Differentiable Learning of Logical Rules for Knowledge Base Reasoning

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)

\[
\alpha \quad \text{query} (Y, X) \leftarrow R_n (Y, Z_n) \land \cdots \land R_1 (Z_1, X)
\]

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

<table>
<thead>
<tr>
<th>Confidence</th>
<th>Rule Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td><code>partially_contains(C,A) ← contains(B,A) \land contains(B,C)</code></td>
</tr>
<tr>
<td>0.45</td>
<td><code>partially_contains(C,A) ← contains(A,B) \land contains(B,C)</code></td>
</tr>
<tr>
<td>0.35</td>
<td><code>partially_contains(C,A) ← contains(C,B) \land contains(B,A)</code></td>
</tr>
<tr>
<td>1.00</td>
<td><code>marriage_location(C,A) ← nationality(C,B) \land contains(B,A)</code></td>
</tr>
<tr>
<td>0.35</td>
<td><code>marriage_location(B,A) ← nationality(B,A)</code></td>
</tr>
<tr>
<td>0.24</td>
<td><code>marriage_location(C,A) ← place_lived(C,B) \land contains(B,A)</code></td>
</tr>
<tr>
<td>1.00</td>
<td><code>film_edited_by(B,A) ← nominated_for(A,B)</code></td>
</tr>
<tr>
<td>0.20</td>
<td><code>film_edited_by(C,A) ← award_nominee(B,A) \land nominated_for(B,C)</code></td>
</tr>
</tbody>
</table>
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 1: Knowledge base completion datasets statistics.

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

Table 4: Datasets statistics.

<table>
<thead>
<tr>
<th></th>
<th># Data</th>
<th># Relation</th>
<th># Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMLS</td>
<td>5960</td>
<td>46</td>
<td>135</td>
</tr>
<tr>
<td>Kinship</td>
<td>9587</td>
<td>25</td>
<td>104</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Knowledge base</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>directed_by</td>
<td>What year was the movie Blade Runner released?</td>
</tr>
<tr>
<td>written_by</td>
<td>Who is the writer of the film Blade Runner?</td>
</tr>
<tr>
<td>starred_actors</td>
<td>The starred actors of Blade Runner:</td>
</tr>
<tr>
<td></td>
<td>Ridley Scott, Harrison Ford, Sean Young</td>
</tr>
</tbody>
</table>
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
• $P(Y,Z) \land Q(Z,X) = M_p \ast M_q \ast v_x$
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

\[ s = \sum_l \left( \alpha_l \left( \prod_{k \in \beta_l} M_{R_k} v_x \right) \right) , \text{score}(y \mid x) = v_y^T s \]

\[ \max_{\{\alpha_l, \beta_l\}} \sum_{\{x,y\}} \text{score}(y \mid x) = \max_{\{\alpha_l, \beta_l\}} \sum_{\{x,y\}} v_y^T \left( \sum_l \left( \alpha_l \left( \prod_{k \in \beta_l} M_{R_k} v_x \right) \right) \right) \]
Differentiable rules

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

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

\[
\begin{align*}
  u_0 &= v_x \\
  u_t &= \sum_k a_t^k M_{R_k} \left( \sum_{\tau=0}^{t-1} b_t^\tau u_\tau \right) \quad \text{for } 1 \leq t \leq T \\
  u_{T+1} &= \sum_{\tau=0}^T b_{T+1}^\tau u_\tau
\end{align*}
\]
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

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

*Figure 2: The neural controller system.*
Objective

- Maximize $\log v_y^T u$
- (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 \{a_t | t = 1, \ldots, T\} and \{b_t | t = 1, \ldots, T + 1\}

Notation: Let \( R_t = \{r_1, \ldots, r_t\} \) be the set of partial rules at step \( t \). Each rule \( r_t \) is represented by a pair of \((\alpha, \beta)\) as described in Equation 1, where \( \alpha \) is the confidence and \( \beta \) is an ordered list of relation indexes.

Initialize: \( R_0 = \{r_0\} \) where \( r_0 = (1, \phi) \).

for \( t \leftarrow 1 \) to \( T + 1 \) do

Initialize: \( \hat{R}_t = \emptyset \), a placeholder for storing intermediate results.

for \( \tau \leftarrow 0 \) to \( t - 1 \) do

for rule \((\alpha, \beta)\) in \( R_\tau \) do

Update \( \alpha \leftarrow \alpha \cdot b^T_t \). Store the updated rule \((\alpha, \beta)\) in \( \hat{R}_t \).

if \( t \leq T \) then

Initialize: \( R_t = \emptyset \)

for rule \((\alpha, \beta)\) in \( \hat{R}_t \) do

for \( k \leftarrow 1 \) to \(|R_t|\) do

Update \( \alpha \leftarrow \alpha \cdot a^k_t \), \( \beta \leftarrow \beta \) append \( k \). Add the updated rule \((\alpha, \beta)\) to \( R_t \).

else

\( R_t = \hat{R}_t \)

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.

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

<table>
<thead>
<tr>
<th></th>
<th>ISG</th>
<th>Neural LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>T = 2</td>
<td>T = 3</td>
<td>T = 2</td>
</tr>
<tr>
<td>UMLS</td>
<td>43.5</td>
<td>69.6</td>
</tr>
<tr>
<td>Kinship</td>
<td>59.2</td>
<td>73.3</td>
</tr>
</tbody>
</table>
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].

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Network</td>
<td>78.5</td>
</tr>
<tr>
<td>QA system</td>
<td>93.5</td>
</tr>
<tr>
<td>Key-Value Memory Network [16]</td>
<td>93.9</td>
</tr>
<tr>
<td>Neural LP</td>
<td>94.6</td>
</tr>
</tbody>
</table>
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