Large-Scale Reasoning over Knowledge Graphs

Hongyu Ren

STANFORD INFOLAB

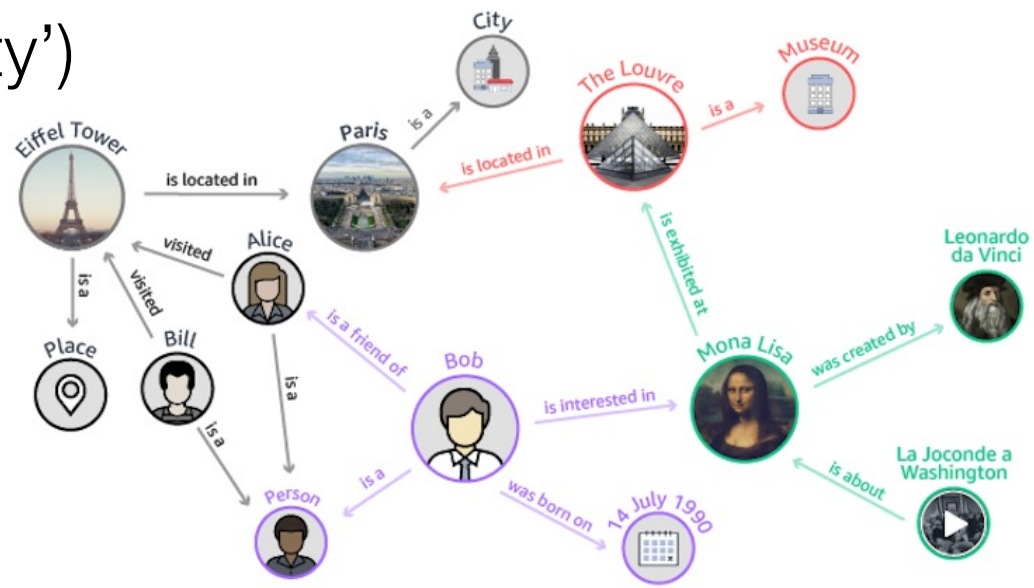
Joint work with W. Hu, H. Dai, B. Dai, X. Chen, M. Yasunaga, H. Sun, D. Zhou, D. Schuurmans, J. Leskovec

Knowledge Graphs

- Knowledge Graphs are heterogenous graphs
  - Multiple types of entities and relations exist
- Facts are represented as triples (h, r, t)
  - ('Mona Lisa', 'created_by', 'Leonardo da Vinci')
  - ('Paris', 'is_a', 'City')
  - …
- Examples:

WIKIDATA

Freebase™



Logical Reasoning

Goal: Complex logical reasoning on KG

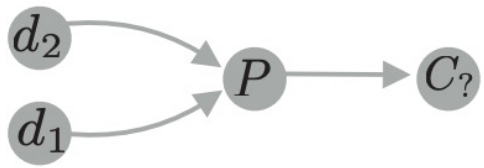
- For example:
  - Where did Canadian Turing Award winners graduate?
  - Who are current presidents of European countries which never held a (soccer) world cup?
  - Predict drugs that might target proteins that are associated with SARS-CoV2?

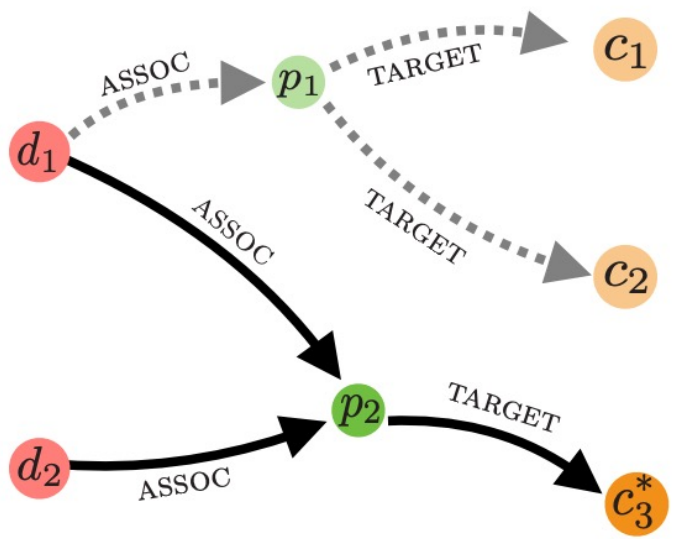
Examples:

Applications in biomedicine:

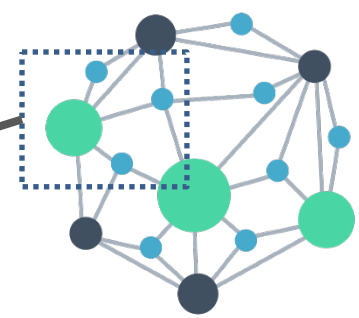
$C_?.\exists P : \text{ASSOC}(d_1, P) \wedge \text{ASSOC}(d_2, P) \wedge \text{TARGET}(P, C_?)$

"Predict drugs $C_?$ that might target proteins that are associated with the given disease nodes d_1 and d_2 "





Bio KG



Examples:

Applications in recommender systems:

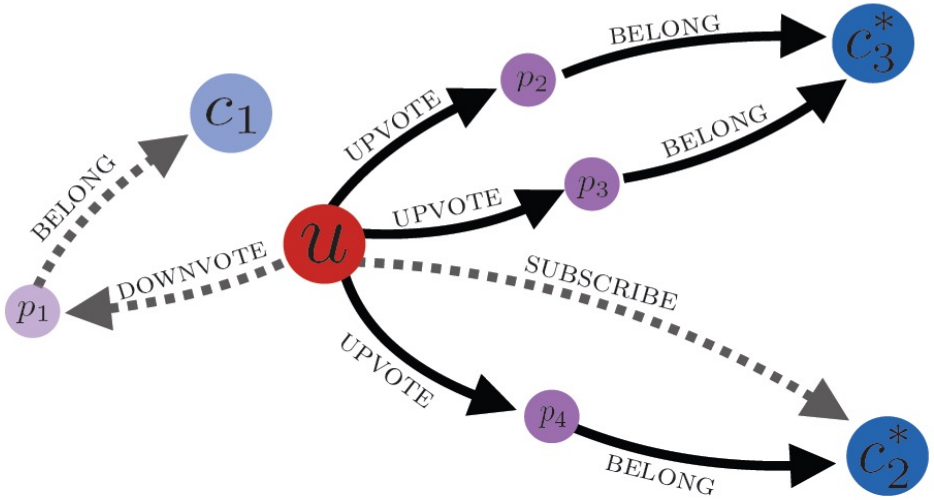
Query formula

$C_?.\exists P : \text{UPVOTE}(u, P) \wedge \text{BELONG}(P, C_?)$

"Predict communities $C_?$ in which user u is likely to upvote a post"

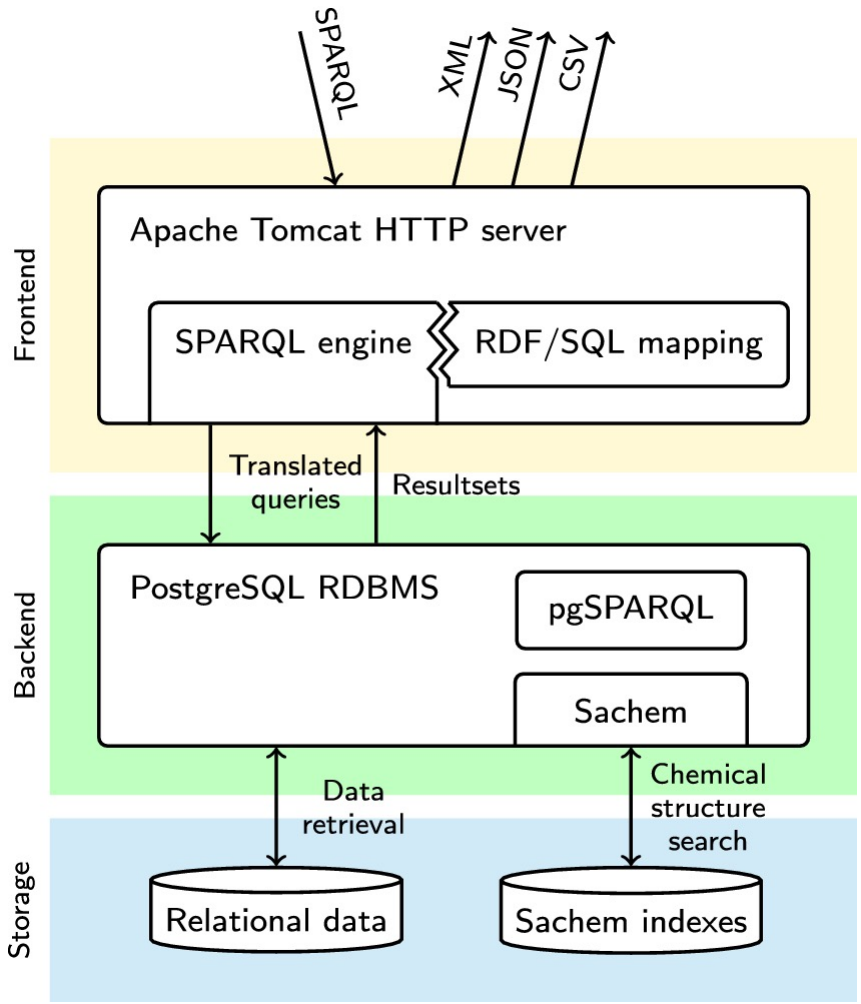
Query DAG

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

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

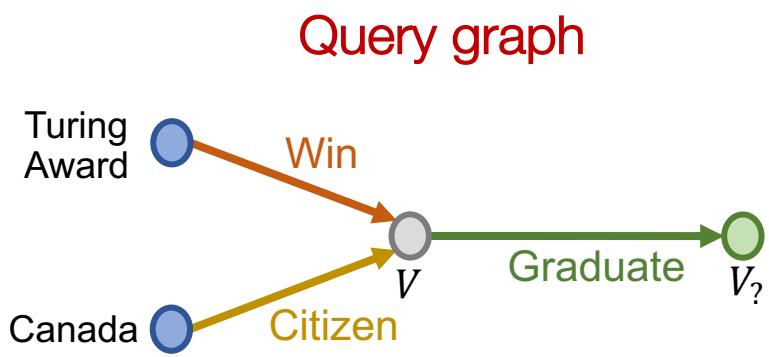
- How to capture uncertainty
- How to impute missing relations
- How to efficiently answer queries on large graphs

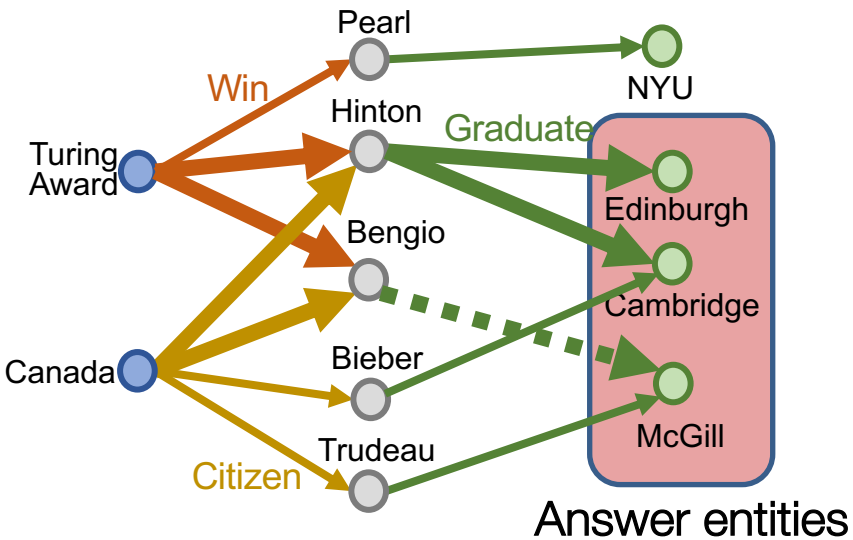
Our Task: Complex Link Prediction

Given arbitrary logical query, **directly** predict the answer entities:

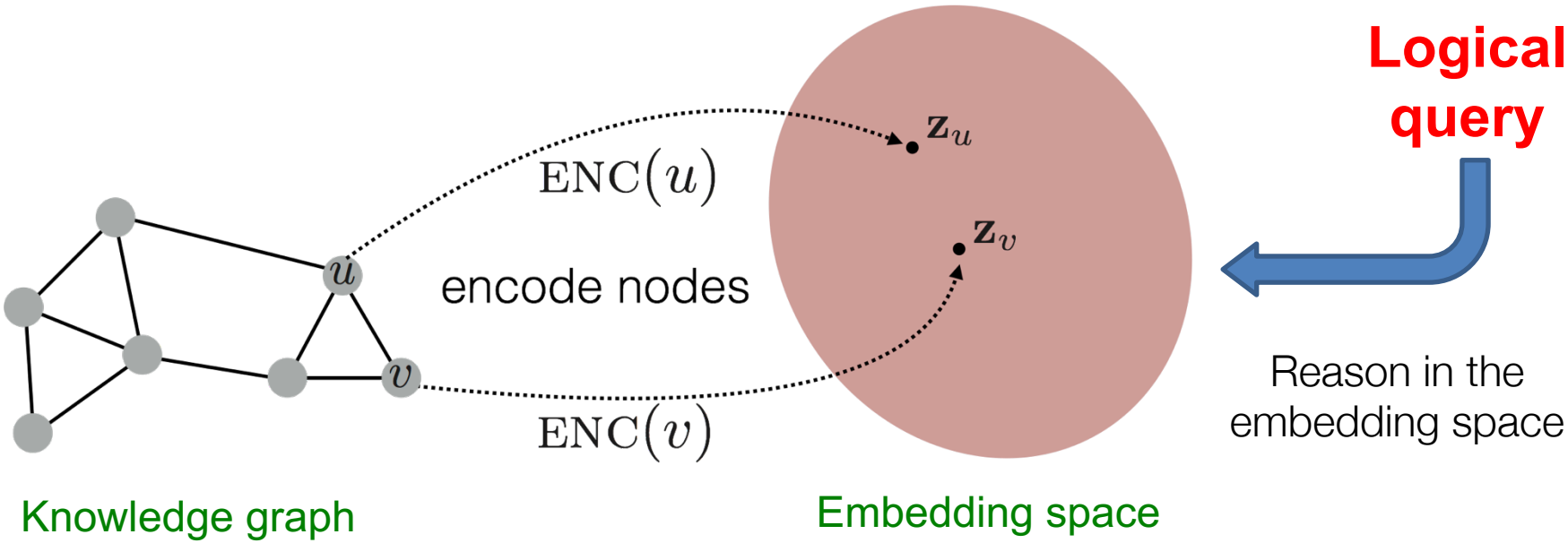
"Where did all Canadian citizens with Turing Award graduate?"

Logical query: $q = V_? \,.\, \exists\, V : Win(TuringAward, V) \wedge Citizen(Canada, V) \wedge Graduate(V, V_?)$





Our General Idea



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

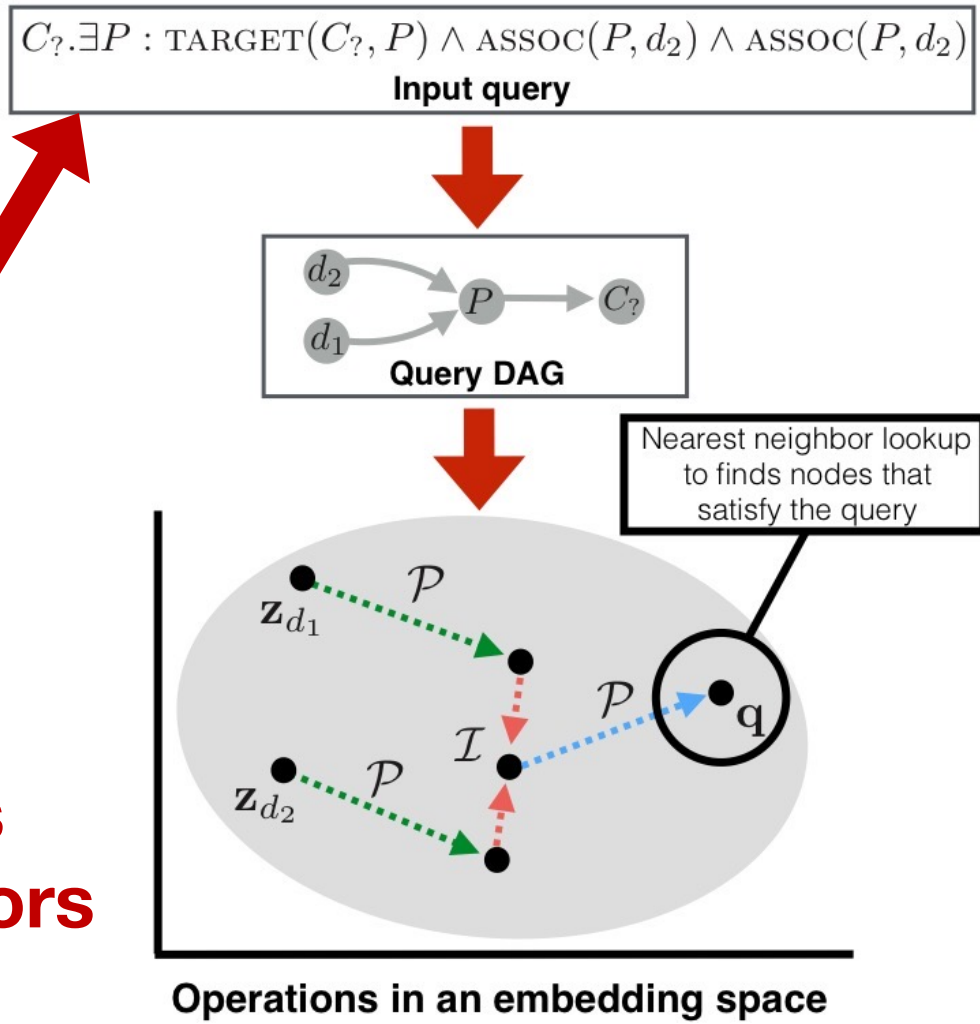
[Embedding Logical Queries on Knowledge Graphs. Hamilton, et al., NeurIPS 2018]
[Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. Ren, et al., ICLR 2020]

Our Framework

Goal: Answer logical queries

E.g.: "**Predict drugs C** likely *target* proteins X *associated* with diseases d_1 and d_2"

Idea: Logical operators become spatial operators



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 (∃,∧,∨)
- **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

Our Idea: **Query2Box**

Idea:

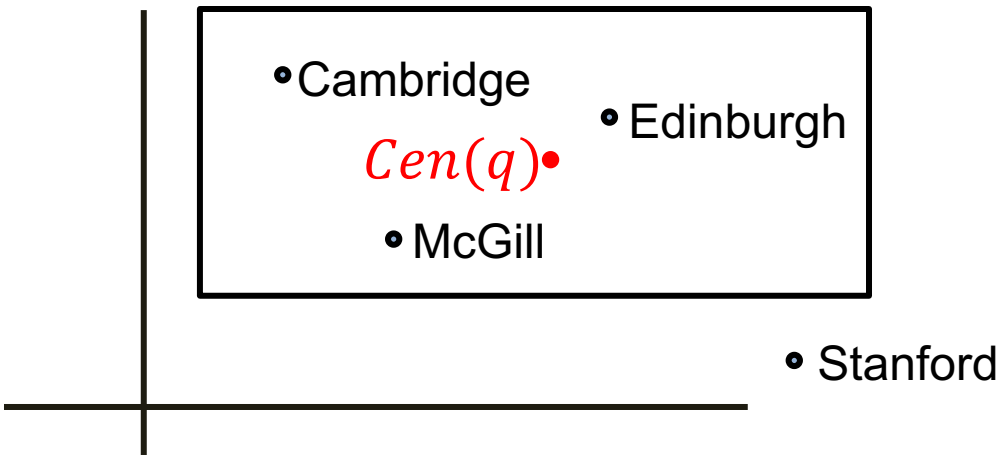
- **1)** Embed nodes of the graph
-
-
-
-
- **3)** Answers to the query are entities close to the embedding of the query

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

Embedding Queries

Query2Box embedding:

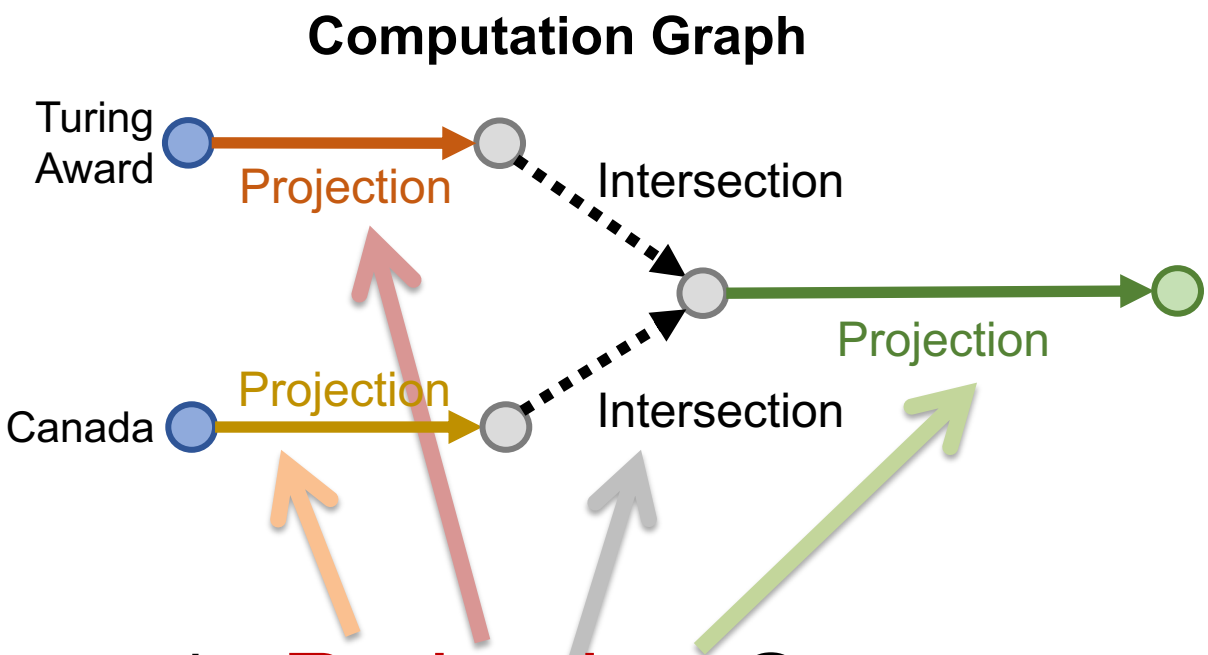
Embed queries with hyper-rectangles (boxes): $\mathbf{q} = (Cen(q), Off(q))$.



Embedding Space

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

Embedding Queries

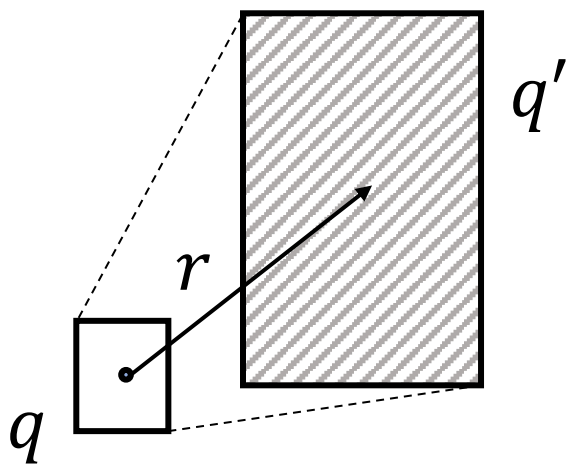


- Geometric Projection Operator
- Geometric Intersection Operator

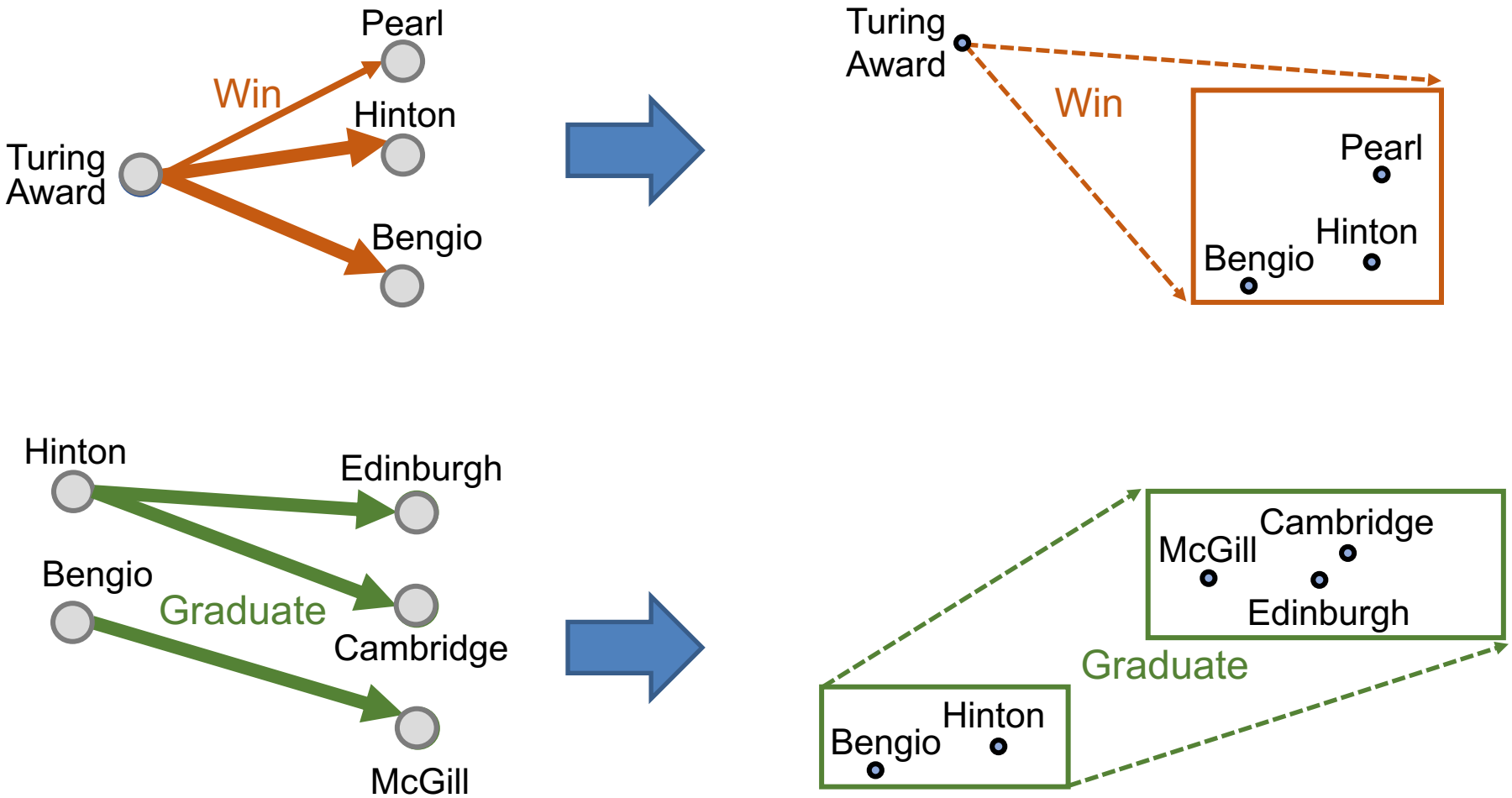
Projection Operator

Geometric Projection Operator $\mathcal{P}$

- $\mathcal{P}$: Box × Relation → Box



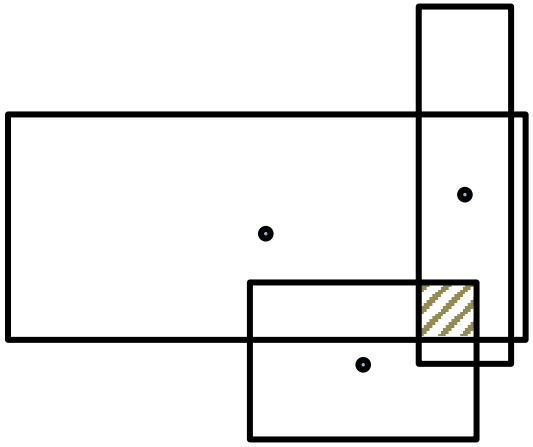
Projection Operator: Example



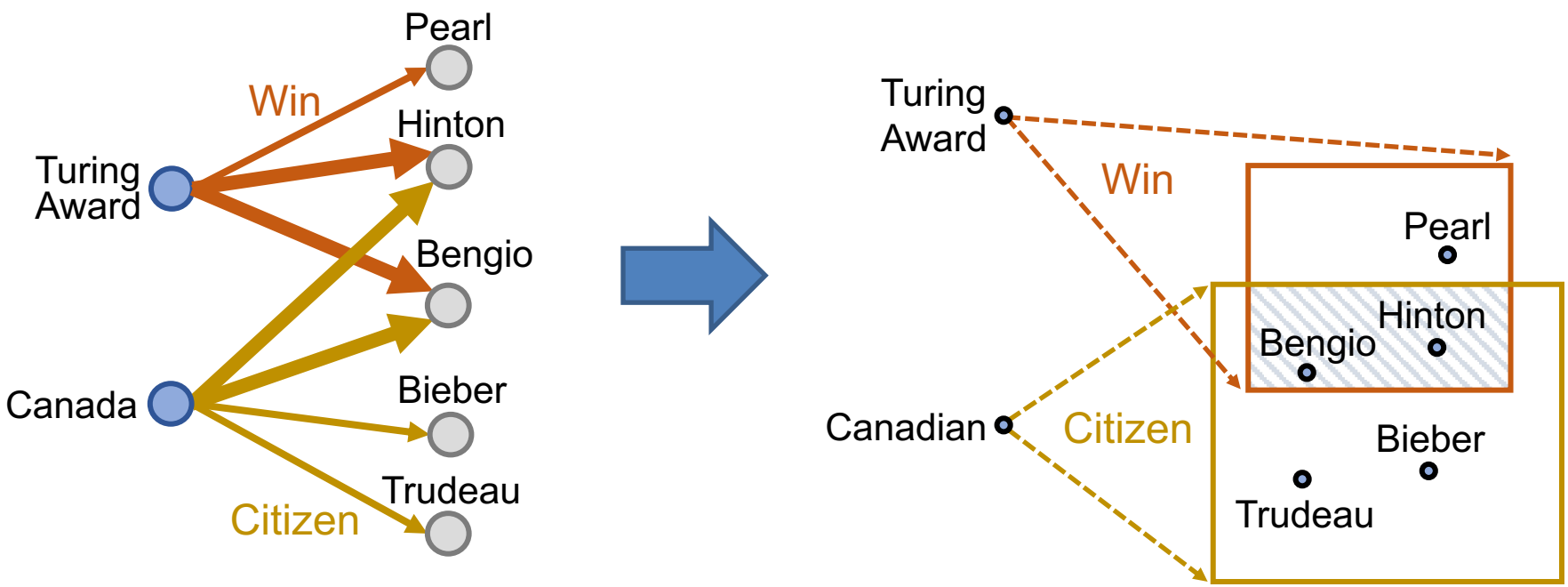
Intersection Operator

Geometric Intersection Operator $\mathcal{I}$

- $\mathcal{I}$: Box × ⋯× Box → Box
  - The new center is a weighted average
  - The new offset shrinks



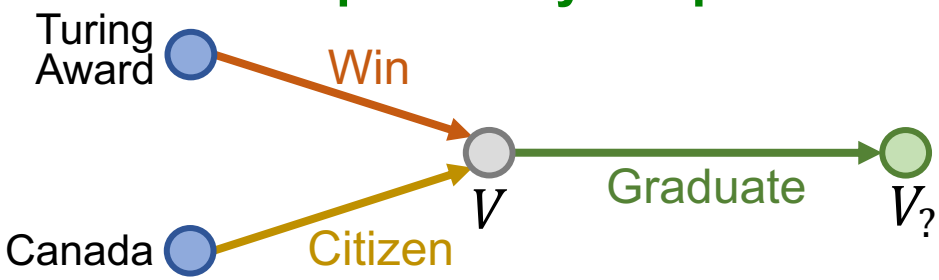
Intersection Operator: Example



Example

"Where did Canadian citizens with Turing Award graduate?"

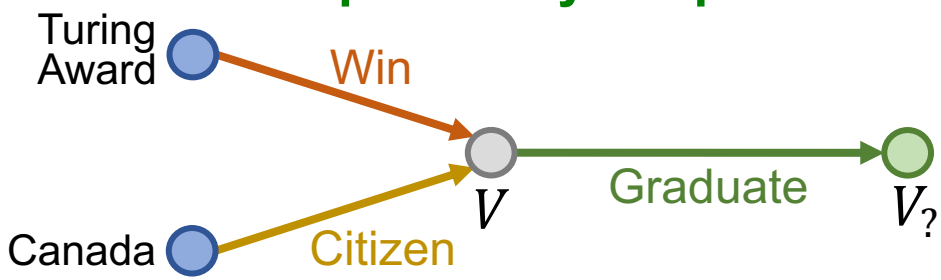
Dependency Graph



Example

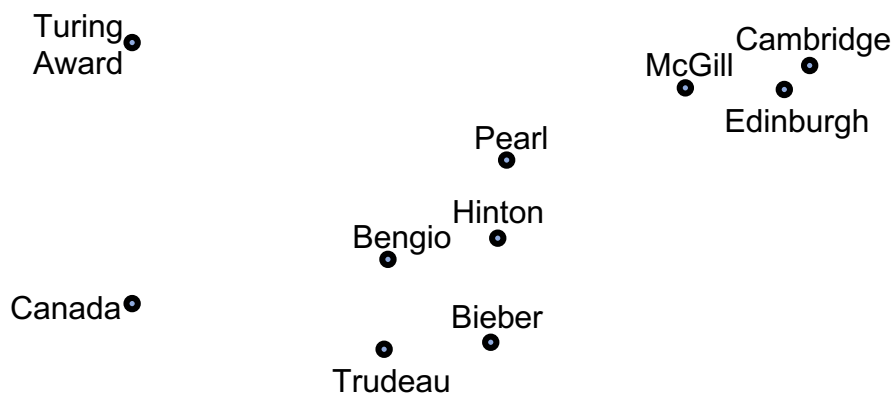
"Where did Canadian citizens with Turing Award graduate?"

Dependency Graph



Computation Graph

Embedding Process

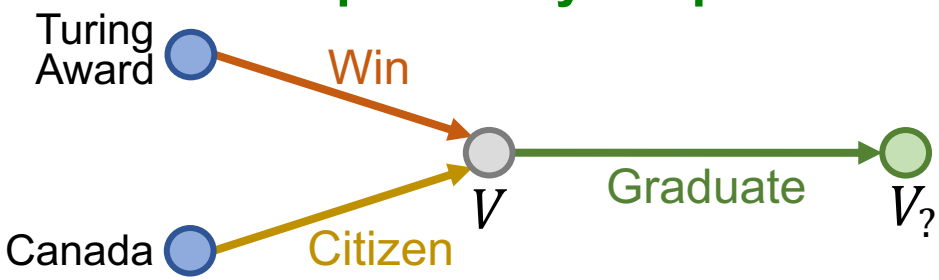


Each point corresponds to a set of entities

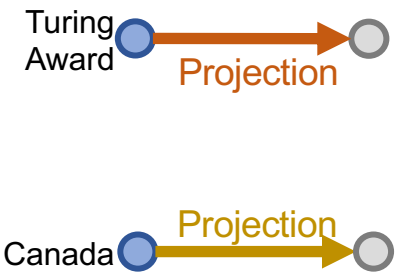
Example

"Where did Canadian citizens with Turing Award graduate?"

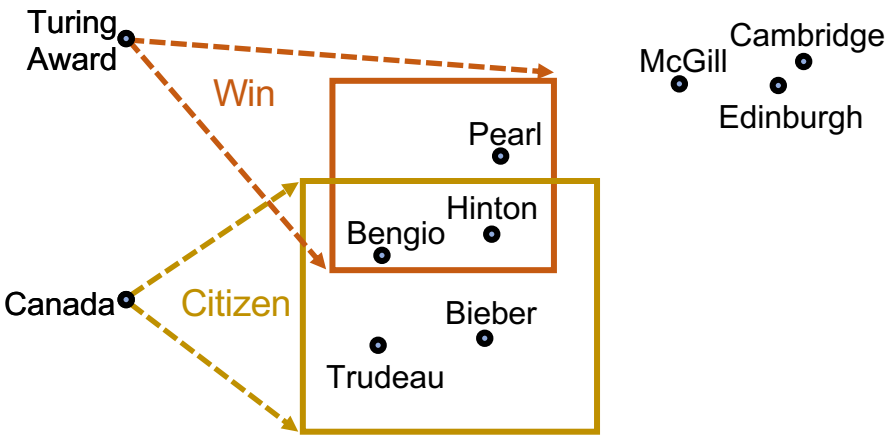
Dependency Graph



Computation Graph



Embedding Process

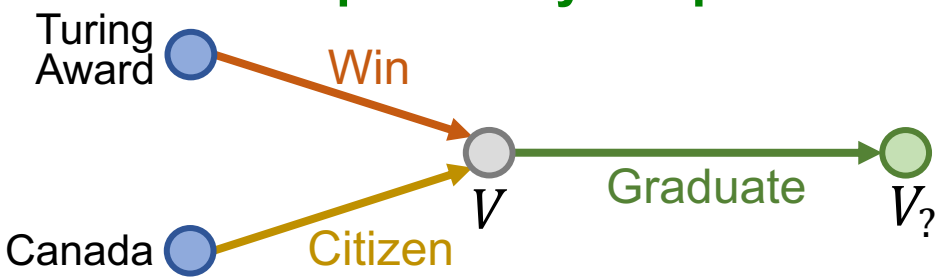


Each point corresponds to a set of entities

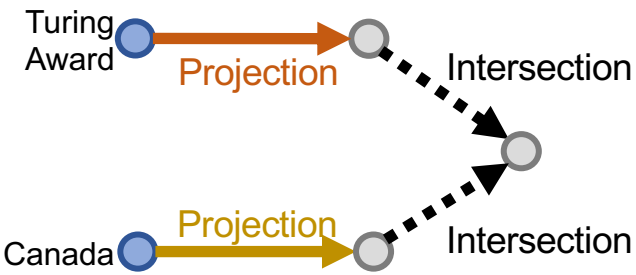
Example

"Where did Canadian citizens with Turing Award graduate?"

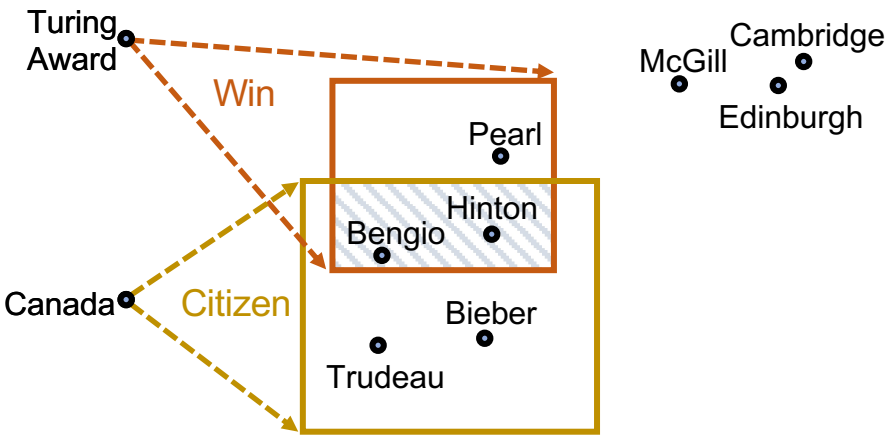
Dependency Graph



Computation Graph



Embedding Process

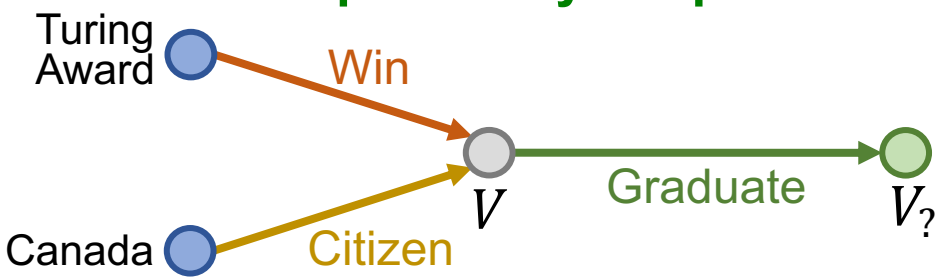


Each point corresponds to a set of entities

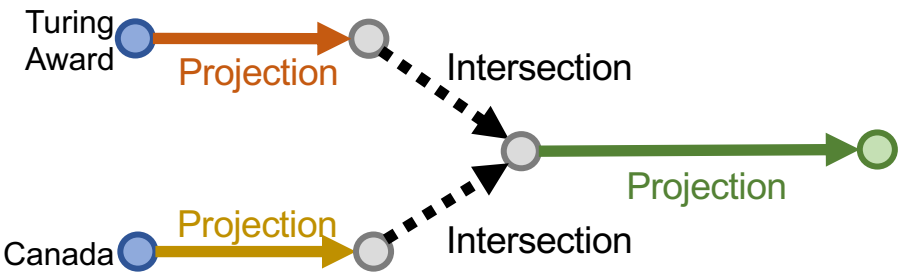
Example

"Where did Canadian citizens with Turing Award graduate?"

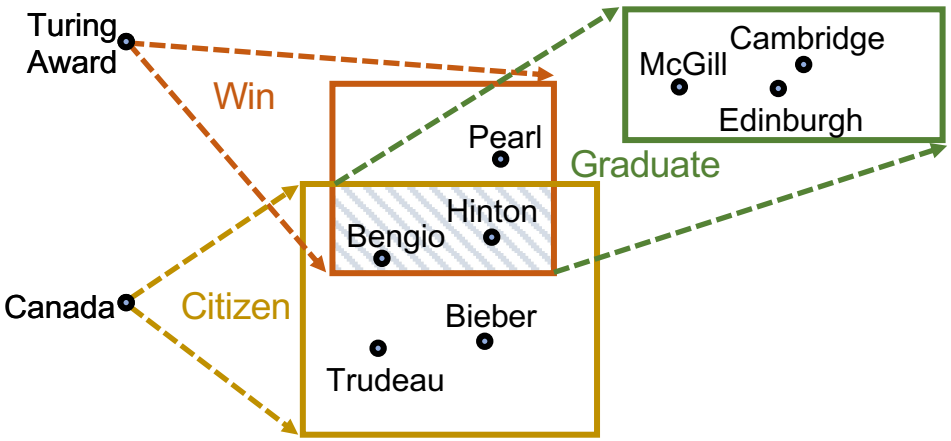
Dependency Graph



Computation Graph



Embedding Process



Each point corresponds to a set of entities

How to Handle Disjunction

So far we can handle Conjunctive queries

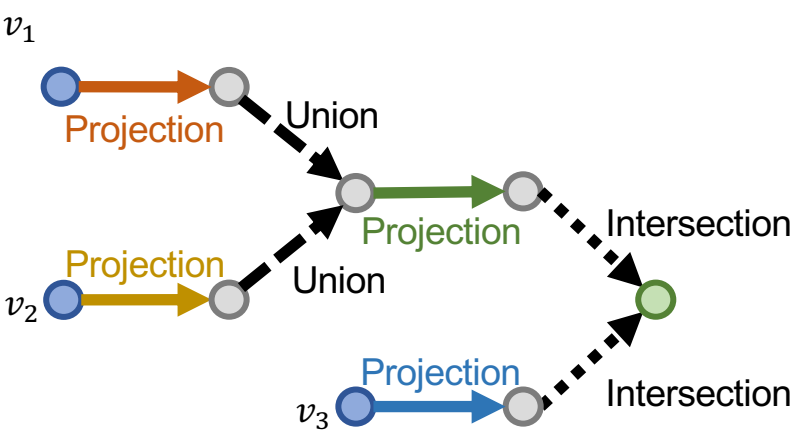
Can we learn a geometric disjunction operator?

- Theorem (paraphrased): For a KG with M nodes, we need embedding dimension of M to handle disjunction

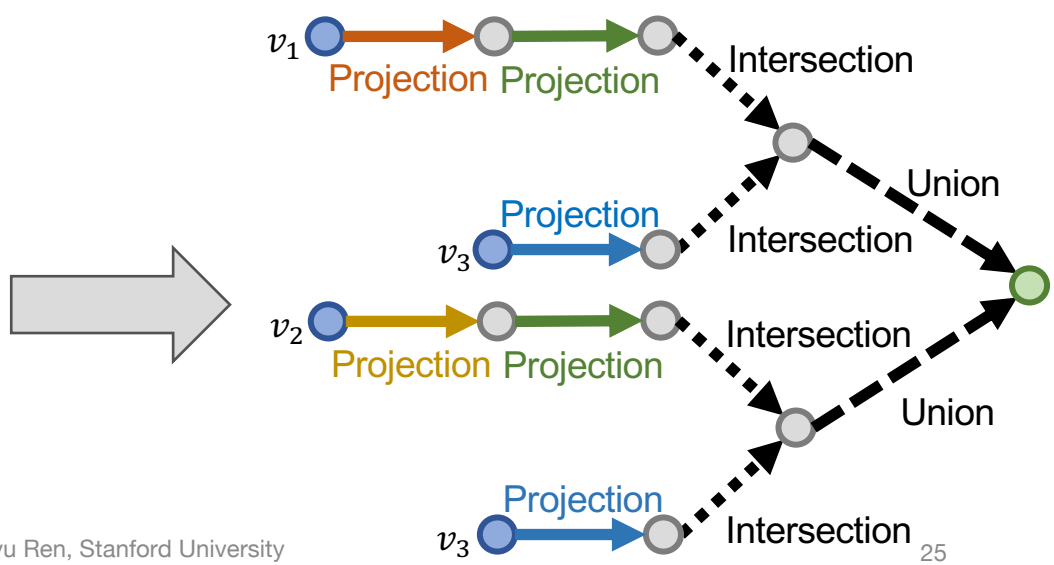
Disjunctive Normal Form

- Any query with AND and OR can be transformed into equivalent Disjunctive Normal Form (disjunction of conjunctive queries).
- $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$

Original Computation Graph



Converted Computation Graph



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:

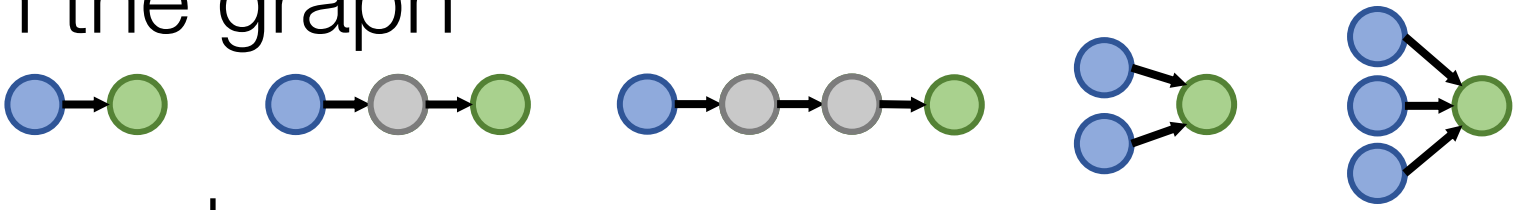
- We can answer any query (even those we have never seen before)

Robustness to noise:

- Graph can contain missing and noisy relationships

Self-Supervised Training

Training examples: A batch of queries on the graph



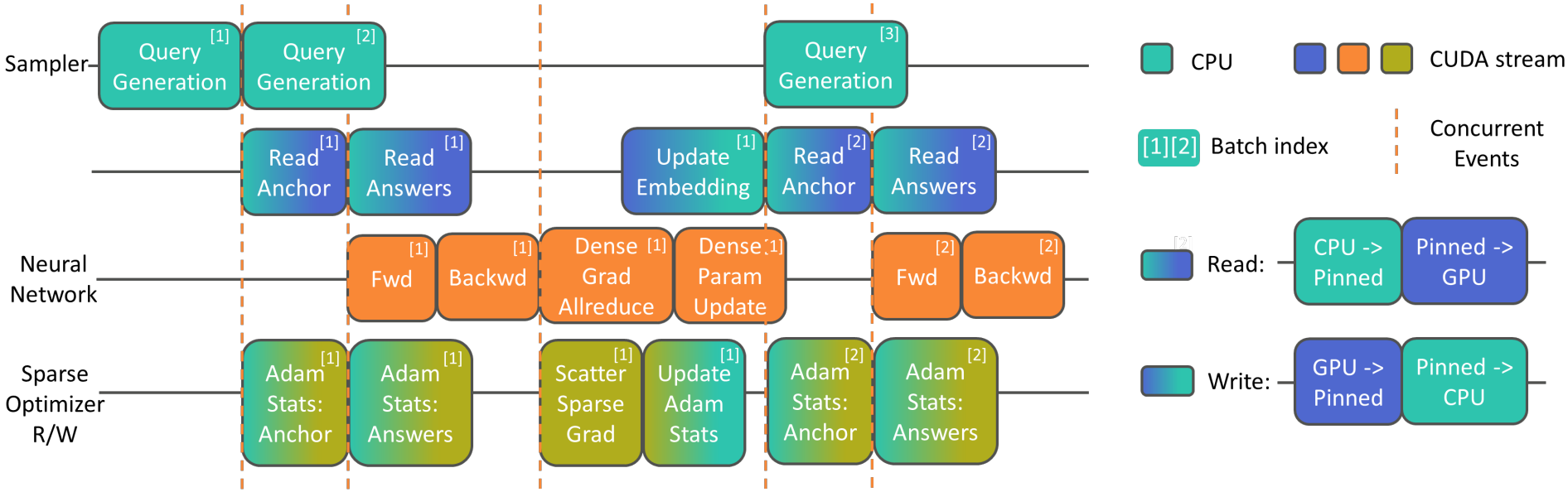
For each query,

- One Positive: Path with a known answer
- $\boldsymbol{k}$ (~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

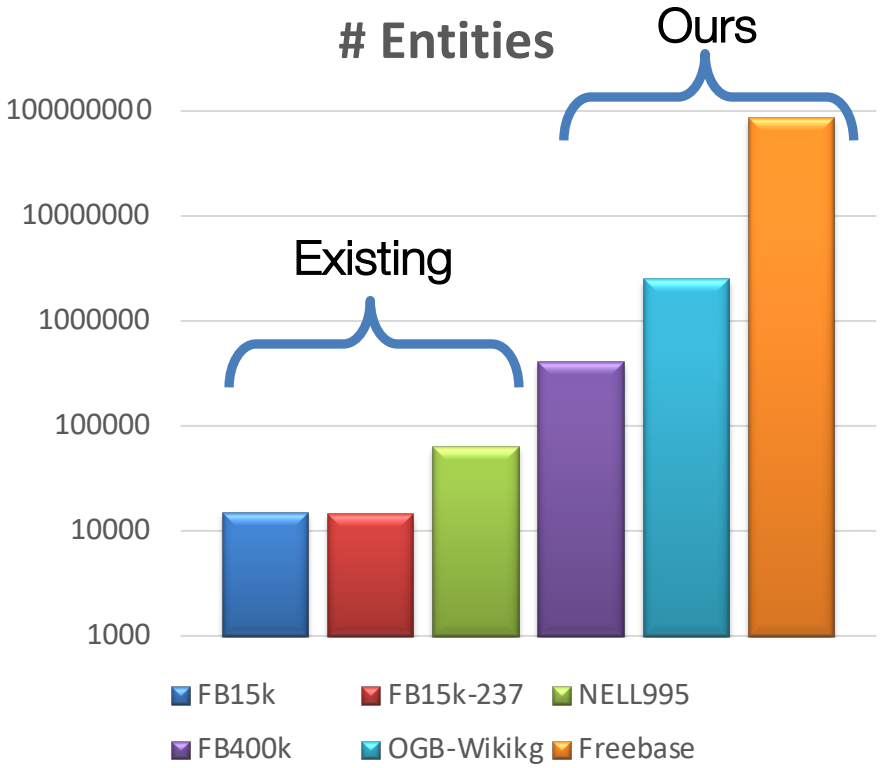
Scalable Reasoning KG embedding framework (SrKG)

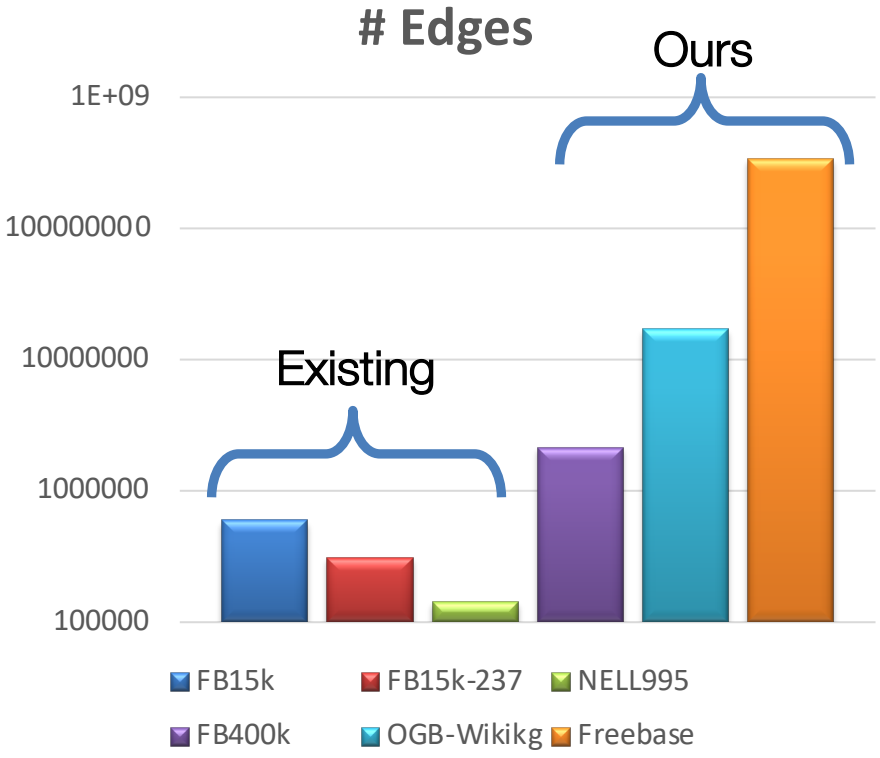
- Asynchronous Design with Optimized Pipeline
  - Multi-thread sampler with pre-fetching mechanism
  - Sparse/Dense embedding R/W



Large-Scale Benchmarks

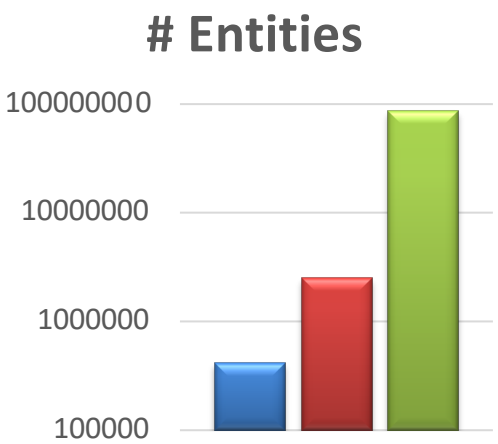
- We propose three new large KGs to benchmark KG reasoning algorithms. (1365x larger)

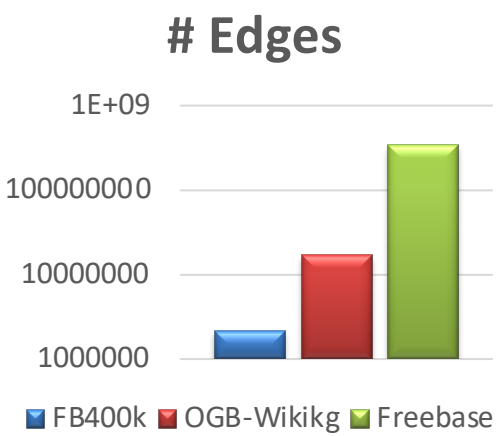




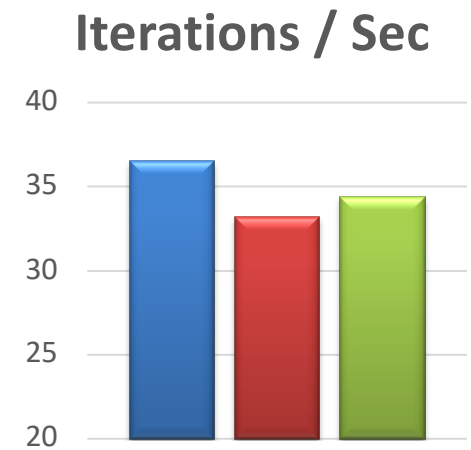
Large KG Performance

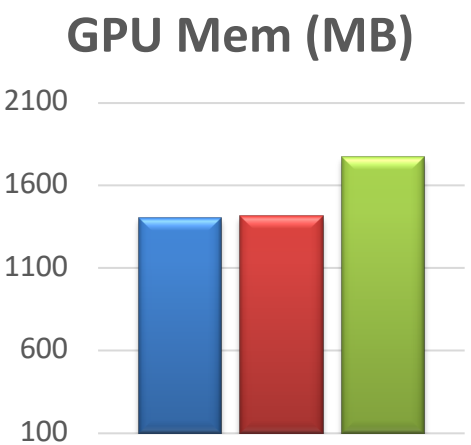
Dataset \ Model	GQE [8]	Q2B [9]	BetaE [10]	RotatE-m	Distmult-m	Complex-m	Vanilla [10]
FB400k	36.02	**51.74**	50.50	43.00	38.78	41.62	TLE
ogbl-wikikg2	32.91	41.88	**44.42**	34.07	28.41	29.52	TLE
Freebase	80.71	**85.67**	84.33	80.55	70.24	70.59	TLE



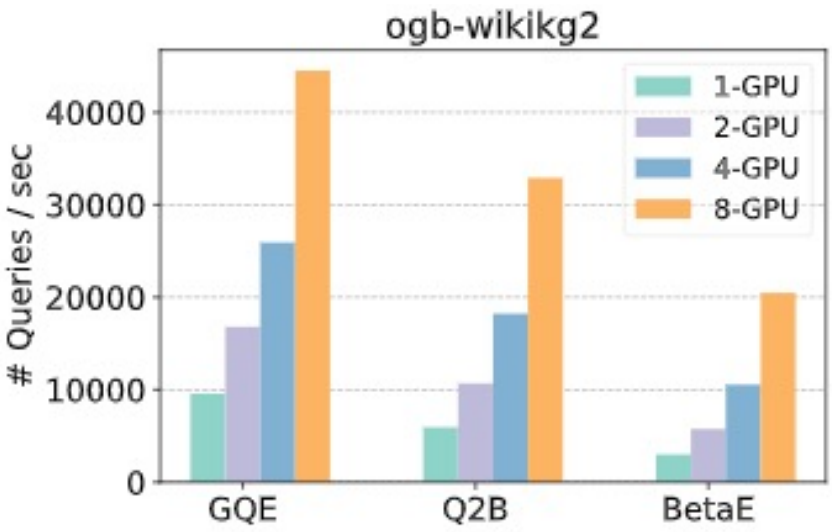


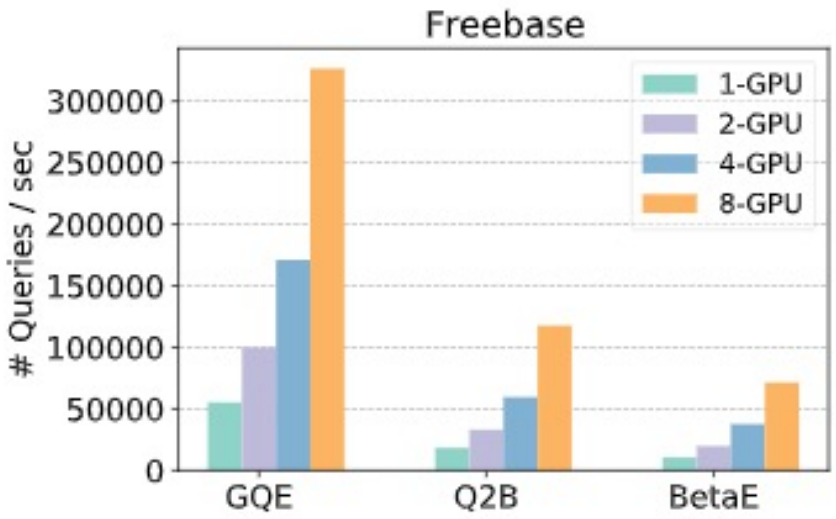
- SrKG supports 6 different methods
- Prior implementation runs out of GPU memory and time limit on the three large KGs
- SrKG enables (almost) graph-size agnostic speed and GPU memory usage
- Embeddings effectively find missing answers!





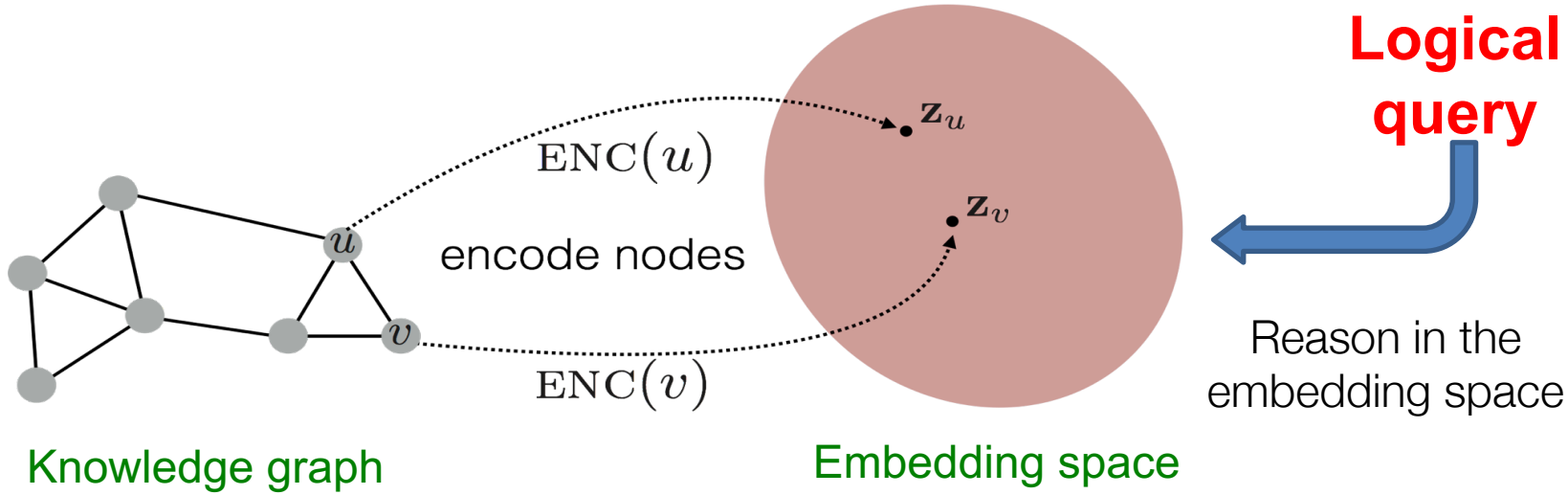
Multi-GPU Speedup





- 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

- Query2box: Reasoning Over Knowledge Graphs In Vector Space Using Box Embeddings. H. Ren, W. Hu, J. Leskovec. ICLR, 2020.
- Scaling up Logical Query Embeddings on Knowledge Graphs. H. Ren, H. Dai, B. Dai, X. Chen, D. Zhou, J. Leskovec, D. Schuurmans. ICML SSL workshop 2021.
- Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. H. Ren, J. Leskovec. NeurIPS 2020.
- LEGO: Latent Execution-Guided Reasoning for Multi-Hop Question Answering on Knowledge Graphs. H. Ren, H. Dai, B. Dai, X. Chen, M. Yasunaga, H. Sun, D. Schuurmans, J. Leskovec, D. Zhou. ICML SSL workshop 2021.

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

Collaborators