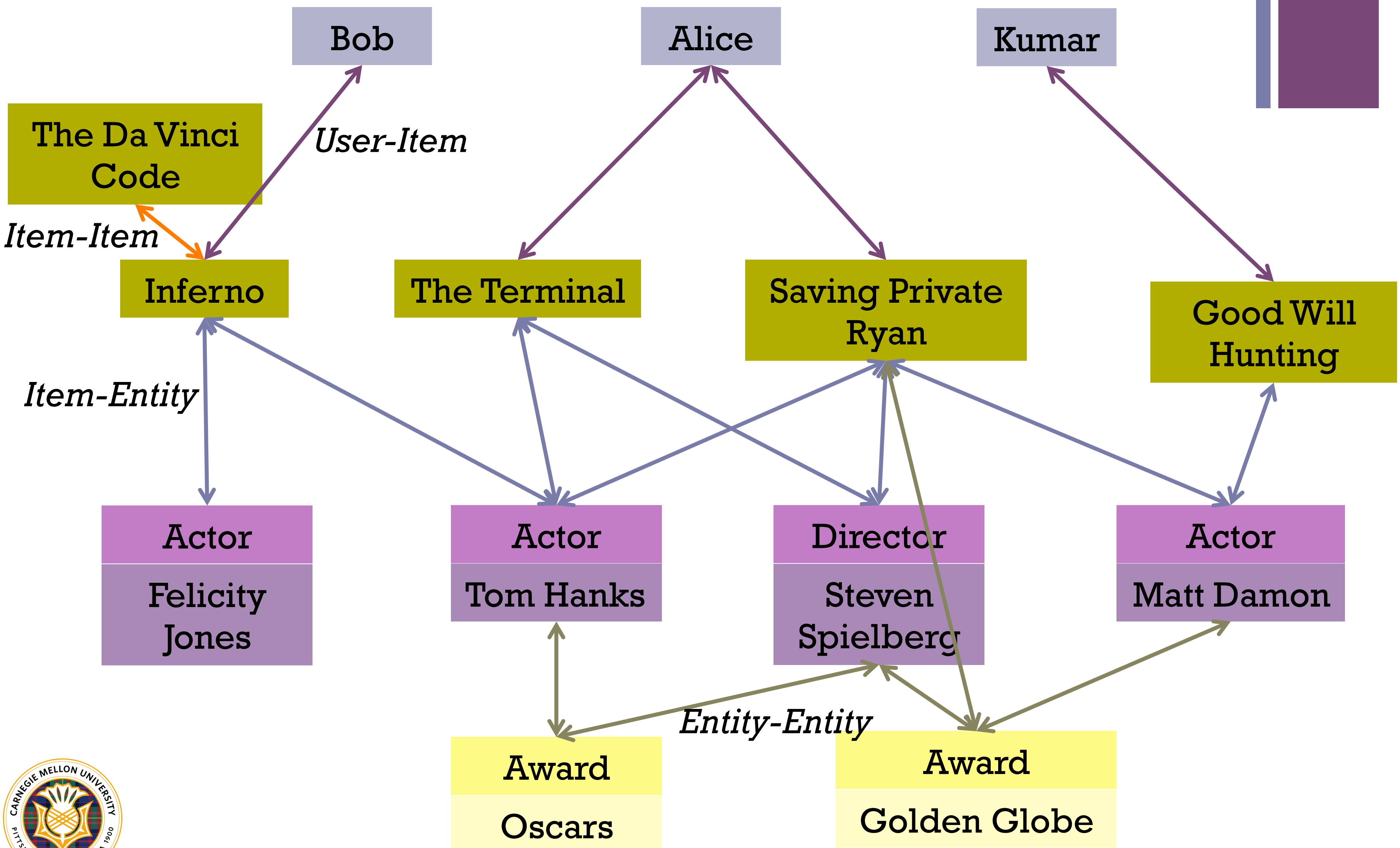
Personalized Recommendations using Knowledge Graphs

Rose Catherine Kanjirathinkal & Prof. William Cohen
Carnegie Mellon University

The Problem

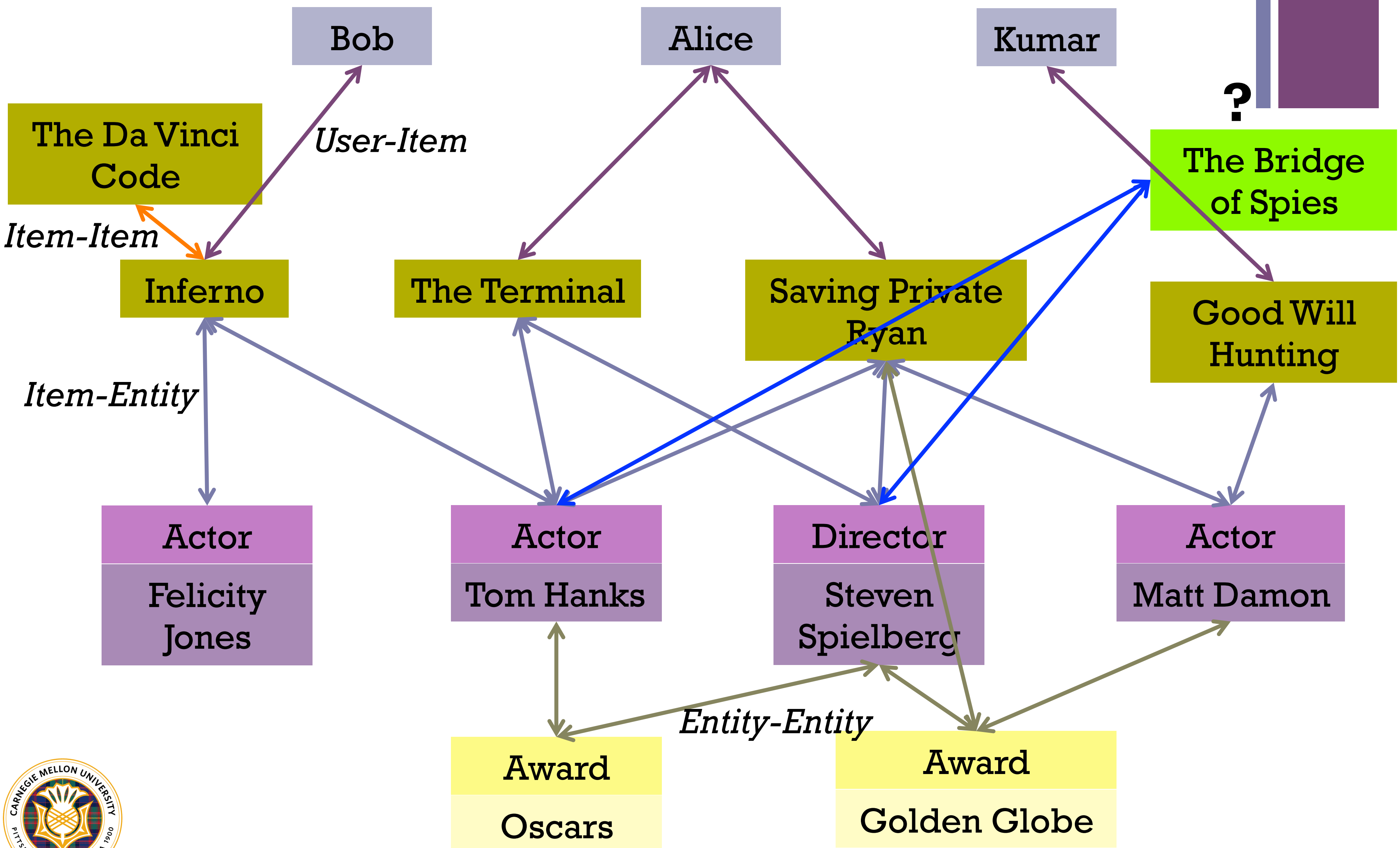
- Generate content-based recommendations on sparse real world data using **knowledge graphs (KG)**
- Knowledge Graph:
  - Think of the content (also referred to as entities) as nodes
  - Add links between:
    - Items-Entities: e.g. Bridge of Spies ←→ Tom Hanks
    - User-Items: e.g. Alice ←→ Saving Private Ryan
    - Entities-Entities: e.g. Tom Hanks ←→ Best Actor
    - Items-Items: e.g. Finding Nemo ←→ Finding Dory

Example



Bob
Alice
Kumar
The Da Vinci Code
User-Item
Item-Item
Inferno
The Terminal
Saving Private Ryan
Good Will Hunting
Item-Entity
Actor
Felicity Jones
Actor
Tom Hanks
Director
Steven Spielberg
Actor
Matt Damon
Entity-Entity
Award
Oscars
Award
Golden Globe

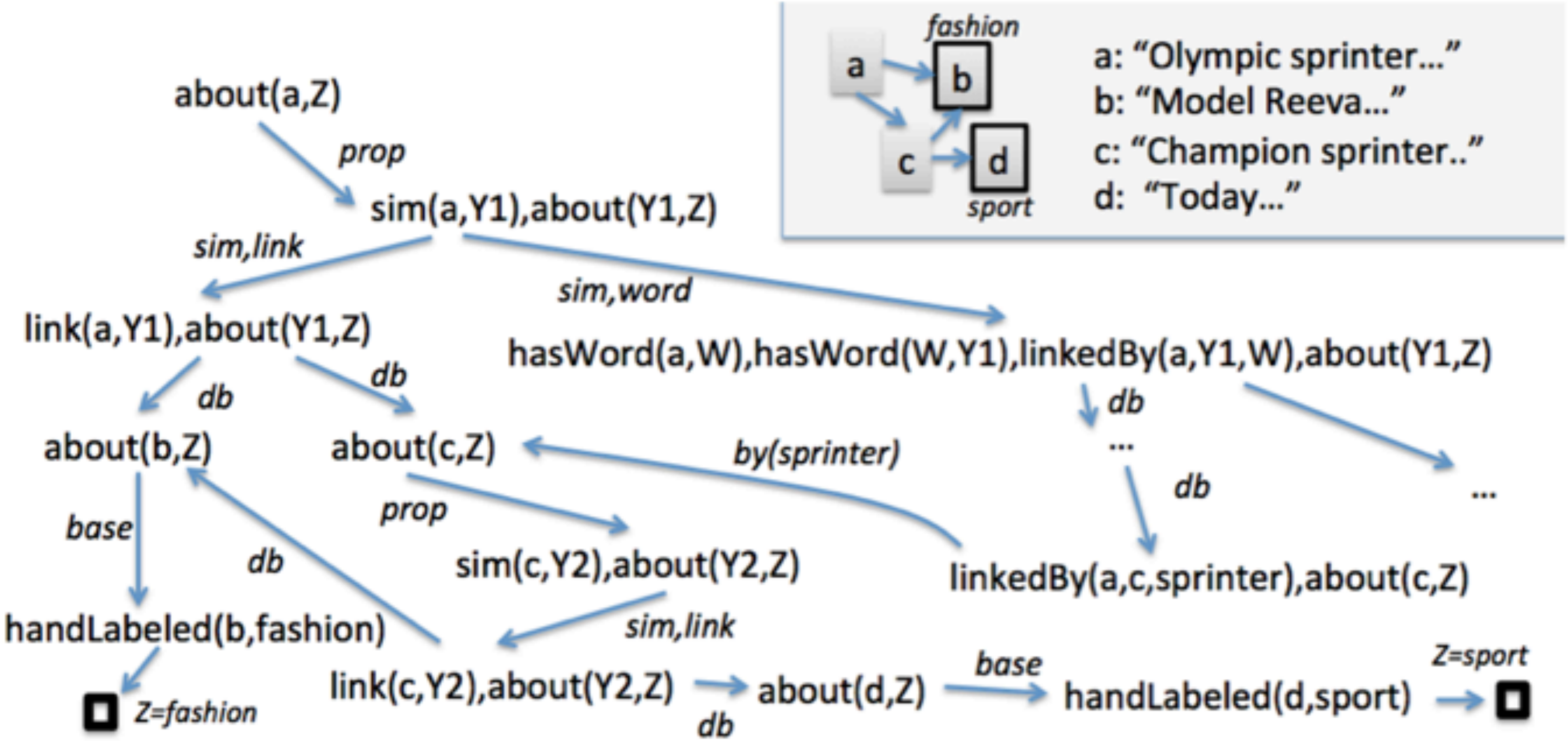
+ Example



Bob
Alice
Kumar
?
The Da Vinci Code
User-Item
The Bridge of Spies
Item-Item
Inferno
The Terminal
Saving Private Ryan
Good Will Hunting
Item-Entity
Actor
Felicity Jones
Actor
Tom Hanks
Director
Steven Spielberg
Actor
Matt Damon
Entity-Entity
Award
Oscars
Award
Golden Globe

+ Proposed Approach

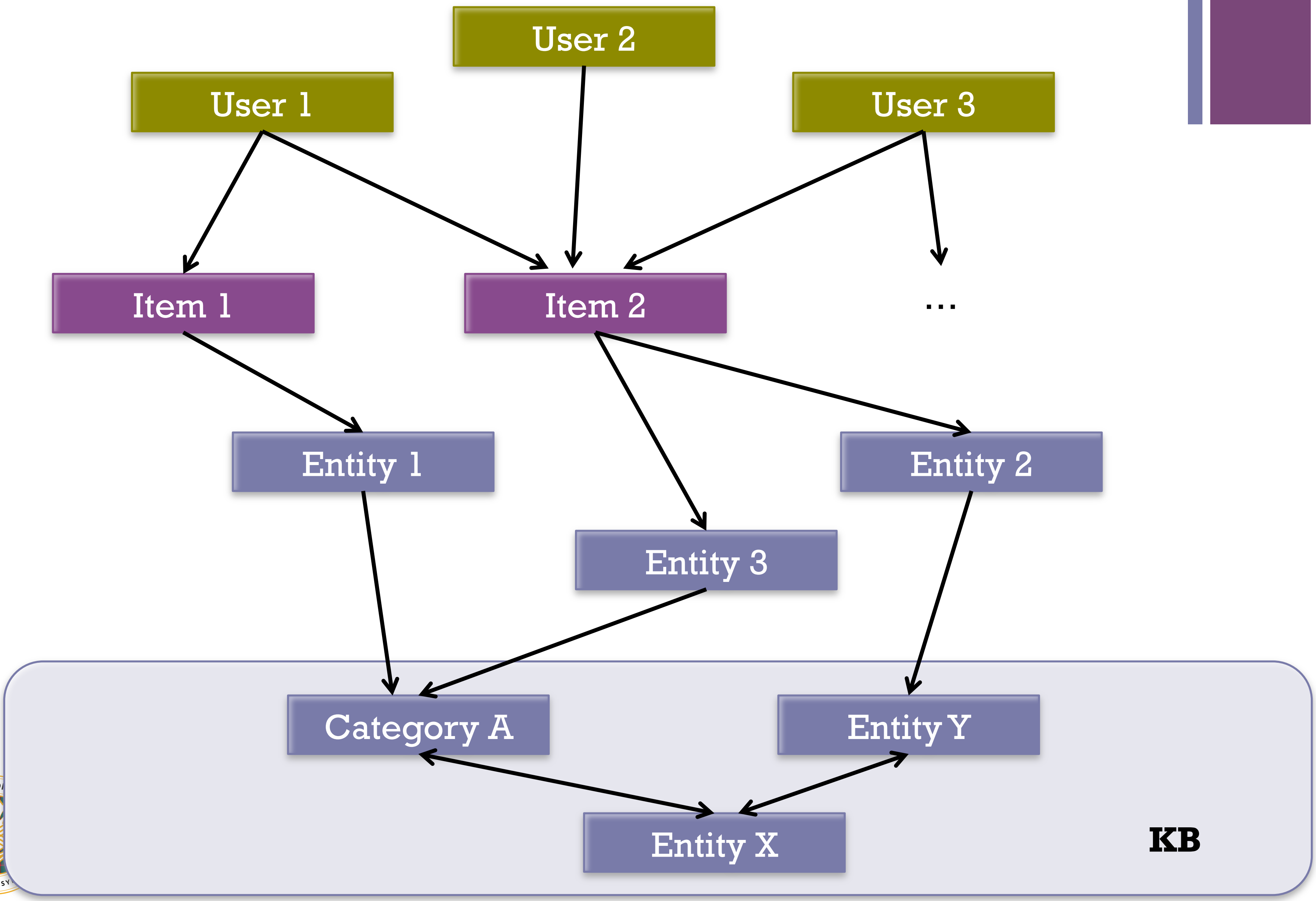
- Given that a user has liked specific movies and entities in the past, rank new movies (e.g. The Bridge of Spies) for that user using ProPPR
- ProPPR: **Pro**gramming with **P**ersonalized **P**age**R**ank
  - First order probabilistic logic system
  - Accepts rules and queries in a language similar to stochastic logic programs
  - Inference using a variant of personalized PageRank
  - During training, can learn weights of edges for performing the walk



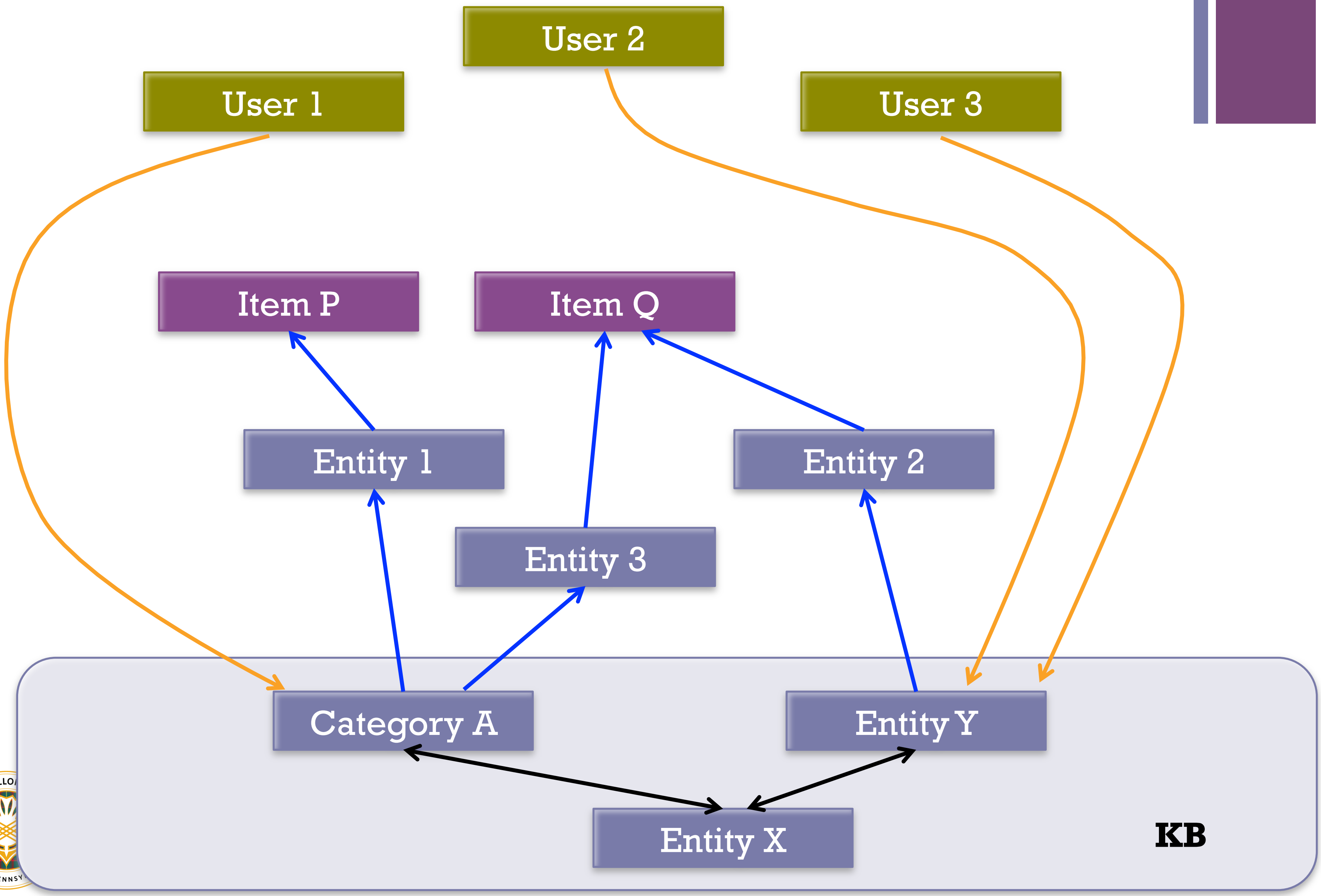
- The search space is a graph !
- Score for a query soln (e.g., "Z=sport" for "about(a,Z)") depends on random walk probability of reaching a □ node

```
about(X,Z) :- handLabeled(X,Z)              # base.
about(X,Z) :- sim(X,Y),about(Y,Z)           # prop.
sim(X,Y) :- links(X,Y)                      # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)                         # sim,word.
linkedBy(X,Y,W) :- true                     # by(W).
```

+ Observe:


User 2
User 1
User 3
Item 1
Item 2
...
Entity 1
Entity 2
Entity 3
Category A
Entity Y
Entity X
KB

Want to learn:



Proposed Approach Step 1: SeedSet

- Step 1: Generate a seedset

seedset(U,E) :- reviewed(U,M), link(M,X),
related(X,E), isEntity(E).

related(X,X) :- .

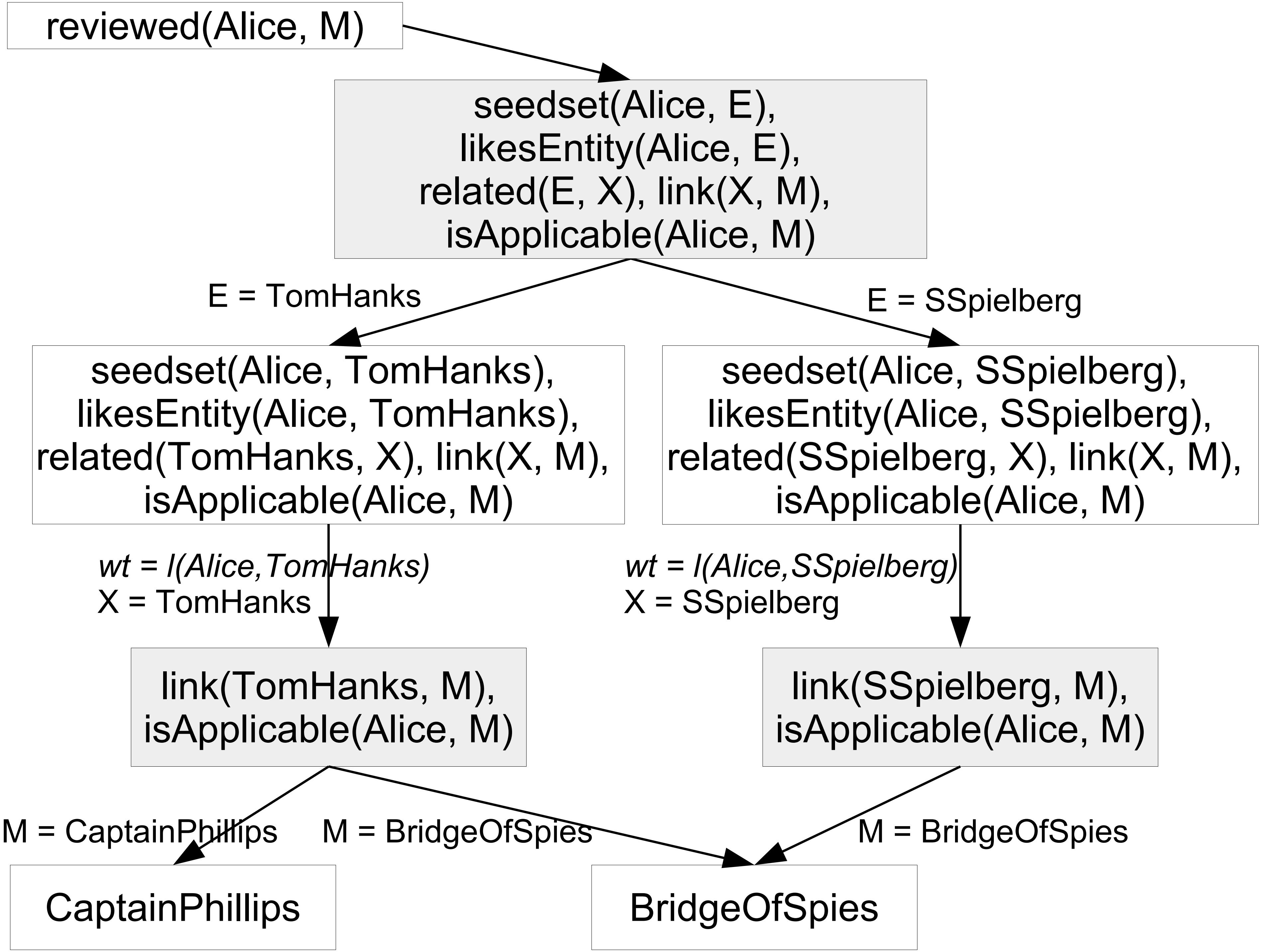
related(X,E) :- link(X,Z), related(Z,E).

- E.g. seedset(Alice,E) → E = TomHanks, StevenSpielberg

+ Approach 1: EntitySim

reviewed(U,M) :- seedset(U,E1), **likesEntity**(U,E1),
related(E1,E2), link(E2,M),
isApplicable(U,M).

likesEntity(U,E) :- { f(U,E)}.


reviewed(Alice, M)
seedset(Alice, E), likesEntity(Alice, E), related(E, X), link(X, M), isApplicable(Alice, M)
E = TomHanks
E = SSpielberg
seedset(Alice, TomHanks), likesEntity(Alice, TomHanks), related(TomHanks, X), link(X, M), isApplicable(Alice, M)
seedset(Alice, SSpielberg), likesEntity(Alice, SSpielberg), related(SSpielberg, X), link(X, M), isApplicable(Alice, M)
wt = I(Alice,TomHanks)
X = TomHanks
wt = I(Alice,SSpielberg)
X = SSpielberg
link(TomHanks, M), isApplicable(Alice, M)
link(SSpielberg, M), isApplicable(Alice, M)
M = CaptainPhillips
M = BridgeOfSpies
M = BridgeOfSpies
CaptainPhillips
BridgeOfSpies

+ Approach 2: TypeSim

- **Types** of entities/nodes available
- E.g. **Tom Hanks** is an **Actor**, **Pittsburgh** is a **City**
- Additional Rule A: Learn the popularity/predictability of each type and entity
- E.g. predictive power of Actor > Country, Tom Hanks > lesser known
- Additional Rule B: Learn Type Associations – general traversal probability between types
- E.g. Actor → Movie > Country → Movie

Approach 2: TypeSim - RuleSet

reviewed(U,M) :- seedset(U,E), likesEntity(U,E),
popularEntity(E), related(E,X), link(X,M),
isApplicable(U,M).

popularEntity(E) :- entityOfType(E,T), popularType(T) { p(E)}.

popularType(T) :- { p(T)}.

related(X,X) :- .

related(X,E) :- link(X,Z), **typeAssoc**(X,Z), related(Z,E).

typeAssoc(X,Z) :- entityOfType(X,S), entityOfType(Z,T),
typeSim(S,T).

typeSim(S,T) :- { t(S,T)}.

+ Approach 3: GraphLF

- Latent Factor models – successful in Collaborative Filtering
- Map users and items to the same feature space of hidden dimensions
- E.g. comedy vs. drama, amount of action, depth of character development, un-interpretable
- Each factor measures user's preference for movies that are high in that factor
- Predict based on rating data; no user/item information required

Approach 3: GraphLF - RuleSet

reviewed(U,M) :- **related**(U,E), **related**(E,X), link(X,M),
isApplicable(U,M).

related(U,E) :- seedset(U,E), **simLF**(U,E).

related(X,X) :- .

related(X,Y) :- link(X, Z), **simLF**(X, Z), related(Z, Y).

simLF(X, Y) :- isDim(D), val(X, D), val(Y, D).

val(X,D) :- { v(X,D)}.

Model Complexity

- Model Complexity = number of parameters learned
- EntitySim: $O(n)$, n = #users
  - Because of constant seedset size
- TypeSim: $O(n + e + t^2)$, e = #entities, t = #types
  - t^2 : Because of type association between pairs of types
- GraphLF: $O(n + e + m)$, m = #items
- Typically, $m >> t$
- EnitySim < TypeSim < GraphLF

Recommendation using KG: State-of-the-art method

- **HeteRec_p** [*X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han.* Personalized entity recommendation: A heterogeneous information network approach. *In Proc. 7th ACM Int. Conf. on Web Search and Data Mining, WSDM '14*]
- Main Idea: Find user-item preferences when the rating is not explicitly available, using **Metapaths**
  - Metapath : a path on the TYPE / schema of the KG.
    - E.g. User → Movie → Actor → Movie
- Drawbacks:
  - Choose & tune hyper parameters: (a) Metapaths (b) number of clusters
  - Requires a rich KB with types for entities and links.

Experiments

Dataset	#Users	#Items	#Reviews
Yelp (2013)	43,873	11,537	229,907
IM100K-UIUC	943	1360	89,626
IM100K*	940	1566	93,948

- Timestamp sort and 80% earlier → Training, 20% newer → Test
- Content in Yelp: location city/state, type of business (restaurant, hospital, shopping), cuisine (american, sushi, indian), …
- Content in IM100K: actor, director, studio, genre, country, language

+ Additional Baselines

- Popularity: Recommend the popular items to users.
- Co-Click: Estimate conditional probabilities between items and recommend items with an aggregated conditional probability calculated using the training data of the target user.
- NMF: Non-negative matrix factorization – without using the paths
- Hybrid-SVM: Use SVM-based ranking function to learn a global recommendation model with user implicit feedback and meta-paths based similarity measures
- Naïve bayes – uses the content of items as features

+ Yelp: Performance Comparison

Method	P@1	MRR
Popularity	0.0074	0.0228
Co-Click	0.0147	0.0371
NMF	0.0162	0.0382
Hybrid SVM	0.0122	0.0337
HeteRec_p	0.0213	0.0513
EntitySim	0.0221	0.0641
TypeSim	0.0444	0.0973 [↑ 89%]
GraphLF	0.0482 [↑ 126%]	0.0966
NB	0	0.0087

- Using Type info & Latent Factorization gives improvements
- TypeSim vs. GraphLF: No clear winner
- NB: Using content without graph – poor performance

+ IM100K: Performance Comparison

(on IM100K-UIUC)

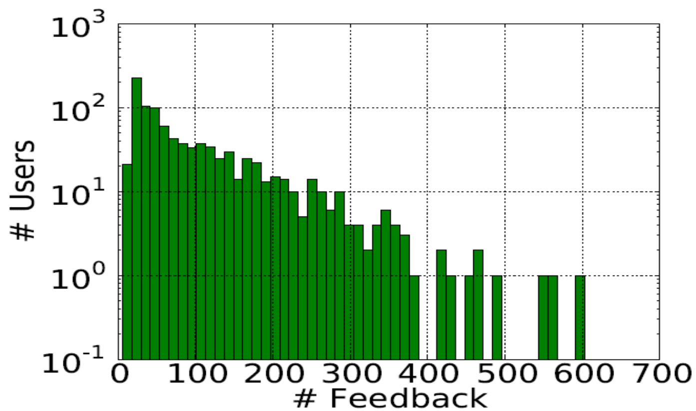
Method	P@1	MRR
Popularity	0.0731	0.1923
Co-Click	0.0668	0.2041
NMF	0.2064	0.4938
Hybrid SVM	0.2087	0.4493
HeteRec_p	0.2121	0.5530
EntitySim	0.3485	0.5010
TypeSim	0.353 [↑ 66%]	0.5053
GraphLF	0.3248	0.4852 [↓ −12%]
NB	0.312	0.4069

- Slightly different datasets: Methods cannot be compared directly – appear to be comparable
- EntitySim & NB: good performance
- Conjecture: simple methods suffice with enough training examples per user, enough content per item

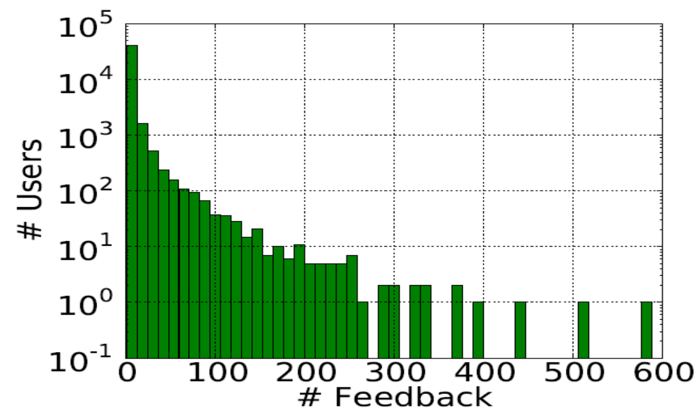
+ Rating Matrix Density

Users	Movie A	Movie B	Movie C	Movie D
User 1	5		2	
User 2		4	3	
User 3		1	2	
User 4	3	5	1	
User 5	2			3

- Density = #Ratings / (#Users x #Items)
- Density of Yelp = 0.00077
- Density of IM100K* = 0.06382
  - (82 times more dense)



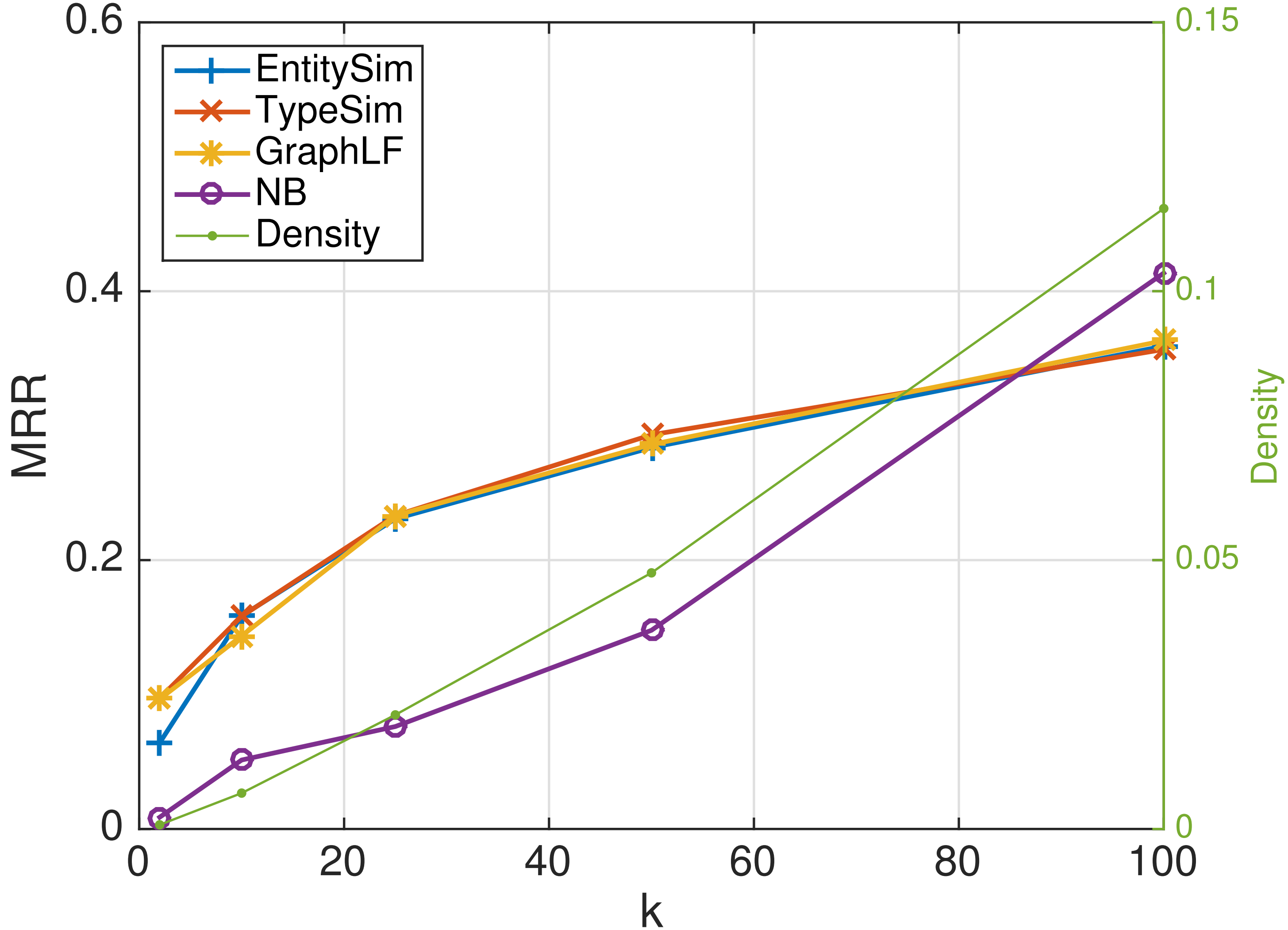
(b) IMDb Feedback Distribution



(c) Yelp Feedback Distribution

- Study the performance as density increases:
  - Create datasets by filtering out users and businesses with lesser than k ratings, where k = 10, 25, 50, 100

Performance vs. Density



Conclusions

- Proposed 3 methods that use KGs for making personalized recommendations
  - EntitySim: Uses the graph links
  - TypeSim: Uses additional type information
  - GraphLF: Combines Latent Factorization with Graphs
- Our methods gave large improvements compared to the state-of-the art method that uses knowledge graphs
- Studied the behavior of the methods as rating matrix density increased:
  - Type info became redundant
  - In sparse datasets, KG is an important source of information, especially at low densities.

Thank You!