Knowledge Graphs

Few slides from CS224W @ Stanford

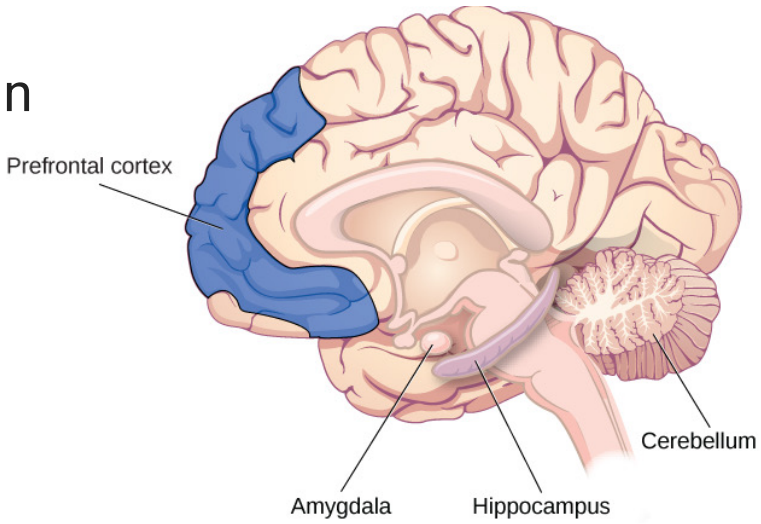
Mrinmaya Sachan
msachan@ethz.ch

The need for a Knowledge Store in AI systems

There is increasing evidence that humans **store information** in a part of their brains (most probably the **hippocampus**).

They can readily **retrieve** information as needed:

- When a memory is **created**, information flows from the **cortex**, the part of the brain rich in nerve cells, to the **hippocampus**, the central switching point for memories in the brain.
- The information flows in the opposite direction when we **retrieve** a memory



Perhaps, AI models should be able to store and interact with information/knowledge as well.

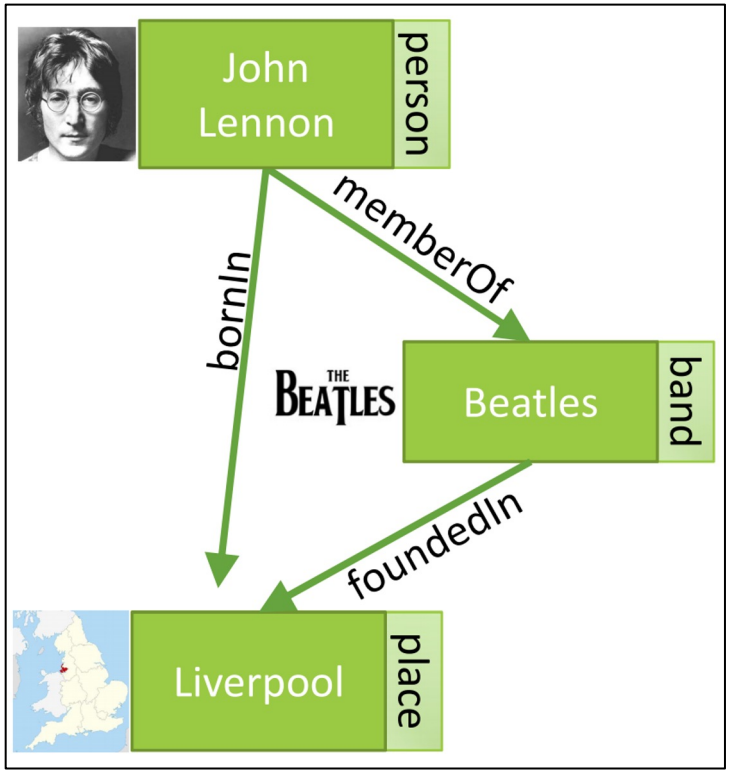
Knowledge Graphs

Describes **entities** in the real world and their relationships

Nodes are entities

Nodes are labeled with their types

Edges between two nodes capture relationships between entities



Easy and sound way of representing world knowledge in a way that is usable by computers

Knowledge Graphs: Another representation

Represented as set of **Subject, Relation, Object** triples

- (John Lennon, memberOf, Beatles)
- (Beatles, foundedIn, Liverpool)
- (John Lennon, bornin, Liverpool)

Industry Scale Knowedge Graphs

GOOGLE KNOWLEDGE GRAPH

- https://developers.google.com/knowledge-graph/

AMAZON PRODUCT GRAPH

FACEBOOK GRAPH API

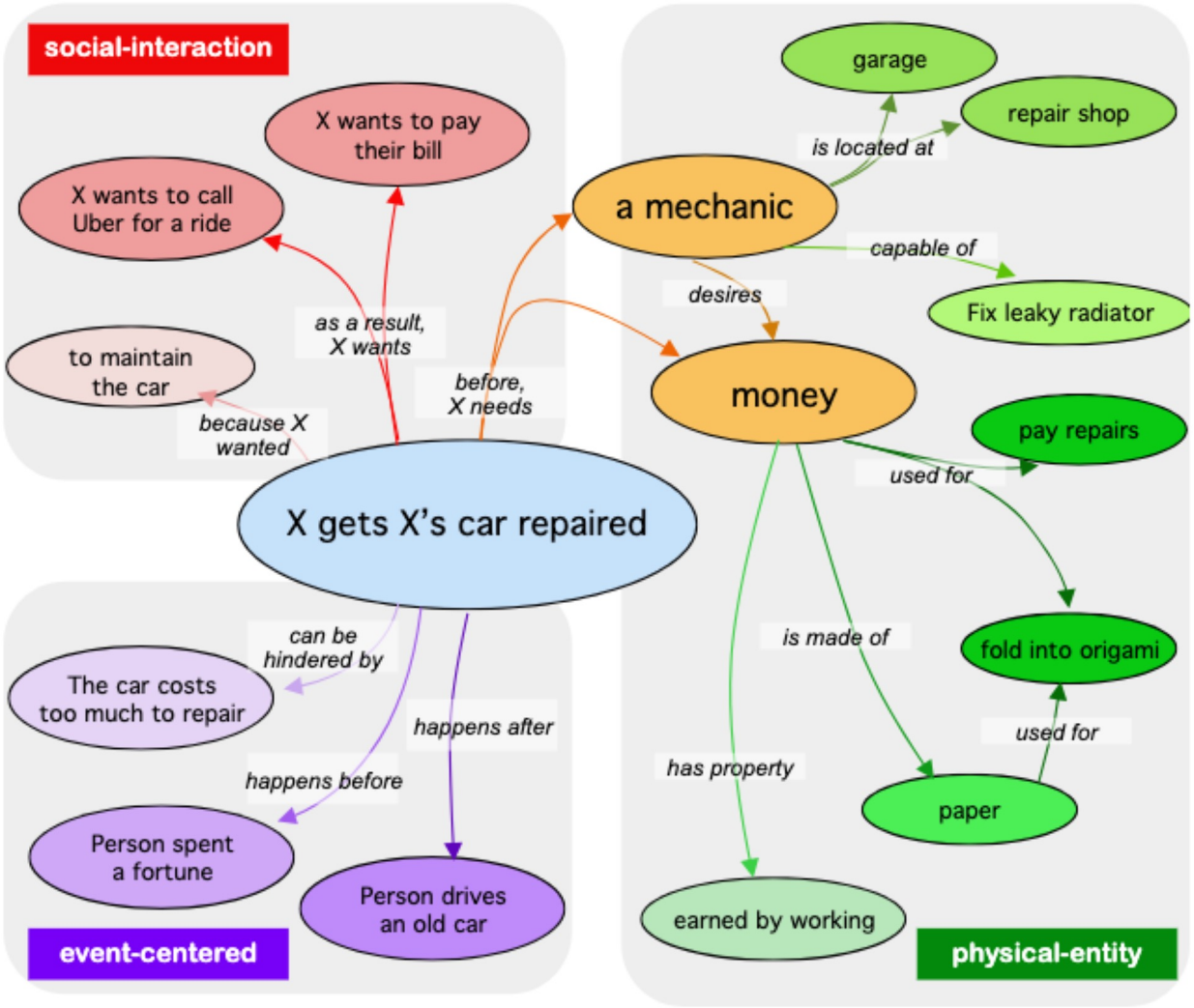
- https://developers.facebook.com/docs/graph-api/

MICROSOFT SATORI

LINKEDIN KNOWLEDGE GRAPH

Academic Knowledge Graphs

YAGO (MPI), ConcepNet (MIT Media Labs), NELL (CMU), **ATOMIC** (UW/AI2)



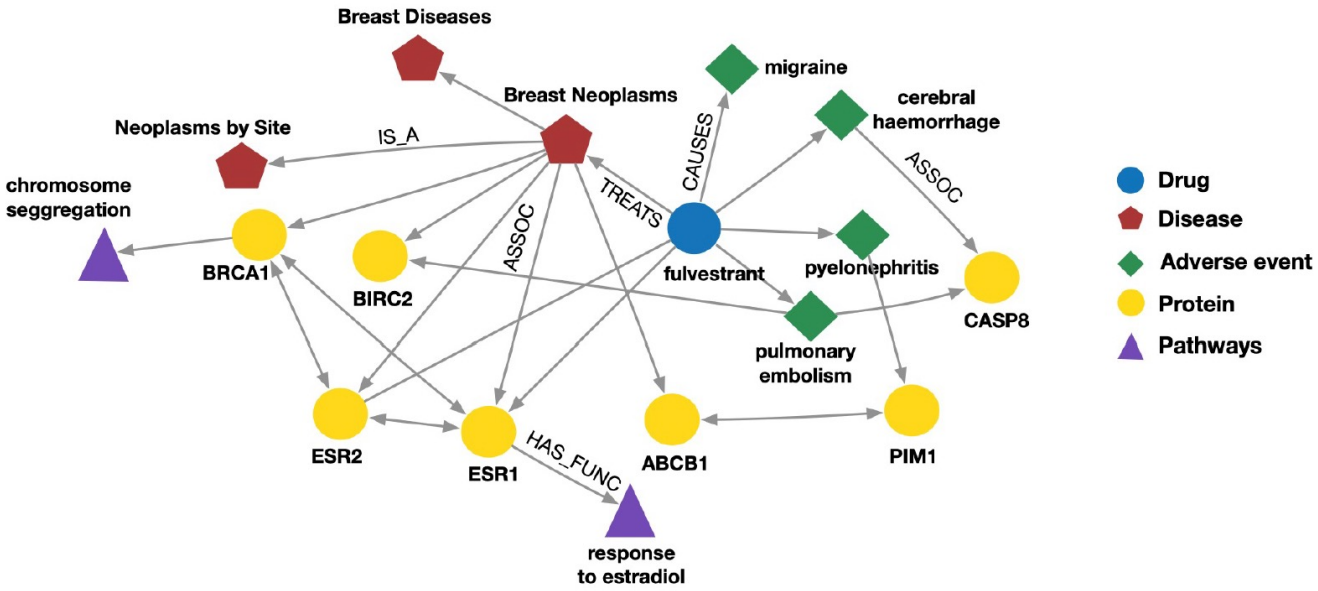
Commonsense Knowledge

Domain specific KGs: Biology and Healthcare

MeSH (Medline/PubMed), UMLS (Unified Medical Language System), Disease Ontology, Cell Ontology, Gene Ontology

Node types: drug, disease, adverse event, protein, pathways

Relation types: has_func, causes, assoc, treats, is_a



Applications of Knowedge Graphs

Serving information:

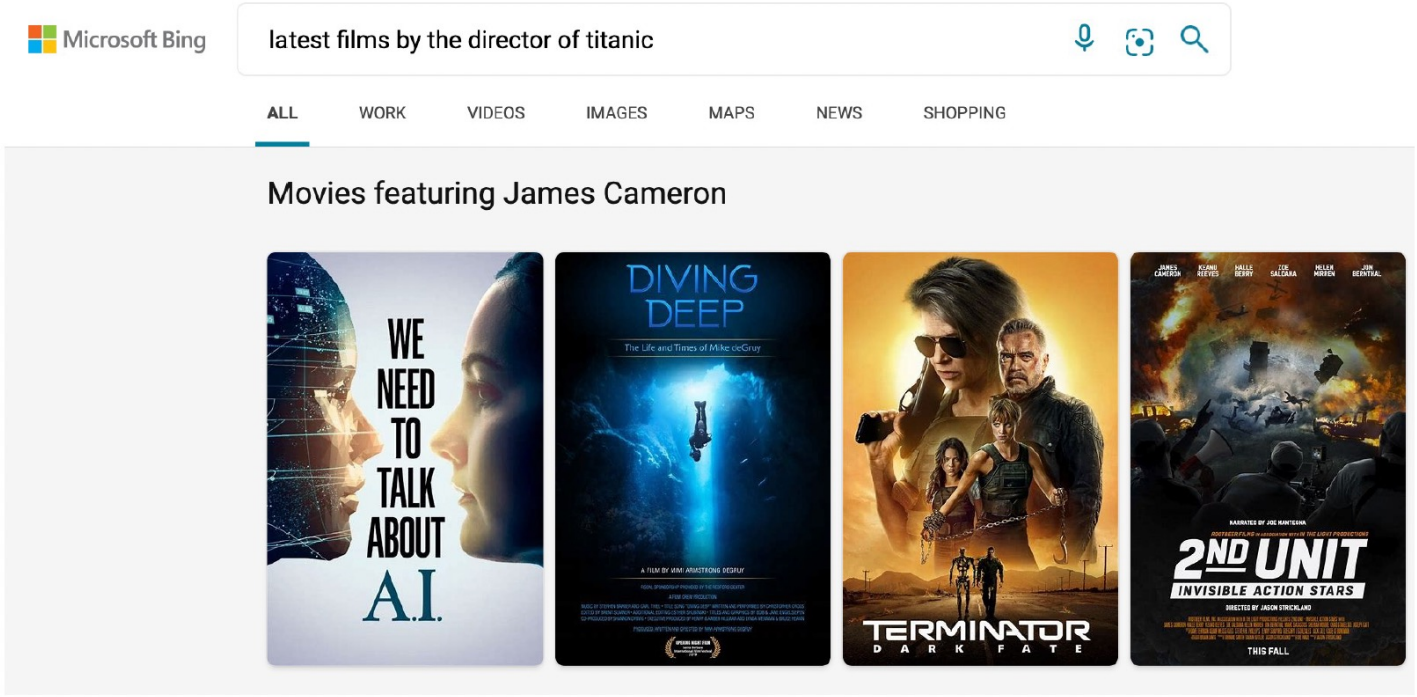


Image credit: Bing

Question answering and conversation agents

However…

- KGs are inherently **symbolic**
  - We can search a KG and run interpreted queries such as "foundedIn(Beatles, **?**)"

- It is challenging to incorporate them in deep learning architectures that require **continuous representations**

Thus…

- We need a way to embed a knowledge graph into a **continuous** vector space!

Knowledge Graph Representations

Knowledge Graph Representations

Given a collection of facts denoted as (h, r, t) from our KG

We wish to learn embeddings **h**, **r**, **t** for the head and tail entities and relation. Let $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^k$

How can we learn these representations?

The key idea is to learn an embedding space such that **h** when "combined" with **r** is close to **t**

Knowledge Graph Representations

The key idea is to learn an embedding space such that **h** when "combined" with **r** is close to **t**

Let $f_r(\boldsymbol{h}, \boldsymbol{t})$ measure the **compatibility** of the embedding of the head entity with the embedding of the tail entity given the embedding of the relation.

- In other words, f quantifies the **plausibility** of the embedded representation of a given fact.

We want:

Score f_r should be high if (h, r, t) exists in the KB, else f is low

A more refined way to think about this…

Objective:

$$\min \sum_{(h,r,t) \in KG} L(f_\theta(\boldsymbol{h}, \boldsymbol{r}), \boldsymbol{t})$$

Here, f_θ combines $\boldsymbol{h}$ and $\boldsymbol{r}$

L measures the closeness between "the combination of $\boldsymbol{h}$ and $\boldsymbol{r}$" and **t**

Twist: To avoid trivial solutions, minimize L for known facts and maximize L for unknown facts generated using **negative sampling**

$$\min \sum_{(h,r,t) \in KG} L(f_\theta(\boldsymbol{h}, \boldsymbol{r}), \boldsymbol{t}) \quad - \sum_{(h,r,t^-) \notin KG} L(f_\theta(\boldsymbol{h}, \boldsymbol{r}), \boldsymbol{t}^-)$$

Some popular KG embedding methods

Different forms of $L(f_\theta(\boldsymbol{h}, \boldsymbol{r}), \boldsymbol{t})$. Let $\mathbf{e_i} = \boldsymbol{h}$ and $\boldsymbol{e_j} = \boldsymbol{t}$

SE (Bordes et al., 2011)	$\|\|\mathbf{W}_r^{(L)}\mathbf{e}_i - \mathbf{W}_r^{(R)}\mathbf{e}_j\|\|_p$
NTN (Socher et al., 2013)	$\mathbf{u}_r^T f\left(\mathbf{e}_i\mathbf{W}_r^{[1\ldots k]}\mathbf{e}_j + \mathbf{V}_r\begin{bmatrix}\mathbf{e}_i\\ \mathbf{e}_j\end{bmatrix} + \mathbf{b}_r\right)$
TransE (Bordes et al., 2013)	$\|\|\mathbf{e}_i + \mathbf{r} - \mathbf{e}_j\|\|_p$
TransH (Wang et al., 2014)	$\|\|(\mathbf{e}_i - \mathbf{w}_r^T\mathbf{e}_i\mathbf{w}_r) + \mathbf{d}_r - (\mathbf{e}_j - \mathbf{w}_r^T\mathbf{e}_j\mathbf{w}_r)\|\|_2^2$
TransR (Lin et al., 2015)	$\|\|\mathbf{e}_i\mathbf{W}_r + \mathbf{r} - \mathbf{e}_j\mathbf{W}_r\|\|_2^2$
TransD (Ji et al., 2015)	$-\|\|(\mathbf{r}_p\mathbf{e}_i{}_p^T + \mathbf{I})\mathbf{e}_i + \mathbf{r} - (\mathbf{r}_p\mathbf{e}_j{}_p^T + \mathbf{I})\mathbf{e}_j\|\|_2^2$
DistMult (Yang et al., 2014)	$\langle\mathbf{e}_i, \mathbf{r}, \mathbf{e}_j\rangle$
ComplEx (Trouillon et al., 2016)	$\text{Re}(\langle\mathbf{e}_i, \mathbf{r}, \bar{\mathbf{e}_j}\rangle)$
HolE (Nickel et al., 2016)	$\langle\mathbf{r}, \mathbf{e}_i \otimes \mathbf{e}_j\rangle$
SimpleE (Kazemi and Poole, 2018)	$\frac{1}{2}(\langle\mathbf{e}_i^{(h)}, \mathbf{r}, \mathbf{e}_j^{(t)}\rangle + \langle\mathbf{e}_j^{(h)}, \mathbf{r}^{(inv)}, \mathbf{e}_i^{(t)}\rangle)$
ConvE (Dettmers et al., 2018)	$\langle\sigma(vec(\sigma([\mathbf{e}_i; \mathbf{r}] \circ \omega))\mathbf{W}), \mathbf{e}_j\rangle$
RotatE (Sun et al., 2019)	$-\|\|\mathbf{e}_i \bullet \mathbf{r} - \mathbf{e}_j\|\|^2$
HypER (Balažević et al., 2019a)	$\langle\sigma(vec(\sigma(\mathbf{e}_i * vec^{-1}(\mathbf{w}_r\mathbf{H})))\mathbf{W}), \mathbf{e}_j\rangle$
TuckER (Balažević et al., 2019b)	$\mathcal{W} \times_1 \mathbf{e}_i \times_2 \mathbf{w}_r \times_3 \mathbf{e}_j$

1) TransE (Bordes et al. 2013)

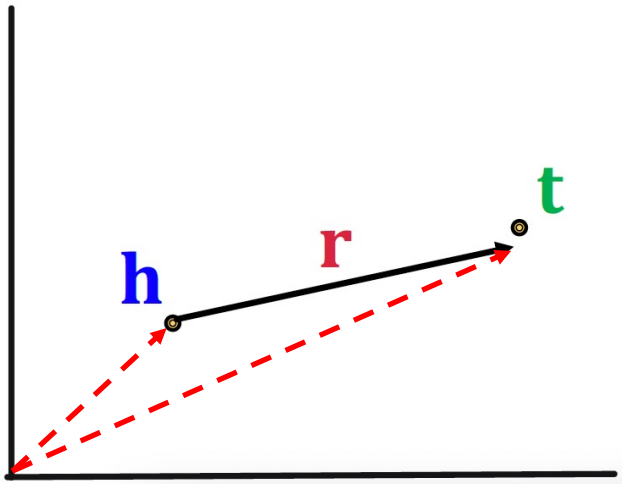
- **Intuition: Translation**

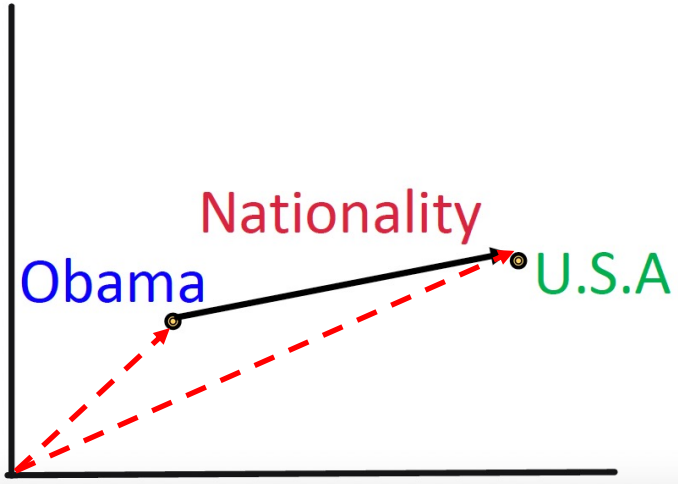
  For a triple (h, r, t), let $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$ be embedding vectors.

  embedding vectors will appear in boldface

- **TransE:** $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given link exists else $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$

Entity scoring function: $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$





SGD on TransE (Bordes et al. 2013)

Algorithm 1 Learning TransE

input Training set $S = \{(h, r, t)\}$, entities and rel. sets E and R, margin γ, embeddings dim. k.

1: **initialize** $r \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $r \in R$
2: $r \leftarrow r / \|r\|$ for each $r \in R$
3: $\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$
4: **loop**
5: $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$
6: $S_{batch} \leftarrow$ sample(S, b) // sample a minibatch of size b
7: $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
8: **for** $(h, r, t) \in S_{batch}$ **do**
9: $(h', r, t') \leftarrow$ sample$(S'_{(h,r,t)})$ // sample a corrupted triplet
10: $T_{batch} \leftarrow T_{batch} \cup \{((h, r, t), (h', r, t'))\}$
11: **end for**
12: Update embeddings w.r.t. $\sum_{((h,r,t),(h',r,t')) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{r}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{r}, \boldsymbol{t'})\right]_+$
13: **end loop**

Initialize relations r and entities e uniformly, then normalize.
γ is the margin.

Sample triplet (h', r, t) that does not appear in the KG.

d represents distance (negative of score)

positive sample

negative sample

Contrastive loss: Favors lower distance (or higher score) for valid triplets, high distance (or lower score) for corrupted ones

2) TransR (Lin et al. 2015)

- **TransE models translation of any relation in the same embedding space.**

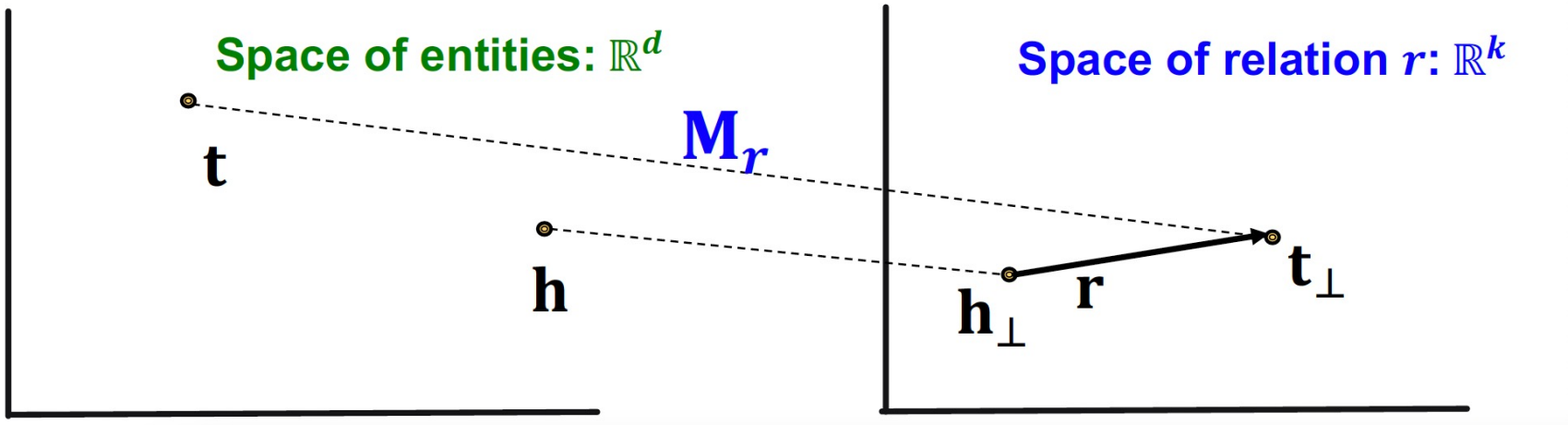
- **Can we design a new space for each relation and do translation in relation-specific space?**

TransR (Lin et al. 2015)

- **TransR**: model **entities** as vectors in the entity space $\mathbb{R}^d$ and model each **relation** as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the **projection matrix**.

Use $\mathbf{M}_r$ to project from entity space $\mathbb{R}^d$ to relation space $\mathbb{R}^k$!

- $\mathbf{h}_\perp = \mathbf{M}_r\mathbf{h},\ \mathbf{t}_\perp = \mathbf{M}_r\mathbf{t}$
- **Score function**: $f_r(h,t) = -||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||$



3) DistMult (Yang et al. 2015)

- **So far:** The scoring function $f_r(h, t)$ is **negative of L1 / L2 distance** in **TransE** and **TransR**

- **Idea: Use bilinear** modeling:

  Score function: $f_r(h, t) = h \cdot A \cdot t$

  $\mathbf{h}, \mathbf{t} \in \mathbb{R}^k, \mathbf{A} \in \mathbb{R}^{k \times k}$

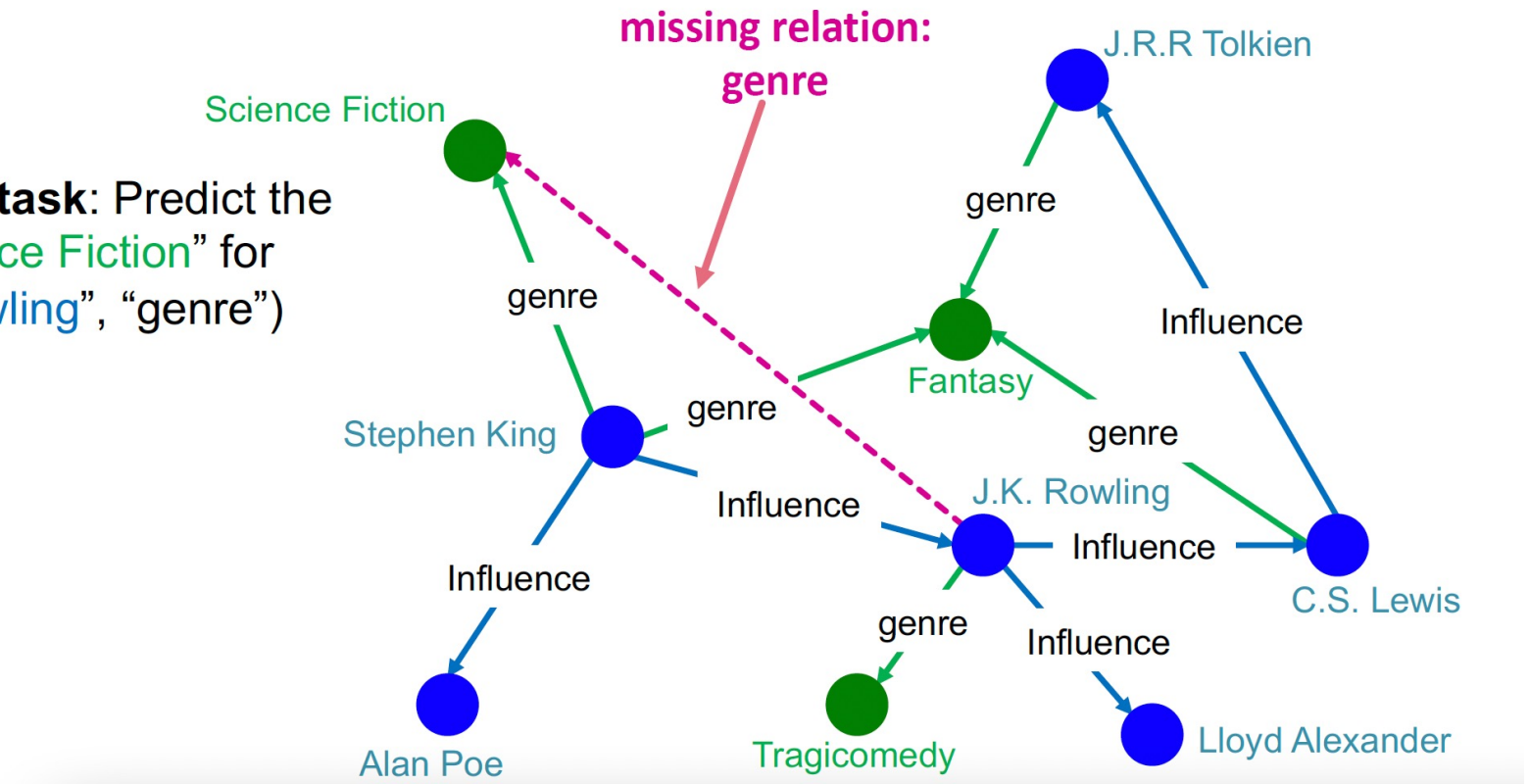
Answering queries (Reasoning) with Knowledge Graphs

Knowledge Graph Completion

Given an enormous KG, can we complete the KG?

- For a given (**head**, **relation**), we predict missing **tails**.

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



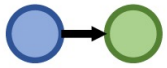
We can use f_r for this prediction, i.e., is $f_r > t$?

Multi-hop reasoning on KGs

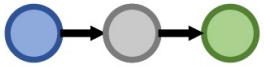
Can we do multi-hop reasoning, i.e., answer complex queries on an incomplete, massive KG?

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))

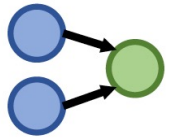
In this lecture, we only focus on answering queries on a KG!
The notation will be detailed next.



One-hop Queries



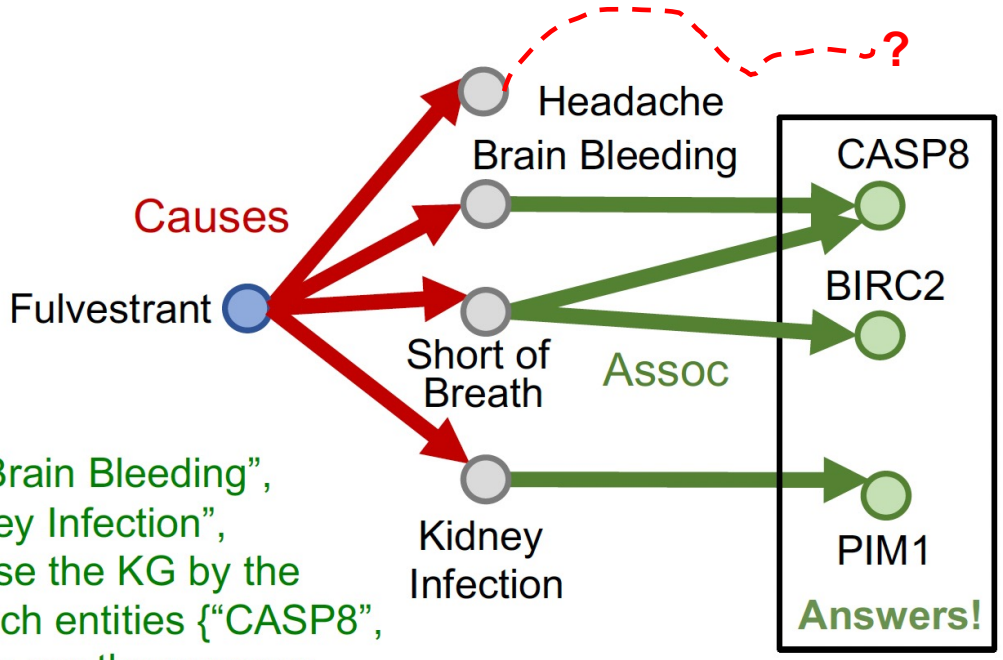
Path Queries



Conjunctive Queries

Traversing the Knowledge Graph

- **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))**



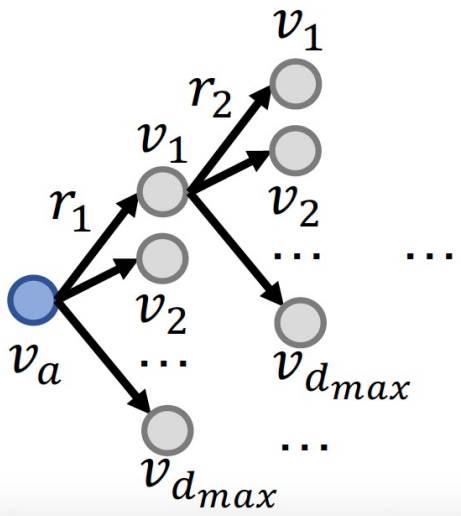
Start from the nodes {"Brain Bleeding", "Short of Breath", "Kidney Infection", "Headache"} and traverse the KG by the relation "Assoc", we reach entities {"CASP8", "BIRC2", "PIM1"}. These are the answers.

However, knowledge graphs are incomplete!

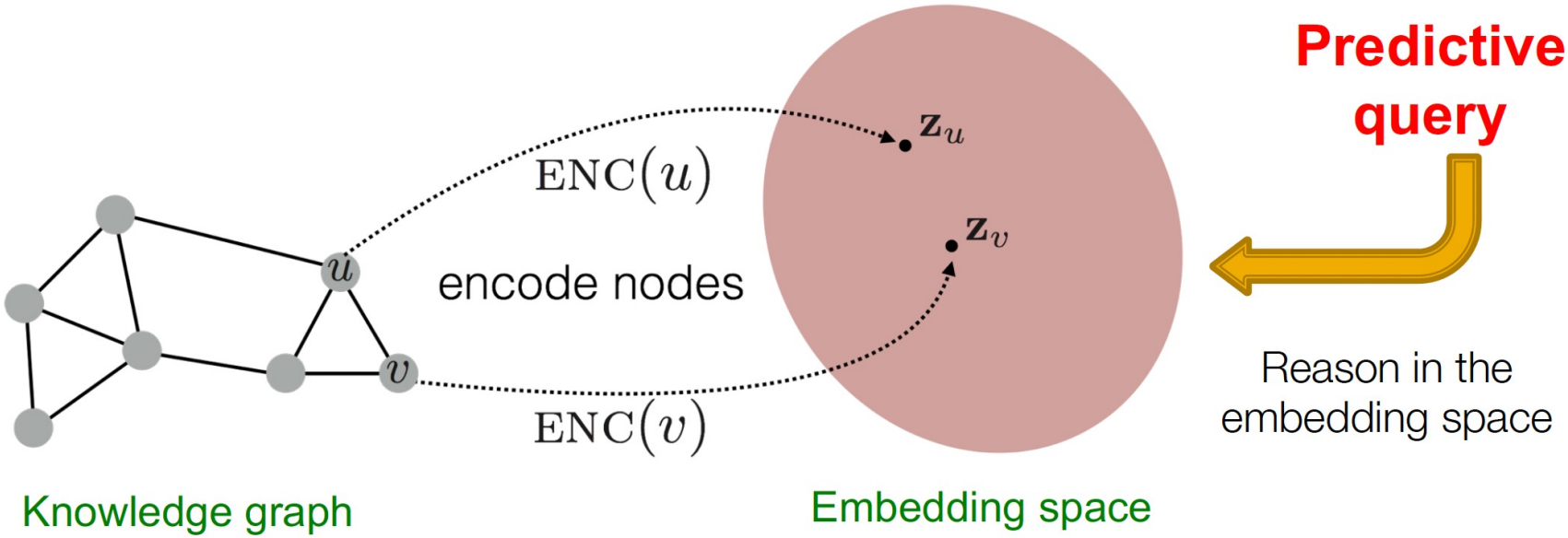
Can KG completion help?

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: $O(d_{max}^L)$



Couple of General Ideas



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

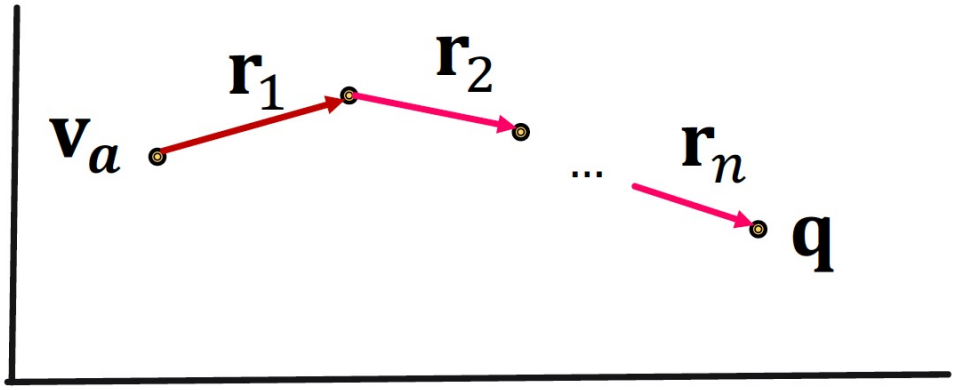
- Embed query into a single **point** in the Euclidean space: answer nodes are close to the query.
- **Query2Box:** Embed query into a hyper-rectangle (**box**) in the Euclidean space: answer nodes are enclosed in the box.

[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]

An Approach to Query Answering with TransE

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

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



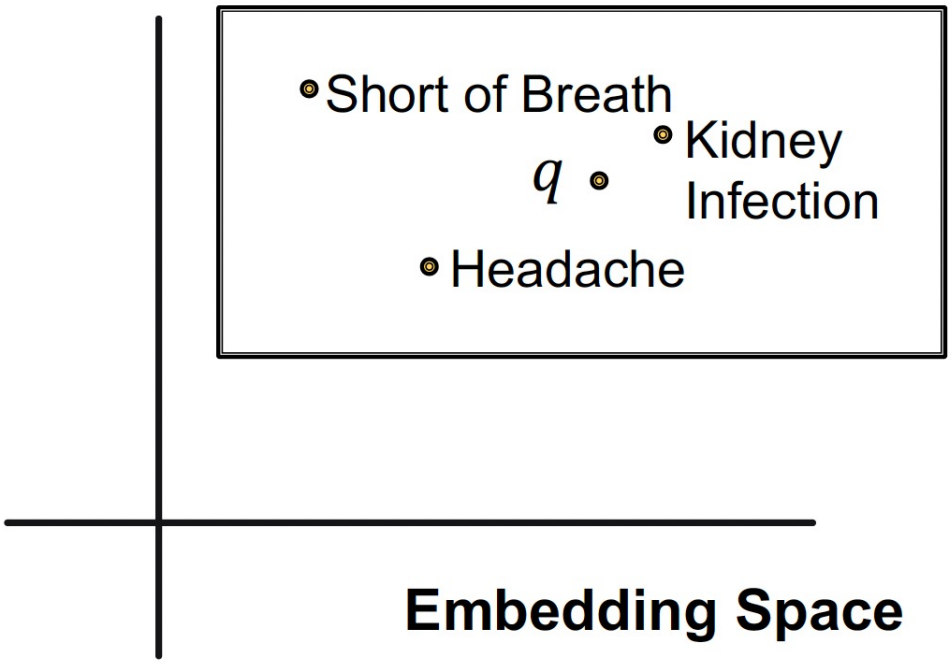
$$\mathbf{q} = \mathbf{v}_a + \mathbf{r}_1 + \cdots + \mathbf{r}_n$$

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

Another approach: Query2Box

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

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



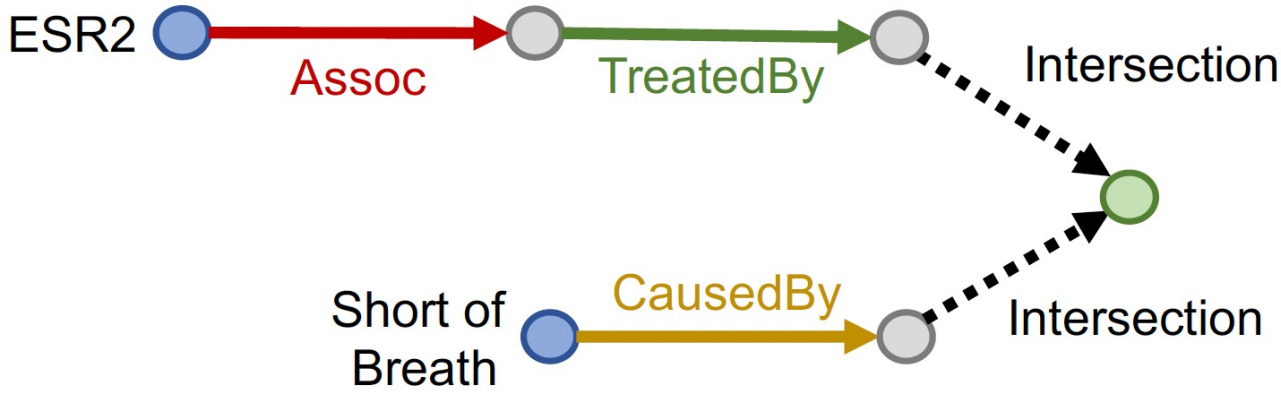
For example, we can embed the adverse events of Fulvestrant with **a box that enclose all the answer entities**.

Query Answering with Query2Box

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

Query plan:

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



(1) Each intermediate node represents a set of entities (box)

Reasoning with Query2Box

- **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**

?

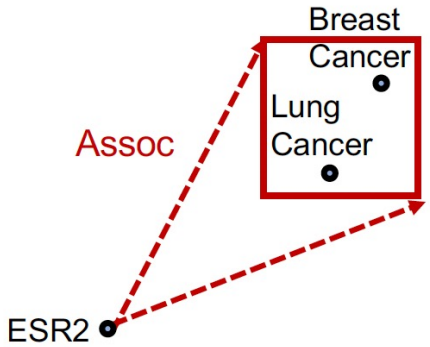
ESR2

Reasoning with Query2Box

- **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



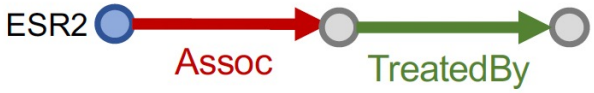
Reasoning with Query2Box

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

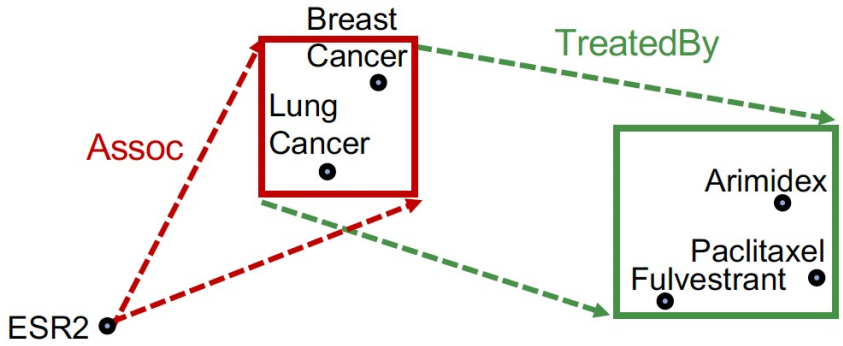
((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))

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

Query Plan



Embedding Space

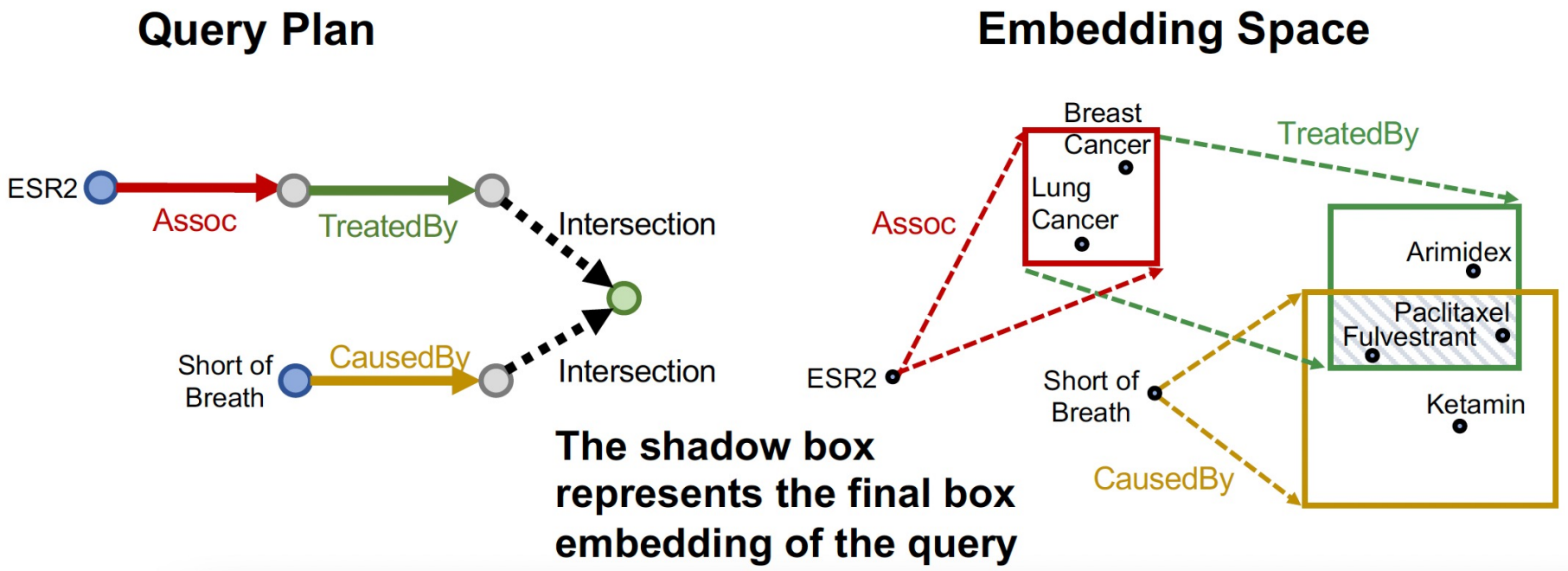


Reasoning with Query2Box

"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**



Summary

- Introduced KG representation learning models with different embedding spaces and expressiveness - TransE / TransR / DistMult / …

- Introduced approaches for answering queries on KGs
  - 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

Thanks!