# DSC250: Advanced Data Mining

## Knowledge Graphs (KGs)

**Zhiting Hu**

Lecture 15, Feb 25, 2025

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

# Outline

- Knowledge graphs

- Presentation
  - Boyang Wang, Zixuan Song: "Inference-Time Intervention: Eliciting Truthful Answers from a Language Model”

  - Benjamin TenWolde, Michael Ko: “Interpreting and Editing Vision-Language Representations to Mitigate Hallucinations”

  - Jay Sawant, Sarthak Kala: “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”

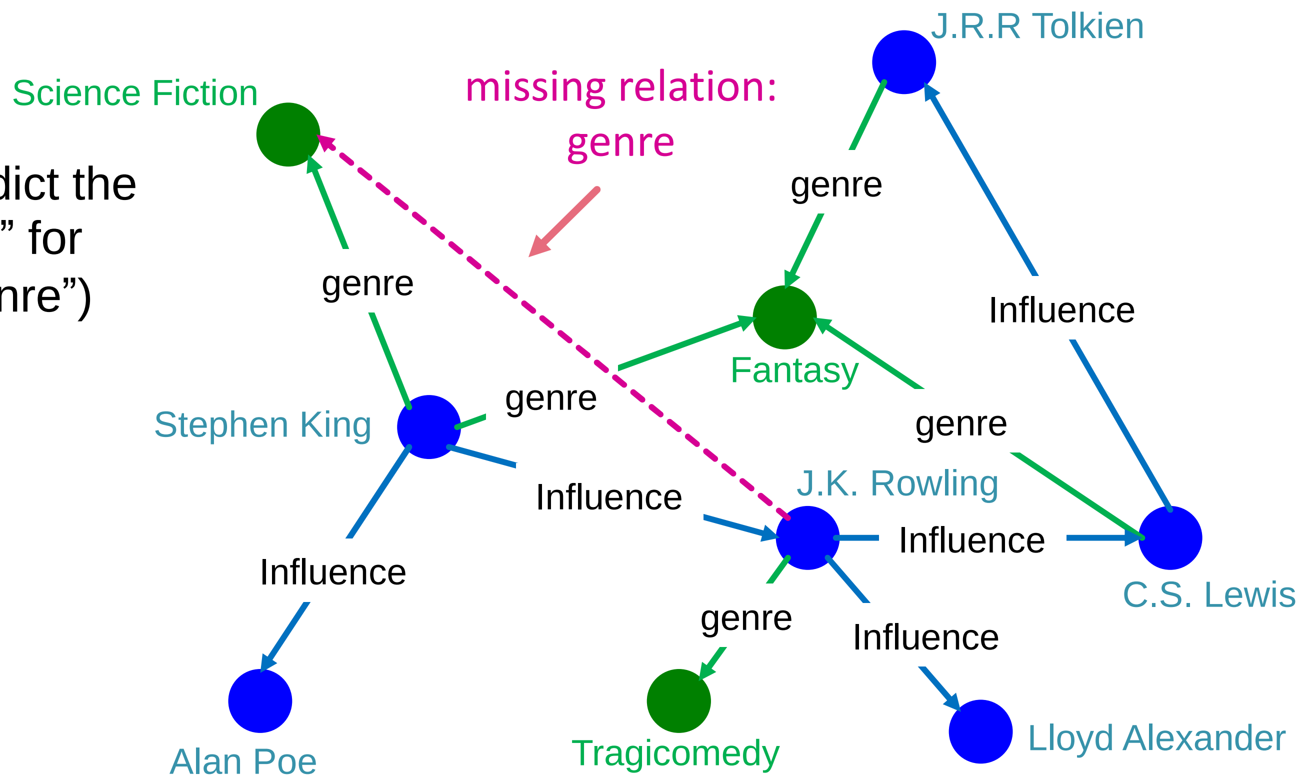
  - Dongting Cai, Zhihan Li: “Music Style Analysis among Haydn, Mozart and Beethoven: an Unsupervised Machine Learning Approach”

# 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")



# Recap: KG Representation

- Edges in KG are represented as **triples** $(h, \$, \%)$
  - head $(h)$ has relation $(\$)$ with tail $(\%)$
- **Key Idea:**
  - Given a triple $(h, \$, \%)$, the goal is that the embedding of $(h, \$)$ **should be close** to the embedding of $\%$
    - How to embed $(h, \#)$?
    - How to define score $\$_\$(h, \%)$?
      - Score $"_!$ is high if $(h, \&, ')$ exists, else $"_!$ is low

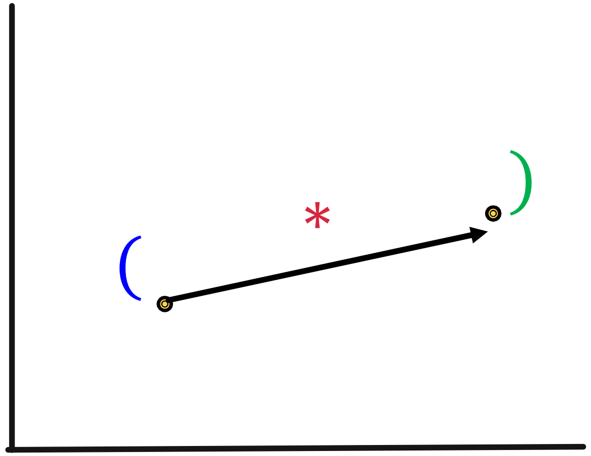
# Recap: TransE for KG Completion

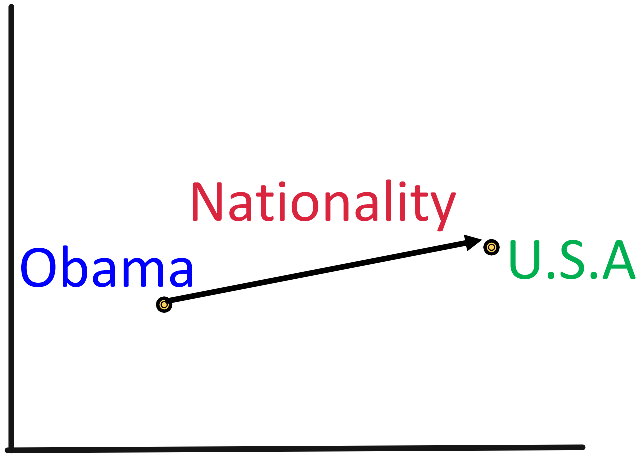
- **Intuition: Translation**

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

  **embedding vectors will appear in boldface**

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




*Bordes et al., Translating embeddings for modeling multi-relational data, NeurIPS 2013.*

# Four Relationship Patterns

- **Symmetric (Antisymmetric) Relations:**

  $$r(h,t) \Rightarrow r(t,h) \;\; (r(h,t) \Rightarrow \neg r(t,h)) \quad \forall h,t$$

  - **Example**:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym (a word with a broader meaning: poodle vs. dog)
- **Inverse Relations:**

  $$r_{(}(h,t) \Rightarrow r_{)}(t,h)$$

  - **Example** : (Advisor, Advisee)
- **Composition (Transitive) Relations:**

  $$r_{)}(x,y) \wedge r_{(}(y,z) \Rightarrow r_{*}(x,z) \quad \forall x,y,z$$

  - **Example**: My mother’s husband is my father.
- **1-to-N relations:**

  $r(h,t_{)}), r(h,t_{(}), \ldots, r(h,t_{+})$ are all True.

  - **Example**: # is “StudentsOf”

# Antisymmetric Relations in TransE

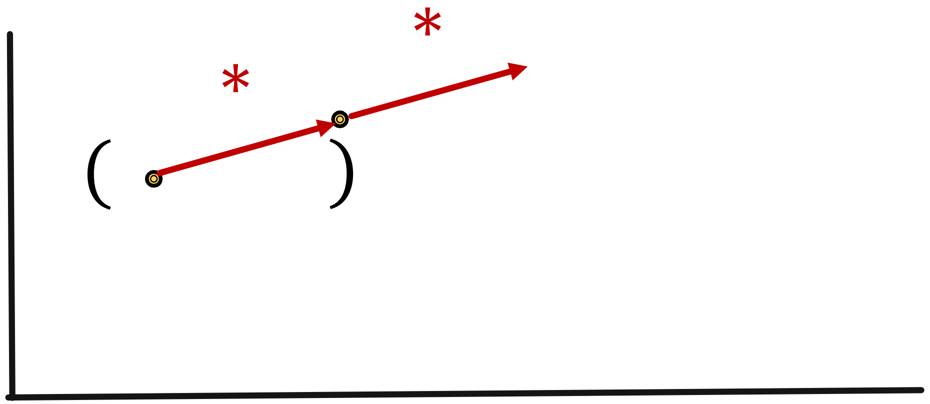
¡ **Antisymmetric Relations:**

$(h , % ⇒ ¬$( % h ) □∀h , %

§ **Example**: Hypernym (a word with a broader meaning: poodle vs. dog)

¡ **TransE** can model antisymmetric relations ü

§ ( + * = ), but ) + * ≠ (

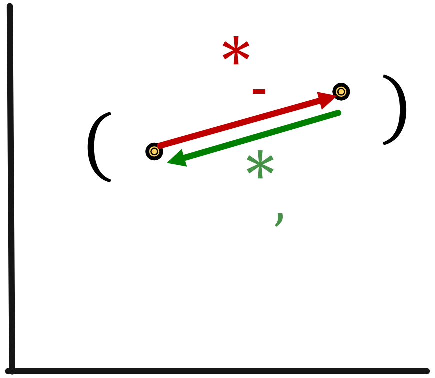


## Inverse Relations in TransE

- **Inverse Relations:**

$$\$_{\#}(h, \%) \Rightarrow \$_{\$}(\%, h)$$

  - **Example** : (Advisor, Advisee)
- **TransE** can model inverse relations ü
  - $( + *_{,} = )$, we can set $*_{)} = -*_{(}$



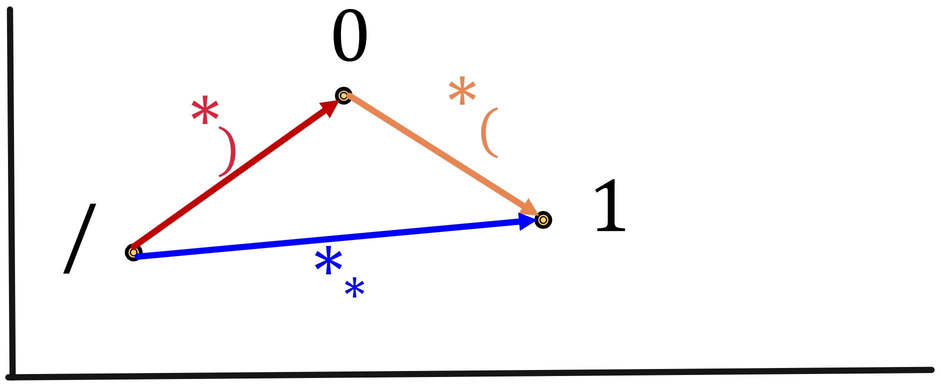
# Composition in TransE

- **Composition (Transitive) Relations:**

  $\$_{\$}(7,8) \wedge \$_{\#}(8,:) \Rightarrow \$_{\%}(7,:)\ \forall 7,8,:$

  - **Example**: My mother's husband is my father.
- **TransE** can model composition relationsü

$$(_{\%} = (_{\$} + (_{\#}$$



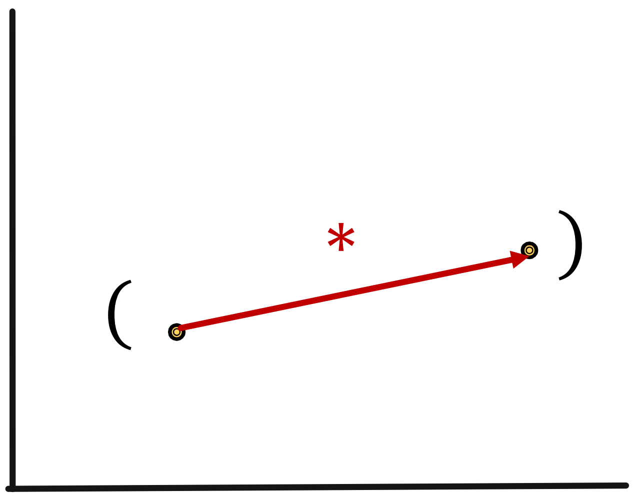
# Limitations of TransE: Symmetric Relations

- **Symmetric Relations:**

$$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$

  - **Example**: Family, Roommate
- **TransE cannot** model symmetric relations ✗

  only if $\mathbf{r} = 0$, $\mathbf{h} = \mathbf{t}$



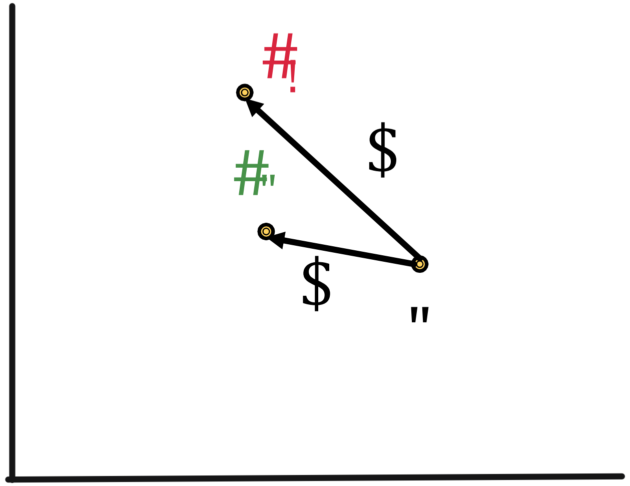
For all $h,t$ that satisfy $r(h,t)$, $r(t,h)$ is also True, which means $||\mathbf{h}+\mathbf{r}-\mathbf{t}|| = 0$ and $||\mathbf{t}+\mathbf{r}-\mathbf{h}|| = 0$. Then $\mathbf{r} = 0$ and $\mathbf{h} = \mathbf{t}$, however $h$ and $t$ are two different entities and should be mapped to different locations.

# Limitations of TransE: 1-to-N Relations

- **1-to-N Relations:**
  - **Example**: $(h, r, t_))$ and $(h, r, t_()$ both exist in the knowledge graph, e.g., $r$ is "StudentsOf"
- **TransE cannot** model 1-to-N relations û
  - **t$_)$** and **t$_($** will map to the same vector, although they are different entities

- $)_\$ = ' + ( = )_\#$
- $)_\$ \neq )_\#$

contradictory!



# KG Completion Methods

| Model | Score | Embedding | Sym. | Antisym. | Inv. | Compos. | 1-to-N |
|---|---|---|---|---|---|---|---|
| TransE | $-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$ | $\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^{!}$ | û | ü | ü | ü | û |
| TransR | $-\|\|\boldsymbol{M}_{"}\,\mathbf{h}+\mathbf{r}$ $-\boldsymbol{M}_{"}\,\mathbf{t}\|\|$ | $\mathbf{h},\mathbf{t}\in\mathbb{R}^{!}$ , $\mathbf{r}\in\mathbb{R}^{\#}$, $\boldsymbol{M}_{"}\in\mathbb{R}^{\#\times !}$ | ü | ü | ü | ü | ü |
| DistMult | $<\mathbf{h},\mathbf{r},\mathbf{t}>$ | $\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^{!}$ | ü | û | û | û | ü |
| ComplEx | $\mathrm{Re}(<\mathbf{h},\mathbf{r},\bar{\mathbf{t}}>)$ | $\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{C}^{!}$ | ü | ü | ü | û | ü |
| | | | | | | | |
| RotateE | $-\|\|\mathbf{h}\circ\mathbf{r}-t\|\|$ | $\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{C}^{!}$ | ü | ü | ü | ü | ü |

# Outline

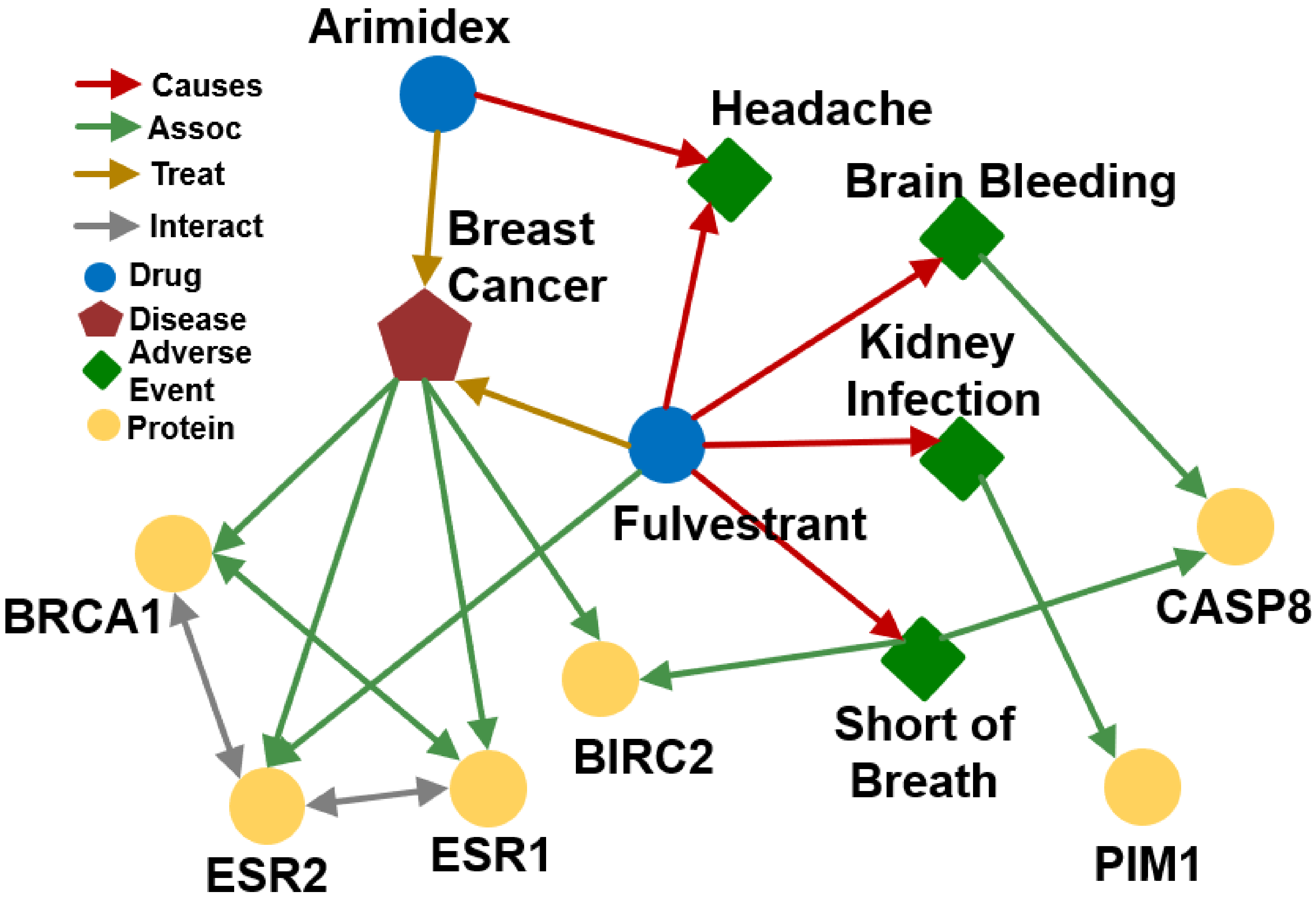
- Overview
- Knowledge Graph Completion (Link Prediction)
- **Reasoning on Knowledge Graphs**

# 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

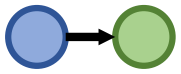
# Example KG: Biomedicine



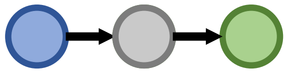
# Predictive Queries on KG

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

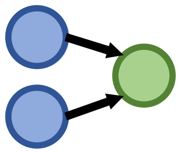
| Query Types | Examples: Natural Language Question, Query |
|---|---|
| **One-hop Queries** | What adverse event is caused by Fulvestrant?<br>(e:Fulvestrant, (r:Causes)) |
| **Path Queries** | What protein is associated with the adverse event caused by Fulvestrant?<br>(e:Fulvestrant, (r:Causes, r:Assoc))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



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 $(h, \$, \%)$ in the KG?

- **One-hop query:** Is $\%$ an answer to query $(h, \$)$?
  - **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

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

  - $v_a$ is an "anchor" entity,
  - Let answers to $q$ in graph $G$ be denoted by $[\![q]\!]_G$.

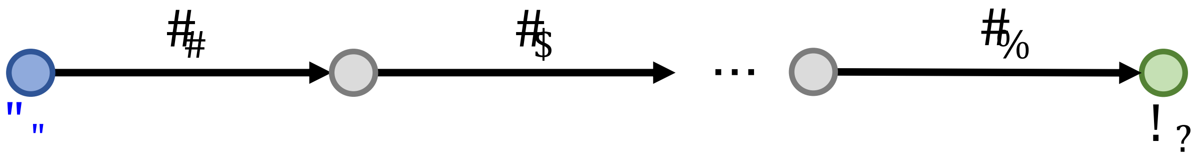
# Path Queries

- Generalize one-hop queries to path queries by **adding more relations on the path**.
- An $n$-hop path query $q$ can be represented by

$$q = (v_{!}, (r_{"}, \dots, r_{\#}))$$

  - $!_{!}$ is an "anchor" entity,
  - Let answers to " in graph \$ be denoted by ⟦"⟧" .

**Query Plan** of $q$:

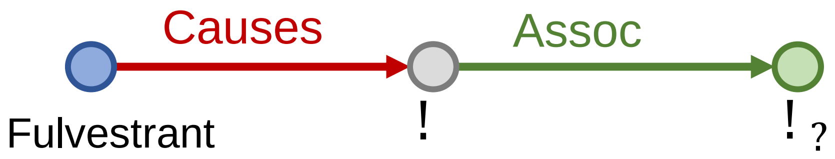


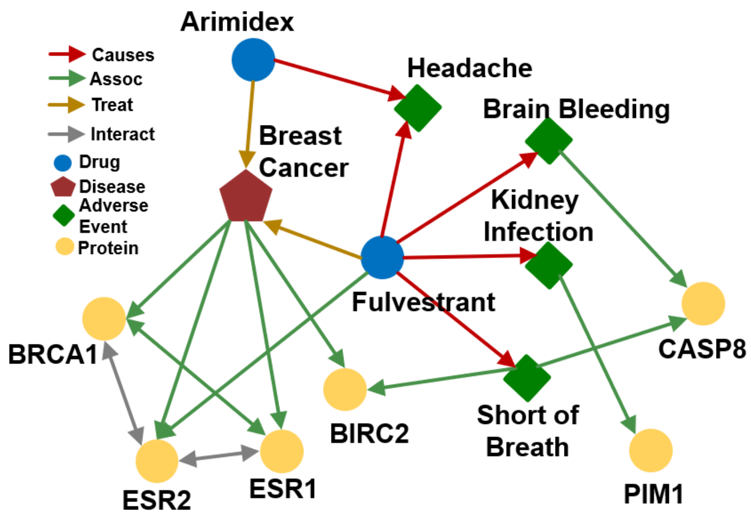
**Query plan of path queries is a chain.**

## Path Queries

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

- *! is **e:Fulvestrant**
- ($', $$) is (**r:Causes**, **r:Assoc**)
- **Query:** (**e:Fulvestrant**, (**r:Causes**, **r:Assoc**))



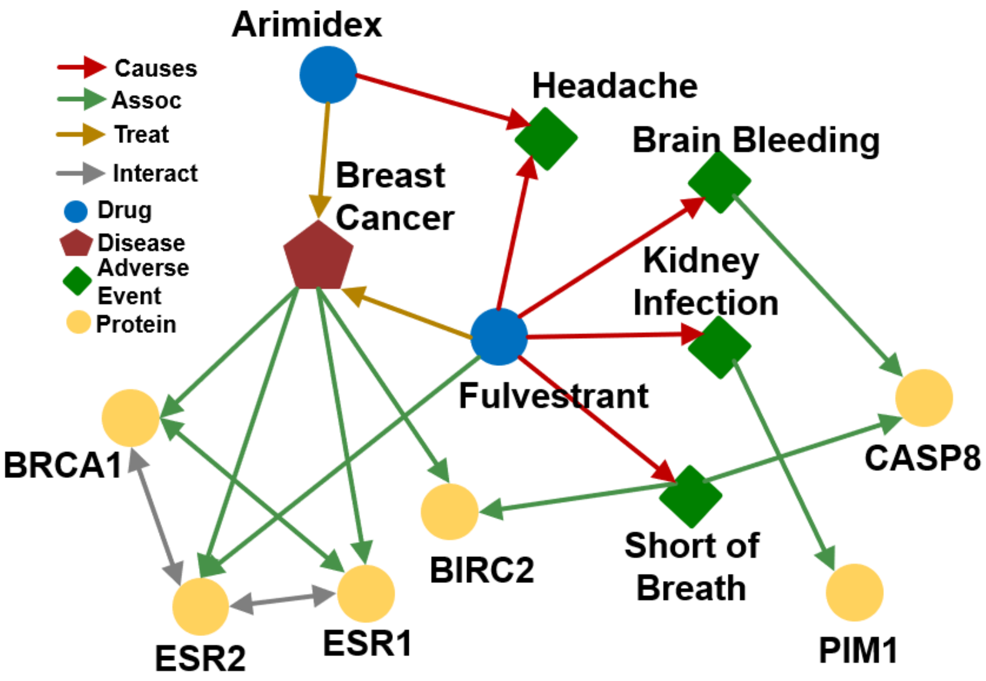



## Path Queries

**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?**



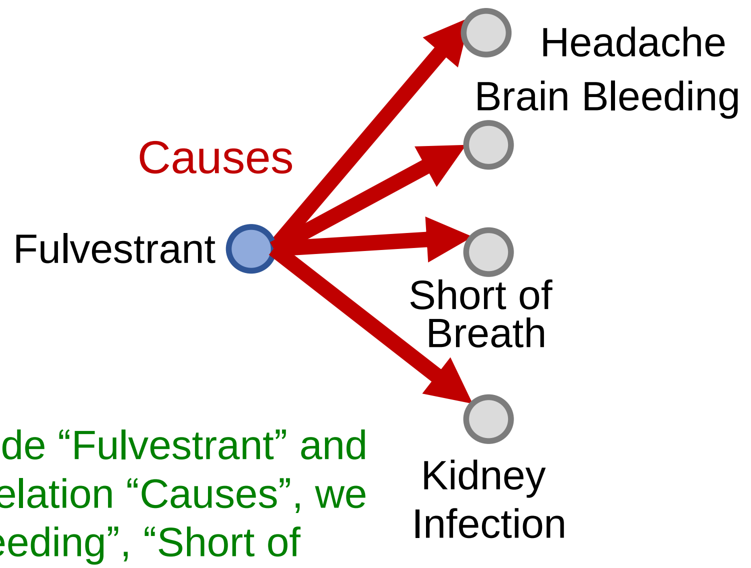
# Traversing Knowledge Graphs

- **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))**


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

# Traversing Knowledge Graphs

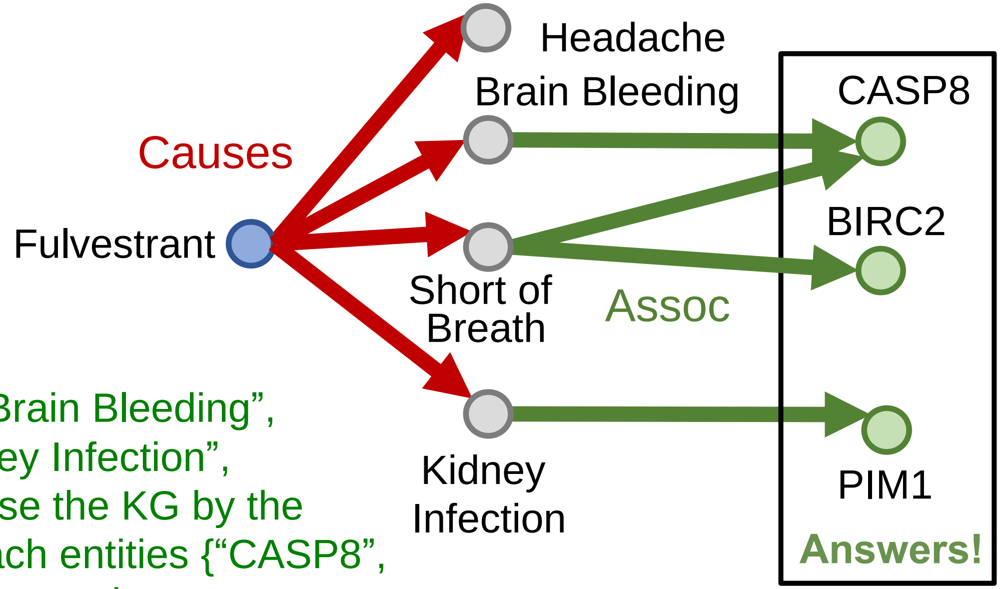
- **We answer path queries by traversing the KG:** *"What proteins are **associated** with adverse events **caused** by **Fulvestrant**?"*
- **Query: (e:Fulvestrant, (r:Causes, r:Assoc))**




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))**




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, KGs are incomplete

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

# 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**

## Can KG Completion Help?

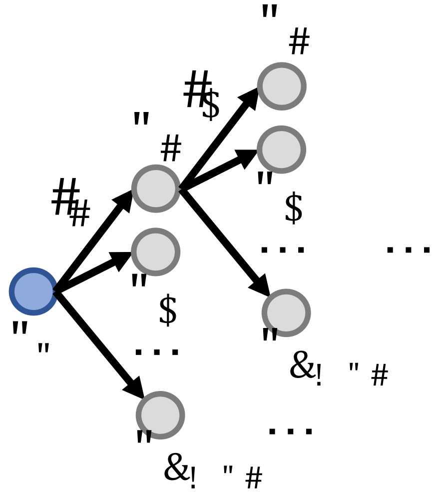
**Can we first do KG completion and then traverse the completed (probabilistic) KG?**

# 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 , : - (. ' %! &)

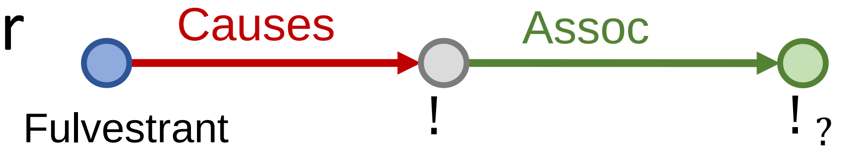


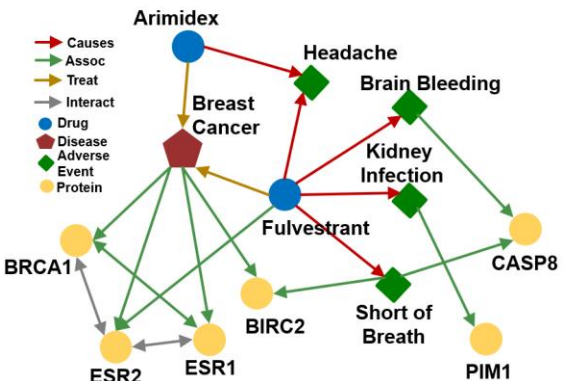
## Task: Predictive Queries

- 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.

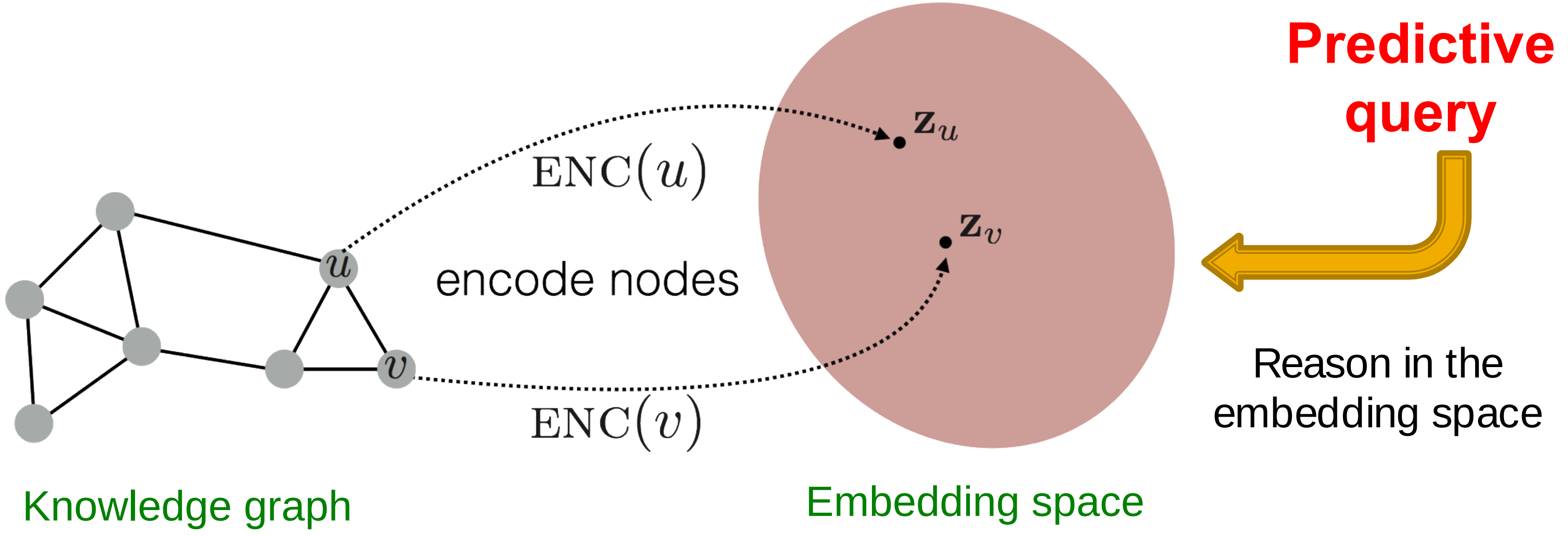
# Task: Predictive Queries

- 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: <u>Predictive queries</u>**
  - Want to be able to answer arbitrary queries while implicitly imputing for the missing information
  - **Generalization of the link prediction task**



# A 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.

[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]

# Traversing KG in Vector Space

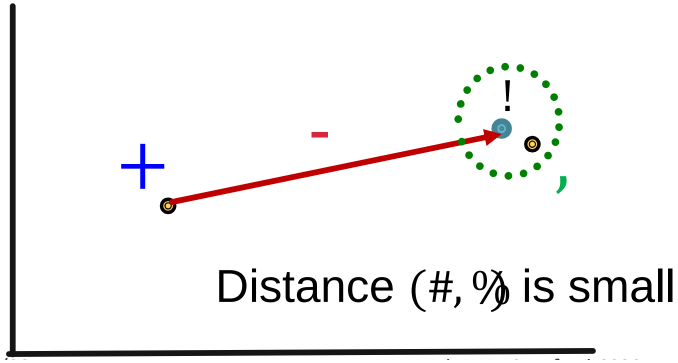
- **Key idea: Embed queries!**
  - Generalize **TransE** to multi-hop reasoning.

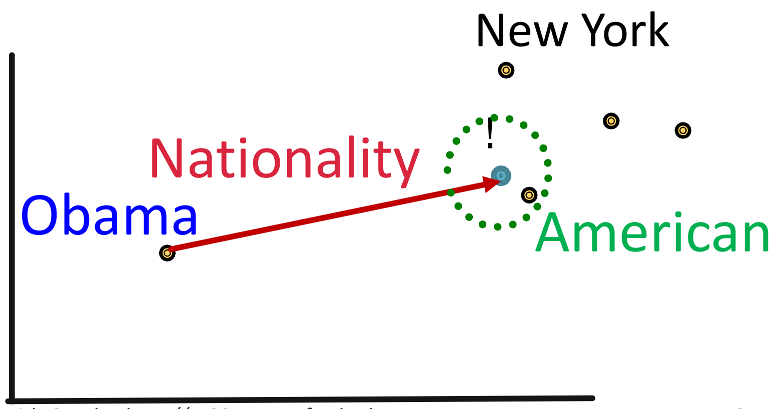
*Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.*

# Traversing KG in Vector Space

- **Key idea: Embed queries!**
  - Generalize **TransE** to multi-hop reasoning.
  - **Recap: TransE:** Translate ! to " using # with score function $ (ℎ, ') = −||! + # − "||.
  - 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** $\mathbf{t}$

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

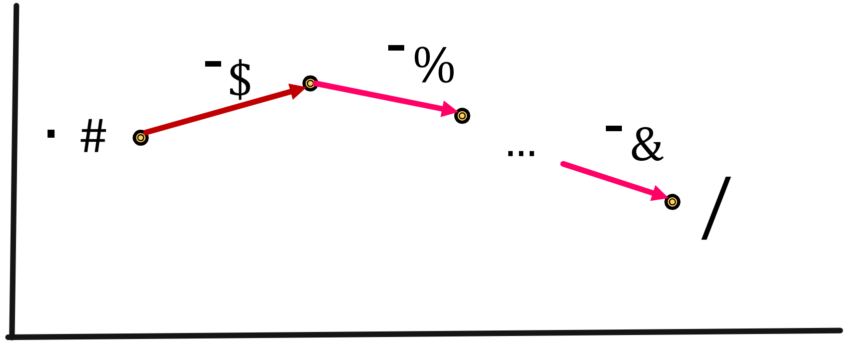


# Traversing KG in Vector Space

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

**Given a path query** $/ = (0_{(}, (1_{)}, \ldots, 1_{*}))$,




$$2 = 3_{!} + 5_{'} + \cdots + 5_{\#}$$

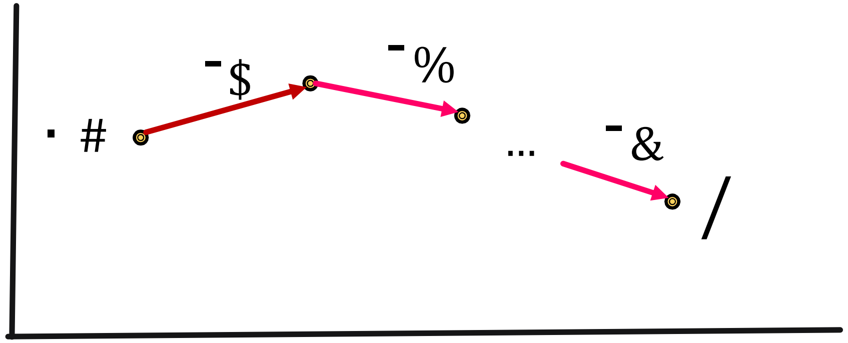
Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.

# Traversing KG in Vector Space

¡ **Key idea: Embed queries!**

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

**Given a path query** $/ = (0_{(}, (1_{)}, \ldots, 1_{*})),$




$$2 = 3_{!} + 5_{'} + \cdots + 5_{\#}$$

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

*Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.*

# Traversing KG in Vector Space

**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:**


Guu, et al., *Traversing knowledge graphs in vector space*, EMNLP 2015.

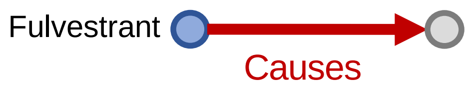
# Traversing KG in Vector Space

**Embed path queries in vector space.**

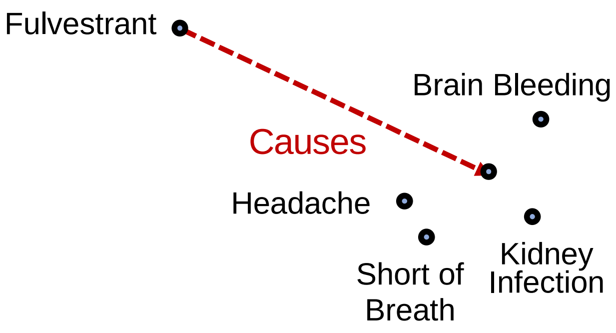
- **Question:** *"What proteins are **associated** with adverse events **caused** by **Fulvestrant**?"*
- **Query:** (e:Fulvestrant, (r:Causes , r:Assoc))

**Follow the query plan:**

**Query Plan**




**Embedding Process**




Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.

# Traversing KG in Vector Space

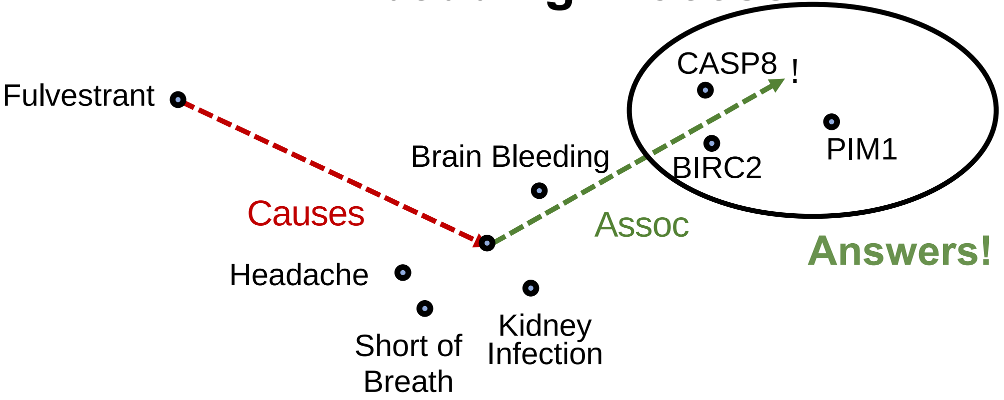
**Embed path queries in vector space.**

- **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**



# Traversing KG in Vector Space

**Insights:**

- We can train **TransE** to optimize knowledge graph completion objective
- 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**.

# Recommender System (RecSys)

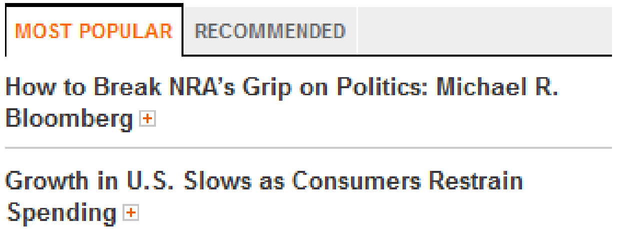
Slides adapted from:

- Y. Sun, CS 247: Advanced Data Mining
- Jure Leskovec, Stanford CS224W: Machine Learning with Graphs

# Recommender System Examples





# Why Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …
- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

# Formulating the RecSys problem (I): Matrix Completion

| Users | Movie1 | Movie2 | Movie3 | Movie4 | Movie5 | Movie6 | $\cdots$ |
|---|---|---|---|---|---|---|---|
| User1 | ? | ? | 4 | ? | 1 | ? | $\cdots$ |
| User2 | 2 | 5 | 2 | ? | ? | 2 | $\cdots$ |
| User3 | ? | ? | 5 | 3 | 2 | 4 | $\cdots$ |
| User4 | 1 | ? | ? | 4 | ? | ? | $\cdots$ |
| User5 | 2 | 3 | ? | ? | ? | ? | $\cdots$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |

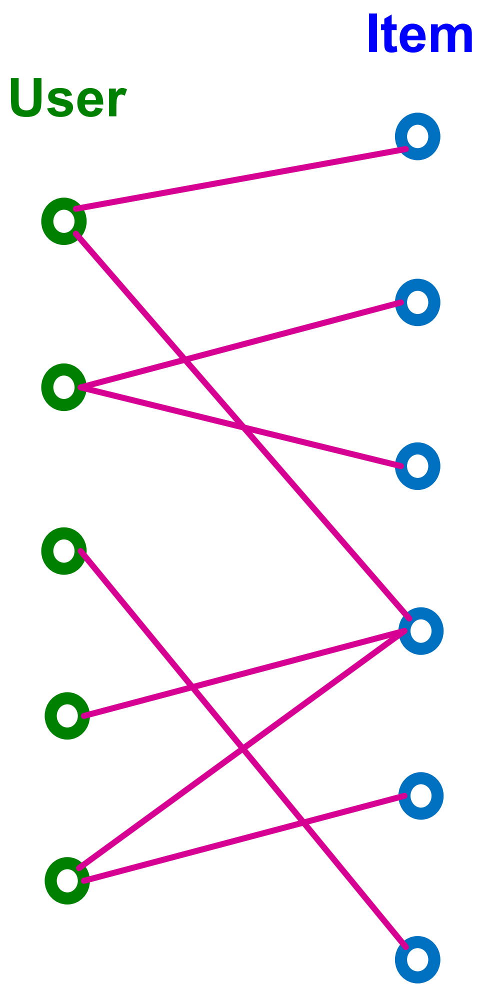
# Explicit Feedback vs. Implicit Feedback

- Explicit Feedback
  - Know the ratings
- Implicit Feedback
  - only know whether user and item has interacted
  - Like (1) vs. unknown (0)

| | | Items | | | | |
|---|---|---|---|---|---|---|
| Users | |  |  |  |  | ... |
| | Alice | 1 | 1 | 0 | 0 | |
| | Bob | 0 | 0 | 1 | 1 | |
| | Corey | 1 | 0 | 1 | 0 | |
| | ... | | | | | |

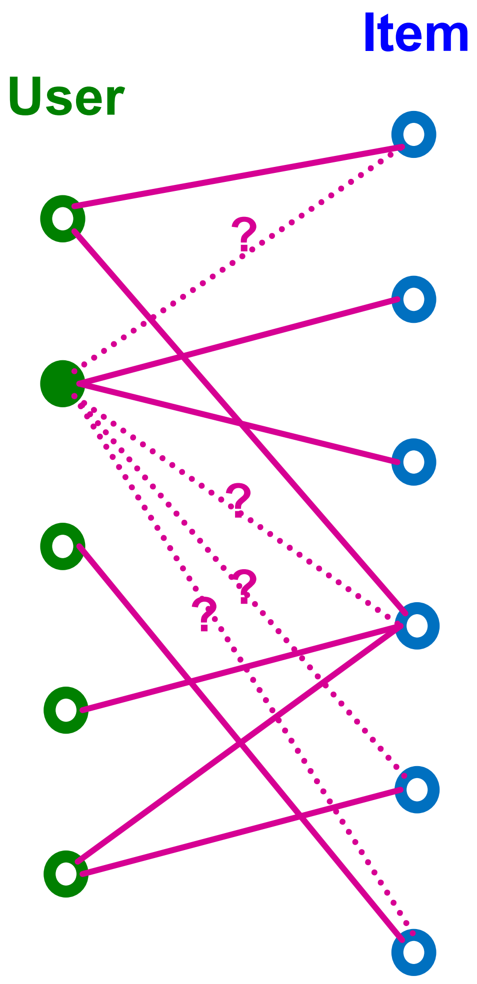
# Formulating the RecSys problem (II): Link Prediction

- Recommender system can be naturally modeled as a **bipartite graph**
  - A graph with two node types: **users** and **items**.
  - **Edges** connect users and items
    - Indicates user-item interaction (e.g., click, purchase, review etc.)
    - Often associated with timestamp (timing of the interaction).



# Formulating the RecSys problem (II): Link Prediction

- **Given**
  - Past user-item interactions
- **Task**
  - Predict new items each user will interact in the future.
  - Can be cast as **link prediction** problem.
    - Predict new user-item interaction edges given the past edges.
  - For ! ∈ #, % ∈ &, we need to get a real-valued **score** ' (! , %).



# Modern Recommender System

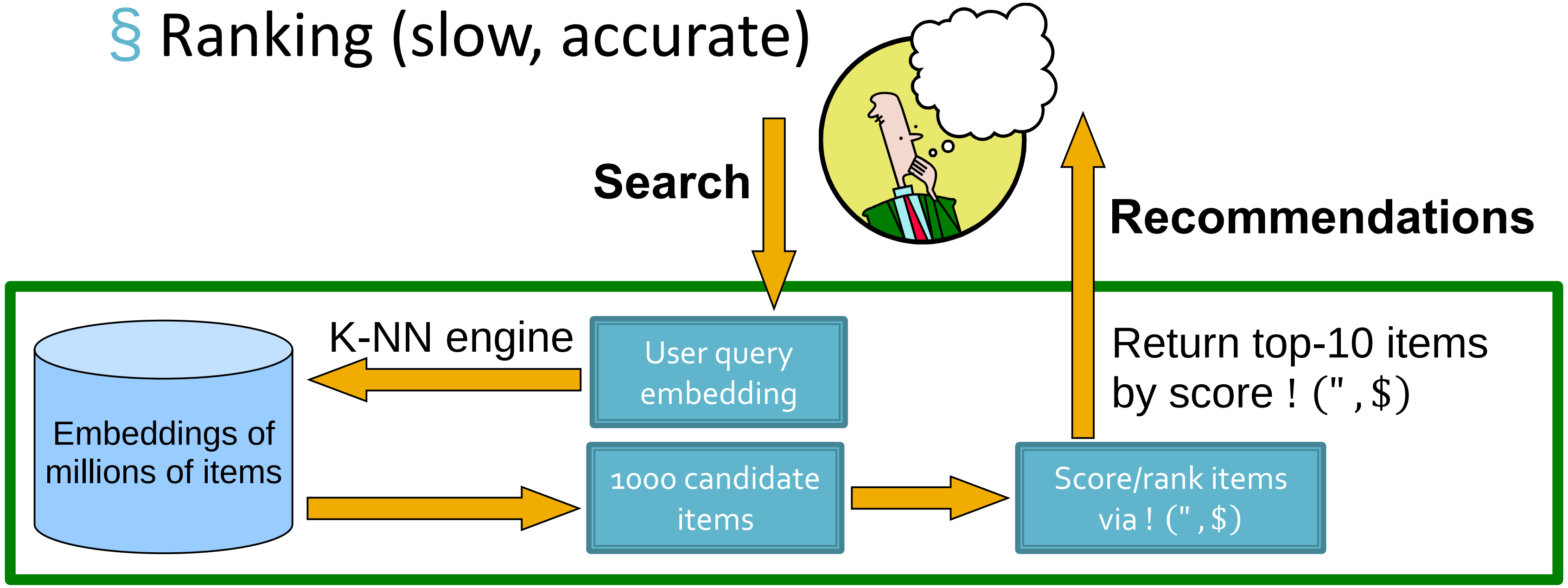
- **Problem:** Cannot evaluate ! (" , $) for every user " – item $ pair.

Example ! (" , $):
! (" , $) = & · &

# Modern Recommender System

- **Problem:** Cannot evaluate ! (" , $) for every user " – item $ pair.
- **Solution:** 2-stage process:
  - Candidate generation (cheap, fast)
  - Ranking (slow, accurate)

Example ! (" , $):
! (" , $) = & · &



## Top-K Recommendation

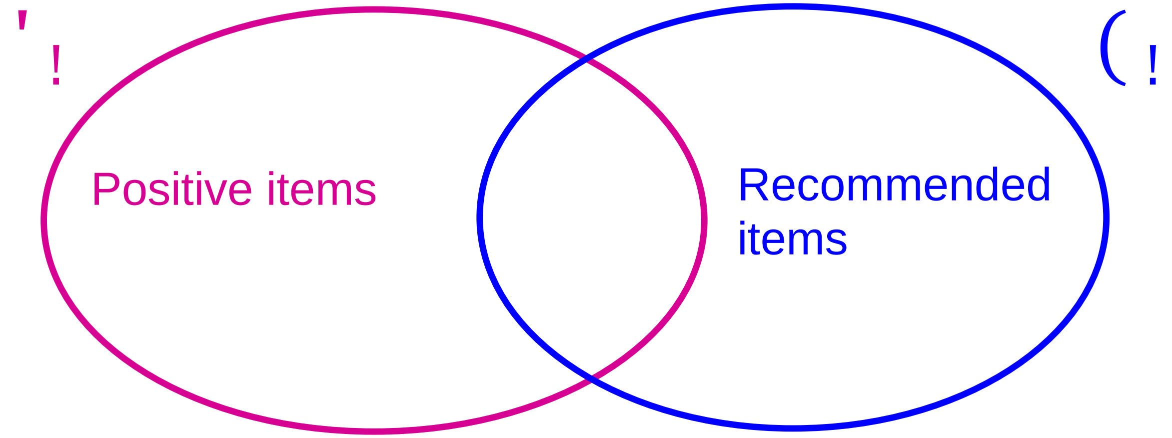
- For each user, we recommend % items.
  - **For recommendation to be effective, * needs to be much smaller than the total number of items (up to billions)**
  - + is typically in the order of 10—100.

## Top-K Recommendation

- For each user, we recommend % items.
  - **For recommendation to be effective, * needs to be much smaller than the total number of items (up to billions)**
  - + is typically in the order of 10—100.
- The goal is to include as many **positive items** as possible in the top-% recommended items.
  - **Positive items = Items that the user will interact with in the future.**
- **Evaluation metric**: Recall@% (defined next)

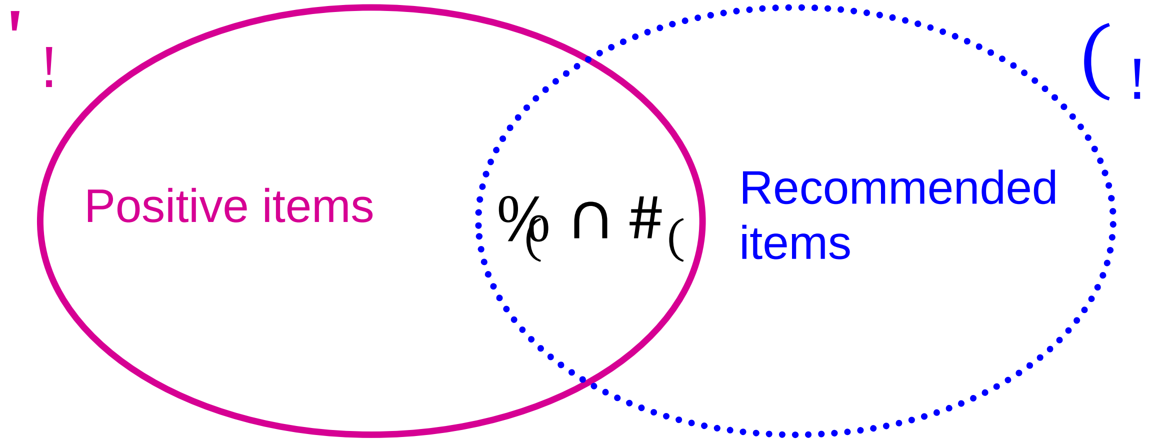
# Evaluation Metric: Recall@K

- **For each user &,**
  - Let ,! be a set of positive items the user will interact in the future.
  - Let -! be a set of items recommended by the model.
    - In top-! recommendation, |#( | = ! .
    - Items that the user has already interacted are excluded.



## Evaluation Metric: Recall@K

- **Recall@) for user & is |* " ∩ , " |/|* " |.**
  - Higher value indicates more positive items are recommended in top-+ for user ! .




- The final Recall@% is computed by averaging the recall values across all users.

# Methods

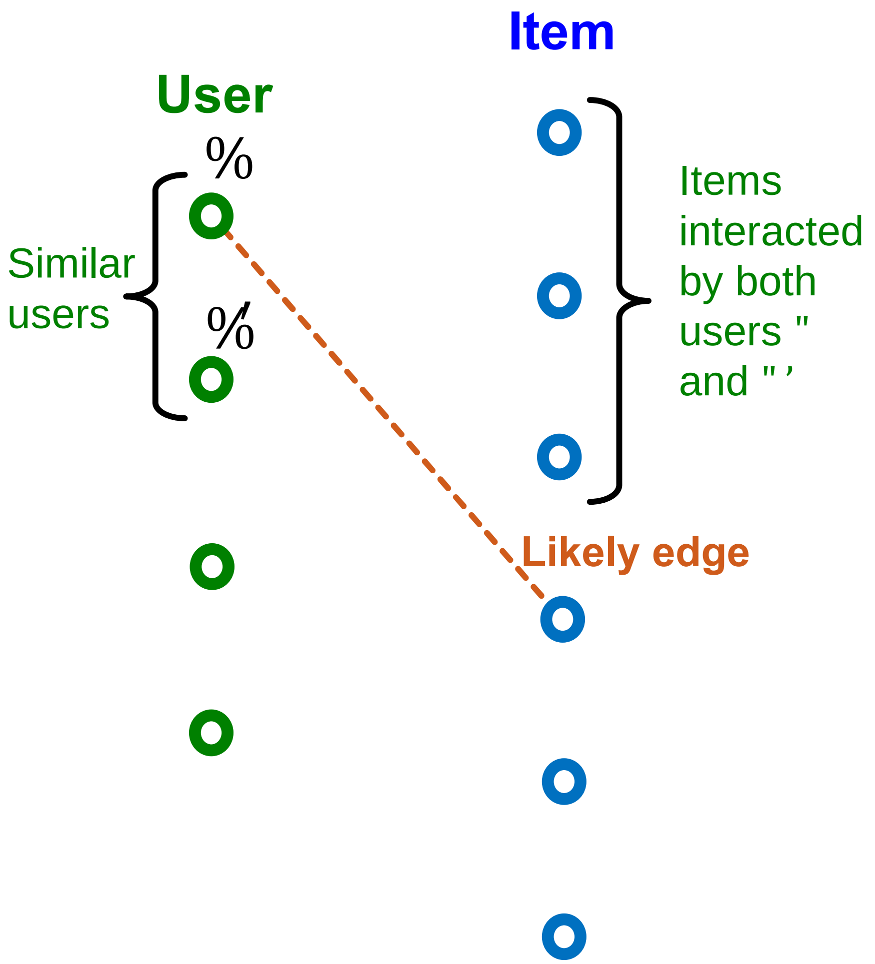
- Collaborative filtering
- Content-based recommendation
- Hybrid methods

# Methods

- **Collaborative filtering**
- Content-based recommendation
- Hybrid methods

# Collaborative Filtering (CF)

- **Underlying idea: Collaborative filtering**
  - Recommend items for a user by **collecting preferences of many other similar users.**
  - **Similar users tend to prefer similar items.**
- **Key question: How to capture similarity between users/items?**



# Questions?