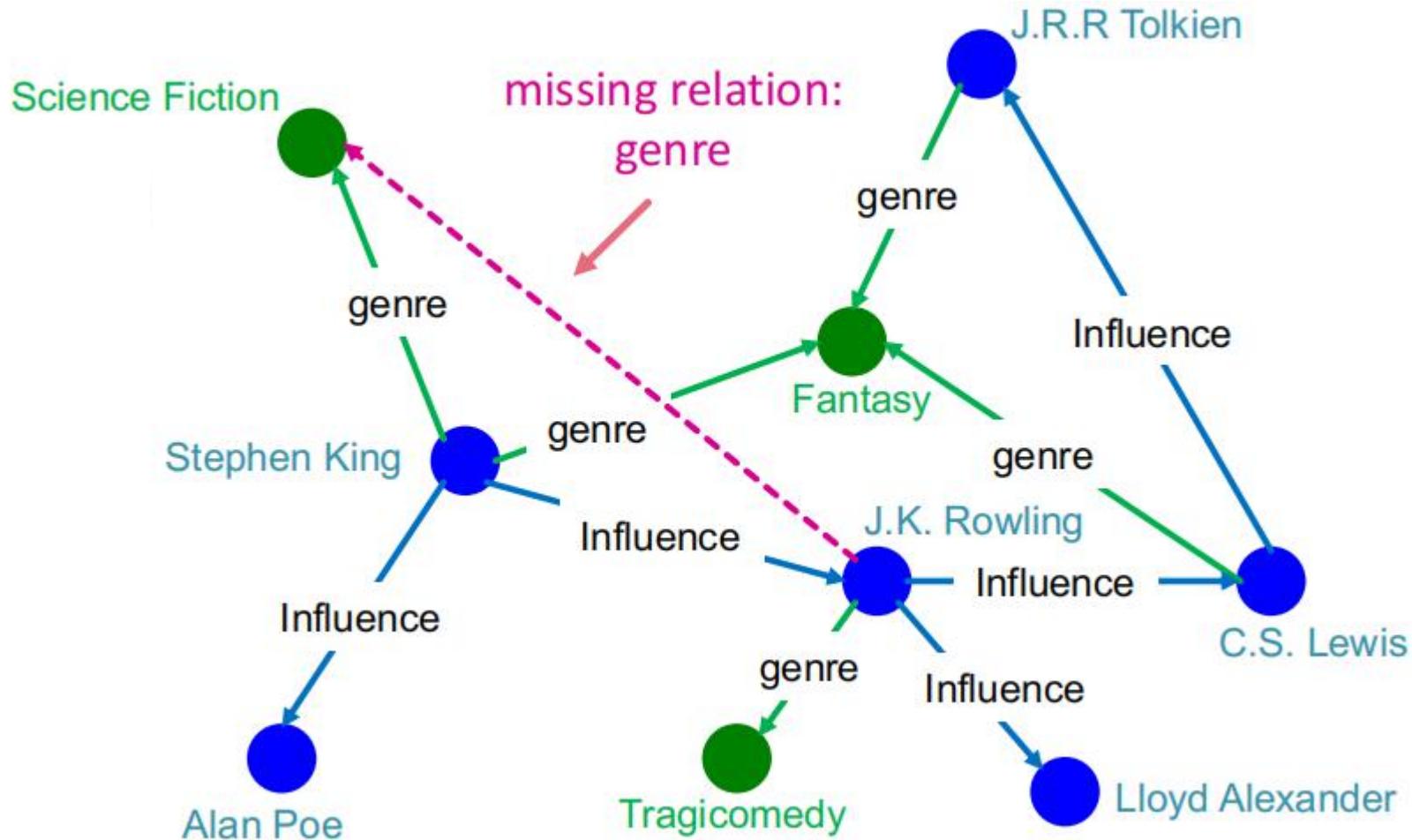


Reasoning over Knowledge Graphs

— — Embed with Box Embedding

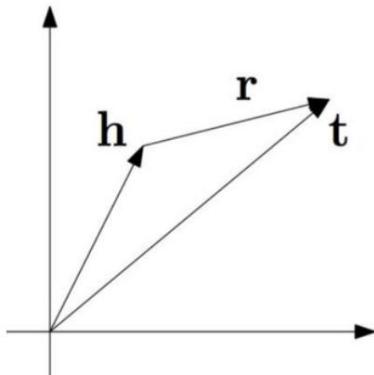
2021/5/13 朱静丹

Given an enormous KG, can we complete the KG?



TransE

TransE: “Translating Embeddings for Modeling Multi-relational Data” (NIPS2013)



$$\mathcal{L} = \sum_{(h, \ell, t) \in S} \sum_{(h', \ell, t') \in S_{(h, \ell, t)}} [\gamma + d(\mathbf{h} + \ell, \mathbf{t}) - d(\mathbf{h}' + \ell, \mathbf{t}')]_+$$

TransE模型属于翻译模型：

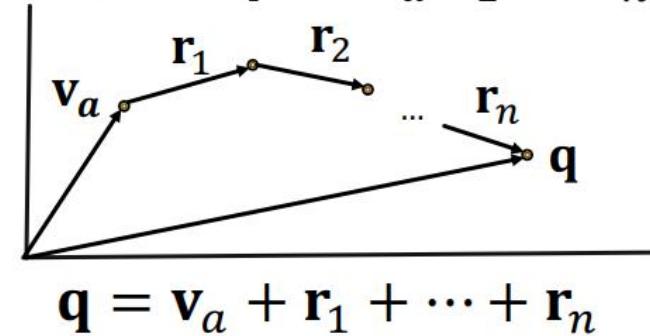
直观上，将每个三元组实例 (head, relation, tail) 中的关系relation看做从实体head到实体tail的翻译，通过不断调整h、r和t (head、relation和tail的向量)，使 $(\mathbf{h} + \mathbf{r})$ 尽可能与 \mathbf{t} 相等，即 $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$

Generalize TransE to multi-hop reasoning.

Traversing knowledge graphs in vector space(EMNLP 2015)

- Recap: **TransE**: Translate **h** to **t** using **r** with score function $f_r(h, t) = -\|h + r - t\|$.
- Another way to interpret this is that:
 - Query embedding**: $q = h + r$
 - Goal: **query embedding** q is **close** to the **answer embedding** t

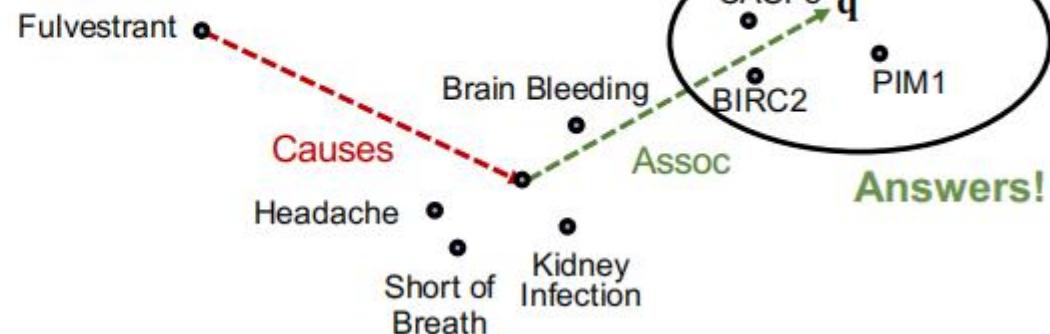
Given a path query $q = (v_a, r_1, \dots, r_n)$,



Query Plan



Embedding Process

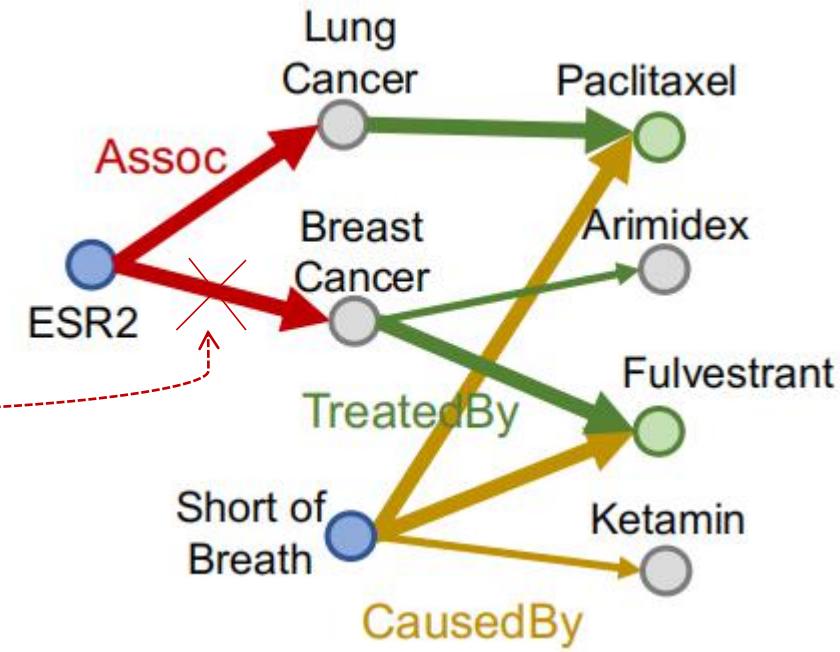


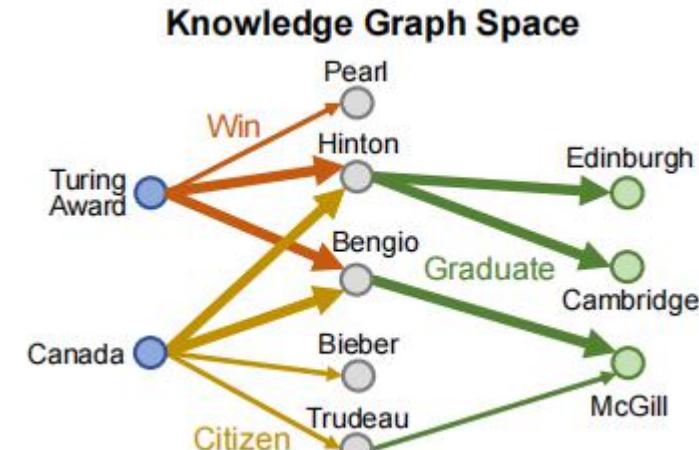
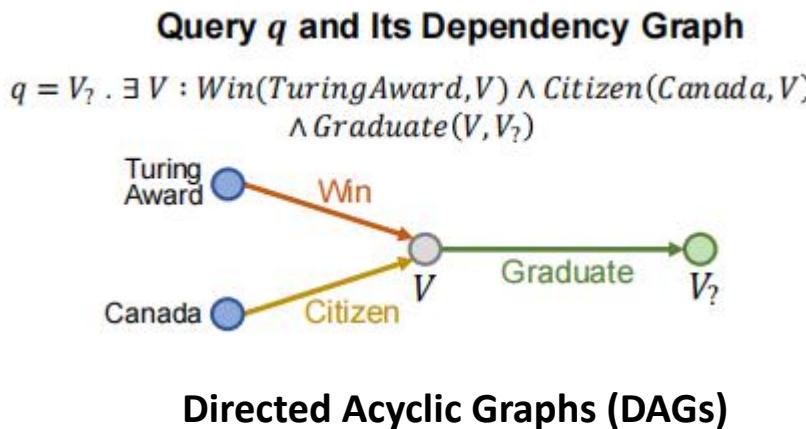
"What are drugs that cause **Short of Breath** and treat diseases associated with protein **ESR2**?"

If this link is missing, then we can not find the answer **Fulvestrant**

- Short of Breath
- Kidney Infection
- Headache

Embedding Space





- (1) Computational complexity of subgraph matching is **exponential** in the query size, and thus cannot scale to modern KGs;
- (2) Subgraph matching is very sensitive as it cannot correctly answer queries with **missing relations**.

KNOWLEDGE GRAPHS AND CONJUNCTIVE QUERIES

Conjunctive queries are a subclass of the first-order logical queries that use existential (\exists) and conjunction (\wedge) operations. They are formally defined as follows.

$$q[V?] = V? . \exists V_1, \dots, V_k : e_1 \wedge e_2 \wedge \dots \wedge e_n, \quad (1)$$

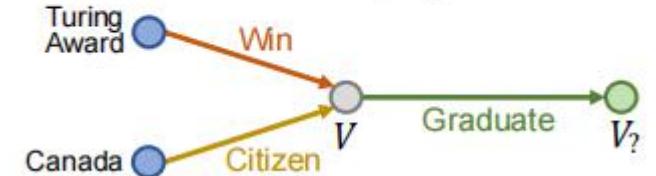
where $e_i = r(v_a, V), V \in \{V?, V_1, \dots, V_k\}, v_a \in \mathcal{V}, r \in \mathcal{R}$,

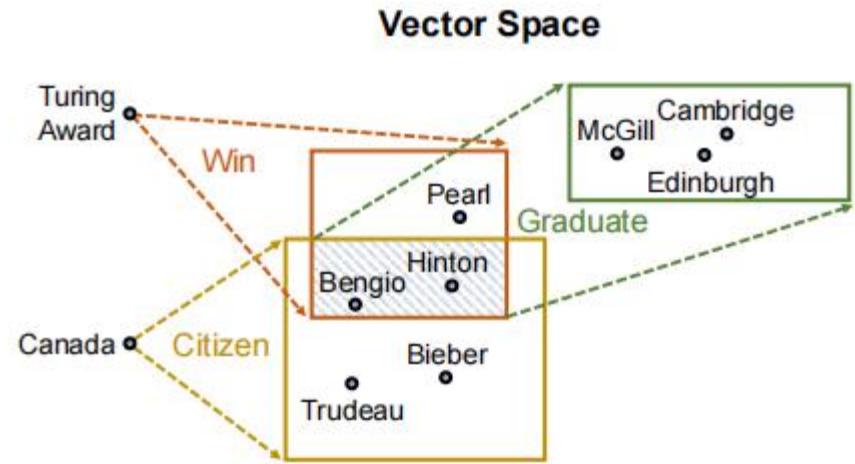
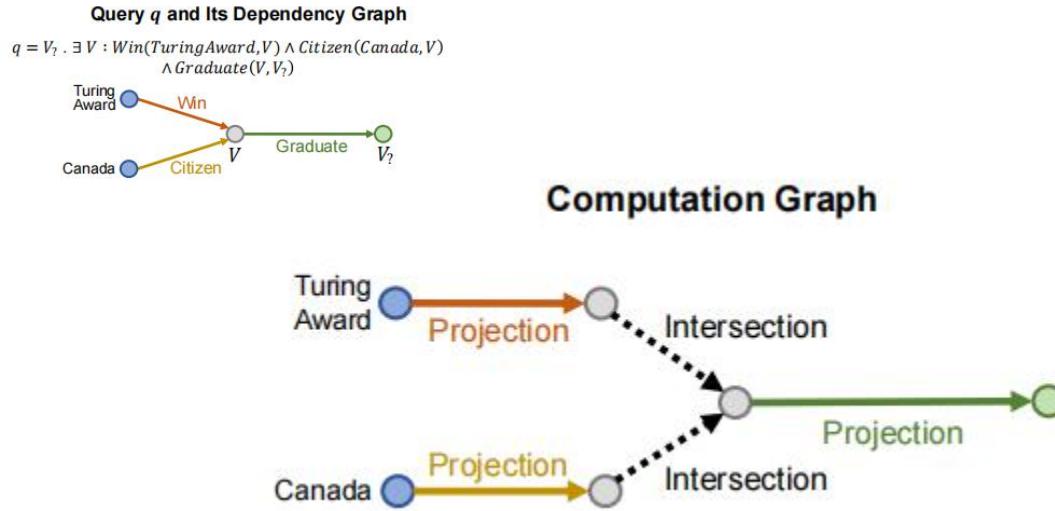
or $e_i = r(V, V'), V, V' \in \{V?, V_1, \dots, V_k\}, V \neq V', r \in \mathcal{R}$,

where v_a represents non-variable anchor entity, V_1, \dots, V_k are existentially quantified bound variables, $V?$ is the target variable. The goal of answering the logical query q is to find a set of entities $\llbracket q \rrbracket \subseteq \mathcal{V}$ such that $v \in \llbracket q \rrbracket$ iff $q[v] = \text{True}$. We call $\llbracket q \rrbracket$ the *denotation set* (i.e., answer set) of query q .

Query q and Its Dependency Graph

$$q = V? . \exists V : \text{Win}(\text{TuringAward}, V) \wedge \text{Citizen}(\text{Canada}, V) \wedge \text{Graduate}(V, V?)$$



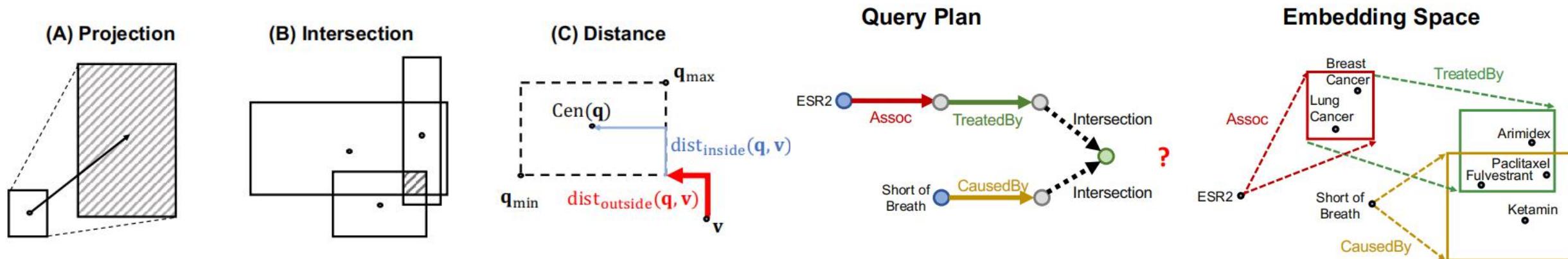


- **Projection:** Given a set of entities $S \subseteq \mathcal{V}$, and relation $r \in \mathcal{R}$, this operator obtains $\cup_{v \in S} A_r(v)$, where $A_r(v) \equiv \{v' \in \mathcal{V} : r(v, v') = \text{True}\}$.
- **Intersection:** Given a set of entity sets $\{S_1, S_2, \dots, S_n\}$, this operator obtains $\cap_{i=1}^n S_i$.

$$\text{Box}_p \equiv \{\mathbf{v} \in \mathbb{R}^d : \text{Cen}(p) - \text{Off}(p) \preceq \mathbf{v} \preceq \text{Cen}(p) + \text{Off}(p)\}$$

Initial boxes for source nodes.

Formally, we set the initial box embedding as $(\mathbf{v}, \mathbf{0})$, where $\mathbf{v} \in \mathbb{R}^d$ is the anchor entity vector and $\mathbf{0}$ is a d -dimensional all-zero vector.



Geometric projection operator.

We associate each relation $r \in \mathcal{R}$ with relation embedding $\mathbf{r} = (Cen(\mathbf{r}), Off(\mathbf{r})) \in \mathbb{R}^{2d}$ with $Off(\mathbf{r}) \succeq 0$.

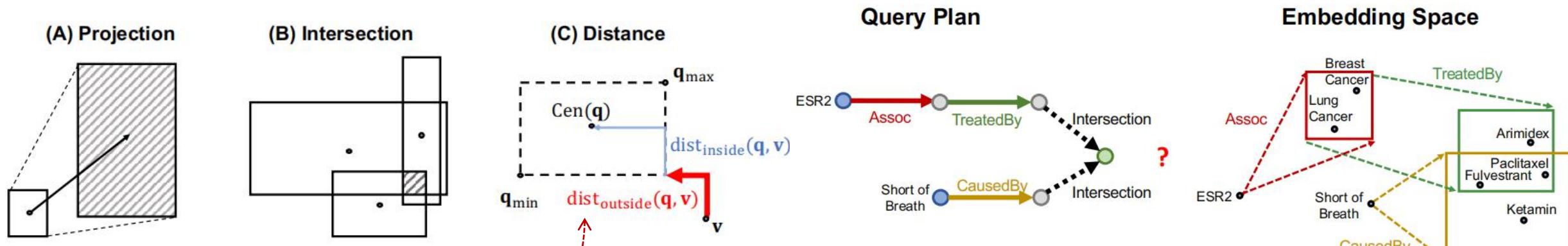
Given an input box embedding \mathbf{p} , we model the projection by $\mathbf{p} + \mathbf{r}$, where we sum the centers and sum the offsets.

“self-attention” score

$$Cen(\mathbf{p}_{\text{inter}}) = \sum_i \mathbf{a}_i \odot Cen(\mathbf{p}_i), \quad \boxed{\mathbf{a}_i = \frac{\exp(MLP(\mathbf{p}_i))}{\sum_j \exp(MLP(\mathbf{p}_j))}},$$

$$Off(\mathbf{p}_{\text{inter}}) = \text{Min}(\{Off(\mathbf{p}_1), \dots, Off(\mathbf{p}_n)\}) \odot \boxed{\sigma(\text{DeepSets}(\{\mathbf{p}_1, \dots, \mathbf{p}_n\}))},$$

sigmoid function



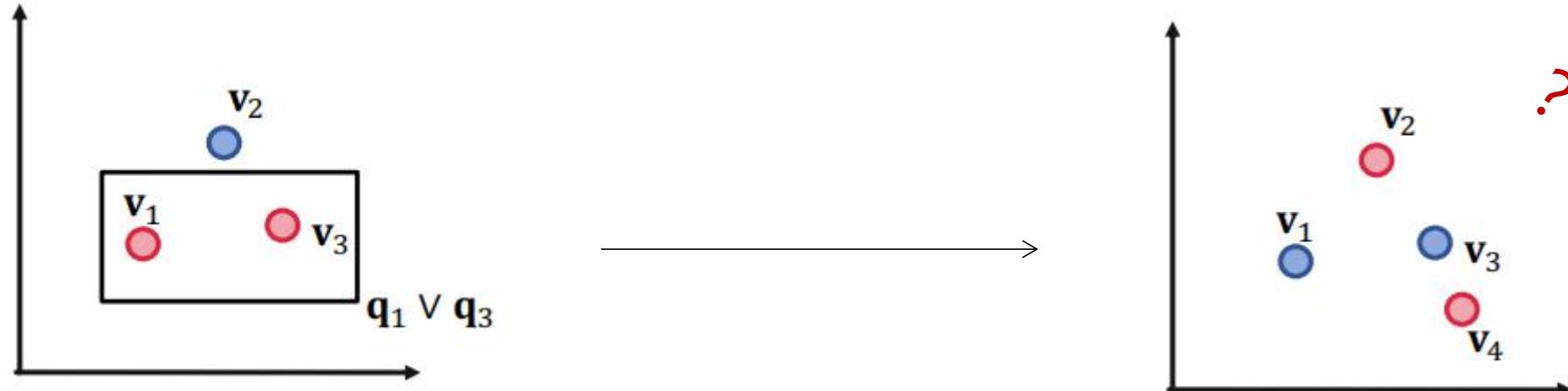
Entity-to-box distance.

$$\text{dist}_{\text{box}}(v; q) = \text{dist}_{\text{outside}}(v; q) + \alpha \cdot \text{dist}_{\text{inside}}(v; q)$$

where $q_{\max} = \text{Cen}(q) + \text{Off}(q) \in \mathbb{R}^d$, $q_{\min} = \text{Cen}(q) - \text{Off}(q) \in \mathbb{R}^d$ and $0 < \alpha < 1$ is a fixed scalar, and

$$\begin{aligned}\text{dist}_{\text{outside}}(v; q) &= \|\text{Max}(v - q_{\max}, \mathbf{0}) + \text{Max}(q_{\min} - v, \mathbf{0})\|_1, \\ \text{dist}_{\text{inside}}(v; q) &= \|\text{Cen}(q) - \text{Min}(\text{Max}(q_{\max}, \mathbf{0}), \text{Max}(q_{\min}, v))\|_1.\end{aligned}$$

“close enough” to the query center



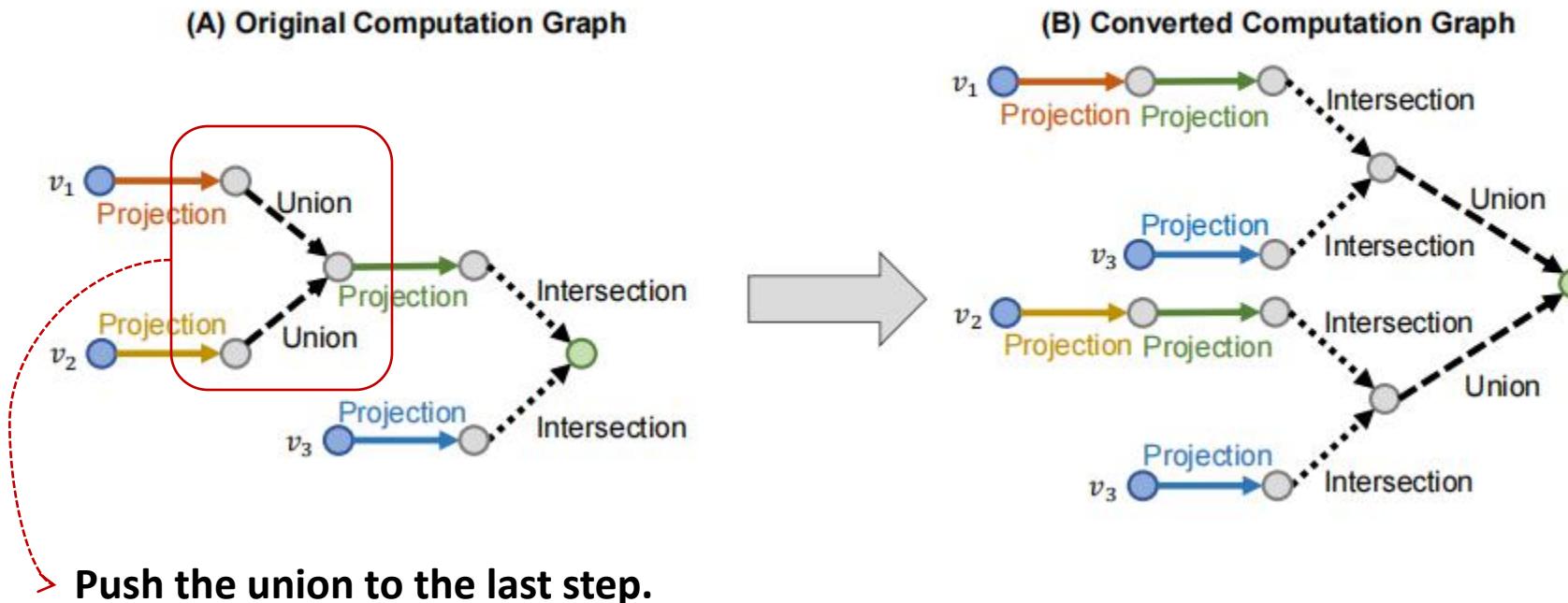
We want **red dots** (answers) to be in the box while the blue dots (negative answers) to be outside the box.

For 3 points, 2-dimension is okay!

We can not design a box embedding for $q_2 \vee q_4$, that only v_2 and v_4 ? are in the box but v_1 and v_2 are outside the box.

Any **first-order logic** can be transformed into the equivalent **DNF** (Davey & Priestley, 2002).

- **Union:** Given a set of entity sets $\{S_1, S_2, \dots, S_n\}$, this operator obtains $\cup_{i=1}^n S_i$.



Training

- Randomly sample a query q from the training graph G_{train} , answer $v \in [q]_{G_{train}}$, and a **negative sample** $v' \in [q]_{G_{train}}$.
- Embed the query \mathbf{q} .
- Calculate the score $f_q(v)$ and $f_q(v')$
- Optimize the loss ℓ to maximize $f_q(v)$ while minimize $f_q(v')$

$$L = -\log \sigma(\gamma - \text{dist}_{\text{box}}(\mathbf{v}; \mathbf{q})) - \sum_{i=1}^k \frac{1}{k} \log \sigma(\text{dist}_{\text{box}}(\mathbf{v}'_i; \mathbf{q}) - \gamma)$$

a fixed scalar margin

the number of negative entities

Given a training set of queries and their answers, we optimize a negative sampling loss (Mikolov et al., 2013) to effectively optimize our distance-based model (Sun et al., 2019):

Result

Method	Avg	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k										
Q2B	0.484	0.786	0.413	0.303	0.593	0.712	0.211	0.397	0.608	0.33
GQE	0.386	0.636	0.345	0.248	0.515	0.624	0.151	0.310	0.376	0.273
GQE-DOUBLE	0.384	0.630	0.346	0.250	0.515	0.611	0.153	0.320	0.362	0.271
FB15k-237										
Q2B	0.268	0.467	0.24	0.186	0.324	0.453	0.108	0.205	0.239	0.193
GQE	0.228	0.402	0.213	0.155	0.292	0.406	0.083	0.17	0.169	0.163
GQE-DOUBLE	0.23	0.405	0.213	0.153	0.298	0.411	0.085	0.182	0.167	0.16
NELL995										
Q2B	0.306	0.555	0.266	0.233	0.343	0.48	0.132	0.212	0.369	0.163
GQE	0.247	0.418	0.228	0.205	0.316	0.447	0.081	0.186	0.199	0.139
GQE-DOUBLE	0.248	0.417	0.231	0.203	0.318	0.454	0.081	0.188	0.2	0.139

Table 2: H@3 results of QUERY2BOX vs. GQE on FB15k, FB15k-237 and NELL995.

Method	Avg	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k										
Q2B	0.484	0.786	0.413	0.303	0.593	0.712	0.211	0.397	0.608	0.330
Q2B-AVG	0.468	0.779	0.407	0.300	0.577	0.673	0.199	0.345	0.607	0.326
Q2B-DEEPSETS	0.467	0.755	0.407	0.294	0.588	0.699	0.197	0.378	0.562	0.324
Q2B-AVG-1P	0.385	0.812	0.262	0.173	0.463	0.529	0.126	0.263	0.653	0.187
Q2B-SHAREDOFFSET	0.372	0.684	0.335	0.232	0.442	0.559	0.144	0.282	0.417	0.252
FB15k-237										
Q2B	0.268	0.467	0.24	0.186	0.324	0.453	0.108	0.205	0.239	0.193
Q2B-AVG	0.249	0.462	0.242	0.182	0.278	0.391	0.101	0.158	0.236	0.189
Q2B-DEEPSETS	0.259	0.458	0.243	0.186	0.303	0.432	0.104	0.187	0.231	0.190
Q2B-AVG-1P	0.219	0.457	0.193	0.132	0.251	0.319	0.083	0.142	0.241	0.152
Q2B-SHAREDOFFSET	0.207	0.391	0.199	0.139	0.251	0.354	0.082	0.154	0.15	0.142
NELL995										
Q2B	0.306	0.555	0.266	0.233	0.343	0.480	0.132	0.212	0.369	0.163
Q2B-AVG	0.283	0.543	0.250	0.228	0.300	0.403	0.116	0.188	0.36	0.161
Q2B-DEEPSETS	0.293	0.539	0.26	0.231	0.317	0.467	0.11	0.202	0.349	0.16
Q2B-AVG-1P	0.274	0.607	0.229	0.182	0.277	0.315	0.097	0.18	0.443	0.133
Q2B-SHAREDOFFSET	0.237	0.436	0.219	0.201	0.278	0.379	0.096	0.174	0.217	0.137

Table 3: H@3 results of QUERY2BOX vs. several variants on FB15k, FB15k-237 and NELL995.

補: Embedding Logical Queries on Knowledge Graphs((NeurIPS 2018)

Algorithm 1: Query embedding generation

Input :Query anchor nodes \mathcal{A} , query variable nodes \mathcal{B} ,
query edges \mathcal{E}_q , a map d_q from query variables
to their degree in the query DAG

Output:Query embedding \mathbf{q}

$Q = \text{dictionary mapping every } V_i \in \mathcal{B} \text{ to an empty set};$

for $\tau(v_i, V_j) \in \mathcal{E}_q : v_i \in \mathcal{A}$ **do**
| $Q[V_j] = Q[V_j] \cup \mathcal{P}(\mathbf{z}_{v_i}, \tau)$

while $|Q.\text{key_set}| > 0$ **do**

$A = \text{empty dictionary};$

for $V_i \in Q.\text{key_set} : |Q[V_i]| = d_q(V_i)$ **do**
| $A[V_i] = \mathcal{I}(Q[V_i]);$
| $\text{delete } Q[V_i];$

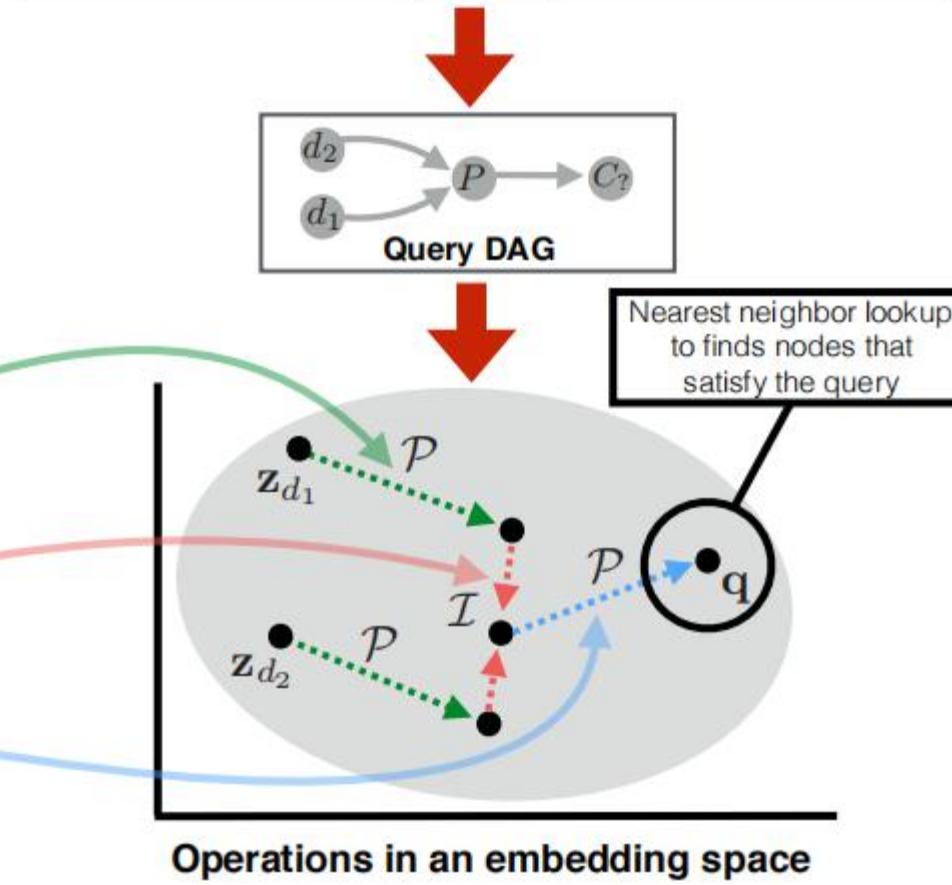
for $V_i \in A.\text{key_set}$ **do**

for $\tau(V_j, V_k) \in \mathcal{E}_q : V_j = V_i$ **do**
| $Q[V_k] = Q[V_k] \cup \mathcal{P}(A[V_i], \tau);$

return $A[V_?];$

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

Input query



Thanks