DSC250: Advanced Data Mining

Knowledge Graphs
Recommeder System

Zhiting Hu

Lecture 16, November 21, 2023

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Logistics

- Zhiting's Office Hour this week:
  - Wed, Nov.22, 11:30am
  - Friday Nov.24: Thanksgiving holidays

Outline

- Knowledge Graphs
- Recommender System

- 5 paper presentations
  - Trevor Tuttle, Cyril Gorlla
  - Baraa Zekeria
  - Shobhit Dronamraju, Morgan Kelley
  - Shreya Pakala, Tejo Nandini Baddula
  - Ananay Sharma, Swapnil Ghosh

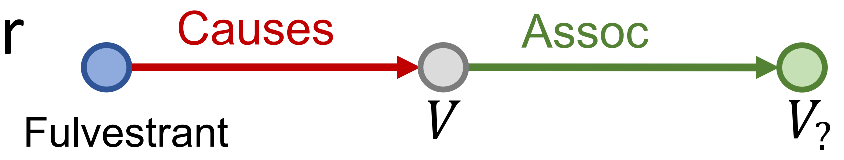
Knowledge Graphs (KGs)

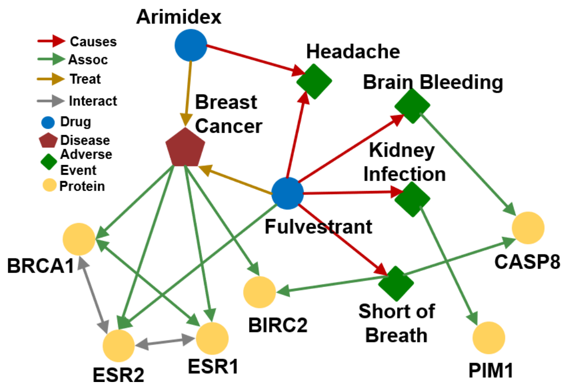
Slides adapted from:

- Jure Leskovec, Stanford CS224W: Machine Learning with Graphs

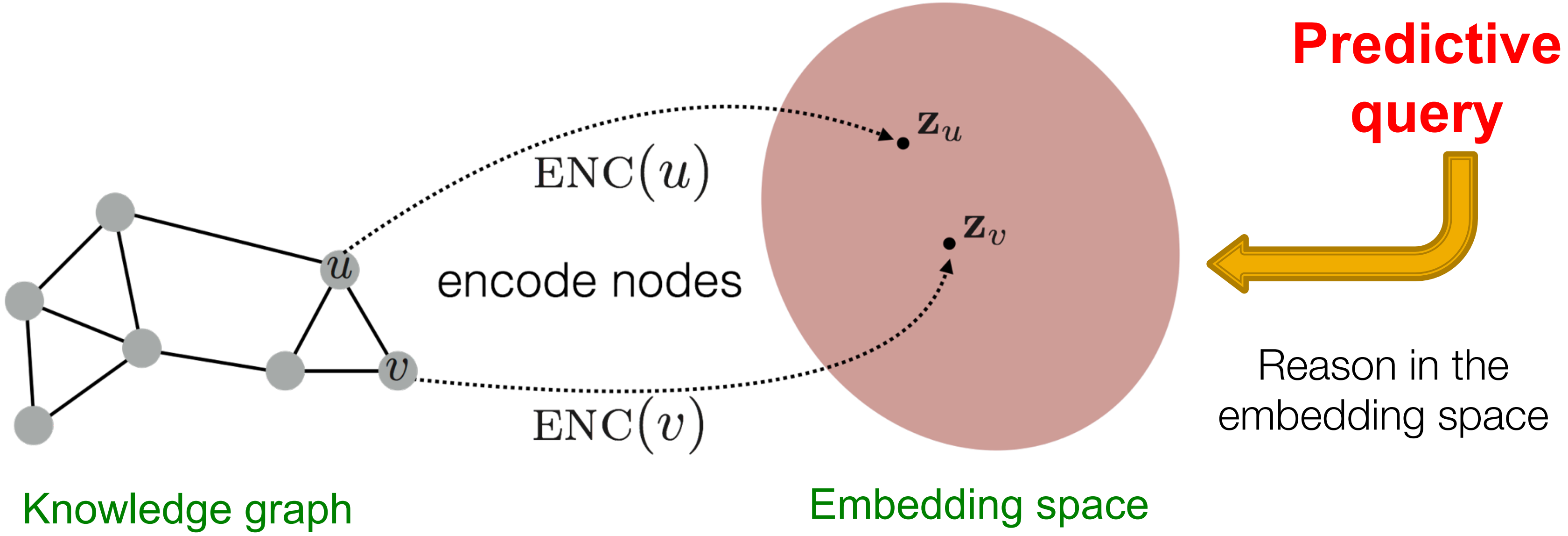
Recap: Task: Predictive Queries

- 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**





Recap: A General Idea



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

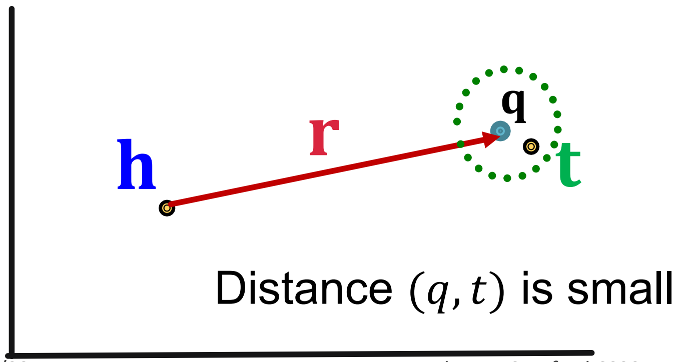
- Embed query into a single **point** in the Euclidean space: answer nodes are close to the query.
- **Query2Box:** Embed query into a hyper-rectangle (**box**) in the Euclidean space: answer nodes are enclosed in the box.

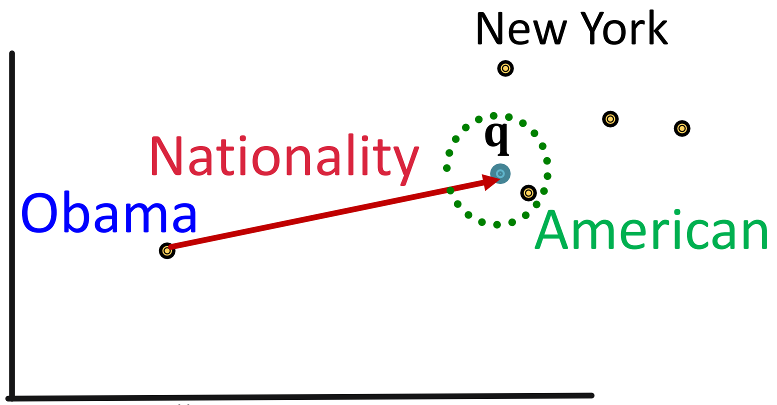
[Embedding Logical Queries on Knowledge Graphs. Hamilton, et al., NeurIPS 2018]
[Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. Ren, et al., ICLR 2020]

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 $f_r(h,t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$.
  - 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** $\mathbf{t}$

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



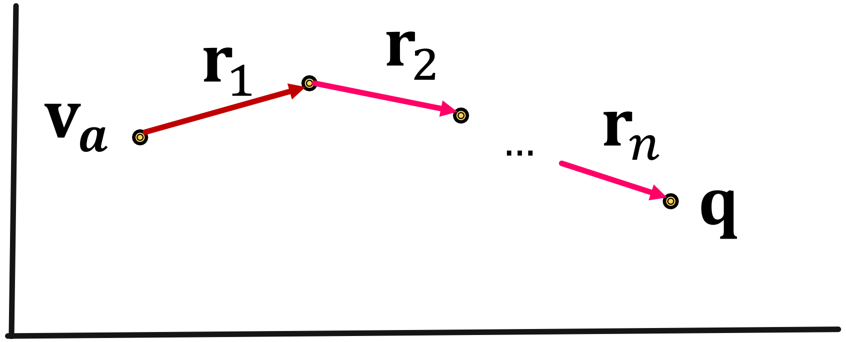


Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.

Traversing KG in Vector Space

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

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



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

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

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.

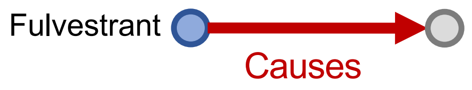
Traversing KG in Vector Space

Embed path queries in vector space.

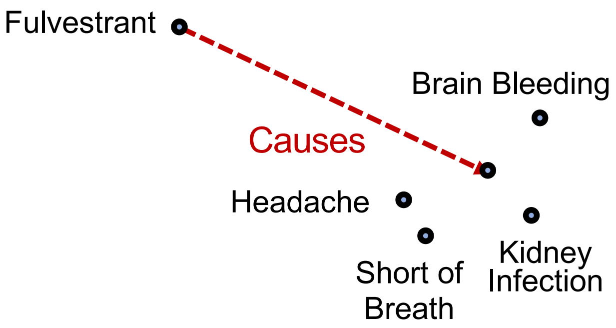
- **Question:** "*What proteins are* ***associated*** *with adverse events* ***caused*** *by* ***Fulvestrant****?*"
- **Query:** (**e:Fulvestrant**, (**r:Causes** , **r:Assoc**))

Follow the query plan:

Query Plan



Embedding Process



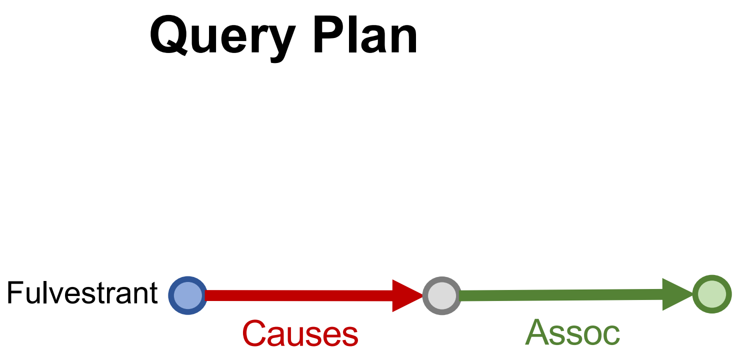
Guu, et al., Traversing knowledge graphs in vector space, EMNLP 2015.

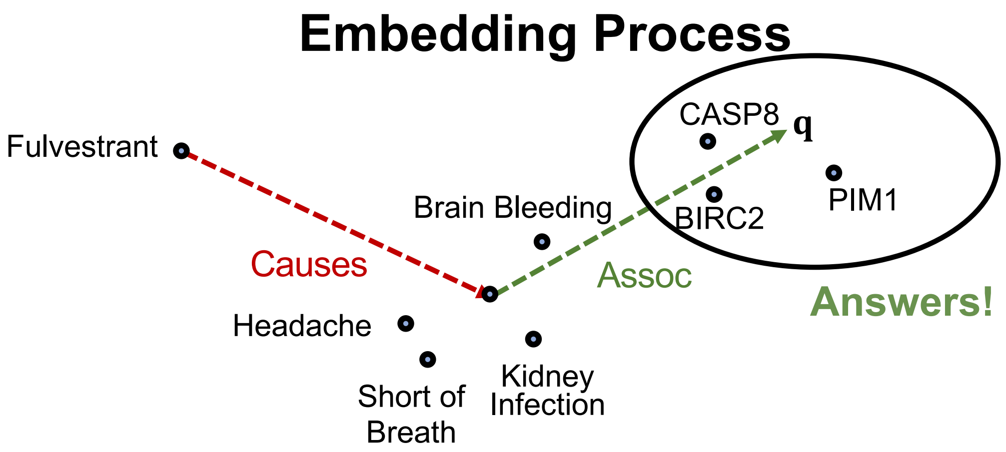
Traversing KG in Vector Space

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

Insights:

- We can train **TransE** to optimize knowledge graph completion objective (Lecture 11)

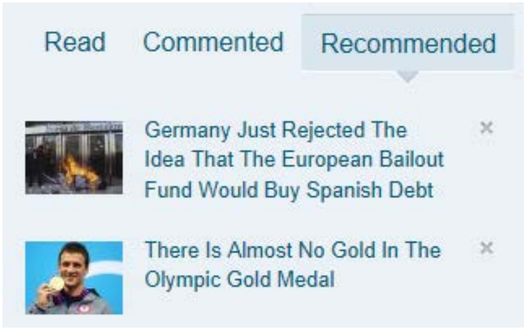
- Since **TransE** can naturally handle **compositional relations**, it can handle path queries by translating in the latent space **for multiple hops using addition of relation embeddings**.

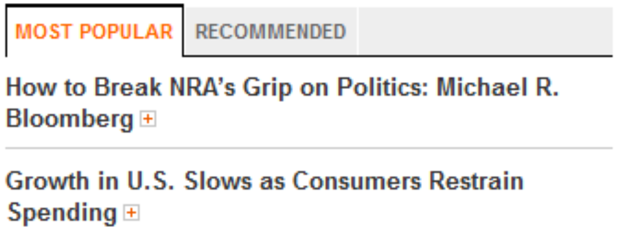
Recommender System (RecSys)

Slides adapted from:

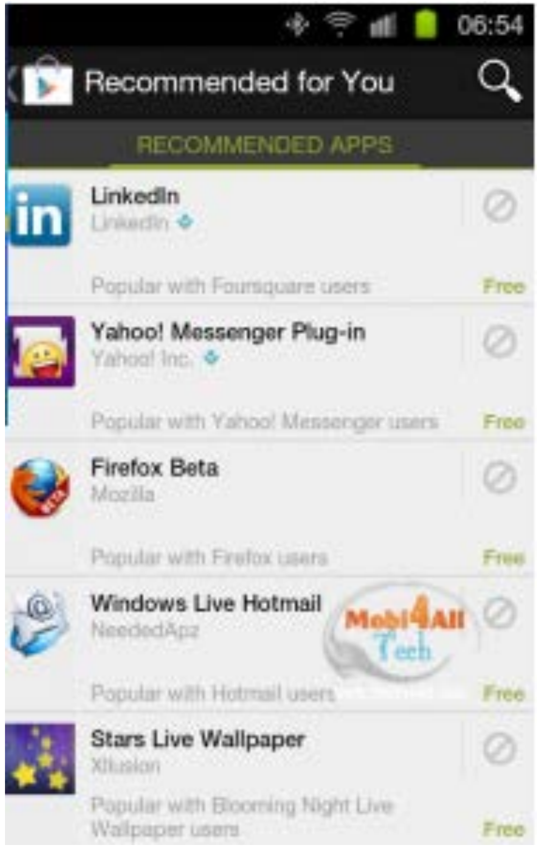
- Y. Sun, CS 247: Advanced Data Mining
- Jure Leskovec, Stanford CS224W: Machine Learning with Graphs

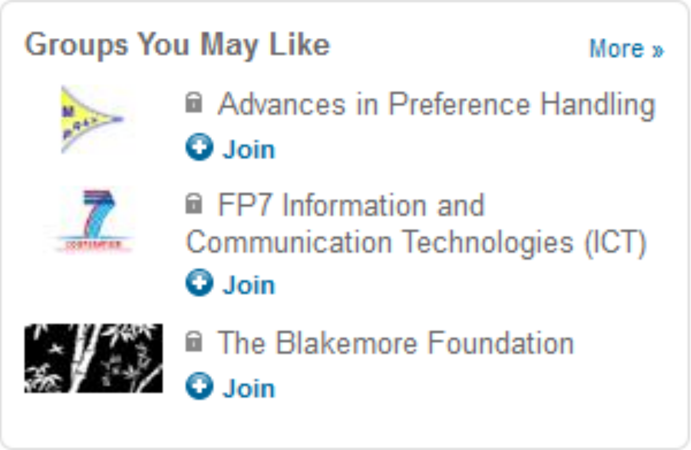
Recommender System Examples

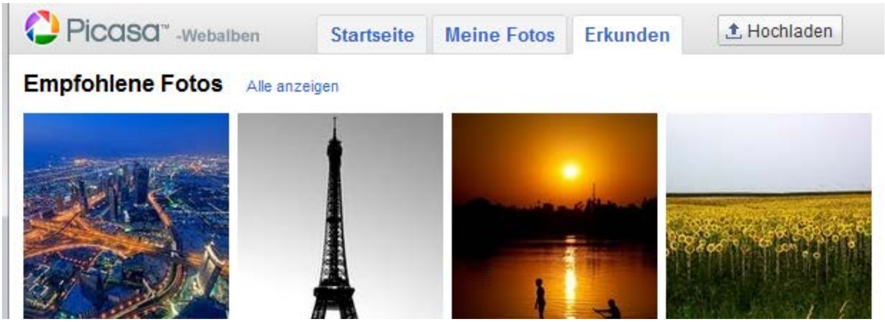




Recommender System Examples







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
  - …

Recommendation as Matrix Completion

Users	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	⋯
User1	?	?	4	?	1	?	⋯
User2	2	5	2	?	?	2	⋯
User3	?	?	5	3	2	4	⋯
User4	1	?	?	4	?	?	⋯
User5	2	3	?	?	?	?	⋯
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

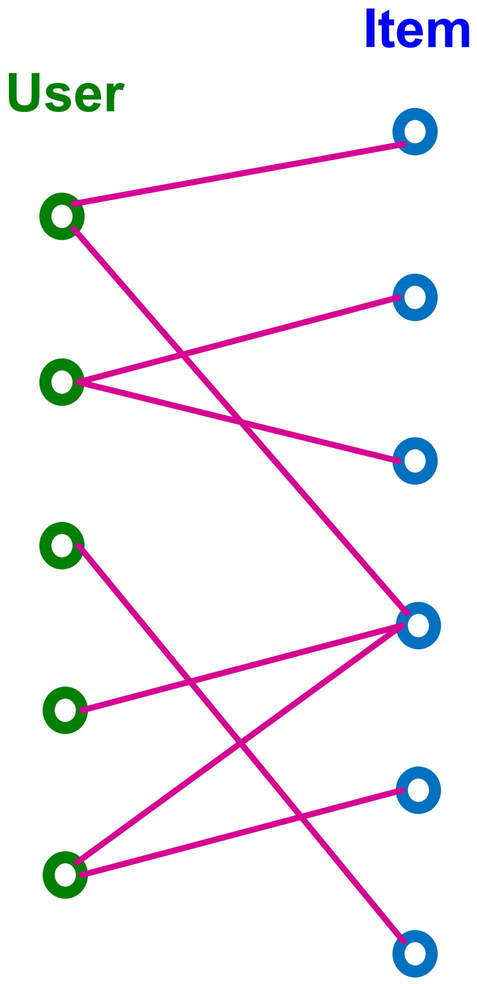
Explicit Feedback vs. Implicit Feedback

- 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	
	...					

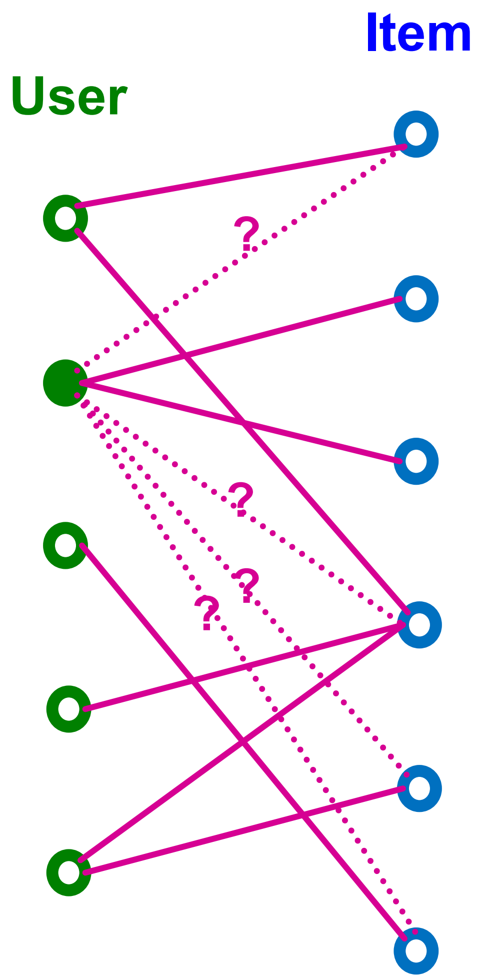
Recommendation as Link Prediction

- 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).



Recommendation as Link Prediction

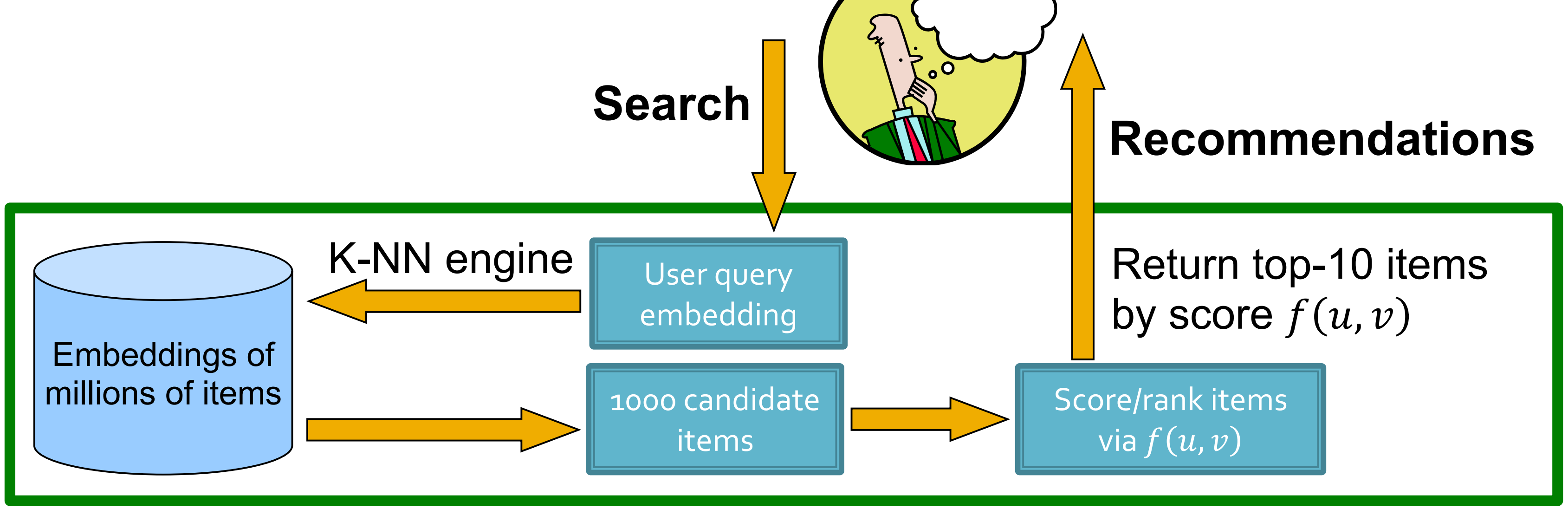
- **Given**
  - Past user-item interactions
- **Task**
  - Predict new items each user will interact in the future.
  - Can be cast as **link prediction** problem.
    - Predict new user-item interaction edges given the past edges.
  - For $u \in \boldsymbol{U}, v \in \boldsymbol{V}$, we need to get a real-valued **score** $f(u, v)$.



Modern Recommender System

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

Example $f(u, v)$:
$f(u, v) = z_u \cdot z_v$

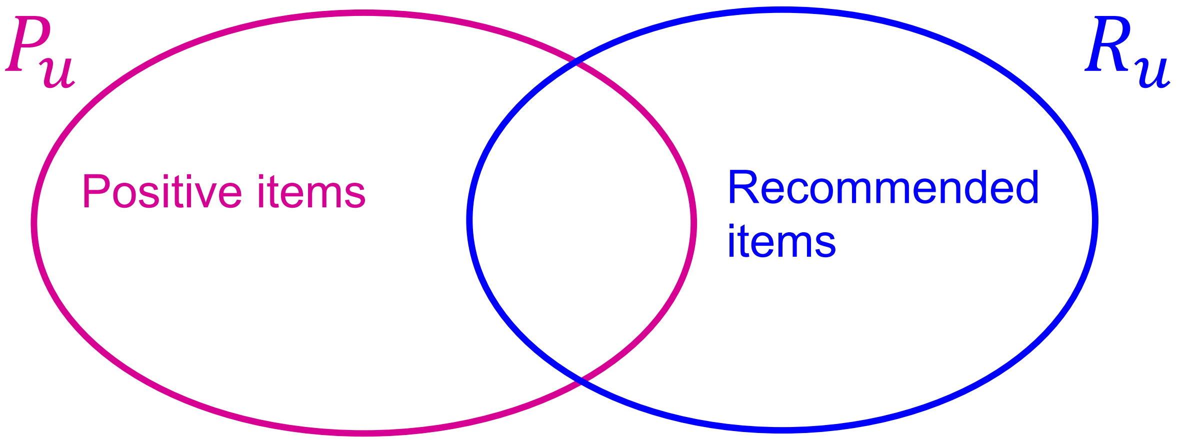


Top-K Recommendation

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

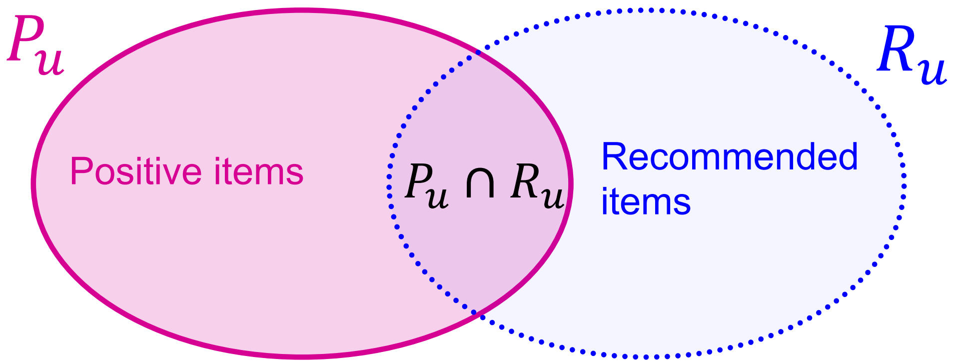
Evaluation Metric: Recall@K

- **For each user u,**
  - Let P_u be a set of positive items the user will interact in the future.
  - Let R_u be a set of items recommended by the model.
    - In top-K recommendation, $|R_u| = K$.
    - Items that the user has already interacted are excluded.



Evaluation Metric: Recall@K

- **Recall@K for user u is $|P_u \cap R_u|/|P_u|$.**
  - Higher value indicates more positive items are recommended in top-K for user u.



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

Methods

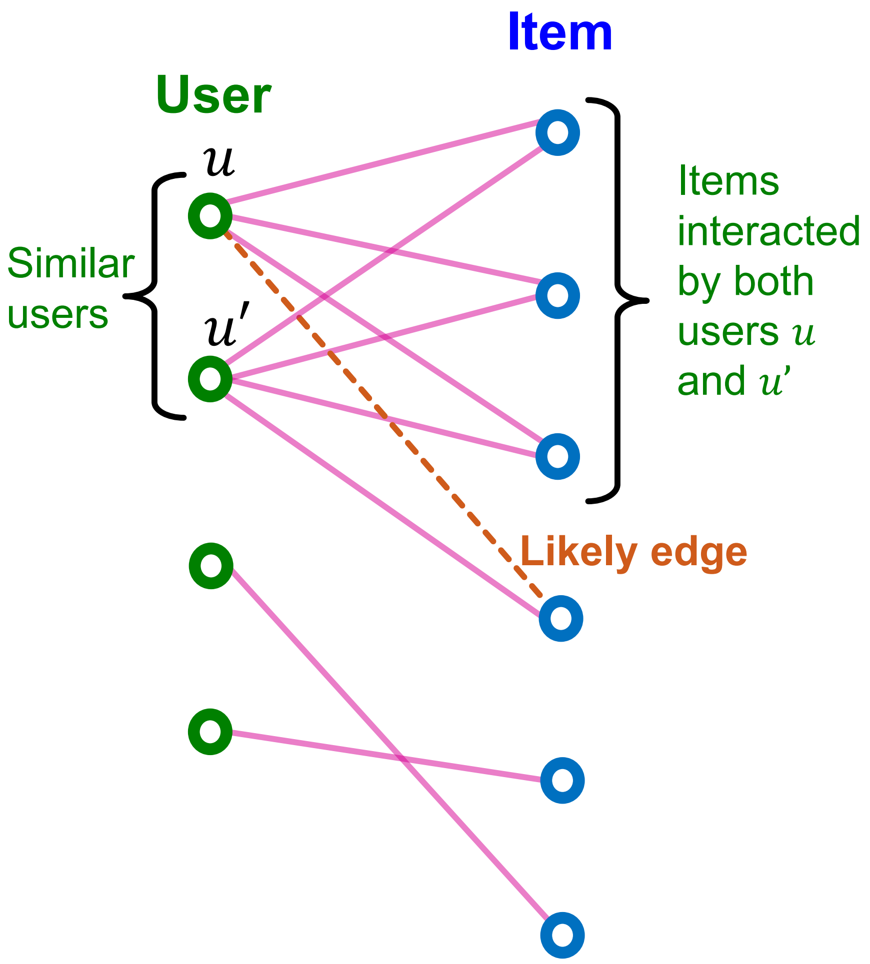
- Collaborative filtering
- Content-based recommendation
- Hybrid methods

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?