Yale

Logical Query Reasoning in Knowledge Graphs

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

Readings

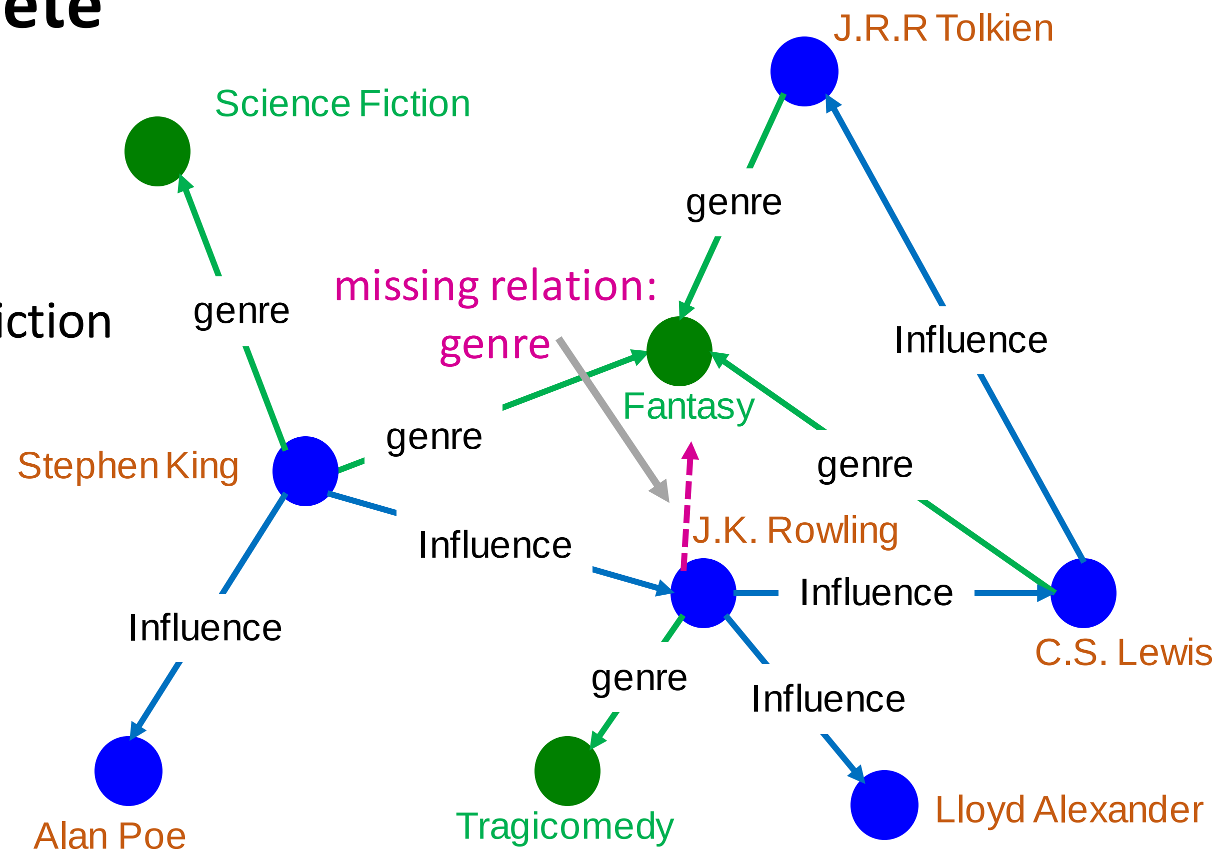
- Readings are updated on the website (syllabus page)
- **Lecture 20 readings:**
  - Neural Distance Embeddings
  - Hyperbolic Cone Embedding
- **Lecture 21 readings:**
  - Query2Box
  - Google PathQuery
- NeurIPS 2022 Workshop (Frontiers of Graph Learning)

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 "Fantasy" for ("J.K. Rowling", "genre")



- **Today**: **multi-hop** reasoning over KG for complex queries

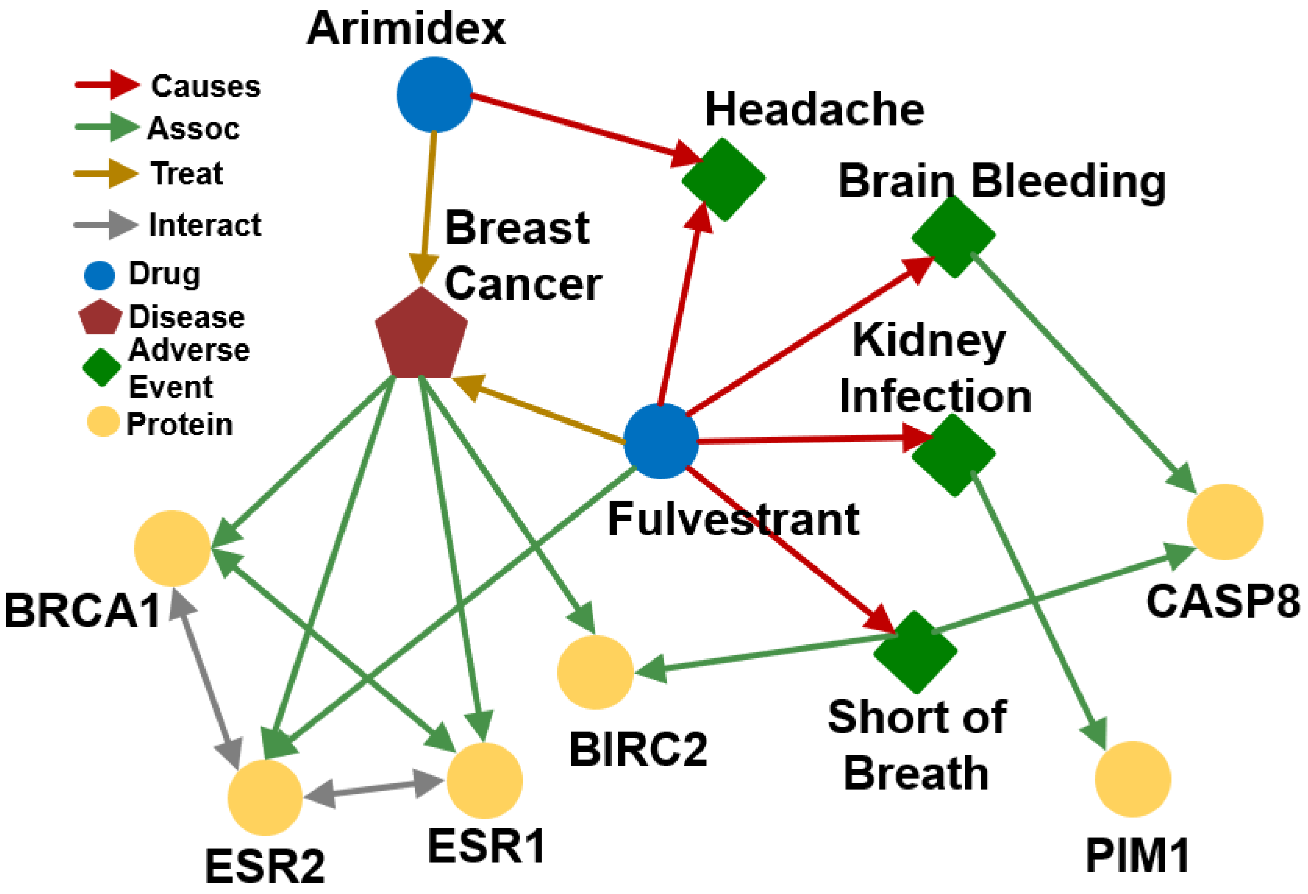
Outline of Today's Lecture

- **Queries on KG**
- **Traversing KG in Embedding Space**
- **Box Embeddings**

Outline of Today's Lecture

- **Queries on KG**
- **Traversing KG in Embedding Space**
- **Box Embeddings**

Example KG: Biomedicine



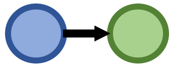
Predictive Queries on KG

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

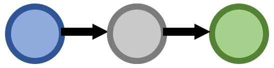
Query Types	Examples: Natural Language Question, Query
One-hop Queries	What adverse event is caused by Fulvestrant? (e:Fulvestrant, (r:Causes))
Path Queries	What protein is associated with the adverse event caused by Fulvestrant? (e:Fulvestrant, (r:Causes, r:Assoc))
Conjunctive Queries	What is the drug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

Inverse of (r:Treat) Inverse of (r:Cause)

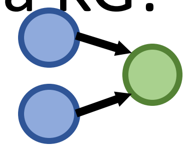
- In this lecture, we only focus on answering **queries** on a KG!



One-hop Queries



Path Queries



Conjunctive Queries

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?

$$\Updownarrow$$

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

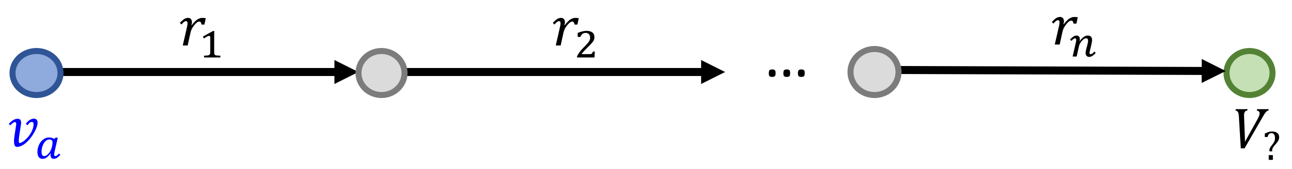
Path Queries

- 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, answers are denoted by $[\![q]\!]_G$.

Query Plan of q:

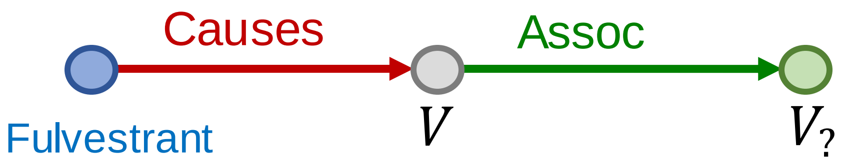


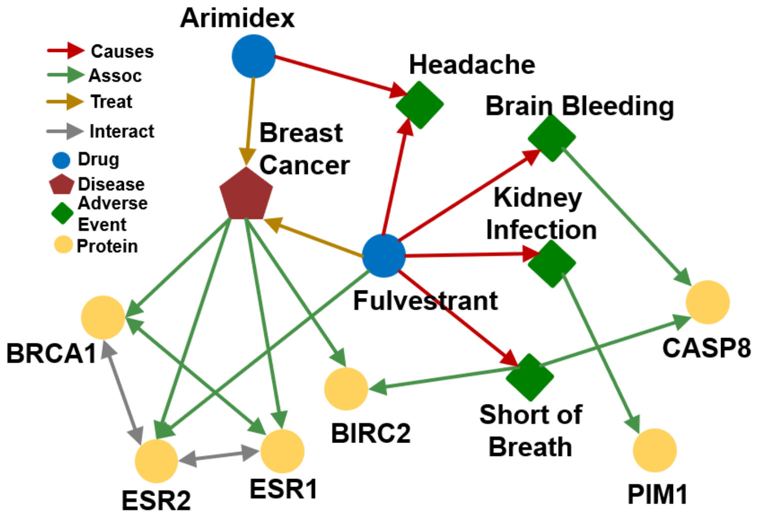
Query plan of path queries is a chain.

Path Queries: Example

"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**))

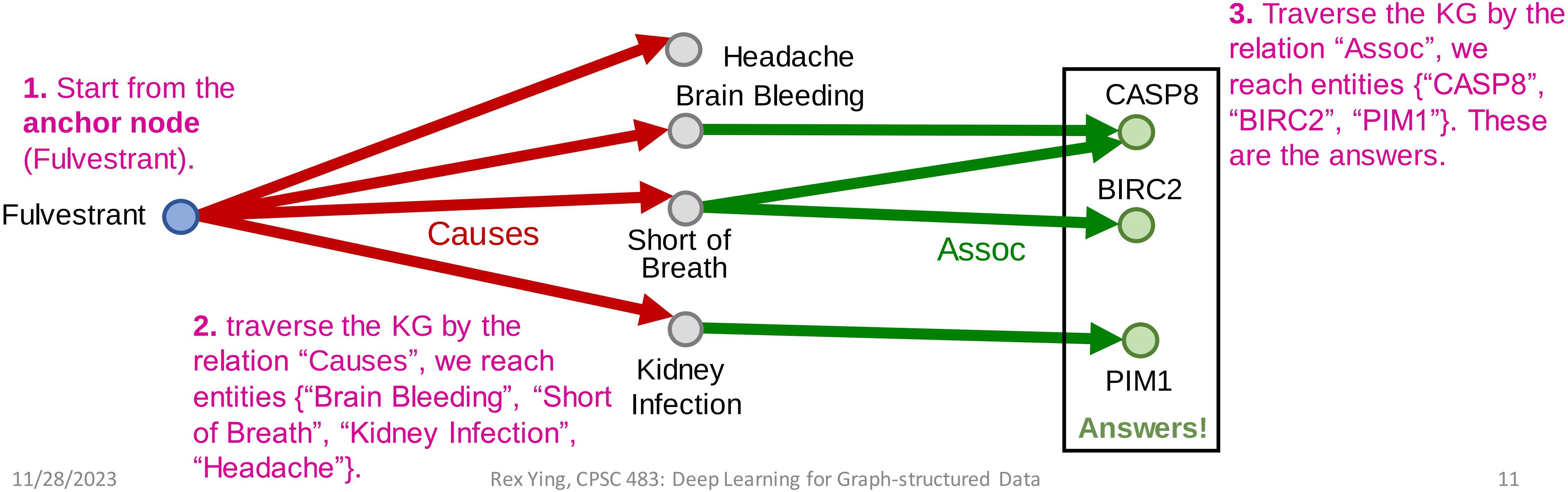




Given a KG, how to answer a path query?

Traversing Knowledge Graphs

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



KGs are incomplete

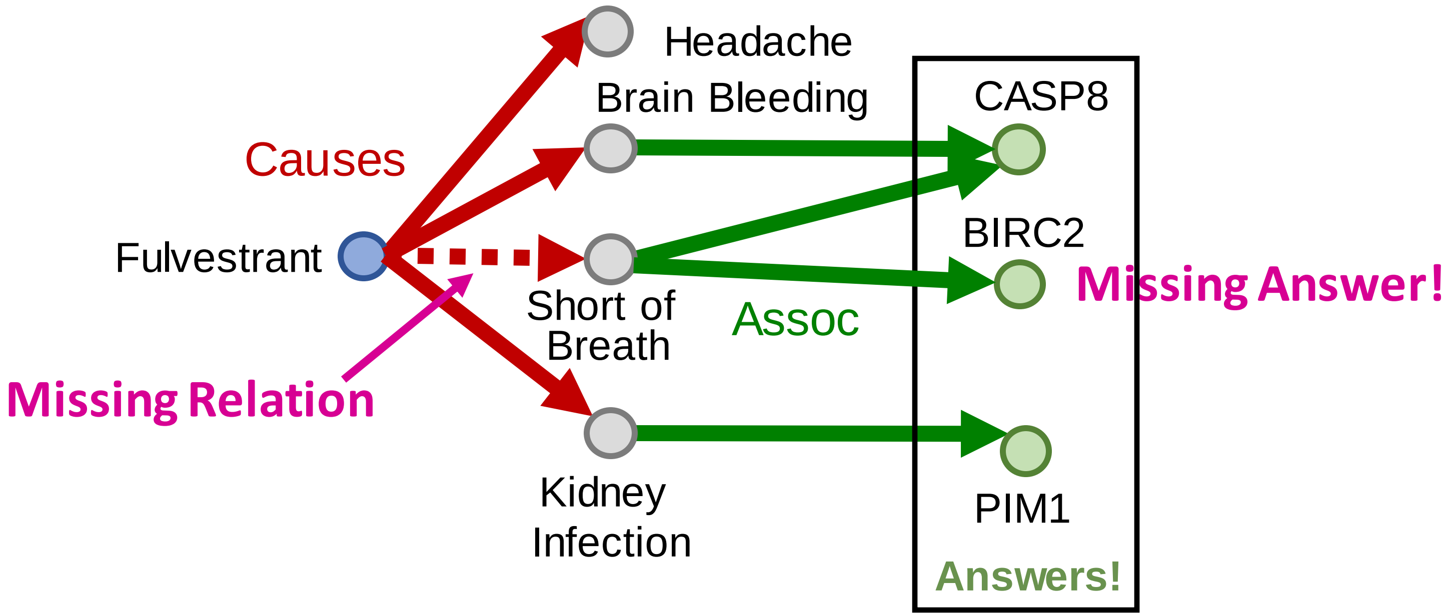
- **Answering queries seems easy: Just traverse the graph.**

- **However, KGs are notoriously incomplete:**
  - 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

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

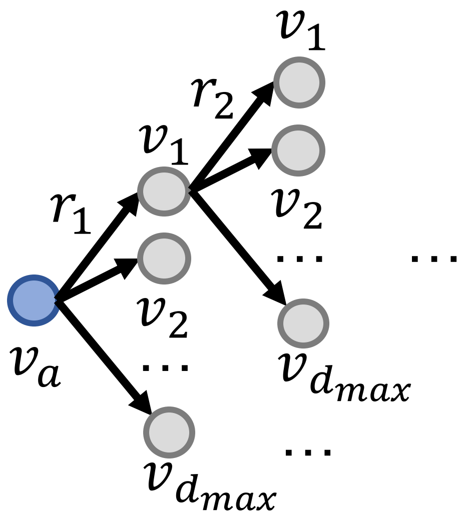


Can KG Completion Help?

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

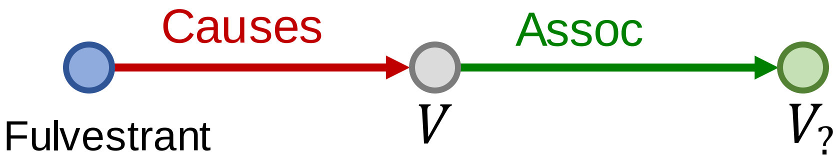
- **It's difficult.** The "completed" KG is a **dense graph**!
  - Most (h, r, t) triples (edge on KG) will have some non-zero probability.

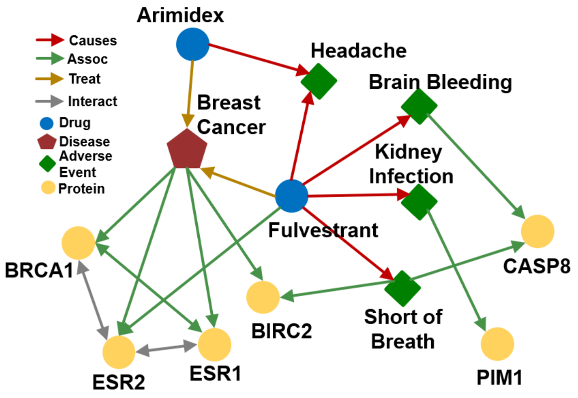
- Time complexity of traversing a dense KG is exponential as a function of the path length L: $O(d_{max}^L)$



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**





Outline of Today’s Lecture

- Queries on KG

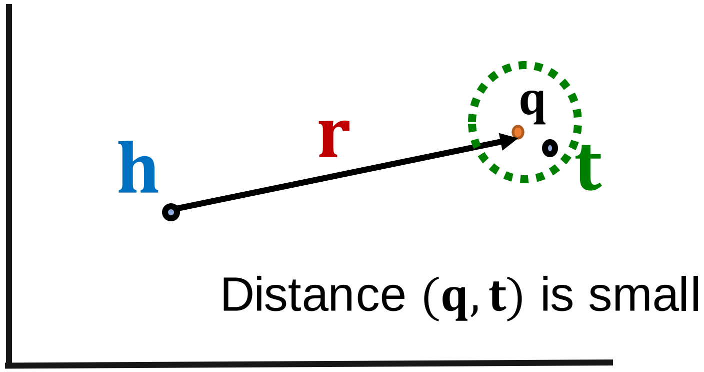
- **Traversing KG in Embedding Space**

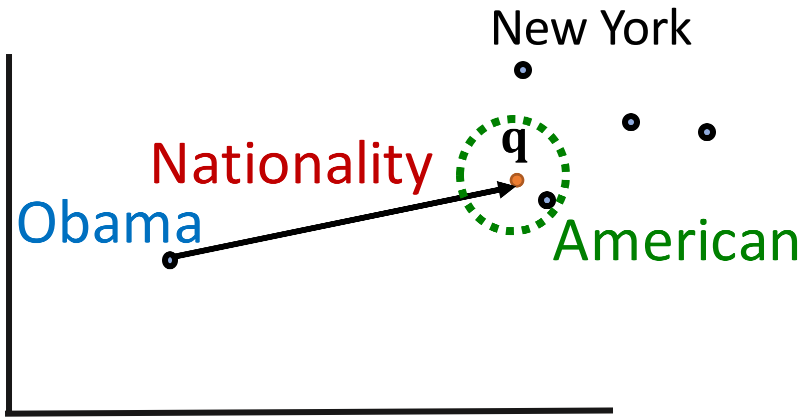
- Box Embeddings

Idea: Traversing KG in Embedding 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 $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$.
  - Another way to interpret this is that:
    - **Query embedding**: $\mathbf{q} = \mathbf{h} + \mathbf{r}$ (**Note** that $\mathbf{q}$ is the **embedding** of q)
    - Goal: **query embedding** $\mathbf{q}$ is **close** to the **answer embedding** $\mathbf{t}$

$$f_q(t) = -\|\mathbf{q} - \mathbf{t}\|$$

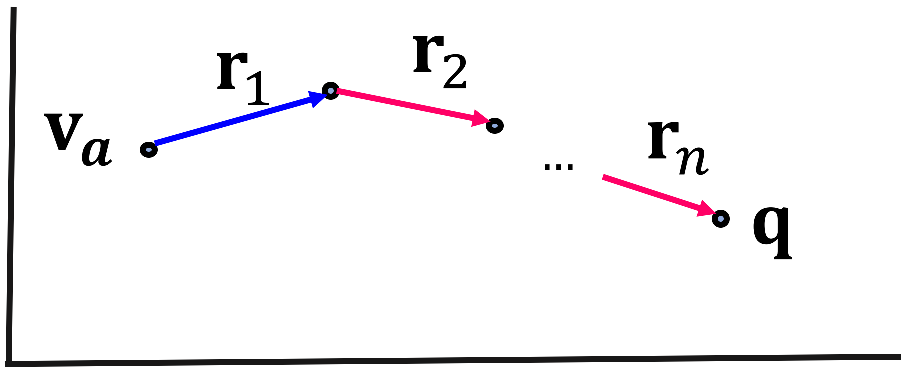




Traversing KG in Embedding Space

- **Key idea: Embed queries!**
  - Generalize **TransE** to multi-hop reasoning.

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



$$\mathbf{q} = \mathbf{v}_a + \mathbf{r}_1 + \cdots + \mathbf{r}_n$$

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

Traversing KG in Embedding Space: Example

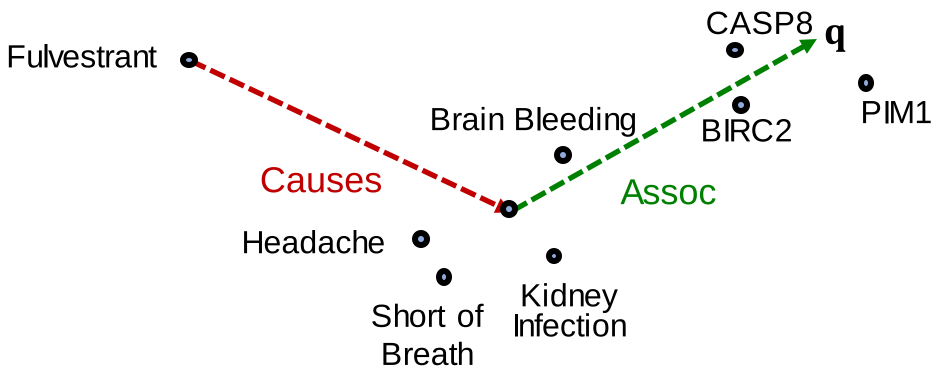
Embed path queries in vector space.

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

Follow the query plan:

Query Plan

Embedding Process



Traversing KG in Embedding Space: Insights

Insights:

- We can train TransE to optimize knowledge graph completion objective (Lecture 18)

- Since TransE can naturally handle composition relations, it can handle path queries by translating in the latent space **for multiple hops using addition of relation embeddings**.

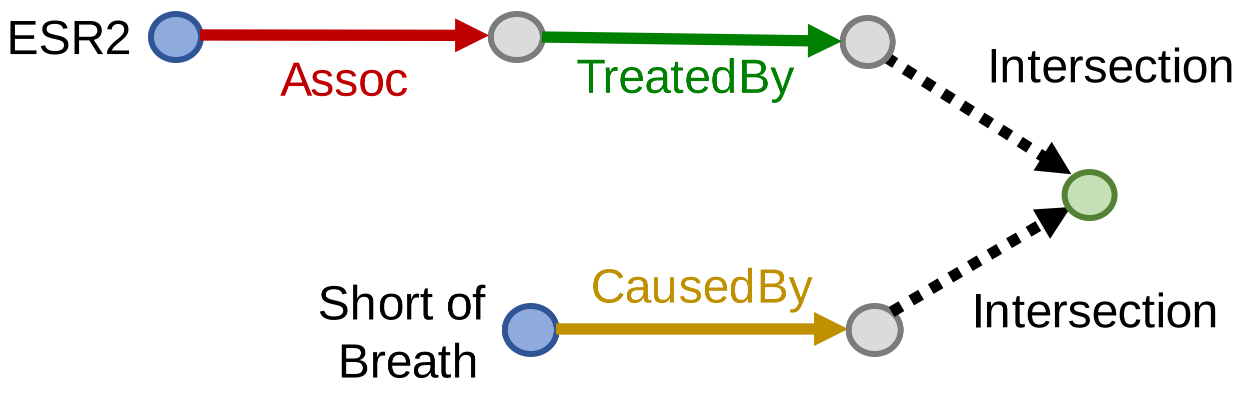
Conjunctive Queries (1)

Can we answer **more complex queries with conjunction (AND) 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:

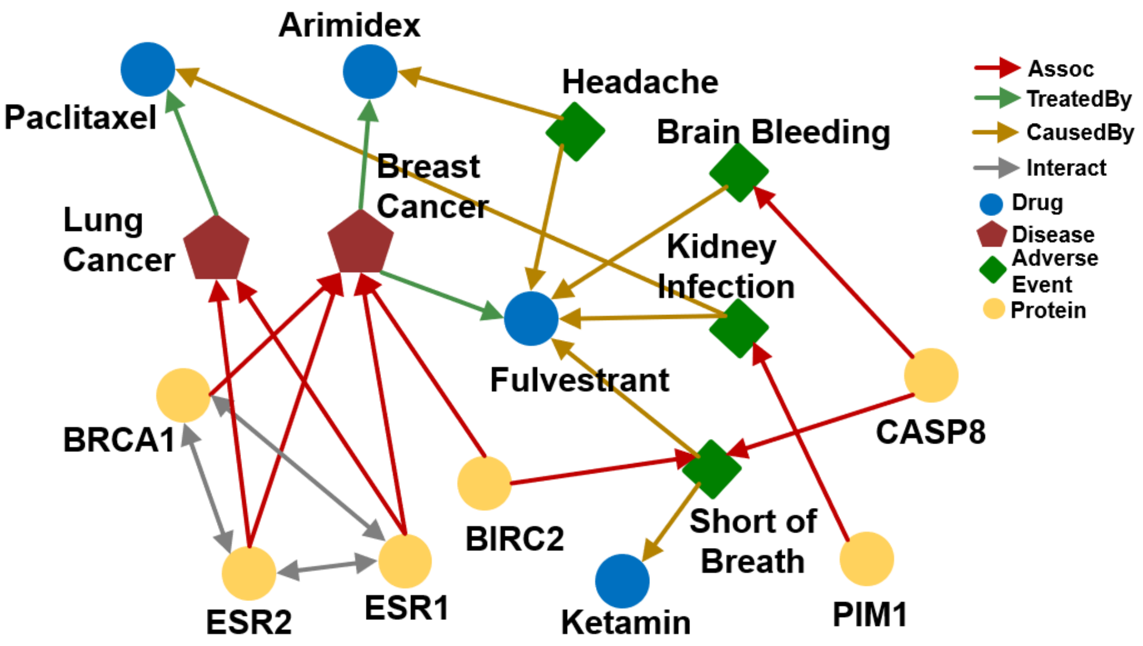


Conjunctive Queries (2)

- *"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?

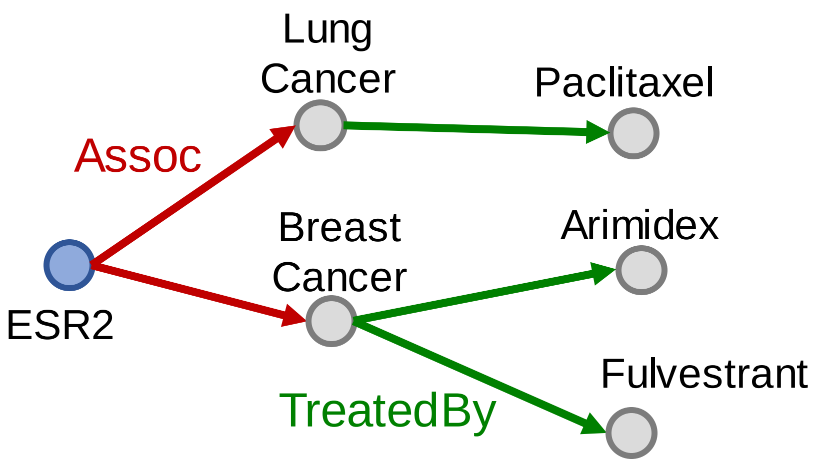


Traversing KG for Conjunctive Queries (1)

- *"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:**



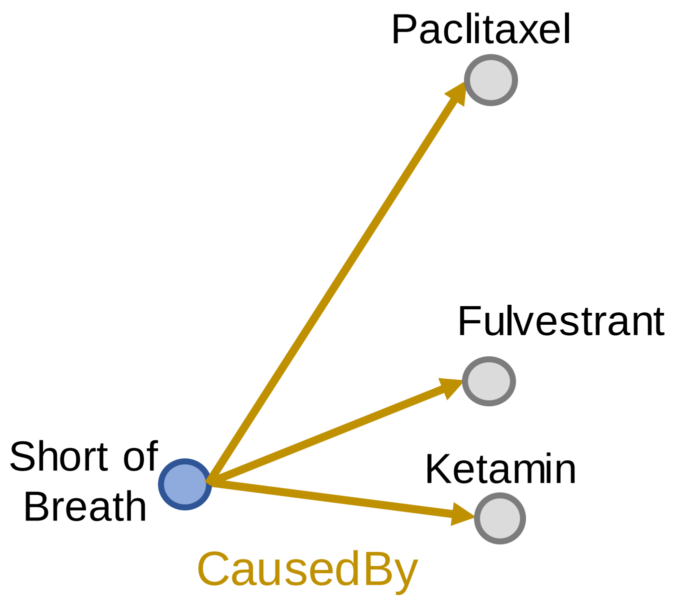
Traversing from a set of entities {"Lung Cancer", "Breast Cancer"} by relation TreatedBy, we achieve a set of entities: {"Paclitaxel", "Arimidex", "Fulvestrant"}

Traversing KG for Conjunctive Queries (2)

- *"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:**



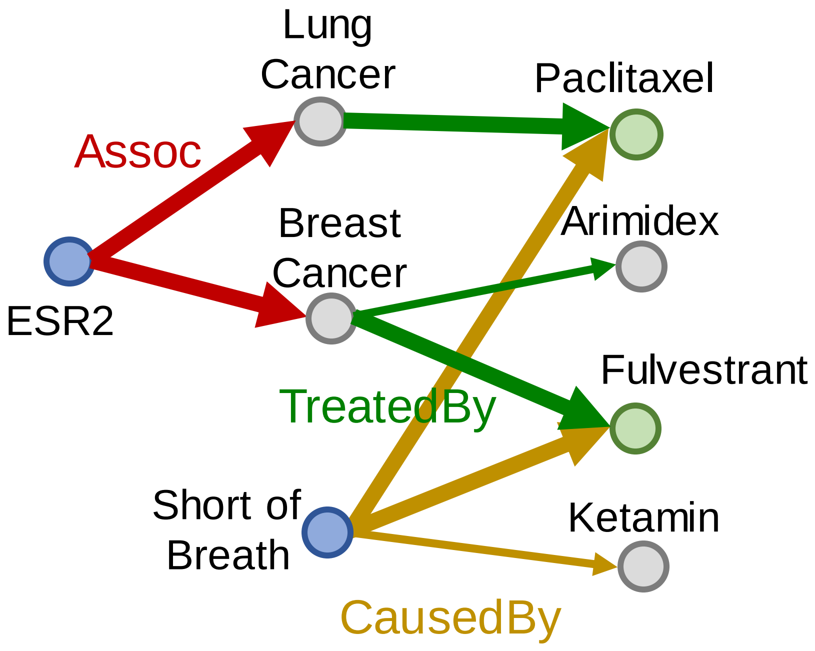
Traverse from the second anchor "Short of Breath" by relation "CausedBy", we achieve a set of entities: {"Fulvestrant", "Ketamin", "Paclitaxel"}

Traversing KG for Conjunctive Queries (3)

- *"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 arrive at the answers {"Fulvestrant", "Paclitaxel"}

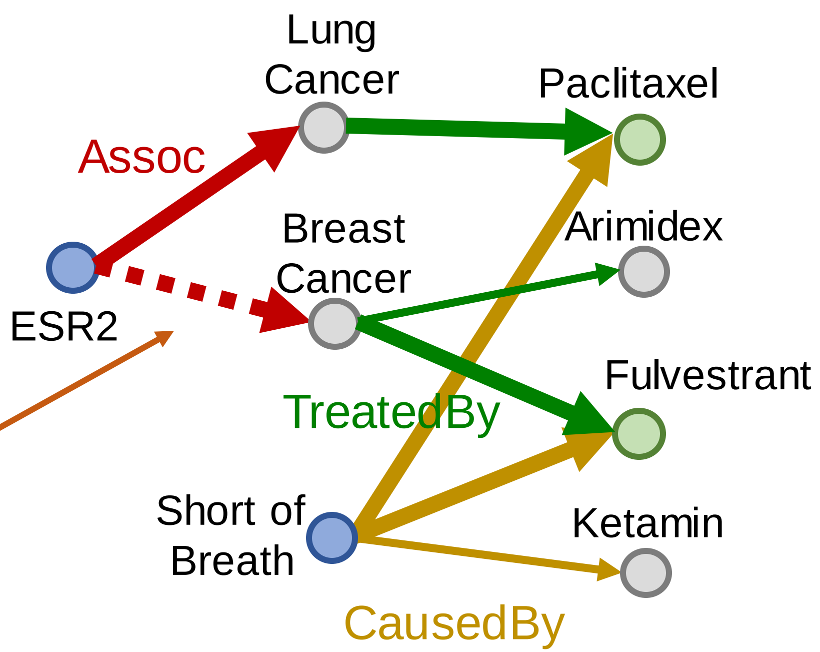
Traversing KG for Conjunctive Queries (4)

- *"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))

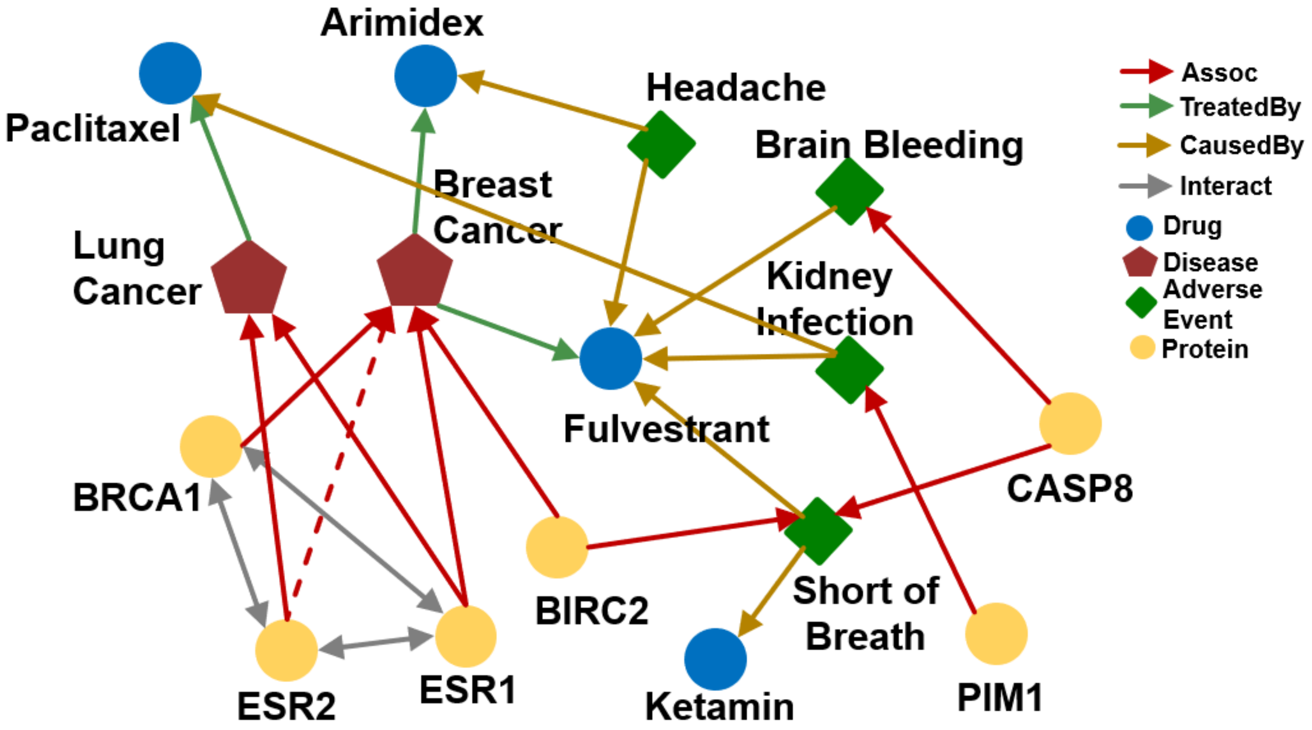
Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

If this link is missing, then we cannot find the answer Fulvestrant



Traversing KG for Conjunctive Queries (5)

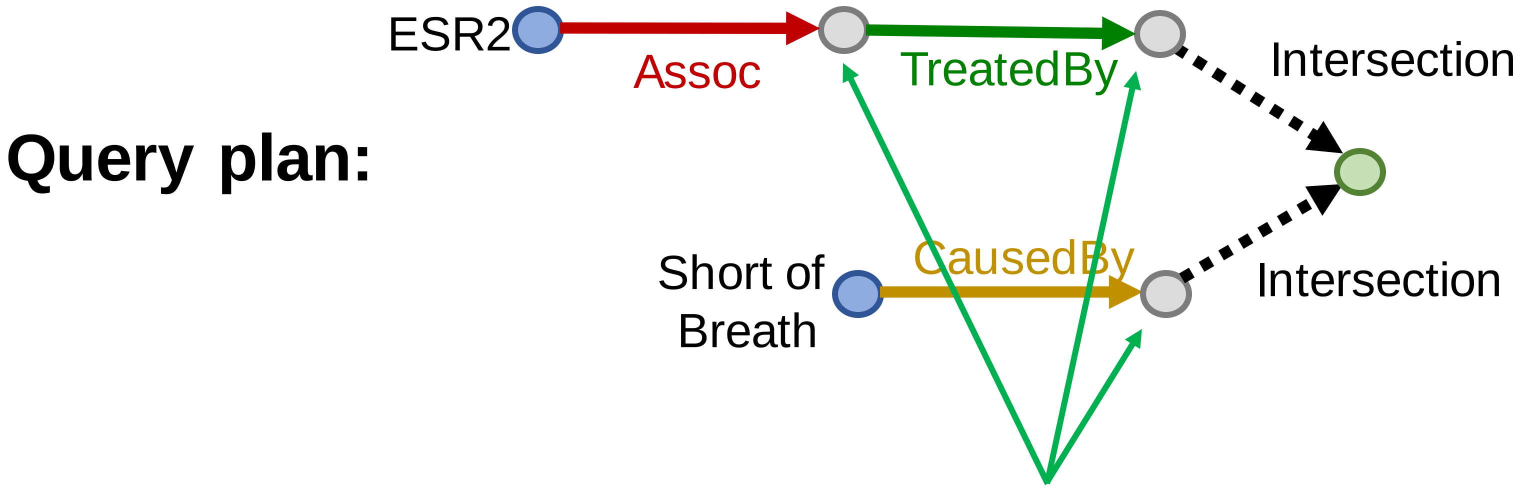
- How can we use embeddings to implicitly impute the **missing (ESR2, Assoc, Breast Cancer)?**
- **Intuition:** **ESR2** interacts with both **BRCA1** and **ESR1**. Both proteins are associated with **breast cancer**.



Represent a set of Entities (1)

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

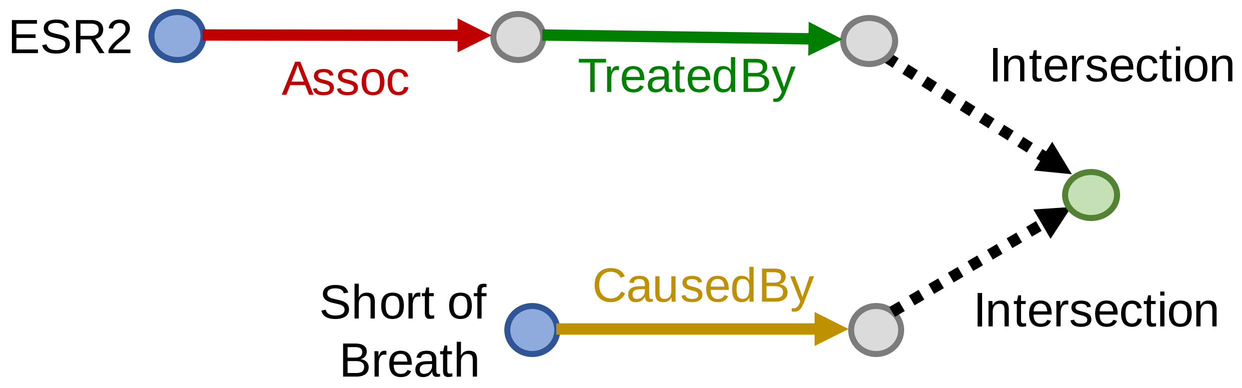
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))



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

Represent a set of Entities (2)

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



- **(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?**

Outline of Today's Lecture

- Queries on KG
- Traversing KG in Embedding Space
- **Box Embeddings**

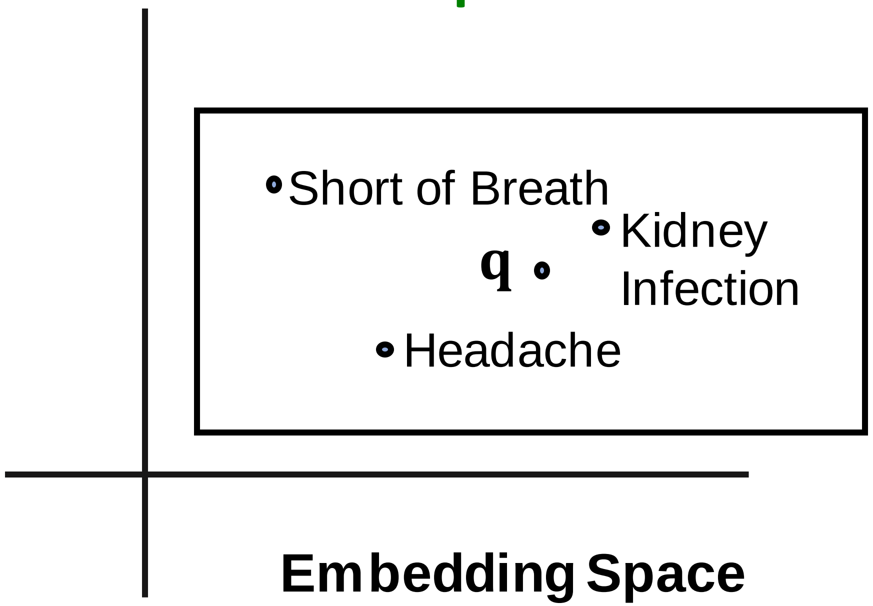
Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. H. Ren*, W. Hu*, J. Leskovec. *International Conference on Learning Representations (ICLR)*, 2020.

Box Embeddings

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

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

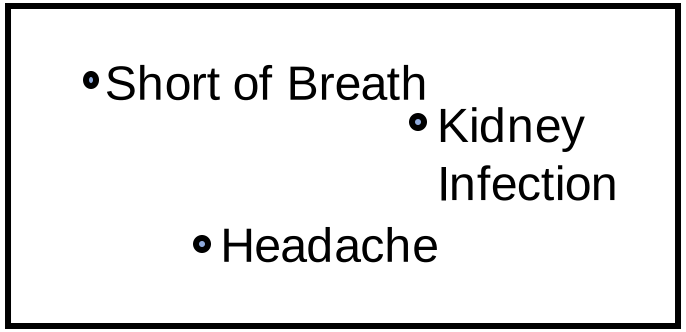
position size



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

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



Embed with Box Embedding (1)

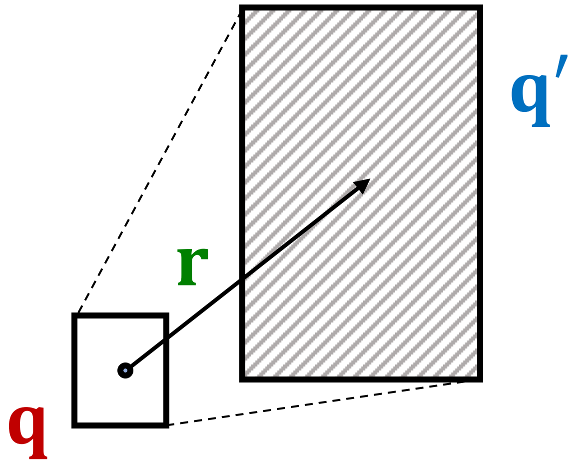
Things to figure out:

- **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:**
  - New operator, inputs are boxes and output is a box
  - Intuitively models intersection of boxes

Projection Operator

- **Projection Operator** $\mathcal{P}$
- **Intuition**:
  - Take the current box as input and use the relation embedding to **project and expand** the box!
- $\mathcal{P}$: Box × Relation → Box

$$\mathrm{Cen}(\mathbf{q}') = \mathrm{Cen}(\mathbf{q}) + \mathrm{Cen}(\mathbf{r})$$
$$\mathrm{Off}(\mathbf{q}') = \mathrm{Off}(\mathbf{q}) + \mathrm{Off}(\mathbf{r})$$



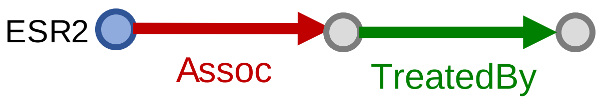
Embed with Box Embedding (2)

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

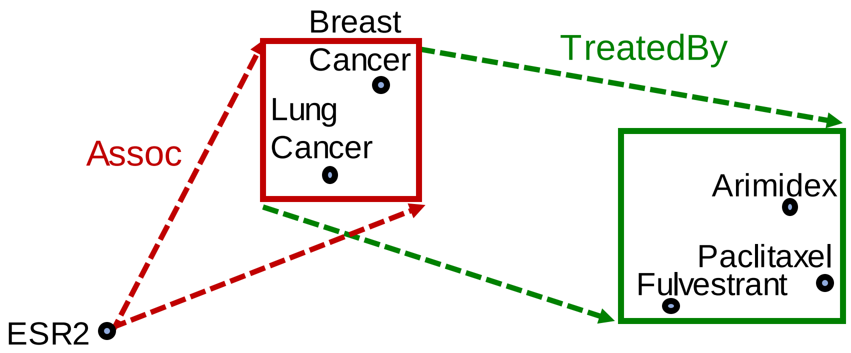
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))

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

Query Plan



Embedding Space



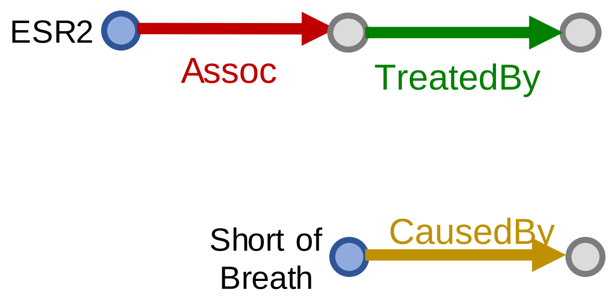
Embed with Box Embedding (3)

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

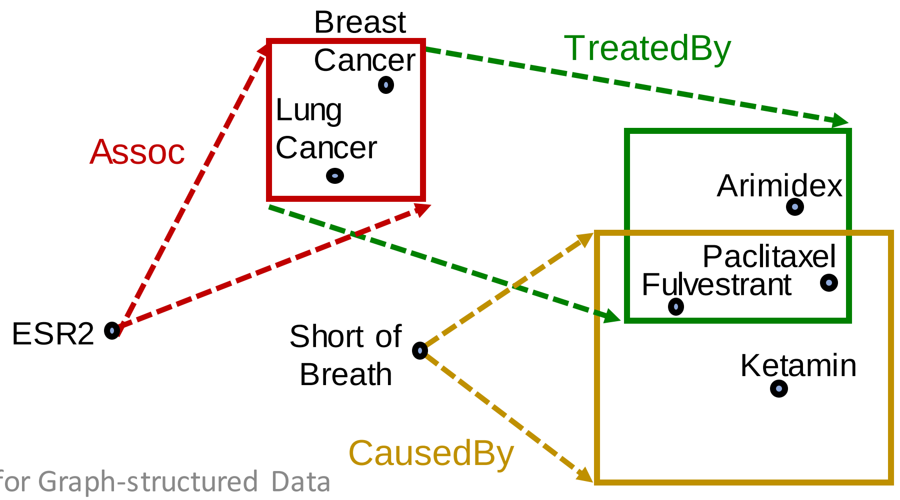
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))

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

Query Plan

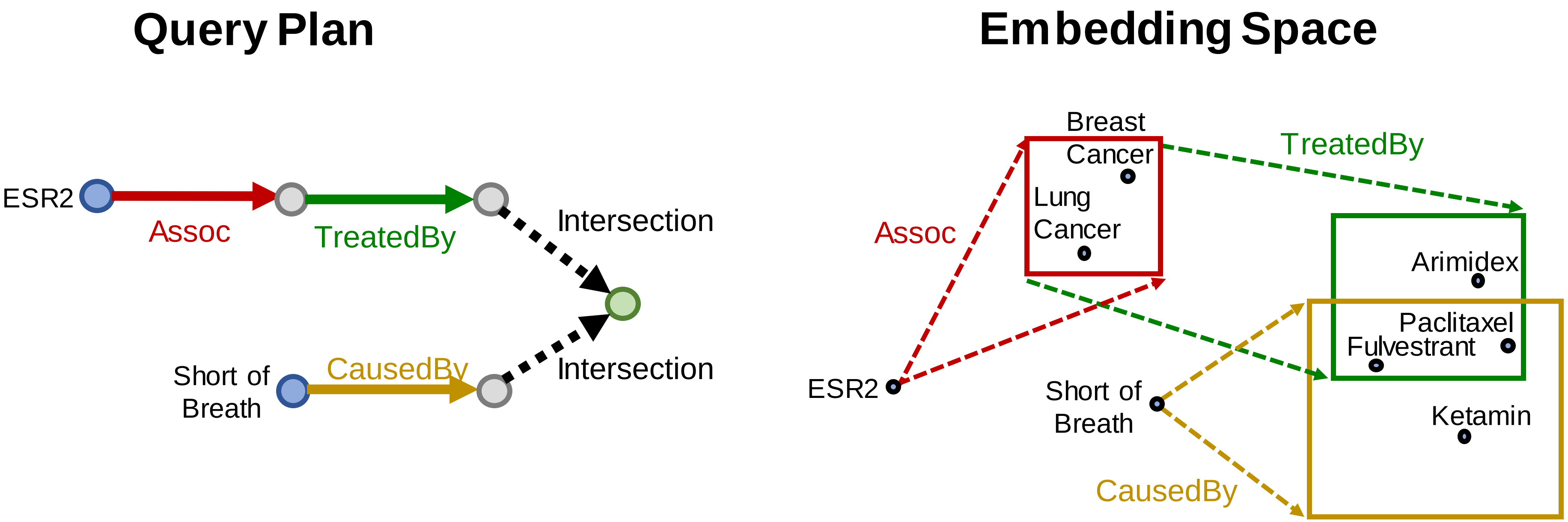


Embedding Space



Embed with Box Embedding (4)

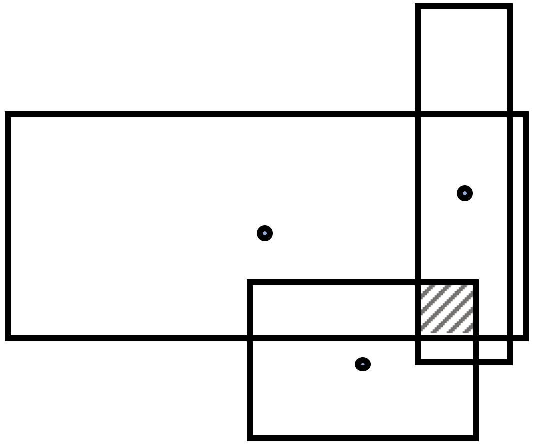
How do we take intersection of boxes?



Note: One possible choice here would be to directly take set intersection, however, using richer learnable parameterization (will be introduced next) is more expressive and robust.

Intersection Operator (1)

- **Geometric Intersection Operator $\mathcal{J}$**
  - 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)
- $\mathcal{J}$: Box $\times \cdots \times$ Box $\rightarrow$ Box



Intersection Operator (2)

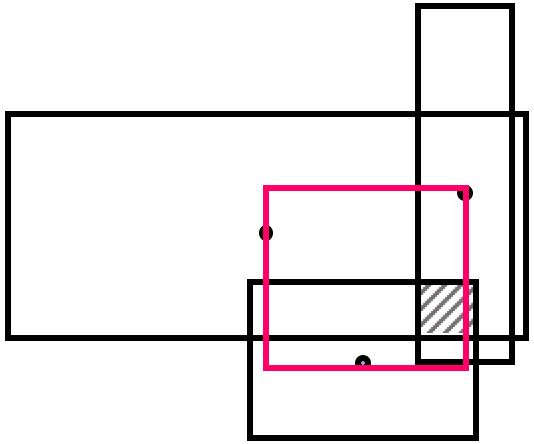
- **Geometric Intersection Operator $\mathcal{J}$**
- $\mathcal{J}$: Box × ⋯ × Box → Box

Hadamard product
(element-wise product)

$$\mathrm{Cen}(\mathbf{q}_{\mathrm{inter}}) = \sum_i \boldsymbol{w}_i \odot \mathrm{Cen}(\mathbf{q}_i)$$

$$\boldsymbol{w}_i = \frac{\exp(f_{\mathrm{cen}}(\mathrm{Cen}(\mathbf{q}_i)))}{\sum_j \exp(f_{\mathrm{cen}}(\mathrm{Cen}(\mathbf{q}_j)))} \qquad \mathrm{Cen}(\mathbf{q}_i) \in \mathbb{R}^d \quad \boldsymbol{w}_i \in \mathbb{R}^d$$

- **Intuition**: The center should be in the red region!
- **Implementation**: The center is a **weighted sum** of the input box centers
- $\boldsymbol{w}_i \in \mathbb{R}^d$ is calculated by a neural network f_{cen} (with trainable weights)
- $\boldsymbol{w}_i$ **represents a "self-attention" score for the center of each input** $\mathrm{Cen}(\mathbf{q}_i)$.



Intersection Operator (3)

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

guarantees shrinking

Sigmoid function: squashes output in (0,1)

- **Geometric Intersection Operator $\mathcal{J}$**
- $\mathcal{J}$: Box $\times \cdots \times$ Box $\rightarrow$ Box

$$\mathrm{Off}(\mathbf{q}_{inter}) = \min\big(\mathrm{Off}(\mathbf{q}_1), \ldots, \mathrm{Off}(\mathbf{q}_n)\big) \odot \sigma(f_{\text{off}}(\mathrm{Off}(\mathbf{q}_1), \ldots, \mathrm{Off}(\mathbf{q}_n)))$$

- **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**.

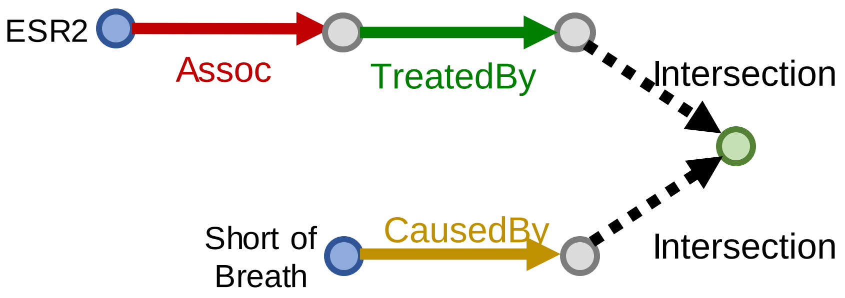
Embed with Box Embedding (5)

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

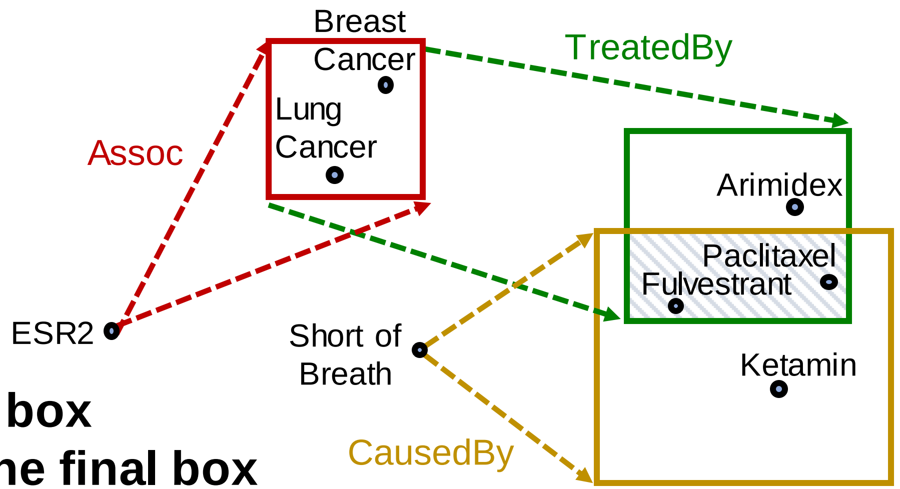
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))

- **Use box intersection operator**

Query Plan



Embedding Space



The shadow box represents the final box embedding of the query

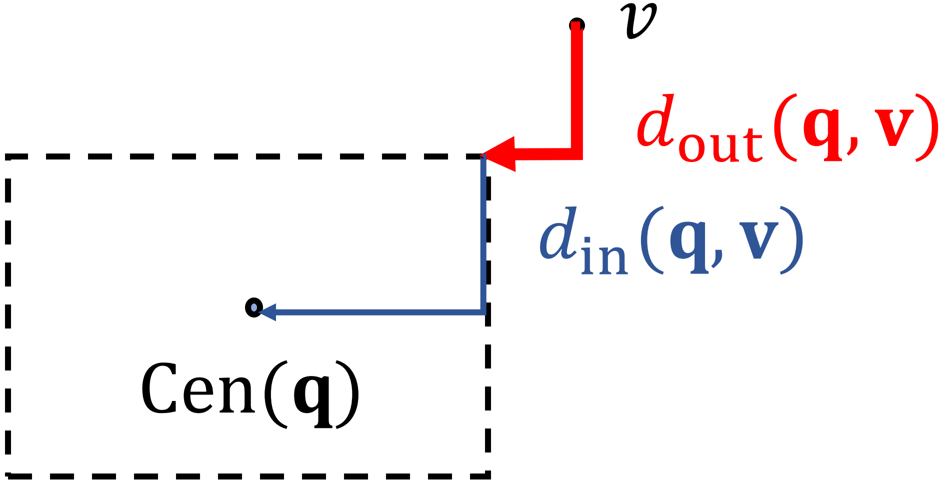
Entity-to-Box Distance

- **How do we define the score function $\boldsymbol{f_q(v)}$ (negative distance)?**
  ($\boldsymbol{f_q(v)}$ captures inverse distance of a node $\boldsymbol{v}$ as answer to $\boldsymbol{q}$)
- Given a query box $\mathbf{q}$ and entity embedding (box) $\mathbf{v}$,

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

  where $0 < \alpha < 1$.

- **Intuition**: if the point is enclosed in the box, the distance should be **downweighted**.
- $\boldsymbol{f_q(v) = -d_{\text{box}}(\mathbf{q},\mathbf{v})}$



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 **E**xistential **P**ositive **F**irst-**o**rder (**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?**

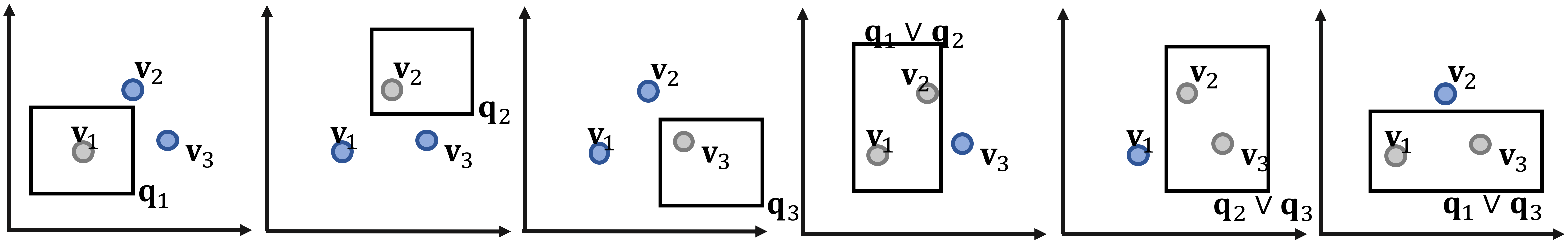
Embedding AND-OR Queries (1)

- **Can we embed AND-OR queries in a low-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:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in a **two-dimensional** plane?

Embedding AND-OR Queries (2)

- **Example 1:**
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in a **two-dimensional** plane?

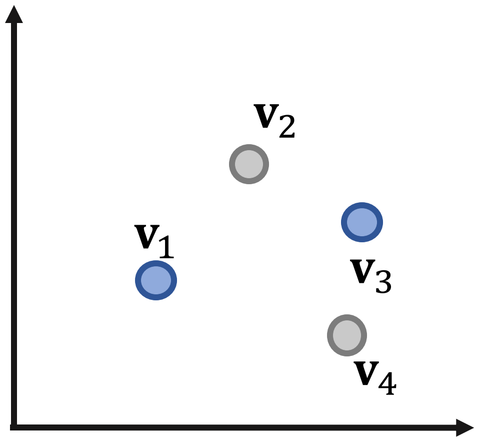
We want **grey dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box



- For 3 points, 2-dimension is okay! **How about 4 points?**

Embedding AND-OR Queries (3)

- **Example 2:**
  - Given 4 queries q_1, q_2, q_3, q_4 with answers:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}, [\![q_4]\!] = \{v_4\},$
  - 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 (4)

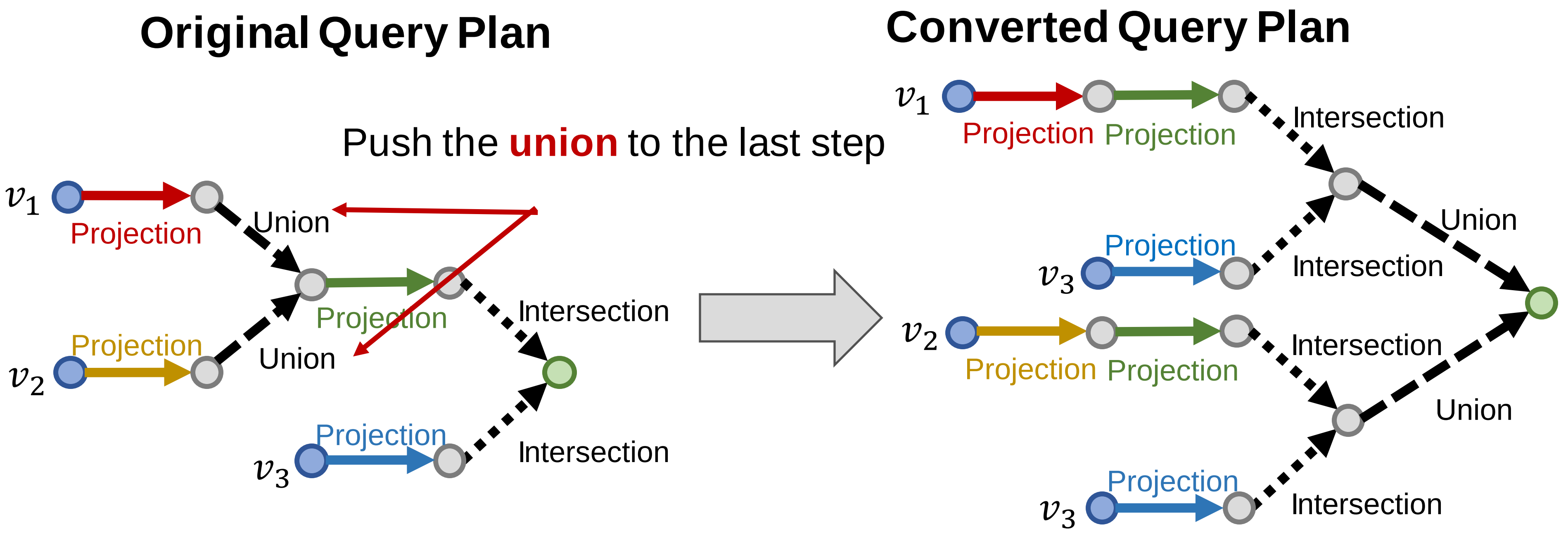
Can we embed AND-OR queries in low-dimensional vector space?

- **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.
- Find the formal theorem and proof in *Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings*, Appendix A.

Embedding AND-OR Queries (5)

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**!



Disjunctive Normal Form

Disjunctive Normal Form

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

- **Given any AND-OR query $\boldsymbol{q}$,**

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

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 (1)

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

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

- **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 v is close to one conjunctive query q_i, then v should be close to q **in the embedding space**

Distance Between q and an Entity (2)

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

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

- **The process of embedding any AND-OR query q**
  1. Transform q to **equivalent DNF** $q_1 \vee \cdots \vee q_m$
  2. **Embed** $q_1, \ldots, q_m$ do get $\mathbf{q}_1, \ldots, \mathbf{q}_m$
  3. Calculate the (box) distance $d_{\text{box}}(\mathbf{q}_i, \mathbf{v})$
  4. Take the **minimum** of all distance
  5. **The final score** $f_q(v) = -d_{\text{box}}(\mathbf{q}, \mathbf{v})$

Training Overview

- **Overview and Intuition** (similar to KG completion):
  - Given a query q, maximize the score $f_q(v)$ for answers $v \in [\![q]\!]$ and minimize the **scoring function** $f_q(v')$ for negative answers $v' \notin [\![q]\!]$
- 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?**

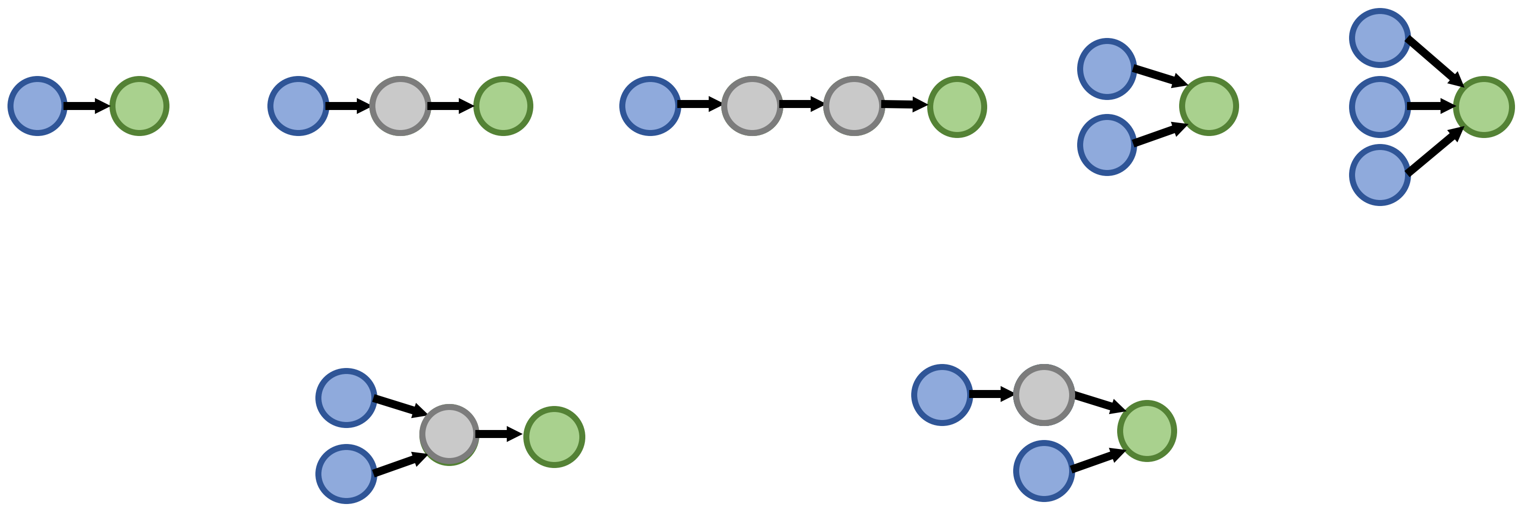
Training Pipeline

- **Training**:
  1. Randomly sample a query q from the training graph G_{train}, answer $v \in [\![q]\!]_{G_{\text{train}}}$, and a negative sample $v' \notin [\![q]\!]_{G_{\text{train}}}$.
     - Negative sample: Entity of same type as v but not answer.
  2. Embed the query q to $\mathbf{q}$.
  3. Calculate the **scoring function** $f_q(v)$ and $f_q(v')$.
  4. Optimize the loss ℓ to maximize $f_q(v)$ while minimize $f_q(v')$:

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

Query Generation from Templates (1)

- **Generate queries from multiple query templates:**



Query Generation from Templates (2)

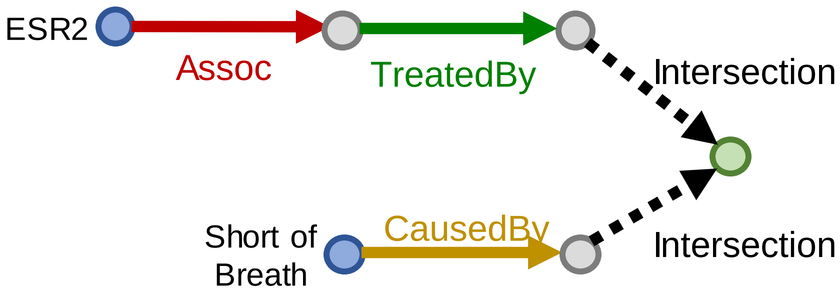
- How can we generate **a complex query**?
- We start with a **query template**
- **Query template** can be viewed as an abstraction of the query
- We generate a query by instantiating every variable with a concrete entity and relation from the KG
  - E.g., instantiate **Anchor1** with **ESR2** (a node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

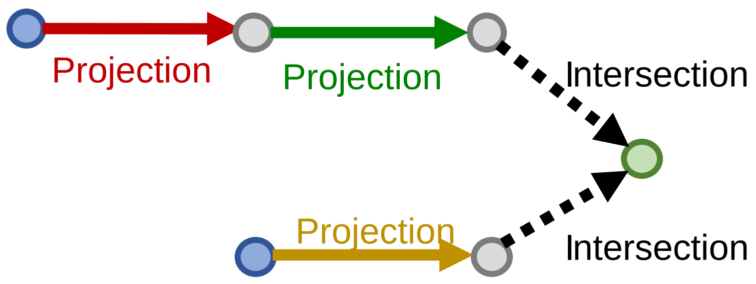
Query

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

Query Template

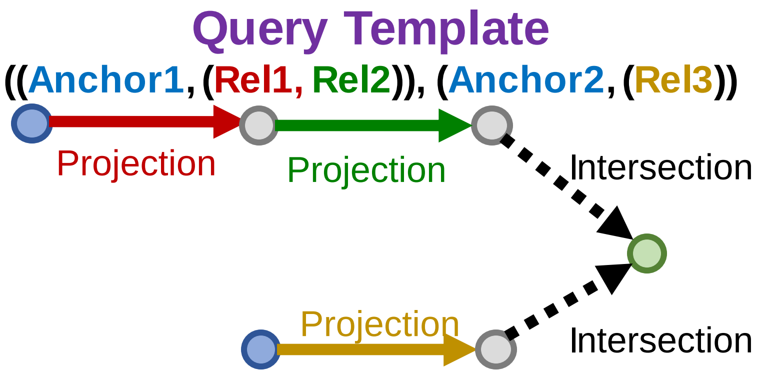
((Anchor1, (Rel1, Rel2)), (Anchor2, (Rel3))

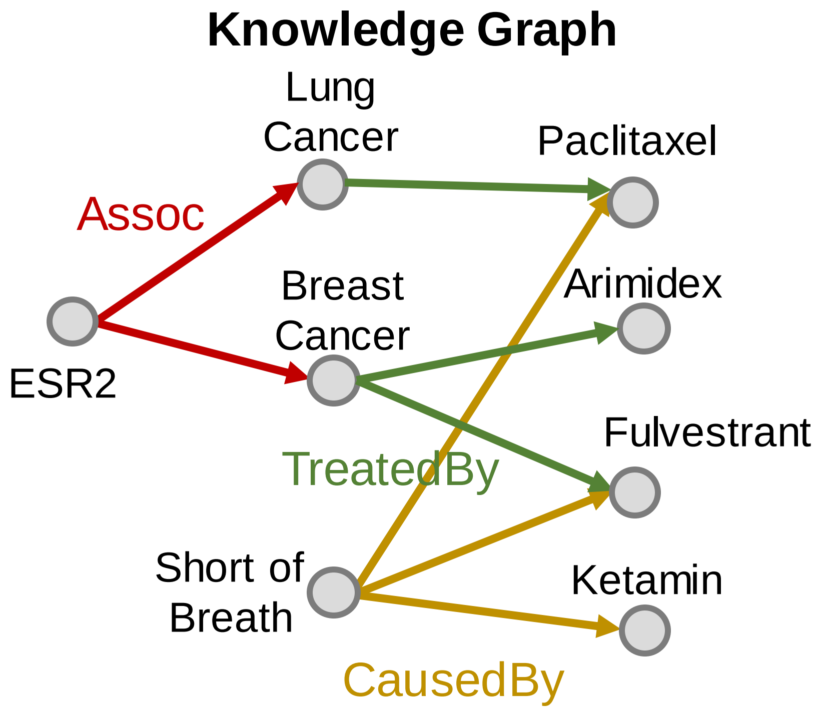




Query Generation from Templates (3)

- **How to instantiate a query template given a KG?**

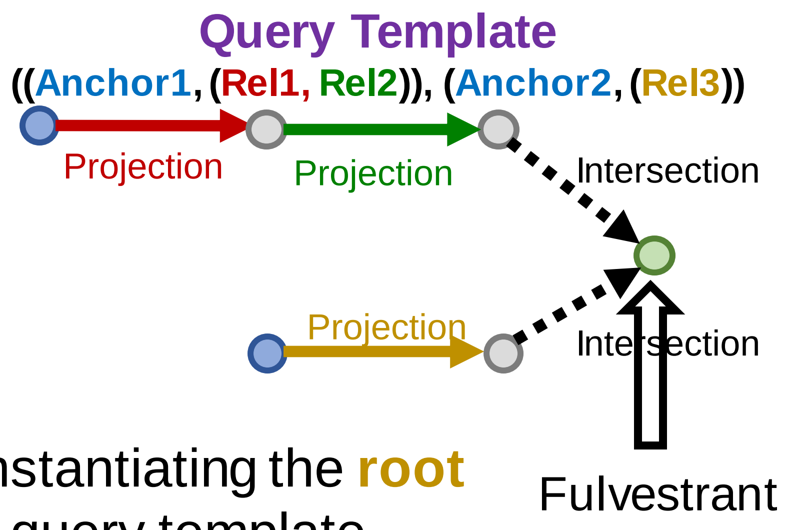


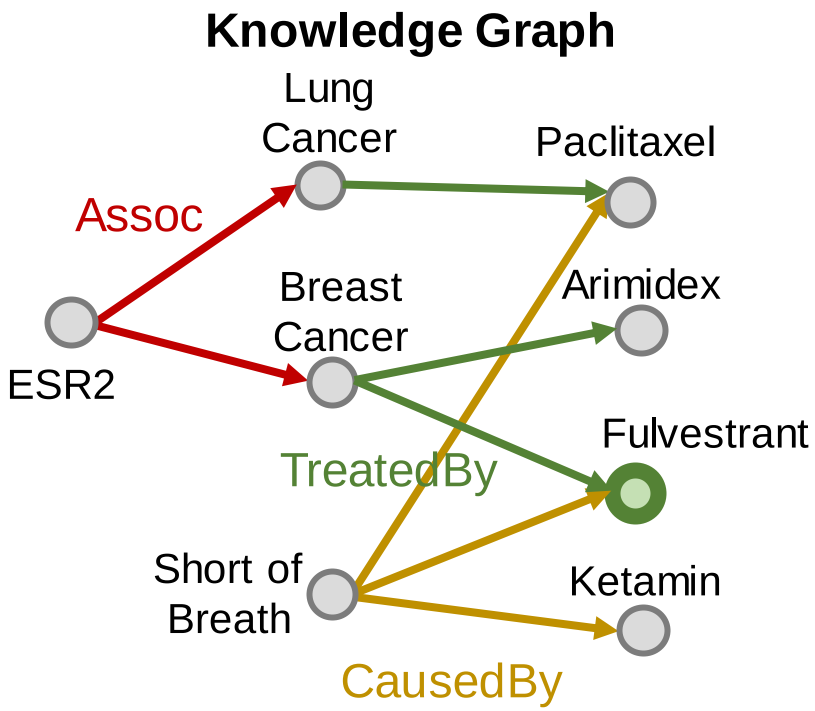


Overview:
Start from instantiating the **answer node** of the query template and then iteratively instantiate the other edges and nodes until we ground **all the anchor nodes**

Query Generation from Templates (4)

- **How to instantiate a query template given a KG?**

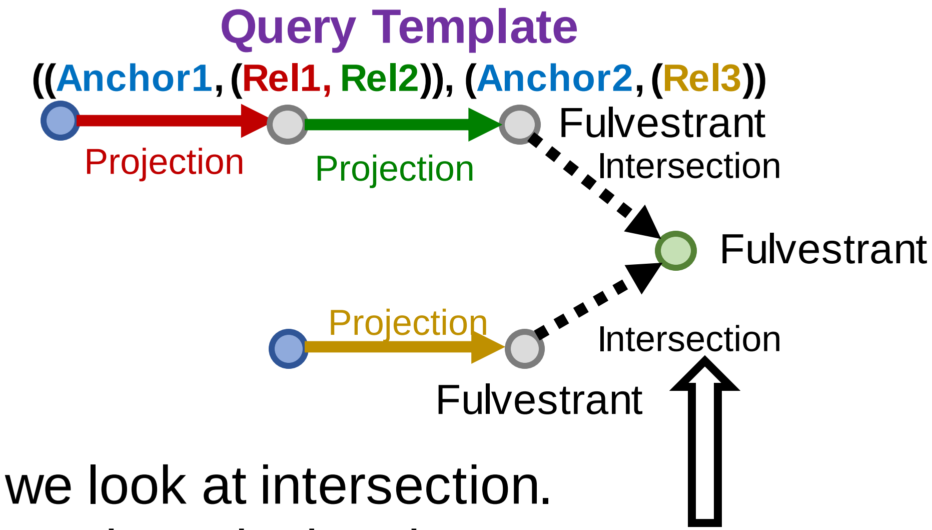


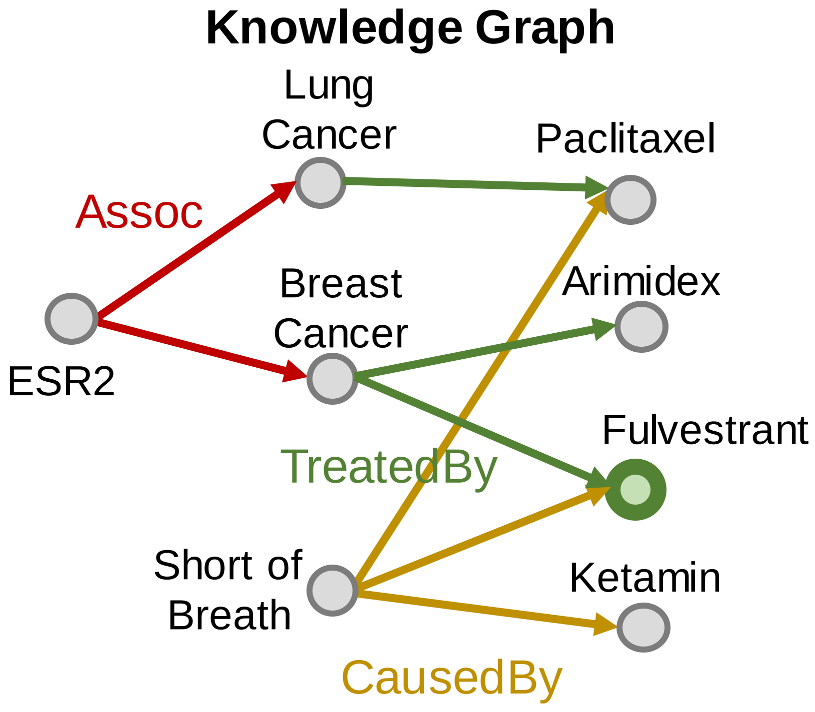


Start from instantiating the **root node** of the query template. Randomly pick one entity from KG as the root node, e.g., we pick **Fulvestrant**.

Query Generation from Templates (5)

- **How to instantiate a query template given a KG?**

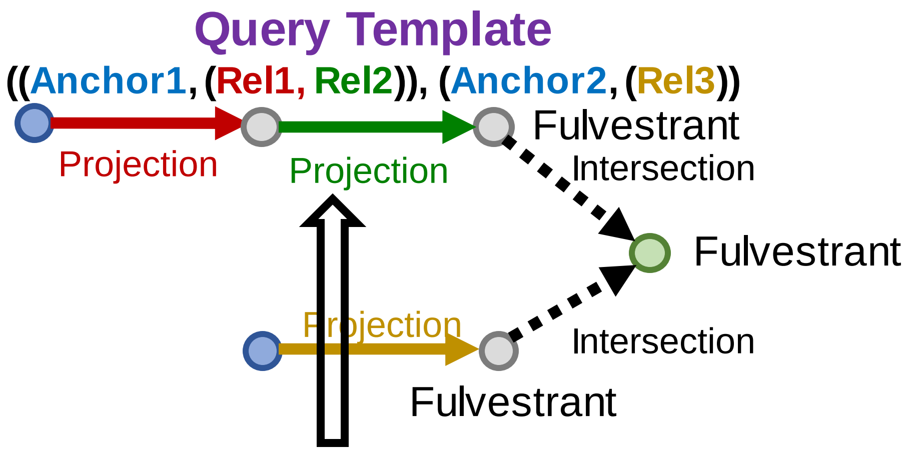


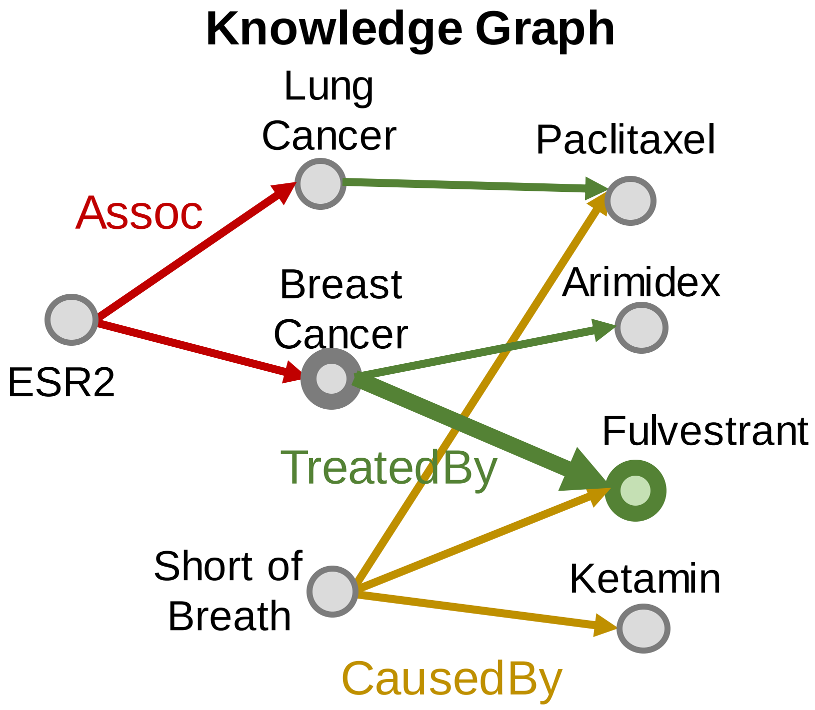


Now we look at intersection. What we have is that the intersection of the sets of entities is **Fulvestrant**, then naturally the two sets should also contain **Fulvestrant**.

Query Generation from Templates (6)

- **How to instantiate a query template given a KG?**

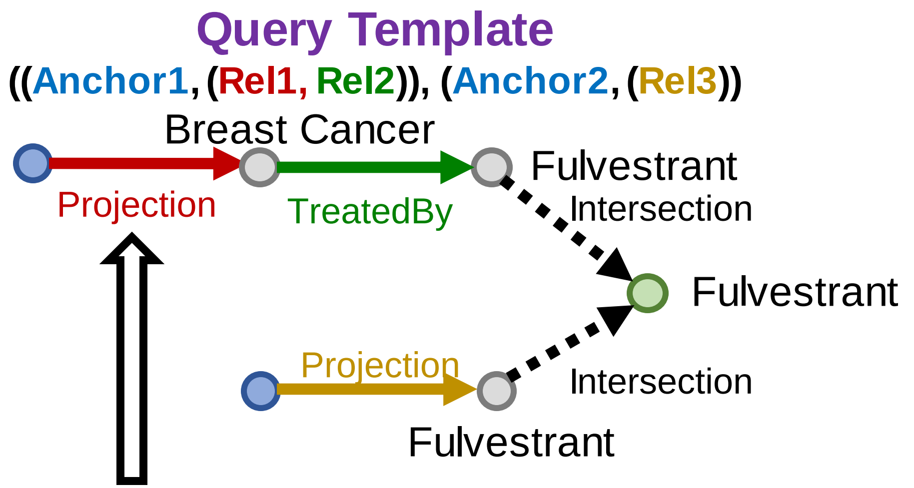


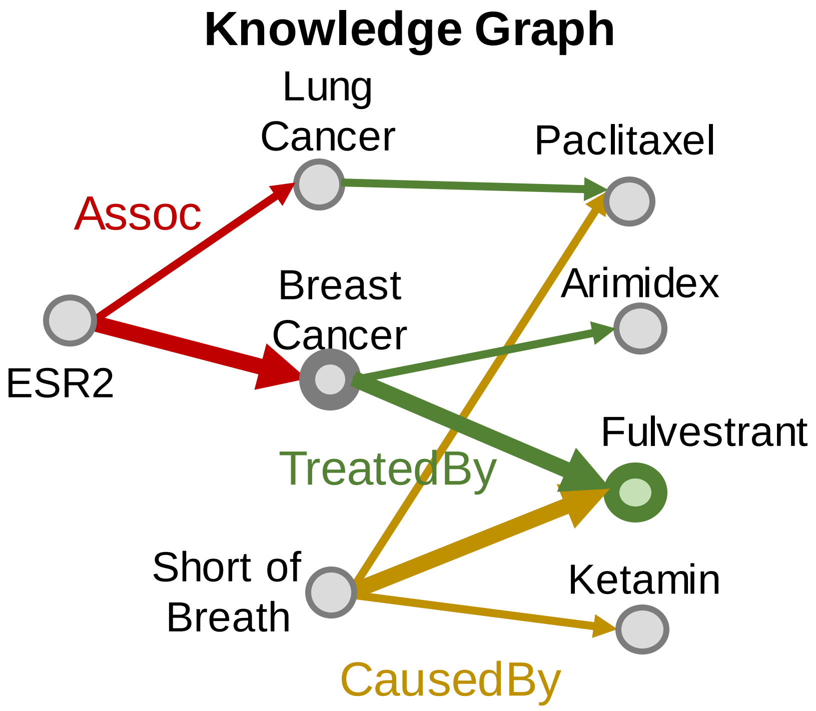


We instantiate the **Projection edge** in the template by randomly sample one relation associated with the current entity **Fulvestrant**. For example, we may select relation **TreatedBy**, and check what entities are connected to **Fulvestrant** with **TreatedBy**: {**Breast Cancer**}.

Query Generation from Templates (7)

- **How to instantiate a query template given a KG?**





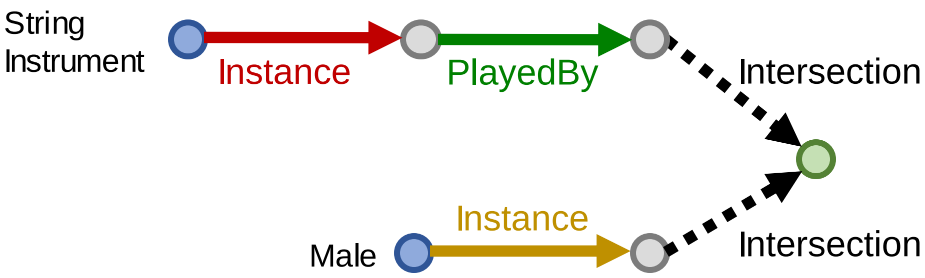
We first look at one branch and ground the **Projection edge** with the relation associated with **Breast Cancer**, e.g., **Assoc**. Then we check what entities are connected to **Breast Cancer** with **Assoc**: {**ESR2**}.

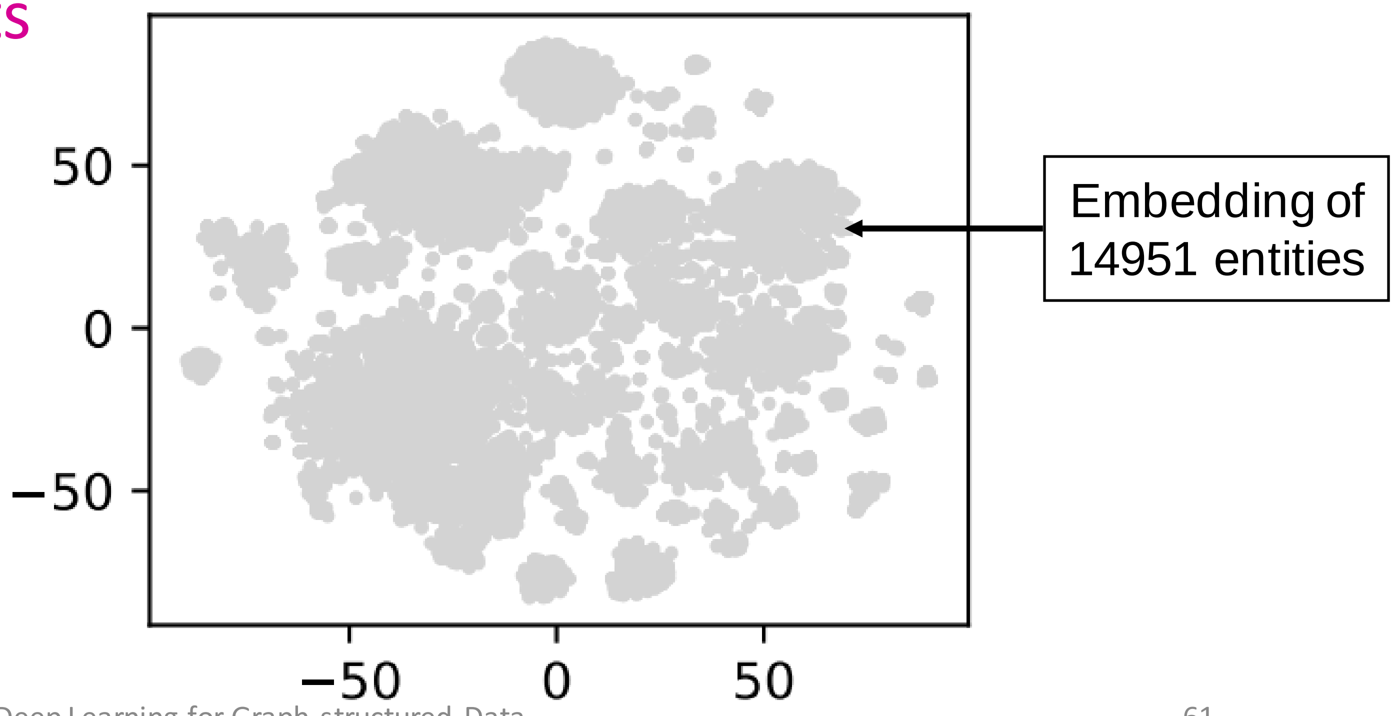
Visualization of Box Embeddings (1)

- **What do box embeddings actually learn?**

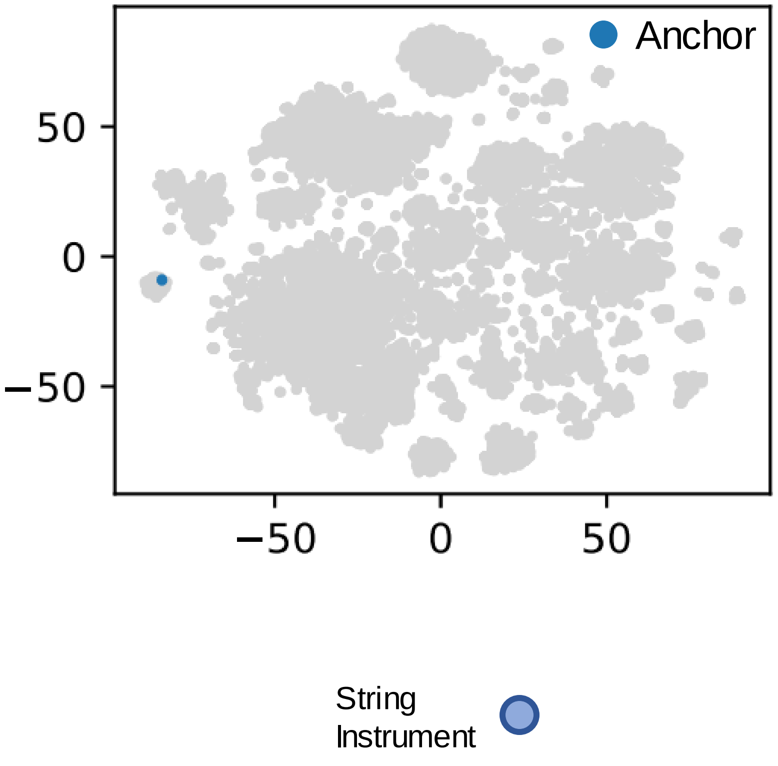
Example: *“List* ***male instrumentalists*** *who* ***play*** *string instruments”*

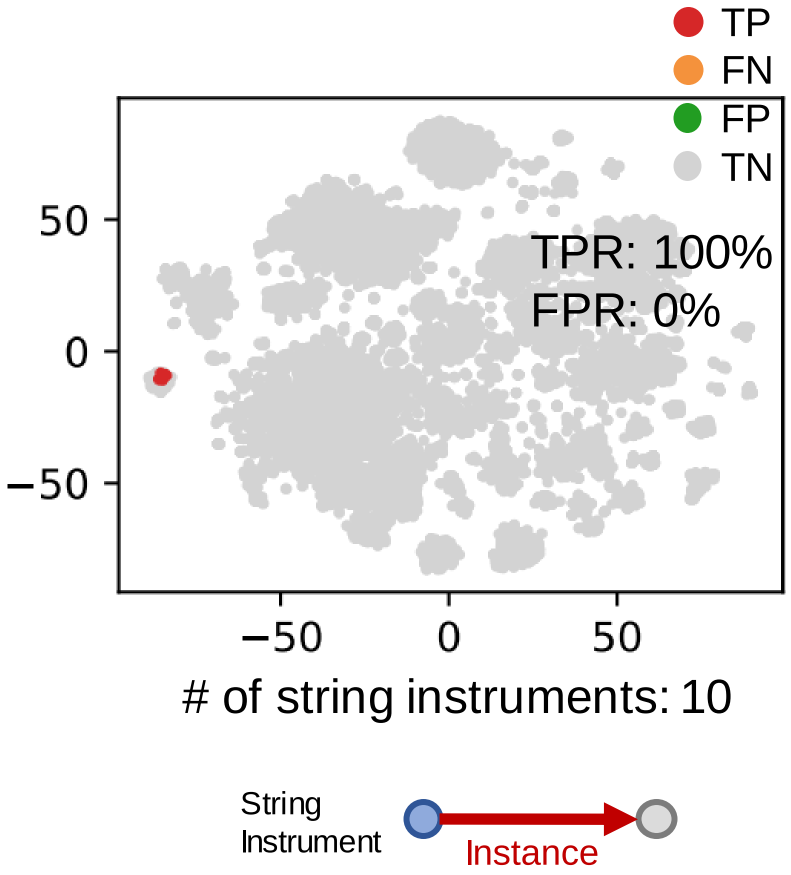
- We use **t-SNE** to reduce the embedding space to a 2-dimensional space, in order to visualize the query results

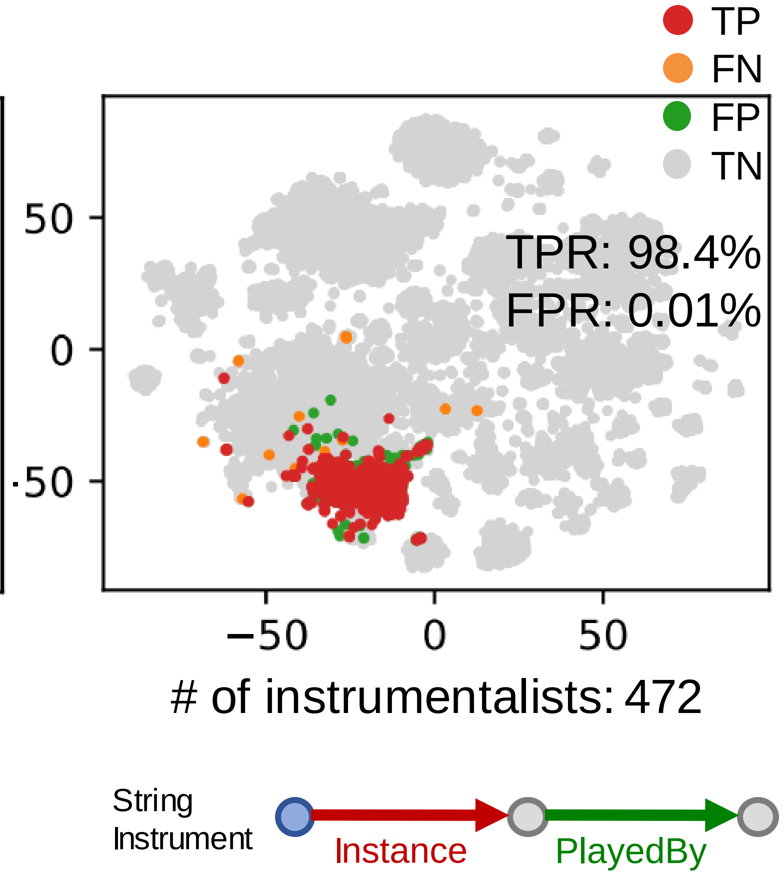




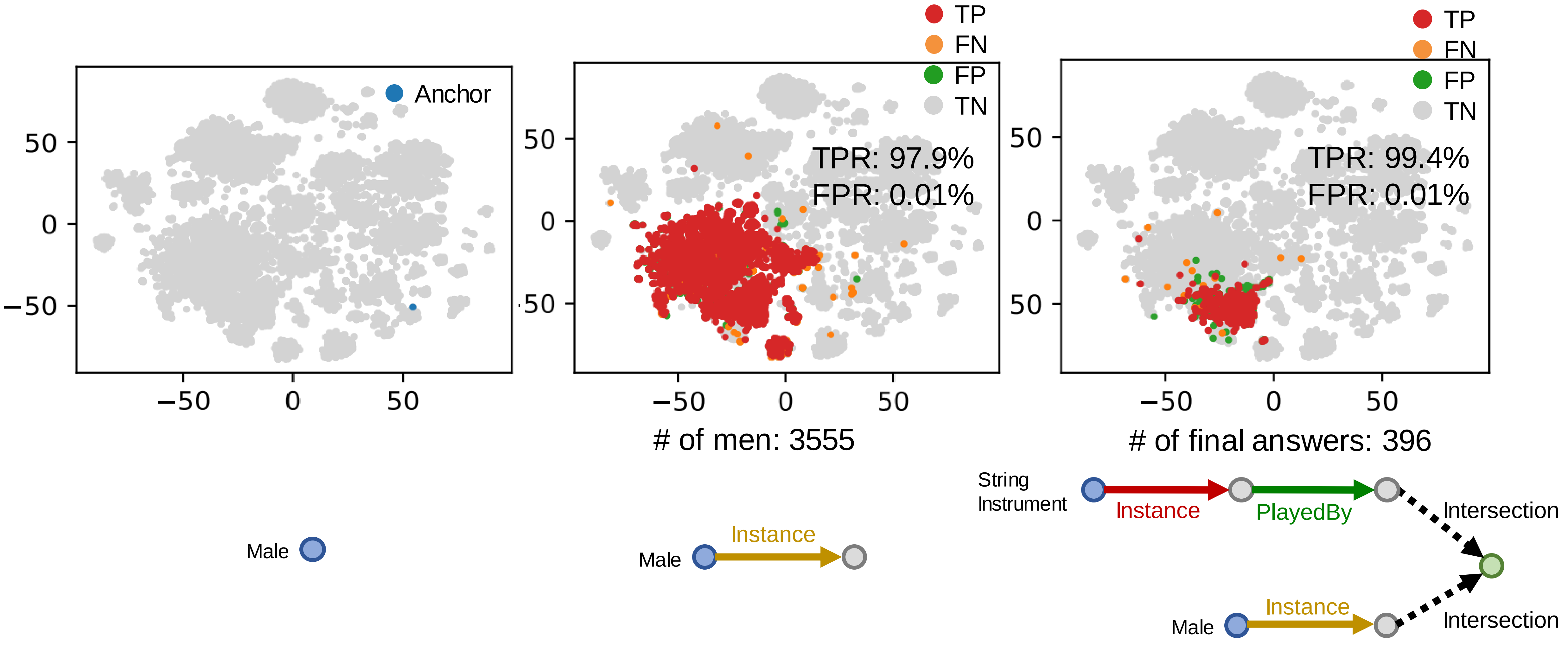
Visualization of Box Embeddings (2)







Visualization of Box Embeddings (3)



Summary

- **Queries on KG**
  - **One-hop queries, Path queries, Conjunctive queries**
- **Traversing KG in the embedding space**
  - We embed the query by composing learned operators (e.g. **TransE**)
  - Embedding of the query is **close** to its answers in the embedding space
- **Query2Box**
  - **Box Embeddings** to represent a set of entities
  - Embed **AND-OR** queries by transforming them into their equivalent Disjunctive Normal Form (DNF).