Reasoning in Knowledge Graphs using Deep Learning

Joint work with W. Hamilton, R. Ying, J. You, P. Bajaj, M. Zitnik, D. Jurafsky

Jure Leskovec
Knowledge Graph

“leonardo da vinci”
String

Leonardo da Vinci
Recognized entity

Leonardo da Vinci
Recognized entity
Related entities

Leonardo da Vinci
Recognized entity
Related entities
Named Relationship
Example KG
Many Domains

Knowledge Graph

- Wikidata
- Freebase
- Cyc
- GeoNames
- NELL
- PROSPERA
- YAGO
- DBpedia
- ConceptNet
- OpenIE
- GDelt
- WordNet
- Metaweb
- Knowledge Vault
Examples (1)
Examples (2)

- Biological interactions
- Online communities
Applications of KGs

The panels are generated from what’s called the Google Knowledge Graph

- Data comes from Wikipedia, CIA World Factbook, and other online sources.
- May be responsible for a significant decline in Wikipedia visits.
- Holds 70 billion facts [2017]
Applications of KGs

We want answers, not just links!
Applications in recommender systems:

Query formula:

\[ C_\varepsilon \exists P : \text{UPVOTE}(u, P) \land \text{BELONG}(P, C_\varepsilon) \]

“Predict communities \( C_\varepsilon \) in which user \( u \) is likely to upvote a post”

Example subgraphs that satisfy the query.

Jure Leskovec, Stanford University
Queries on Graphs

Applications in biomedicine:

\[ C_? \cdot \exists P : \text{ASSOC}(d_1, P) \land \text{ASSOC}(d_2, P) \land \text{TARGET}(P, C_?) \]

“Predict drugs \( C_? \) that might target proteins that are associated with the given disease nodes \( d_1 \) and \( d_2 \) ”
Why is it Hard?

- Heterogeneity: Lack of schema, or quite large schema (65K for DBpedia)
- Noise and incompleteness
- Uncertainty
- Massive size
- Fast query time

Relational Data (Structured) vs.

Heterogeneous Graph Data (Semi-structured)
Why is it Hard?

**Key challenge:** Big graphs and queries can involve noisy and unobserved data!

Some links might be noisy or unobserved or haven’t occurred yet.

**Problem:** Naïve link prediction and graph template matching are too expensive.
Idea: Subgraph Search

If two entities are close in the query graph, they should also be close in the data graph.

- Subgraph Isomorphism
  - NP-hard → time consuming
  - Too strict to find approximate matches

- Subgraph Similarity Metrics
  - Graph Edit Distance, Max. Common Subgraph, # of Missing Edges
  - Not suitable to preserve closeness among entities
Embedding Logical Queries on Knowledge Graphs (QGE)

Predictive Queries

\[ q = \exists V_1, ..., V_m : e_1 \land e_2 \land ... \land e_n, \]
where \( e_i = \tau(v_j, V_k), V_k \in \{ V?, V_1, ..., V_m \}, v_j \in V, \tau \in \mathcal{R} \)
or \( e_i = \tau(V_j, V_k), V_j, V_k \in \{ V?, V_1, ..., V_m \}, j \neq k, \tau \in \mathcal{R}. \)

- **Conjunctive query language**, where we allow at most one free variable \( V? \):
  - However, we discover unobserved relationships
  - And not just answer queries that exactly satisfy the observed edges

- Every query \( q \) has some unobserved answer set \( \{ q \} \) that we aim to predict
  - We assume that \( \{ q \} \) is not fully observed in our training data
Our Idea: QGE

Use representation learning to map a graph into a Euclidean space and learning to reason in that space.

$$f: u \rightarrow \mathbb{R}^d$$

Feature representation, embedding
Query Graph Embeddings

Embed any conjunctive graph query into a low-dimensional space

- Represent logical query operations as geometric operation
- Generate an embedding $z_q$ for a query $q$ such that answers $v \in \{q\}$ are “close” to the query:

$$\text{score}(q, z_v) = \frac{q \cdot z_v}{||q|| ||z_v||}$$
Overview of QGE Framework

Goal: Answer logical queries

E.g.: “Predict drugs C likely target proteins X associated with diseases $d_1$ and $d_2$”

Idea: Logical operators become spatial operators

Query Graph Embedding

- Represent the query \( q \) using its DAG dependency graph
- Start with the embeddings \( z_{v_1}, \ldots, z_{v_n} \) of its anchor nodes
- Apply geometric operators, \( P \) and \( I \), to these embeddings to obtain an embedding \( z_q \) of the query \( q \)
Benefits of QGE

Scalability and efficiency:
- Any query can be reduced to a couple of matrix operations and a single k-nearest neighbor search

Generality:
- We can answer any query (even those we have never seen before)

Robustness to noise:
- Graph can contain missing and noisy relationships
Query: “Predict drugs C likely target proteins X associated with diseases A and B”

1. Start with embeddings of diseases A and B
Example: Drug Discovery

Query: “Predict drugs $C$ likely target proteins $X$ associated with diseases A and B”

2. Project according to the “associated” relation
Example: Drug Discovery

Query: “Predict drugs C likely target proteins X associated with diseases A and B”

3. Take intersection of the tweet embeddings

Proteins likely to be associated with both diseases A and B
Example: Drug Discovery

Query: “Predict drugs $C$ likely target proteins $X$ associated with diseases $A$ and $B$.”

How do we make this work?
Model Specification

Given: Knowledge graph

Find...

- Node embeddings $z_u$ for node $u$
- Projection operator $P$: $P(i, \tau) = R_\tau \cdot z_i$
  - Applies transition of relation $\tau$ to $i$
- Intersection operator $I$:
  $I(u_1...n) = W_\gamma \cdot \text{AGG}_{j=1...n}(\text{NN}(u_i))$
  - Set intersection in the embedding space

$\tau$... edge type
$\gamma$... node type
$R_\tau$... matrix
$W_\gamma$... matrix
$\Psi$... aggregator
$\text{NN}$... neural net

Model Training

Training strategy

- **Positive examples:** Query graph $q$ and its answers nodes $v^*$
- **Negative examples:** Query graph $q$ and its non-answers nodes $v$
- **Find:** $z_u, P, I$ such that

$$\mathcal{L}(q) = \max(0, 1 - \text{score}(q, z_{v^*}) + \text{score}(q, z_{v_N}))$$
Model Training

Training examples: Queries on the graph

- **Positives**: Path with a known answer
- **“Standard” negatives**: Random nodes of the correct answer type
- **“Hard” negatives**: Correct answers if a logical conjunction is relaxed to a disjunction
Experimental Setup

- **Training set:**
  - Remove 10% of KD edges
  - Sample training queries and (non)answers

- **Test set:**
  - Test queries/answers from the full graph
  - Ensure that the test queries are not directly answerable in the training graph
    - Every test query has at least one deleted edge
  - **Note:** Query template matching would have accuracy of random guessing
Experimental Setup

- Train to $10^6$ queries with 2 edges and $10^6$ queries on 3 edges
- Performance measure:
  - ROC AUC: ranking of answers vs. non-answers
Performance

Performance on different query types:

AUC

Query graph

Bio
Reddit
Performance

EVALUATING ON HARD NEGATIVE EXAMPLES
Conclusion

- **QGE**: Query Graph Embeddings:
  - Embed the query graph
  - Logical operations become spatial operations

- Composability of queries:
  - Explicitly training for composability gives +13% AUC

- Instance vs. multi-hop generalization
Future Work

- Natural future directions include generalizing the space of logical queries
  - How to handle logical negation or disjunction
- Using graph neural networks to incorporate richer feature information on nodes and edges.
References

- Tutorial on Representation Learning on Networks at WWW 2018
  http://snap.stanford.edu/proj/embeddings-www/

- Inductive Representation Learning on Large Graphs.
  W. Hamilton, R. Ying, J. Leskovec. NIPS 2017

- Representation Learning on Graphs: Methods and Applications.
  W. Hamilton, R. Ying, J. Leskovec.

- Embedding Logical Queries on Knowledge Graphs.

- Code:
  - http://snap.stanford.edu/graphsage
  - https://github.com/williamleif/graphqembed