Differentiable Learning of Logical Rules for Knowledge Base Reasoning

Fan Yang, Zhilin Yang, William W. Cohen (2017)

Presented by Benjamin Striner, 10/17/2017

Contents

- Why logic?
- Tasks and datasets
- Model
- Results

Why Logical Rules?

- Logical rules have the potential to generalize well
- Logical rules are explainable and understandable
- Train and test entities do not need to overlap

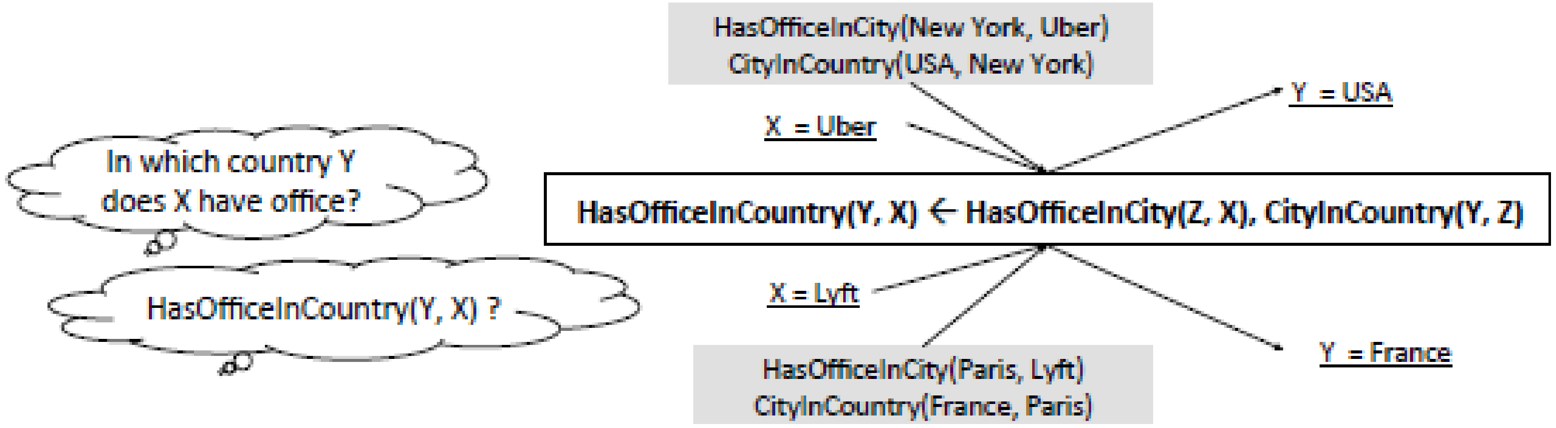

Figure 1: Using logical rules (shown in the box) for knowledge base reasoning.

Learning logical rules

- Goal is to learn logical rules (simple inference rules)
- Each rule has a confidence (alpha)

$$\alpha \quad \texttt{query}(\texttt{Y},\texttt{X}) \leftarrow \texttt{R}_\texttt{n}(\texttt{Y},\texttt{Z}_\texttt{n}) \wedge \cdots \wedge \texttt{R}_1(\texttt{Z}_1,\texttt{X})$$

Table 3: Examples of logical rules learned by Neural LP on FB15KSelected.

1.00	partially_contains(C,A) ← contains(B,A) ∧ contains(B,C)
0.45	partially_contains(C,A) ← contains(A,B) ∧ contains(B,C)
0.35	partially_contains(C,A) ← contains(C,B) ∧ contains(B,A)
1.00	marriage_location(C,A) ← nationality(C,B) ∧ contains(B,A)
0.35	marriage_location(B,A) ← nationality(B,A)
0.24	marriage_location(C,A) ← place_lived(C,B) ∧ contains(B,A)
1.00	film_edited_by(B,A) ← nominated_for(A,B)
0.20	film_edited_by(C,A) ← award_nominee(B,A) ∧ nominated_for(B,C)

Dataset and Tasks

Tasks

- Knowledge base completion
- Grid path finding
- Question answering

Knowledge Base Completion

- Training knowledge base is missing edges
- Predict the missing relationships

Knowledge Base Completion Datasets

- Wordnet
- Freebase
- Unified Medical Language System (UMLS)
- Kinship: relationships among a tribe

Table 4: Datasets statistics.

	# Data	# Relation	# Entity
UMLS	5960	46	135
Kinship	9587	25	104

Table 1: Knowledge base completion datasets statistics.

Dataset	# Facts	# Train	# Test	# Relation	# Entity
WN18	106,088	35,354	5,000	18	40,943
FB15K	362,538	120,604	59,071	1,345	14,951
FB15KSelected	204,168	67,947	20,466	237	14,541

Grid path finding

- Generate 16x16 grid, relationships are directions
- Allows large but simple dataset
- Evaluated similarly to KBC

Question answering

- KB contains tuples of movie information
- Answer natural language (but simple) questions

Table 6: A subset of the WIKIMOVIES dataset.

Knowledge base	directed_by (Blade Runner, Ridley Scott) written_by (Blade Runner, Philip K. Dick) starred_actors (Blade Runner, Harrison Ford) starred_actors (Blade Runner, Sean Young)
Questions	What year was the movie Blade Runner released? Who is the writer of the film Blade Runner?

Model

TensorLog

- Matrix multiplication can be used for simple logic
- E are entities
  - Encoded as one-hot vector v
- R are relationships
  - Encoded as adjacency matrix M
- P(Y,Z)^Q(Z,X) = Mp*Mq*vx

Learning a rule

- Rule is a product over relationship matrices
- Each rule has a confidence (alpha)
- L indexes over all rules
- Objective is to select rule that results in best score
- Many possible rules

$$\sum_l \alpha_l \Pi_{\mathrm{k}\in\beta_l} \mathrm{M}_{\mathrm{R_k}} \qquad \mathrm{s} = \sum_l \left(\alpha_l \left(\Pi_{\mathrm{k}\in\beta_l} \mathrm{M}_{\mathrm{R_k}} \mathrm{v_x}\right)\right), \ \mathrm{score}(\mathrm{y} \mid \mathrm{x}) = \mathrm{v}_{\mathrm{y}}^T \mathrm{s}$$

$$\max_{\{\alpha_l, \beta_l\}} \sum_{\{\mathrm{x,y}\}} \mathrm{score}(\mathrm{y} \mid \mathrm{x}) = \max_{\{\alpha_l, \beta_l\}} \sum_{\{\mathrm{x,y}\}} \mathrm{v}_{\mathrm{y}}^T \left(\sum_l \left(\alpha_l \left(\Pi_{\mathrm{k}\in\beta_l} \mathrm{M}_{\mathrm{R_k}} \mathrm{v_x}\right)\right) \right)$$

Differentiable rules

- Exchange product and sum
- Now learning a single rule, each step is combination of relationships

$$\prod_{t=1}^{T} \sum_{\mathrm{k}}^{|\mathbf{R}|} a_t^{\mathrm{k}} \mathrm{M}_{\mathrm{R_k}}$$

Attention and recurrence

- Attention over previous memories “memory attention vector” (b)
- Attention over relationship matrices “operator attention vector” (a)
- Controller (next slide) determines attention

$$\mathbf{u_0} = \mathbf{v_x}$$

$$\mathbf{u_t} = \sum_{\mathbf{k}}^{|\mathbf{R}|} a_t^{\mathbf{k}} \mathrm{M}_{\mathrm{R_k}} \left(\sum_{\tau=0}^{t-1} b_t^{\tau} \mathbf{u}_{\tau} \right) \quad \text{for } 1 \leq t \leq T$$

$$\mathbf{u_{T+1}} = \sum_{\tau=0}^{T} b_{T+1}^{\tau} \mathbf{u}_{\tau}$$

Controller

- Recurrent controller produces attention vectors
  - Input is query (END token when t=T+1)
  - Query is embedded in continuous space
  - LSTM used for recurrence

$$\mathbf{h_t} = \text{update}\left(\mathbf{h_{t-1}}, \text{input}\right)$$
$$\mathbf{a_t} = \text{softmax}\left(W\mathbf{h_t} + b\right)$$
$$\mathbf{b_t} = \text{softmax}\left(\left[\mathbf{h_0}, \ldots, \mathbf{h_{t-1}}\right]^T \mathbf{h_t}\right)$$

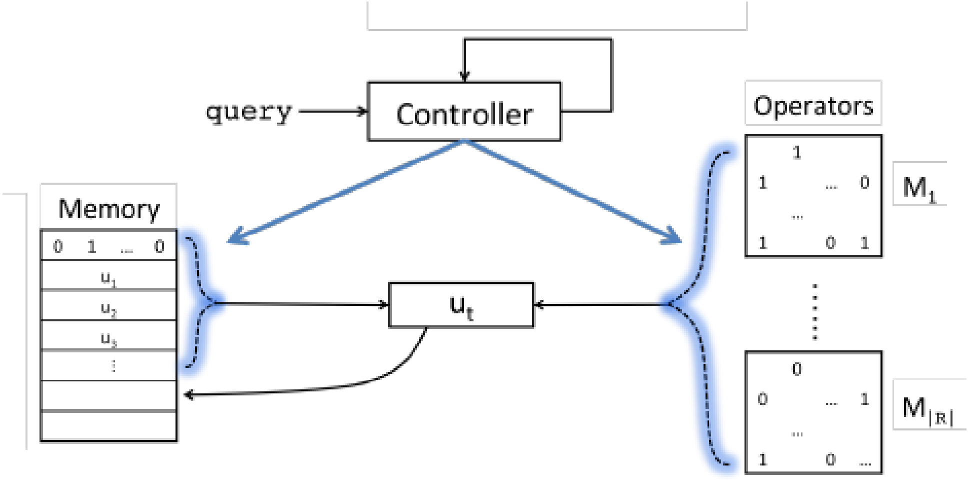

Figure 2: The neural controller system.

Objective

- Maximize $\log \mathbf{v}_{\mathbf{y}}^{T}\mathbf{u}$
- (Relationships and entities are positive)
- No max-margin, negative sampling, etc.

Recovering logical rules

Algorithm 1 Recover logical rules from attention vectors

Input: attention vectors $\{\mathbf{a_t} \mid t = 1, \ldots, T\}$ and $\{\mathbf{b_t} \mid t = 1, \ldots, T + 1\}$
Notation: Let $R_t = \{r_1, \ldots, r_l\}$ be the set of partial rules at step t. Each rule r_l is represented by a pair of (α, β) as described in Equation 1, where α is the confidence and β is an ordered list of relation indexes.
Initialize: $R_0 = \{r_0\}$ where $r_0 = (1, ())$.
for $t \leftarrow 1$ to $T + 1$ **do**
  Initialize: $\widehat{R_t} = \emptyset$, a placeholder for storing intermediate results.
  for $\tau \leftarrow 0$ to $t - 1$ **do**
    for rule (α, β) in R_τ **do**
      Update $\alpha \leftarrow \alpha \cdot b_t^\tau$. Store the updated rule (α, β) in $\widehat{R_t}$.
  if $t \leq T$ **then**
    Initialize: $R_t = \emptyset$
    for rule (α, β) in $\widehat{R_t}$ **do**
      for $k \leftarrow 1$ to $|\mathbf{R}|$ **do**
        Update $\alpha \leftarrow \alpha \cdot a_t^k$, $\beta \leftarrow \beta$ append k. Add the updated rule (α, β) to R_t.
  else
    $R_t = \widehat{R_t}$
return R_{T+1}

Results

KBC Results

- Outperforms previous work

Table 2: Knowledge base completion performance comparison. TransE [4] and Neural Tensor Network [24] results are extracted from [29]. Results on FB15KSelected are from [25]. Ensemble results are in the parentheses.

	WN18		FB15K		FB15KSelected	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
Neural Tensor Network	0.53	66.1	0.25	41.4	-	-
TransE	0.38	90.9	0.32	53.9	-	-
DISTMULT [29]	0.83	94.2	0.35	57.7	0.25	40.8
Node+LinkFeat [25]	0.94	94.3	0.82	87.0	0.23 (0.27)	34.7 (42.8)
Implicit ReasoNets [23]	-	95.3	-	**92.7**	-	-
Neural LP	**0.99**	**99.8**	**0.83**	91.6	**0.31**	**49.3**

Details

- FB15KSelected is harder because it removes inverse relationships
  - Augment by adding all inverse relationships
- Many possible relationships
  - Restrict to top 128 relationships that have entities in common with query
- Maximum rule length is 2 for all datasets

Additional KBC results

- Performance on UMLS and Kinship

	ISG		Neural LP	
	$T = 2$	$T = 3$	$T = 2$	$T = 3$
UMLS	43.5	43.3	**69.6**	66.4
Kinship	59.2	59.0	**73.3**	72.8

Grid Path Finding results

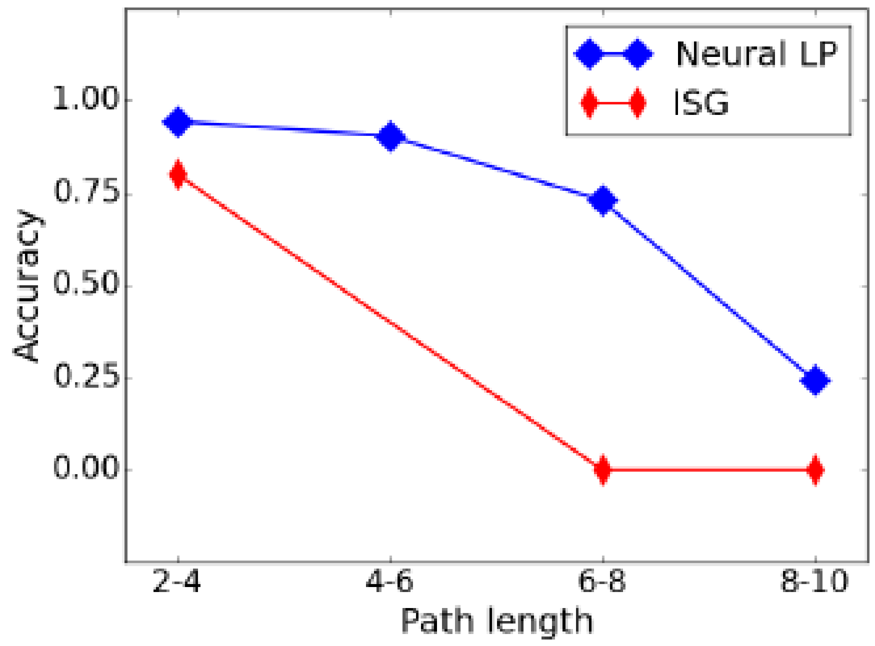

Figure 3: Accuracy on grid path finding.

QA Results

Table 7: Performance comparison. Memory Network is from [28]. QA system is from [4].

Model	Accuracy
Memory Network	78.5
QA system	93.5
Key-Value Memory Network [16]	93.9
Neural LP	**94.6**

QA implementation details

- Identify tail word as the word that is in the database
- Query is mean of embeddings of words
- Limit to 6 word queries and only top 100 most frequent words

Questions/Discussion