

# **Stanford CS224W: Reasoning in Knowledge Graphs using Embeddings**

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

<http://cs224w.stanford.edu>

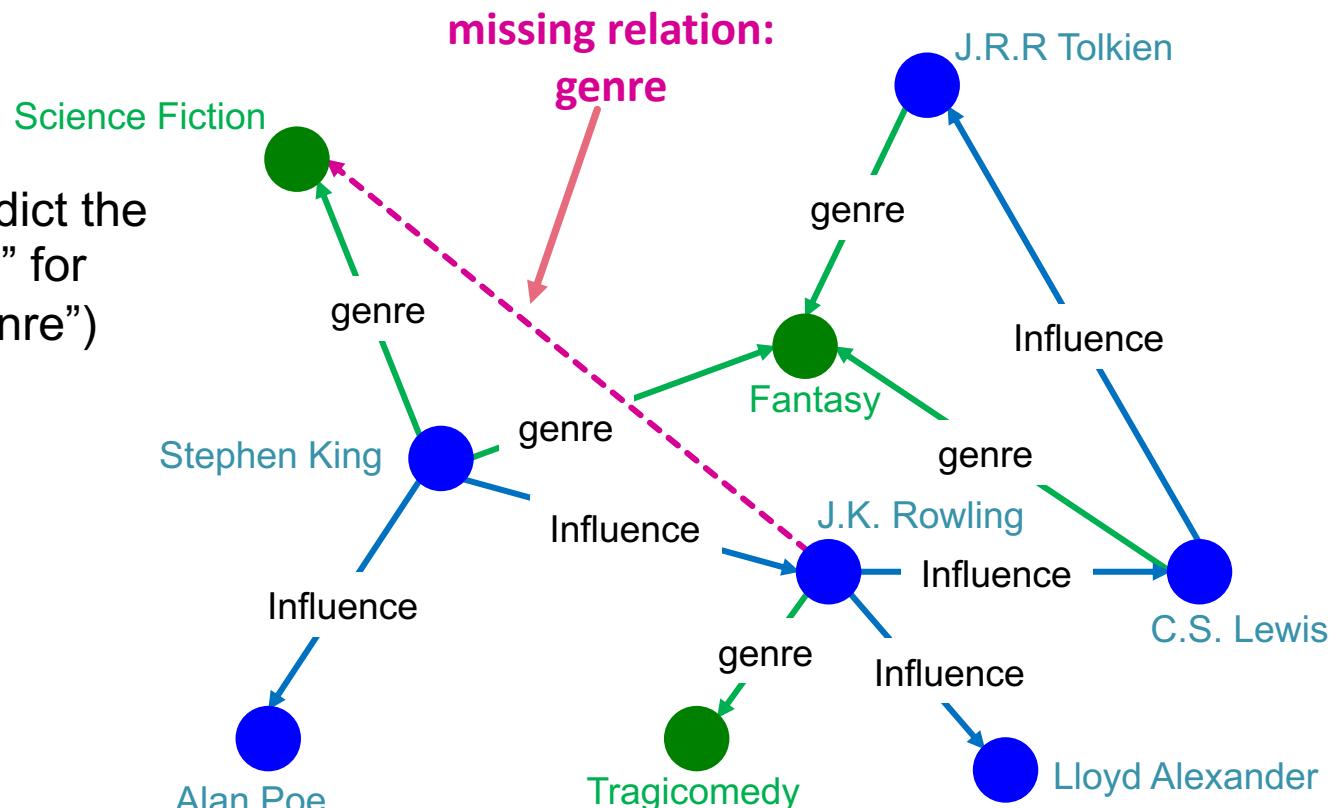


# Recap: KG Completion Task

# Given an enormous KG, can we complete the KG?

- For a given (**head**, **relation**), we predict missing **tails**.
    - (Note this is slightly different from link prediction task)

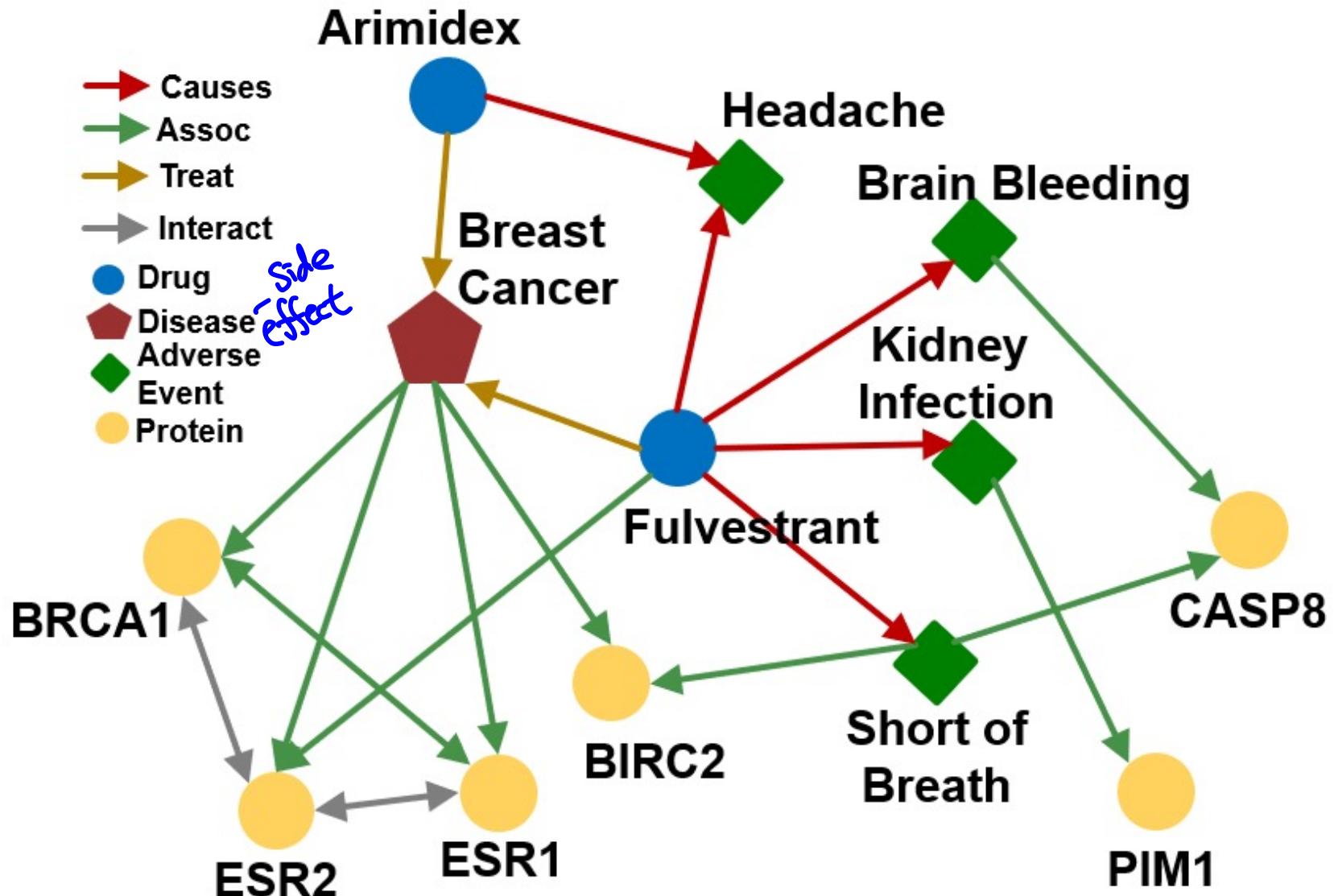
**Example task:** Predict the tail “Science Fiction” for (“J.K. Rowling”, “genre”)



# Today: Reasoning over KGs

- **Goal:**
  - How to perform multi-hop reasoning over KGs?
- **Reasoning over Knowledge Graphs**
  - Answering multi-hop queries
    - Path Queries
    - Conjunctive Queries
  - Query2box

# Example KG: Biomedicine

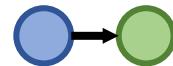


# Predictive Queries on KG

Can we do multi-hop reasoning, i.e., answer complex queries on an incomplete, massive KG?

Query Types	Examples: Natural Language Question, Query
One-hop Queries	What adverse event is caused by Fulvestrant? (e:Fulvestrant, (r:Causes))
Path Queries	What protein is associated with the adverse event caused by Fulvestrant? (e:Fulvestrant, (r:Causes, r:Assoc))
Conjunctive Queries	What is the drug that treats breast cancer and caused headache? ((e:BreastCancer, (r:TreatedBy)), (e:Migraine, (r:CausedBy)))

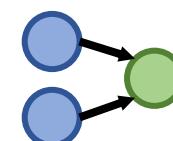
In this lecture, we only focus on answering **queries** on a KG!  
The notation will be detailed next.



One-hop Queries



Path Queries

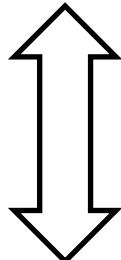


Conjunctive Queries

Goal: able to answer these queries over incomplete knowledge graphs.

# Predictive One-hop Queries

- We can formulate knowledge graph completion problems as answering one-hop queries.
- **KG completion:** Is link  $(h, r, t)$  in the KG?



- **One-hop query:** Is  $t$  an answer to query  $(h, r)$ ?
  - **For example:** What side effects are caused by drug Fulvestrant?

*caused by  
/  
r*

# Path Queries

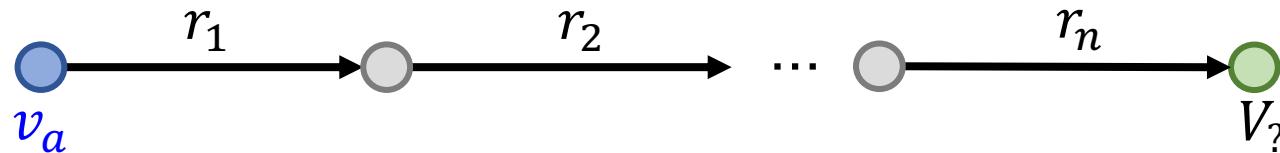
- Generalize one-hop queries to path queries by **adding more relations on the path**.

- An  $n$ -hop path query  $q$  can be represented by

$$q = (v_a, (r_1, \dots, r_n))$$

- $v_a$  is an “anchor” entity,
- Let answers to  $q$  in graph  $G$  be denoted by  $\llbracket q \rrbracket_G$ .

Query Plan of  $q$ :

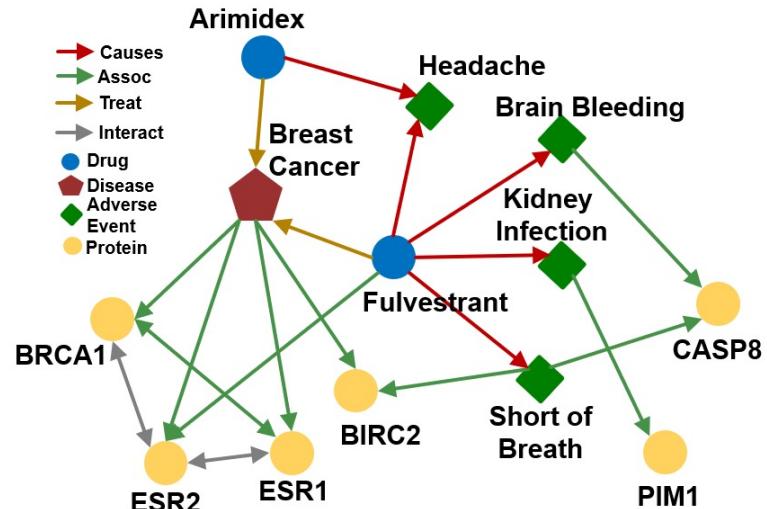
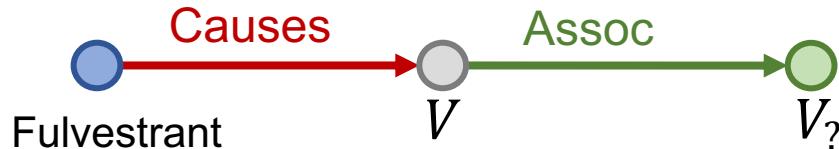


Query plan of path queries is a chain.

# Path Queries

Question: “What proteins are *associated* with adverse events *caused* by *Fulvestrant*?”

- $v_a$  is **e:Fulvestrant**
- $(r_1, r_2)$  is **(r:Causes, r:Assoc)**
- **Query: (e:Fulvestrant, (r:Causes, r:Assoc))**

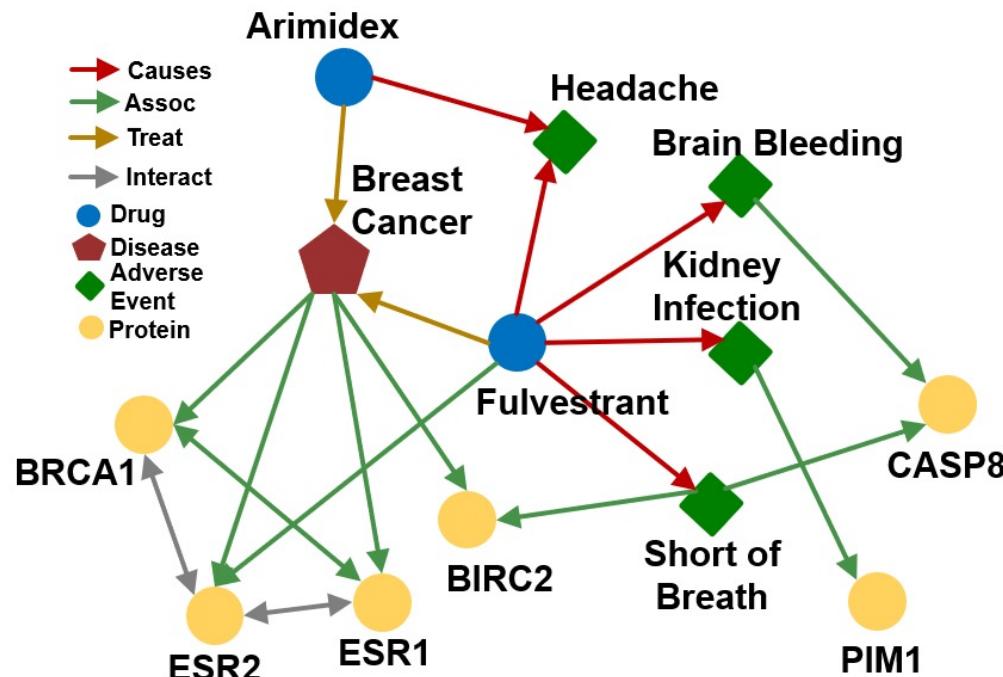


# Path Queries

Question: “What proteins are *associated* with adverse events *caused* by *Fulvestrant*?”

- Query: (e:Fulvestrant, (r:Causes, r:Assoc))

Given a KG, how to answer a path query?



# Traversing Knowledge Graphs

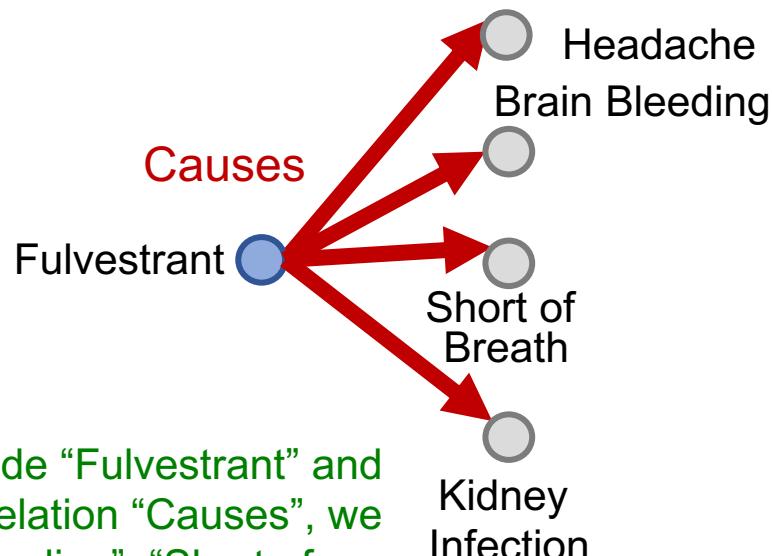
- We answer path queries by traversing the KG:  
“What proteins are *associated* with adverse events *caused* by *Fulvestrant*? ”
- Query: (*e:Fulvestrant*, (*r:Causes*, *r:Assoc*))

Fulvestrant 

Start from the  
**anchor node**  
(Fulvestrant).

# Traversing Knowledge Graphs

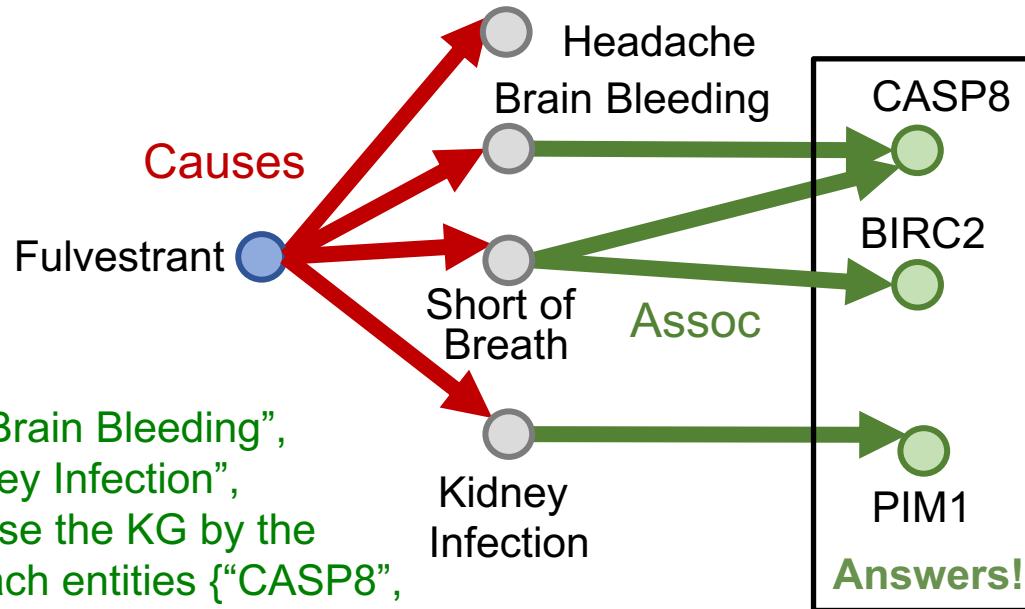
- We answer path queries by traversing the KG:  
“What proteins are *associated* with adverse events *caused* by *Fulvestrant*? ”
- Query: (e:Fulvestrant, (r:Causes, r:Assoc))



Start from the anchor node “Fulvestrant” and traverse the KG by the relation “Causes”, we reach entities {“Brain Bleeding”, “Short of Breath”, “Kidney Infection”, “Headache”}.

# Traversing Knowledge Graphs

- We answer path queries by traversing the KG:  
“What proteins are *associated* with adverse events *caused* by *Fulvestrant*? ”
- Query: (e:Fulvestrant, (r:Causes, r:Assoc))

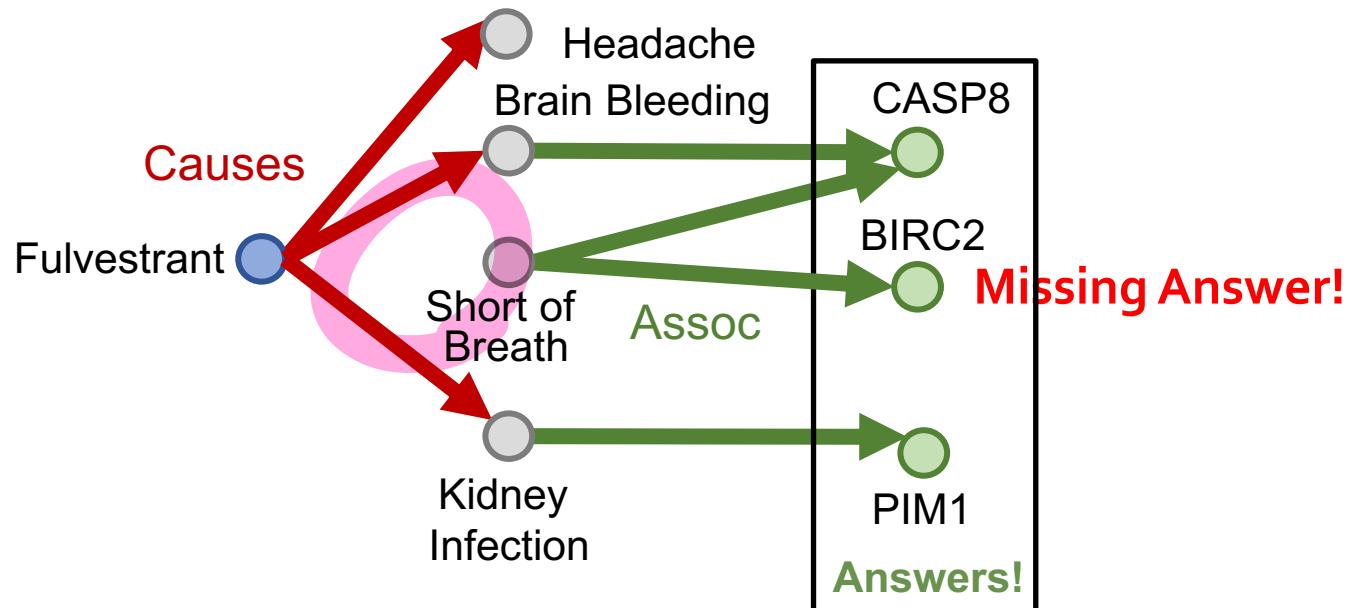


# However, KGs are incomplete

- Answering queries seems easy: Just traverse the graph.
- But KGs are incomplete and unknown:
  - Many relations between entities are missing or are incomplete
    - For example, we lack all the biomedical knowledge
    - Enumerating all the facts takes non-trivial time and cost, we cannot hope that KGs will ever be fully complete
- Due to KG incompleteness, one is not able to identify all the answer entities

# Example: Incomplete KG

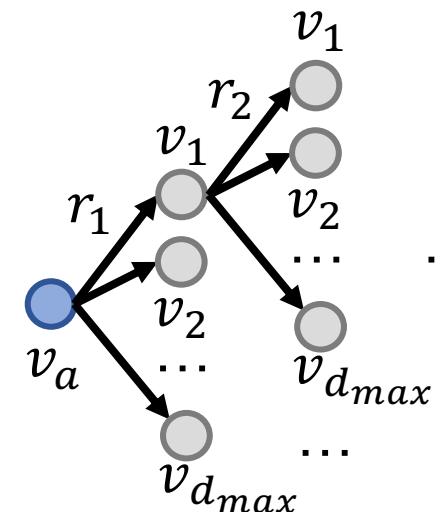
- We answer path queries by traversing the KG:  
“What proteins are *associated* with adverse events *caused* by *Fulvestrant*? ”
- Query: (e:Fulvestrant, (r:Causes, r:Assoc))



# Can KG Completion Help?

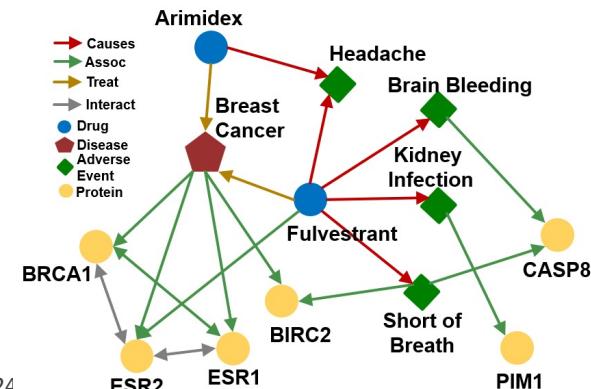
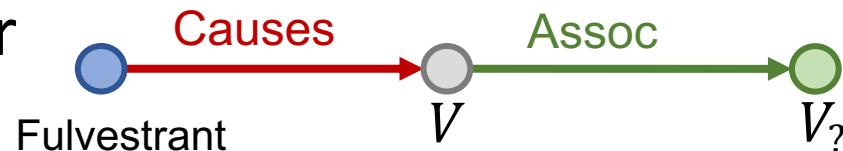
Can we first do KG completion and then traverse the completed (probabilistic) KG?

- No! The “completed” KG is a **dense graph**!
  - Most  $(h, r, t)$  triples (edge on KG) will have some non-zero probability.
- Time complexity of traversing a dense KG is exponential as a function of the path length  $L$ :  $O(d_{max}^L)$



# Task: Predictive Queries

- We need a way to answer path-based queries over an incomplete knowledge graph.
- We want our approach to implicitly impute and account for the incomplete KG.
- **Task: Predictive queries**
  - Want to be able to answer arbitrary queries while implicitly imputing for the missing information
  - **Generalization of the link prediction task**



# **Stanford CS224W: Answering Predictive Queries on Knowledge Graphs**

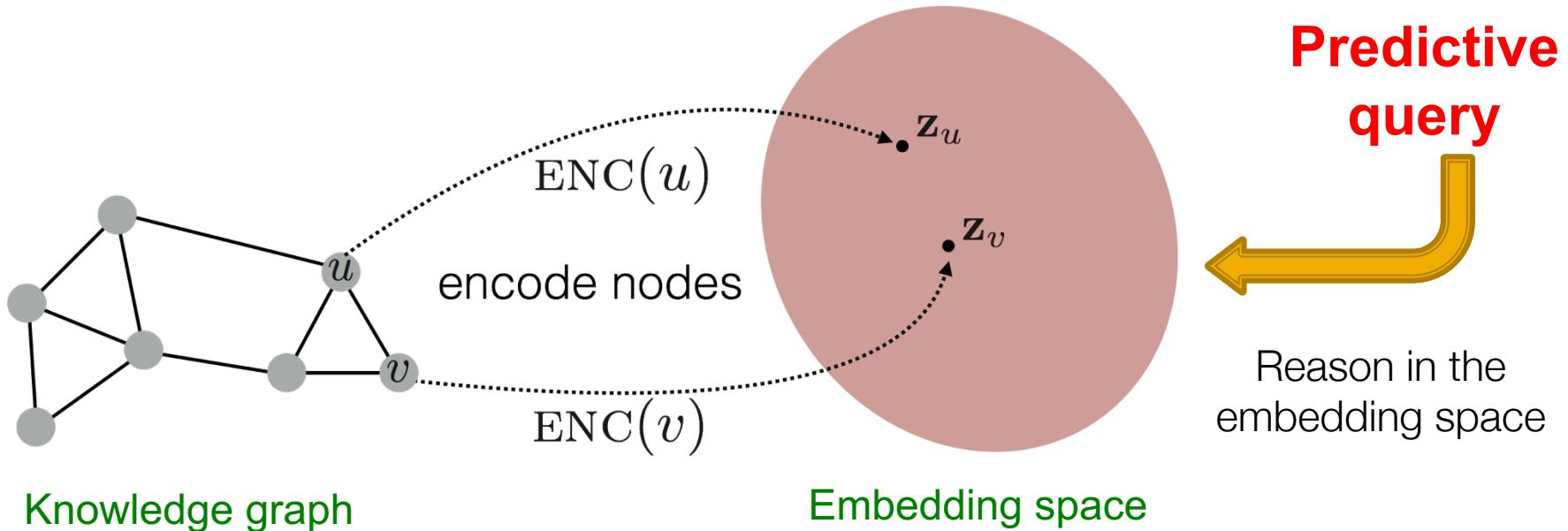
CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

<http://cs224w.stanford.edu>



# General Idea



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

- Embed query into a single **point** in the Euclidean space: answer nodes are close to the query.
- **Query2Box:** Embed query into a hyper-rectangle (**box**) in the Euclidean space: answer nodes are enclosed in the box.

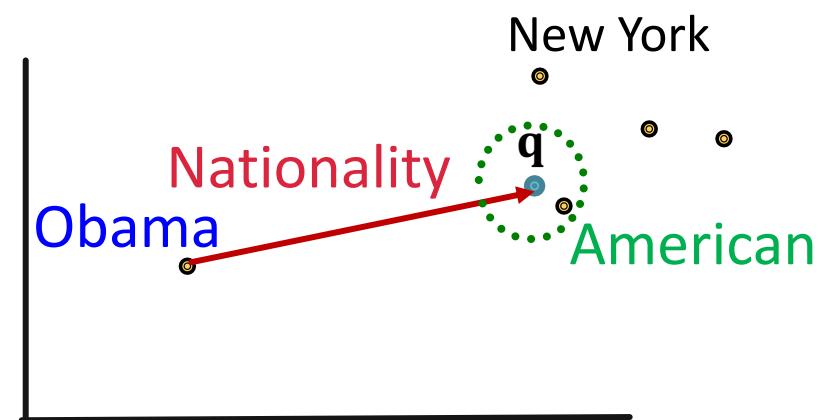
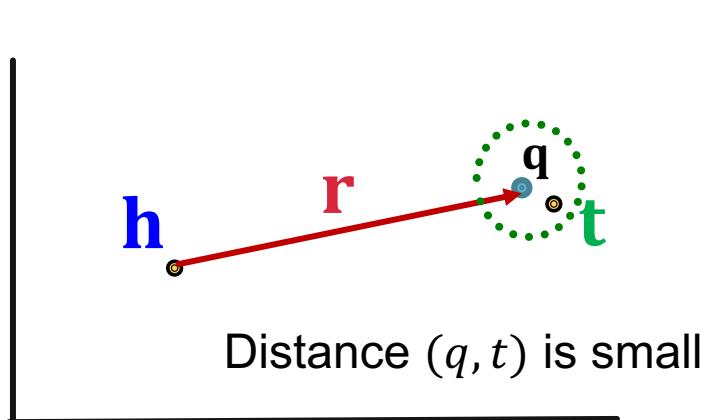
[[Embedding Logical Queries on Knowledge Graphs](#). Hamilton, et al., NeurIPS 2018]

[[Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings](#). Ren, et al., ICLR 2020]

# Idea: Traversing KG in Vector Space

- **Key idea: Embed queries!**

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

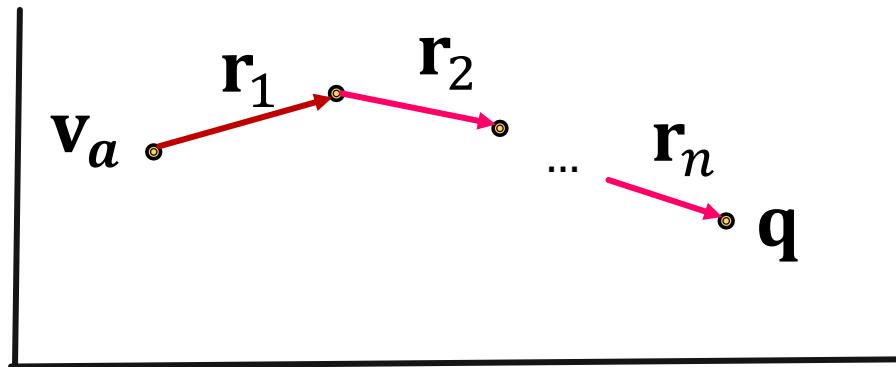


# Traversing KG in Vector Space

- Key idea: Embed queries!

- Generalize **TransE** to multi-hop reasoning.

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



$$q = v_a + r_1 + \cdots + r_n$$

- The embedding process **only involves vector addition, independent of # entities** in the KG!

# Traversing KG in Vector Space (1)

Embed path queries in vector space.

- Question: “What proteins are *associated* with adverse events *caused* by *Fulvestrant*?”
- Query: (e:*Fulvestrant*, (r:*Causes* , r:*Assoc*))

Follow the query plan:

Query Plan

Embedding Process

Fulvestrant •

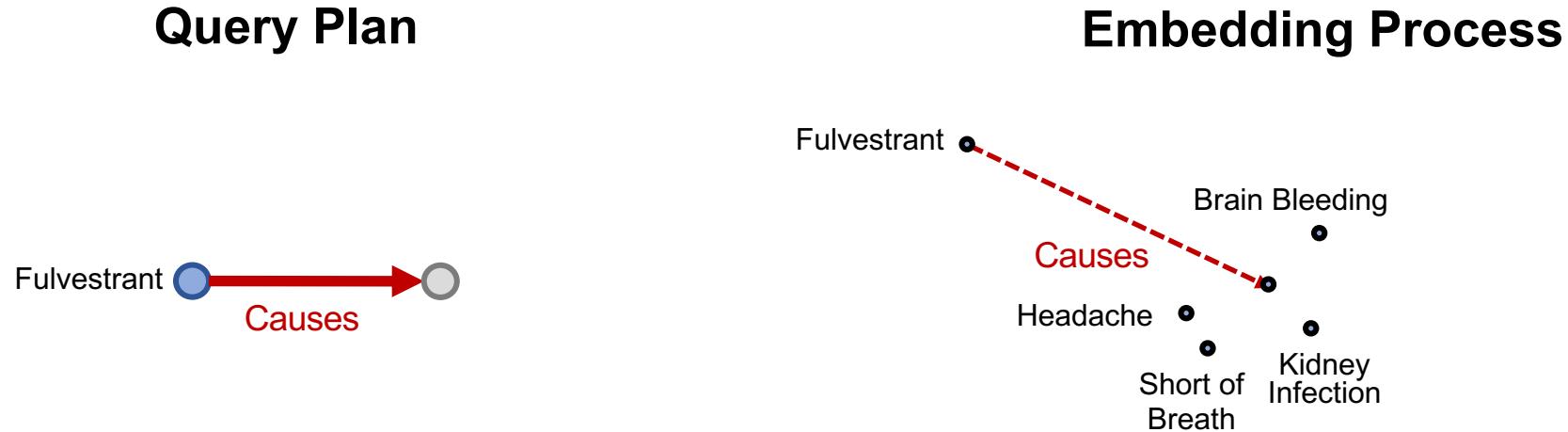
Fulvestrant ●

# Traversing KG in Vector Space (2)

Embed path queries in vector space.

- Question: “What proteins are *associated* with adverse events *caused* by *Fulvestrant*?”
- Query: (e:*Fulvestrant*, (r:*Causes* , r:*Assoc*))

Follow the query plan:

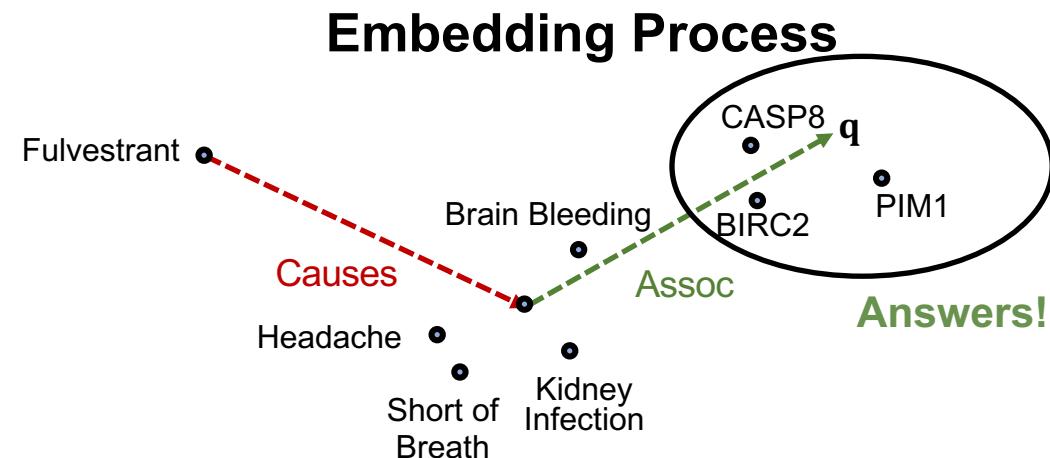
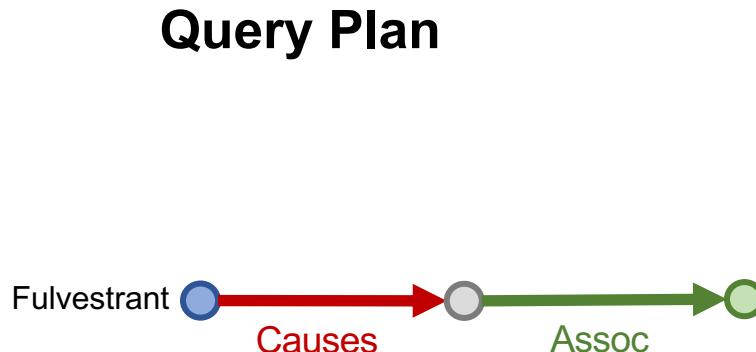


# Traversing KG in Vector Space (3)

Embed path queries in vector space.

- Question: “What proteins are *associated* with adverse events *caused* by *Fulvestrant*?”
- Query: (e:*Fulvestrant*, (r:*Causes* , r:*Assoc*))

Follow the query plan:



# Traversing KG in Vector Space (4)

## Insights:

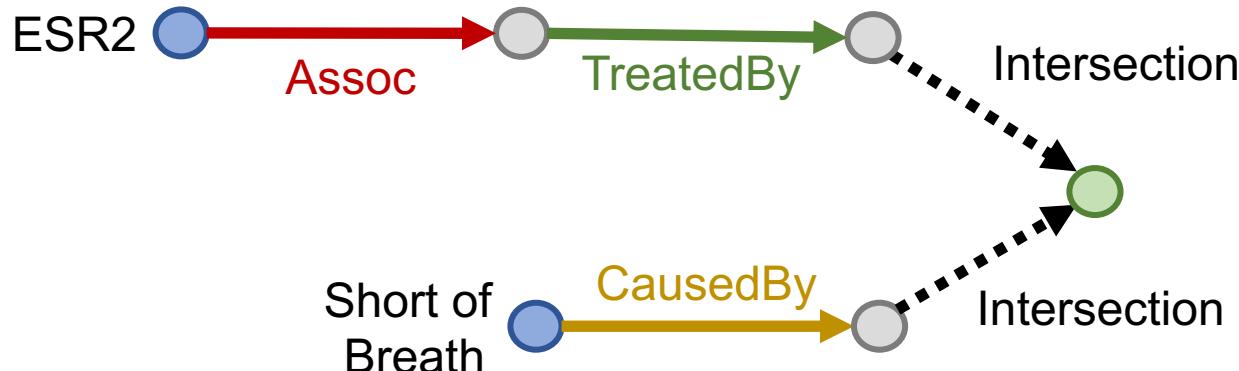
- We can train **TransE** to optimize knowledge graph completion objective (Lecture 10)
- Since **TransE** can naturally handle **compositional relations**, it can handle path queries by translating in the latent space **for multiple hops using addition of relation embeddings**.
- For **TransR** / **DistMult** / **ComplEx**, since they cannot handle compositional relations, they cannot be easily extended to handle **path queries**.

# Conjunctive Queries

Can we answer **more complex queries with logic conjunction operation?**

- **Conjunctive Queries:** “*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

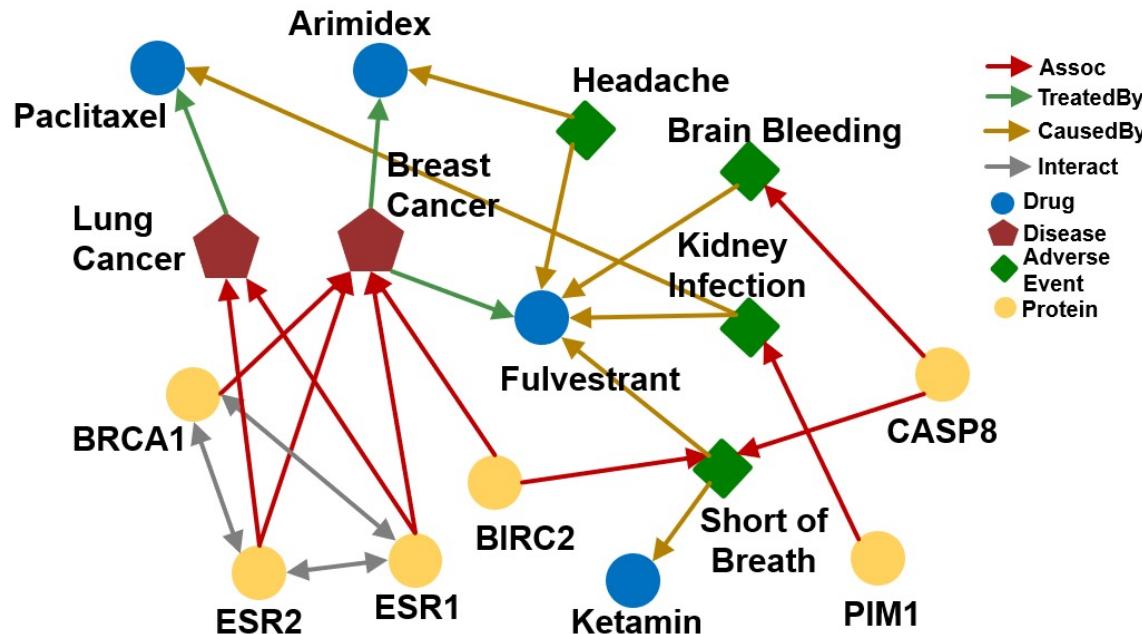
**Query plan:**



# Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

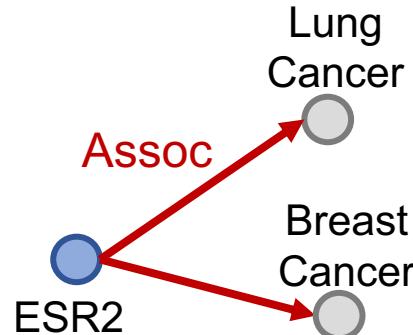
How do we answer the question by KG traversal?



# Traversing KG for Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
 $((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short\ of\ Breath, (r:CausedBy)))$

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath**:

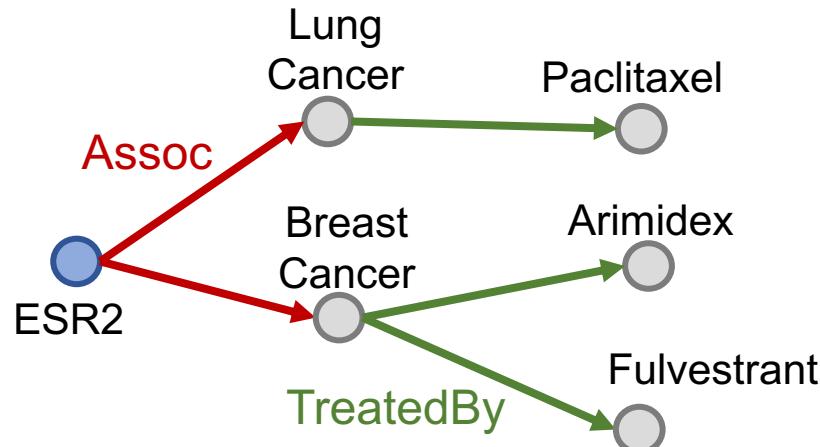


Traverse from the first anchor “ESR2” by relation “Assoc”, we reach a set of entities {“Lung Cancer”, “Breast Cancer”}

# Traversing KG for Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

Traverse KG from **anchor nodes**: ESR2 and Short of Breath:

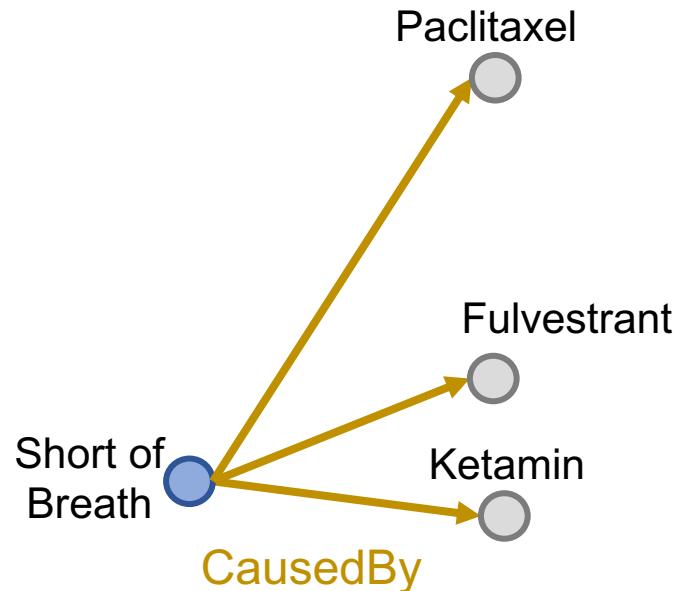


Traverse from the set of entities {"Lung Cancer", "Breast Cancer"} by relation TreatedBy, we reach a set of entities {"Paclitaxel", "Arimidex", "Fulvestrant"}

# Traversing KG for Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath**:

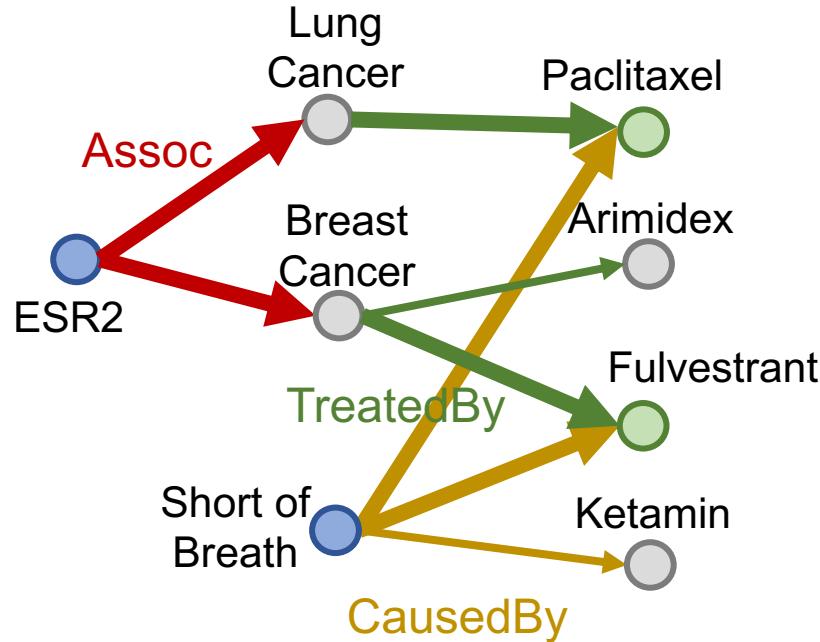


Traverse from the second anchor “Short of Breath” by relation “CausedBy”, we reach a set of entities {“Fulvestrant”, “Ketamin”, “Paclitaxel”}

# Traversing KG for Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

Traverse KG from **anchor nodes**: ESR2 and Short of Breath:

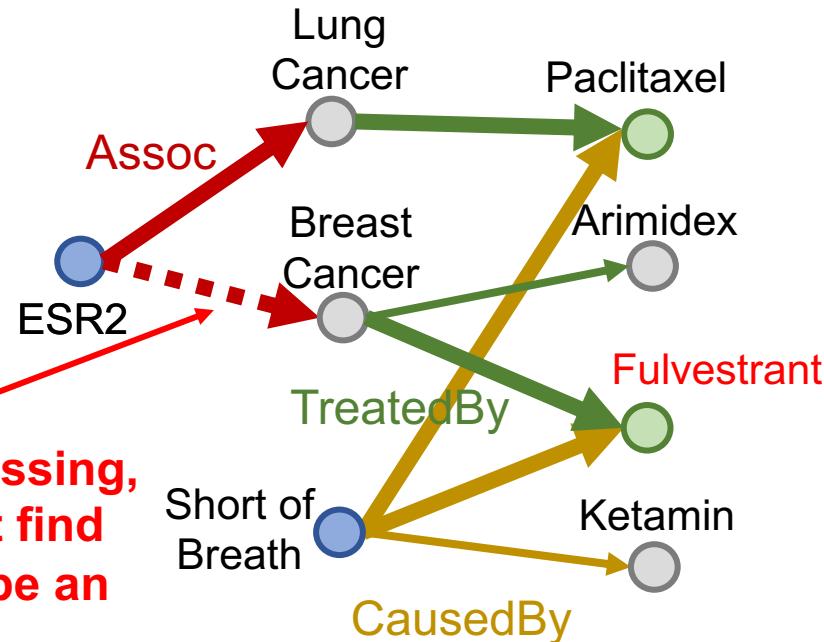


We take intersection between the two sets and get the answers {“Fulvestrant”, “Paclitaxel”}

# Traversing KG for Conjunctive Queries

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”  
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

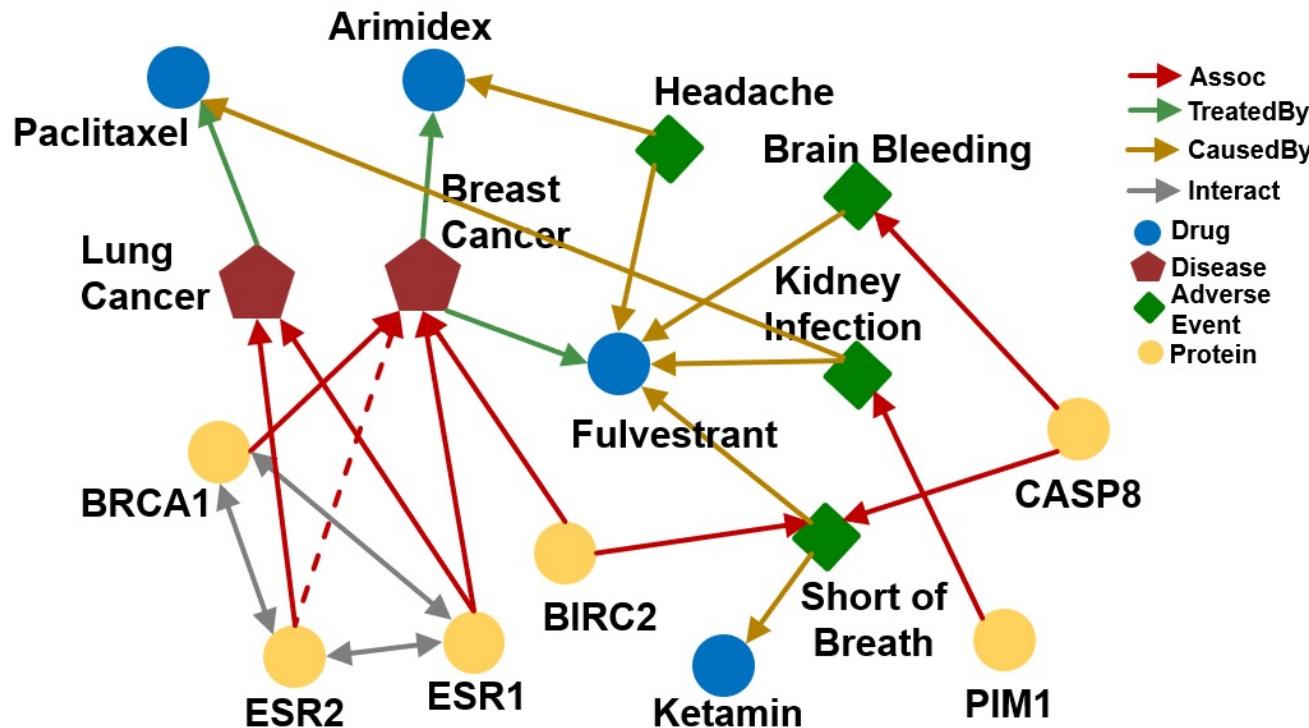
Traverse KG from **anchor nodes**: ESR2 and Short of Breath:



If this link is missing, then we cannot find Fulvestrant to be an answer.

\* If graph incomplete, knowledge graph traversal won't work.

# Traversing KG for Conjunctive Queries



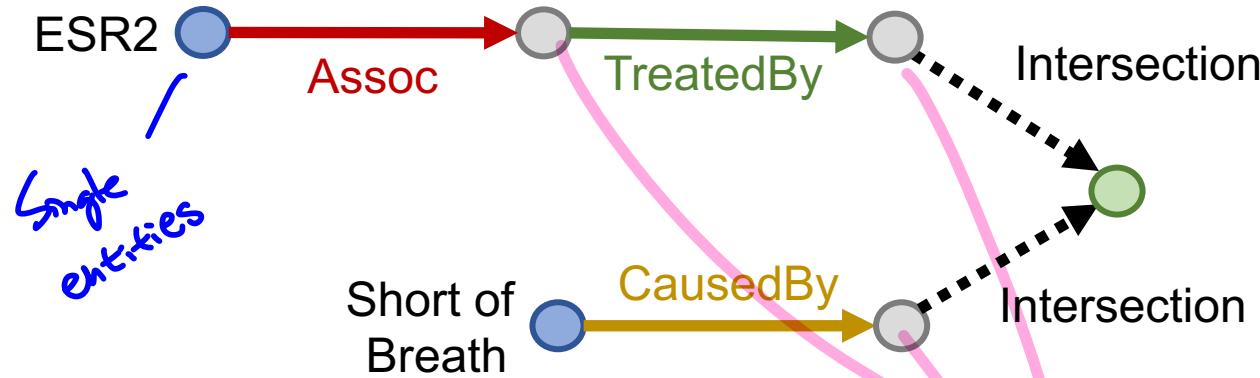
- How can we use embeddings to implicitly impute the missing (ESR2, Assoc, Breast Cancer)?
- **Intuition:** ESR2 interacts with both BRCA1 and ESR1. Both proteins are associated with breast cancer.

# Traversing KG in Vector Space

- “What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?”

((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

Query plan:



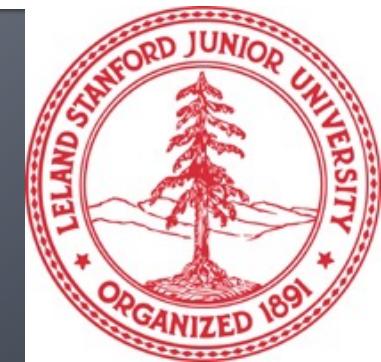
Each intermediate node represents a set of entities, how do we represent it? How do we define the intersection operation in the latent space?

# **Stanford CS224W: Query2box: Reasoning over KGs Using Box Embeddings**

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

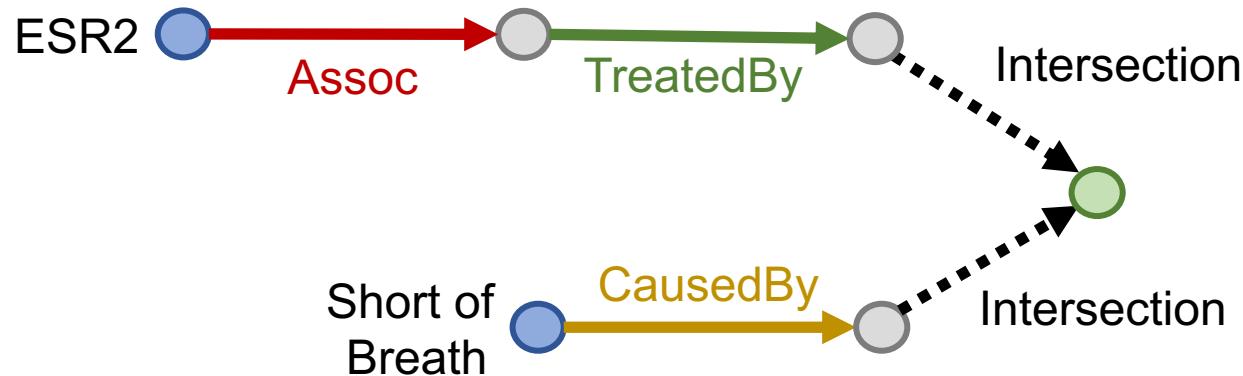
<http://cs224w.stanford.edu>



# Conjunctive Queries

How can we answer **more complex queries with logical conjunction operation?**

**Query plan:**



- (1) Each intermediate node represents a set of entities; how do we represent it?
- (2) How do we define the intersection operation in the latent space?

# Box Embeddings

- Embed queries with **hyper-rectangles (boxes)**

$$\mathbf{q} = (Center(q), Offset(q))$$

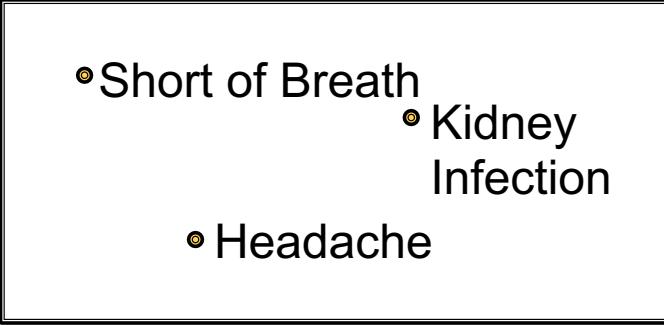
• Short of Breath  
q • Kidney  
• Headache  
Infection

For example, we can embed the adverse events of Fulvestrant with a **box that enclose all the answer entities**.

Embedding Space

# Key Insight: Intersection

- **Intersection of boxes is well-defined!**
- When we traverse the KG to find the answers, each step produces a set of reachable entities.
- **How can we better model these sets?**
  - Boxes are a **powerful abstraction**, as we can project the center and control the offset to model the set of entities enclosed in the box

- 
- Short of Breath
  - Kidney Infection
  - Headache

# Embed with Box Embedding

## Things to figure out:

- **Entity embeddings** (# params:  $d|V|$ ):
  - Entities are seen as zero-volume boxes
- **Relation embeddings** (# params  $2d|R|$ )
  - Each relation takes a box and produces a new box
- **Intersection operator  $f$**  :
  - New operator, inputs are boxes and output is a box
  - Intuitively models intersection of boxes

**Notation**  
d: out degree  
 $|V|$ : # entities  
 $|R|$ : # relations

# Embed with Box Embedding

- **Embed queries in vector space:** “*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*”  
`(e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))`

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath**:

Query plan



Embedding Space

ESR2 •

# Projection Operator

## Projection Operator $\mathcal{P}$

### ■ Intuition:

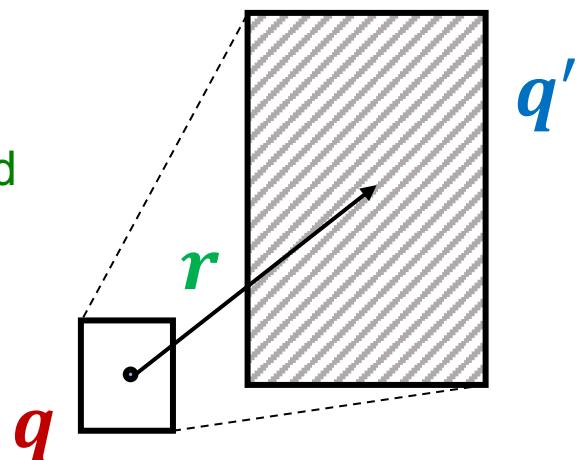
- Take the current box as input and use the **relation embedding** to **project and expand** the box!

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

$$Cen(q') = Cen(q) + Cen(r)$$

$$Off(q') = Off(q) + Off(r)$$

" $\times$ " (cross) means the projection operator is a **relation** from any box and **relation** to a new box



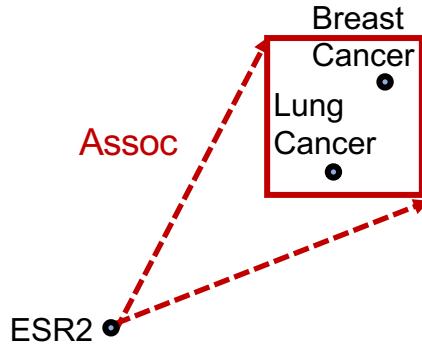
# Embed with Box Embedding

- **Embed queries in vector space:** “*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*”
- Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath**:
- Use **projection operator** again following the query plan.

Query Plan



Embedding Space



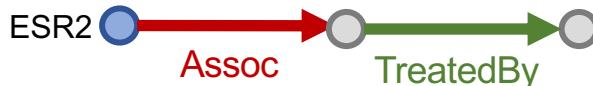
# Embed with Box Embedding

*“What is the drug that causes Short of Breath and treats disease associated with protein ESR2?”*

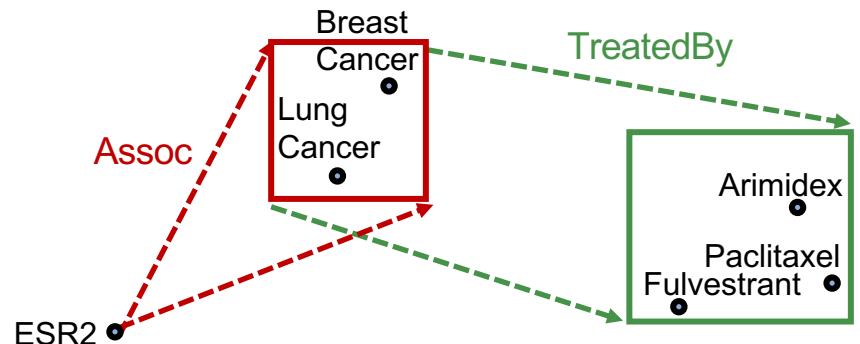
**((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))**

- Use **projection operator** again following the query plan.

**Query Plan**



**Embedding Space**



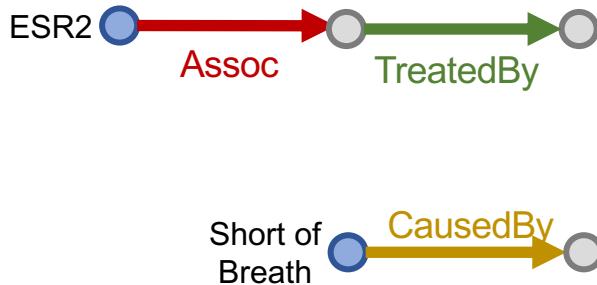
# Embed with Box Embedding

*“What is the drug that causes Short of Breath and treats disease associated with protein ESR2?”*

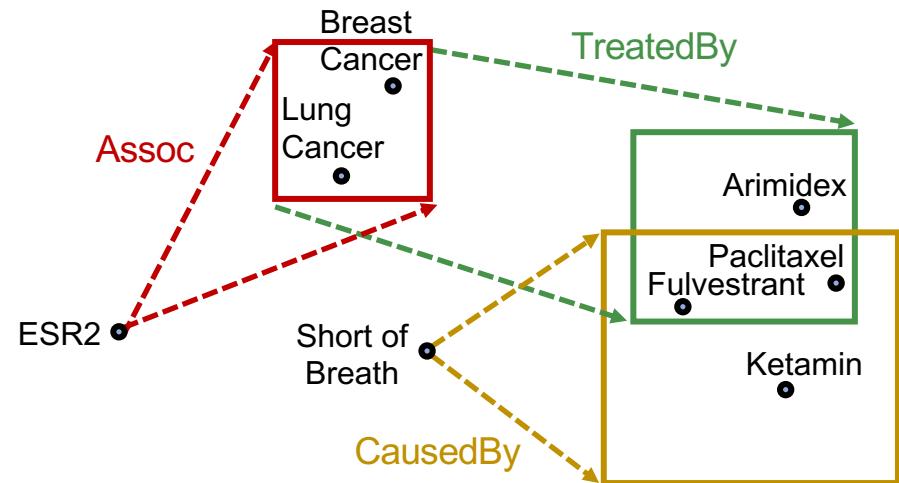
`((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))`

- Use **projection operator** again following the query plan.

**Query Plan**



**Embedding Space**



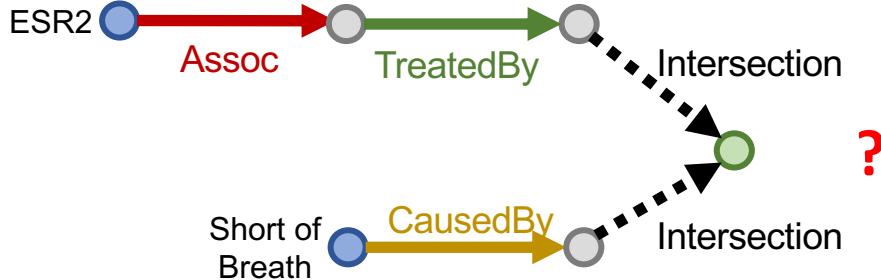
# Embed with Box Embedding

*“What is the drug that causes Short of Breath and treats disease associated with protein ESR2?”*

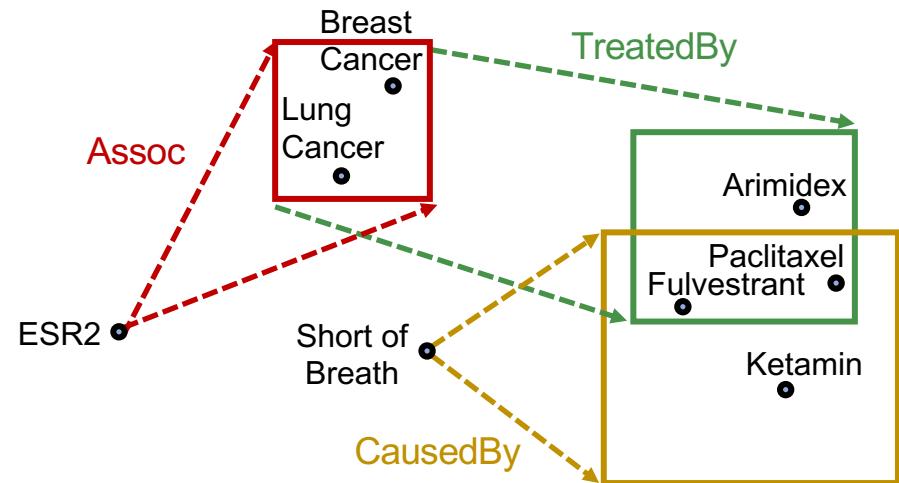
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

## ■ How do we take intersection of boxes?

Query Plan



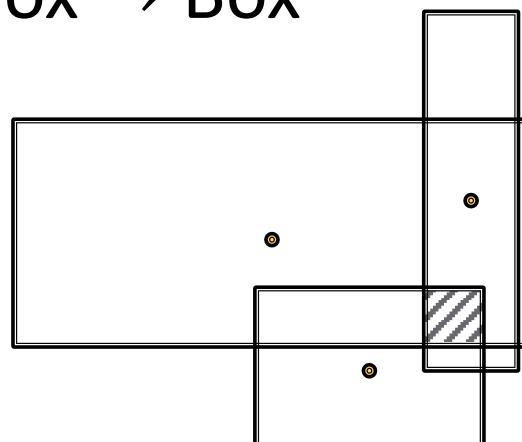
Embedding Space



# Intersection Operator

## Geometric Intersection Operator $\mathcal{I}$

- Take multiple boxes as input and produce the intersection box
- **Intuition:**
  - The center of the new blox should be “close” to the centers of the input boxes
  - The offset (box size) should **shrink** (since the size of the intersected set is **smaller** than the size of all the input set)
- $\mathcal{I} : \text{Box} \times \dots \times \text{Box} \rightarrow \text{Box}$



# Intersection Operator

## Geometric Intersection Operator $\mathcal{I}$

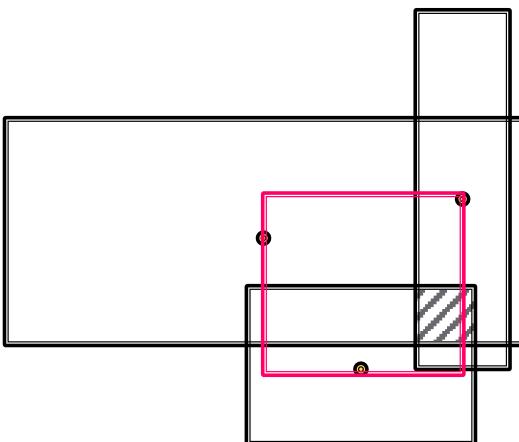
- $\mathcal{I} : \text{Box} \times \dots \times \text{Box} \rightarrow \text{Box}$

Hadamard product  
(element-wise product)

$$Cen(q_{inter}) = \sum_i w_i \odot Cen(q_i)$$

*softmax*

$$w_i = \frac{\exp(f_{cen}(Cen(q_i)))}{\sum_j \exp(f_{cen}(Cen(q_j)))}$$
$$Cen(q_i) \in \mathbb{R}^d$$
$$w_i \in \mathbb{R}^d$$



**Intuition:** The center should be in the **red** region!

**Implementation:** The center is a **weighted sum** of the input box centers

$w_i \in \mathbb{R}^d$  is calculated by a neural network  $f_{cen}$  (with trainable weights)

$w_i$  represents a “**self-attention**” score for the center of each input  $Cen(q_i)$ .

# Intersection Operator

## Geometric Intersection Operator $\mathcal{I}$

- $\mathcal{I} : \text{Box} \times \dots \times \text{Box} \rightarrow \text{Box}$

$$Off(q_{\text{inter}})$$

$$= \min(Off(q_1), \dots, Off(q_n))$$

$$\odot \sigma(f_{off}(Off(q_1), \dots, Off(q_n)))$$

guarantees shrinking

Sigmoid function:  
squashes output in  $(0,1)$

$f_{off}$  is a neural network (with trainable parameters) that extracts the representation of the input boxes to increase expressiveness

**Intuition:** The offset should be smaller than the offset of the input box

**Implementation:** We first **take minimum** of the offset of the input box, and then we make the model more expressive by introducing a new function  $f_{off}$  to extract the **representation** of the input boxes with a **sigmoid function** to **guarantee shrinking**.

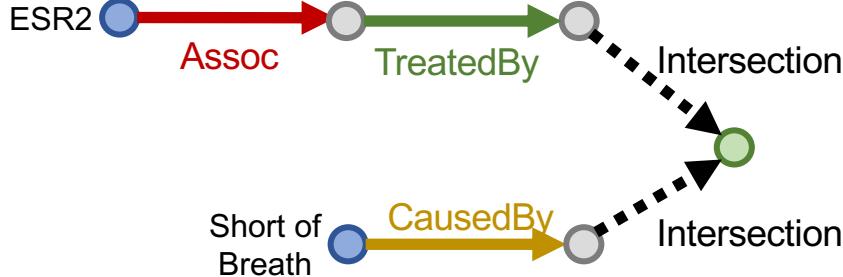
# Embed with Box Embedding

*“What is the drug that causes Short of Breath and treats disease associated with protein ESR2?”*

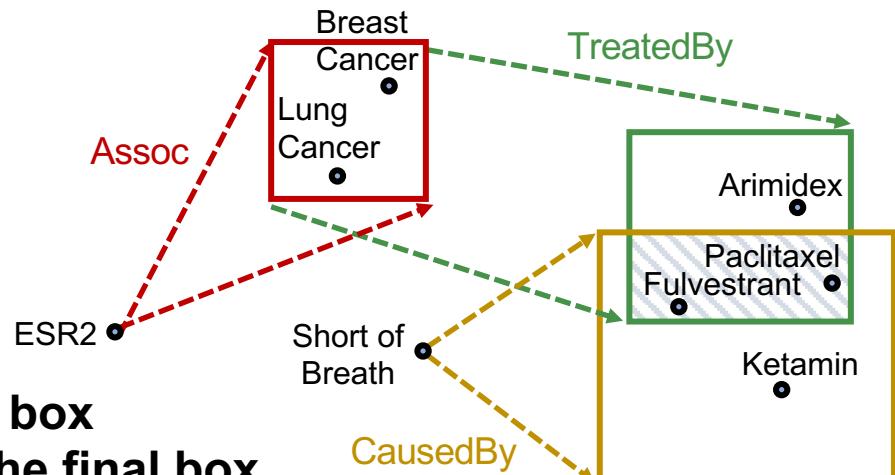
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy)))

- Use box intersection operator

Query Plan



Embedding Space



**The shadow box  
represents the final box  
embedding of the query**

# Entity-to-Box Distance

- How do we define the score function  $f_q(v)$  (negative distance)?

( $f_q(v)$  captures inverse distance of a node  $v$  as answer to  $q$ )

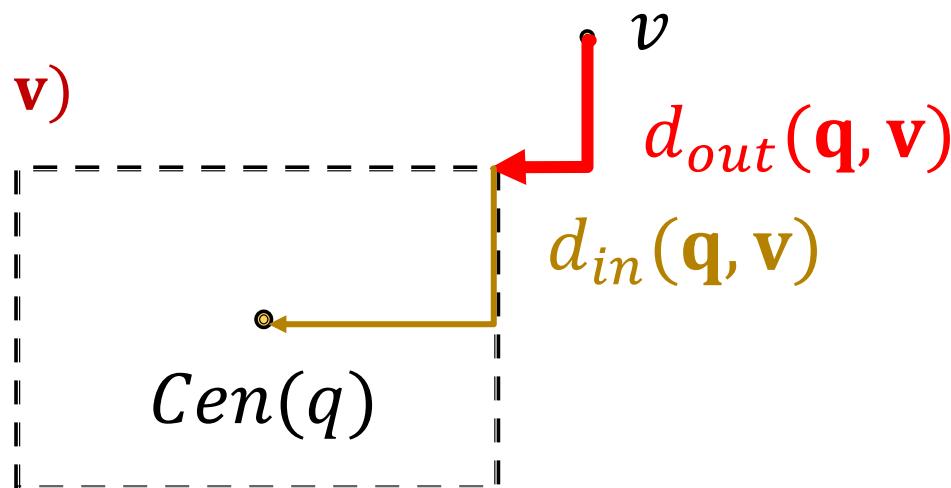
- Given a query box  $q$  and entity embedding (box)  $v$ ,

$$d_{box}(q, v) = d_{out}(q, v) + \alpha \cdot d_{in}(q, v)$$

where  $0 < \alpha < 1$ .

- Intuition: if the point is enclosed in the box, the distance should be downweighted.

- $f_q(v) = -d_{box}(q, v)$



# Extending to Union Operation

- Can we embed complex queries with **union**?  
E.g.: “What drug can treat breast cancer **or** lung cancer?”
- **Conjunctive queries + disjunction** is called Existential Positive First-order (EPFO) queries. We’ll refer to them as **AND-OR** queries.
- Can we also design a disjunction operator and embed AND-OR queries in low-dimensional vector space? 

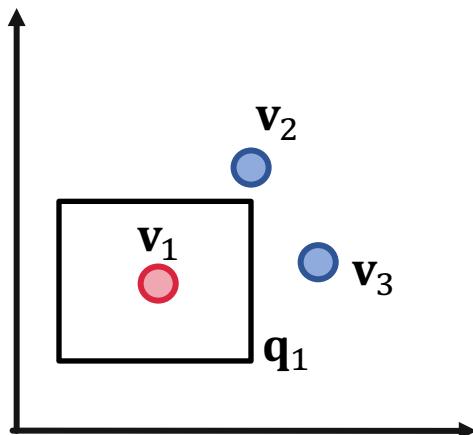
# Embedding AND-OR Queries

- Can we embed AND-OR queries in a low-dimensional vector space?  
~~dimensional vector space?~~
- No! Intuition: Allowing **union** over **arbitrary queries** requires **high-dimensional** embeddings!
- Example:
  - Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}, \llbracket q_2 \rrbracket = \{v_2\}, \llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in a **two-dimensional** plane?

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?

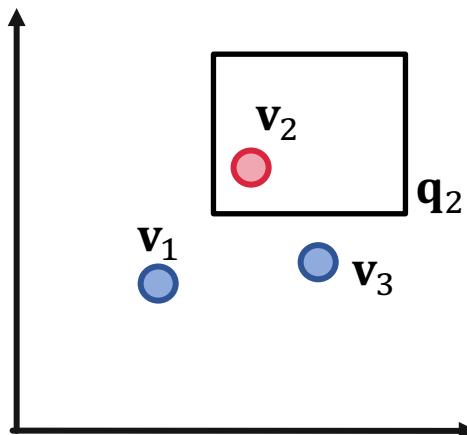


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?

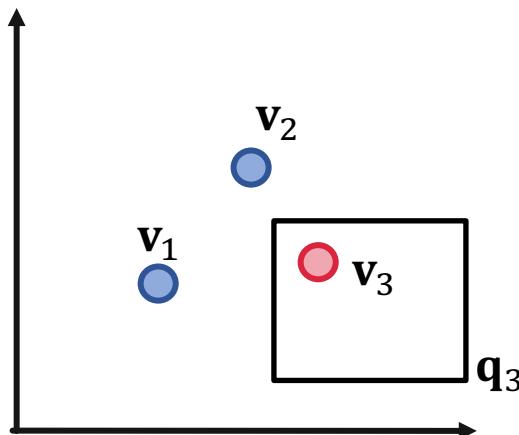


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
- $\llbracket q_1 \rrbracket = \{v_1\}, \llbracket q_2 \rrbracket = \{v_2\}, \llbracket q_3 \rrbracket = \{v_3\}$
- If we allow union operation, can we embed them in two-dimensional plane?

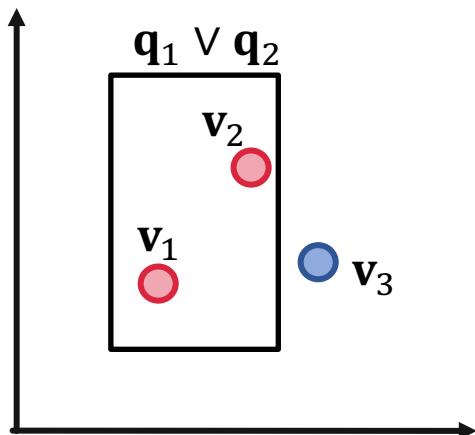


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?

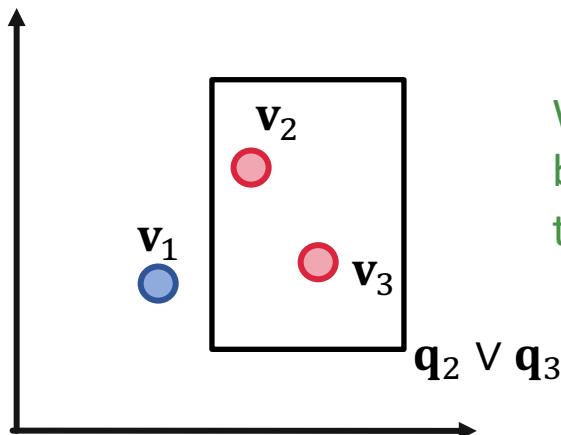


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?

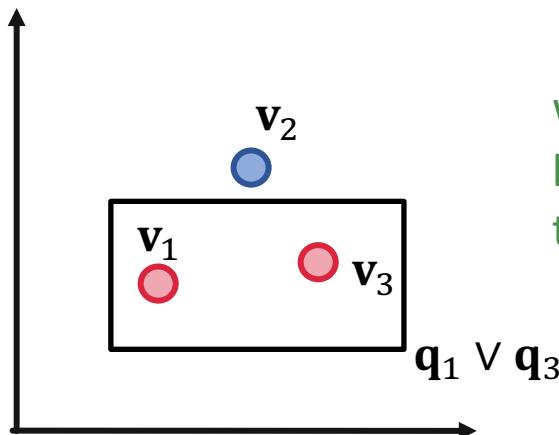


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## Example:

- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?

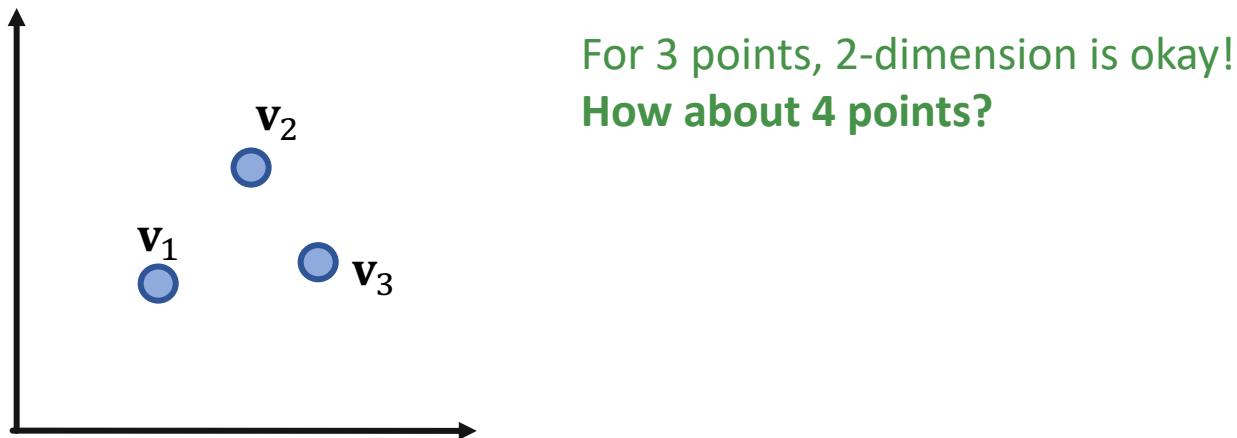


We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

# Embedding AND-OR Queries

## ■ Example:

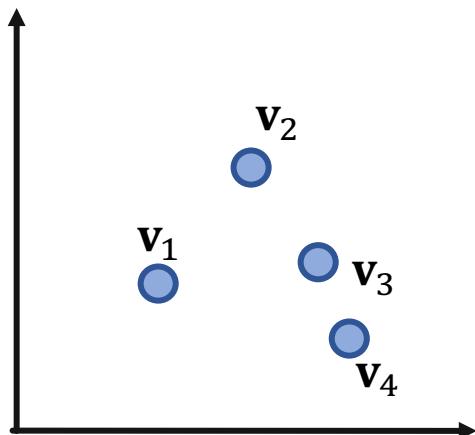
- Given 3 queries  $q_1, q_2, q_3$ , with answer sets:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



# Embedding AND-OR Queries (2)

## ■ Example 2:

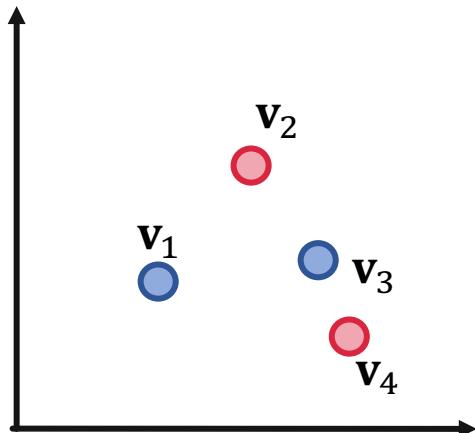
- Given 4 queries  $q_1, q_2, q_3, q_4$  with answers:
  - $\llbracket q_1 \rrbracket = \{v_1\}, \llbracket q_2 \rrbracket = \{v_2\}, \llbracket q_3 \rrbracket = \{v_3\}, \llbracket q_4 \rrbracket = \{v_4\}$ ,
  - If we allow union operation, can we embed them in two-dimensional plane?



# Embedding AND-OR Queries (2)

## Example 2:

- Given 4 queries  $q_1, q_2, q_3, q_4$  with answers:
  - $\llbracket q_1 \rrbracket = \{v_1\}$ ,  $\llbracket q_2 \rrbracket = \{v_2\}$ ,  $\llbracket q_3 \rrbracket = \{v_3\}$ ,  $\llbracket q_4 \rrbracket = \{v_4\}$ ,
  - If we allow union operation, can we embed them in two-dimensional plane?



We cannot 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  $v_3$  are outside the box.

# Embedding AND-OR Queries (2)

Can we embed AND-OR queries in low-dimensional vector space?

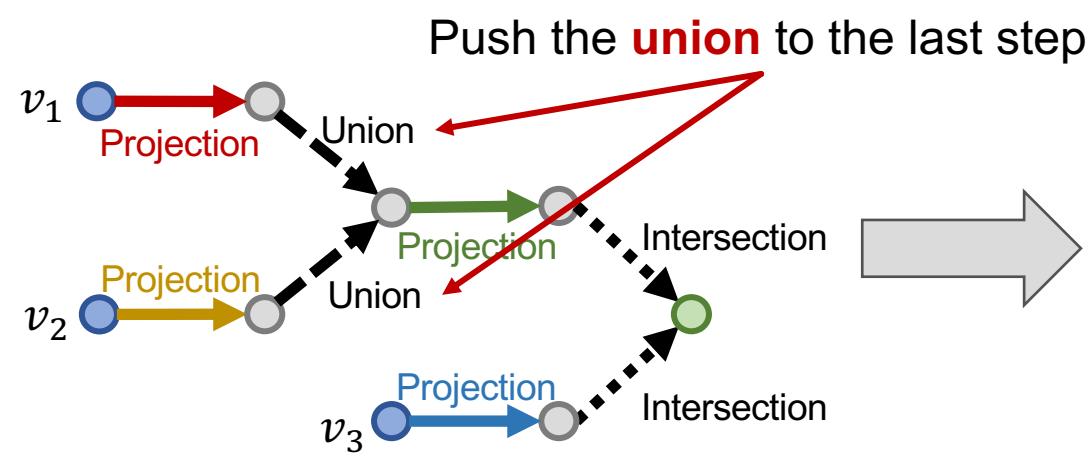
- **Conclusion:** Given any  $M$  conjunctive queries  $q_1, \dots, q_M$  with non-overlapping answers, we need dimensionality of  $\Theta(M)$  to handle all OR queries.
  - For real-world KG, such as FB15k, we find  $M \geq 13,365$ , where  $|V| = 14,951$ .
  - Remember, this is for arbitrary OR queries.

# Embedding AND-OR Queries (3)

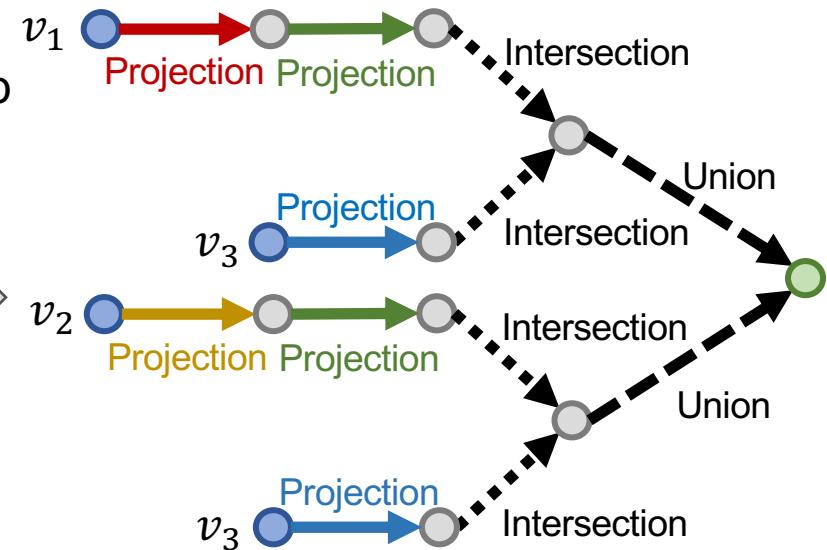
Since we **cannot embed** AND-OR queries in low-dimensional space, can we still handle them?

- Key idea: take all unions out and only do union **at the last step!**

Original Query Plan



Converted Query Plan



# Disjunctive Normal Form

- Any **AND-OR query** can be transformed into equivalent DNF, i.e., **disjunction of conjunctive queries**.

- Given any AND-OR query  $q$ ,

$$q = q_1 \vee q_2 \vee \cdots \vee q_m$$

where  $q_i$  is a **conjunctive query**.

- 
- Now we can first embed all  $q_i$  and then “**aggregate**” at the last step!

# Distance Between $q$ and an Entity

- **Distance** between entity embedding and a DNF  $q = q_1 \vee q_2 \vee \dots \vee q_m$  is defined as:

$$d_{box}(\mathbf{q}, \mathbf{v}) = \min(d_{box}(\mathbf{q}_1, \mathbf{v}), \dots, d_{box}(\mathbf{q}_m, \mathbf{v}))$$

- **Intuition:**
  - As long as  $v$  is the answer to one conjunctive query  $q_i$ , then  $v$  should be the answer to  $q$
  - As long as  $\mathbf{v}$  is close to one conjunctive query  $\mathbf{q}_i$ , then  $\mathbf{v}$  should be close to  $\mathbf{q}$  **in the embedding space**

# Distance Between $q$ and an Entity

- **Distance** between entity embedding and a DNF

$q = q_1 \vee q_2 \vee \dots \vee q_m$  is defined as:

$$d_{box}(\mathbf{q}, \mathbf{v}) = \min(d_{box}(\mathbf{q}_1, \mathbf{v}), \dots, d_{box}(\mathbf{q}_m, \mathbf{v}))$$

- **The process of embedding any AND-OR query  $q$**

1. Transform  $q$  to **equivalent DNF**  $q_1 \vee \dots \vee q_m$
2. **Embed**  $q_1$  to  $q_m$
3. Calculate the (box) distance  $d_{box}(\mathbf{q}_i, \mathbf{v})$
4. Take the **minimum** of all distance
5. **The final score**  $f_q(v) = -d_{box}(\mathbf{q}, \mathbf{v})$

# Stanford CS224W: How to Train Query2box

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

<http://cs224w.stanford.edu>



# Training Overview

- **Overview and Intuition** (similar to KG completion):
  - Given a query embedding  $\mathbf{q}$ , maximize the score  $f_q(v)$  for answers  $v \in \llbracket q \rrbracket$  and minimize the score  $f_q(v')$  for negative answers  $v' \notin \llbracket q \rrbracket$
- **Trainable parameters:**
  - Entity embeddings with  $d|V|$  # params
  - Relation embeddings with  $2d|R|$  # params
  - Intersection operator
- **How to achieve a query, its answers, its negative answers from the KG to train the parameters?**
- **How to split the KG for query answering?**

# Training

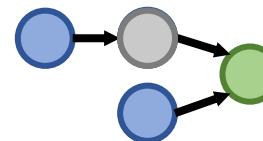
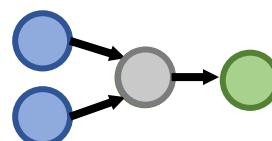
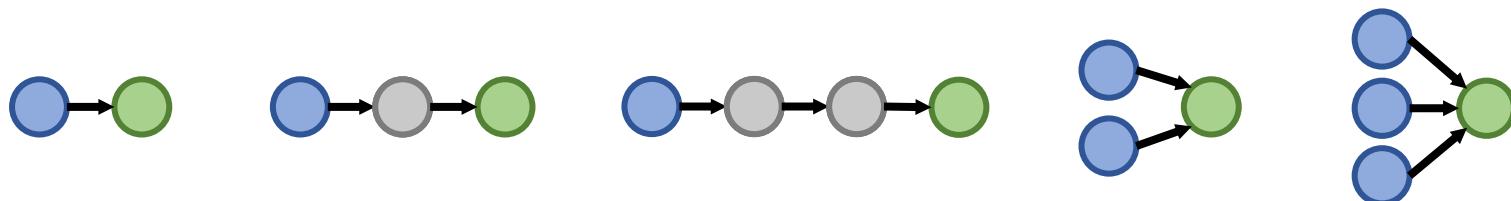
## ■ Training:

1. Randomly sample a query  $q$  from the training graph  $G_{train}$ , answer  $v \in \llbracket q \rrbracket_{G_{train}}$ , and a negative sample  $v' \notin \llbracket q \rrbracket_{G_{train}}$ .
  - Negative sample: Entity of same type as  $v$  but not answer.
2. Embed the query  $\mathbf{q}$ .
3. Calculate the score  $f_q(v)$  and  $f_q(v')$ .
4. Optimize the loss  $\ell$  to maximize  $f_q(v)$  while minimize  $f_q(v')$ :

$$\ell = -\log \sigma(f_q(v)) - \log(1 - \sigma(f_q(v')))$$

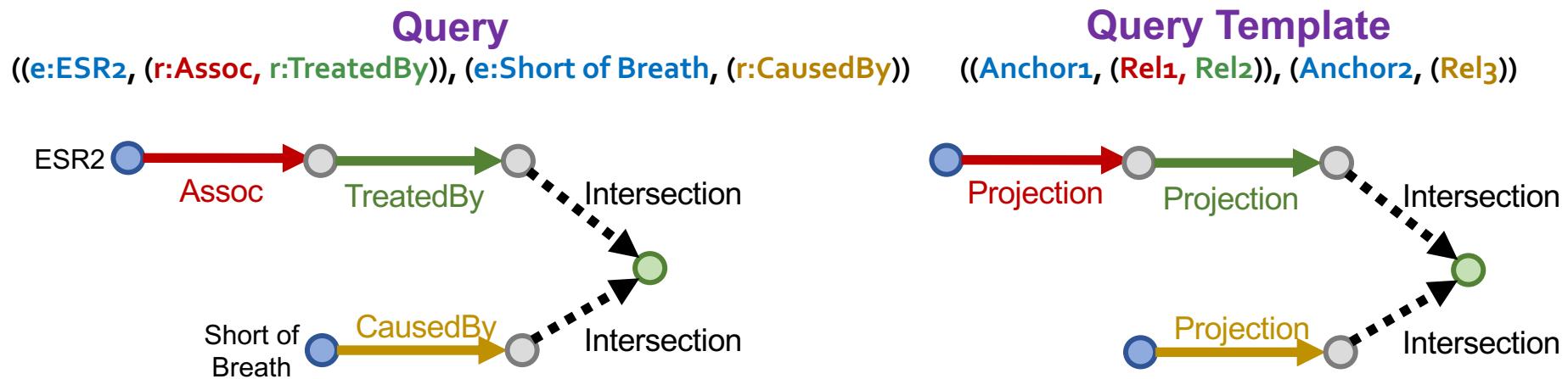
# Query Generation from Templates

- Generate queries from multiple query templates:



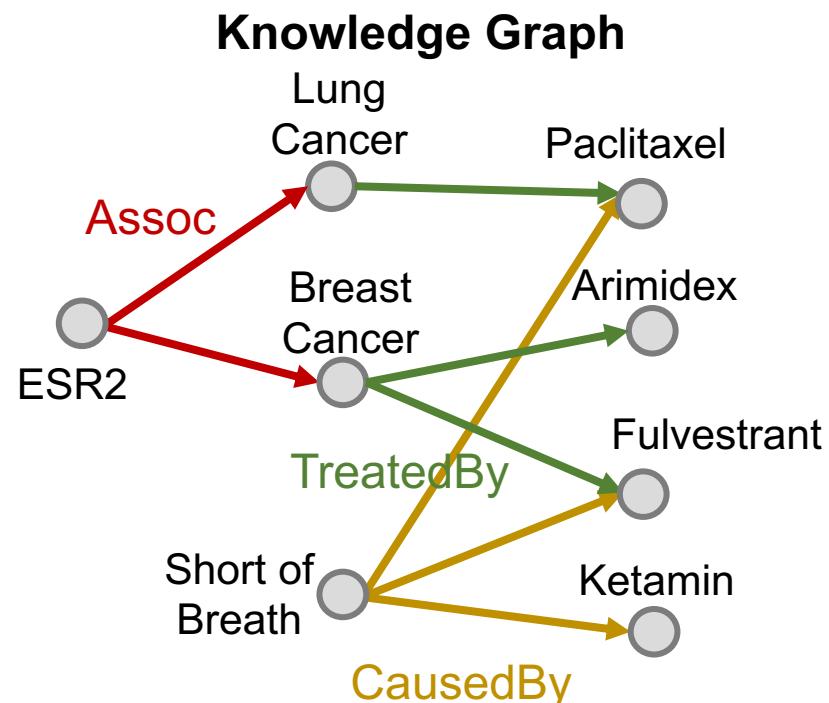
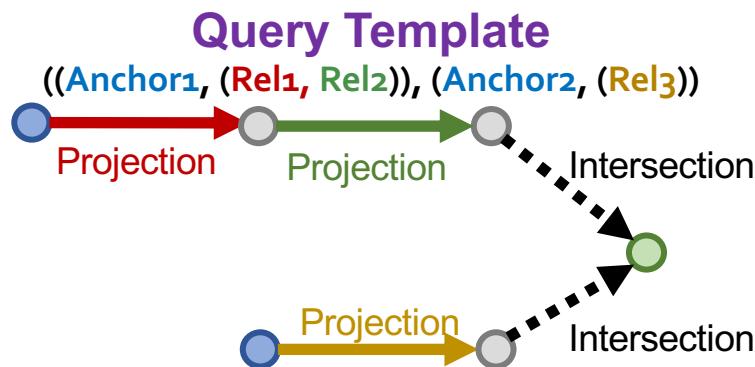
# Query Generation from Templates

- How can we generate a **complex query**?
- We start with a **query template**
- **Query template** can be viewed as an abstraction of the query
- We generate a query by instantiating every variable with a concrete entity and relation from the KG
  - E.g., instantiate **Anchor1** with **ESR2** (a node on KG)
  - E.g., instantiate **Rel1** with **Assoc** (an edge on KG)
- **How to instantiate query template given a KG?**



# Query Generation from Templates

## ■ How to instantiate a query template given a KG?

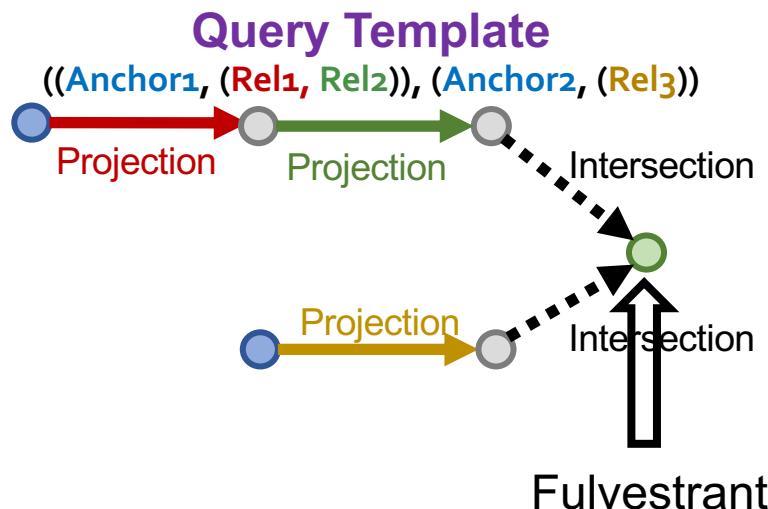


### Overview:

Start from instantiating the **answer node** of the query template and then iteratively instantiate the other edges and nodes until we ground **all the anchor nodes**

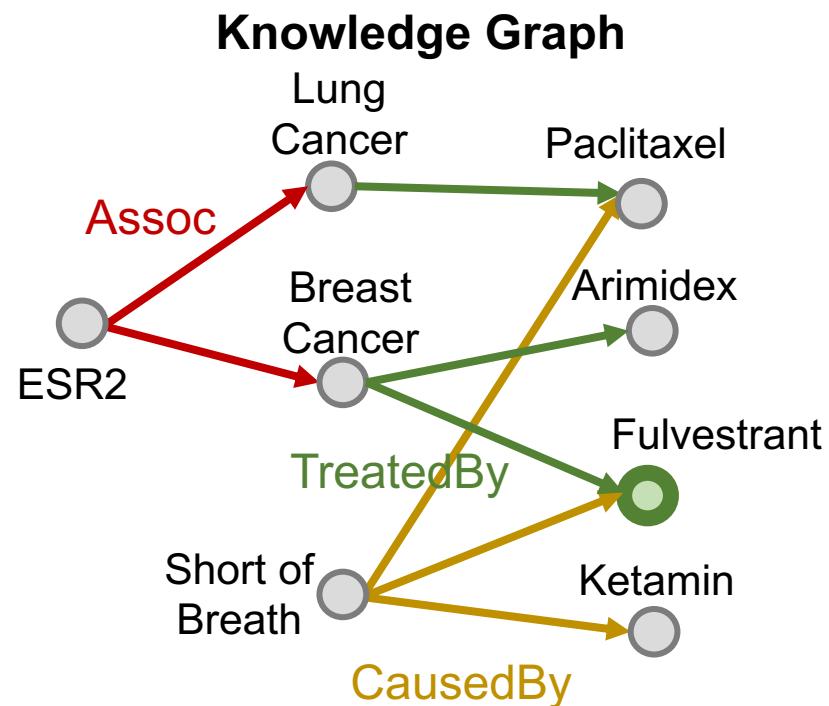
# Query Generation from Templates

## ■ How to instantiate a query template given a KG?



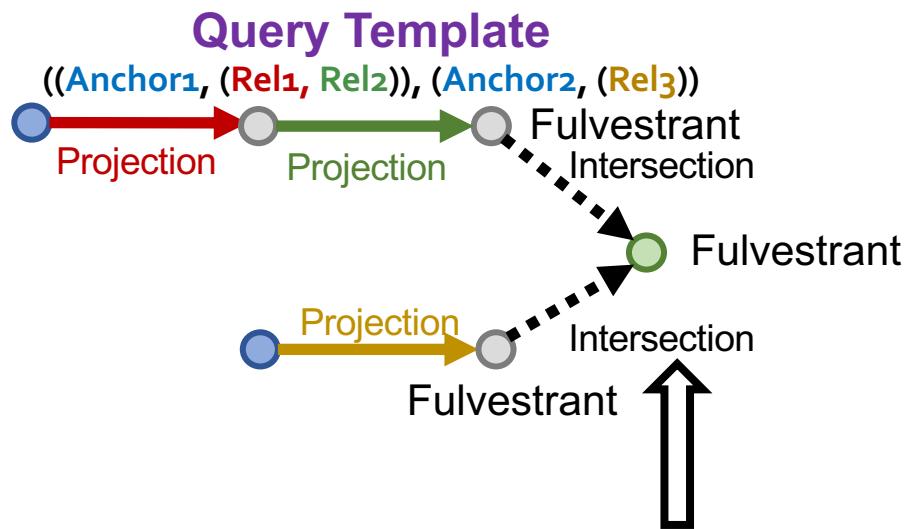
Start from instantiating the **root node** of the query template.

Randomly pick one entity from KG as the root node, e.g., we pick **Fulvestrant**.

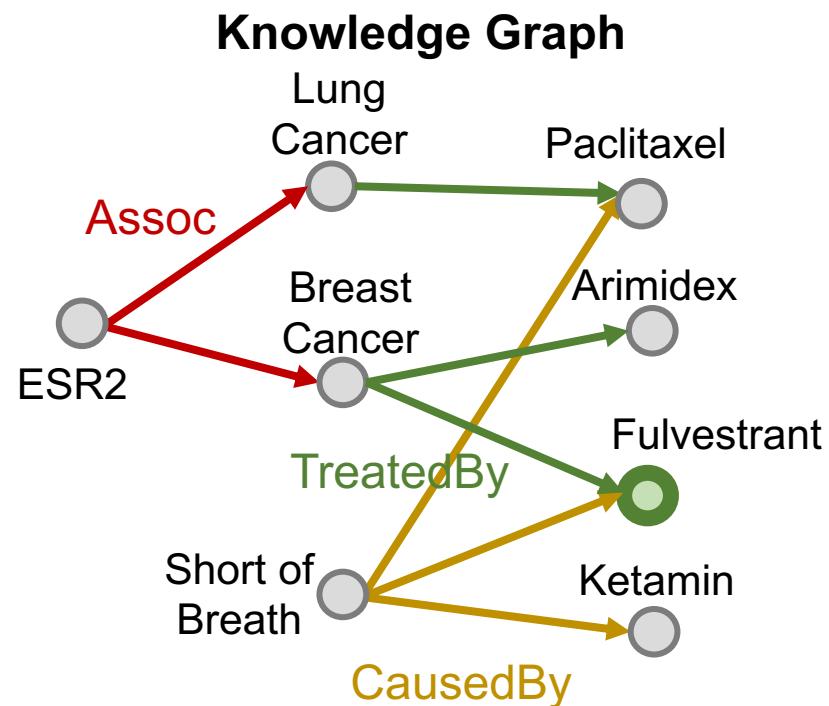


# Query Generation from Templates

- ## ■ How to instantiate a query template given a KG?

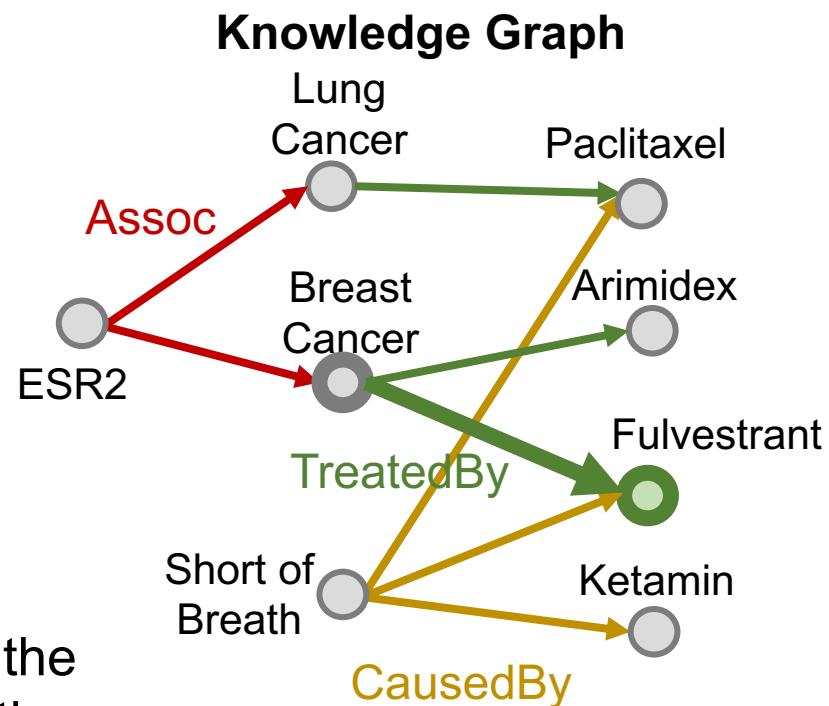
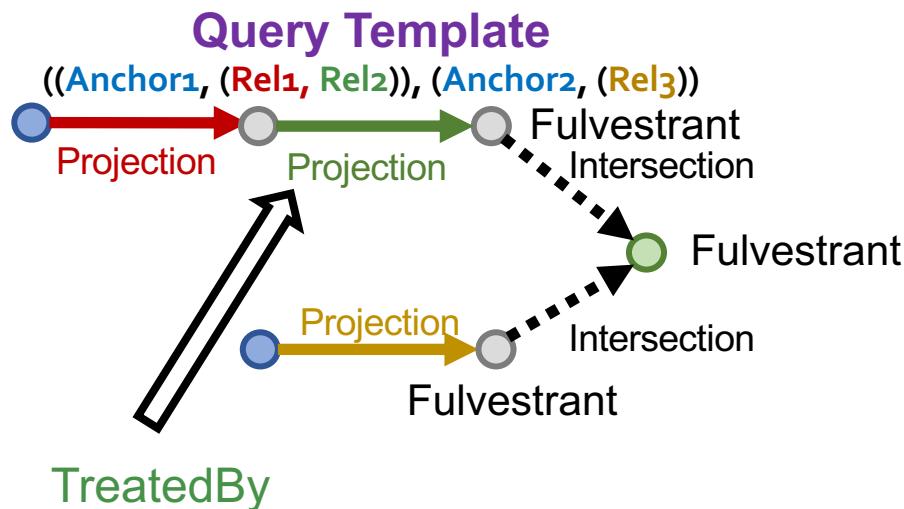


Now we look at intersection.  
What we have is that the  
intersection of the sets of entities  
is **Fulvestrant**, then naturally the  
two sets should also contain  
**Fulvestrant**.



# Query Generation from Templates

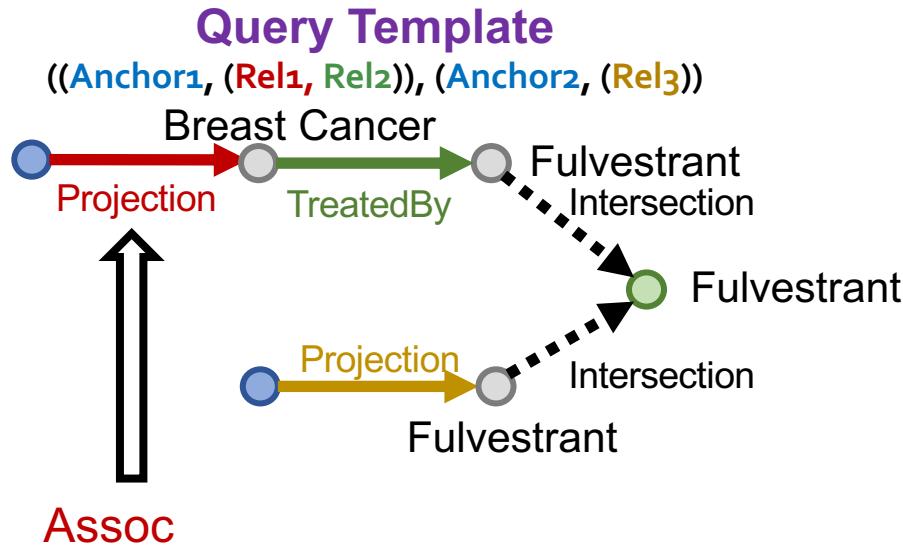
- ## ■ How to instantiate a query template given a KG?



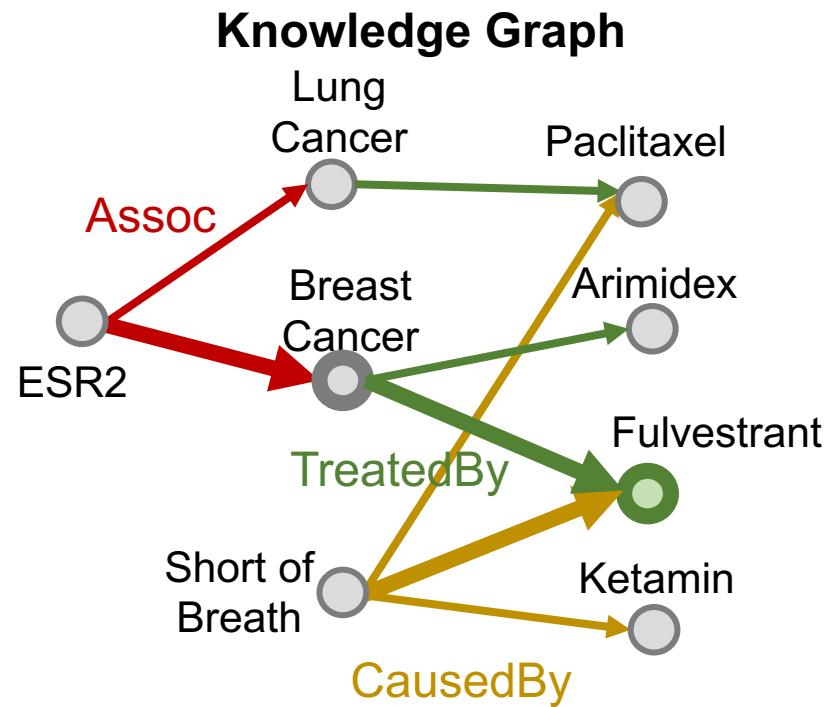
We instantiate the **Projection edge** in the template by randomly sample one relation associated with the current entity **Fulvestrant**. For example, we may select relation **TreatedBy**, and check what entities are connected to **Fulvestrant** with **TreatedBy**: {**Breast Cancer**}.

# Query Generation from Templates

## ■ How to instantiate a query template given a KG?

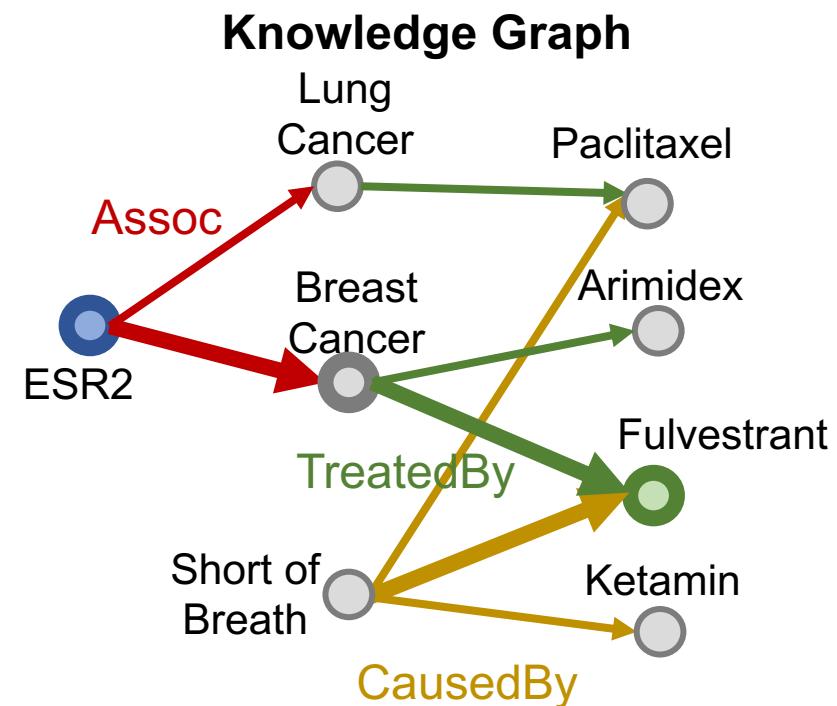
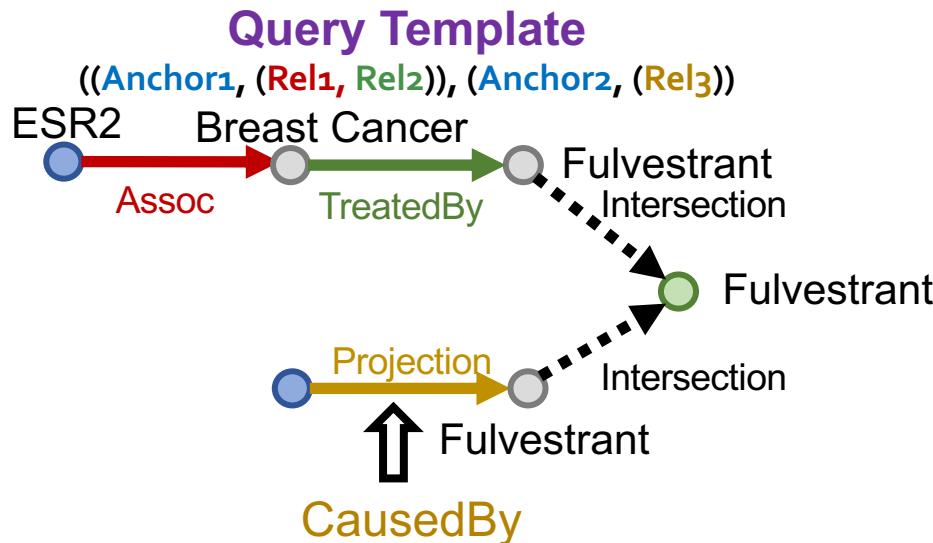


We first look at one branch and ground the **Projection edge** with the relation associated with **Breast Cancer**, e.g., **Assoc**. Then we check what entities are connected to **Breast Cancer** with **Assoc**: **{ESR2}**.



# Query Generation from Templates

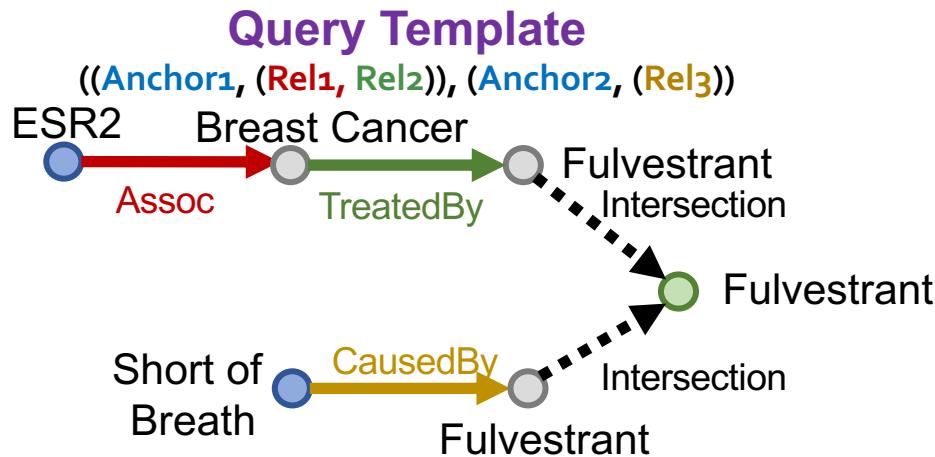
## ■ How to instantiate a query template given a KG?



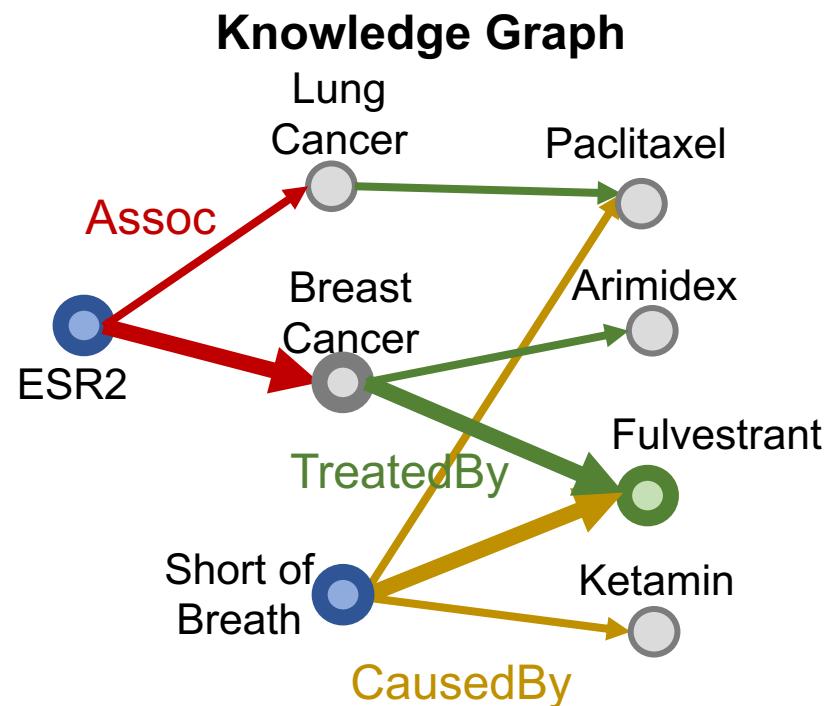
Then we look at the second branch and ground the **Projection edge** with the relation associated with **Fulvestrant**, e.g., **CausedBy**. Then we check what entities are connected to **Fulvestrant** with **CausedBy**: {**Short of Breath**}.

# Query Generation from Templates

## ■ How to instantiate a query template given a KG?

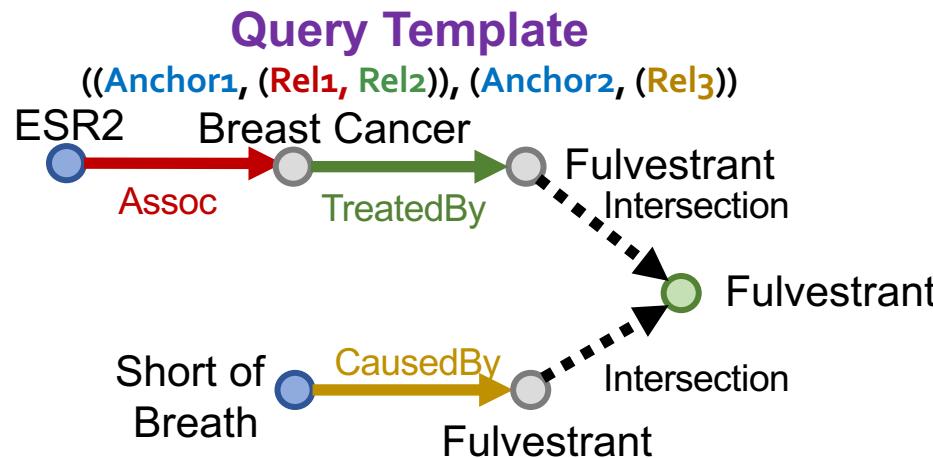


We select entity from {**Short of Breath**}, set it as the anchor node.



# Query Generation from Templates

- How to instantiate a query template given a KG?



Now, we instantiated a **query  $q$** !

$q: ((e: \text{ESR2}, (r: \text{Assoc}, r: \text{TreatedBy})), (e: \text{Short of Breath}, (r: \text{CausedBy})))$

- The query  $q$  **must** have answers on the KG and one of the answers is the instantiated answer node: **Fulvestrant**.
- We may obtain the full set of answers  $\llbracket q \rrbracket_G$  by **KG traversal**.
- We can sample negative answers  $v' \notin \llbracket q \rrbracket_G$

# Stanford CS224W: Example of Query2box

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

<http://cs224w.stanford.edu>



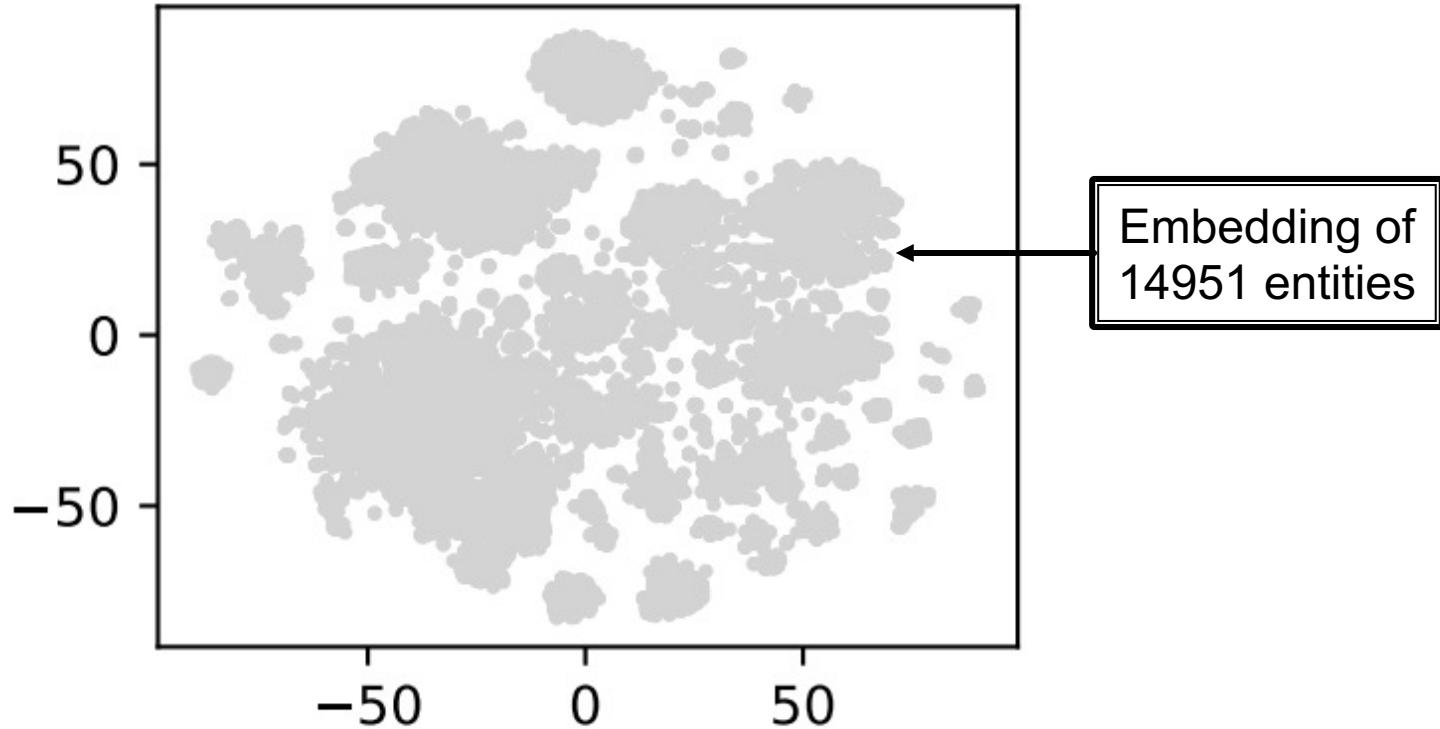
# Visualization

- What do box embeddings actually learn?

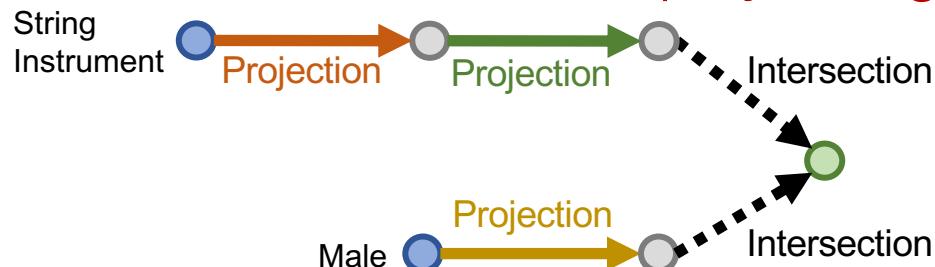
Example: “*List male instrumentalists who play string instruments*”

- We use t-SNE to reduce the embedding space to a 2-dimensional space, in order to **visualize the query results**

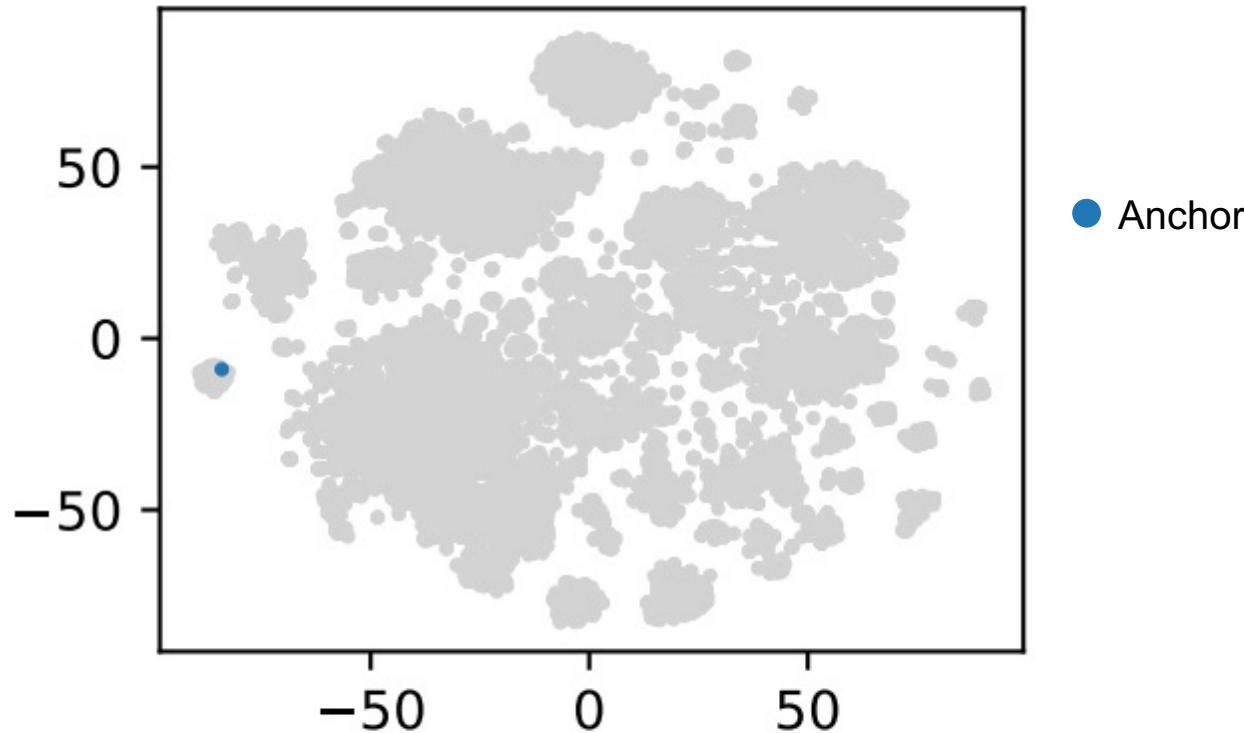
# Embedding Space



“List male instrumentalists who play string instruments”



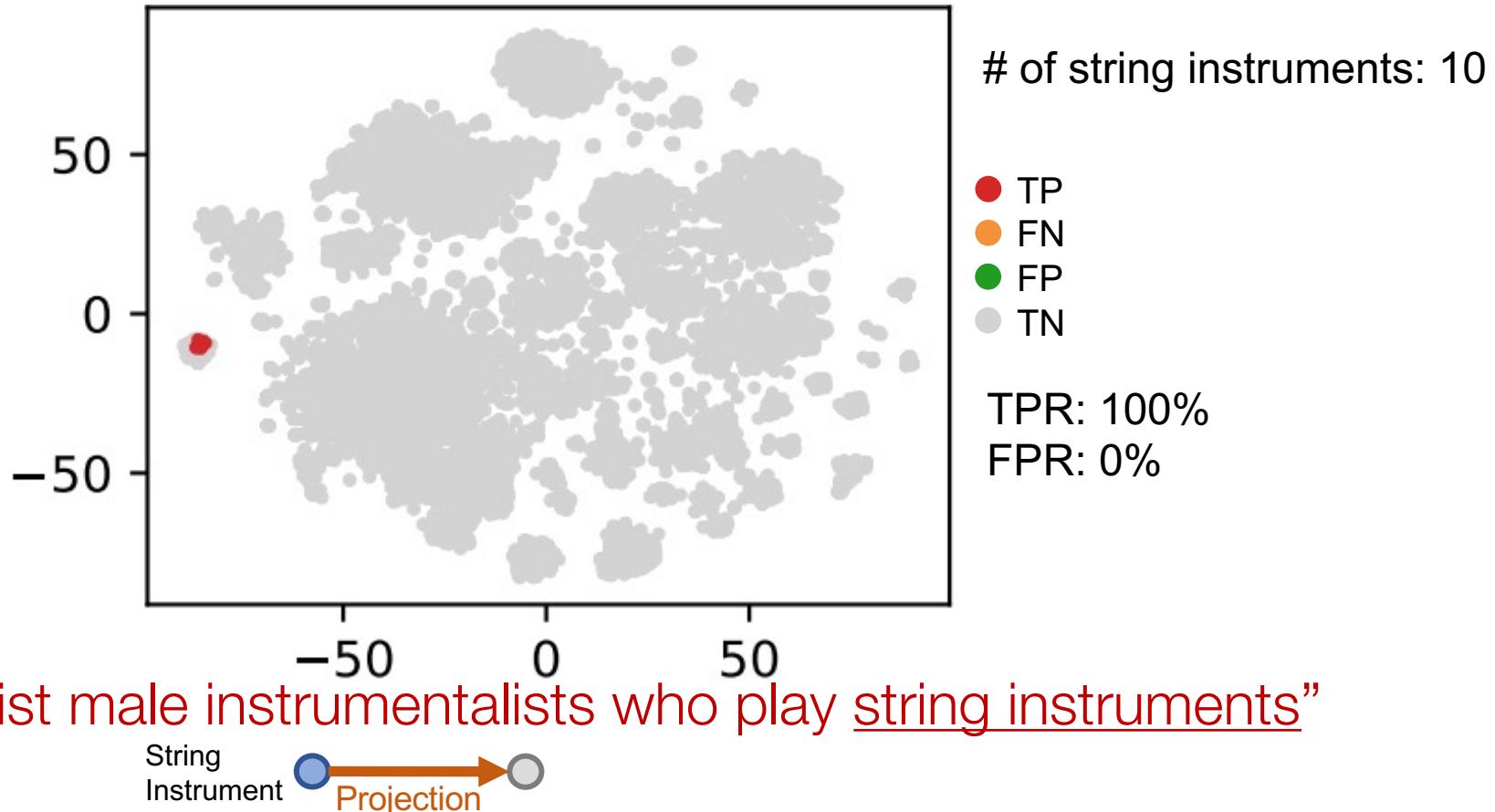
# Embedding Space



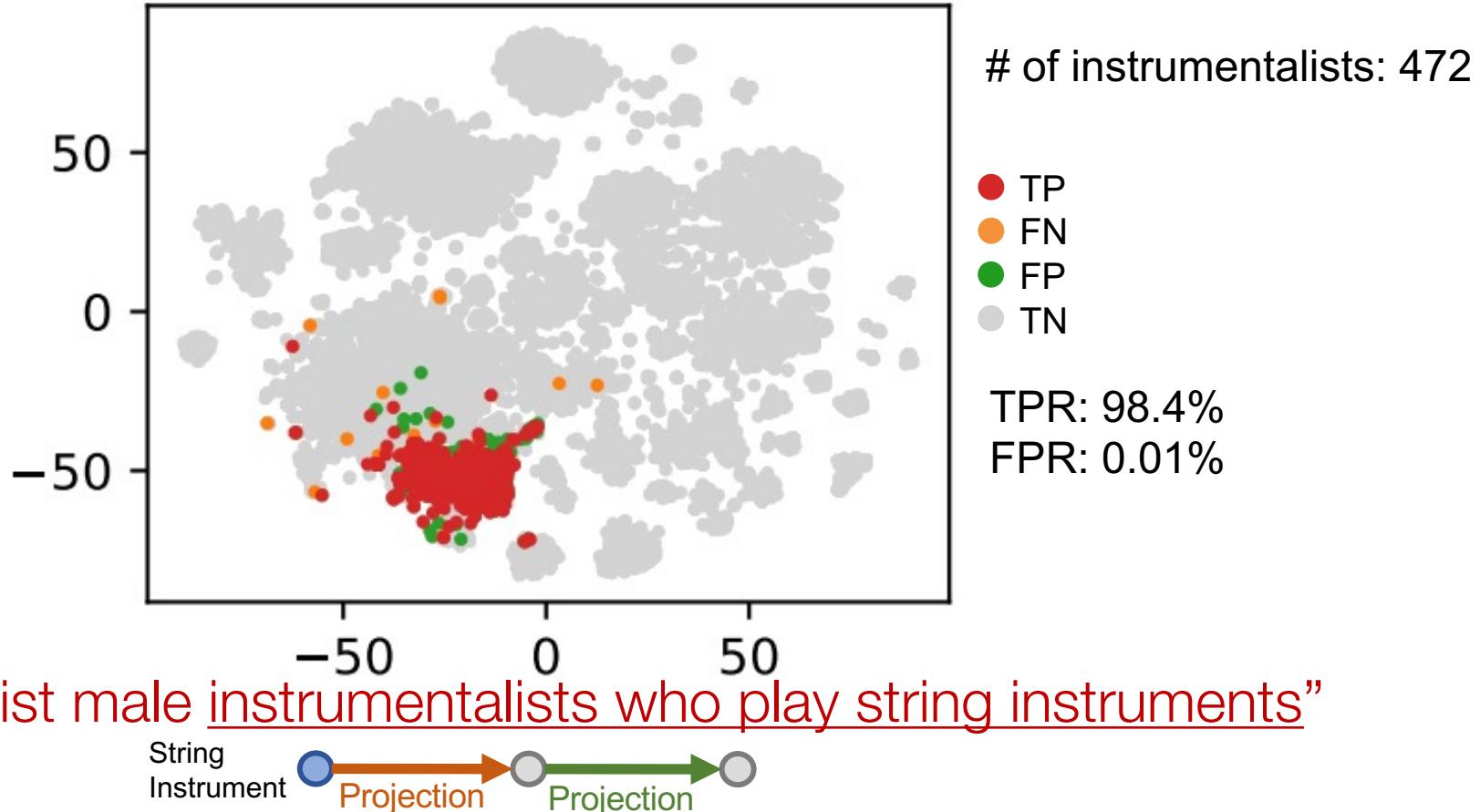
“List male instrumentalists who play string instruments”

String  
Instrument

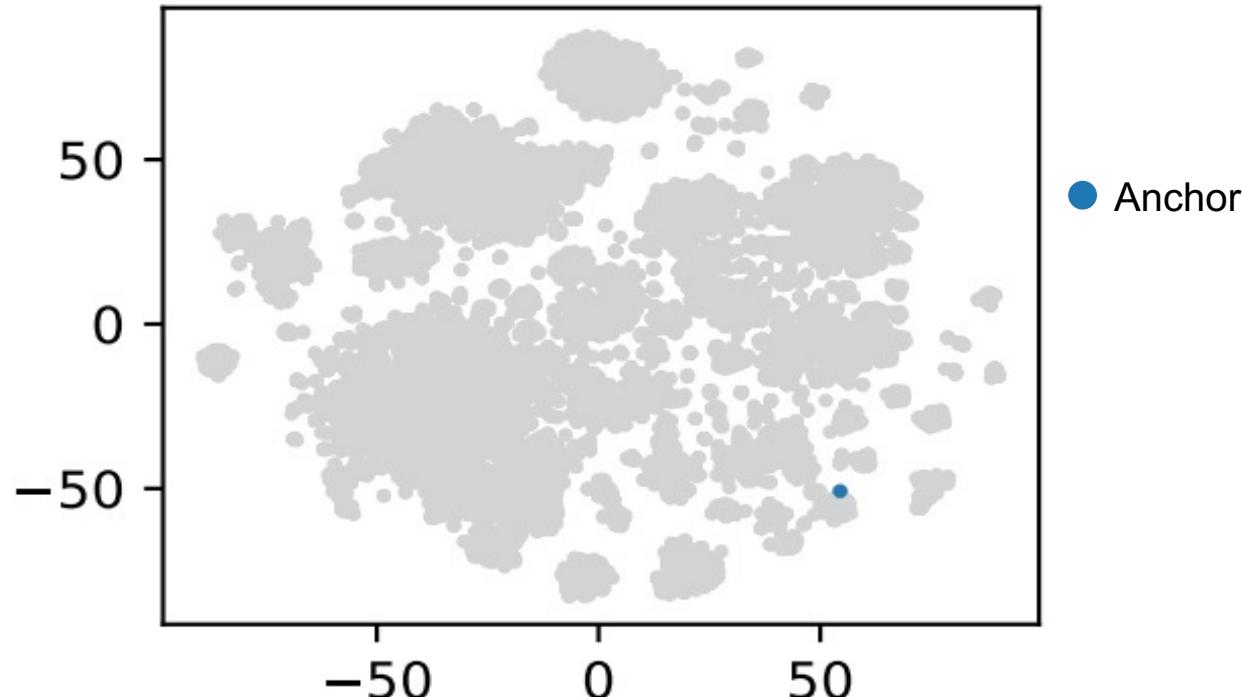
# Embedding Space



# Embedding Space



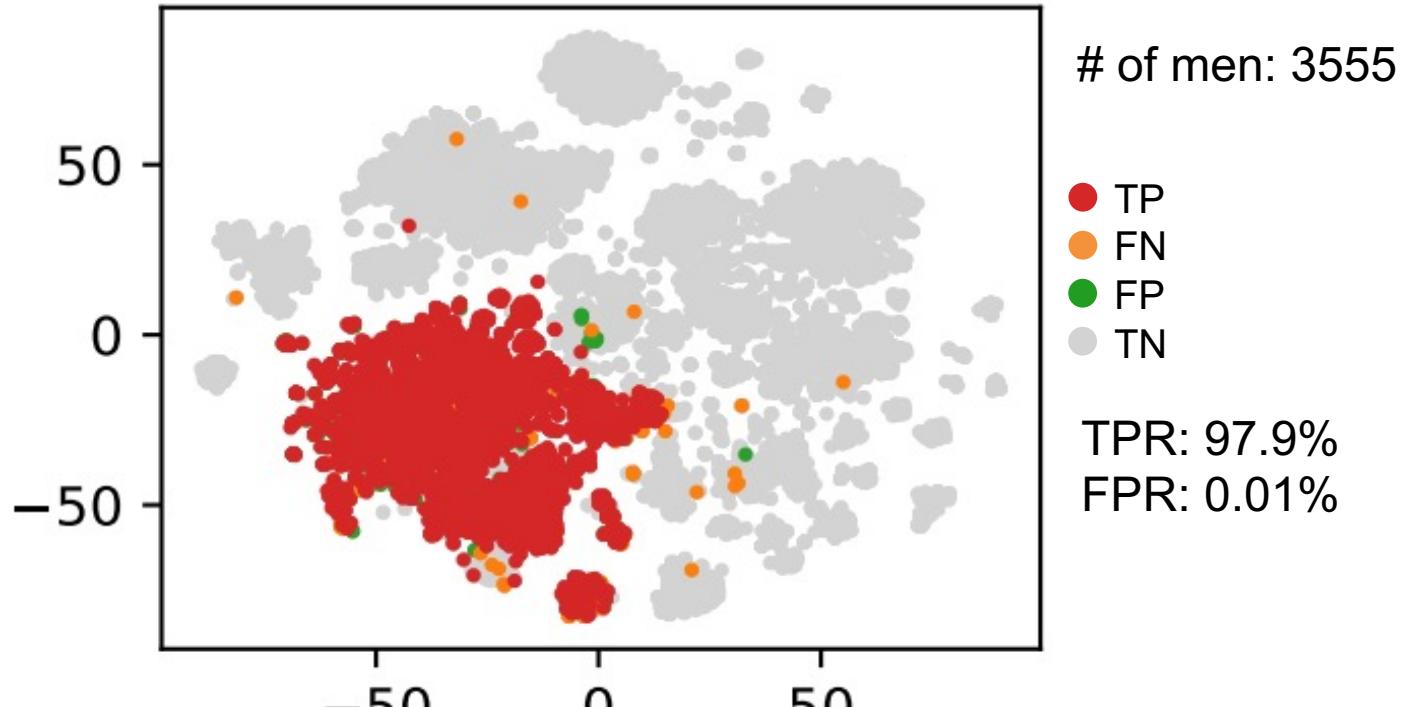
# Embedding Space



“List male instrumentalists who play string instruments”

Male

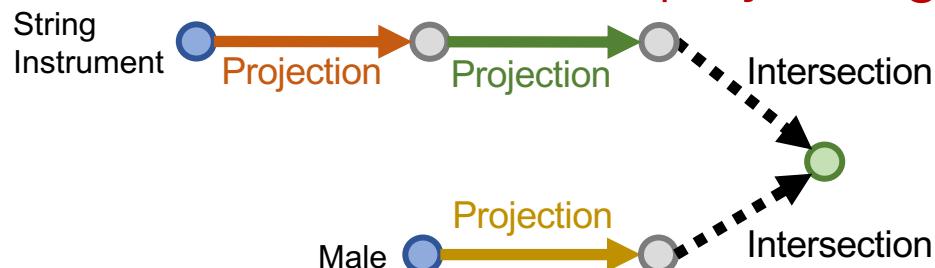
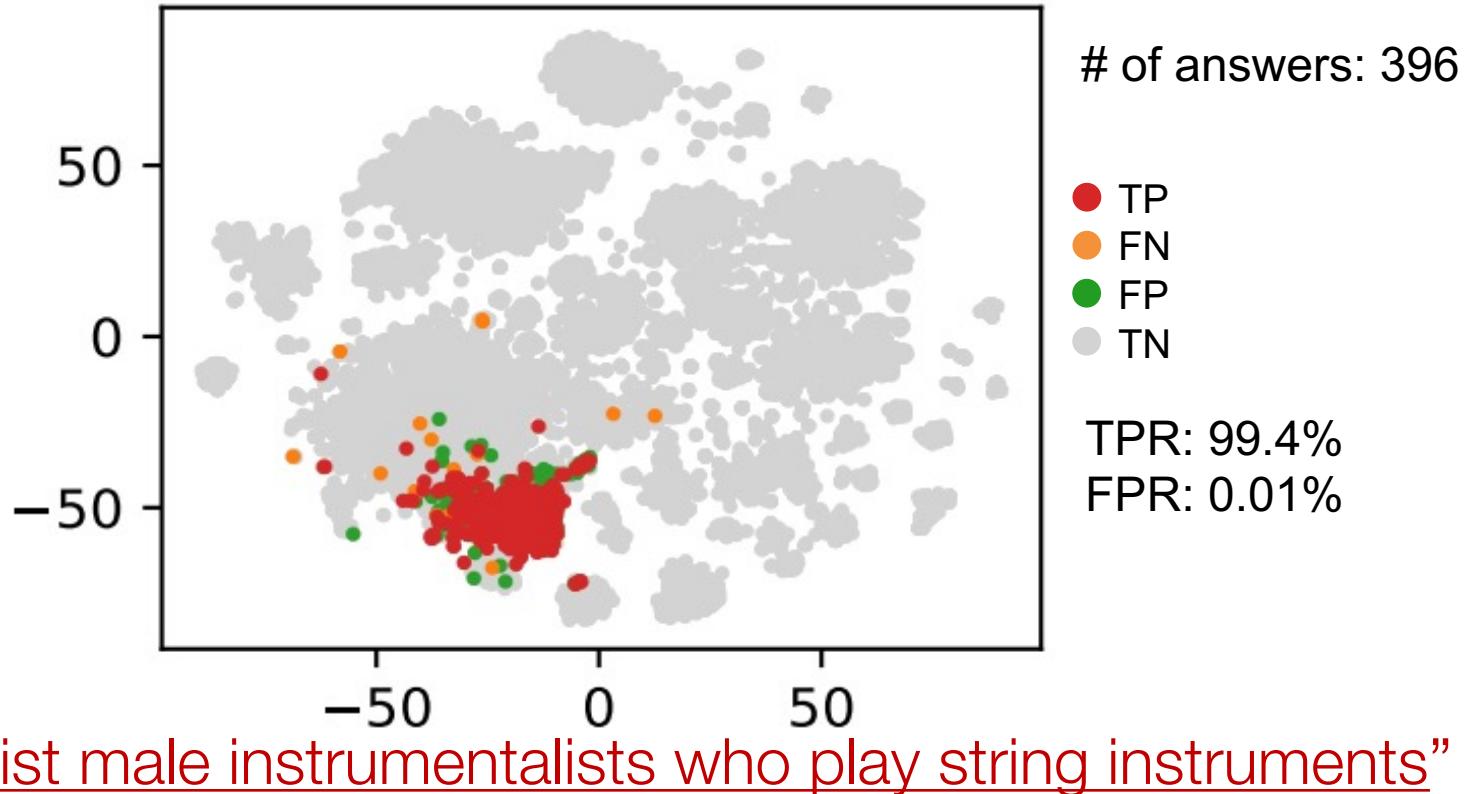
# Embedding Space



“List male instrumentalists who play string instruments”



# Embedding Space



# Summary

- We introduce answering predictive queries on large knowledge graphs.
- The key idea is to embed queries by navigating the embedding space!
  - We embed the query by composing learned operators
  - Embedding of the query is close to its answers in the embedding space