- **Att-Model + Type**: refers to the proposed attentive model with relation and types of entities modeled at each step of RNN path modelling.

Hyper-parameters of each model are tuned on development set. For PRA method, we select c in the range of $\{2^{-5}, 2^{-4}, 2^{-3}, ..., 2^3, 2^4, 2^5\}$ for each individual classifier. For neural models, we also select each hyper-parameter from a range and choose the one with the best score on dev set. The model setting used in our method is as follows: we set the dimension of relation representation and the hidden states $d, h = 250$, and the dimension of entity type $m = 100$. Model are trained for 30 epochs, with batch size = 32, learning rate = $1e^{-3}$ and l2-regularizer $\lambda = 1e^{-5}$. Adam settings are as default: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e^{-8}$.

5.3. Experimental Results

We test the effectiveness of our attentive model on 46 query relations, and report the results in Table 2.

Model	%MAP	%MRR
PRA	55.48	94.02
Path-RNN	52.37	92.46
Single-Model	58.28	93.84
Att-Model	**59.89**	**94.20**
Single-Model+Types	63.88	90.22
Att-Model+Types	**65.03**	**92.75**

Table 2: Experiments results on FC17 dataset

From the results, we can observe that our algorithm achieves the best performance in both evaluation metrics. Specifically, (1) when we only use relations in the path (Att-Model), the proposed model outperforms other methods that also use relation only on MAP score, and achieves the best MRR score against all models in comparison; (2) when we further consider the entities in the path by adding their types into RNN modelling, we find that the proposed method (Att-Model+Types) still achieves considerable improvements than its main opponent (Single-Model+Types) on MRR score, and achieves the best performance among all methods in MAP score. We note that the baseline neural models do not show significantly better performance than PRA, and attribute this to the enormous quantity of ClueWeb triples in the dataset. The semantic ambiguity in the generated Clueweb relations both give rise to sparsity and do harm to RNN modelling process. PathRNN, the model that treats each query relation separately, suffers significantly.

To better show the strength and weakness of the proposed method against Single-Model, we further dive into the MAP score. First, based on relation hierarchy, we group the 46 query relations into 16 categories, as shown in Table 3. Then, we train a single model on the entire FC17 training dataset, and calculate MAP score for each category. The results are listed in Table 4. It can be observed that with the attentive method, 14 out of 16 categories get their performances improved. Among them, the "TV" category has the largest improvement — 4.7% (from 49.4% to 54.1%), and the averaged MAP improvement over these 16 relations reaches 2.1%.

3. We have experimented with all pooling methods mentioned in Neelakantan et al. (2015) and Das et al. (2017) on FC17 dataset, and find LogSumExp still achieves the best performance among them. So we only show scores with LogSumExp operation for brevity.

Category	Relations
Soccer	/soccer/football_player/position_s
Organization	/organization/organization/founders, /organization/organization/locations, /organization/organization/sectors
Business	/business/industry/companies
People	/people/deceased_person/cause_of_death, /people/deceased_person/place_of_death, /people/ethnicity_people/people_family_members, /people/person/nationality, people/person/place_of_birth, /people/person/profession, /people/person/religion
TV	/tv/tv_program/country_of_origin, /tv/tv_program/genre
Time	/time/event/locations
Broadcast	/broadcast/content/artist, /broadcast/content/genre
Book	/book/book/characters, /book/literary_series_works/in_this_series, /book/written_work/original_language
Music	/music/album/genre, /music/artist/genre, /music/artist/label, /music/artist/origin, /music/composition/composer, /music/composition/lyricist, /music/genre/albums
Architecture	/architecture/structure/address
Aviation	/aviation/airport/serves
CVG	/cvg/game_version/game, /cvg/game_version/platform, /cvg/computer_videogame/cvg_genre
Geography	/geography/river/cities, /geography/river/mouth
Education	/education/educational_institution/campuses, /education/educational_institution/school_type
Film	/film/film/cinematography, /film/film/country, /film/film/directed_by, /film/film/film_festivals, /film/film/language, /film/film/music, /film/film/rating, /film/film/sequel
Location	/location/location/contains

Table 3: 46 query relations and their categories.

Relation Category	Single-Model	Att-Model
Soccer	22.5736	**24.9923**
Organization	48.2477	**49.3407**
Business	38.7261	**40.8727**
People	56.9425	**58.3689**
TV	49.4428	**54.1473**
Time	42.4003	**44.4707**
Broadcast	40.5421	38.3817
Book	85.9708	84.7696
Music	56.3312	**58.2192**
Architecture	54.4483	**66.3012**
Aviation	66.1677	**66.7734**
CVG	49.5236	**51.2568**
Geography	51.4230	**53.0589**
Education	65.9925	**67.0354**
Film	69.6533	**71.2469**
Location	83.3504	**85.5956**
Avg.	55.1085	**57.1770**

Table 4: %MAP performance on each relation category. Avg. denotes average MAP scores over the 16 categories. Bold font shows better performance with the proposed attentive model.

5.4. Case Study

In this section, we show the effectiveness of the proposed attention mechanism by two cases. We select two query relations, choose two positive examples and observe paths with high and low attention values, respectively.

Relation	/people/person/place_of_birth
High	was,born,in
Low	/people/deceased_person/place_of_death-/m/05ywg-_about,km,from-/m/0h7x-/location/location/contains
Relation	/film/film/directed_by
High	/film/film/produced_by-/m/0jzr__-/film/producer/film-/m/027j9wd-/film/film/directed_by
Low	_s-/m/016tt2-_at-/m/06w839_-_/film/film_story_contributor/film_story_credits

Table 5: Examples of attention mechanism in FC17 dataset, where _ denote reverse relation.

From Table 5, we can see that paths with high attention to query relation are more predictive, while paths with low attention tend to be randomly connected and lack the ability of prediction.

6. Conclusion and Future Work

In this paper, we point out the weakness of existing path combination methods and propose a novel path-based relation inference model. This model works by learning effective entity pair representation with attentive path combination. Given an entity pair and the set of paths connecting the pair, our model allows for integrating information from each path, and forms a dynamic representation of the pair with respect to each candidate query relation. We empirically evaluate the proposed method on a real-world dataset. Experimental results show that it achieves better performance than state-of-the-art path-based relation inference methods.

Our future research will include the following topics:

Subgraph integration. In this paper, we only consider the collection of paths connecting an entity pair as its representation. More information can be augmented to the representation by considering the context of each entity node.

Attention on types. State-of-the-art methods only average the representations of entity types for each entity on path, neglecting that different types may play different roles in relation inference. This observation can be modelled by attention mechanism.

Acknowledgments

We would like to thank all the reviewers for their insightful and valuable suggestions, which significantly improve the quality of this paper. This work is supported by the National Natural Science Foundation of China (grants No. 61402465), the National Key Research and Development Program of China (grants No. 2016QY03D0503), and the Fundamental Theory and Cutting Edge Technology Research Program of the Institute of Information Engineering, Chinese Academy of Sciences (grant No. Y7Z0261101).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 325–332. ACM, 2016.

DZW Yang Chen and Daisy Zhe Wang. Web-scale knowledge inference using markov logic networks. In *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*, pages 106–110. Association for Computational Linguistics, 2013.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: first results. *arXiv preprint arXiv:1412.1602*, 2014.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141, Valencia, Spain, April 2017. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

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). *Note: http://lemurproject. org/clueweb09/FACC1/Cited by*, 5, 2013.

Matt Gardner and Tom M Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, 2015.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. Improving learning and inference in a large knowledge-base using latent syntactic cues.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. 2014.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *HLT-NAACL*, pages 851–861, 2015.

Shangpu Jiang, Daniel Lowd, and Dejing Dou. Learning to refine an automatically extracted knowledge base using markov logic. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 912–917. IEEE, 2012.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ni Lao and William W Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics, 2011.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Yuanfei Luo, Quan Wang, Bin Wang, and Li Guo. Context-dependent knowledge graph embedding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1656–1661, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1191.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

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

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China, July 2015. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

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.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *Proceedings of the 12th International Semantic Web Conference-Part I*, pages 542–557. Springer-Verlag New York, Inc., 2013.

Ray Richardson and Alan F Smeaton. Using wordnet in a knowledge-based approach to information retrieval. 1995.

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

Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. Knowledge base completion via coupled path ranking. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1318, Berlin, Germany, August 2016. Association for Computational Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. AAAI Press, 2014.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM, 2014.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.