A Review of Relational Machine Learning for Knowledge Graphs

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

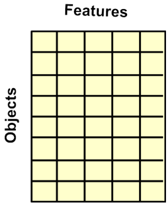
(arXiv:1503.00759v1)

Discussion by: Piyush Rai

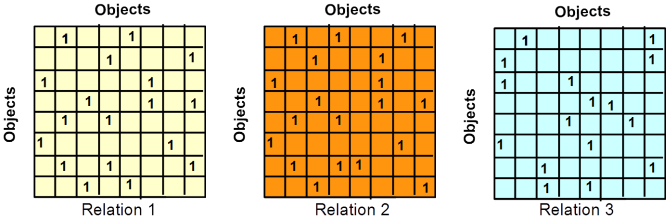
June 12, 2015

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

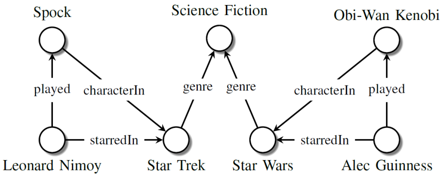


Figure: Nodes: Entities; Edges: Relations

- 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:

subject	***predicate***	***object***
(LeonardNimoy,	*profession,*	*Actor)*
(LeonardNimoy,	*starredIn,*	*StarTrek)*
(LeonardNimoy,	*played,*	*Spock)*
(Spock,	*characterIn,*	*StarTrek)*
(StarTrek,	*genre,*	*ScienceFiction)*

Open vs Closed World Assumption

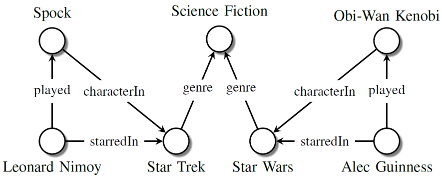


Figure: Nodes: Entities; Edges: Relations

- 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

Creation Method	**Schema-Based Projects**
Curated	Cyc/OpenCyc , WordNet , UMLS
Collaborative	Wikidata , Freebase
Auto. Semi-Structured	YAGO , DBPedia , Freebase
Auto. Unstructured	Knowledge Vault , NELL , PATTY DeepDive/Elementary , PROSPERA
Creation Method	**Schema-Free Projects**
Auto. Unstructured	ReVerb , OLLIE , PRISMATIC

SIZE OF SOME SCHEMA-BASED KNOWLEDGE BASES

	Number of		
Knowledge Graph	**Entities**	**Relation Types**	**Facts**
Freebase	40 M	35,000	637 M
Wikidata	13 M	1,643	50 M
DBpedia	4.6 M	1,367	68 M
YAGO2	10 M	72	120 M
Google Knowledge Graph	570 M	35,000	18,000 M

What might Knowledge Bases be useful for?

- Improved search results (Google's Knowledge Graph; Microsoft's Satori)

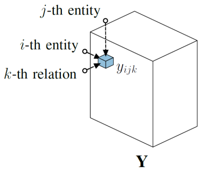
- Question Answering systems (e.g., IBM's Watson)
- Decision support systems in healthcare (e.g., LinkedLifeData)

Machine Learning with/for Knowledge Bases

- 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 $\mathcal{D}^+$
- How to generate the set $\mathcal{D}^-$ of negative examples (i.e., for which $y_{ijk} = 0$)?
  - One way is to assume everything not in $\mathcal{D}^+$ to be negative (subject to the constraints on the entity/relation type). Such $\mathcal{D}^-$ can be very massive.
  - Another way is to generate $\mathcal{D}^-$ as

$$\begin{aligned}\mathcal{D}^- =& \{(e_\ell, r_k, e_j) \mid e_i \neq e_\ell \wedge (e_i, r_k, e_j) \in \mathcal{D}^+\} \\ &\cup \{(e_i, r_k, e_\ell) \mid e_j \neq e_\ell \wedge (e_i, r_k, e_j) \in \mathcal{D}^+\}\end{aligned}$$

  - Caveat: Still no guarantee that each entry in $\mathcal{D}^-$ is necessarily negative

Statistical Modeling of Knowledge Graphs

- The random variables $y_{ijk} \in \{0, 1\}$ in $\mathbf{Y}$ 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

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

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

$$\mathrm{Ber}(y|p) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases} \qquad (2)$$

- 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

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

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 nagative example in each round..)

Statistical Modeling of Knowledge Graphs

Two main approaches

- **Latent feature models**
  - Assume each entity e_i to have an embedding $\mathbf{e}_i \in \mathbb{R}^{H_e}$
  - Assume each relation type r_k to be parameterized by some $\mathbf{W}_k$
  - Define score of a triplet (e_i, r_k, e_j) as some function $f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{W}_k)$, e.g.,

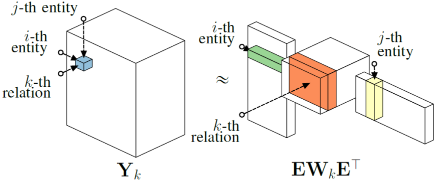
$$
\begin{aligned}
f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{W}_k) &= \mathbf{e}_i^\top \mathbf{W}_k \mathbf{e}_j \quad \text{where } \mathbf{W}_k \in \mathbb{R}^{H_e \times H_e} \\
f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{W}_k) &= -dist(\mathbf{e}_i + \mathbf{W}_k, \mathbf{e}_j) \quad \text{where } \mathbf{W}_k \in \mathbb{R}^{H_e}
\end{aligned}
$$

  - Score can be turned into a probability if needed (e.g., via a logistic function)
  - Assumptions can be imposed on $\mathbf{e}_i$'s and $\mathbf{W}_k$'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 $\mathbf{E} = [\mathbf{e}_1; \ldots; \mathbf{e}_N]$ contains the latent features (i.e., embeddings) of the N_e entities. $\mathbf{E}$ is shared across all relations
- A bilinear latent feature model for relation r_k (parameterized by $N_h \times N_h$ matrix $\mathbf{W}_k$) models the score as: $\mathbf{Y}_k \approx \mathbf{E}\mathbf{W}_k\mathbf{E}^\top$



- $y_{ijk} \approx \mathbf{e}_i^\top \mathbf{W}_k \mathbf{e}_j = \mathbf{w}_k^\top (\mathbf{e}_i \otimes \mathbf{e}_j)$ where $\mathbf{w}_k = vec(\mathbf{W}_k)$
- Basically, a linear model

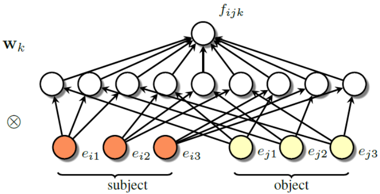
Nonlinear Latent Feature Models

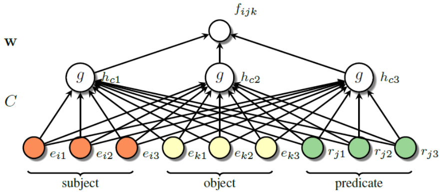
- Replace the linear mapping $f_{ijk} = \mathbf{w}_k^\top(\mathbf{e}_i \otimes \mathbf{e}_j)$ by a nonlinear one

$$\begin{aligned} f_{ijk} &= \mathbf{w}_k^\top g(\mathbf{h}_a) \qquad \text{(where } g \text{ is some nonlinear function)} \\ \mathbf{h}_a &= \mathbf{A}_k^\top \phi_{ij} \\ \phi_{ij} &= [\mathbf{e}_i; \mathbf{e}_j] \end{aligned}$$

- Another model:

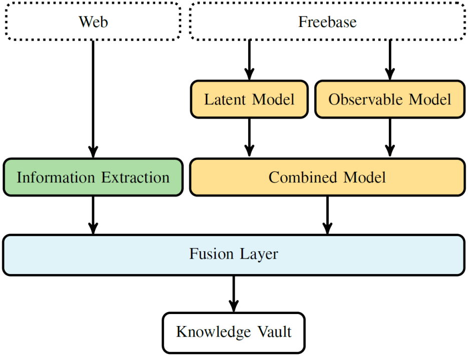
$$\begin{aligned} f_{ijk} &= \mathbf{w}^\top g(\mathbf{h}_c) \qquad \text{(where } g \text{ is some nonlinear function)} \\ \mathbf{h}_c &= \mathbf{C}^\top \phi_{ijk} \\ \phi_{ijk} &= [\mathbf{e}_i; \mathbf{e}_j; \mathbf{r}_k] \end{aligned}$$





Architecture of Google Knowledge Vault

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?