A Review of Relational Machine Learning for Knowledge Graphs

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich

(arXiv:1503.00759v1)

Discussion by: Piyush Rai

June 12, 2015
Learning from Relational Data

- Non-Relational data: objects as features

- (Multi)-Relational data: objects as their relationship(s) to other objects
Outline

- Introduction to Knowledge Graphs
  - Knowledge Representation
  - Open vs Closed World Assumption
  - Knowledge Base Construction
  - Uses of Knowledge Graphs
  - Typical Learning Tasks on Knowledge Graphs

- Statistical Relational Learning on Knowledge Graphs
  - Problem Formulation and Training Data Generation
  - Penalized Maximum Likelihood Training
  - Pairwise Loss Training
  - Latent Feature Models and Graph Feature Models

- Latent Feature Models
- Current and Future Directions
Knowledge Graph Representation

**Figure:** Nodes: Entities; Edges: Relations

- Can be extracted from unstructured/semi-structured data and stored using triplets of the form **subject-predicate-object** or **entity-relation-entity**

**Leonard Nimoy** was an actor who played the character **Spock** in the science-fiction movie **Star Trek**

can be expressed via the following set of SPO triples:

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Leonard Nimoy,</td>
<td>profession,</td>
<td>Actor)</td>
</tr>
<tr>
<td>(Leonard Nimoy,</td>
<td>starredIn,</td>
<td>Star Trek)</td>
</tr>
<tr>
<td>(Leonard Nimoy,</td>
<td>played,</td>
<td>Spock)</td>
</tr>
<tr>
<td>(Spock,</td>
<td>characterIn,</td>
<td>Star Trek)</td>
</tr>
<tr>
<td>(Star Trek,</td>
<td>genre,</td>
<td>Science Fiction)</td>
</tr>
</tbody>
</table>
Open vs Closed World Assumption

- Closed world assumption (CWA): Non-existing triplet = false relationship

- Open world assumption (OWA): Non-existing triplet = unknown relationship
  - More appropriate as knowledge graphs are highly incomplete

- Local-closed world assumption (LCWA)
  - Once we have observed \((e_i, r_k, e_j)\), any non-existing \((e_i, r_k, .)\) is indeed false
  - Appropriate for functional relations (e.g., bornIn)
Knowledge Graph / Knowledge Base Construction

- **Curated approaches**
  - Triplets are created manually by a closed group of experts
  - Data is reliable; algorithms can easily get high accuracies

- **Collaborative approaches**
  - Triplets are created manually by an open group of volunteers
  - Data is reliable but incomplete; algorithms can easily get high accuracies

- **Automatic Knowledge Base Construction (AKBC)**
  - Automated semi-structured approaches: Triplets extracted automatically from semi-structured text such as infoboxes in Wikipedia, via hand-crafted rules, learned rules, regular expressions, etc.
  - Automated unstructured approaches: Triplets extracted automatically from unstructured text via Machine Learning and NLP techniques

Some Real-World Knowledge Bases

- Schema-based: Entities and relations have unique identifiers
- Schema-free: Multiple entities/relations could refer to the same semantics (e.g., bornIn and placeOfBirth, both may be present in the knowledge base)

### Knowledge Base Construction Projects

<table>
<thead>
<tr>
<th>Creation Method</th>
<th>Schema-Based Projects</th>
<th>Schema-Free Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curated</td>
<td>Cyc/OpenCyc, WordNet, UMLS</td>
<td>ReVerb, OLLIE, PRISMATIC</td>
</tr>
<tr>
<td>Collaborative</td>
<td>Wikidata, Freebase</td>
<td></td>
</tr>
<tr>
<td>Auto. Semi-Structured</td>
<td>YAGO, DBPedia, Freebase</td>
<td></td>
</tr>
<tr>
<td>Auto. Unstructured</td>
<td>Knowledge Vault, NELL, PATTY, DeepDive/Elementary</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROSPERA</td>
</tr>
</tbody>
</table>

### Size of Some Schema-Based Knowledge Bases

<table>
<thead>
<tr>
<th>Knowledge Graph</th>
<th>Number of Entities</th>
<th>Relation Types</th>
<th>Facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>40 M</td>
<td>35,000</td>
<td>637 M</td>
</tr>
<tr>
<td>Wikidata</td>
<td>13 M</td>
<td>1,643</td>
<td>50 M</td>
</tr>
<tr>
<td>DBpedia</td>
<td>4.6 M</td>
<td>1,367</td>
<td>68 M</td>
</tr>
<tr>
<td>YAGO2</td>
<td>10 M</td>
<td>72</td>
<td>120 M</td>
</tr>
<tr>
<td>Google Knowledge Graph</td>
<td>570 M</td>
<td>35,000</td>
<td>18,000 M</td>
</tr>
</tbody>
</table>
What might Knowledge Bases be useful for?

- Improved search results (Google’s Knowledge Graph; Microsoft’s Satori)
- Question Answering systems (e.g., IBM’s Watson)
- Decision support systems in healthcare (e.g., LinkedLifeData)
Feature learning (i.e., embeddings) for entities and relations

Link-Prediction
- Discovering new facts from existing facts in the knowledge base
- Correcting wrong facts (e.g., Obama was born in Kenya) using reliable/correct facts (e.g., Obama is president of USA) in the knowledge base

Entity/Relation Resolution
- Obama, Barack Obama, 44th US President, all refer to the same person
- Born-in, place-of-birth, both refer to the same relation

Entity/Relation Clustering

Entity/Relation Ranking. E.g.,
- Given an entity $e_1$ and relation $(r)$, give a ranked list of entities $e_2$ on the other side of the relation ($e_1 - r - ?$)
- Given a pair of entities $(e_1, e_2)$, predict the most-likely relations ($e_1 - ? - e_2$)
Statistical Modeling of Knowledge Graphs

- Given: knowledge graph/knowledge base, consisting of \( N_e \) entities, \( N_r \) relations, and facts (triplets: entity1-relation-entity2)

- Set of entities \( \mathcal{E} = \{ e_1, \ldots, e_{N_e} \} \), set of relational \( \mathcal{R} = \{ r_1, \ldots, r_{N_r} \} \)

- Each possible triplet \( x_{ijk} = (e_i, r_k, e_j) \), with \( y_{ijk} = \{0, 1\} \) denoting its existence/validity

- Can store all possible triplets using a binary tensor \( \mathbf{Y} \in \{0, 1\}^{N_e \times N_e \times N_r} \)

- Interpretation of \( y_{ijk} = 0 \) depends on open/closed/local-closed world assumption. Number of 1s is usually very small in either case.
How to get negative examples?

- Most knowledge graphs contain only positive examples (no false facts)
- Thus, positive examples \((y_{ijk} = 1)\) are naturally given to us
- Denote positive examples \((e_i, r_k, e_j)\), s.t. \(y_{ijk} = 1\), by the set \(D^+\)
- How to generate the set \(D^-\) of negative examples (i.e., for which \(y_{ijk} = 0\))?
  - One way is to assume everything not in \(D^+\) to be negative (subject to the constraints on the entity/relation type). Such \(D^-\) can be very massive.
  - Another way is to generate \(D^-\) as
    \[
    D^- = \{(e_{\ell}, r_k, e_j) \mid e_i \neq e_{\ell} \land (e_i, r_k, e_j) \in D^+\} \\
    \cup \{(e_i, r_k, e_{\ell}) \mid e_j \neq e_{\ell} \land (e_i, r_k, e_j) \in D^+\}
    \]
  - Caveat: Still no guarantee that each entry in \(D^-\) is necessarily negative
The random variables $y_{ijk} \in \{0, 1\}$ in $Y$ are correlated with each other.

Three main ways to model the correlations:

- **M1:** $y_{ijk}$’s are iid given the latent features of entities and relations (latent feature models).
- **M2:** $y_{ijk}$’s are iid given observed graph features and additional parameters (graph feature models).
- **M3:** $y_{ijk}$’s have local interactions (Markov Random Fields).

M1 and M2 predict the existence of $y_{ijk}$ via a score function $f(x_{ijk}; \Theta)$.

Here is a typical probabilistic approach to parameter learning in M1 and M2:

$$
P(Y|D, \Theta) = \prod_{i=1}^{N_e} \prod_{j=1}^{N_e} \prod_{k=1}^{N_r} \text{Ber}(y_{ijk} | \sigma(f(x_{ijk}, \Theta))) 
$$

where $\sigma(u) = 1/(1 + e^{-u})$ is the sigmoid (logistic) function, and

$$
\text{Ber}(y|p) = \begin{cases} 
p & \text{if } y = 1 \\
1 - p & \text{if } y = 0 
\end{cases}
$$

Training via (penalized) maximum likelihood, or fully Bayesian inference.
Pairwise Loss based Training

- If we can't trust negatives to be really negative, have the model score them lower than the positives

\[
\min_{\Theta} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \mathcal{L}(f(x^+; \Theta), f(x^-; \Theta)) + \lambda \text{reg}(\Theta)
\]

where \( \mathcal{L}(f, f') \) is a margin-based ranking loss function such as

\[
\mathcal{L}(f, f') = \max(1 + f - f', 0).
\]

- Note: \( f \) can be a function or a probability model.

- Optimization-based, penalized ML, or Bayesian, any approach can be used for parameter estimation.

- Online methods preferred (sample one positive and one negative example in each round..)
Statistical Modeling of Knowledge Graphs

Two main approaches

- **Latent feature models**
  - Assume each entity $e_i$ to have an embedding $e_i \in \mathbb{R}^{H_e}$
  - Assume each relation type $r_k$ to be parameterized by some $W_k$
  - Define score of a triplet $(e_i, r_k, e_j)$ as some function $f(e_i, e_j, W_k)$, e.g.,
    
    $$f(e_i, e_j, W_k) = e_i^\top W_k e_j \quad \text{where } W_k \in \mathbb{R}^{H_e \times H_e}$$
    
    $$f(e_i, e_j, W_k) = -\text{dist}(e_i + W_k, e_j) \quad \text{where } W_k \in \mathbb{R}^{H_e}$$
  
  - Score can be turned into a probability if needed (e.g., via a logistic function)
  - Assumptions can be imposed on $e_i$'s and $W_k$'s (e.g., sparsity, non-negativity)

- **Graph feature models**
  - Score of a triplet $(e_i, r_k, e_j)$ depends on graph-based notions (e.g., number of all paths of some length $L$ or less, number of common neighbors).
  - Some commonly used methods: Katz index, Adamic-Adar index, Page Ranking algorithm

Latent feature models and graph feature models can also be combined (some recent work; see reference [103] in the survey paper)
More on Latent Feature Models

- Assume the $N_e \times N_h$ matrix $E = [e_1; \ldots; e_N]$ contains the latent features (i.e., embeddings) of the $N_e$ entities. $E$ is shared across all relations.

- A bilinear latent feature model for relation $r_k$ (parameterized by $N_h \times N_h$ matrix $W_k$) models the score as: $Y_k \approx EW_kE^\top$

- $y_{ijk} \approx e_i^\top W_k e_j = w_k^\top (e_i \otimes e_j)$ where $w_k = \text{vec}(W_k)$

- Basically, a linear model
Nonlinear Latent Feature Models

- Replace the linear mapping $f_{ijk} = w_k^\top (e_i \otimes e_j)$ by a nonlinear one

  $$f_{ijk} = w_k^\top g(h_a)$$

  (where $g$ is some nonlinear function)

  $$h_a = A_k^\top \phi_{ij}$$

  $$\phi_{ij} = [e_i; e_j]$$

- Another model:

  $$f_{ijk} = w^\top g(h_c)$$

  (where $g$ is some nonlinear function)

  $$h_c = C^\top \phi_{ijk}$$

  $$\phi_{ijk} = [e_i; e_j; r_k]$$
A hybrid, fusion-based architecture consisting of latent feature model, graph feature model, and an information extraction component.
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g., “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g.,
  - “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations.

- Incorporating other sources (e.g., text) in addition to knowledge base data.

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011).

- Models that support complex queries on probabilistic knowledge graphs, e.g.,
  - “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities).

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data).
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g.,
  - “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g., “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g., “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Extensions and Future Work

- Incorporating type constraints (e.g., relation “married-to” can only involve entities that correspond to “people”) or functional constraints (e.g., a person can be born in only one city).

- Generalizing to new entities and new relations

- Incorporating other sources (e.g., text) in addition to knowledge base data

- Including other dimensions such as time (e.g., Larry Page was Google CEO from 2001-2011)

- Models that support complex queries on probabilistic knowledge graphs, e.g.,
  - “Find the athlete who is from Romania who won gold in 3000m and bronze in 1500m in 1984 Olympics”

- Richer model structures (e.g., hierarchies/clusters among relations/entities)

- Scaling up to web-scale knowledge bases (also making the model depend only on the known facts, i.e., the 1s, in the data)
Thanks! Questions?