Knowledge Graph Embeddings: Recent Advances

William Wang

UC SANTA BARBARA

Joint work with Liwei Cai
CIPS Summer School 2018
Outline

• Related Work
• KBGAN: Algorithm
• Experiments
• Conclusion
Related Work

- Embedding-based method
  - RESCAL, Nickel et al, 2011
  - TransE, Bordes et al, 2013
  - Neural Tensor Network, Socher et al, 2013
  - TransR/CTransR, Lin et al, 2015
  - Complex Embeddings, Trouillon et al, 2016

Embedding methods allow us to compare, and find similar entities in the vector space.
RESCAL (Nickel et al., 2011)

• Tensor factorization on the
  • (head)entity-(tail)entity-relation tensor.
TransE (Bordes et al., 2013)

• Assumption: in the vector space, when adding the relation to the head entity, we should get close to the target tail entity.

• Margin based loss function:
  • Minimize the distance between \((h+l)\) and \(t\).
  • Maximize the distance between \((h+l)\) to a randomly sampled tail \(t'\) (negative example).

\[
\mathcal{L} = \sum_{(h,l,t) \in S} \sum_{(h',l,t') \in S'_h} \left[ \gamma + d(h + l, t) - d(h' + l, t') \right]_+
\]
Neural Tensor Networks (Socher et al., 2013)

- Model the bilinear interaction between entity pairs with tensors.

\[
U^T f(e_1^T W^{[1:2]} e_2 + V e_1 + b)
\]
Poincaré Embeddings (Nickel and Kiela, 2017)

• Idea: learn hierarchical KB representations by looking at hyperbolic space.

\[ d(u, v) = \text{arcosh} \left( 1 + 2 \frac{||u - v||^2}{(1 - ||u||^2)(1 - ||v||^2)} \right). \]

Figure 1: (a) Due to the negative curvature of \( B \), the distance of points increases exponentially (relative to their Euclidean distance) the closer they are to the boundary. (c) Growth of the Poincaré distance \( d(u, v) \) relative to the Euclidean distance and the norm of \( v \) (for fixed \( ||u|| = 0.9 \)). (b) Embedding of a regular tree in \( B^2 \) such that all connected nodes are spaced equally far apart (i.e., all black line segments have identical hyperbolic length).
ConvE (Dettmers et al, 2018)

• 1. Reshape the head and relation embeddings into “images”.
• 2. Use CNNs to learn convolutional feature maps.
It all started in 2013

Me: How did you get negative examples from knowledge graphs?

William Cohen: We did some samplings from the knowledge graph.

Me: OK... 😐
Reality about Knowledge Bases

• Only positive facts are stored, and no negative examples are stored.
  • This makes sense, for efficiency considerations.

• But for machine learning (e.g., margin-based models)
  • We often need negative examples.
Negative Sampling is Pervasive

- TransE (Bordes et al., 2013): Replace head/tail entity with a randomly sampled entity from KB to create a negative example.

- Margin-based loss function:
  - Positive Examples: Minimize the distance between \((h+l)\) and \(t\).
  - Negative Examples: Maximize the distance between \((h+l)\) to a randomly sampled tail \(t'\) (negative example).

\[
\mathcal{L} = \sum_{(h,\ell,t) \in S} \sum_{(h',\ell,t') \in S'_{(h,\ell,t)}} \left[ \gamma + d(h + \ell, t) - d(h' + \ell, t') \right]_+
\]
Negative Sampling’s Main Issue

• Main Issue for KB Embedding:
  • It often generates low-quality negative examples that do not help you learn good embedding models.

LocatedIn(NewOrleans, Google)
KBGAN: Learning to Generate High-Quality Negative Examples

Idea: use adversarial learning to iteratively learn better negative examples.
KBGAN: Overview

• Both G and D are KG embedding models.

• Input:
  • Pre-trained generator G with score function $f_G(h,r,t)$.
  • Pre-trained discriminator D with score function $f_D(h,r,t)$.

• Adversarial Learning:
  • Use softmax to score and rank negative triples.
  • Update D with original positive examples and highly-ranked negative examples.
  • Pass the reward for policy gradient update for G.

• Output:
  • Adversarially trained KG embedding discriminator D.
KBGAN: Adversarial Negative Training

For each positive triple from the minibatch:

1. Generator: Rank negative examples.

\[
p_G(h', r, t'|h, r, t) = \frac{\exp f_G(h', r, t')}{\sum_{(h^*, r, t^*) \in \text{Neg}(h, r, t)} \exp f_G(h^*, r, t^*)}
\]


\[
L_D = \sum_{(h, r, t) \in \mathcal{T}} \left[ f_D(h, r, t) - f_D(h', r, t') + \gamma \right]_+ \\
(h', r, t') \sim p_G(h', r, t'|h, r, t) \quad (3)
\]
KBGAN: Adversarial Training (cont’d)

3. Compute the Reward for the Generator.
   \[ r = -f_D(h', r, t'). \]

4. Policy gradient update for the Generator.
   \[ G_G \leftarrow G_G + (r - b) \nabla_{\theta_G} \log p_s; \]

The baseline b is total reward sum / mini-batch size.
Experimental Settings

• Datasets: three standard KB completion datasets.
• Hyperparameters: documented in details in the paper.
• Metrics: Hits@10 and Mean Reciprocal Rank (MRR).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#r</th>
<th>#ent.</th>
<th>#train</th>
<th>#val</th>
<th>#test</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15k-237</td>
<td>237</td>
<td>14,541</td>
<td>272,115</td>
<td>17,535</td>
<td>20,466</td>
</tr>
<tr>
<td>WN18</td>
<td>18</td>
<td>40,943</td>
<td>141,442</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>WN18RR</td>
<td>11</td>
<td>40,943</td>
<td>86,835</td>
<td>3,034</td>
<td>3,134</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Method</th>
<th>FB15k-237</th>
<th></th>
<th>WN18</th>
<th></th>
<th>WN18RR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>H@10</td>
<td>MRR</td>
<td>H@10</td>
<td>MRR</td>
<td>H@10</td>
</tr>
<tr>
<td><strong>TRANSE</strong></td>
<td>-</td>
<td>42.8†</td>
<td>-</td>
<td>89.2</td>
<td>-</td>
<td>43.2†</td>
</tr>
<tr>
<td><strong>TRANSd</strong></td>
<td>-</td>
<td>45.3†</td>
<td>-</td>
<td>92.2</td>
<td>-</td>
<td>42.8†</td>
</tr>
<tr>
<td><strong>DISTMult</strong></td>
<td>24.1†</td>
<td>41.9†</td>
<td>82.2</td>
<td>93.6</td>
<td>42.5†</td>
<td>49.1†</td>
</tr>
<tr>
<td><strong>COMPLEX</strong></td>
<td>24.0†</td>
<td>41.9†</td>
<td>94.1</td>
<td>94.7</td>
<td><strong>44.4†</strong></td>
<td><strong>50.7†</strong></td>
</tr>
<tr>
<td><strong>TRANSE</strong> (pre-trained)</td>
<td>24.2</td>
<td>42.2</td>
<td>43.3</td>
<td>91.5</td>
<td>18.6</td>
<td>45.9</td>
</tr>
<tr>
<td><strong>KBGAN</strong> (TRANSE + DISTMult)</td>
<td>27.4</td>
<td>45.0</td>
<td>71.0</td>
<td><strong>94.9</strong></td>
<td>21.3</td>
<td>48.1</td>
</tr>
<tr>
<td><strong>KBGAN</strong> (TRANSE + COMPLEX)</td>
<td><strong>27.8</strong></td>
<td>45.3</td>
<td>70.5</td>
<td><strong>94.9</strong></td>
<td>21.0</td>
<td>47.9</td>
</tr>
<tr>
<td><strong>TRANSd</strong> (pre-trained)</td>
<td>24.5</td>
<td>42.7</td>
<td>49.4</td>
<td>92.8</td>
<td>19.2</td>
<td>46.5</td>
</tr>
<tr>
<td><strong>KBGAN</strong> (TRANSd + DISTMult)</td>
<td><strong>27.8</strong></td>
<td><strong>45.8</strong></td>
<td>77.2</td>
<td>94.8</td>
<td>21.4</td>
<td>47.2</td>
</tr>
<tr>
<td><strong>KBGAN</strong> (TRANSd + COMPLEX)</td>
<td>27.7</td>
<td><strong>45.8</strong></td>
<td>77.9</td>
<td>94.8</td>
<td>21.5</td>
<td>46.9</td>
</tr>
</tbody>
</table>
Convergence Analysis
Conclusion

• We propose an adversarial learning approach for generating high-quality negative examples.

• Our approach is model-agnostic, and it can be applied to various knowledge graph embedding models.

• Our work has shown improvements with various settings on two datasets.
Thank you!

- Code: [https://github.com/cai-lw/KBGAN](https://github.com/cai-lw/KBGAN)