Large-Scale Reasoning over Knowledge Graphs

Hongyu Ren

Knowledge Graphs

- Knowledge Graphs are heterogeneous graphs
  - Multiple types of entities and relations exist
- Facts are represented as triples \((h, r, t)\)
  - (‘Paris’, ‘is_a’, ‘City’)
  - …
- Examples:

  ![Knowledge Graph Diagram]

  WIKIDATA

  Freebase

Hongyu Ren, Stanford University
Goal: Complex logical reasoning on KG

- For example:
  - Where did Canadian Turing Award winners graduate?
  - Who are current presidents of European countries which never held a (soccer) world cup?
  - Predict drugs that might target proteins that are associated with SARS-CoV2?
Examples:

Applications in biomedicine:

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

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

Bio KG
Applications in recommender systems:

Query formula

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

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

Example subgraphs that satisfy the query
Traditional Approach

- Translate the question into a structured query
- Execute the query on the knowledge graph
- Match grounded entities
- “Traverse” the knowledge graph along the relations
Challenges

- How to capture uncertainty
- How to impute missing relations
- How to efficiently answer queries on large graphs

KGs are incomplete and noisy

Traversing them requires time exponential in the query depth

Completing the KG creates additional links, which further slows down query answering

Hongyu Ren, Stanford University
Our Task: Complex Link Prediction

Given arbitrary logical query, directly predict the answer entities:

“Where did all Canadian citizens with Turing Award graduate?”

Logical query:

\[ q = V ? \cdot \exists V : \text{Win}(\text{Turing Award}, V) \land \text{Citizen}(\text{Canada}, V) \land \text{Graduate}(V, V ?) \]

Query graph

Answer entities
Our General Idea

- Map queries into embedding space.
- Learn to reason in that space

[Embedding Logical Queries on Knowledge Graphs, Hamilton, et al., NeurIPS 2018]
[Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings, Ren, et al., ICLR 2020]
Our Framework

**Goal:** Answer logical queries

E.g.: “Predict drugs C likely target proteins X associated with diseases d₁ and d₂”

**Idea:** Logical operators become spatial operators

\[ C? \cdot \exists P : \text{TARGET}(C?, P) \wedge \text{ASSOC}(P, d_2) \wedge \text{ASSOC}(P, d_2) \]

\[ \text{Query DAG} \]

Operations in an embedding space
Our Idea: **Query2Box**

**Idea:**

1) Embed nodes of the graph

2) For every logical operator learn a spatial operator

**So that:**

1) Take an arbitrary logical query. Decompose it into a set of logical operators ($\exists, \land, \lor$)

2) Apply a sequence of spatial operators to embed the query

3) Answers to the query are entities close to the embedding of the query
Idea:

1) Embed nodes of the graph

Key insight:

Represent query as a box. Operations (intersection etc.) are well defined over boxes.

3) Answers to the query are entities close to the embedding of the query
Embedding Queries

Query2Box embedding:

Embed queries with hyper-rectangles (boxes): \( q = (Cen(q), Off(q)) \).

Embedding Space

[Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures, Vilnis, et al., ACL 2018]
Embedding Queries

- Geometric Projection Operator
- Geometric Intersection Operator
Projection Operator

Geometric Projection Operator $\mathcal{P}$

- $\mathcal{P} : \text{Box} \times \text{Relation} \rightarrow \text{Box}$
Projection Operator: Example

Hinton

Edinburgh

Bengio

Graduate

Cambridge

McGill

Turing Award

Win

Pearl

Hinton

Bengio

Turing Award

Win

Pearl

Hinton

Bengio

Graduate

Cambridge

McGill

Hongyu Ren, Stanford University
Intersection Operator

Geometric Intersection Operator $\mathcal{I}$

- $\mathcal{I} : \text{Box} \times \cdots \times \text{Box} \rightarrow \text{Box}$
  - The new center is a weighted average
  - The new offset shrinks

Hongyu Ren, Stanford University
Intersection Operator: Example

Turing Award

Canada

Win

Pearl

Hinton

Bengio

Bieber

Trudeau

Canadian

Citizen

Win

Turing Award

Bengio

Hinton

Pearl

Bieber

Trudeau

Citizen

Hongyu Ren, Stanford University
“Where did Canadian citizens with Turing Award graduate?”

Dependency Graph

Turing Award \rightarrow Win \rightarrow Graduate \rightarrow V?

Canada \rightarrow Citizen \rightarrow V
Example

“Where did Canadian citizens with Turing Award graduate?”

**Dependency Graph**

Turing Award → Win → Citizen → Graduate → V

**Computation Graph**

**Embedding Process**

Each point corresponds to a set of entities

Hongyu Ren, Stanford University
Example

“Where did Canadian citizens with Turing Award graduate?”

Dependency Graph

Computation Graph

Embedding Process

Each point corresponds to a set of entities

Hongyu Ren, Stanford University
“Where did Canadian citizens with Turing Award graduate?”

**Dependency Graph**

- Turing Award → Win → Graduate
- Canada → Citizen → Win

**Computation Graph**

- Turing Award → Projection → Intersection
- Canada → Projection → Intersection

**Embedding Process**

- Turing Award → Win → Pearl, Hinton
- Canada → Citizen → Bieber, Trudeau

Each point corresponds to a set of entities.
“Where did Canadian citizens with Turing Award graduate?”

Dependency Graph

Computation Graph

Embedding Process

Each point corresponds to a set of entities
How to Handle Disjunction

So far we can handle Conjunctive queries Can we learn a geometric disjunction operator?

- Theorem (paraphrased): For a KG with $M$ nodes, we need embedding dimension of $M$ to handle disjunction.
**Disjunctive Normal Form**

- Any query with AND and OR can be transformed into equivalent **Disjunctive Normal Form** (disjunction of conjunctive queries).

\[(a \lor b) \land c = (a \land c) \lor (b \land c)\]
Disjunctions: Solution

Given an arbitrary AND-OR query

1) Transform it into an DNF
2) Answer each conjunctive query
3) Overall answer is the union of conjunctive query answers
Benefits of Query2Box

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
Self-Supervised Training

**Training examples:** A batch of queries on the graph

For each query,

- **One Positive:** Path with a known answer
- **$k$ (~200) Negatives:** Non-answer nodes of the correct answer type
- **Contrastive Objective:** Minimize the distance between embedding of query and positive answer and maximize for negatives.
Large-Scale Training

Scalable Reasoning KG embedding framework (SrKG)

- Asynchronous Design with Optimized Pipeline
  - Multi-thread sampler with pre-fetching mechanism
  - Sparse/Dense embedding R/W

Hongyu Ren, Stanford University
Large-Scale Benchmarks

- We propose three new large KGs to benchmark KG reasoning algorithms. (1365x larger)

![Bar chart showing # Entities and # Edges for Existing and Ours for different KGs: FB15k, FB15k-237, NELL995, FB400k, OGB-Wikikg, Freebase.]

Hongyu Ren, Stanford University
Large KG Performance

SrKG supports 6 different methods

Prior implementation runs out of GPU memory and time limit on the three large KGs

SrKG enables (almost) graph-size agnostic speed and GPU memory usage

Embeddings effectively find missing answers!
SrKG achieves almost linear speedup across models and datasets
Conclusion

- Reasoning in the embedding space
- Embeddings for answering complex queries on KGs
- Large-scale system SrKG for training on massive knowledge graphs
References

- **Code:**
  - https://github.com/snap-stanford/KGReasoning
  - https://github.com/snap-stanford/SrKG
Collaborators

Hongyu Ren, Stanford University