Overview

1. Task description
2. Introduction to the dataset
3. Baseline models
4. Trained models
5. Results
6. Conclusion & Lessons learned
Why Large Scale Graph Machine Learning?

- Many graph-structured real-world applications
  - Social Networks
  - Recommender Systems
  - Linked Web documents

- Real world data forms very large graphs
  - Billions of edges or millions of graphs

- Promising domain of active research
What was the Task?

- Construct GNN for large scale graph data
- Keep as much information as possible

- OGB-LSC @ KDD Cup 2021: Link prediction on large scale graphs
  - Multi-relational graph
  - Graph consists of head-relation-tail triples
  - Predict the correct tail for a given head-relation pair
  - Provide the sorted top 10 tails for a given sample of 1001
## The Dataset • WikiKG90M

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of entities</td>
<td>87,143,637</td>
</tr>
<tr>
<td>Number of relations</td>
<td>1,315</td>
</tr>
<tr>
<td>Relation occurrence max</td>
<td>174,439,560</td>
</tr>
<tr>
<td>Relation occurrence mean</td>
<td>381,110.63</td>
</tr>
<tr>
<td>Number of feature dimensions</td>
<td>768</td>
</tr>
<tr>
<td>Number of training samples</td>
<td>501,160,482</td>
</tr>
</tbody>
</table>
The Dataset • WikiKG90M

<table>
<thead>
<tr>
<th>Degree</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outgoing mean</td>
<td>5.75</td>
</tr>
<tr>
<td>Outgoing max</td>
<td>8,320</td>
</tr>
<tr>
<td>Incoming mean</td>
<td>5.75</td>
</tr>
<tr>
<td>Incoming max</td>
<td>36,424,411</td>
</tr>
</tbody>
</table>

The graph shows the distribution of degrees for outgoing and incoming fractions of all nodes. The table lists the mean and maximum degrees for both outgoing and incoming connections.
Baseline Models
Entity Co-Occurrence

- Untrained baseline model
- Relations are ignored completely

- Scoring is based on head and tail occurrences
  - Given a head, the most common tail is scored 1
  - All other tails are scored 0
Pseudo Typing

- Untrained baseline model
- Heads are ignored completely

- Scoring is based on relation and tail occurrences
  - Given a relation, the most common tail is scored 1
  - All other tails are scored 0
Entity Co-Occurrence & Pseudo Typing

- Trained baseline model
- Uses the trained Entity Co-Occurrence & Pseudo Typing baselines as input

- Scores are derived from the input baselines
  - Weighted sum of the individual baseline scores
Trained Models
MLP Embeddings Model

- Uses triples as input, no features
- Entities and relations are represented in vector space
  - Initiate embeddings with random values
  - Embeddings are passed through MLP
  - An interaction function is applied to the final embeddings
- High number of entities forces low dimensional embeddings
- MLP to increase complexity (16 ➔ 64 ➔ 32)
- Negative samples are generated by corruption
Entity & Relation Feature Model

- Uses entity and relation features
  - Features are loaded on demand
  - Features are passed through MLP for “enhancement”
  - An interaction function is applied to the enhanced features

- Feature loading makes up most of training duration
- Negative samples are generated by corruption
ComplEx with PyTorch BigGraph

- Distributed system for learning graph embeddings
- Designed for very large graphs
- Used to train the ComplEx Model:
  - Semantic matching model
  - Calculates the matching latent semantics of entities and relations embodied in their vector space representations
  - Based on complex Embeddings
Graph Convolutional Networks (GCNs)

1. Nodes are represented by a vector
2. Vectors get aggregated for each node (“message”)
3. The vector of the current node gets updated using the messages
4. Process can be repeated by multiple layers
SGCN

- Simplifying Graph Convolutional Networks
- Majority of the benefit arises from the local averaging
- Power of GCNs originates primarily from the repeated graph propagation
- Reduces complexity through removing the nonlinearities
- Does not negatively impact accuracy
CompGCN

- Composition-based Multi-Relation GCN
- Uses inverse Edges
- Uses Embeddings for Relations and Entities
  - Using separate weight matrix for relations
  - Access direct relation representations
  - Facing over-parameterization
- Special: Using subgraphs as batches
CompGCN with Graph Attention Layer Model

- Take the idea of “Attention”, to give each neighbour node unique weights
- Take the strategy of mask graph attention.
  - Calculate attention coefficient with neighbour
  - Multi-head Aggregation
- Benefits comparing with basic GCN
  - Can do inductive job
  - Give different weights to neighbourhoods
Results
Custom Split & Ranking

- **OGB dataset split**
  - Temporal split (September/October/November)
  - No access to testing data solution

- **Custom dataset split**
  - Random split from OGB training data
  - Same split percentages as the OGB split

- **Evaluation metric: Mean Reciprocal Rank (MRR)**
<table>
<thead>
<tr>
<th>Model Name</th>
<th>OGB Validation MRR</th>
<th>Custom Validation MRR</th>
<th>Custom Training MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Co-Occurrence</td>
<td>0.0030</td>
<td>0.0123</td>
<td>0.0122</td>
</tr>
<tr>
<td>Pseudo Typing</td>
<td>0.2280</td>
<td>0.1569</td>
<td>0.1569</td>
</tr>
<tr>
<td>Entity Co-Occurrence + Pseudo Typing</td>
<td>0.2281</td>
<td>0.1663</td>
<td>0.1662</td>
</tr>
<tr>
<td>Entity &amp; Relation Features</td>
<td>0.4649</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MLP Embeddings</td>
<td>-</td>
<td>0.1485*</td>
<td>-</td>
</tr>
<tr>
<td>ComplEx with PyTorch BigGraph</td>
<td>-</td>
<td>-</td>
<td>0.0340*</td>
</tr>
<tr>
<td>CompGCN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CompGCN with Graph Attention</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SGCN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Random Model</td>
<td>0.0029 (50 runs)</td>
<td>0.0029 (50 runs)</td>
<td>0.0029 (50 runs)</td>
</tr>
</tbody>
</table>

* intermediary results on subset
Conclusion & Lessons Learned

● Working with large datasets is hard
  ○ Matrices, embeddings etc. get very large very quickly
  ○ Working memory must be managed efficiently
  ○ Frequent movement between CUDA and working memory
  ○ Project management and better training strategy are important

● Best performing model uses features
  ○ However, most models are still being trained and are improving
  ○ Difficult to draw conclusions currently.
We are happy to take your questions now!
Appendix

Score-Function for ComplEx Model: \( f_r(h, t) = \text{Re}(h^\top \text{diag}(r)t) = \text{Re}(\sum_{i=0}^{d-1} [r]_i \cdot [h]_i \cdot [t]_i) \)

DistMult
Appendix

Mean Reciprocal Rank:

\[
\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}
\]