13TH ACM CONFERENCE ON RECOMMENDER SYSTEMS
COPENHAGEN, DENMARK. 16-20 SEPTEMBER 2019
TUTORIAL: **GRAPH EMBEDDING** (11:00-12:30)

SMORe :

Modularize Graph Embedding for Recommendation

Ting-Hsiang Wang, Texas A&M University


Chih-Ming Chen
(CM)


Ting-Hsiang Wang
(Sean)


Chuan-Ju Wang
(Jennifer)


Ming-Feng Tsai
(Victor)

政治大学 NATIONAL CHENGCHI UNIVERSITY
中央研究院 ACADEMIA SINICA
kkbox
RECSYS 2019

Ting-Hsiang Wang, Texas A&M University

Tutorial Agenda

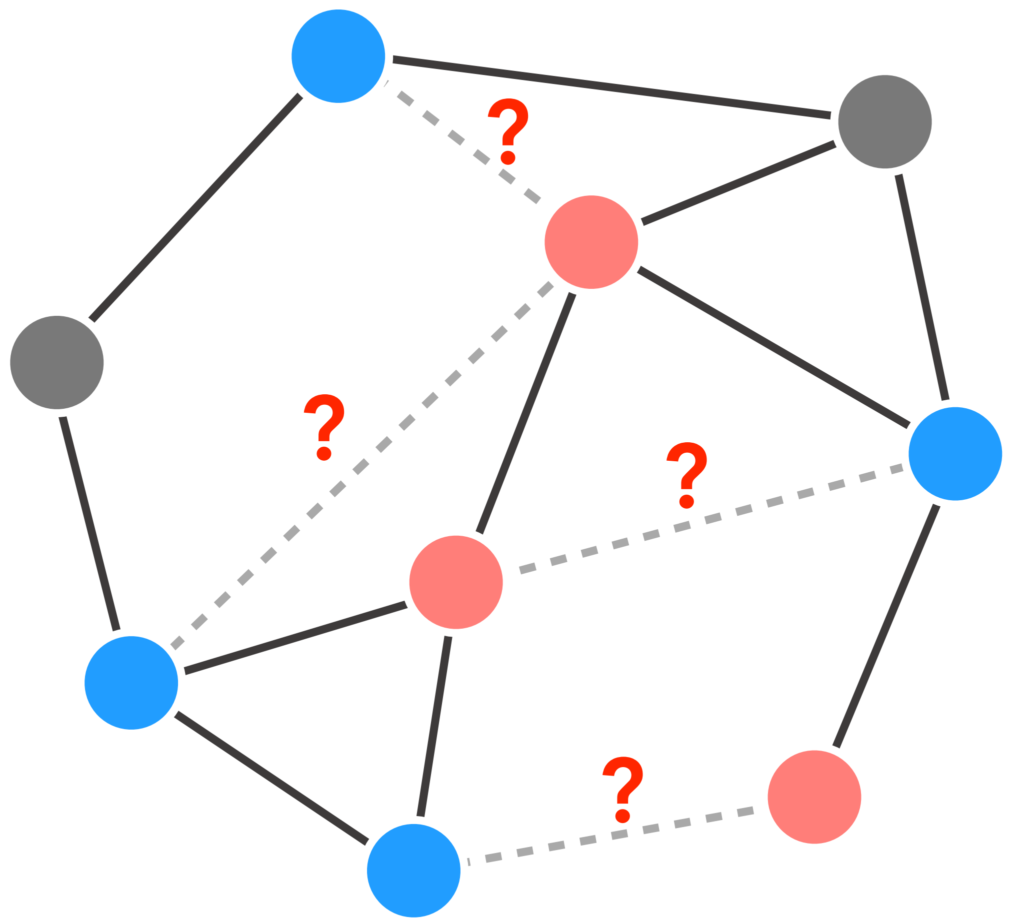
Lecture (Sean & CM, 65 minutes)

Hands-on (CM, 15 minutes)

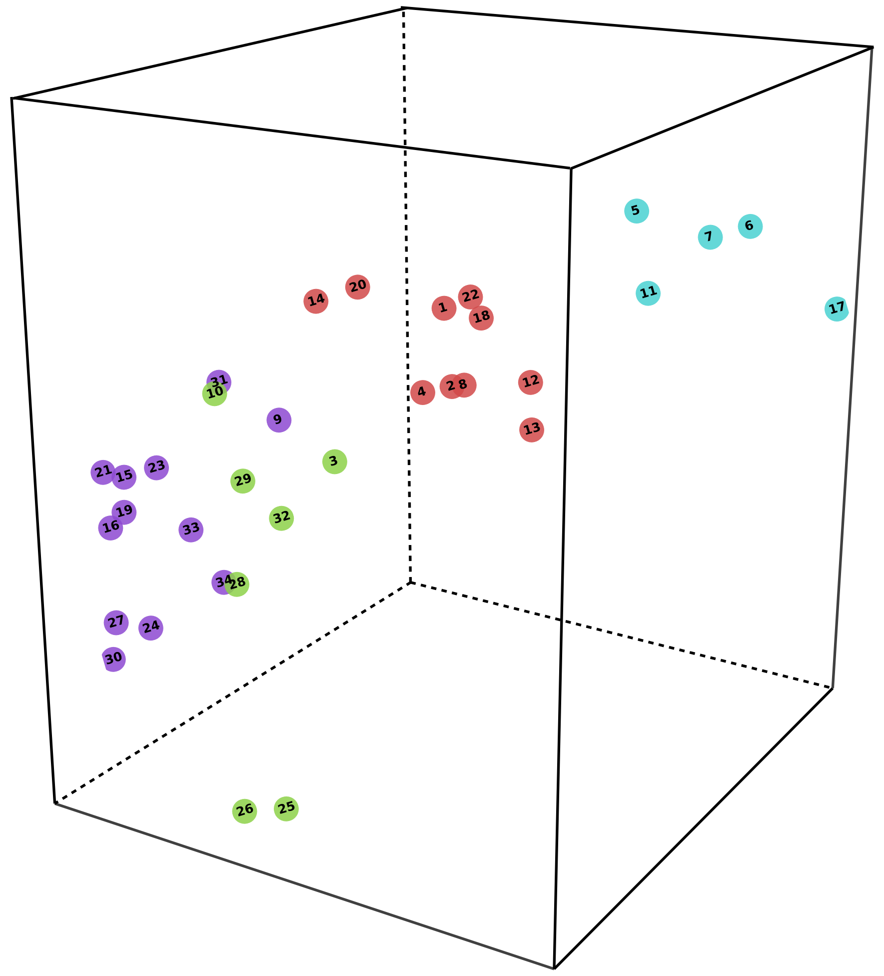
Q&A (Sean & CM, 10 minutes)

QR to Slides, Codes, Abstract

RECSYS 2019



SMORe



Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Lecture Agenda

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

(Sean) **Q2. SMORe** modularization of GE and benefits

(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

SMORe

RECSYS 2019

Ting-Hsiang Wang, Texas A&M University

Lecture Agenda

Ting-Hsiang Wang, Texas A&M University

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

(Sean) **Q2. SMORe** modularization of GE and benefits

(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

SMORe

RECSYS 2019

REC systems are everywhere !

Ting-Hsiang Wang, Texas A&M University

amazon
Book, Toy, Electronics, etc.

facebook
Friend, Interest Group

twitter
Trend, Following

NETFLIX
Movie

LinkedIn
Recruiter, Job Seeker

Google
News, Ad, Media, etc.

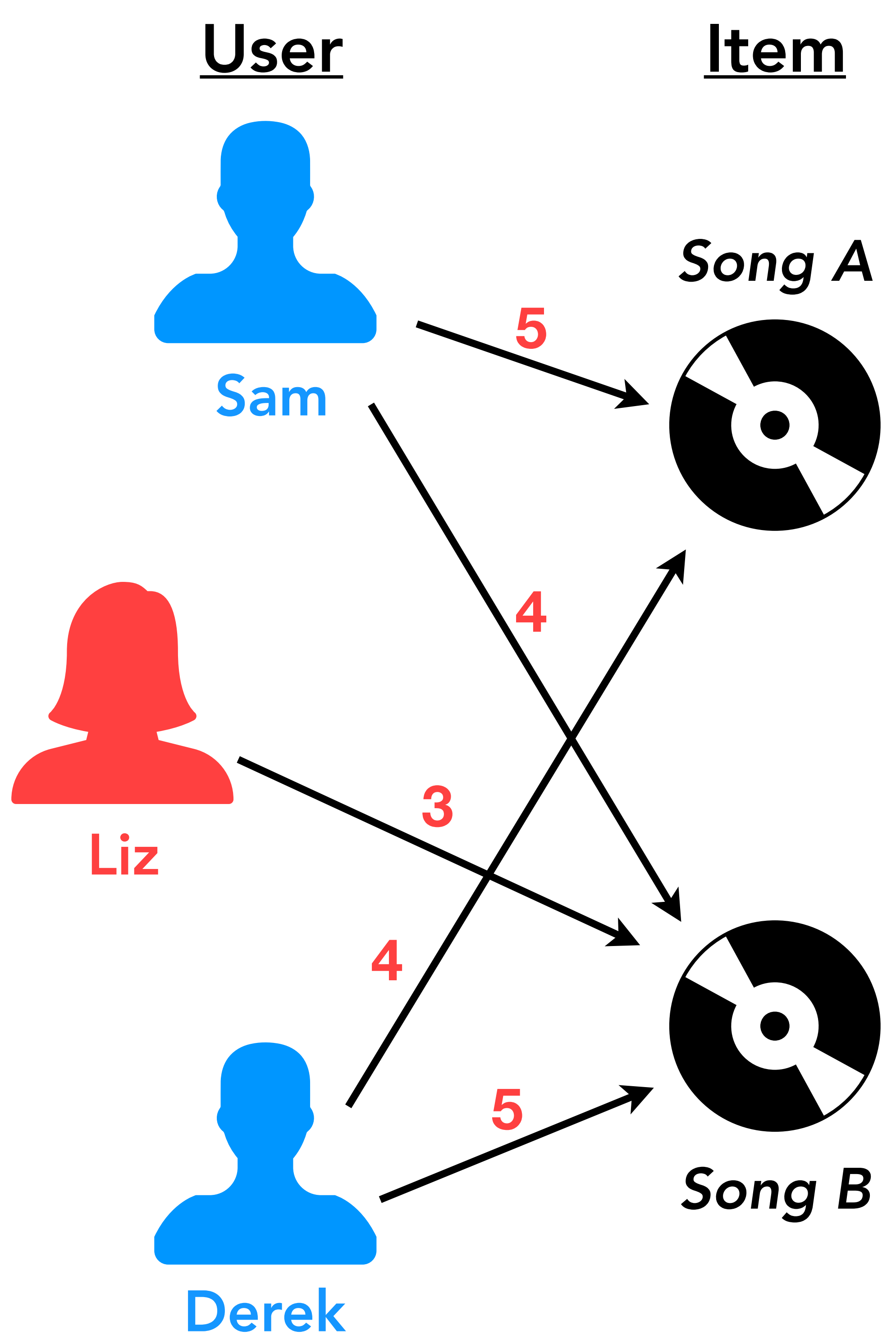
Spotify
Music

yelp
Local Business

And many more …

RECSYS 2019

Let’s start with Collaborative Filtering (CF)



	Song A	*Song B*	*Song C*
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

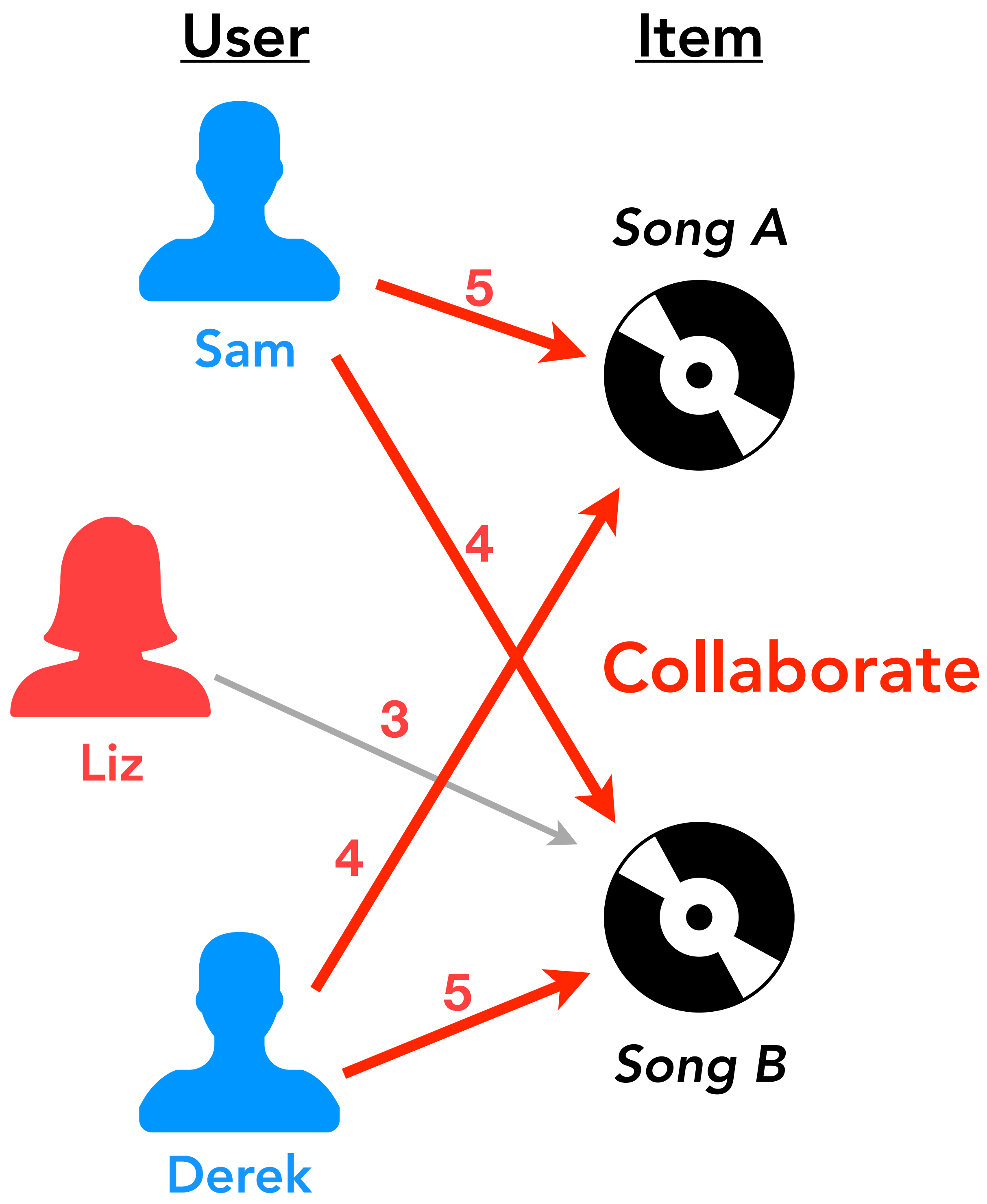
→ Sam and Derek have similar tastes; therefore, if Sam likes *Song C*, it is likely Derek does, too

- REC center around users to provide customized results
- CF assumes **people agree on things are likely to agree on other things**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Let's start with Collaborative Filtering (CF)



	Song A	*Song B*	*Song C*
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

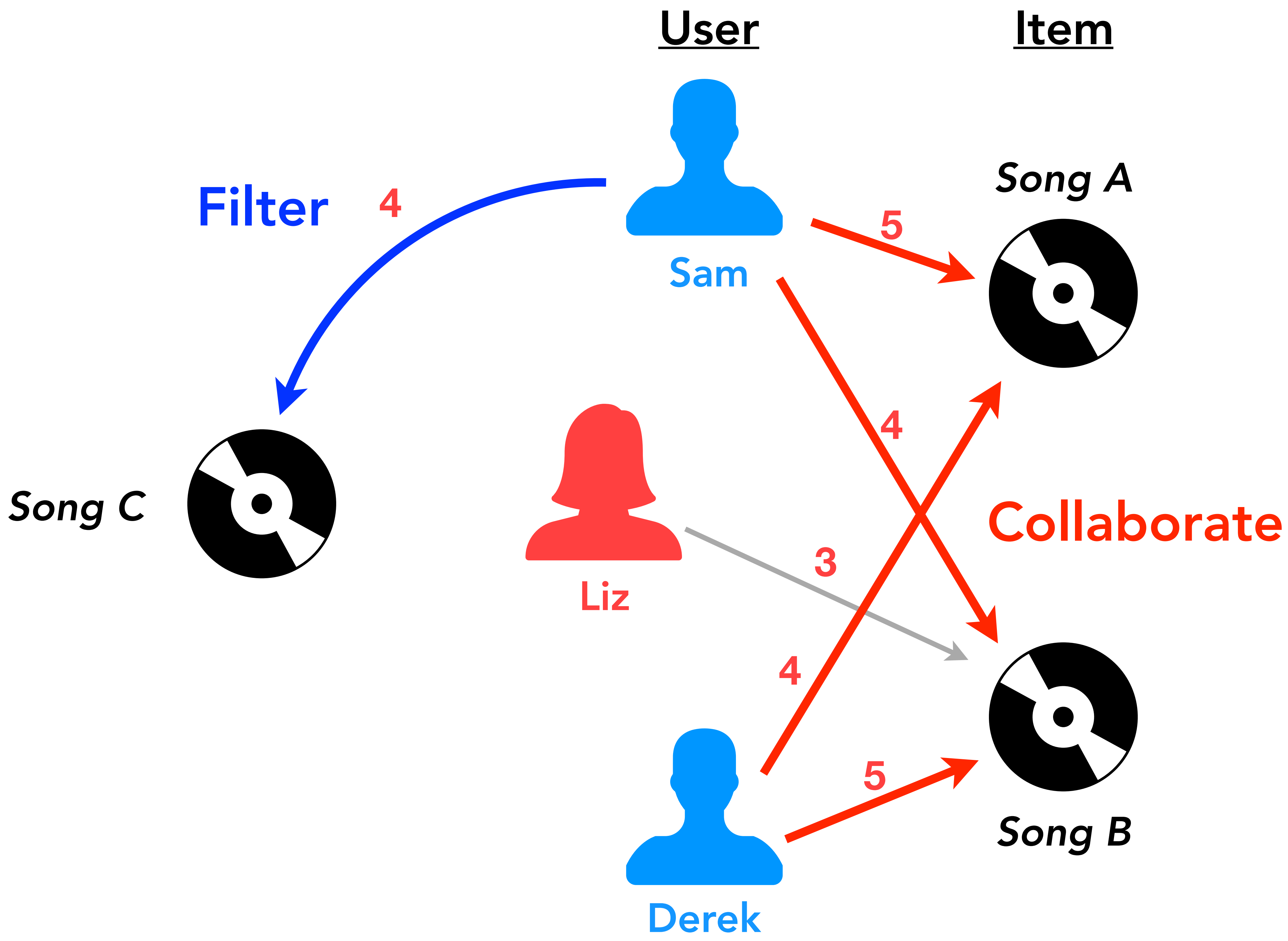
→ Sam and Derek have similar tastes; therefore, if Sam likes *Song C*, it is likely Derek does, too

- REC center around users to provide customized results
- CF assumes **people agree on things are likely to agree on other things**

RECSYS 2019

Ting-Hsiang Wang, Texas A&M University

Let’s start with Collaborative Filtering (CF)



	Song A	*Song B*	*Song C*
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

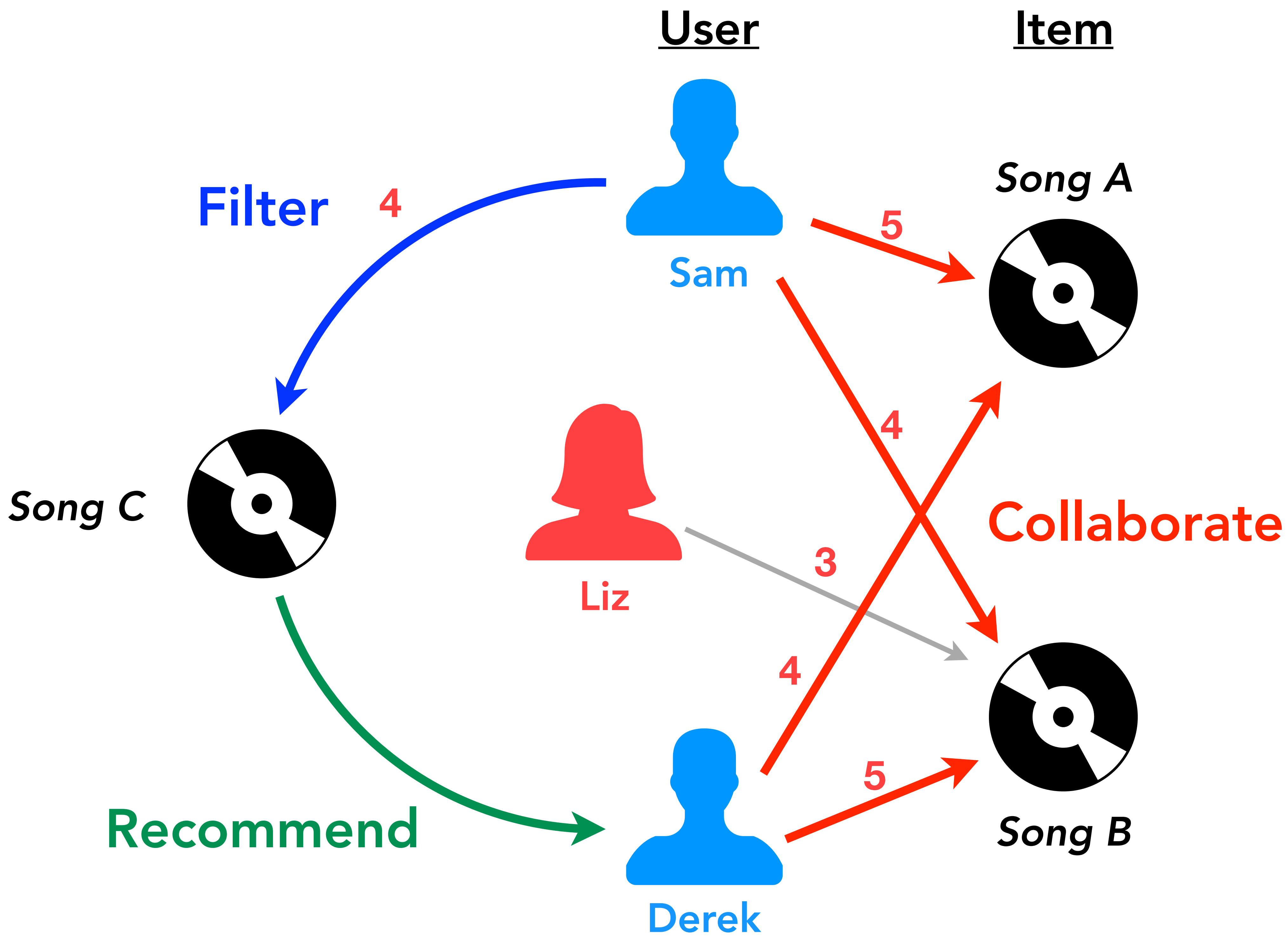
→ Sam and Derek have similar tastes; therefore, if Sam likes *Song C*, it is likely Derek does, too

- REC center around users to provide customized results
- CF assumes **people agree on things are likely to agree on other things**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Let's start with Collaborative Filtering (CF)

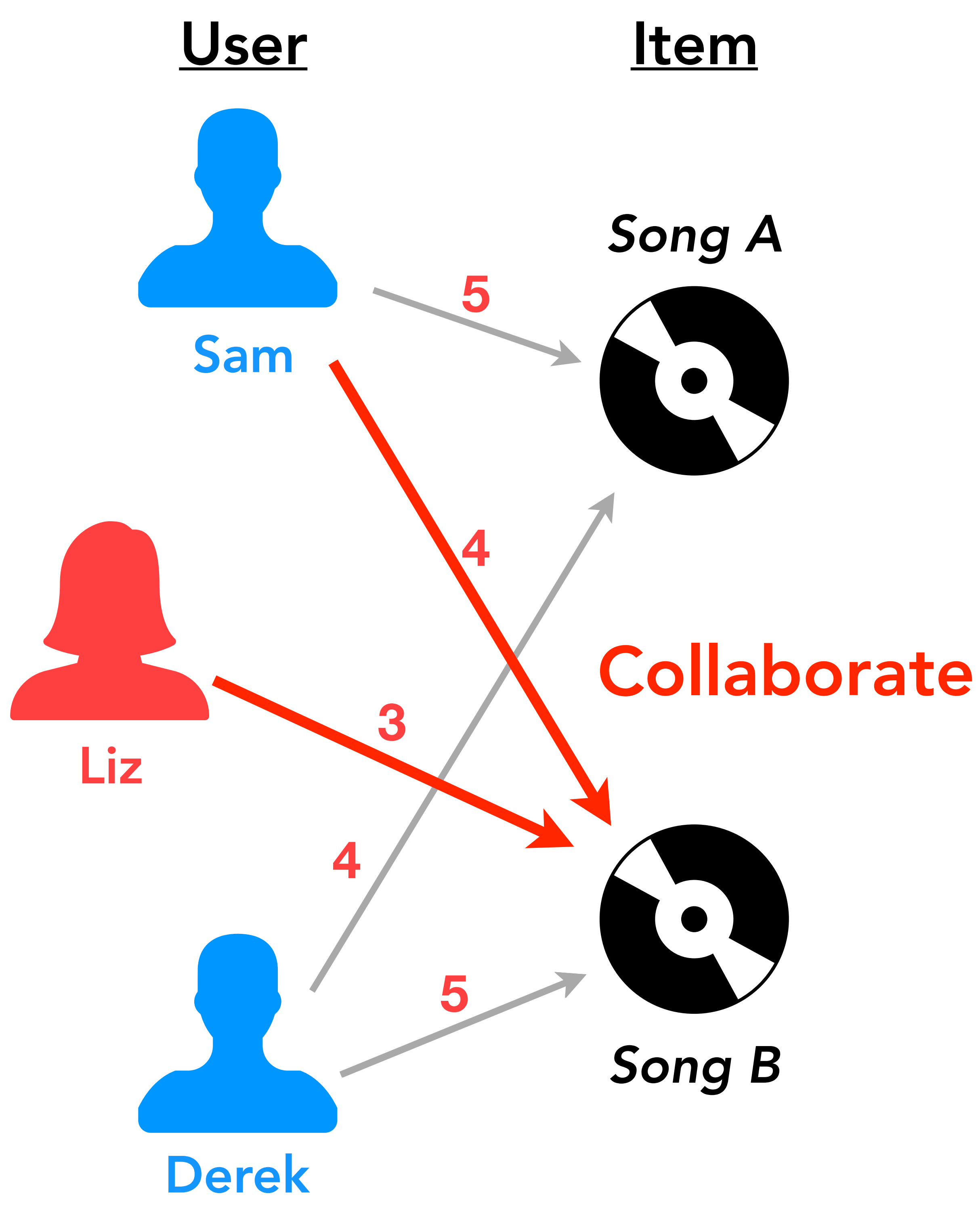


	Song A	Song B	Song C
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

→ Sam and Derek have similar tastes; therefore, if Sam likes *Song C*, it is likely Derek does, too

- REC center around users to provide customized results
- CF assumes **people agree on things are likely to agree on other things**

Challenge (1) : Data Sparsity



	Song A	*Song B*	*Song C*
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

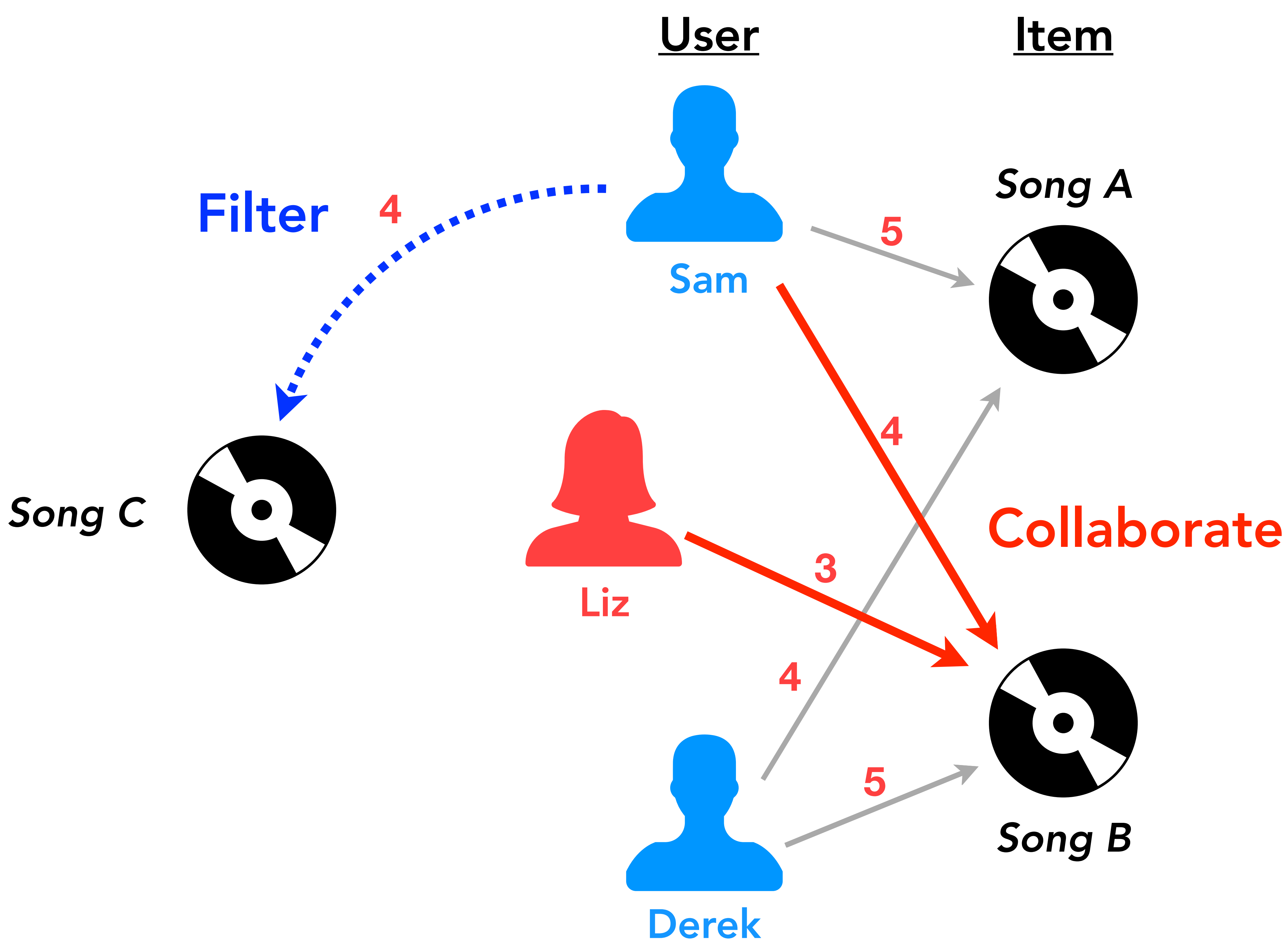
→ **Data sparsity** occurs to Liz as there is **insufficient** ratings from Liz to compare her taste with others'

- When **data sparsity** occurs, it is difficult to make accurate REC with CF

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Challenge (1) : Data Sparsity



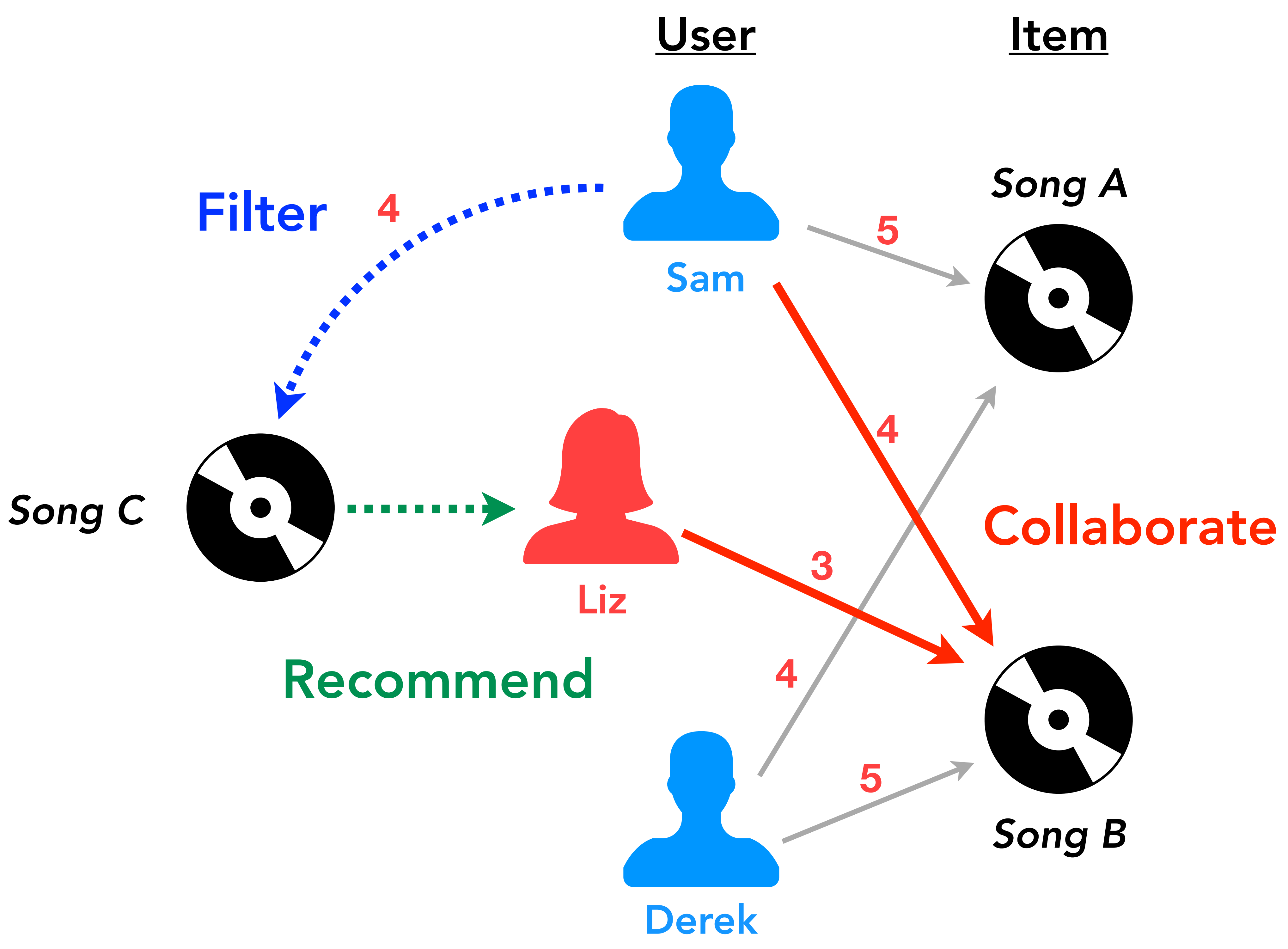
	Song A	Song B	Song C
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

→ Data sparsity occurs to Liz as there is **insufficient** ratings from Liz to compare her taste with others'

- When **data sparsity** occurs, it is difficult to make accurate REC with CF

RECSYS 2019

Challenge (1) : Data Sparsity



	Song A	Song B	Song C
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

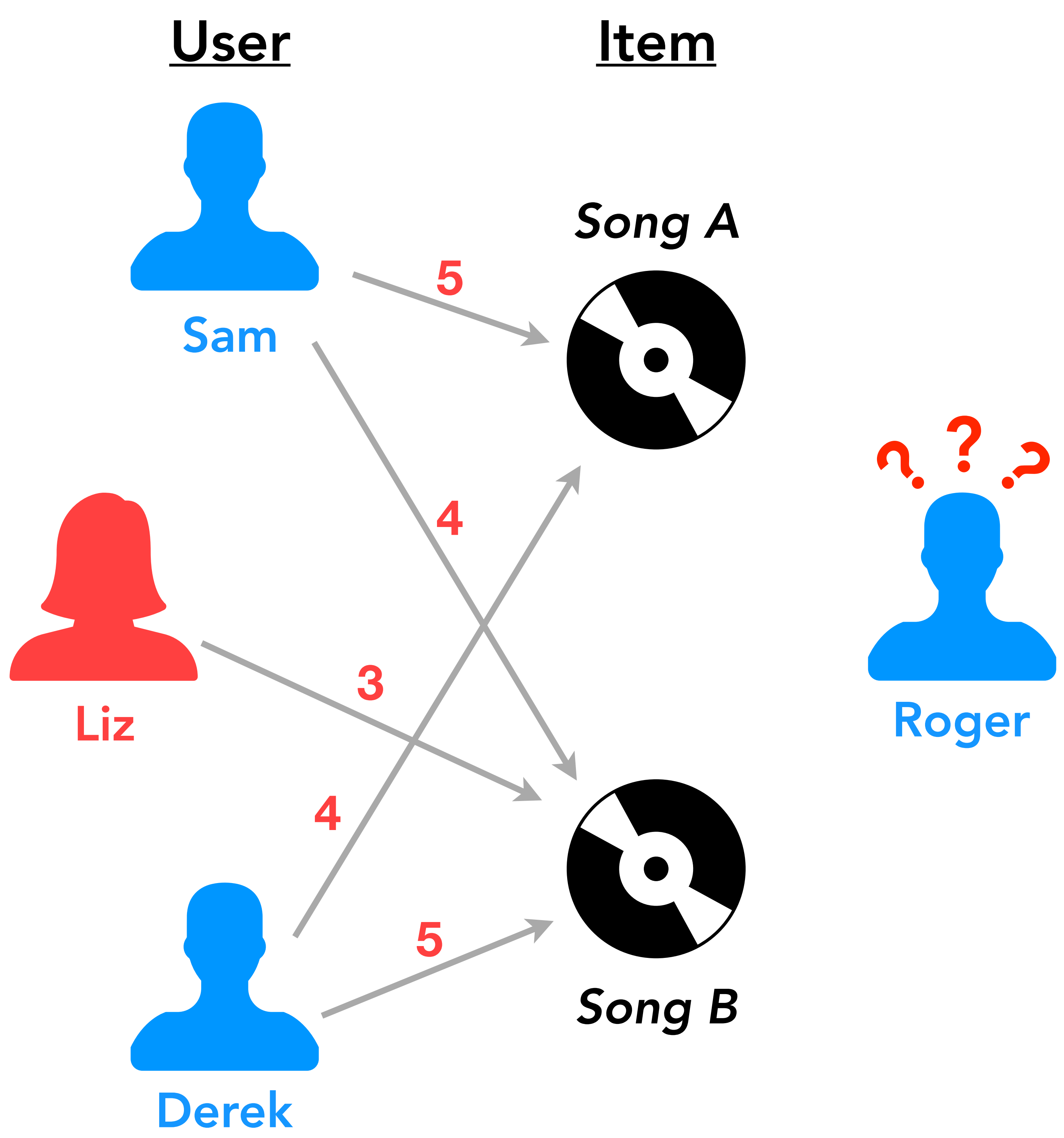
→ **Data sparsity** occurs to Liz as there is **insufficient** ratings from Liz to compare her taste with others'

- When **data sparsity** occurs, it is difficult to make accurate REC with CF

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Challenge (2) : Cold Start



	Song A	*Song B*	*Song C*
Sam	5	4	4
Derek	4	5	?
Liz	-	3	?
Roger	-	-	?

→ **Cold Start** occurs to Roger as there is **NO** song rating from Roger to compare his taste with others'

- When **cold start** occurs, it is impossible to conduct CF since there is **no context** for comparison

Challenge (3) : Constant Cold Start

KKTIX

amazarashi Live Tour 2019 in Shanghai & Taipei

2019年11月2日, 週六 19:30 **November 2, 2019**

Legacy Taipei
台北市八德路一段一號 中5A館

GO OUT CAMP RYUKYU 2019

2019年11月16日 - 11月17日
11月16日 08:30 到 11月17日 18:00 **November 16, 2019**

日本 沖繩縣 縣民之村
日本 沖繩縣 國頭郡恩納村安富祖2028

LisAni ! LIVE TAIWAN 2019

2019年10月19日 - 10月20日
10月19日 16:00 到 10月20日 15:00 **October 16, 2019**

TICC台北國際會議中心
台北市信義路五段1號

→ Today is September 19, 2019

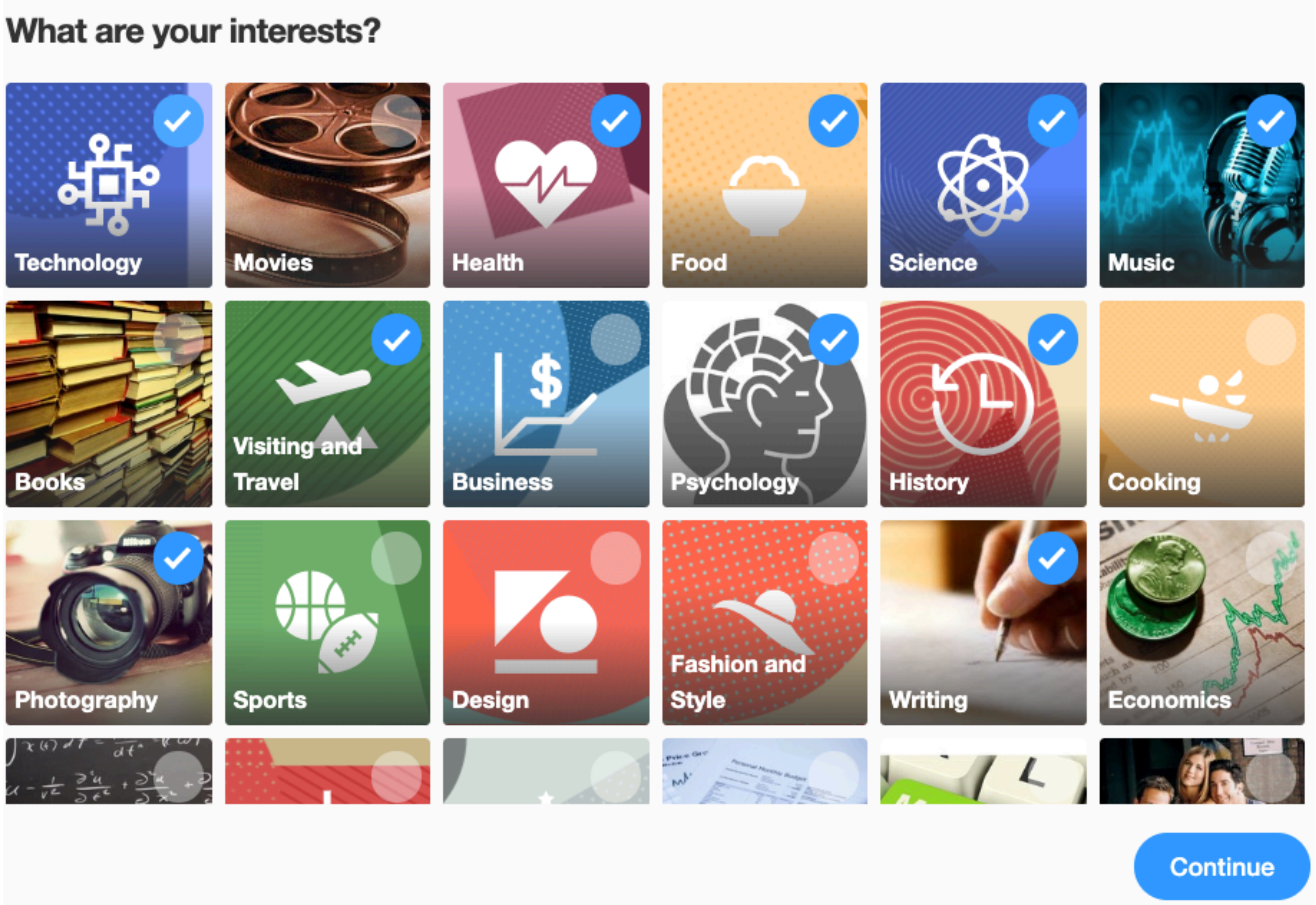
- Some **Cold Start** situations are **constant**, i.e., never warm up
- E.g., event tickets are sold before user-event interactions can occur

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Common Solution : Add Attributes

Quora



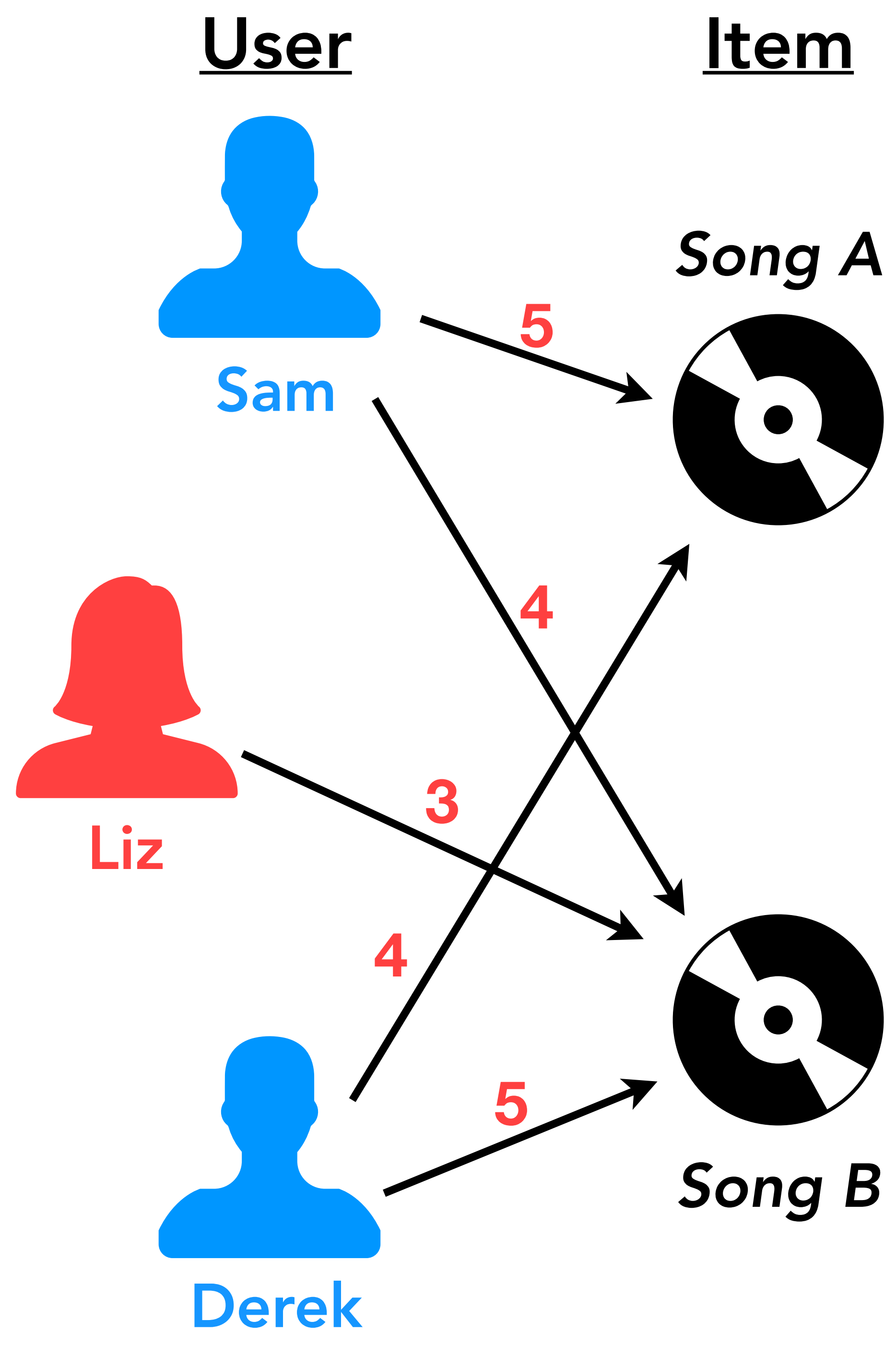
→ REC systems often mitigate **Complete Cold Start** problems by requiring new users to specify **interests** and items be labeled using **tags**

→ By requiring user interests and item tags to come from **predefined label set**, users and items share context for comparison

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

To enrich context for comparison …

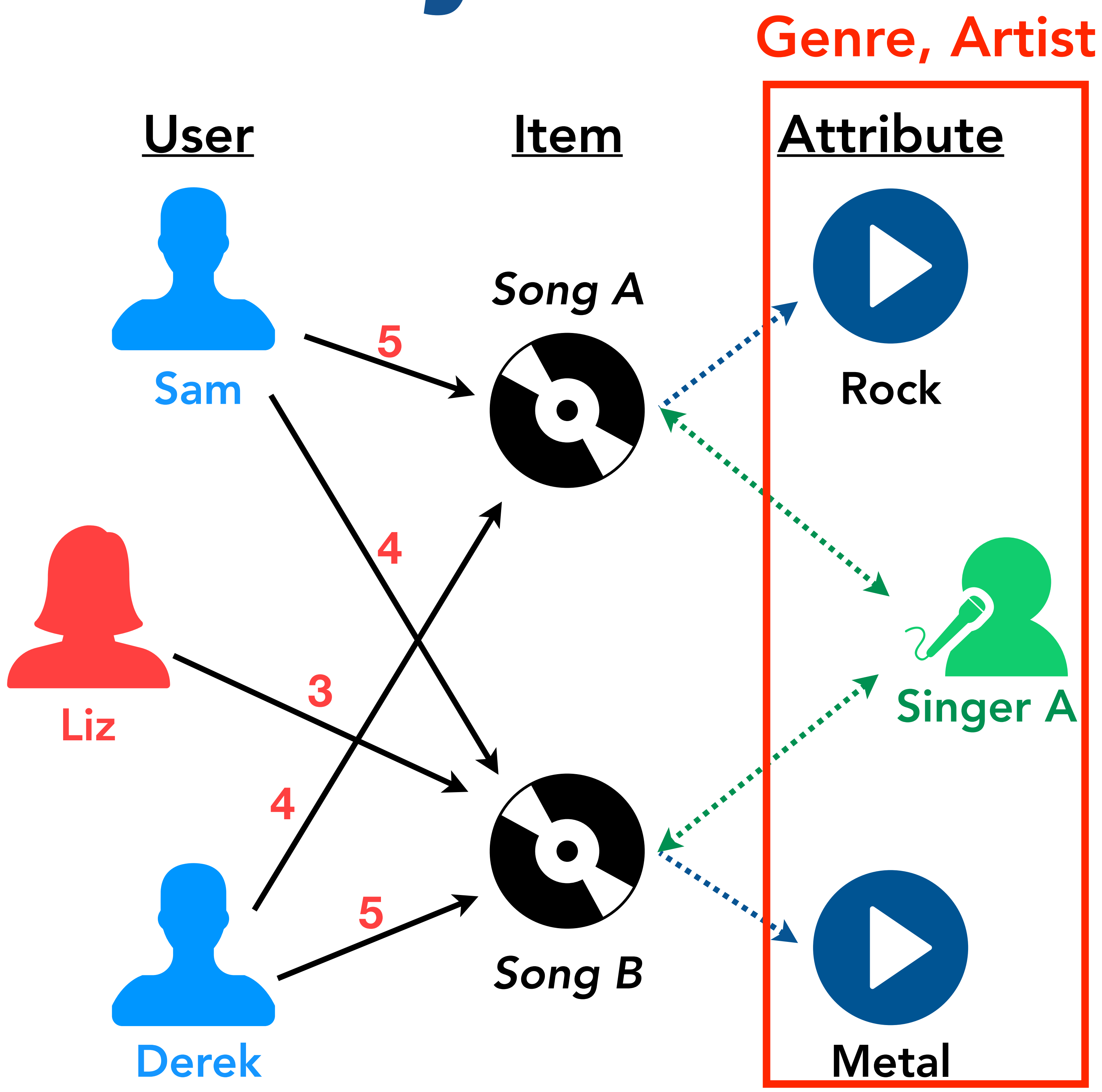


- For additional context for comparison, REC systems often borrow auxiliary information, such as item **content** and user and item **attributes**
- This forms a **Heterogeneous Information Network (HIN)**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

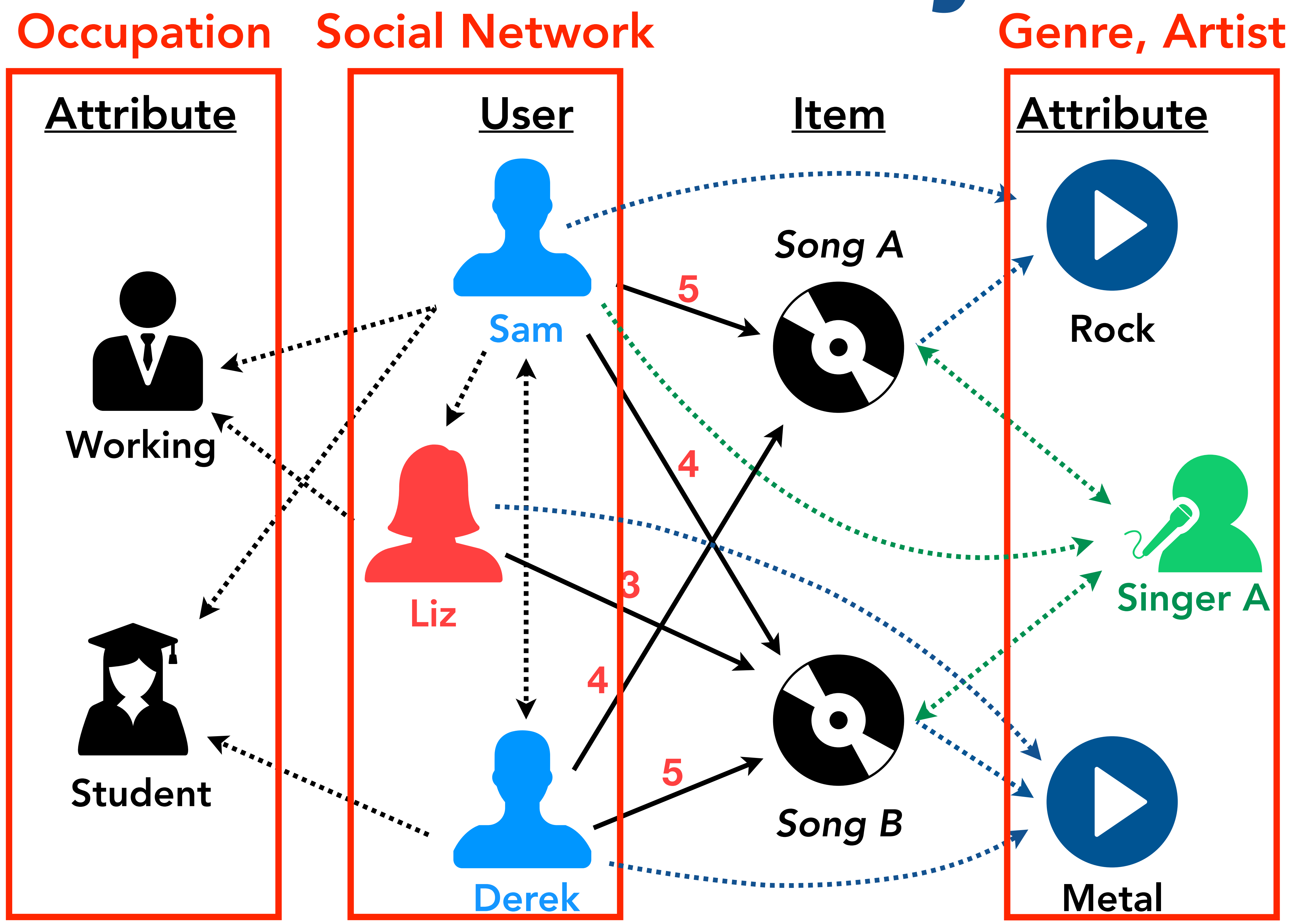
Add auxiliary information …



→ **Item attributes** are added to improve item comparison, e.g., music genre, artist, metadata, etc.

- For additional context for comparison, REC systems often borrow auxiliary information, such as item **content** and user and item **attributes**
- This forms a **Heterogeneous Information Network (HIN)**

And more auxiliary information !



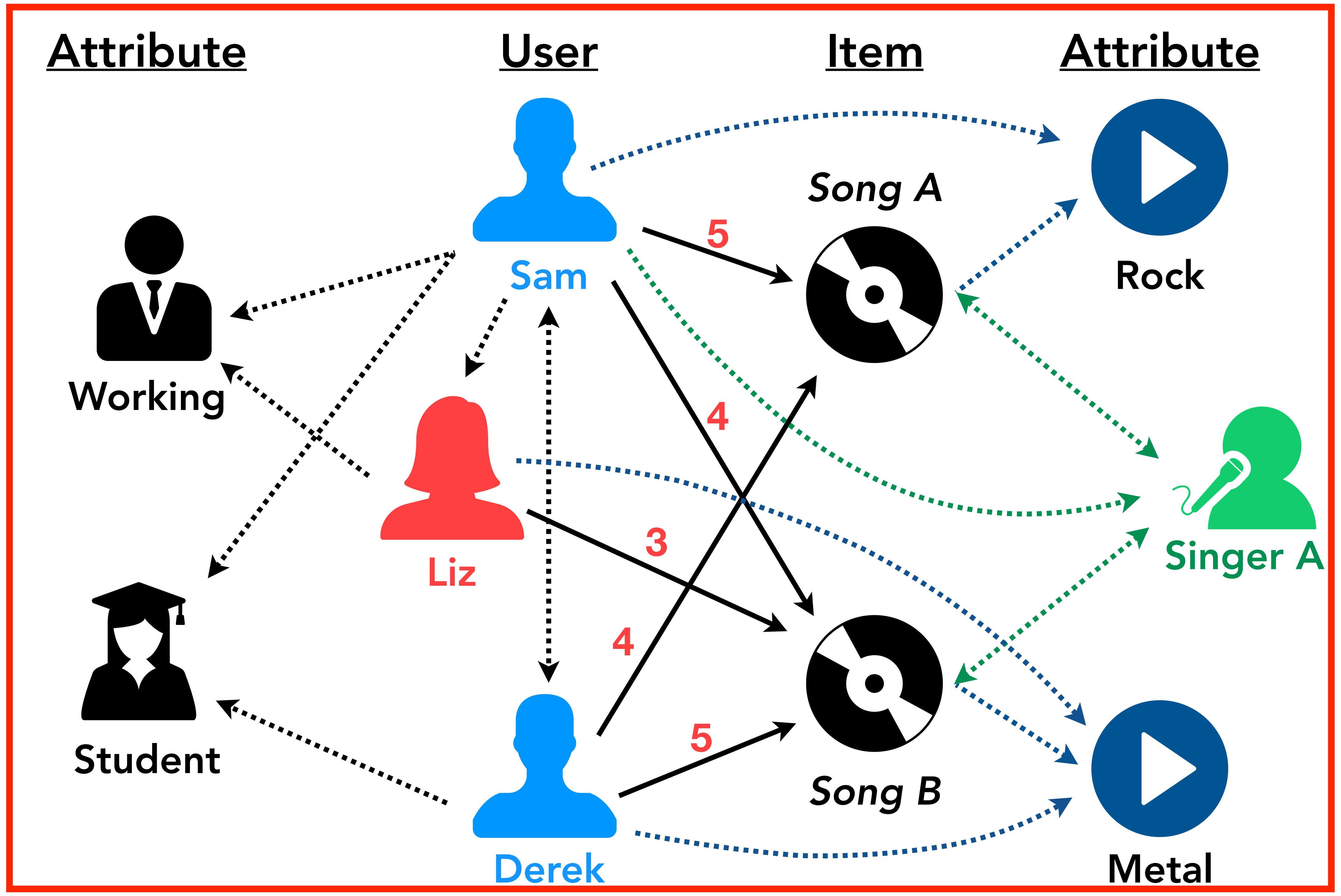
→ **Item attributes** are added to improve item comparison, e.g., music genre, artist, metadata, etc.

→ **User attributes** are added to improve user comparison, e.g., social network, occupation, favorite artist & genre, etc.

- For additional context for comparison, REC systems often borrow auxiliary information, such as item **content** and user and item **attributes**
- This forms a **Heterogeneous Information Network (HIN)**

Ting-Hsiang Wang, Texas A&M University

Graph coordinates all relations !



→ **Item attributes** are added to improve item comparison, e.g., music genre, artist, metadata, etc.

→ **User attributes** are added to improve user comparison, e.g., social network, occupation, favorite artist & genre, etc.

→ **HIN** gives **holistic** view of complex systems

- For additional context for comparison, REC systems often borrow auxiliary information, such as item **content** and user and item **attributes**
- This forms a **Heterogeneous Information Network (HIN)**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Lecture Agenda

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

(Sean) **Q2. SMORe** modularization of GE and benefits

(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

Ting-Hsiang Wang, Texas A&M University

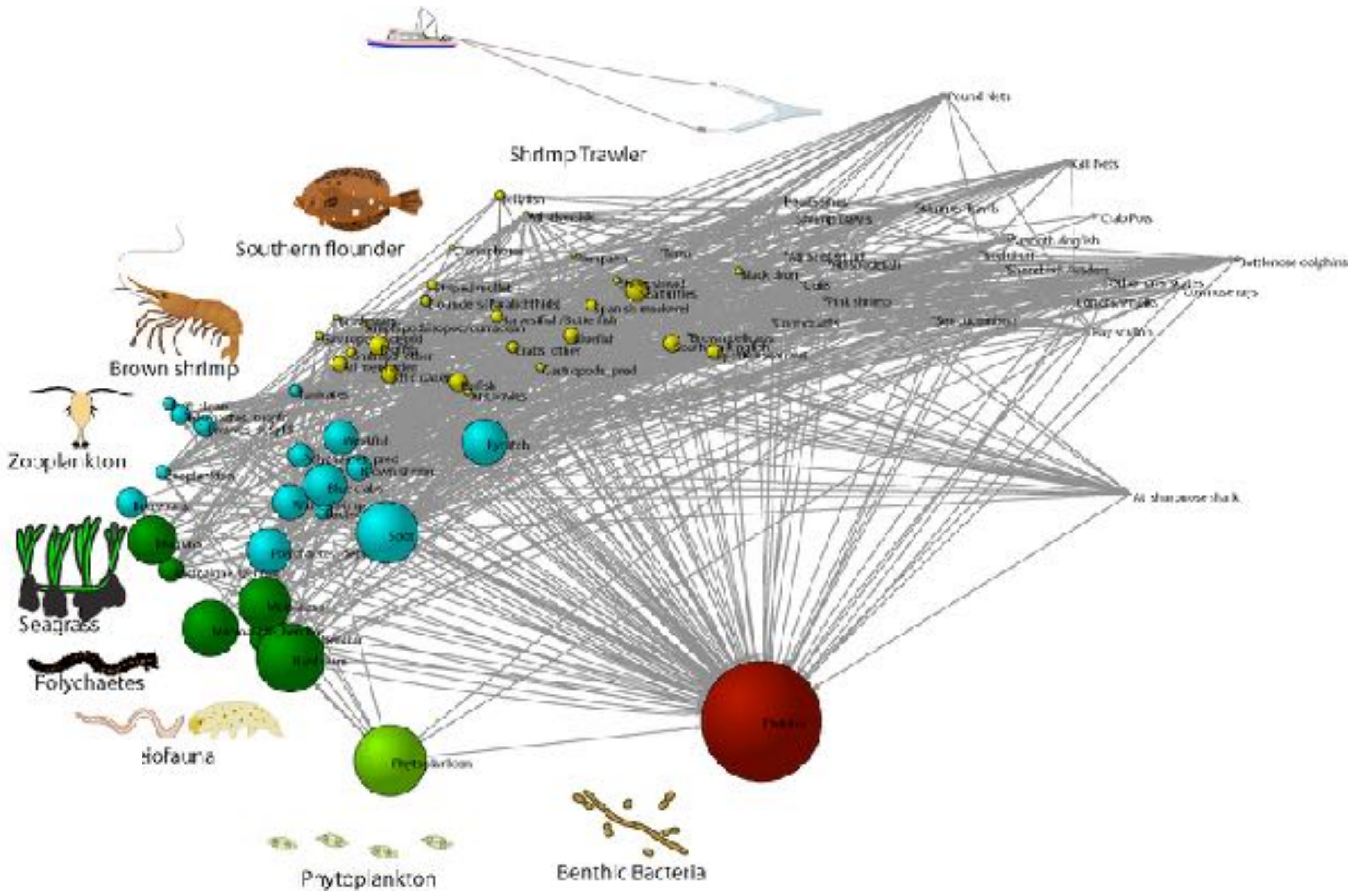
RECSYS 2019

Why Graph ?

1. **Universal language** for describing complex data [1]
   - Many fields have all chosen graph to depict entity interactions

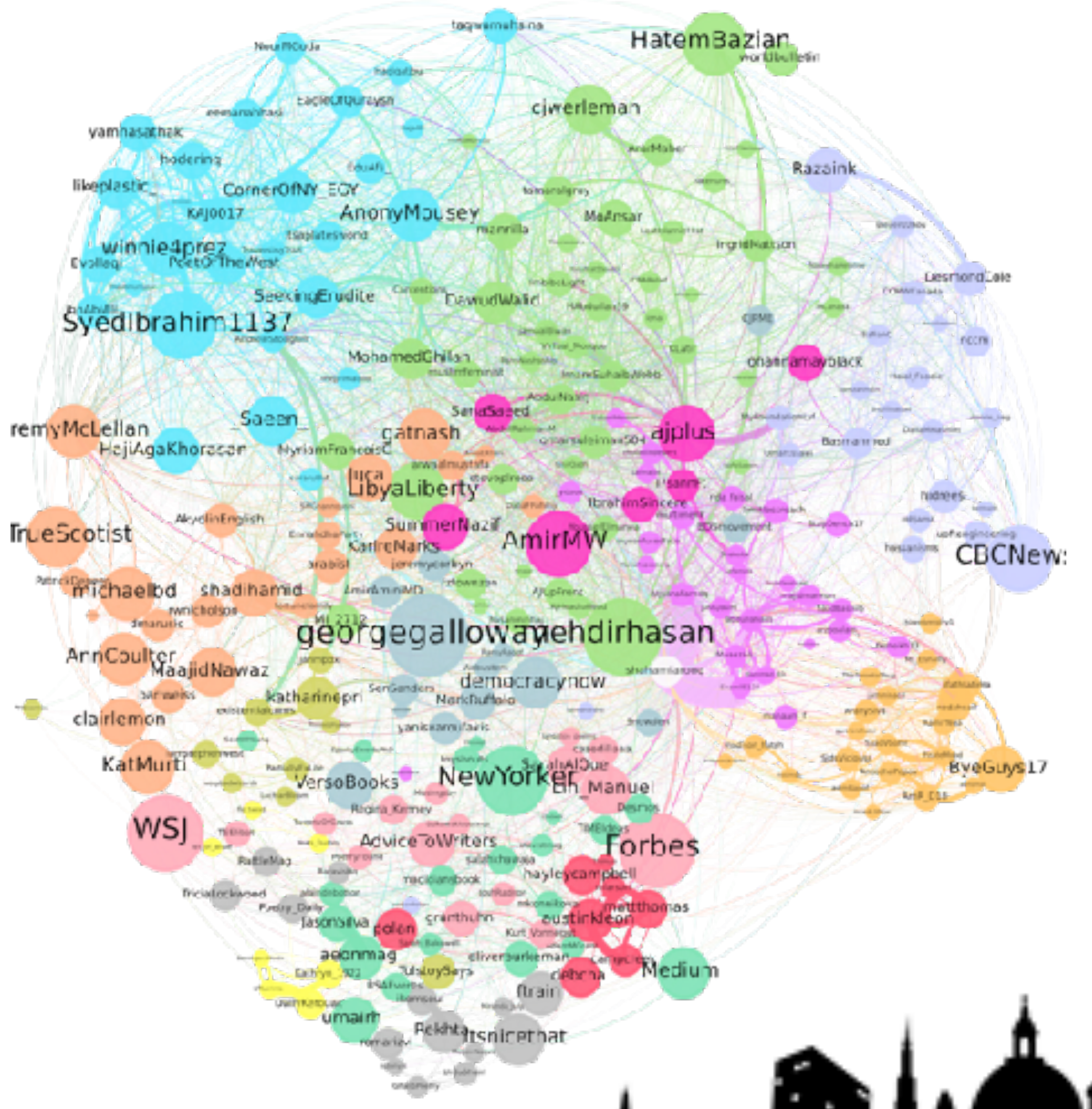
Ting-Hsiang Wang, Texas A&M University

Twitter Ego-Network



Core Sound Food Web

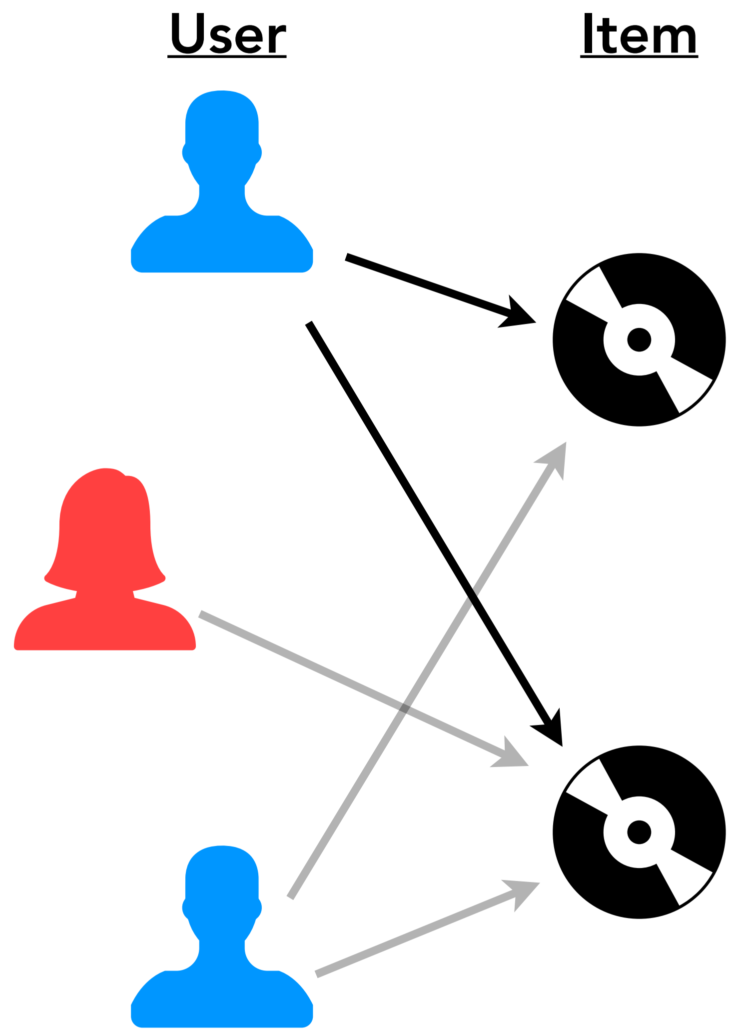
Delta Airline Route Map Network

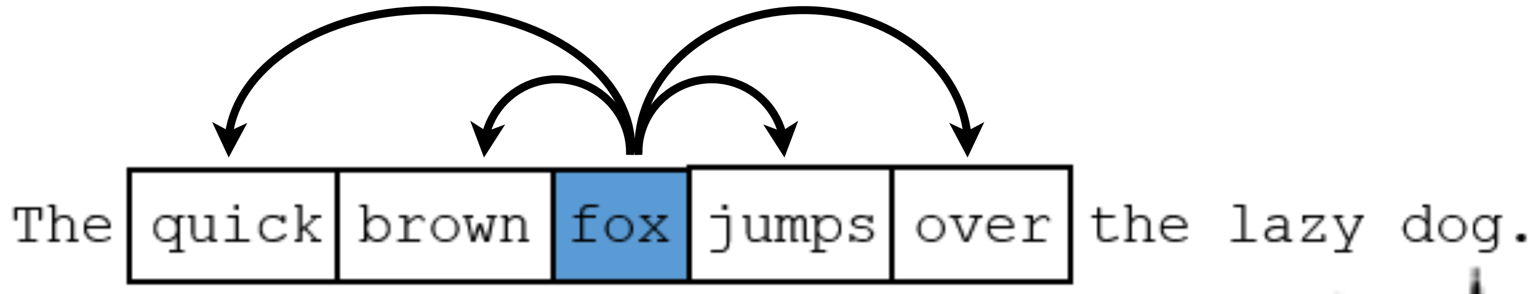


RECSYS 2019

Why Graph ?

1. **Universal language** for describing complex data [1]
    - Many fields have all chosen graph to depict entity interactions
2. **Shared vocabulary** (therefore **ideas**) between fields [1]
    - E.g., **distributional hypothesis** in linguistics vs. **CF** in REC systems



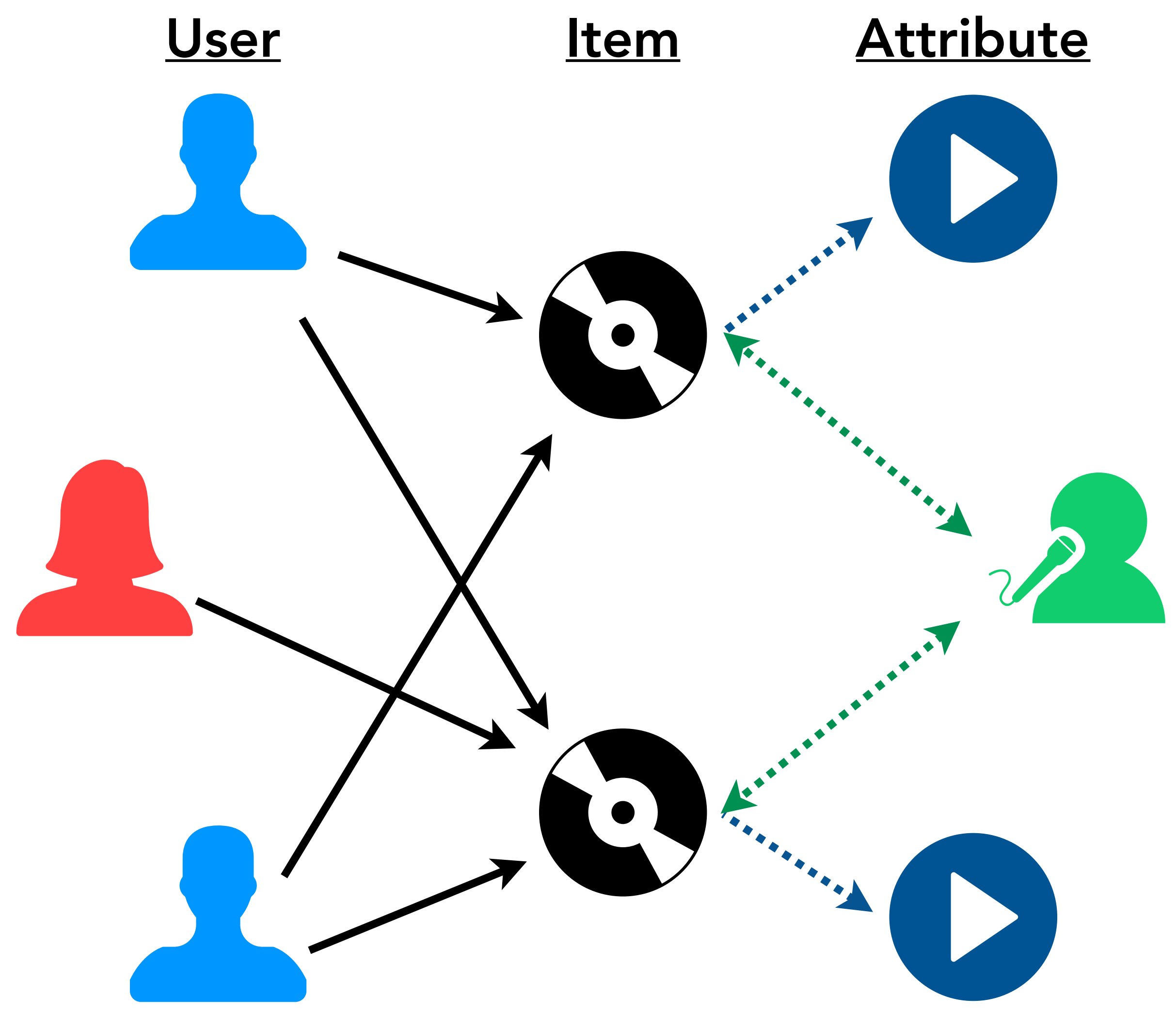


Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Why Graph ?

1. **Universal language** for describing complex data [1]
   - Many fields have all chosen graph to depict entity interactions
2. **Shared vocabulary** (therefore **ideas**) between fields [1]
   - E.g., **distributional hypothesis** in linguistics vs. **CF** in REC systems
3. **Holistic view** of complex systems of interactions
   - Different domains can **easily connect** and be **jointly mined**



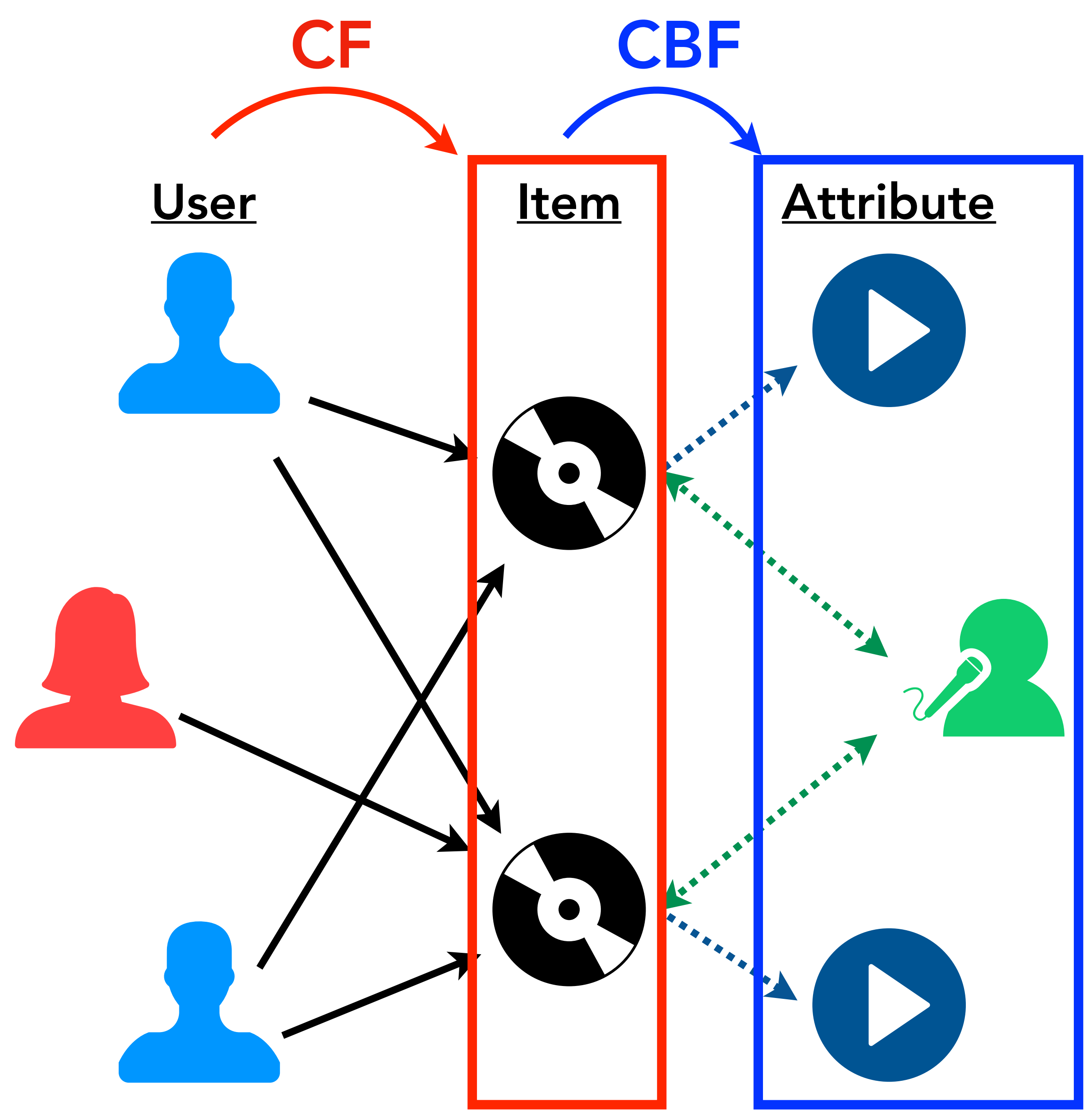
→ **HINs** easily integrates information from different domains

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Why Graph ?

1. **Universal language** for describing complex data [1]
   - Many fields have all chosen graph to depict entity interactions
2. **Shared vocabulary** (therefore **ideas**) between fields [1]
   - E.g., **distributional hypothesis** in linguistics vs. **CF** in REC systems
3. **Holistic view** of complex systems of interactions
   - Different domains can **easily connect** and be **jointly mined**
4. **General definition** of contexts used for entity comparison
   - Model **graph structure** instead specific types of relations

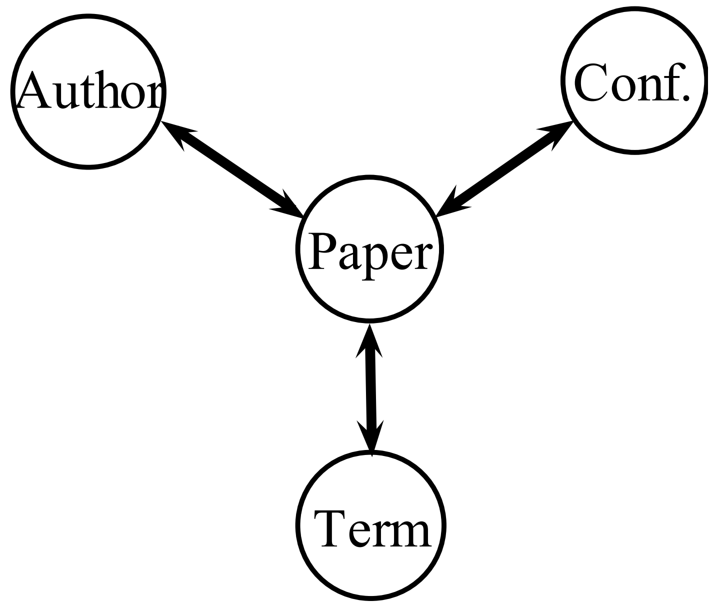


→ Different REC approaches, **similar structure**

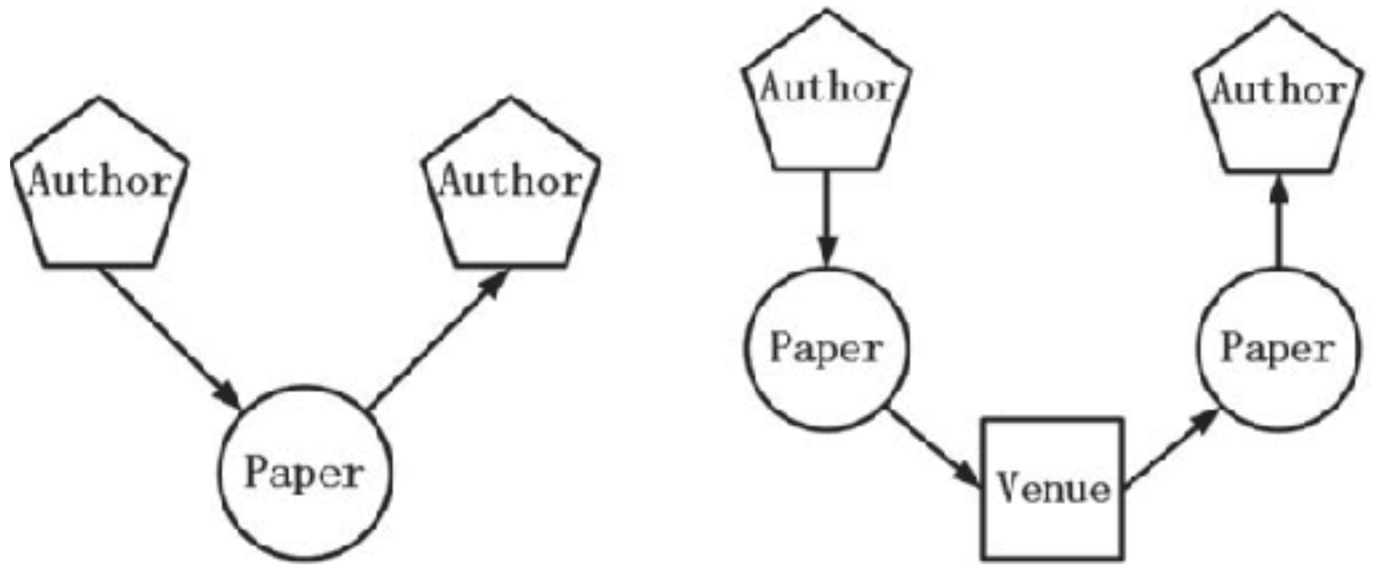
Why Graph ?

Ting-Hsiang Wang, Texas A&M University

1. **Universal language** for describing complex data [1]
   - Many fields have all chosen graph to depict entity interactions
2. **Shared vocabulary** (therefore **ideas**) between fields [1]
   - E.g., **distributional hypothesis** in linguistics vs. **CF** in REC systems
3. **Holistic view** of complex systems of interactions
   - Different domains can **easily connect** and be **jointly mined**
4. **General definition** of contexts used for entity comparison
   - Model **graph structure** instead specific types of relations
5. **Sophisticated models** available from network-related researches
   - Network schema, meta-path, subgraph matching, information propagation, etc.



Star-schema [2]



Meta-path [3]

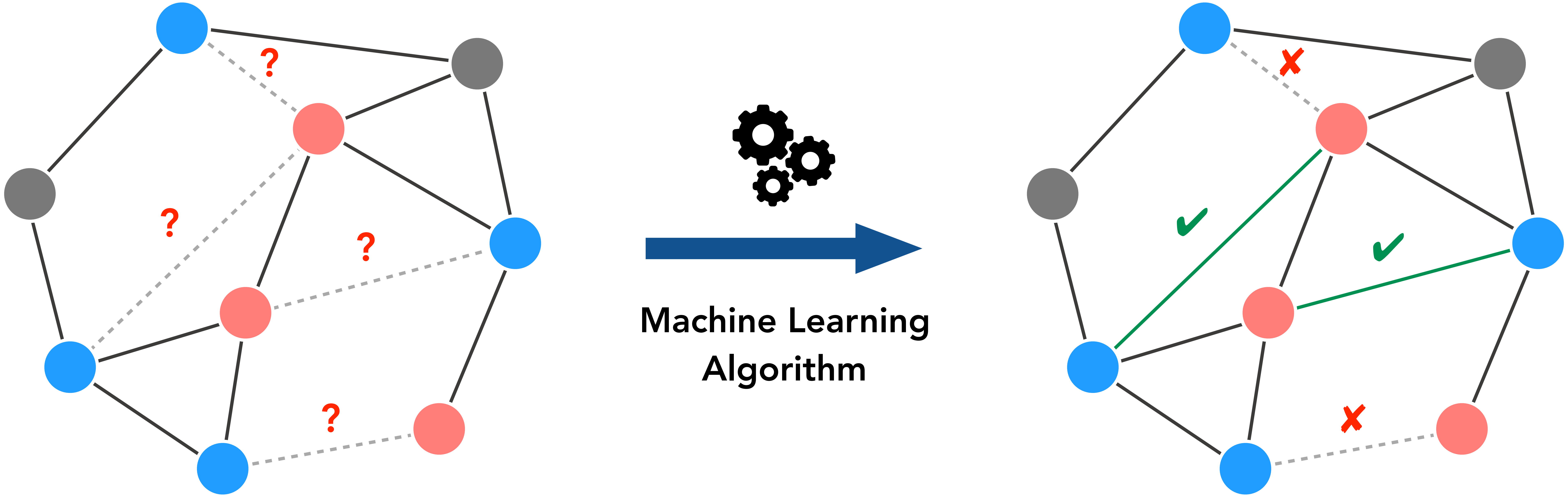
RECSYS 2019

Why Graph ?

1. **Universal language** for describing complex data [1]
2. **Shared vocabulary** (therefore **ideas**) between fields [1]
3. **Holistic view** of complex systems of interactions
4. **General definition** of contexts used for entity comparison
5. **Sophisticated models** available from network-related researches

RECSYS 2019

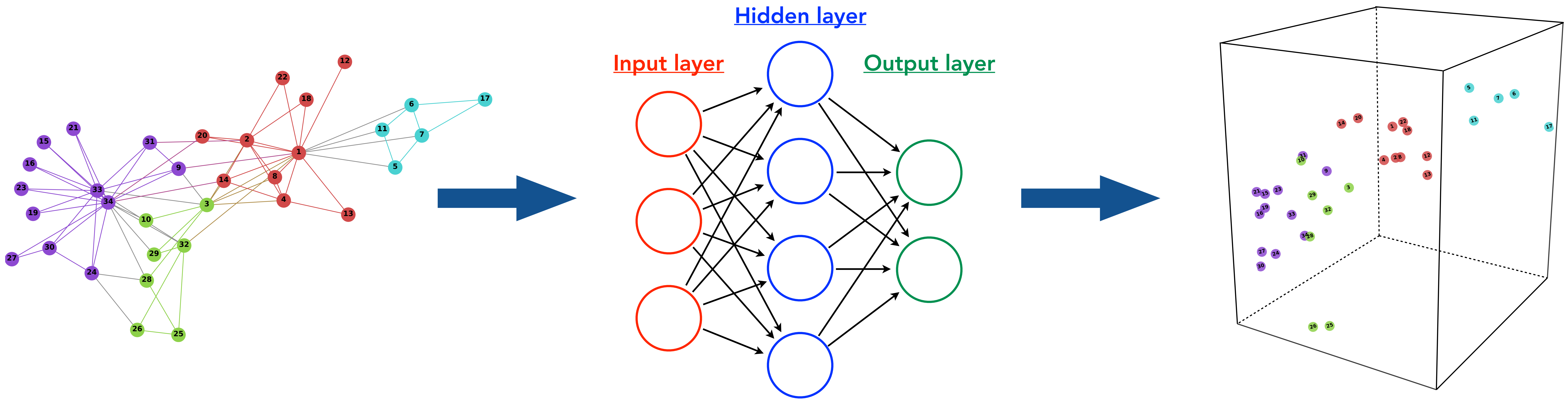
REC as Link Prediction on Graphs



- **Many ways to compare** node similarity: entity types, shared neighbors, distance, etc.
- Early REC models, e.g., CF and CBF, define intuitive relations for similarity measurement
- As models mature and data diversity skyrockets nowadays, **designing features for REC becomes increasingly challenging**

Graph Embedding Pipeline

Supply Graph | **Train Neural Network** | **Return Embeddings** (from hidden Layer)



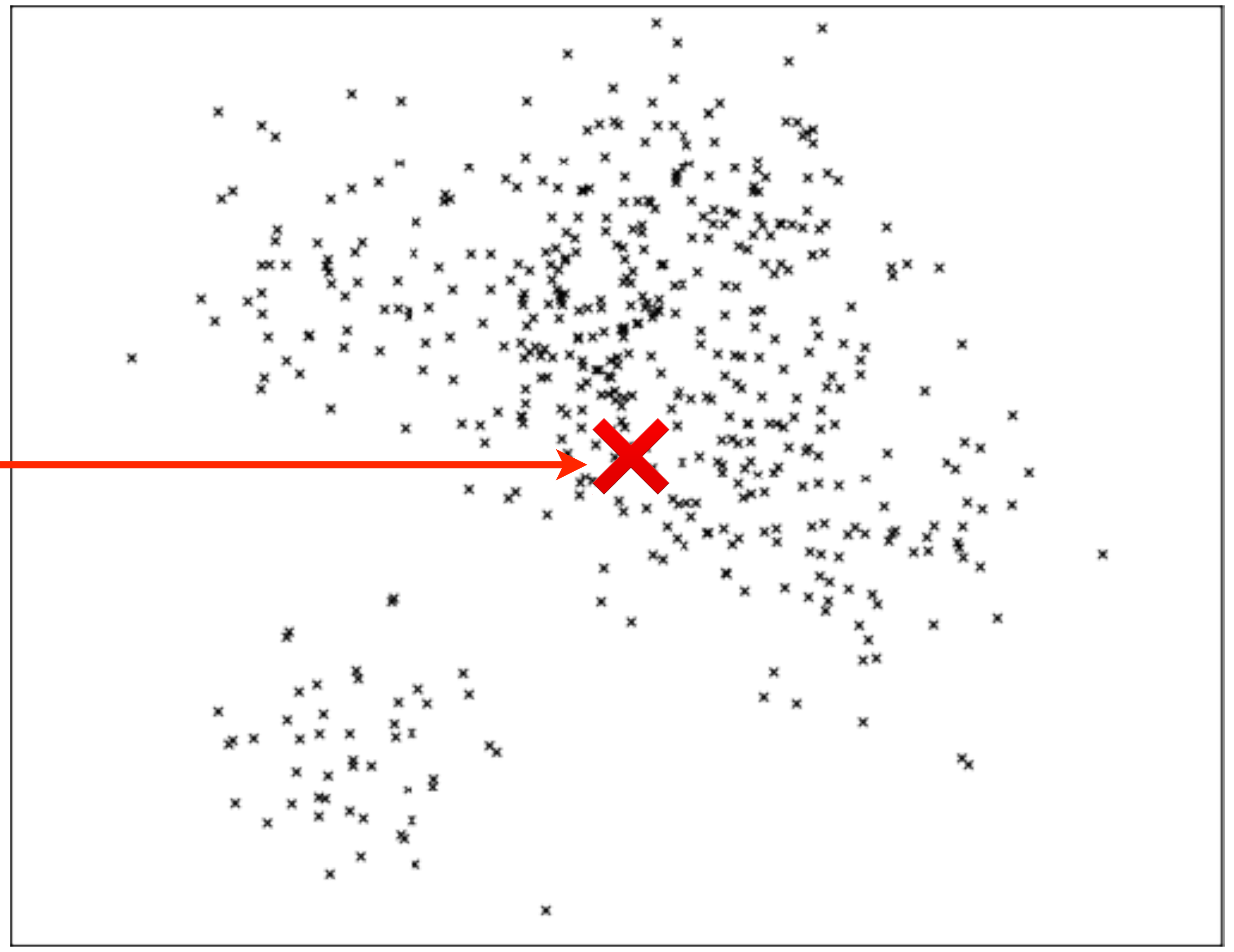
- GE for REC exploits observed links as **graph structures** to predict unobserved links
- Entities are converted into **spatial features (embeddings)** in the hidden layer and iteratively updated such that their interactions approximate values in the output layer
- Entity relatedness is preserved as **spatial properties**, e.g., distance and angle

Ting-Hsiang Wang, Texas A&M University

Why Embedding ?

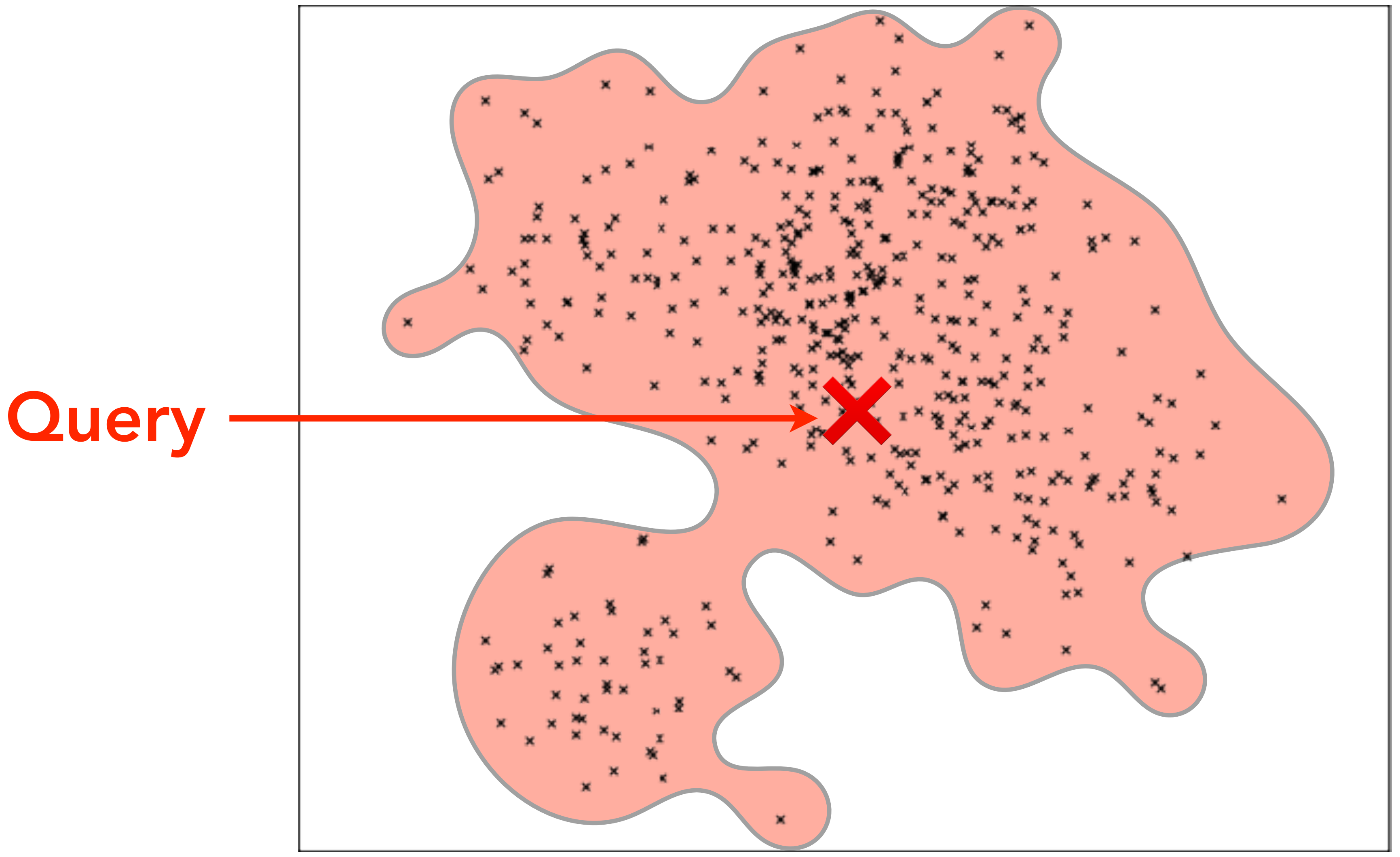
1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log\ n)$ [4]

Query



Why Embedding ?

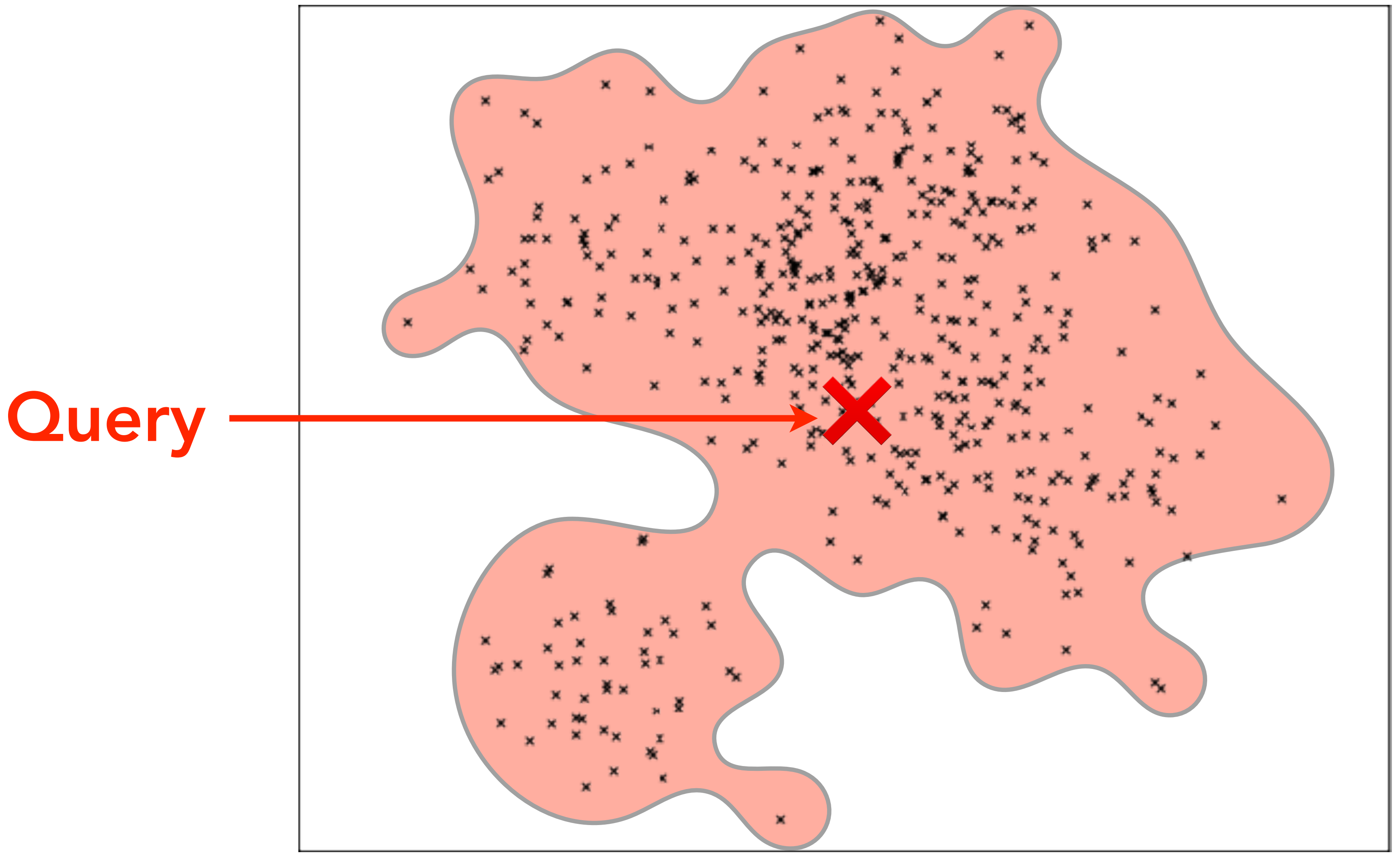
1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]



→ **Brute force exhaustive search** finds the closest nearest neighbors in $O(\log n!)$

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]

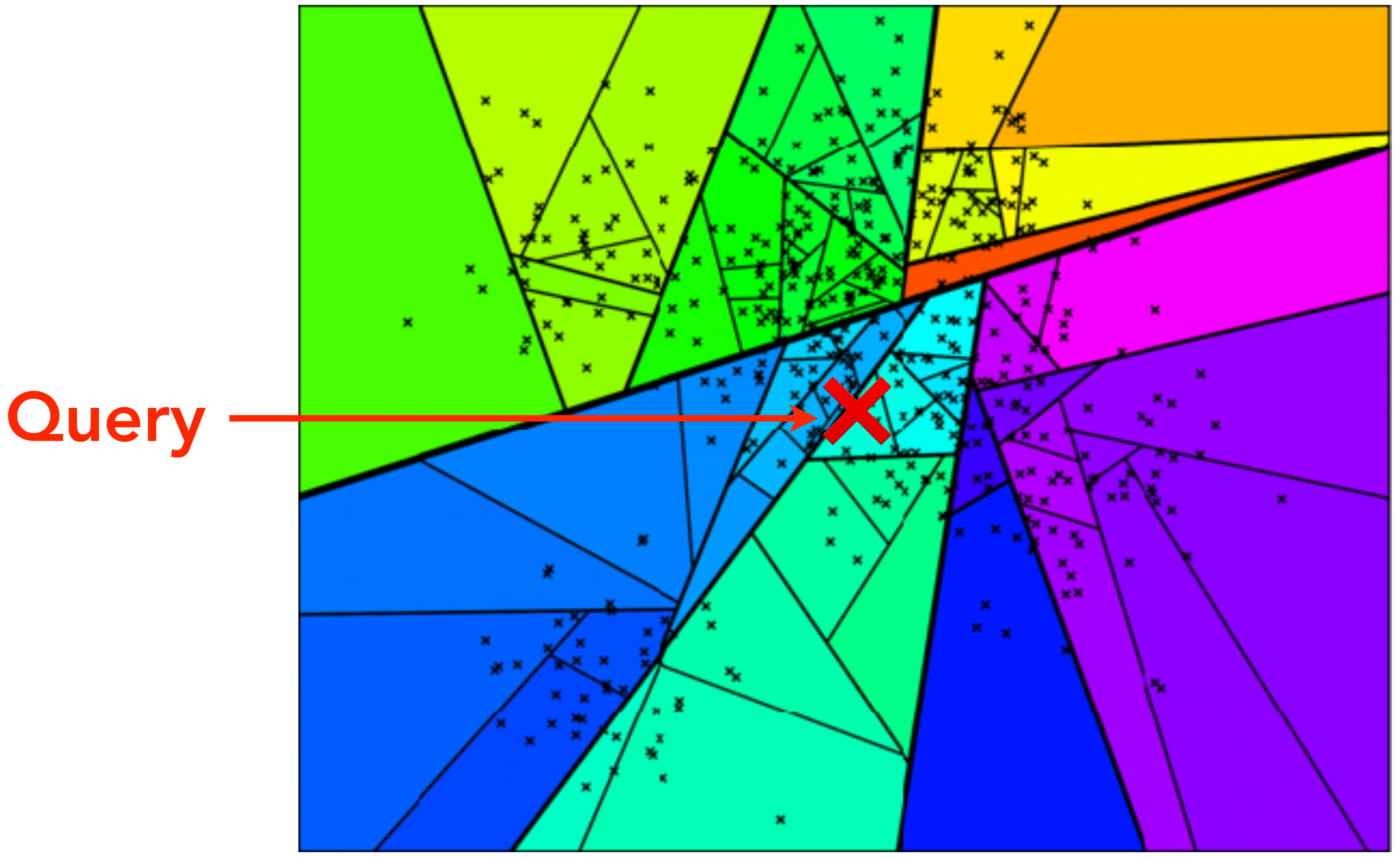


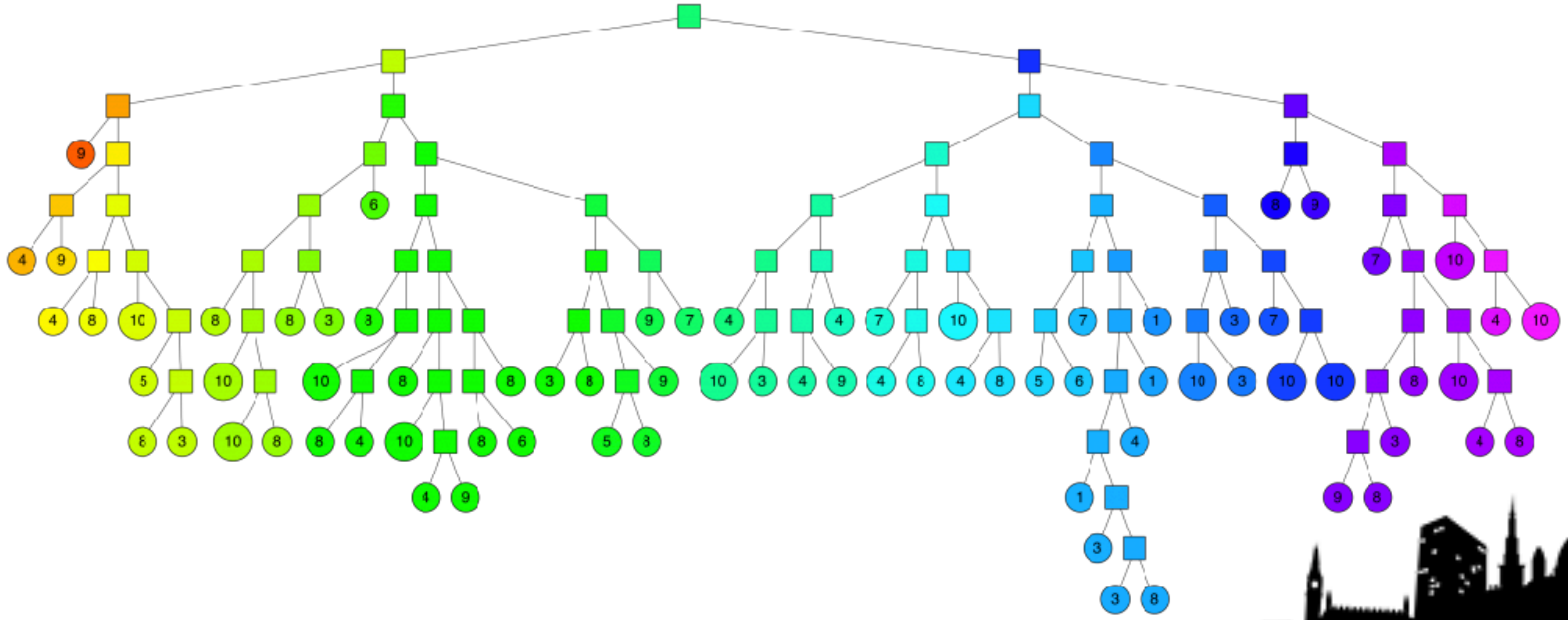
→ **Brute force exhaustive search** finds the closest nearest neighbors in $O(\log n!)$

Ting-Hsiang Wang, Texas A&M University

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
    - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]

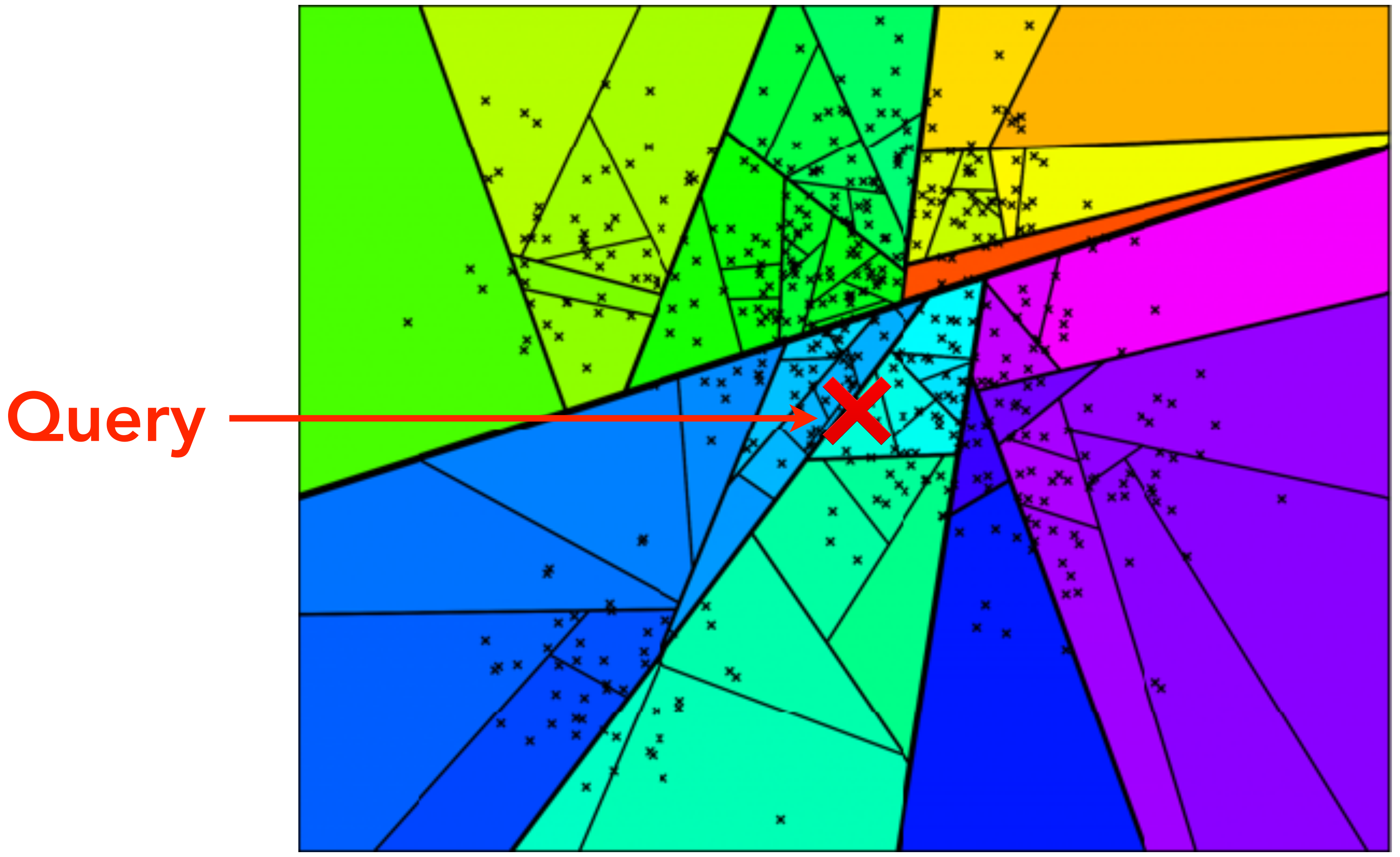


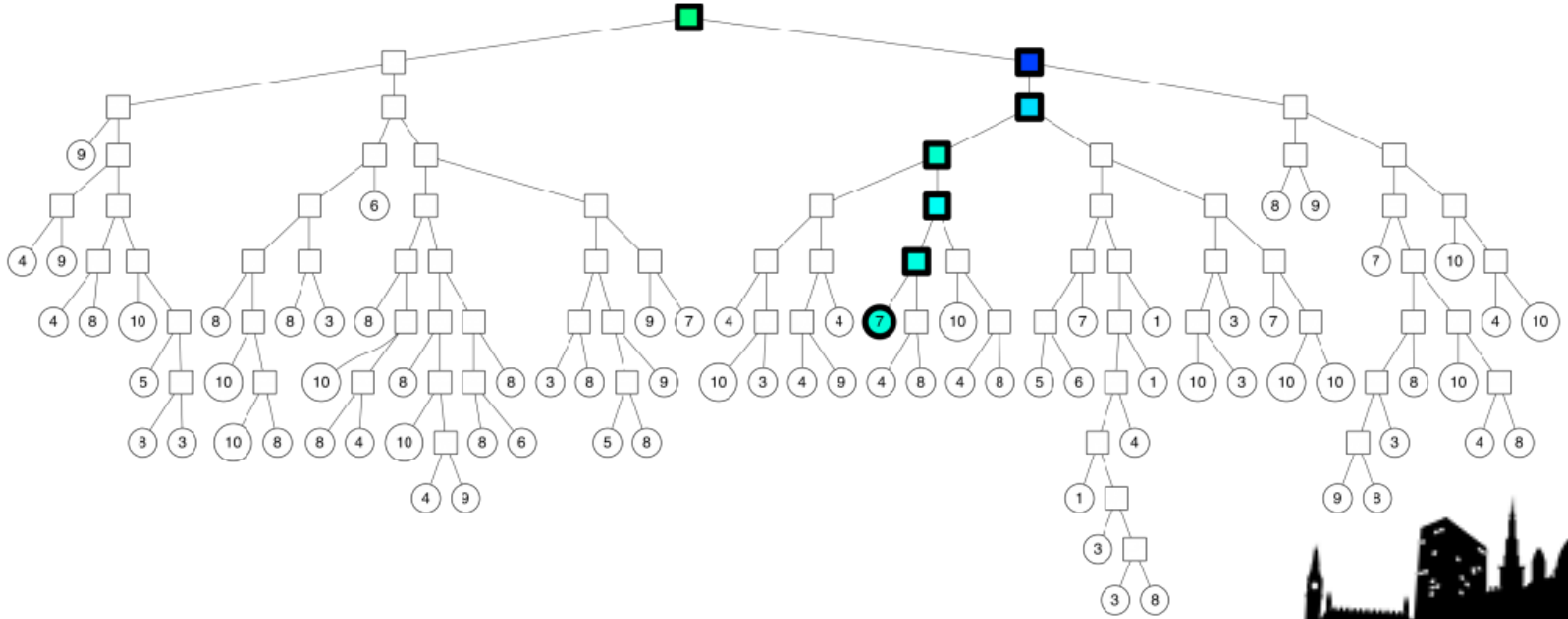


→ **Spotify ANNOY** builds binary tree of subdivisions to quickly find closest neighbors

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log\, n)$ [4]





→ **Spotify ANNOY** builds binary tree of subdivisions to quickly find closest neighbors

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]
2. **Efficient pairwise comparison** due to **dimensionality reduction (DR)**
   - Lower dimension costs less to calculate feature similarity

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
   - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]
2. **Efficient pairwise comparison** due to **dimensionality reduction (DR)**
   - Lower dimension costs less to calculate feature similarity
3. **Reduced space complexity** due to DR
   - Lower dimension costs less storage space

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Why Embedding ?

1. **Efficient retrieval** from **approximate nearest neighbor (ANN)** search methods
    - E.g., **Spotify ANNOY** reduces **curse of dimensionality** during online search by maintaining a binary tree of subdivisions such that **good enough results** can be found in $O(\log n)$ [4]
2. **Efficient pairwise comparison** due to **dimensionality reduction (DR)**
    - Lower dimension costs less to calculate feature similarity
3. **Reduced space complexity** due to DR
    - Lower dimension costs less storage space
4. **Transfer learning** with **pertained embeddings**
    - Pretrained embeddings are better guesses than randomly initialized vectors

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

GE for REC : Challenges

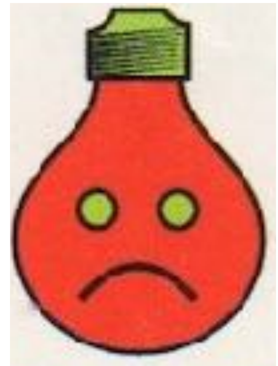
- So **graph embedding** is GREAT for **recommendation** :
    - **Reduces data sparsity** and **cold start** via integrating auxiliary information
    - Provides **holistic view of REC problem** and **jointly mines different relations** in terms of **graph structures**
    - **Trains fast**, **compares fast**, and **retrieves fast** while taking **less space**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

GE for REC : Challenges

- So **graph embedding** is GREAT for **recommendation** :
  - **Reduces data sparsity** and **cold start** via integrating auxiliary information
  - Provides **holistic view of REC problem** and **jointly mines different relations** in terms of **graph structures**
  - **Trains fast**, **compares fast**, and **retrieves fast** while taking **less space**



- **What's the catch ?**

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Lecture Agenda

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

(Sean) **Q2. SMORe** modularization of GE and benefits

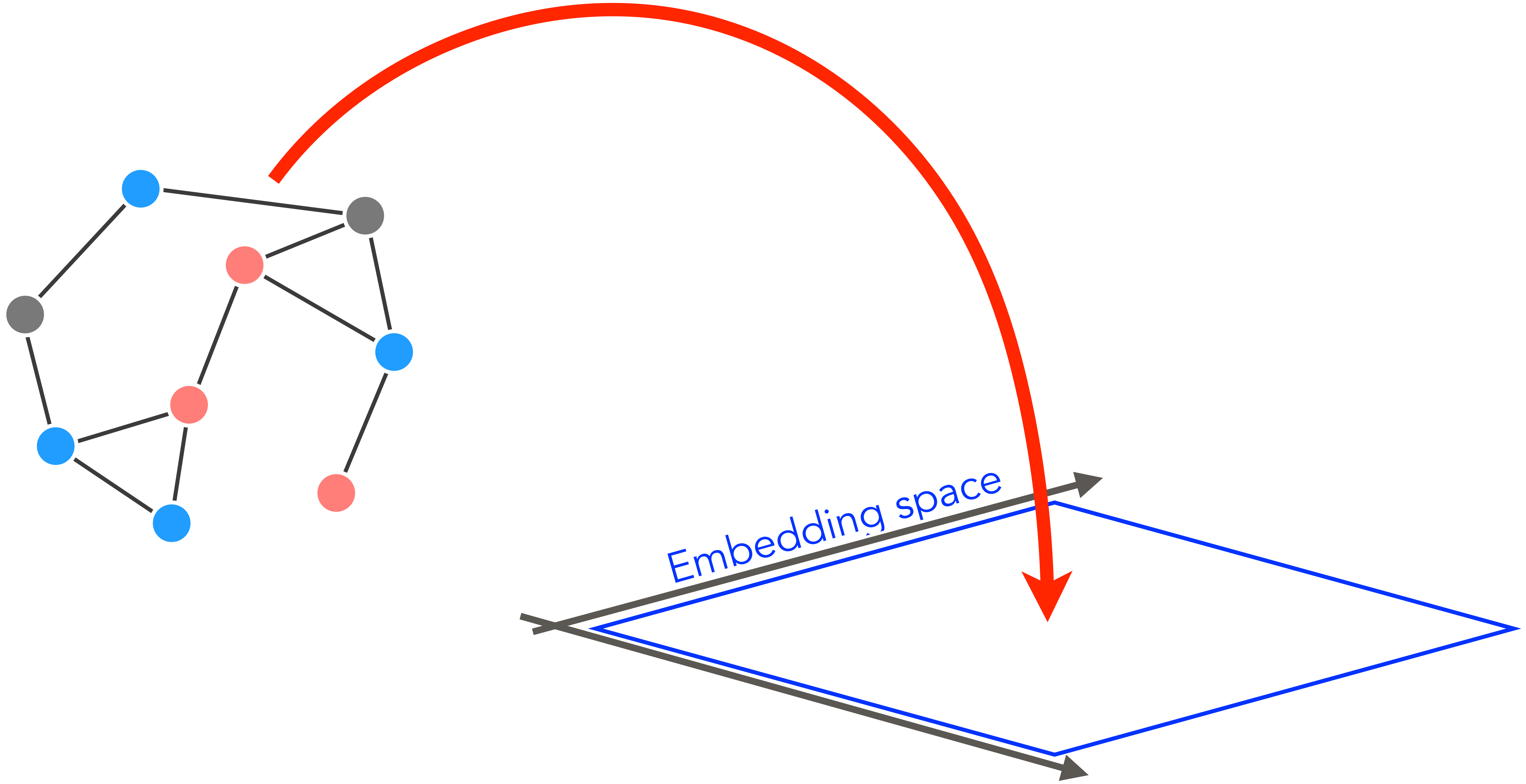
(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

Ting-Hsiang Wang, Texas A&M University

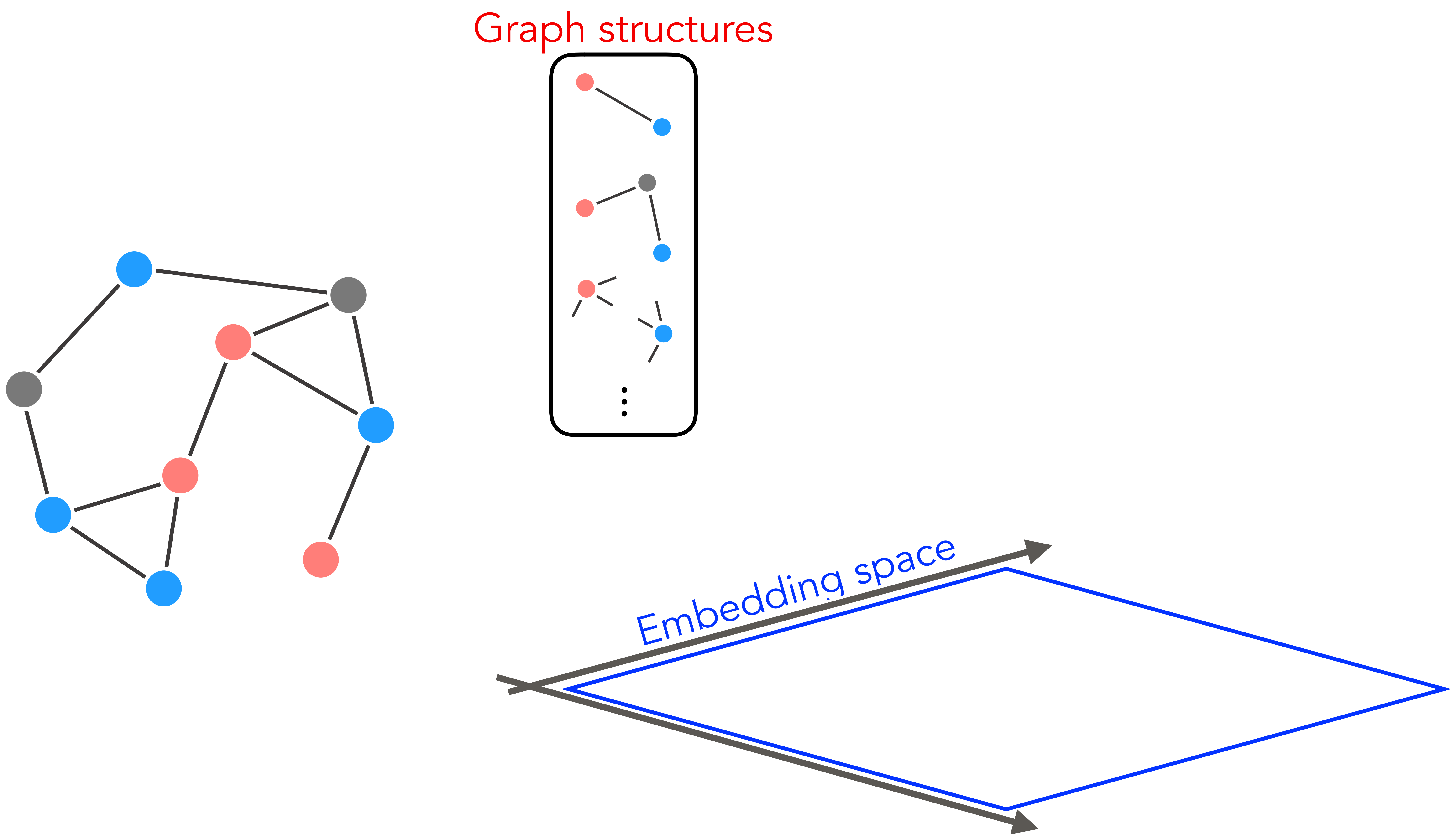
RECSYS 2019

Let's look at GE process …

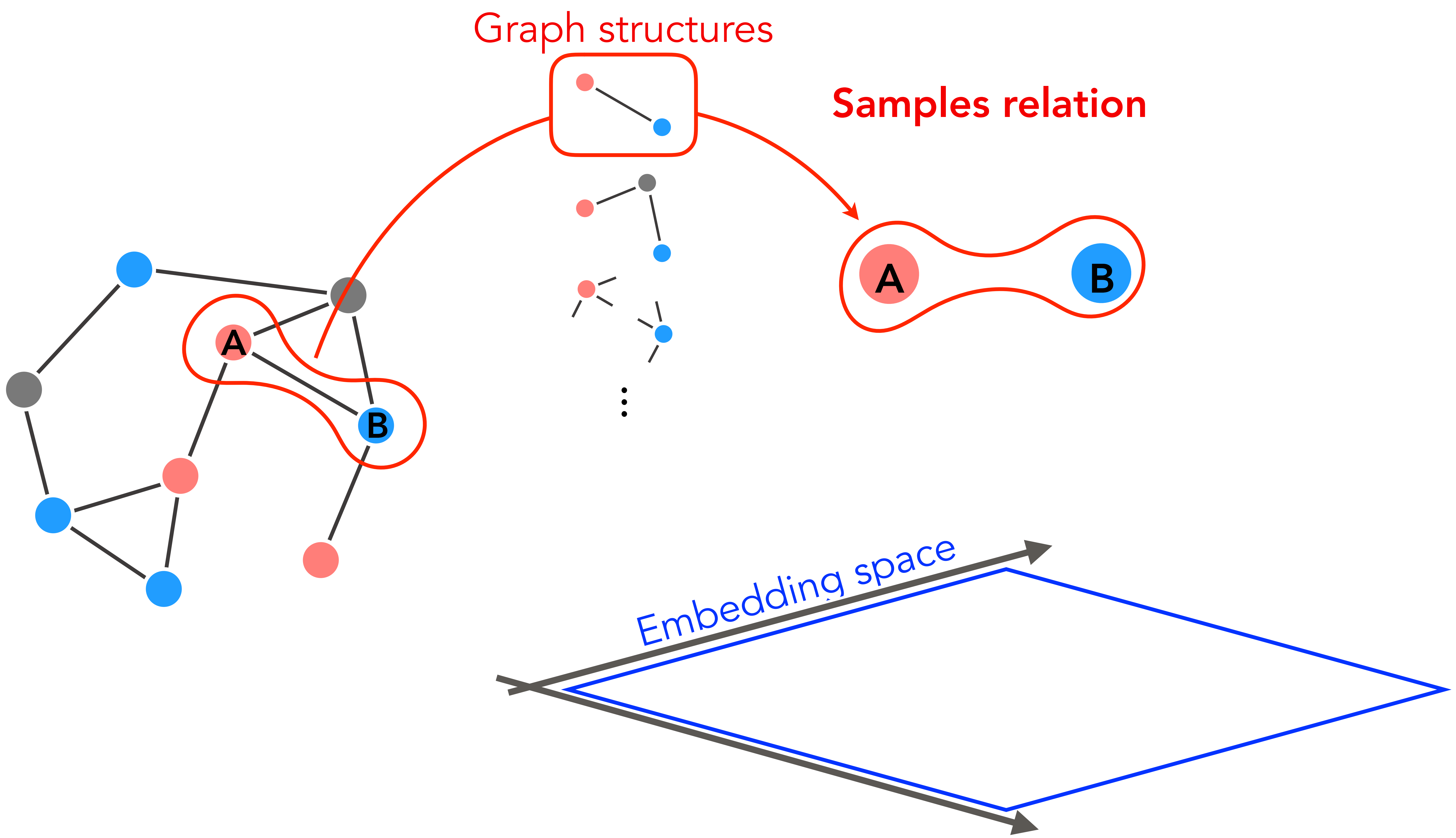


Ting-Hsiang Wang, Texas A&M University

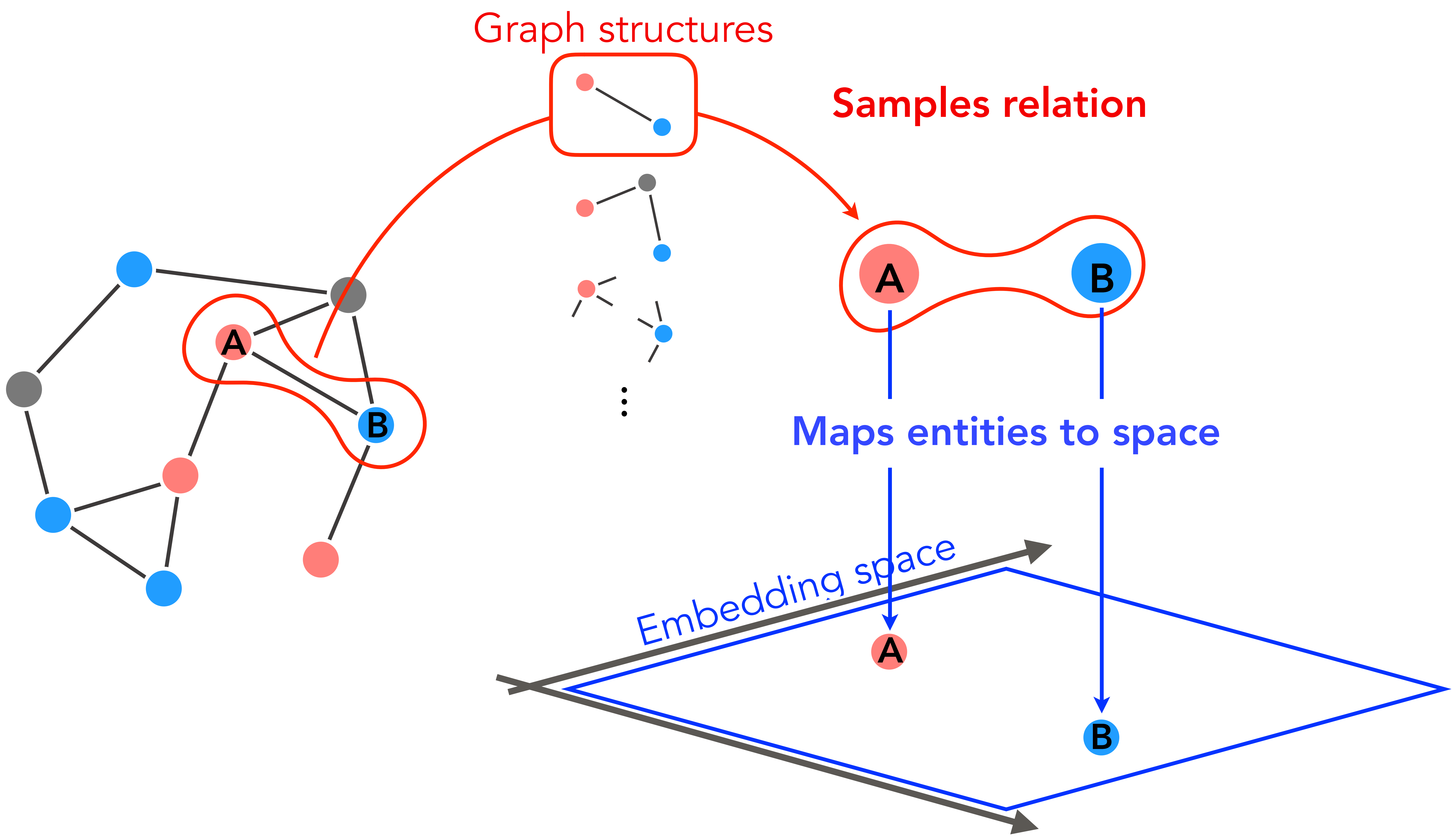
Let's look at GE process ...



Let's look at GE process …

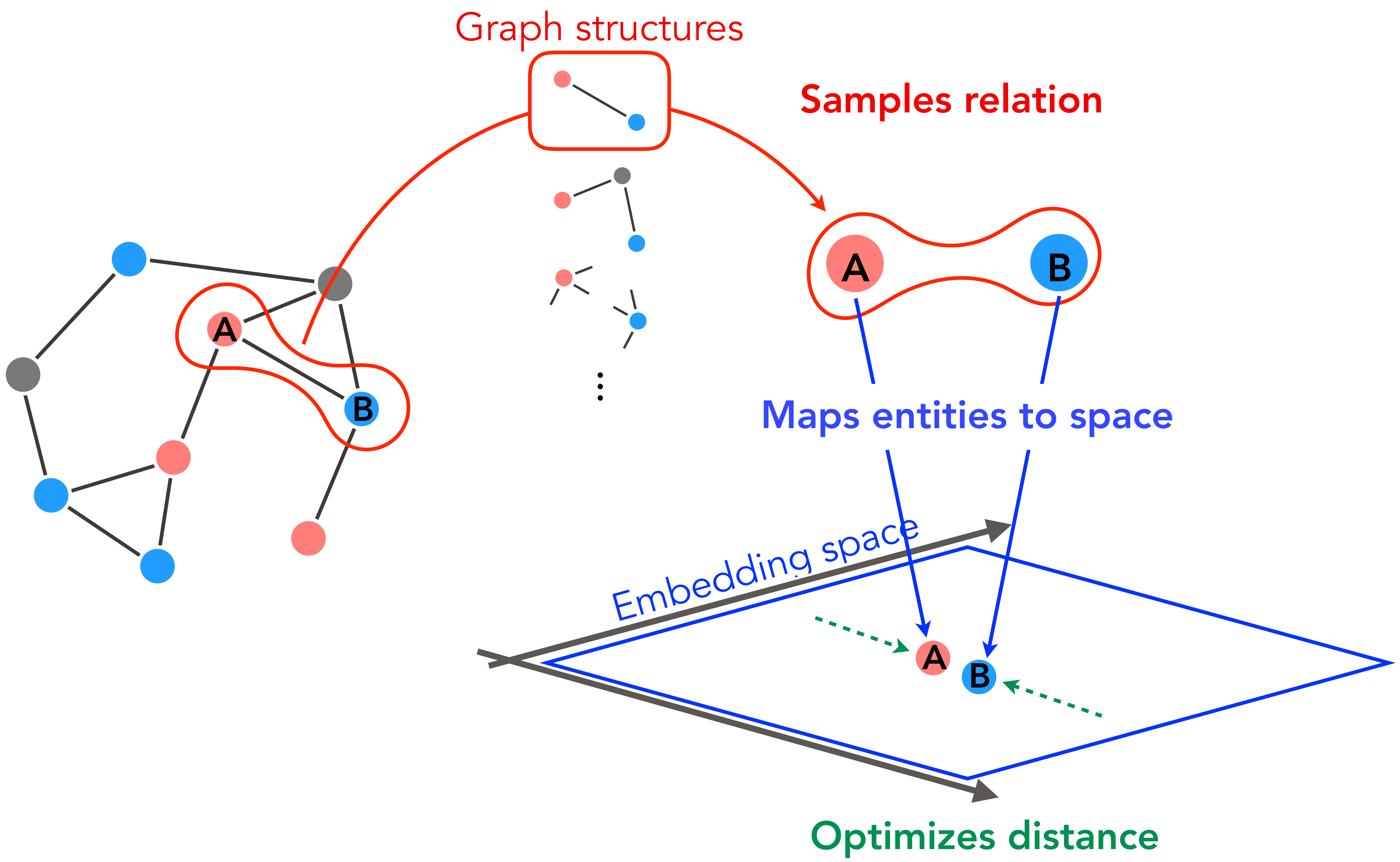


Let's look at GE process …

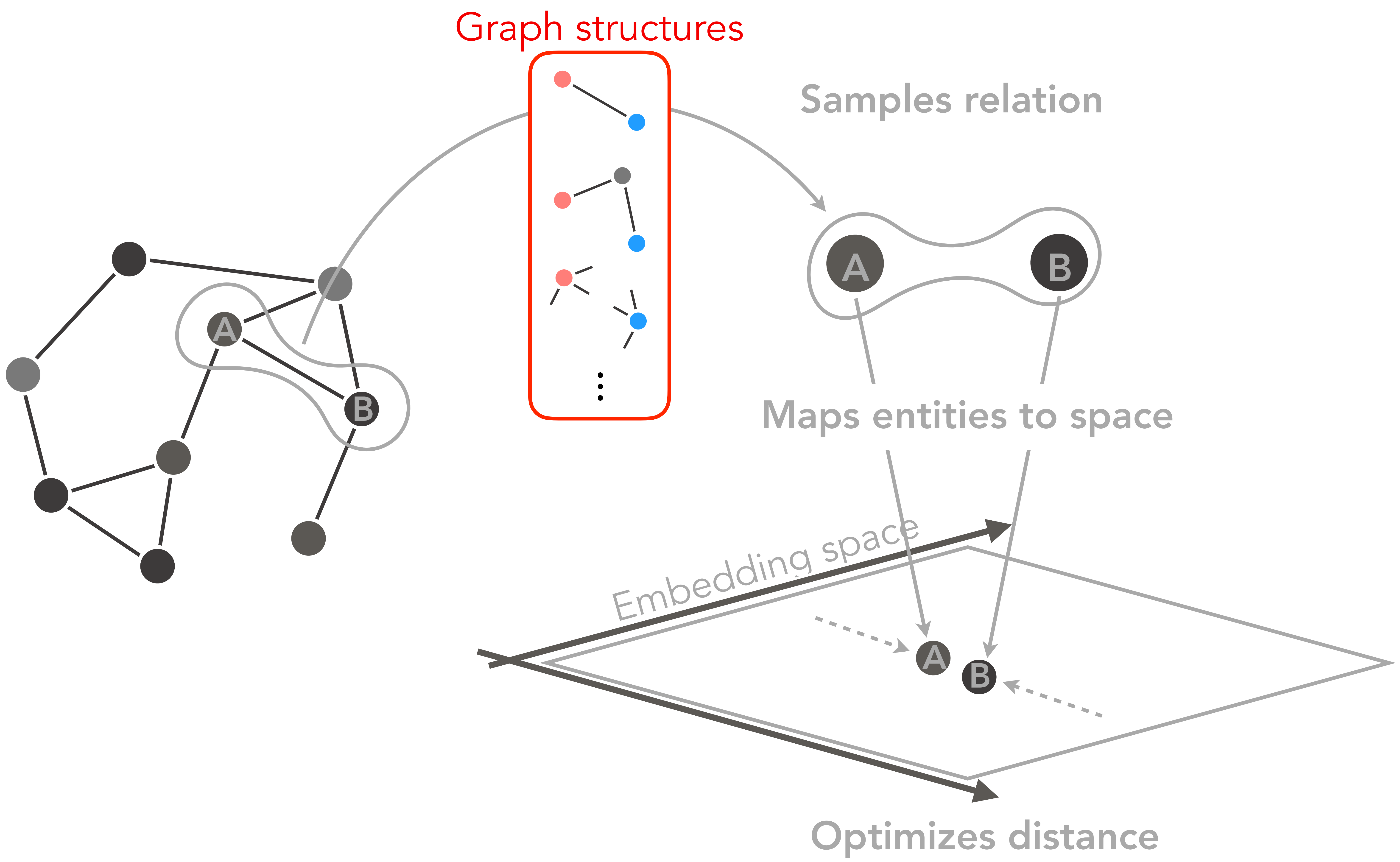


Ting-Hsiang Wang, Texas A&M University

Let’s look at GE process …



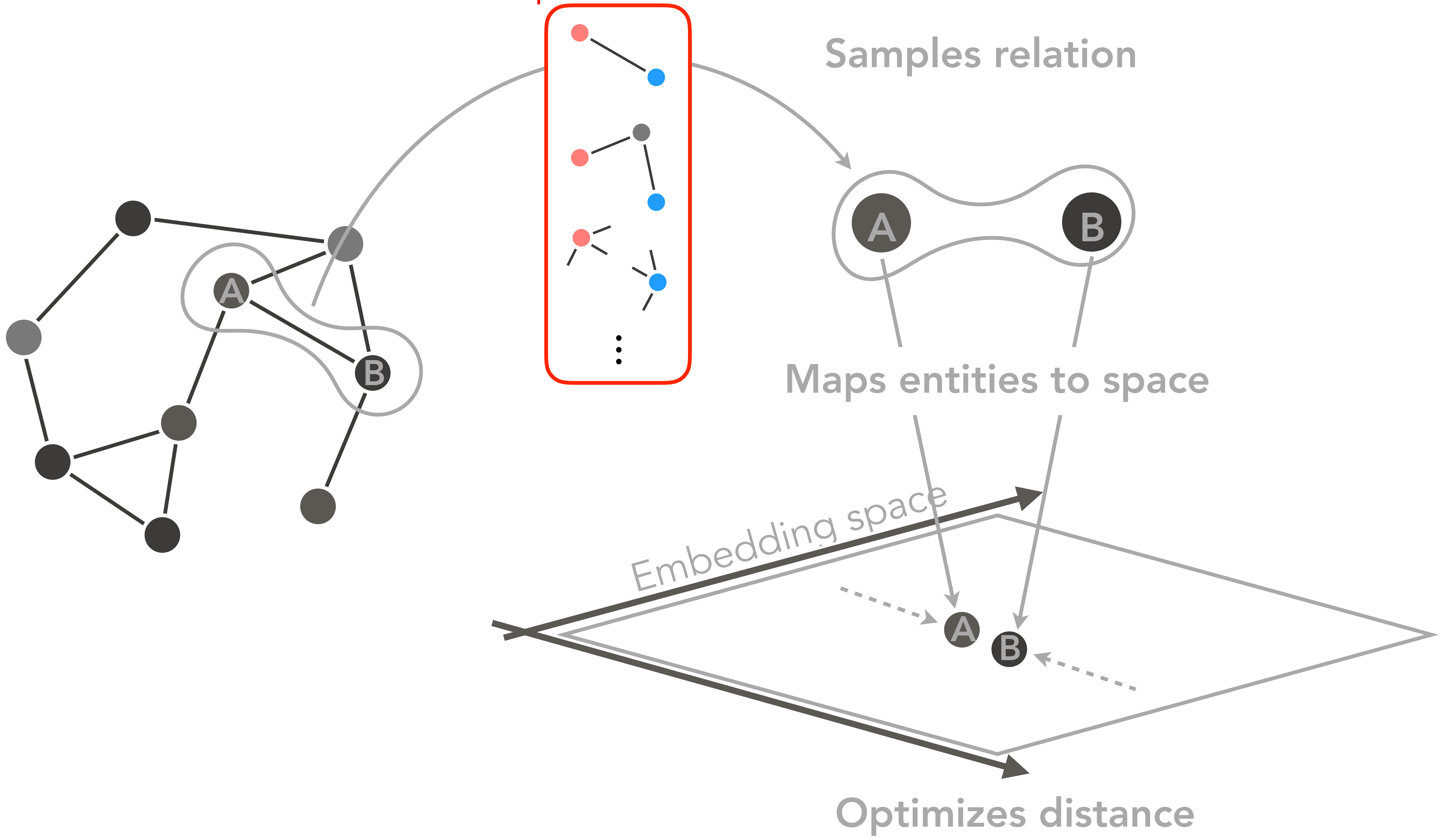
Challenge (1) : Select structure is HARD



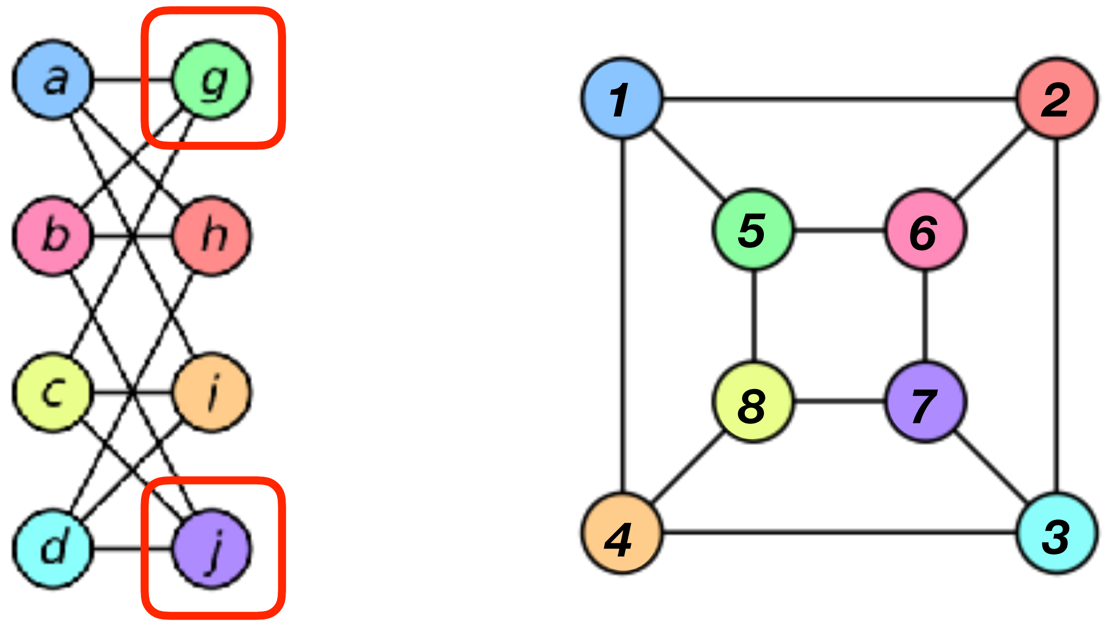
→ Unsure **which graph structures**

best model current REC task

Challenge (1) : Select structure is HARD

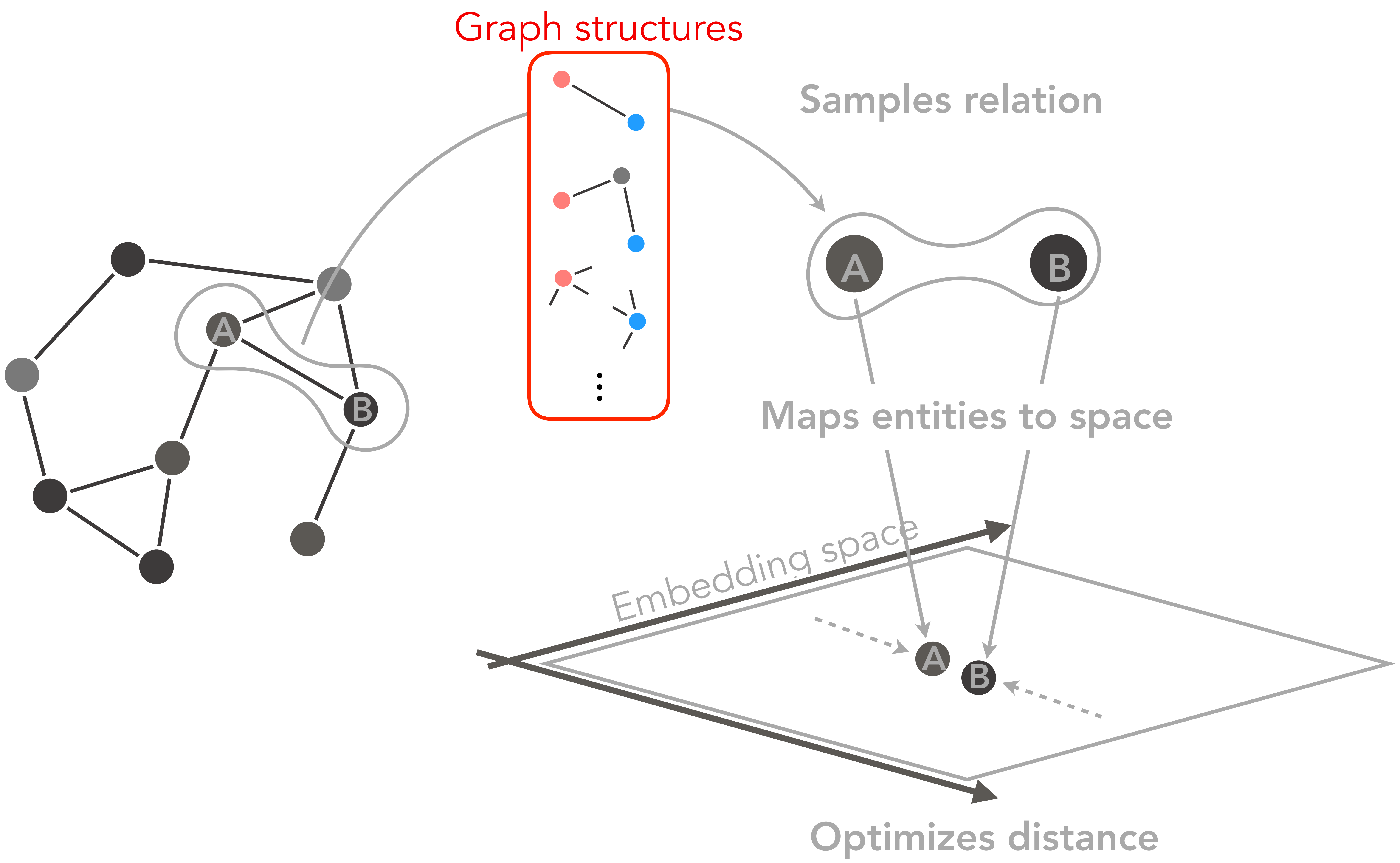


→ Unsure **which graph structures** best model current REC task

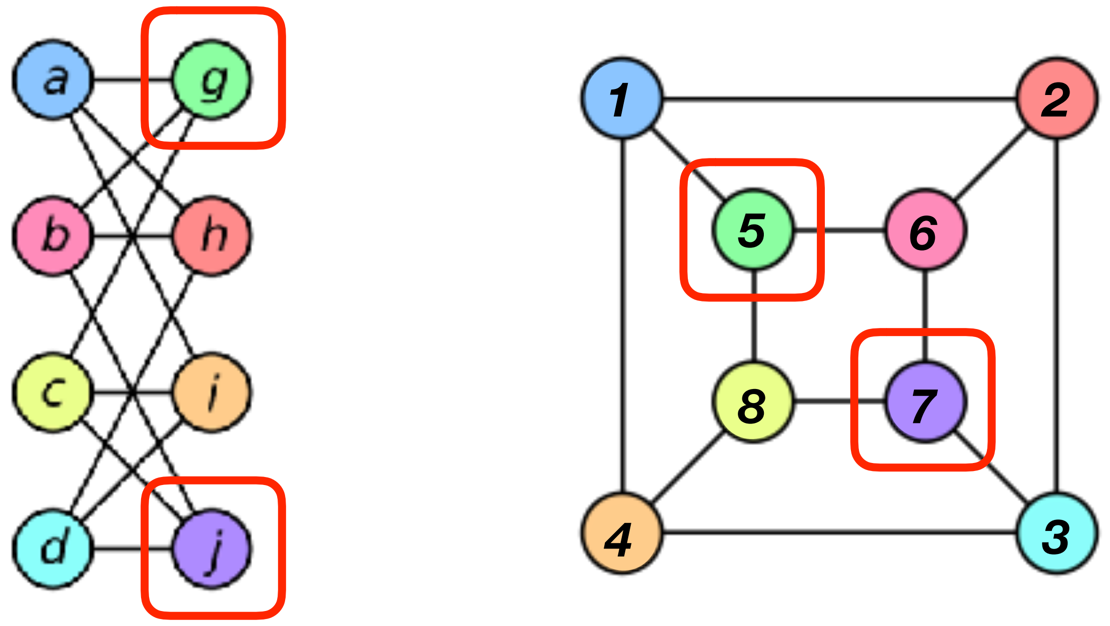


In bipartite, model g & j by **neighborhood** is intuitive; but in planar, 5 & 7 are also shown similar in **betweenness**

Challenge (1) : Select structure is HARD



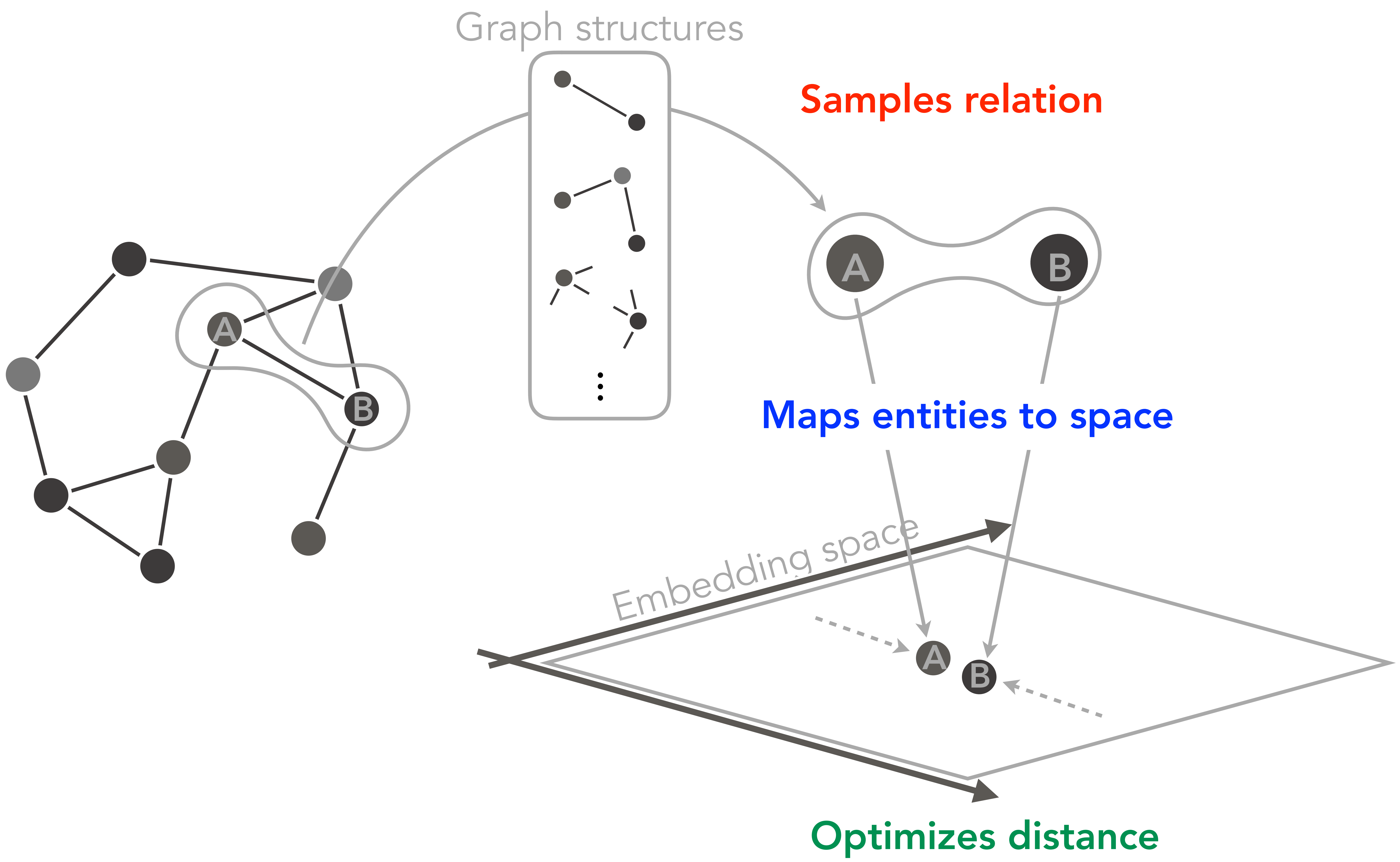
→ Unsure **which graph structures** best model current REC task



In bipartite, model g & j by **neighborhood** is intuitive; but in planar, 5 & 7 are also shown similar in **betweenness**

Ting-Hsiang Wang, Texas A&M University

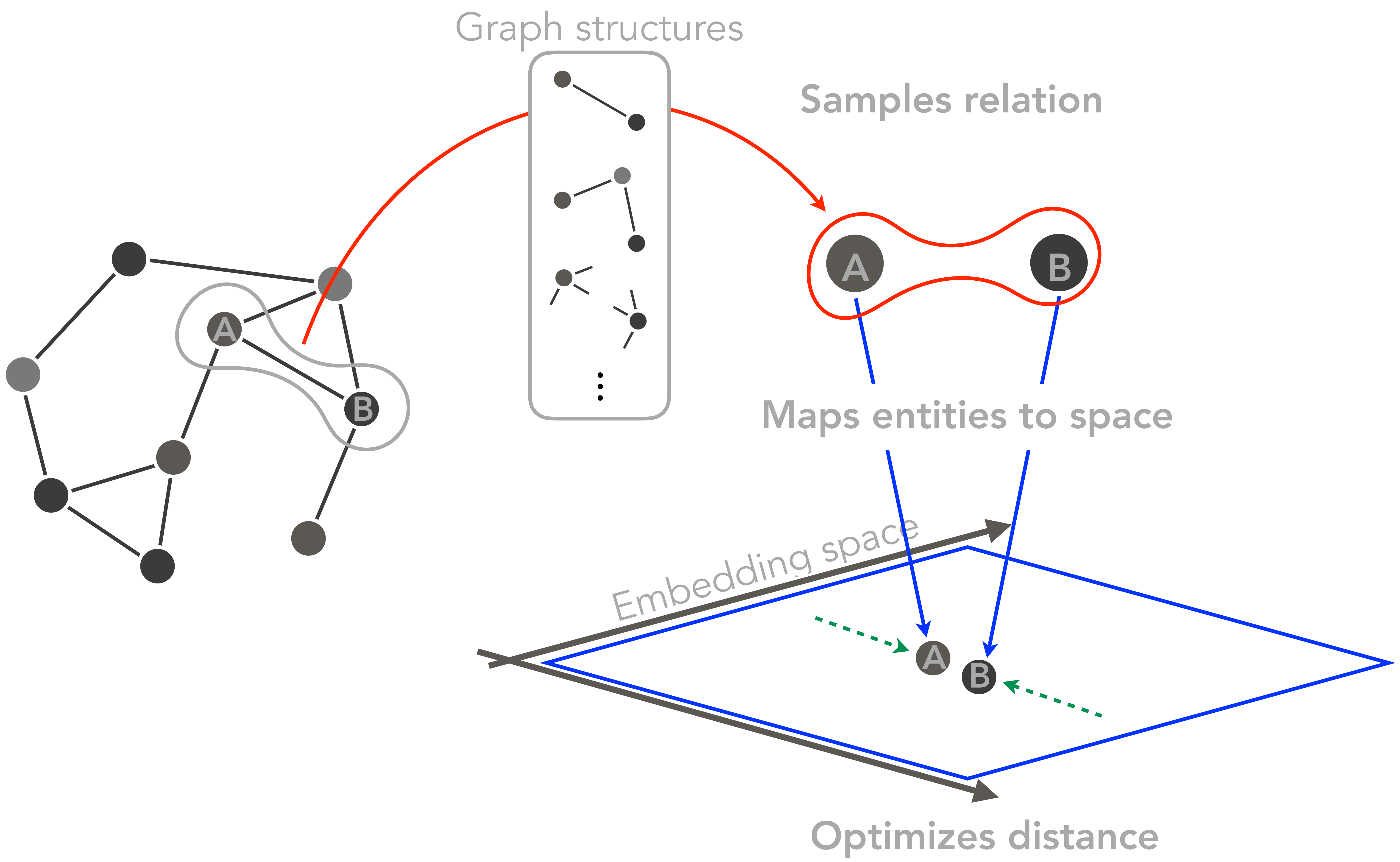
Challenge (2) : Customize GE model



→ Unsure **which graph structures** best model current REC task

→ Wanna **customize GE methods** during each stage of training

Challenge (3) : Make fair comparison



→ Unsure **which graph structures** best model current REC task

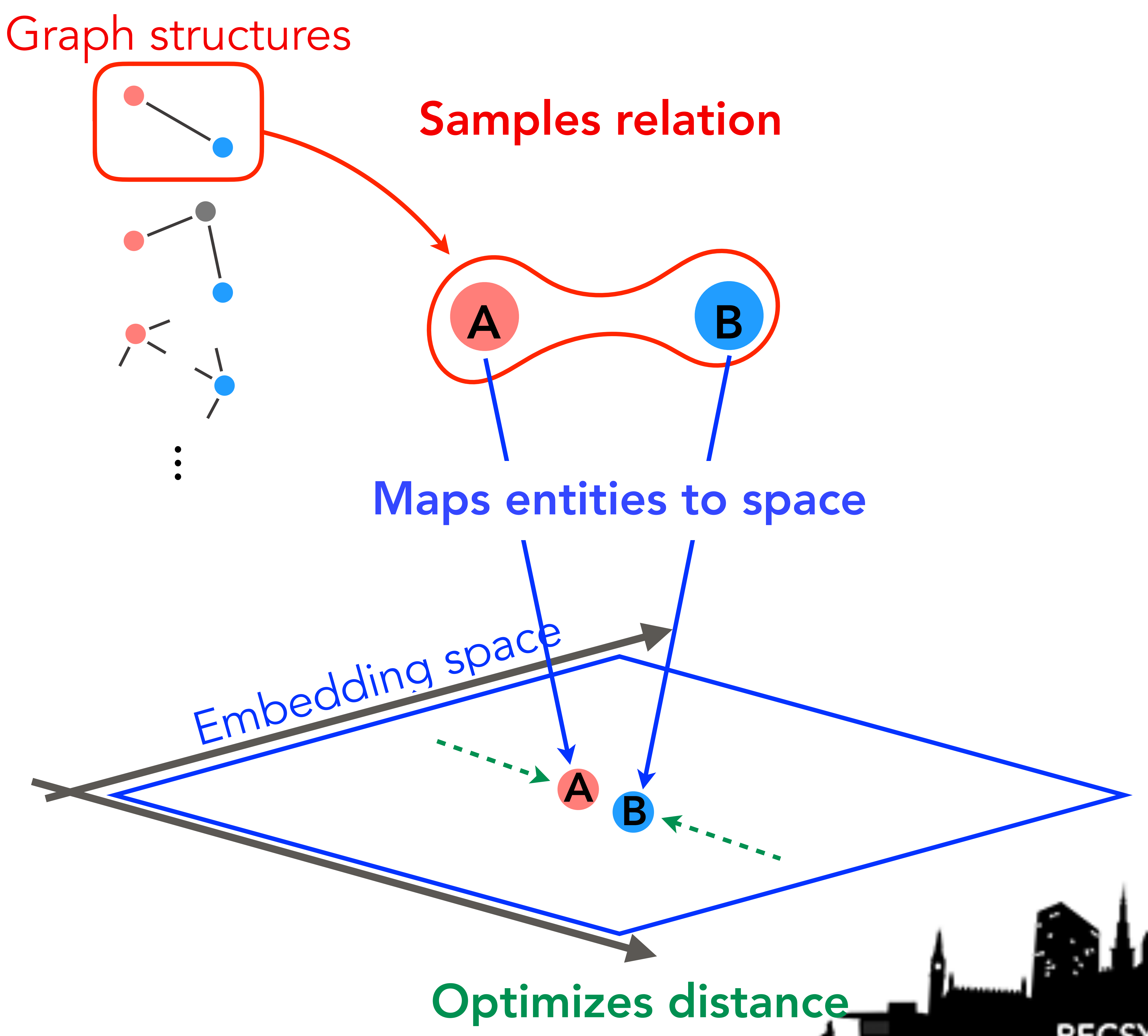
→ Wanna **customize GE methods** during each stage of training

→ **Difficult to compare models** as their implementations often vary

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

Solution : Modularize GE for adaptability !



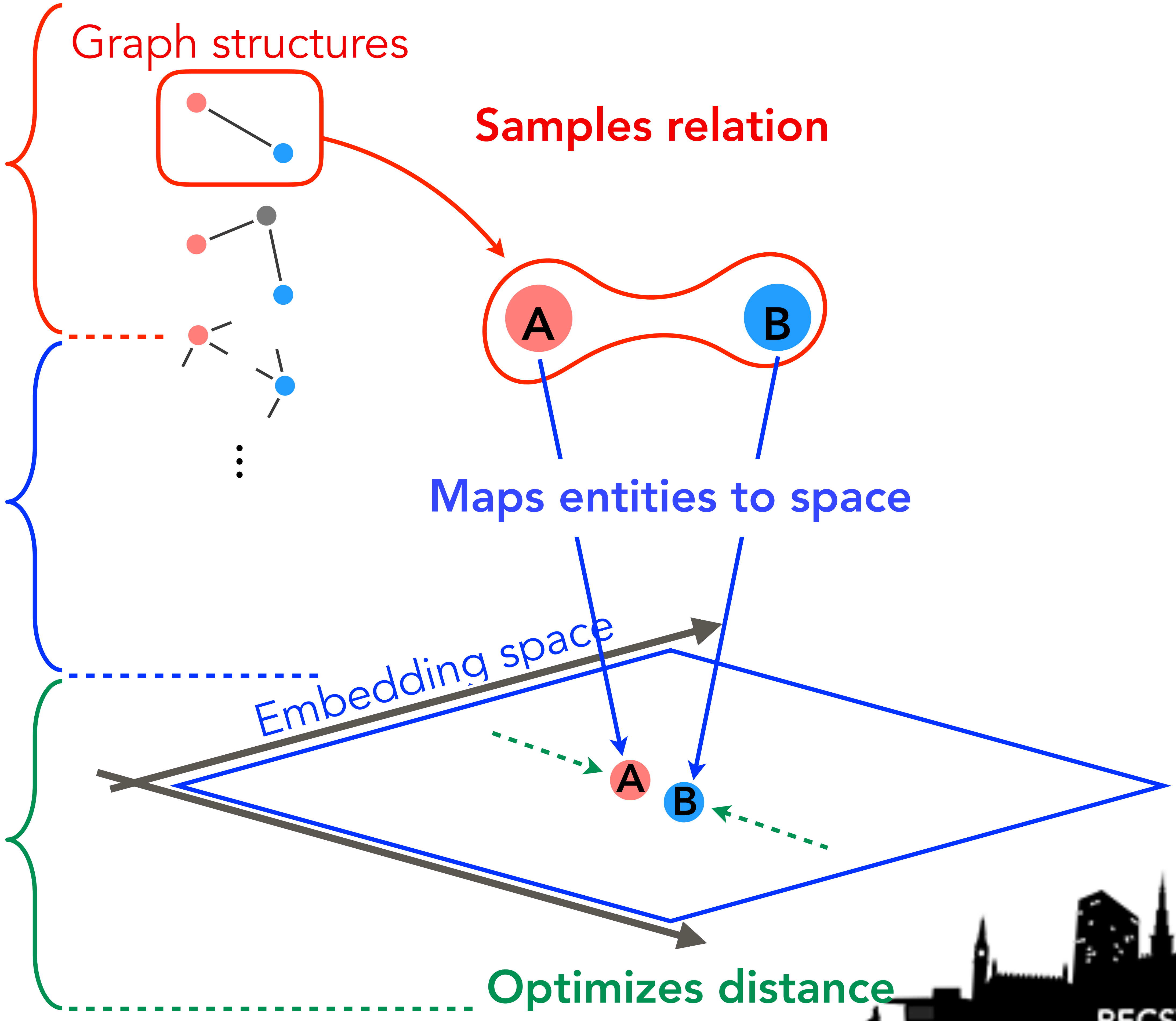
Ting-Hsiang Wang, Texas A&M University

Solution : Modularize GE for adaptability !

Sampler

Mapper

Optimizer



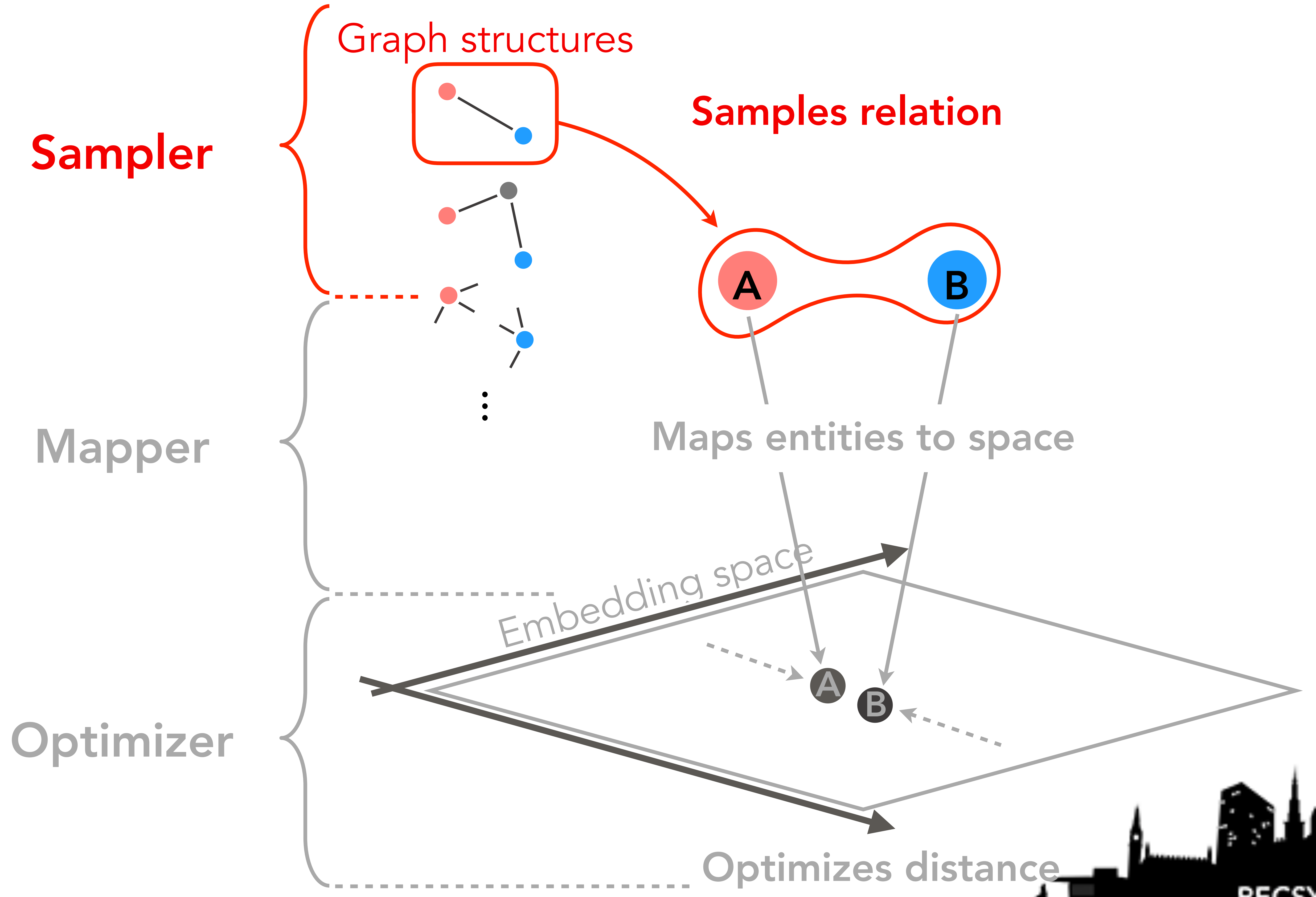
Solution : Modularize GE for adaptability !

- Extracts **graph structures** from dataset while remains **type-agnostic** to sampled entities, i.e., nodes & edges

$$G = (V, E, R, \phi)$$

$$E : \{(v_1, v_2) | (v_1, v_2) \in V \times V\}$$

$$\phi : E \rightarrow R$$

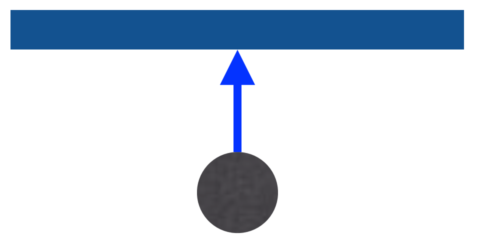


Solution : Modularize GE for adaptability !

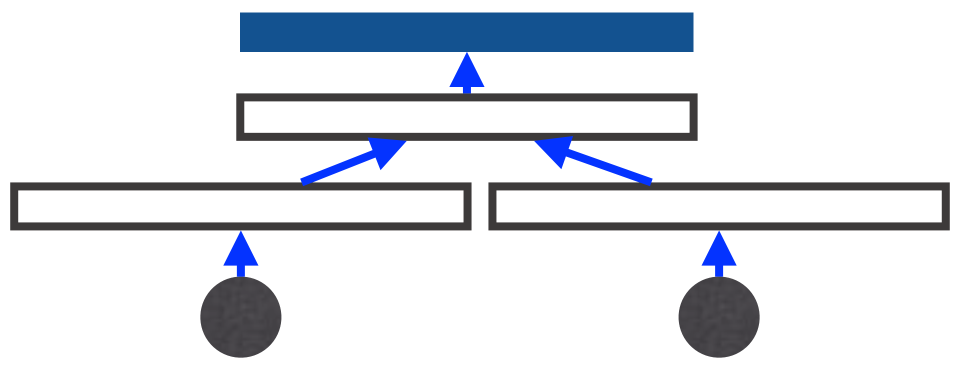
Ting-Hsiang Wang, Texas A&M University

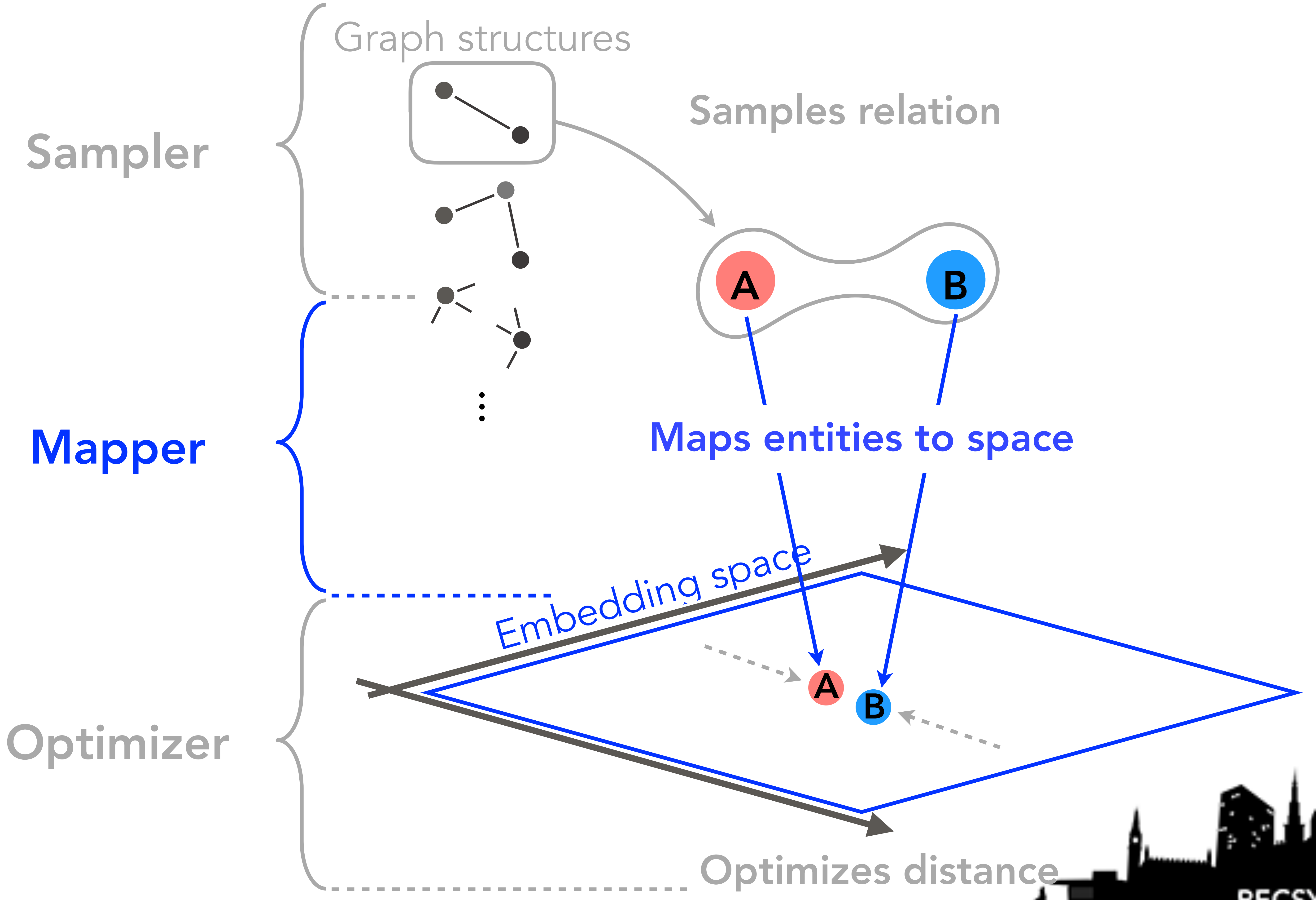
- Converts entities into **spatial features** via **embedding stacking operations**, e.g, lookup, pooling (average, etc.)

$f : (p) \rightarrow \mathbb{R}^d$



$f : (p, q) \rightarrow \mathbb{R}^d$



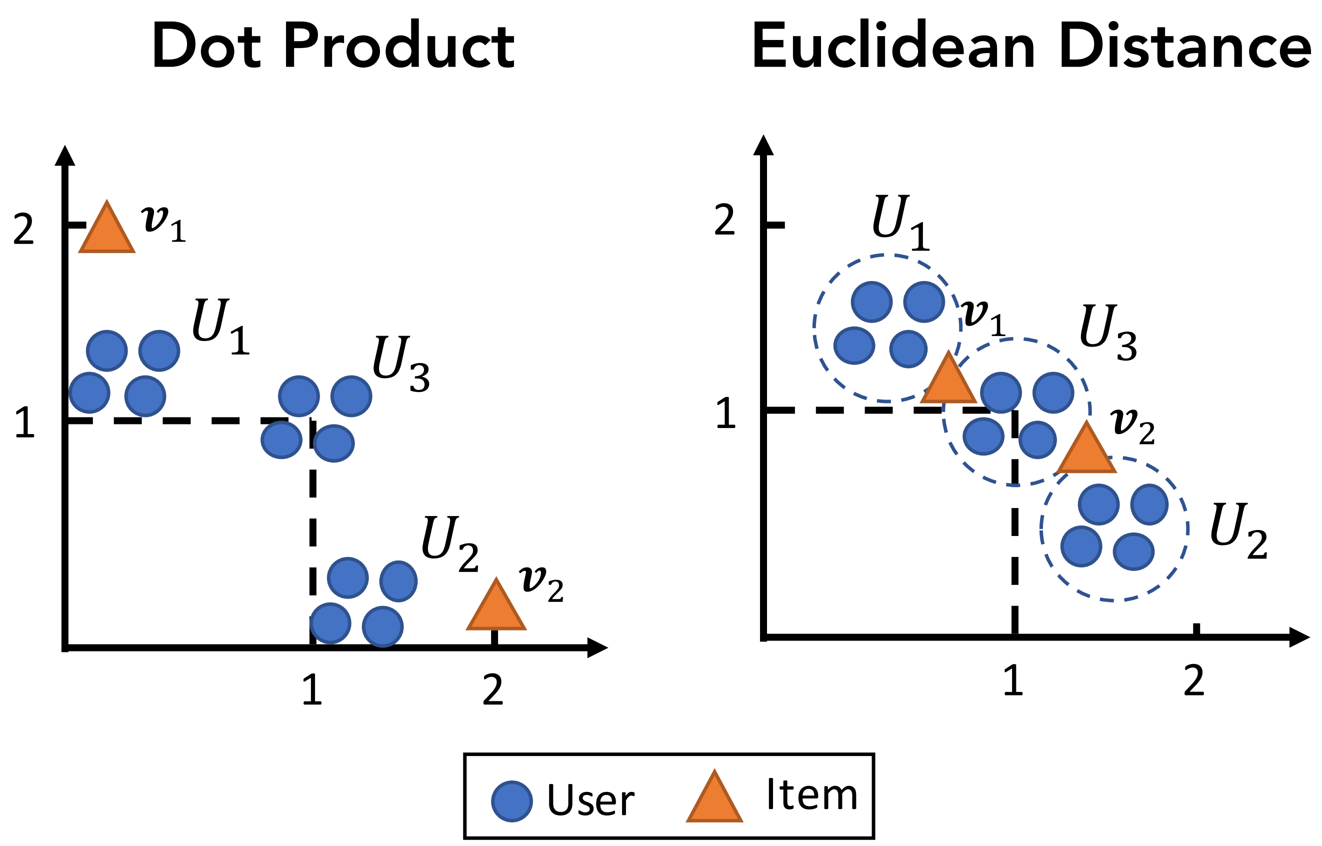


RECSYS 2019

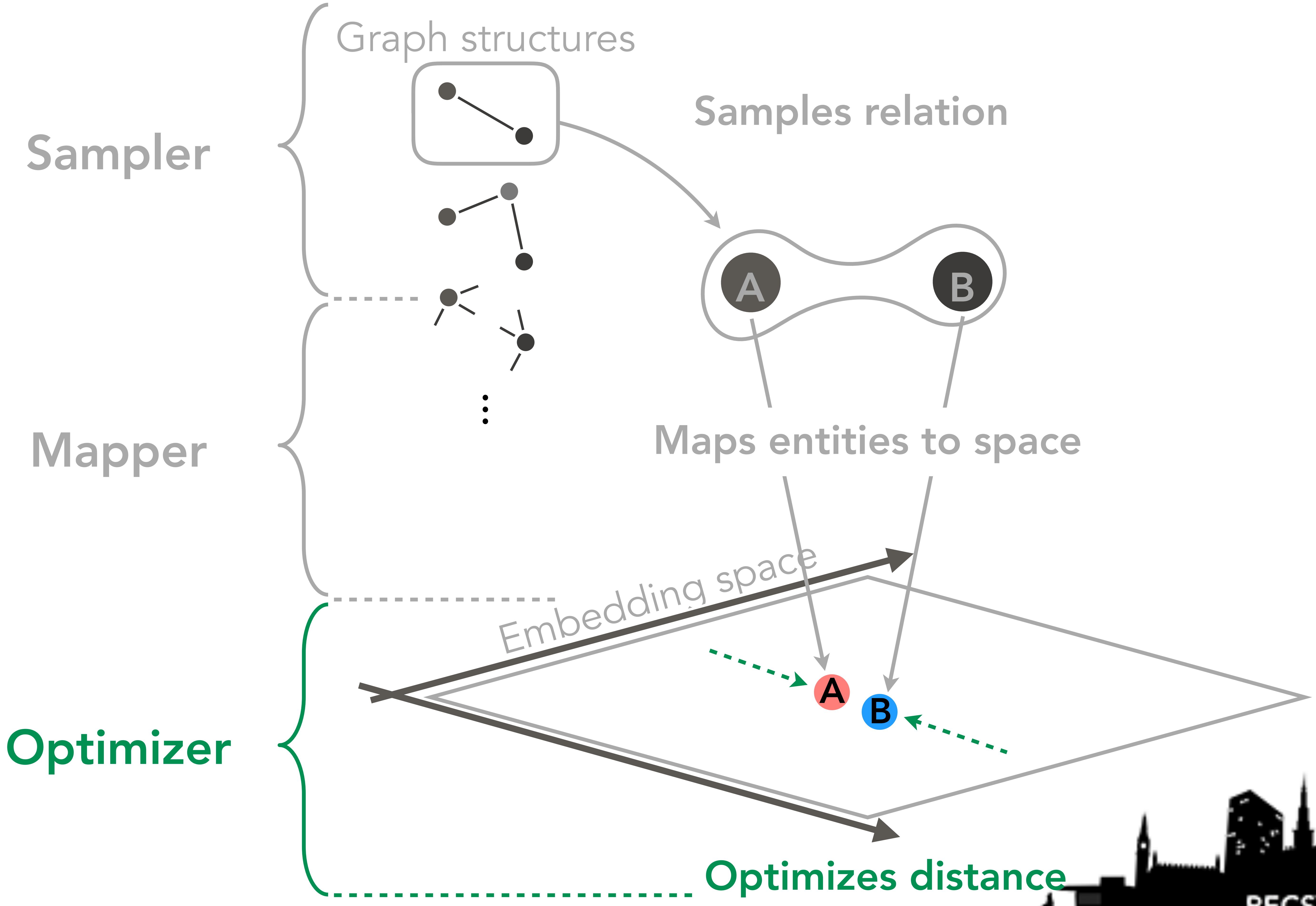
Solution : Modularize GE for adaptability !

- Preserves entity relatedness as **spatial properties** with customizable **similarity metrics** and **loss functions**

	v_1	v_2
U_1	✓	
U_2		✓
U_3	✓	✓



→ Euclidean distance keeps **triangular inequality**; dot product does not [5]



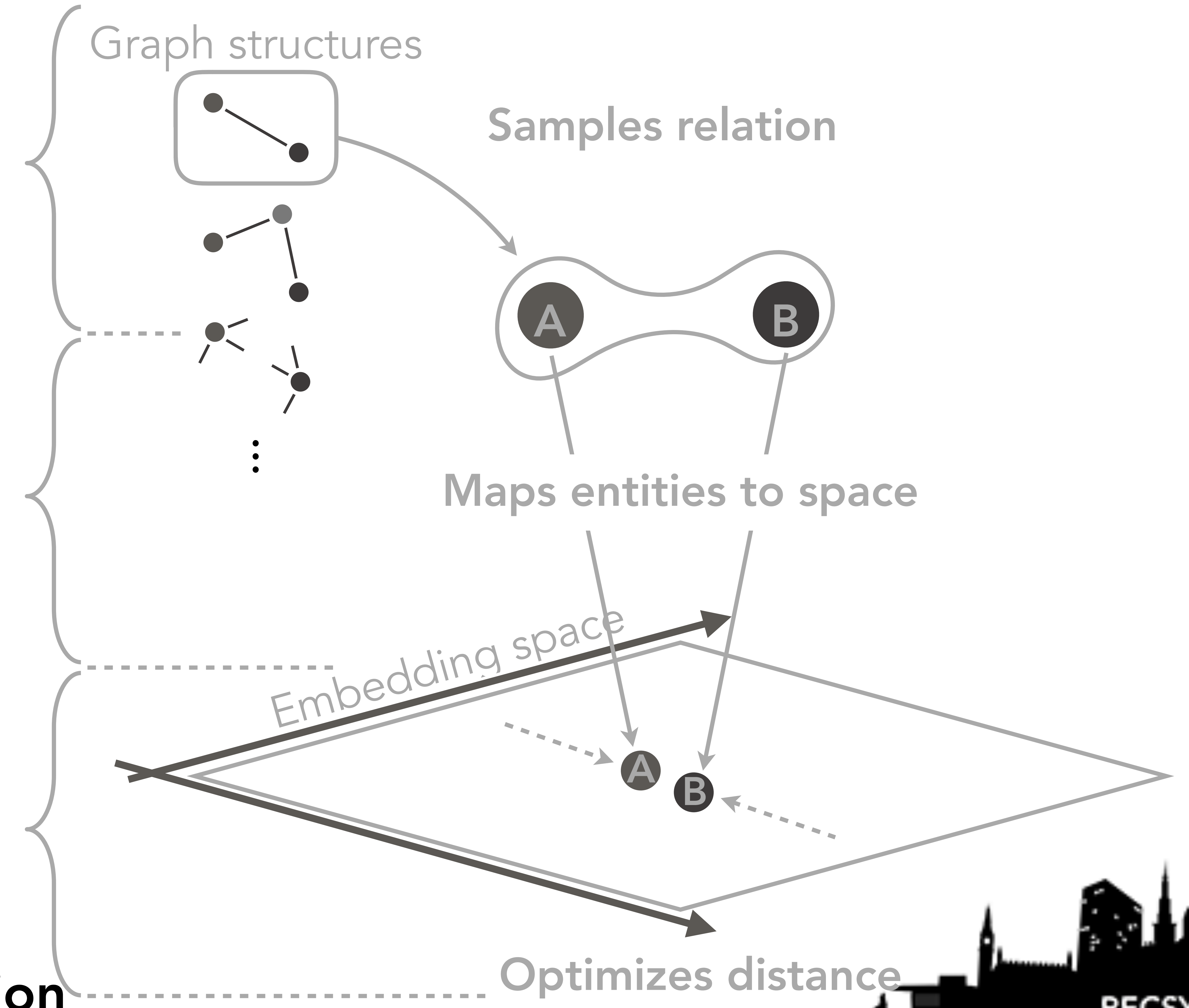
Ting-Hsiang Wang, Texas A&M University

Solution : Modularize GE for adaptability !

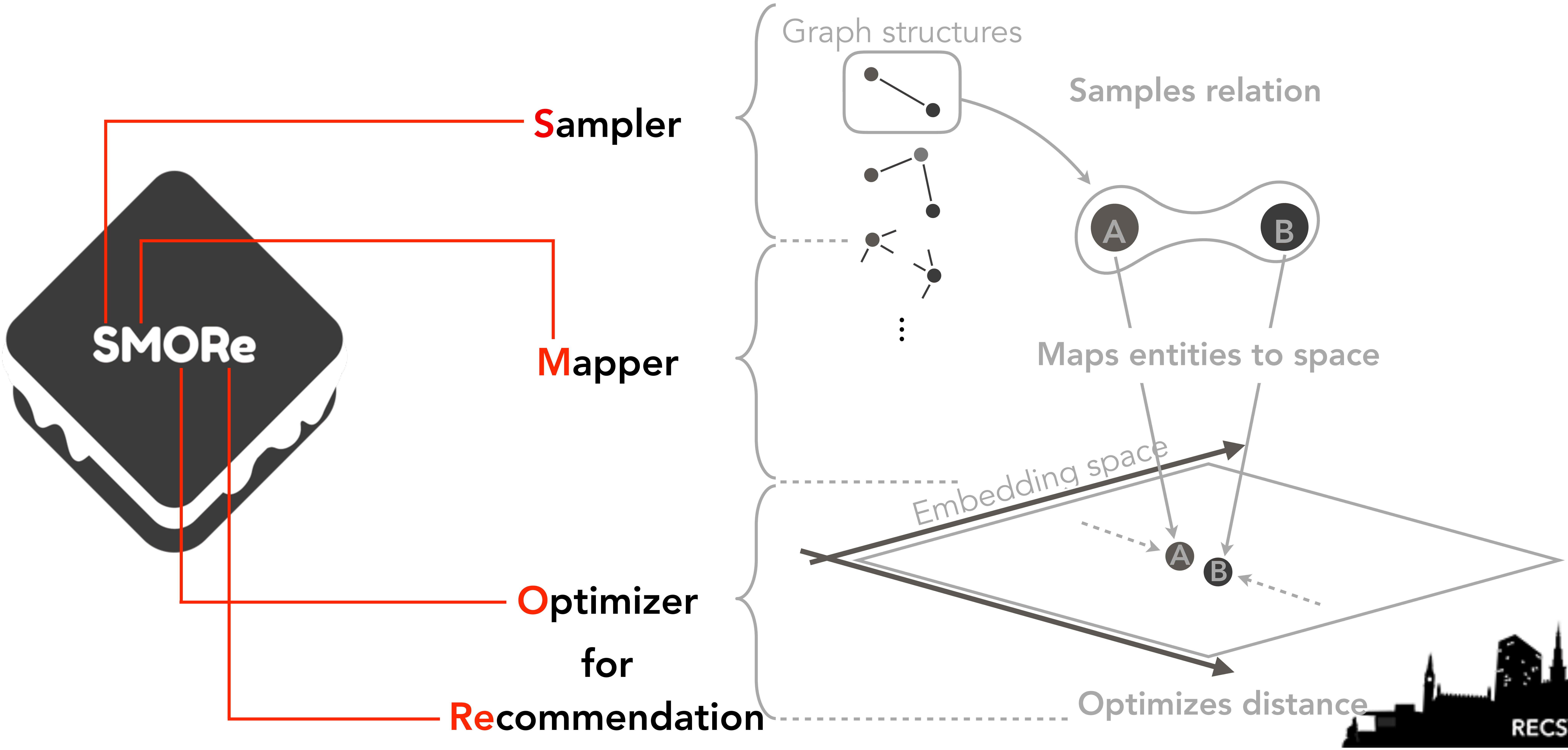
Sampler

Mapper

Optimizer for Recommendation



SMORe : Modular GE toolkit for REC



SMORe : Modular GE toolkit for REC

Ting-Hsiang Wang, Texas A&M University

SMORe is a modular GE toolkit for REC

S'more is a campfire treat with layers

RECSYS 2019

Benefits of SMORe

Ting-Hsiang Wang, Texas A&M University

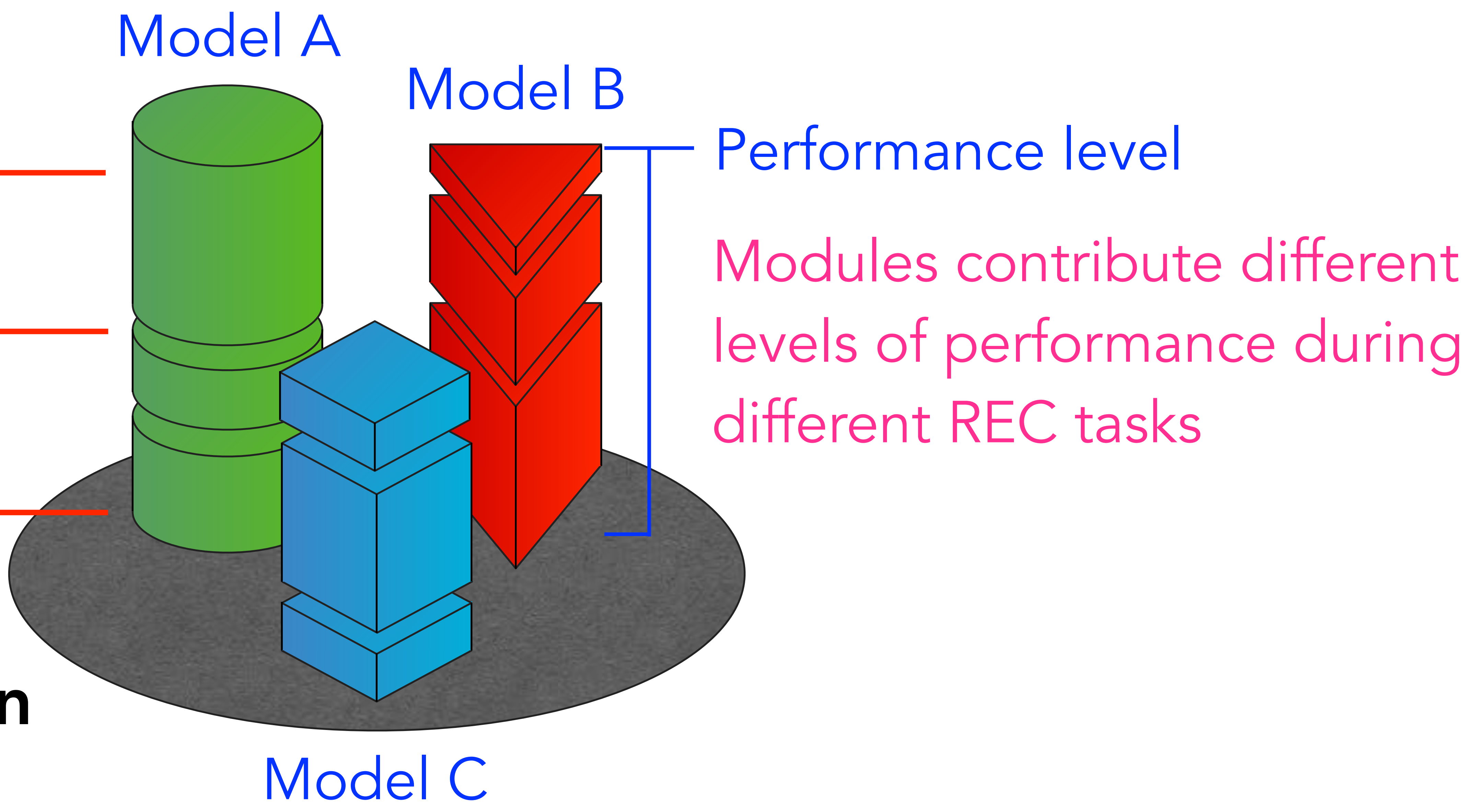
Sampler

Mapper

Optimizer

for

Recommendation



RECSYS 2019

Benefits of SMORe

Ting-Hsiang Wang, Texas A&M University

SMORe

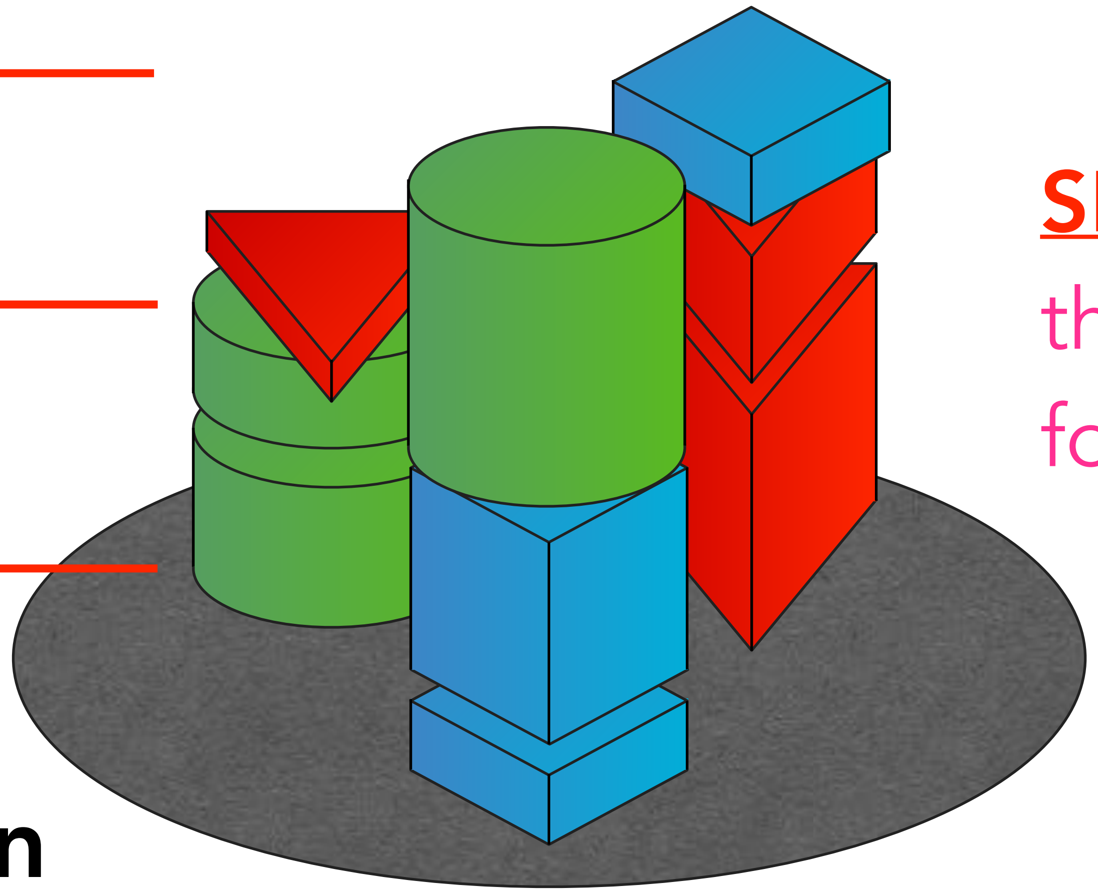
Sampler

Mapper

Optimizer

for

Recommendation



SMORe enables combining the most suitable modules for given REC task

Benefits of SMORe

Ting-Hsiang Wang, Texas A&M University

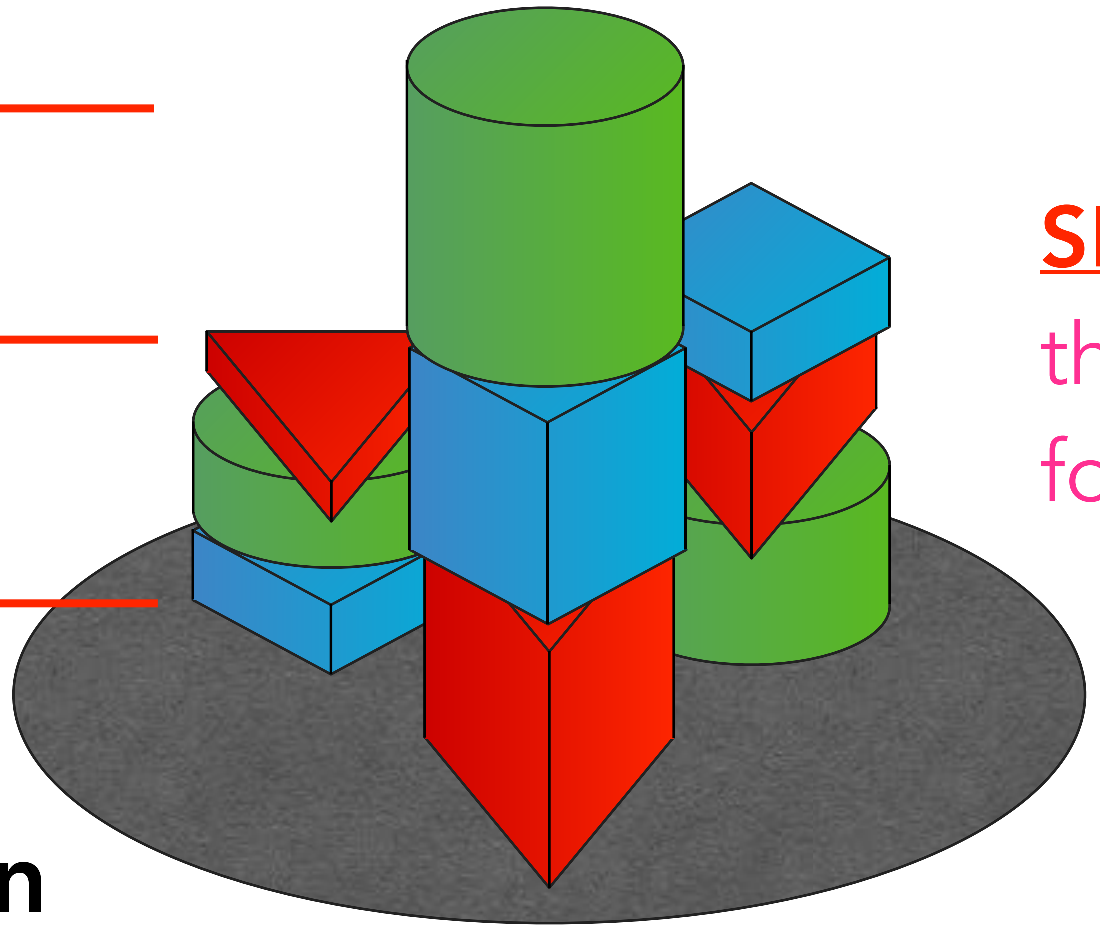
Sampler

Mapper

Optimizer

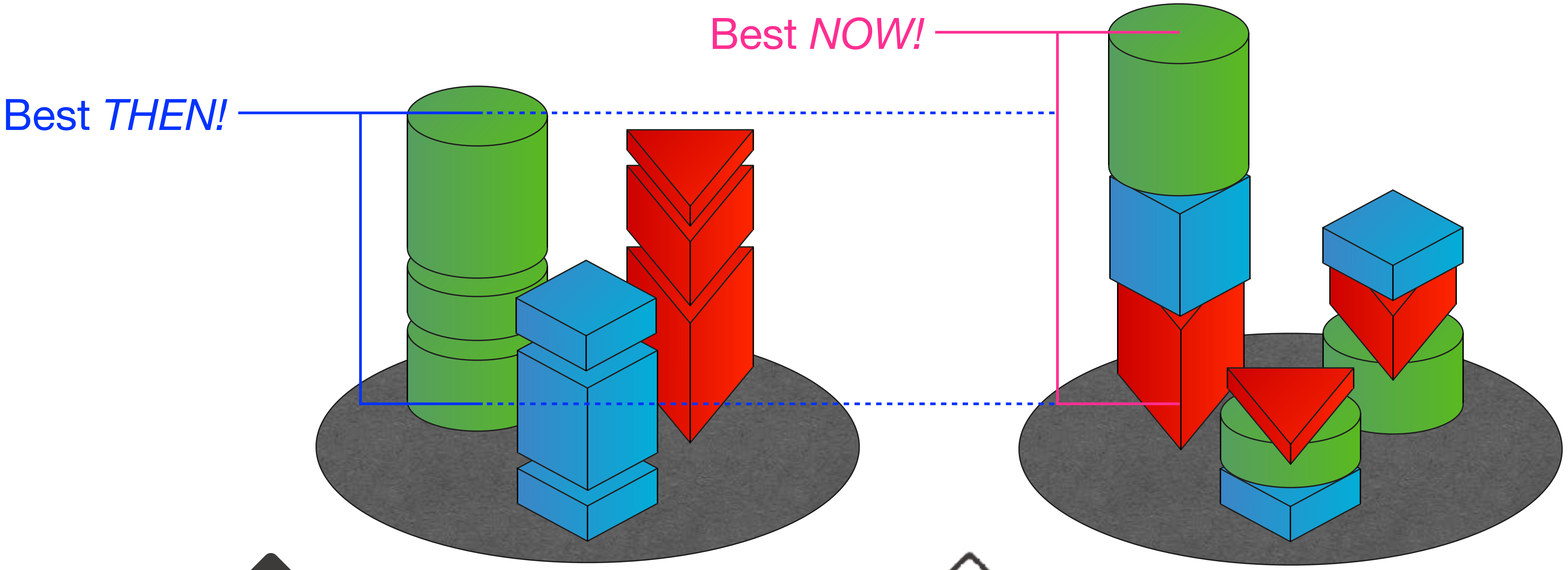
for

Recommendation



SMORe enables combining the most suitable modules for given REC task

Benefits of SMORe



As toolkit for research:

1. Baseline comparison
2. Ballpark approaches
3. One module at a time

As framework for development:

1. Reduces development time
2. Raises performance limit
3. Adapts to new tasks

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

References

1. Hamilton, William L., et al. “Representation on Networks”, WWW 2018 Tutorial
2. Shi, Chuan, et al. “Relevance Search in Heterogeneous Networks”, ACM EDBT 2012
3. Shi, Chuan, et al. “A Survey of Heterogeneous Information Network Analysis”, IEEE TKDE 2016
4. Bernhardsson, Erik “ANNOY library”, https://github.com/spotify/annoy
5. Hsieh, Cheng-Kang, et al. “Collaborative Metric Learning”, WWW 2017

Lecture Agenda

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

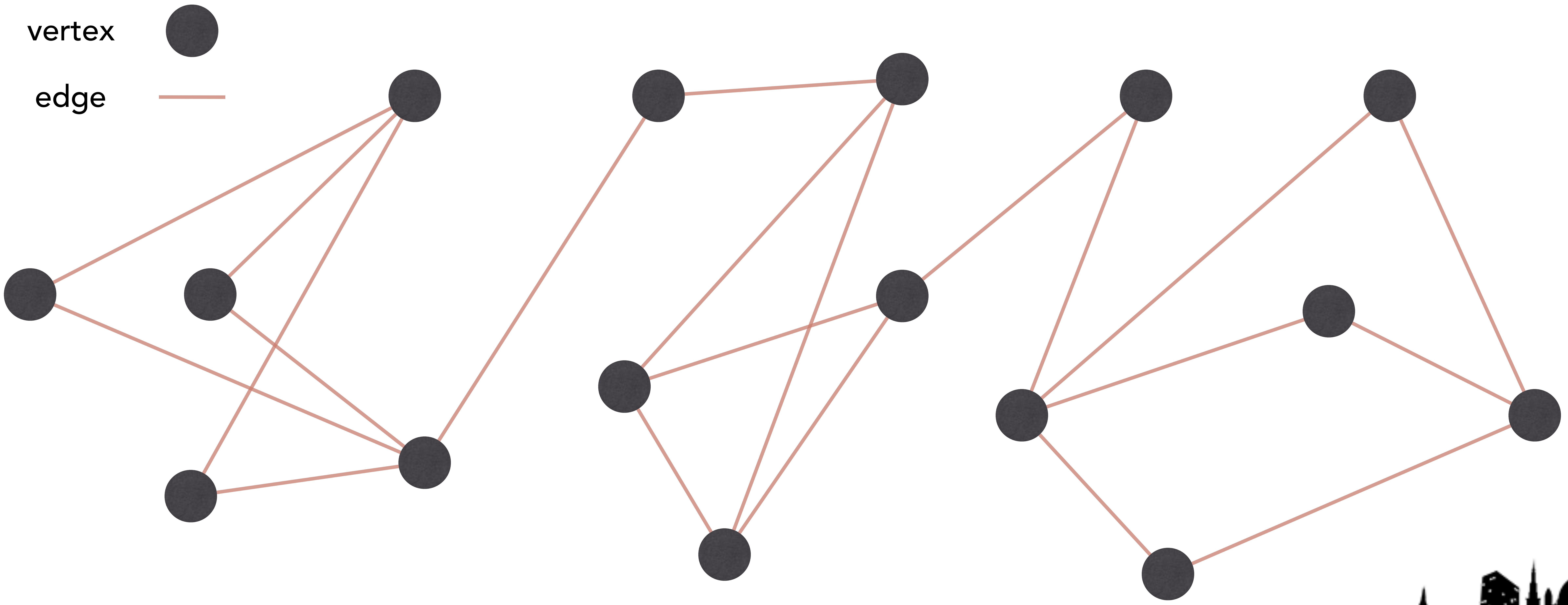
(Sean) **Q2. SMORe** modularization of GE and benefits

(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

RECSYS 2019

Example Graph

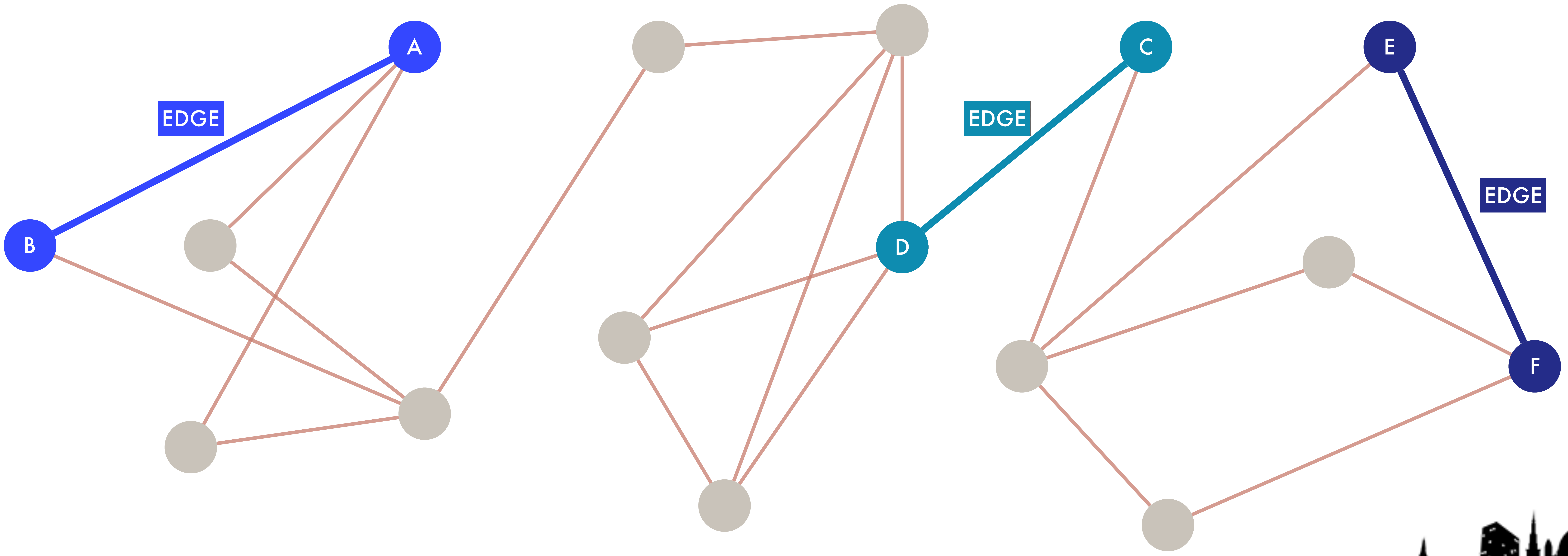


Vertex Structure

- Adjacency
  - vertices which share the same edge
- Neighborhood
  - vertices which share similar connections
- Community
  - vertices which share similar communities
- Centrality
  - vertices which share similar properties

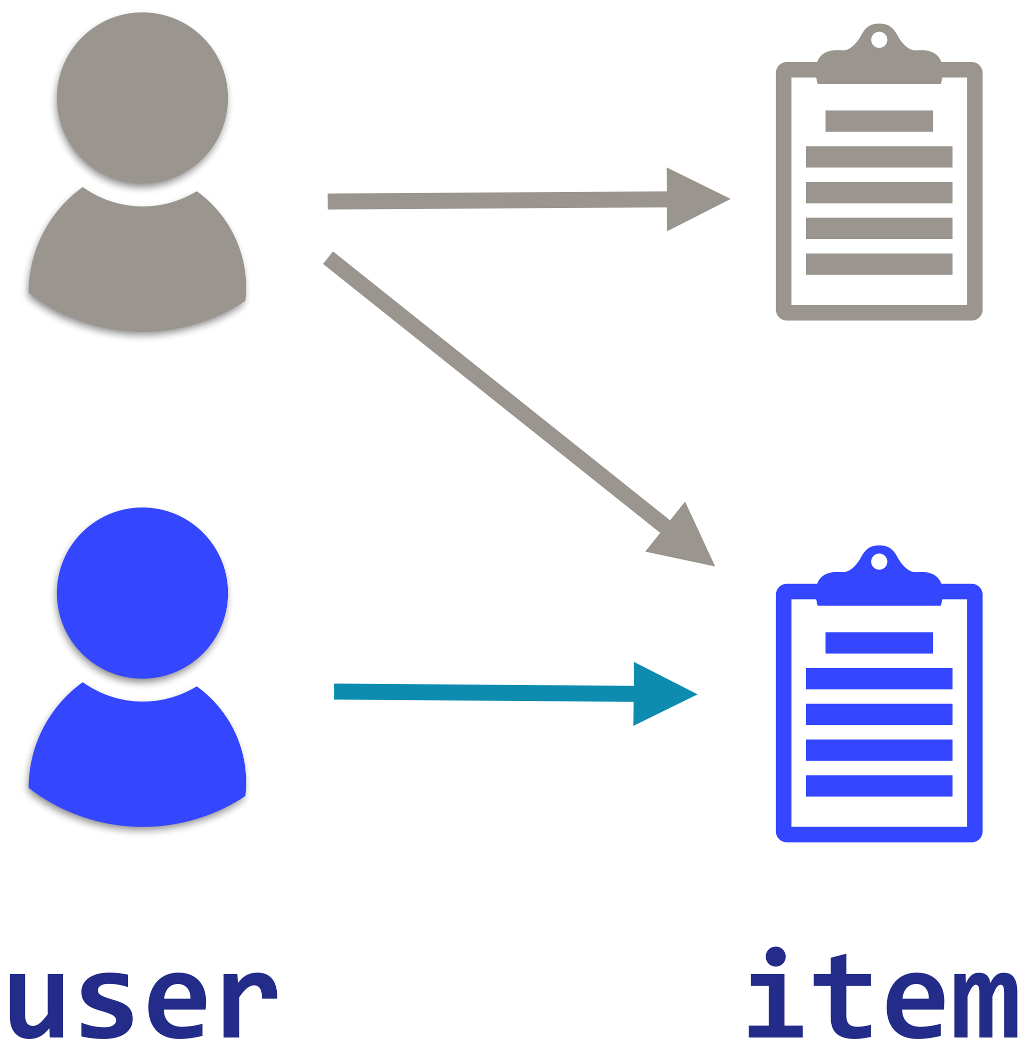
RECSYS 2019

1. Adjacency



Adjacency

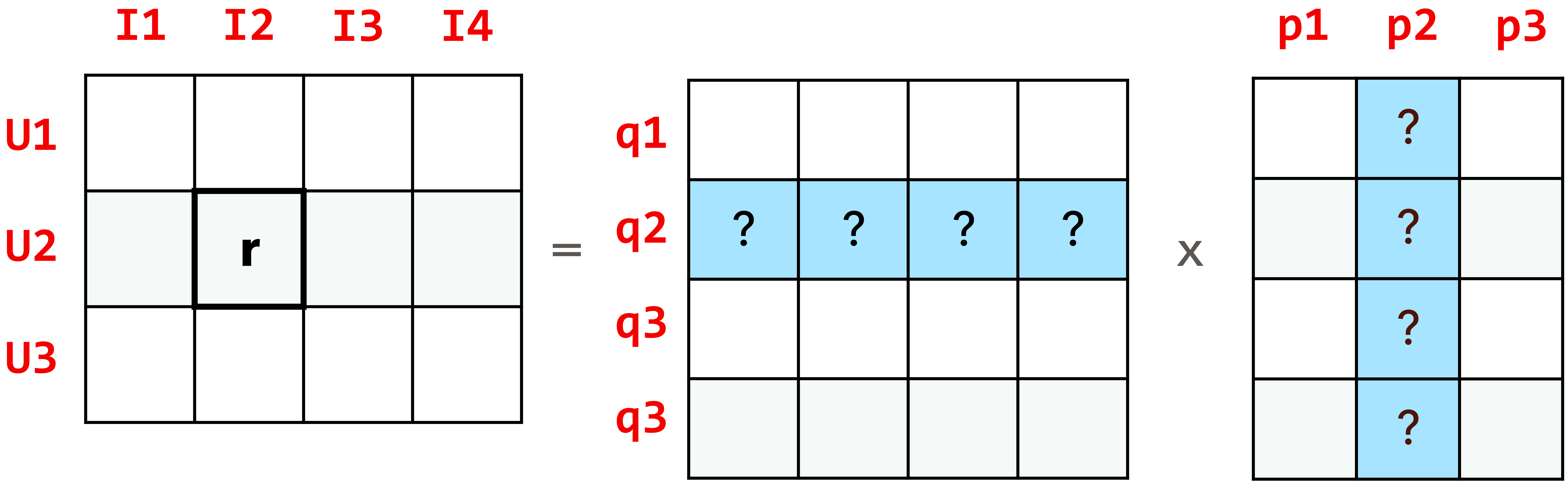
vertices share the same edge are treated as similar pairs



user-item preference

Matrix Factorization

IEEE Computer'09



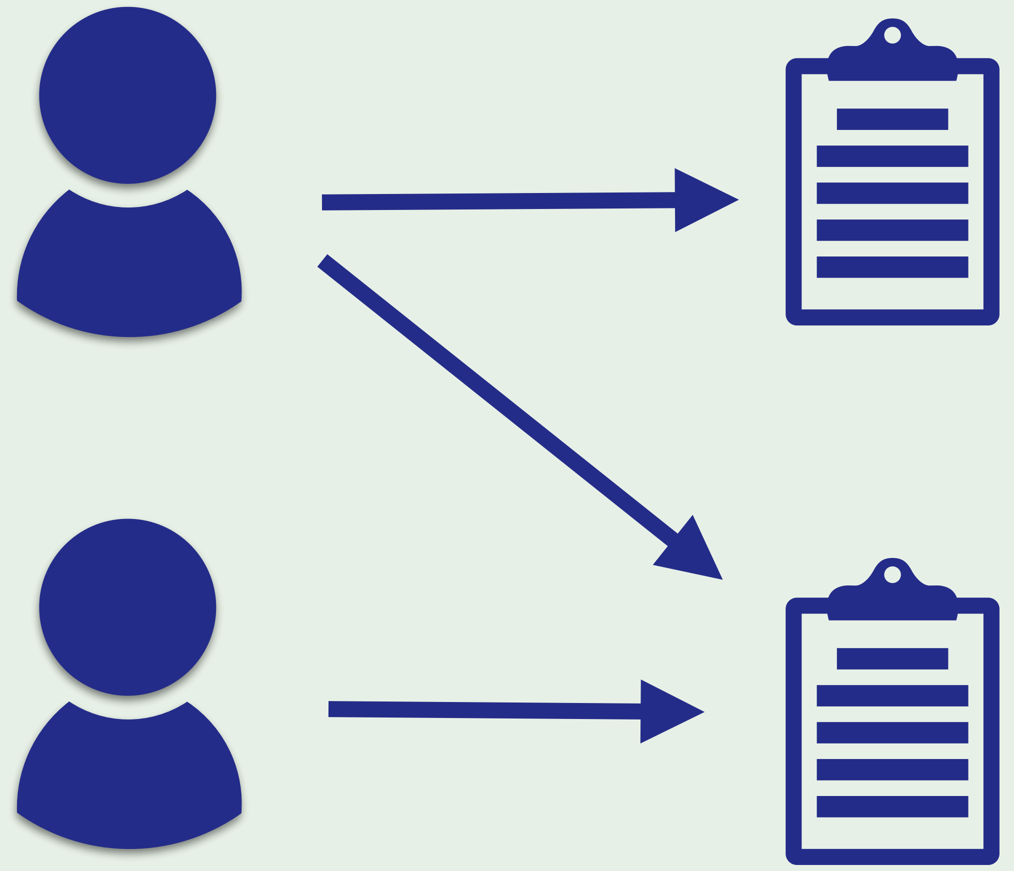
$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2$$

(rating - item*user)

Yehuda Koren, Robert M. Bell, Chris Volinsky: Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8): 30-37 (2009)

Matrix Factorization in SMORe

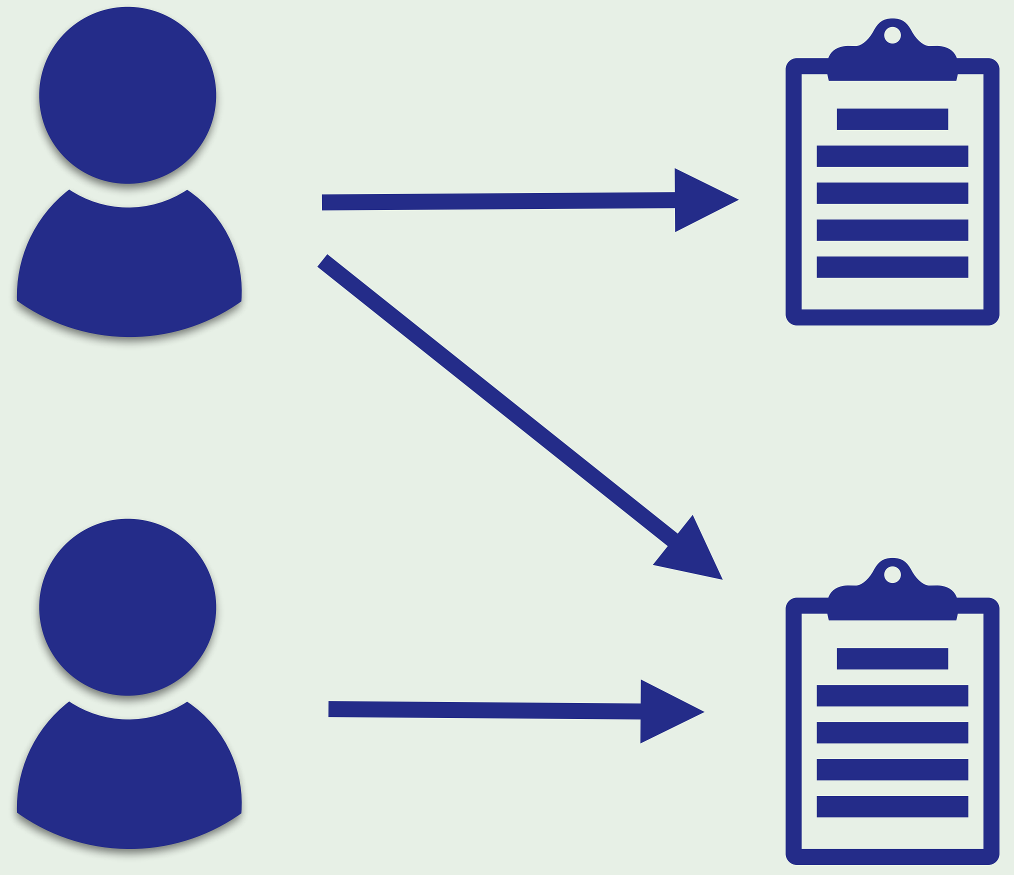
Graph



user-item graph

Matrix Factorization in SMORe

Graph

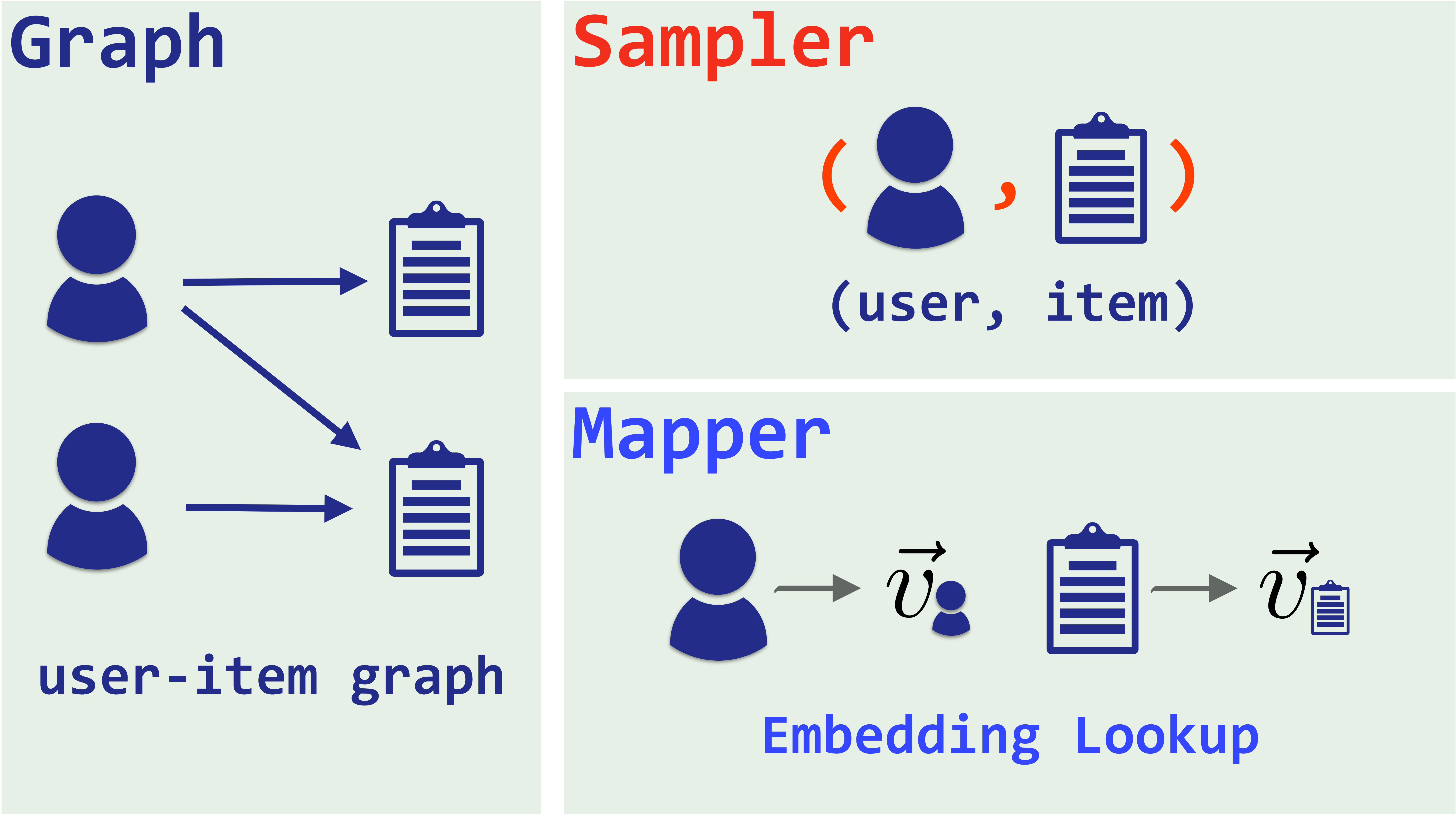


user-item graph

Sampler

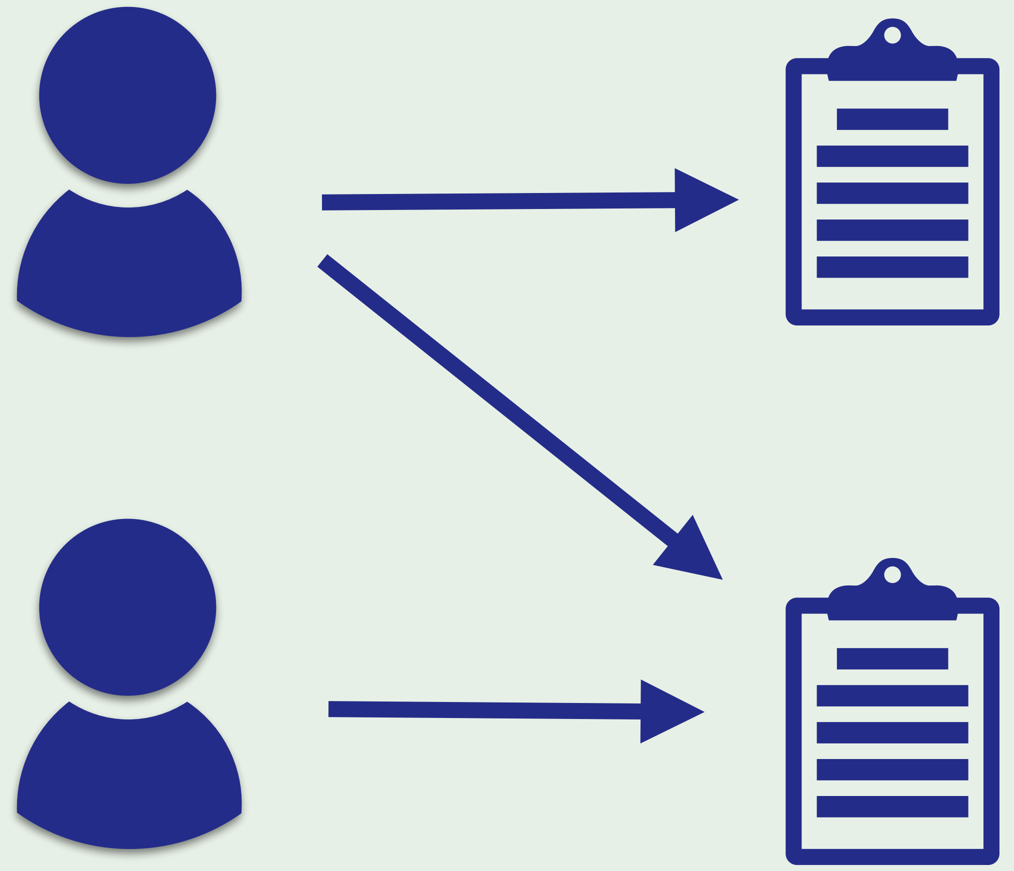
(user, item)

Matrix Factorization in SMORe



Matrix Factorization in SMORe

Graph



user-item graph

Sampler

(user, item)

Mapper

Embedding Lookup

Optimizer

$$(\vec{v} \cdot \vec{v} - y)^2$$

Square Error

RECSYS 2019

Sparse LInear Method (SLIM) *ICDM'11, RecSys'16*

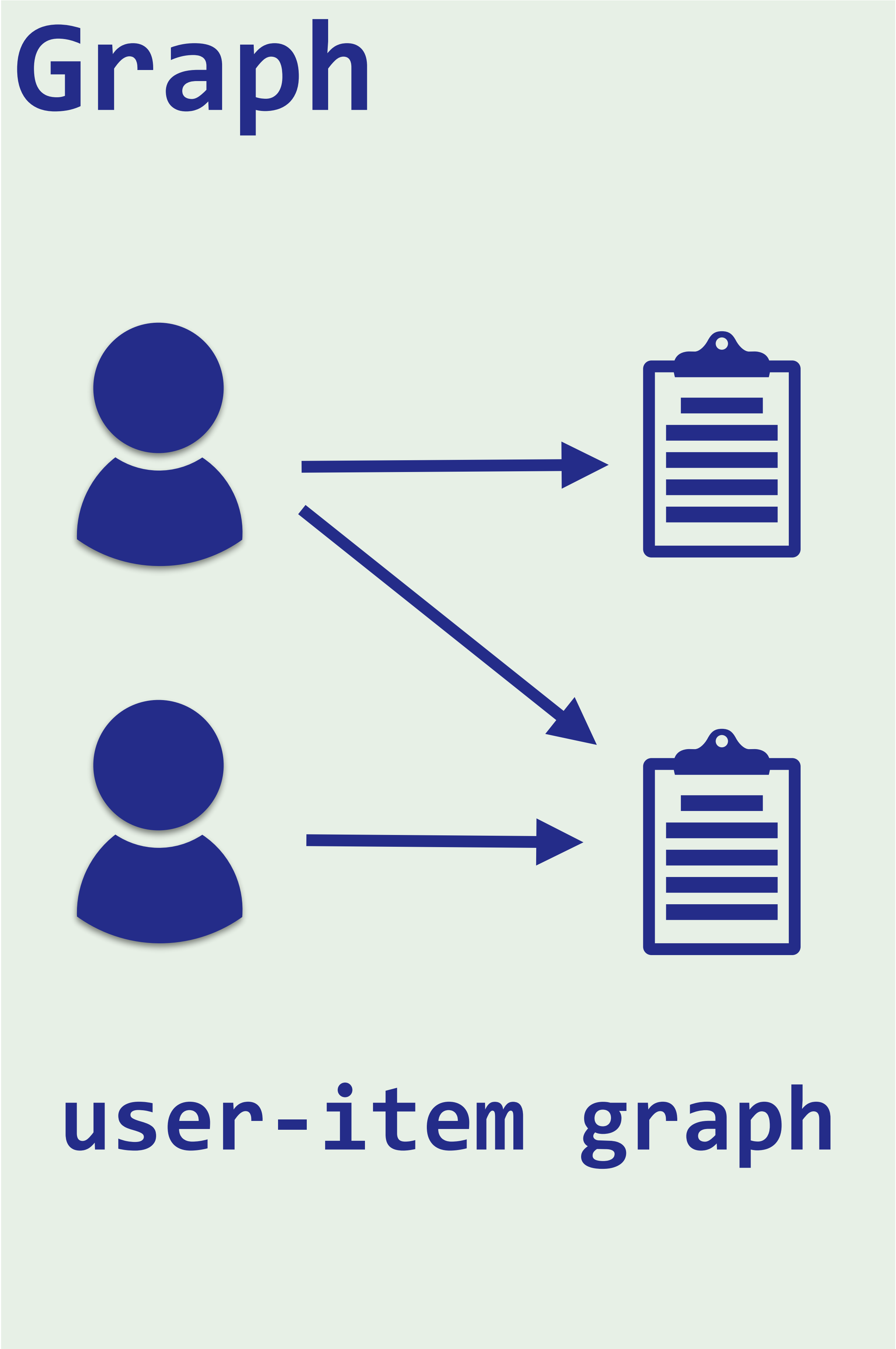
$$
\begin{array}{c|cccc|}
 & \text{I1} & \text{I2} & \text{I3} & \text{I4} \\
\hline
\text{U1} & & & & \\
\text{U2} & 1 & - & 1 & - \\
\text{U3} & & & & \\
\hline
\end{array}
=
\begin{array}{c|cccc|}
 & \text{I1} & \text{I2} & \text{I3} & \text{I4} \\
\hline
\text{U1} & & & & \\
\text{U2} & 1 & 0 & 1 & 0 \\
\text{U3} & & & & \\
\hline
\end{array}
\times
\begin{array}{c|cccc|}
 & \text{I1} & \text{I2} & \text{I3} & \text{I4} \\
\hline
\text{I1} & 0 & ? & & \\
\text{I2} & & 0 & & \\
\text{I3} & & ? & 0 & \\
\text{I4} & & - & & 0 \\
\hline
\end{array}
$$

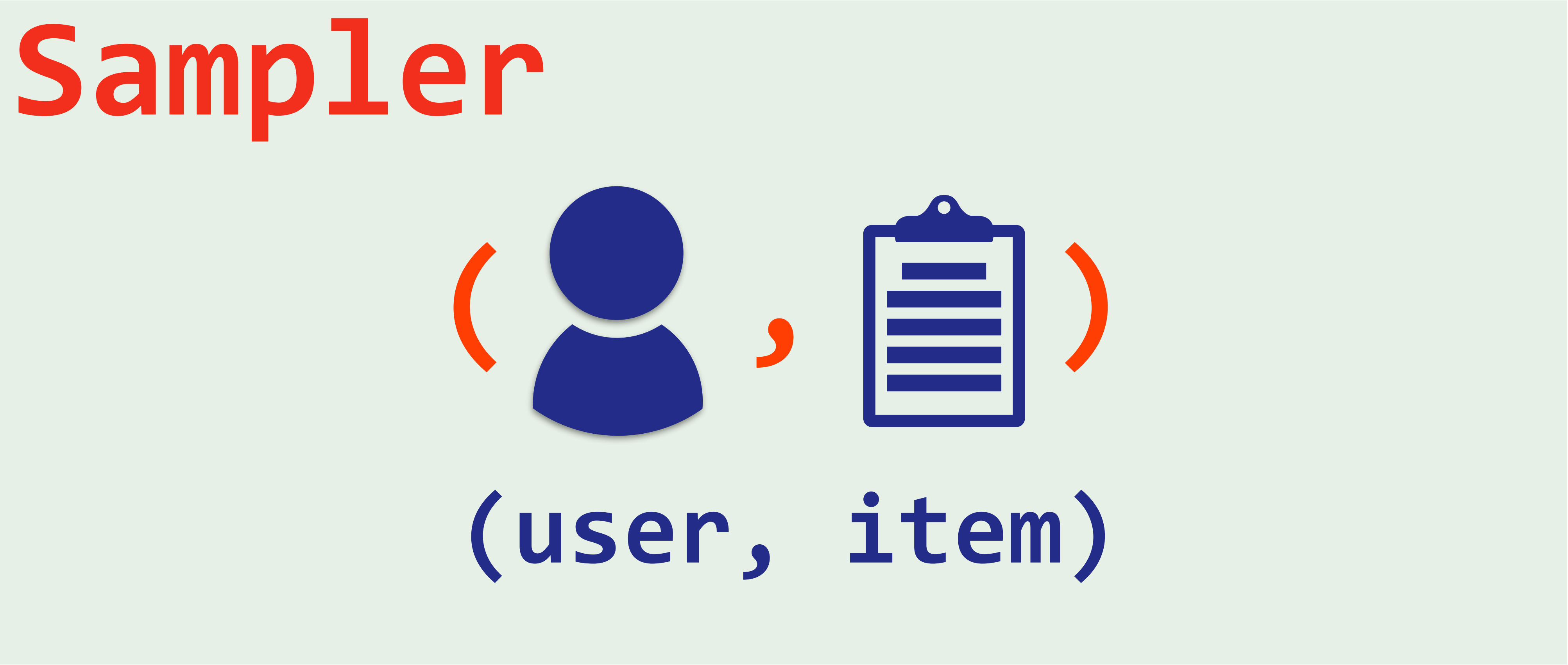
Transition Matrix W

$$
\begin{aligned}
\underset{W}{\text{minimize}} \quad & \frac{1}{2}\|A - AW\|_F^2 \\
\text{subject to} \quad & W \geq 0 \\
& \operatorname{diag}(W) = 0,
\end{aligned}
$$

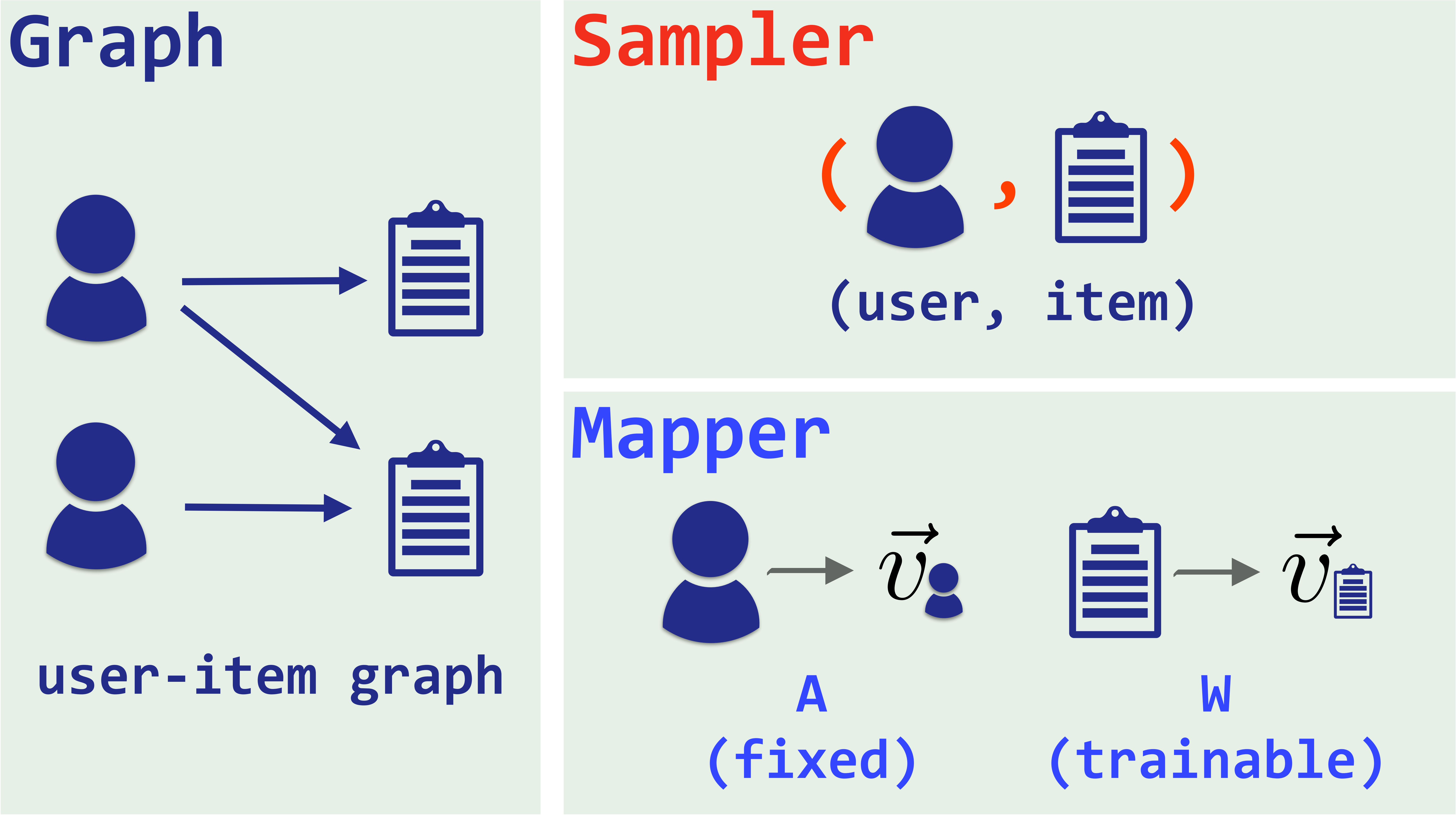
1. *Xia Ning, George Karypis: SLIM: Sparse Linear Methods for Top-N Recommender Systems. ICDM 2011: 497-506*
2. *Evangelia Christakopoulou, George Karypis: Local Item-Item Models For Top-N Recommendation. RecSys 2016: 67-74*

SLIM in SMORe





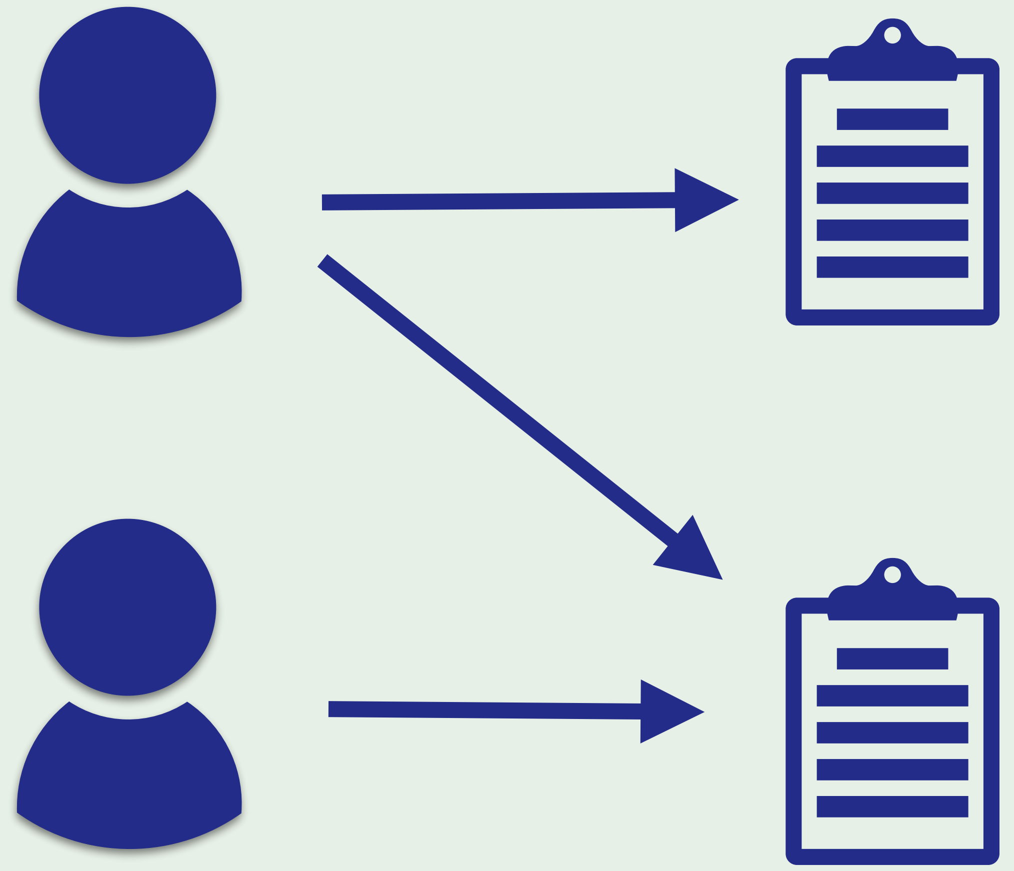
SLIM in SMORe



$$
\begin{aligned}
\underset{W}{\text{minimize}} \quad & \frac{1}{2}\|A - AW\|_F^2 \\
\text{subject to} \quad & W \geq 0 \\
& \text{diag}(W) = 0,
\end{aligned}
$$

SLIM in SMORe

Graph

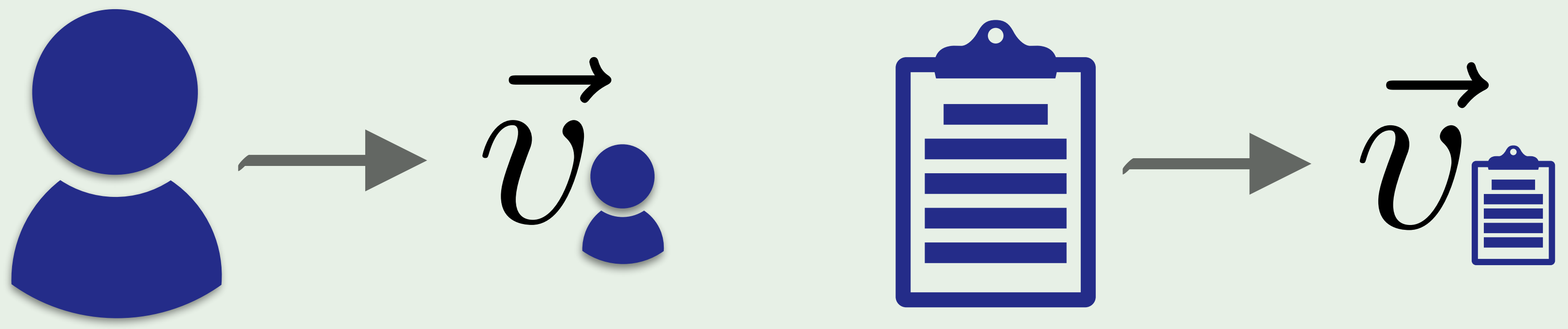


user-item graph

Sampler

(user, item)

Mapper



A
(fixed)

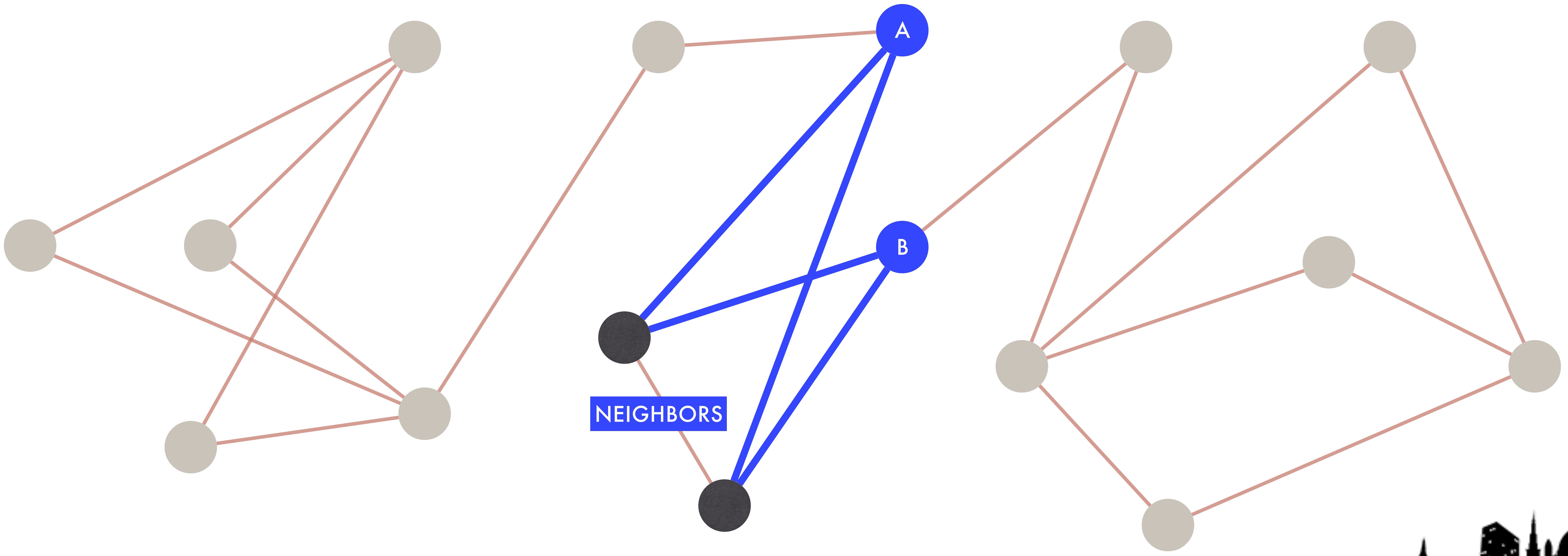
W
(trainable)

Optimizer

$$(\vec{v} \cdot \vec{v} - y)^2$$

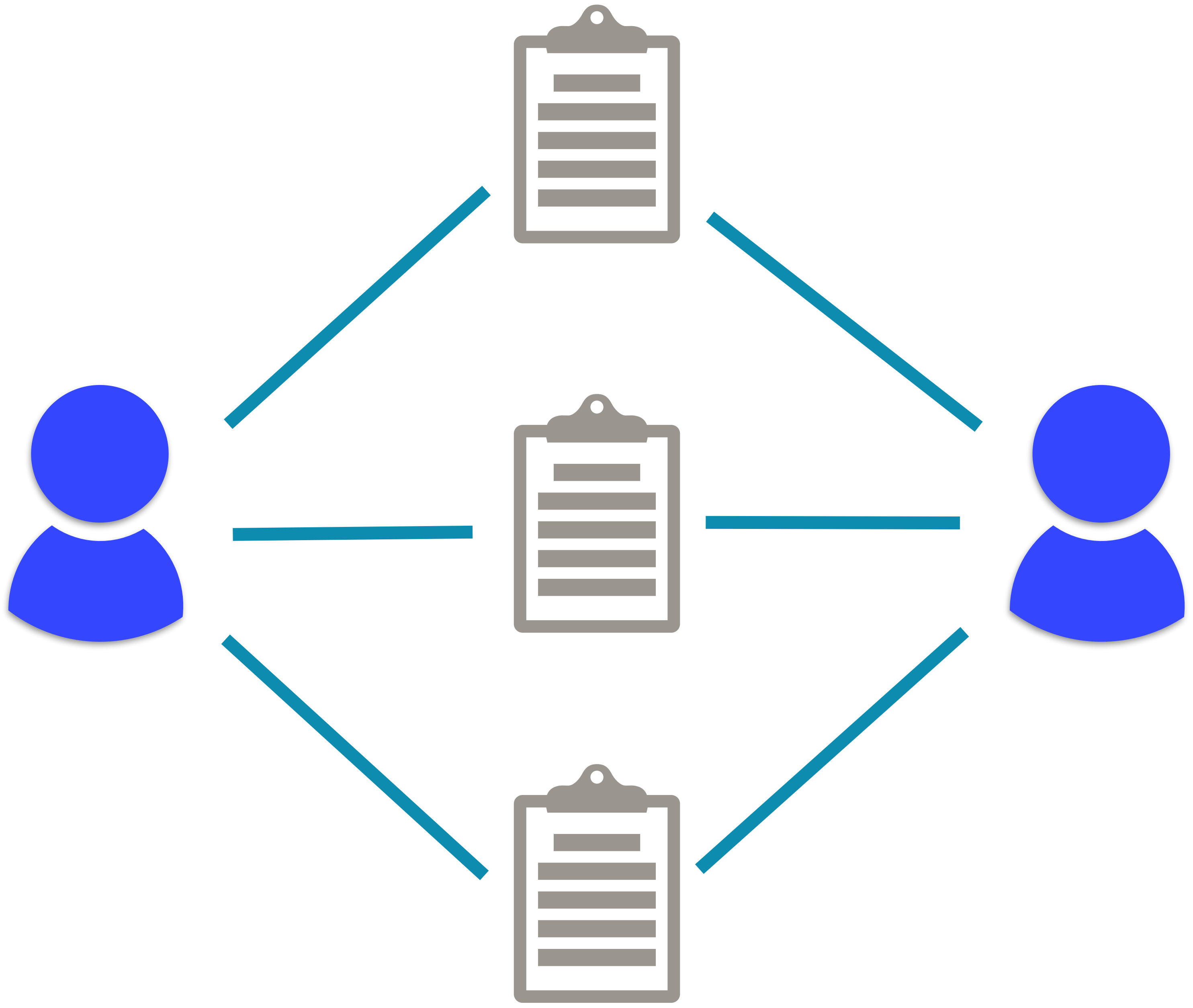
Square Error

2. Neighborhood

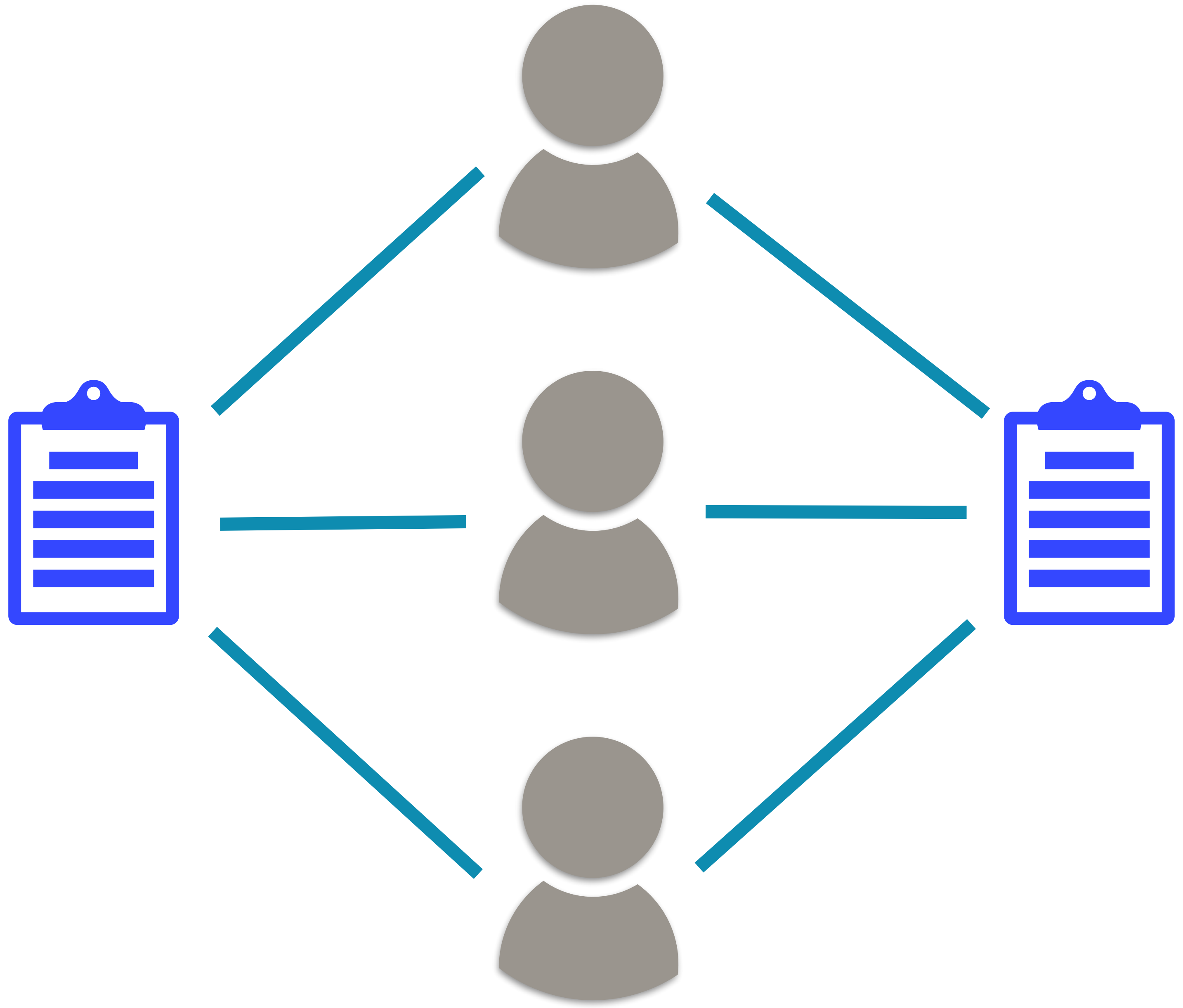


Neighborhood

vertices share similar connections are treated as similar pairs



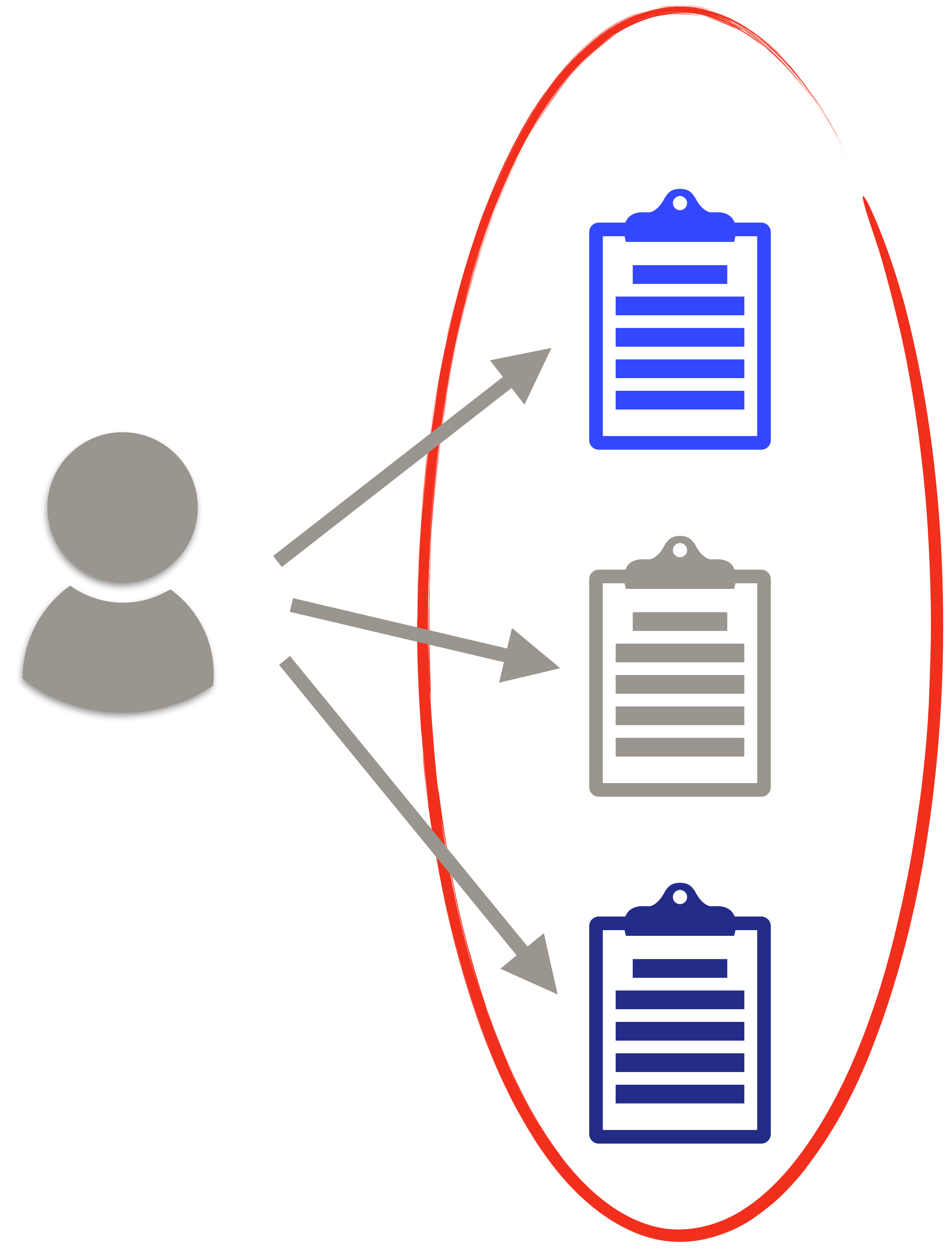
common interests



commonly purchased

item2vec

RecSys'16, MLSP'16



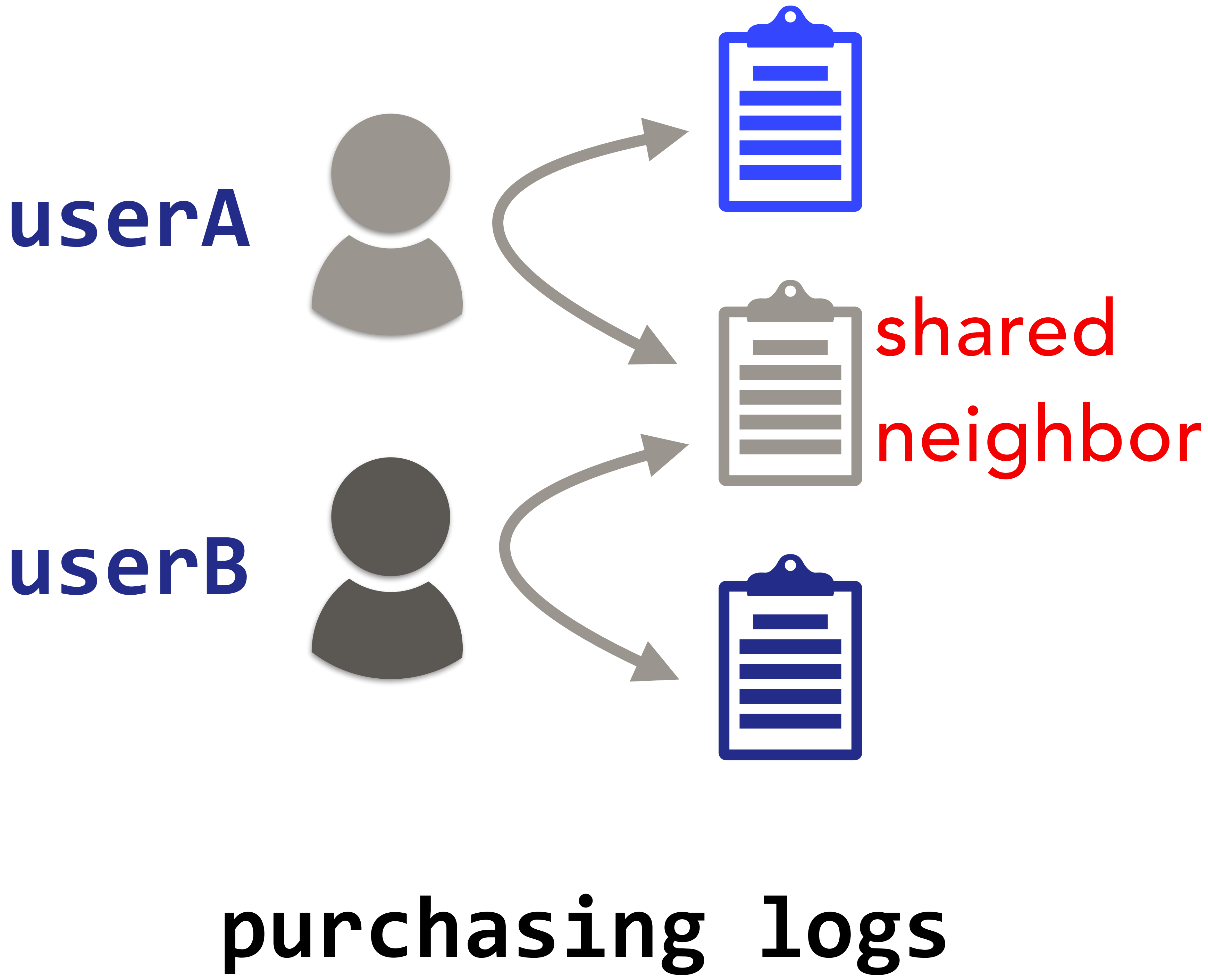
the given K items

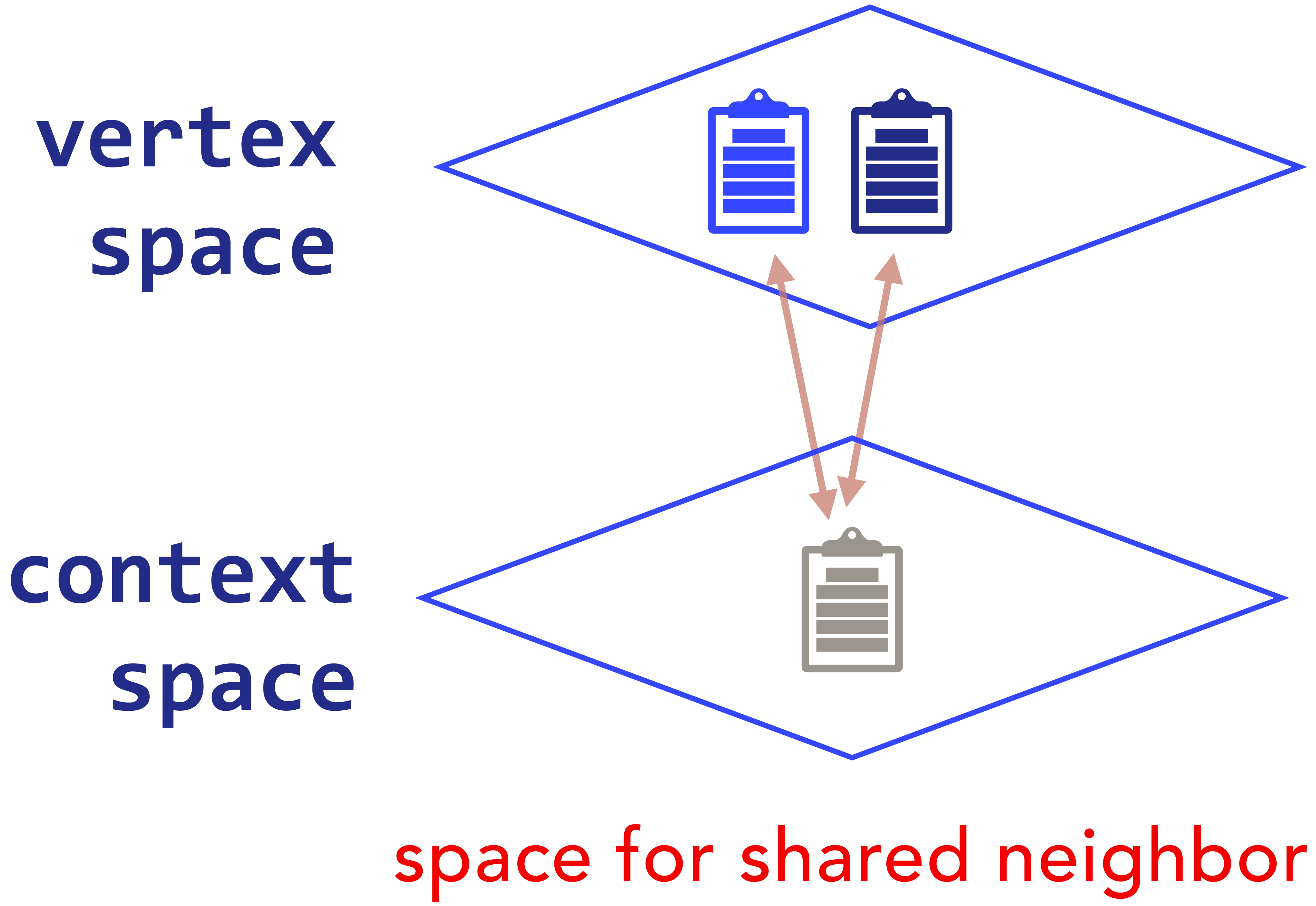
$$\frac{1}{K}\sum_{i=1}^{K}\sum_{j\neq i}^{K}\log p(w_j|w_i)$$

purchasing logs

Oren Barkan, Noam Koenigstein: Item2vec: Neural Item Embedding for Collaborative Filtering. RecSys Posters 2016
Oren Barkan, Noam Koenigstein: ITEM2VEC: Neural item embedding for collaborative filtering. MLSP 2016: 1-6

To Model Neighborhood

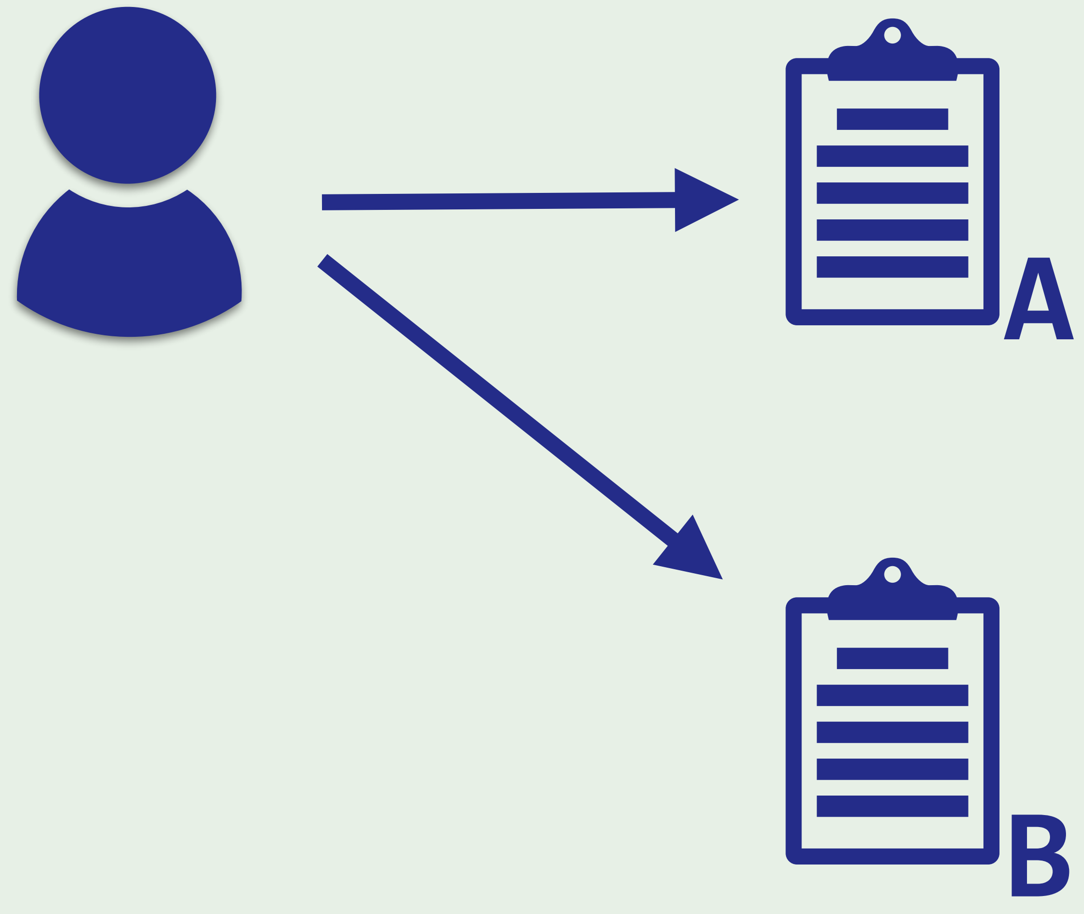




w2v

item2vec in SMORe

Graph



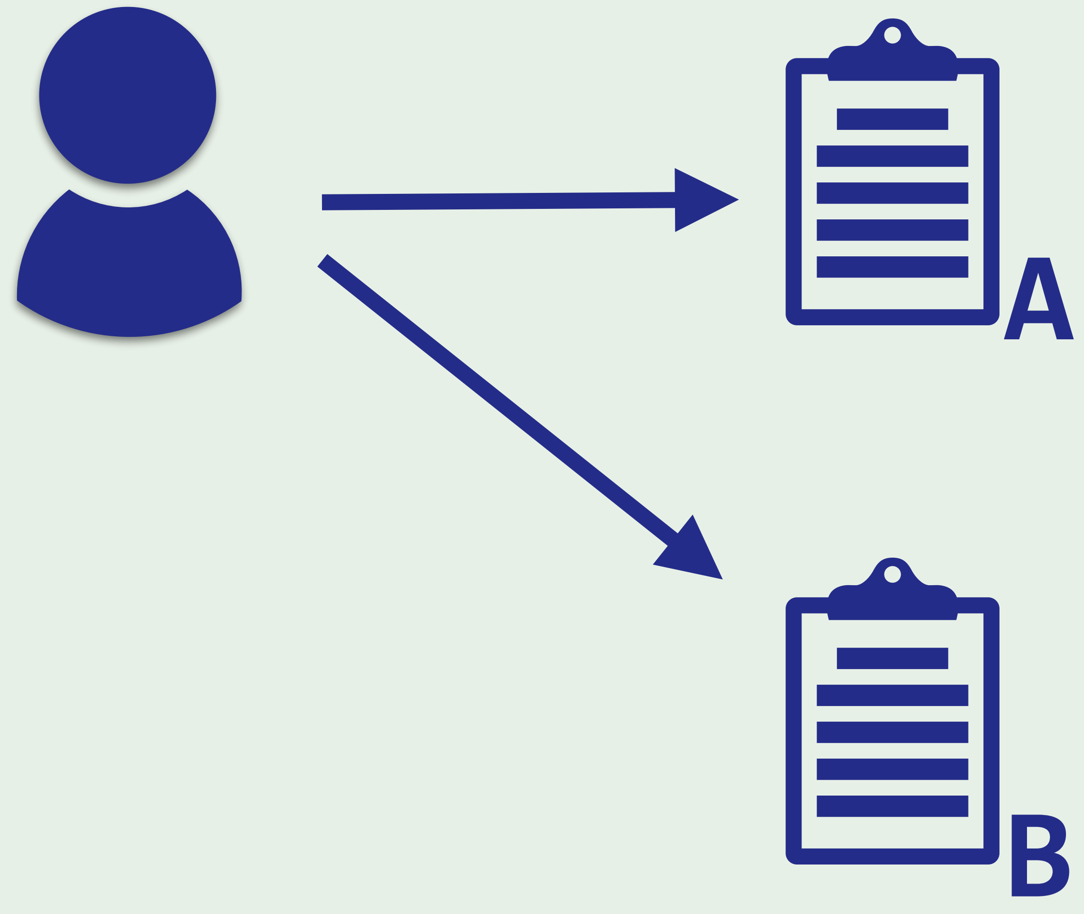
user-item graph

Sampler

(itemA, itemB)

item2vec in SMORe

Graph

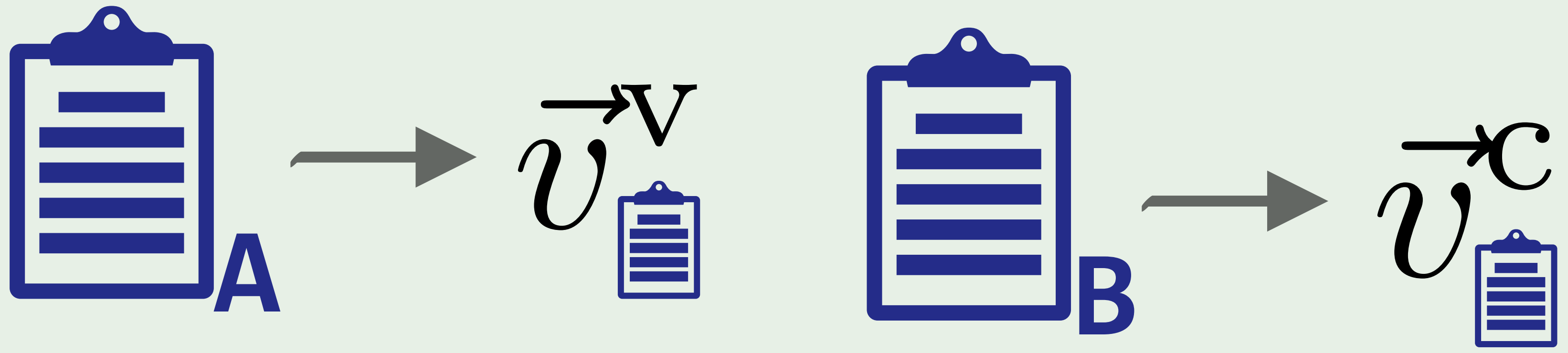


user-item graph

Sampler

(itemA, itemB)

Mapper



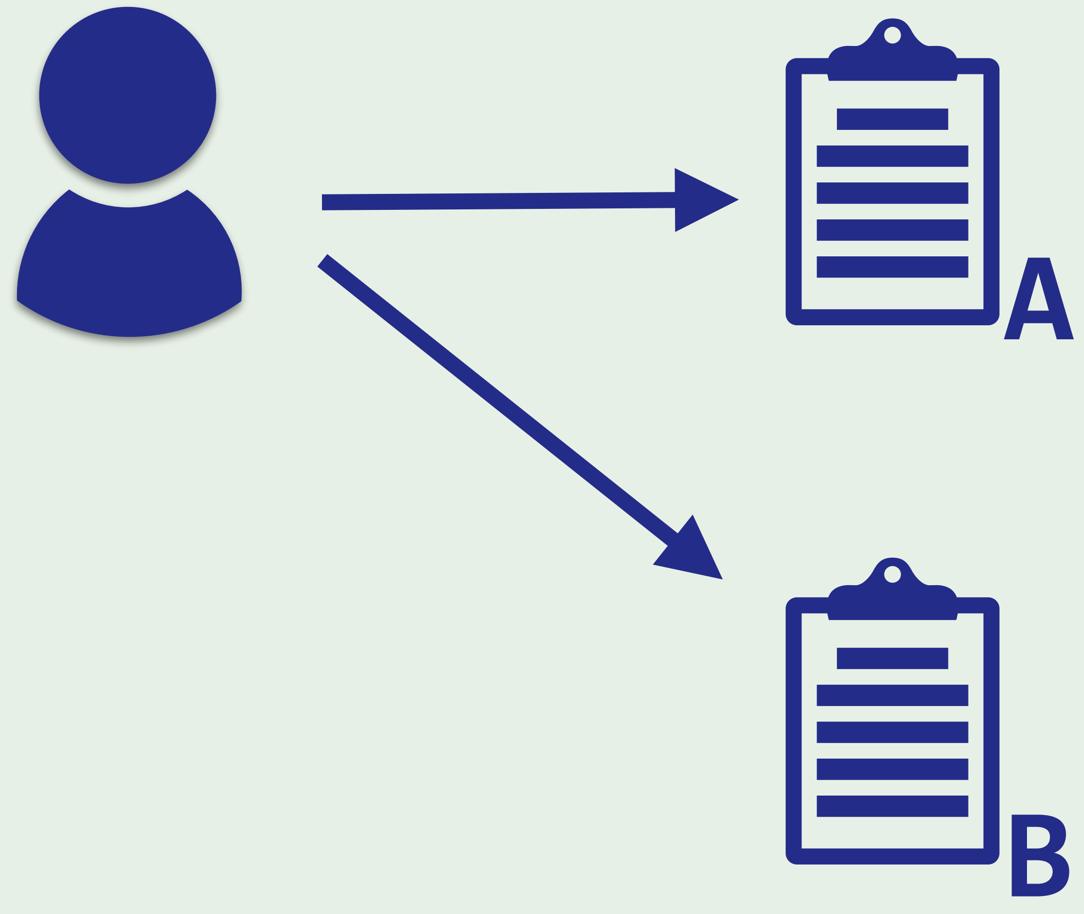
vertex embedding lookup

context embedding lookup

RECSYS 2019

item2vec in SMORe

Graph



user-item graph

Sampler

(A , B)

(itemA, itemB)

Mapper

A → $\vec{v}^{V}$

vertex embedding lookup

B → $\vec{v}^{C}$

context embedding lookup

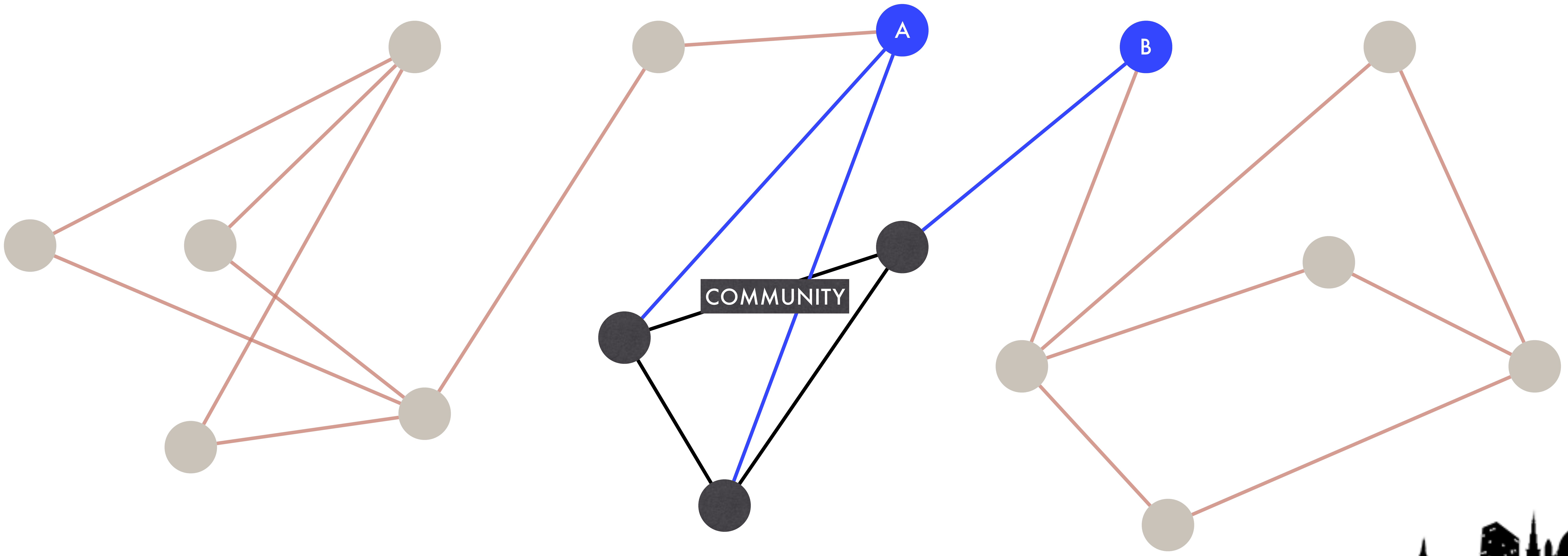
Optimizer

$$\log \sigma(\vec{v}^{V}_{A} \cdot \vec{v}^{C}_{B})$$

Log Likelihood

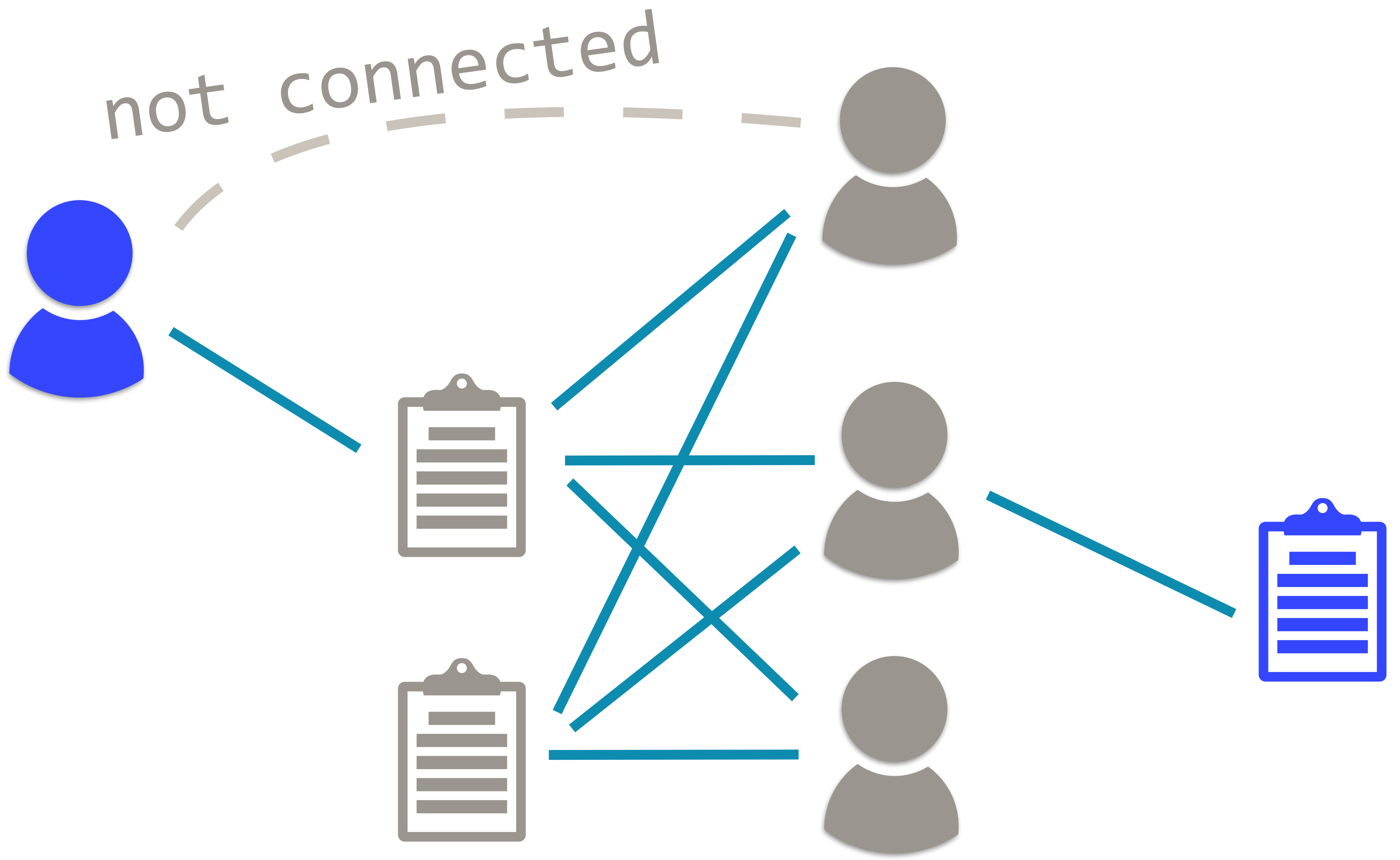
RECSYS 2019

3. Community



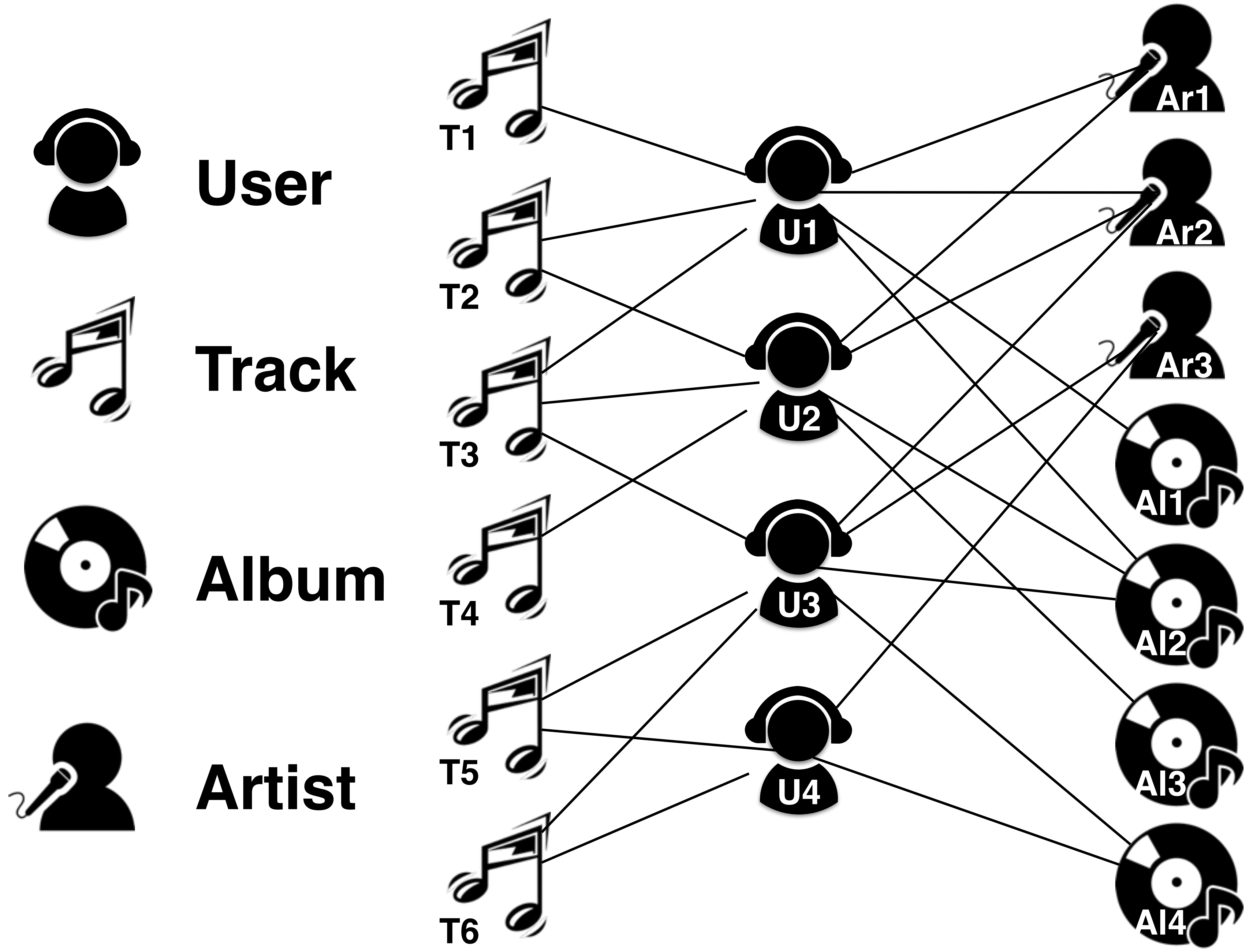
Community

vertices share similar communities are treated as similar pairs



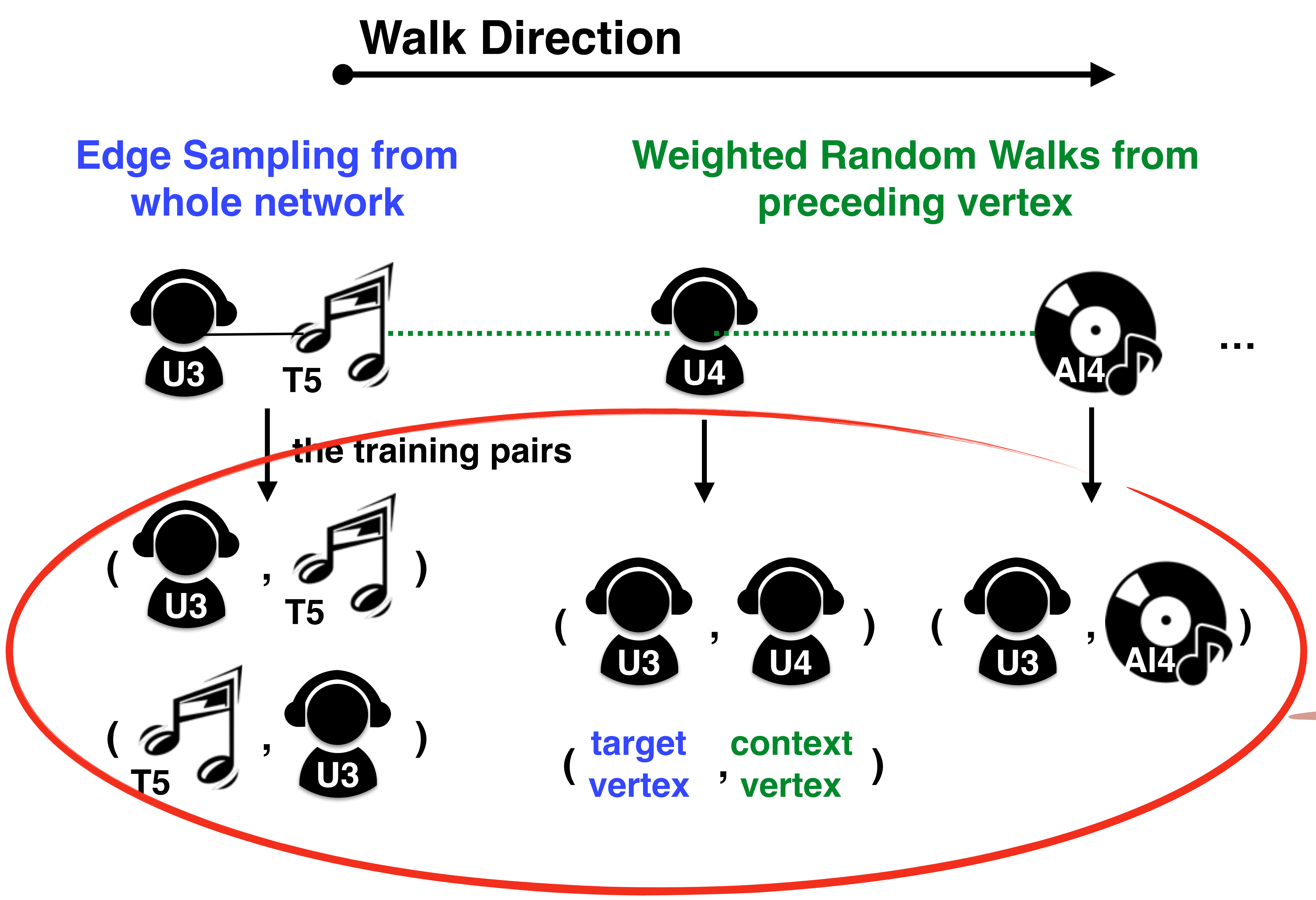
connected to the same group

Heterogeneous Preference Embedding (HPE) *RecSys'16*



Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, Yi-Hsuan Yang: Query-based Music Recommendations via Preference Embedding. RecSys 2016: 79-82

Heterogeneous Preference Embedding (HPE) *RecSys'16*



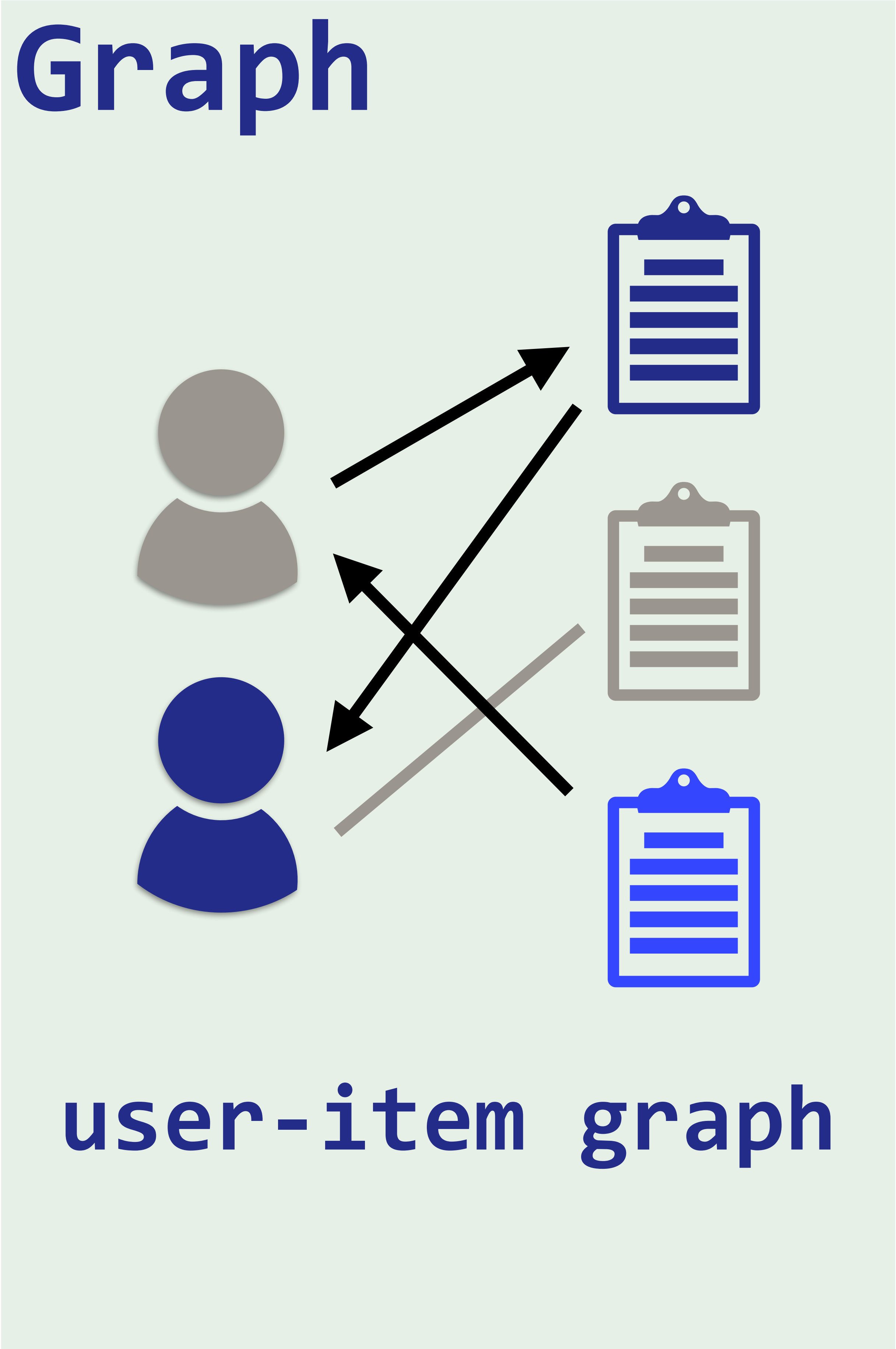
weight embedding

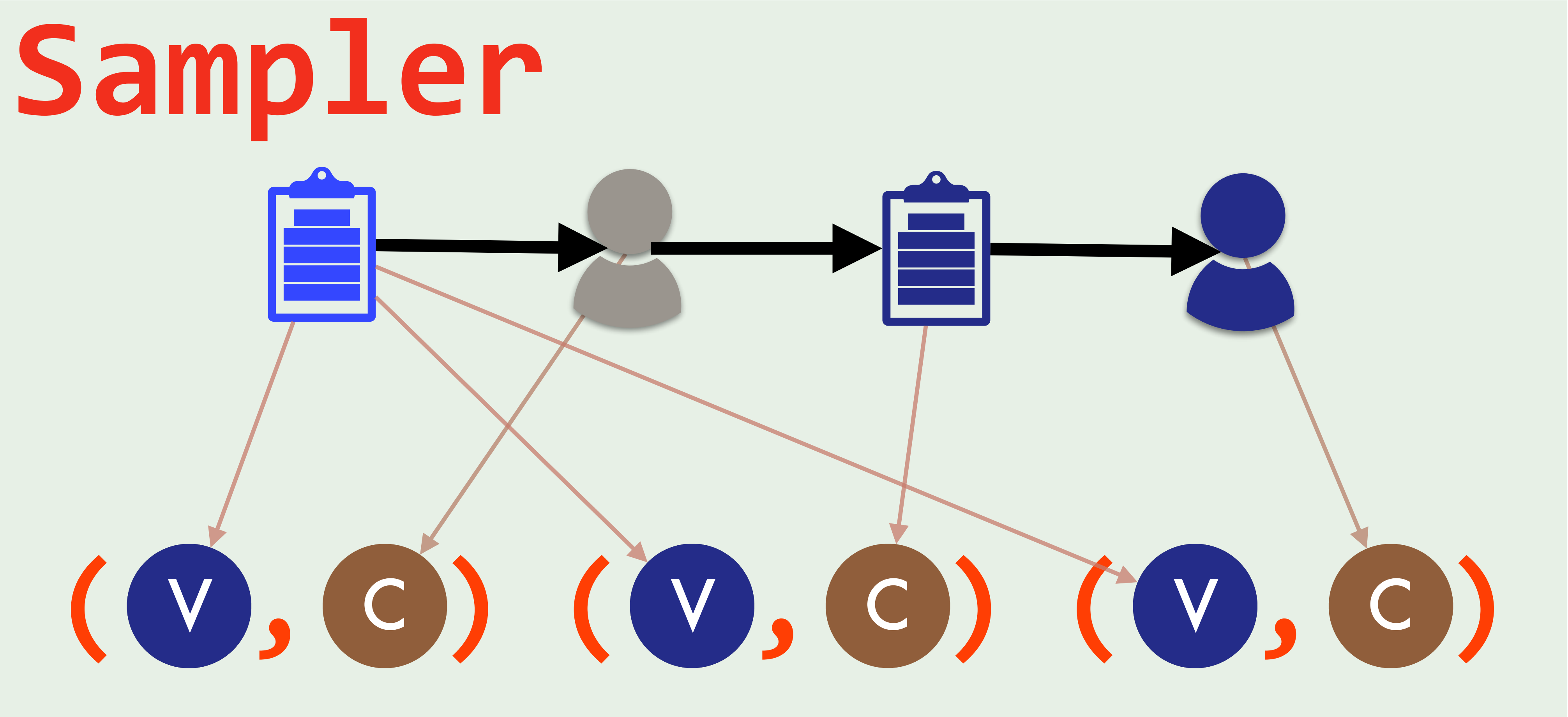
$$\sum_{(i,j)\in S} w_{i,j} \log p(v_j|\Phi(v_i))$$

Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, Yi-Hsuan Yang: Query-based Music Recommendations via Preference Embedding. RecSys 2016: 79-82

RECSYS 2019

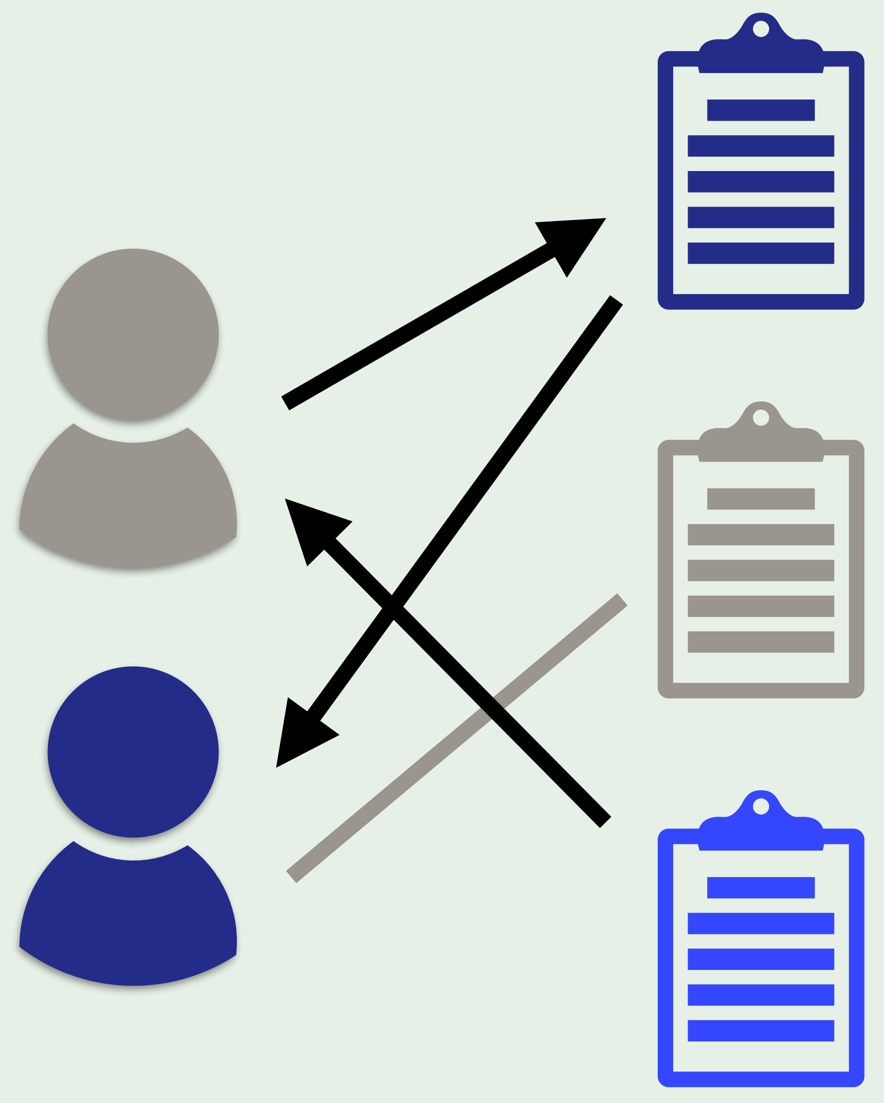
HPE in SMORe





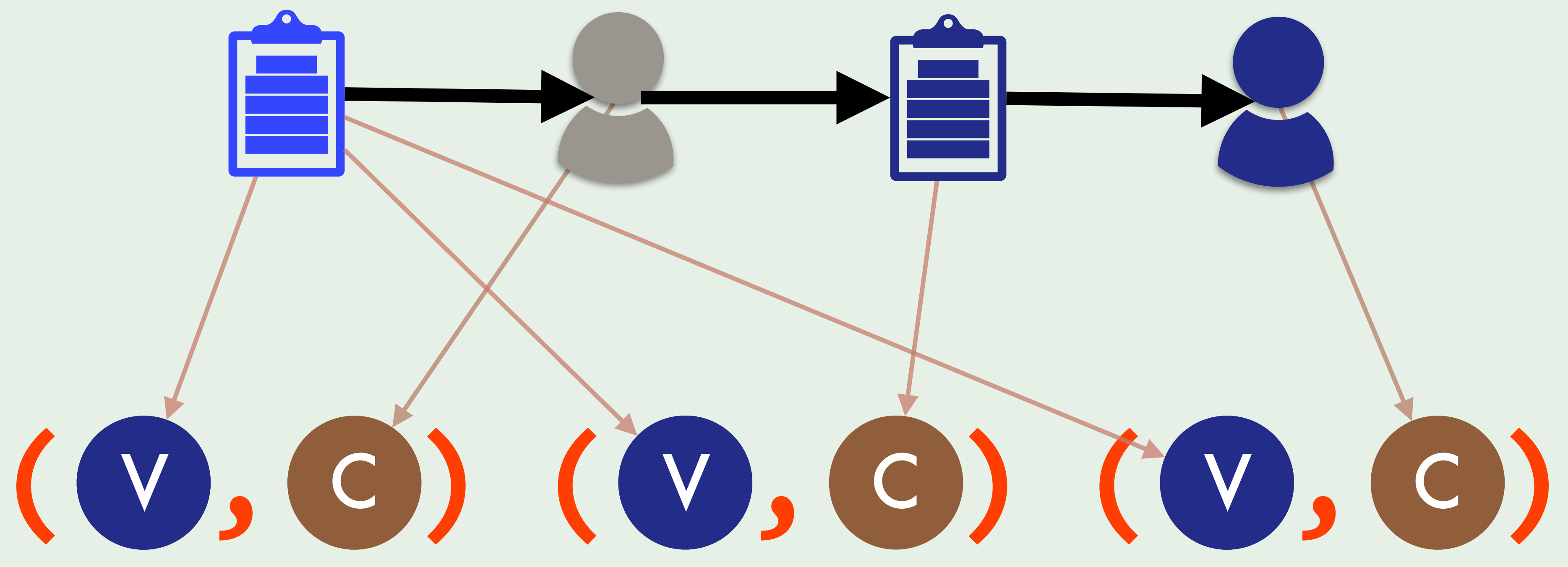
HPE in SMORe

Graph



user-item graph

Sampler



Mapper

V → $\vec{v}^{V}_{v}$

vertex
embedding lookup

C → $\vec{v}^{C}_{c}$

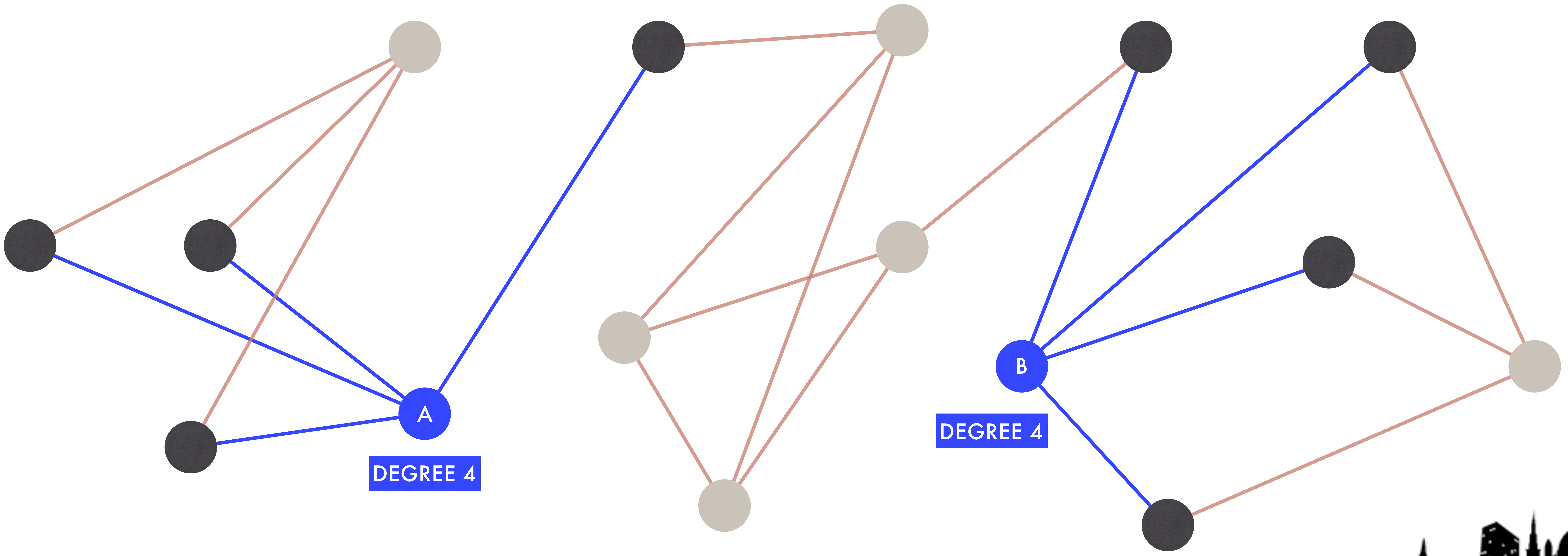
context
embedding lookup

Optimizer

$$\log \sigma(\vec{v}^{V}_{v} \cdot \vec{v}^{C}_{c})$$

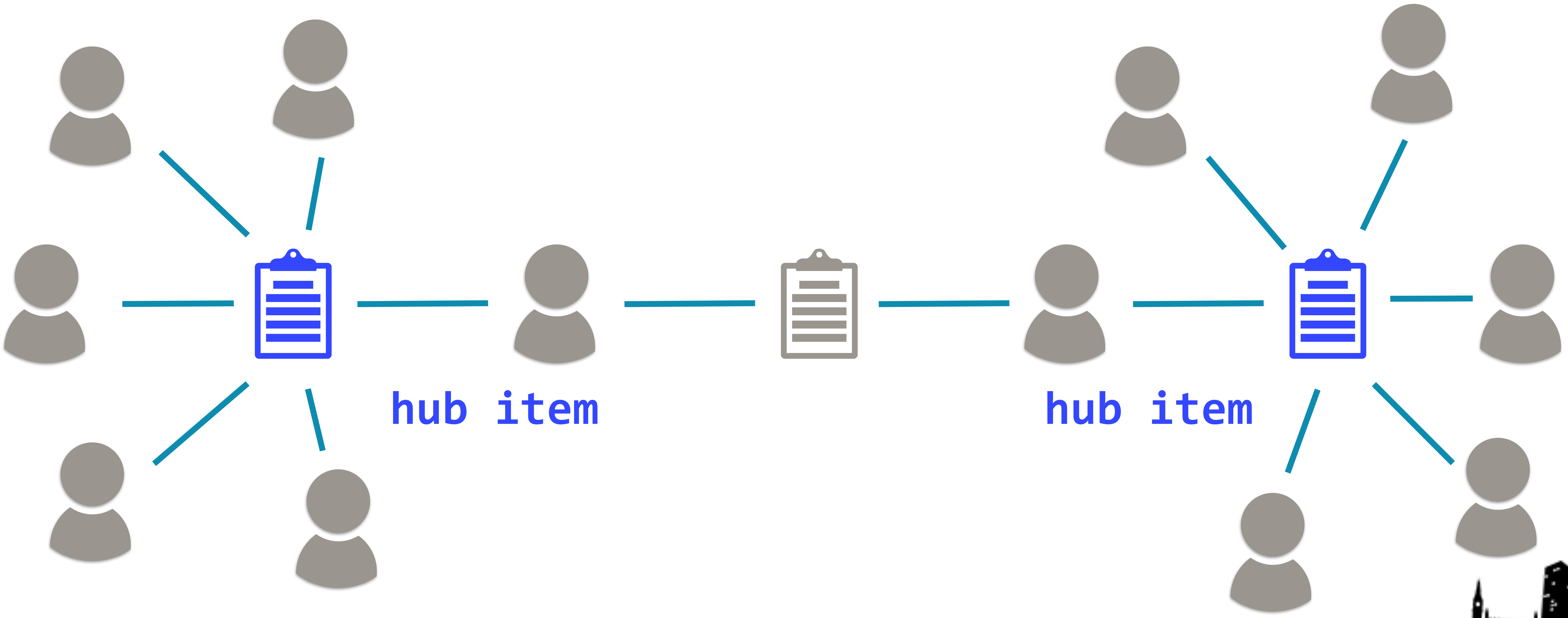
Log Likelihood

4. Centrality (Degree)

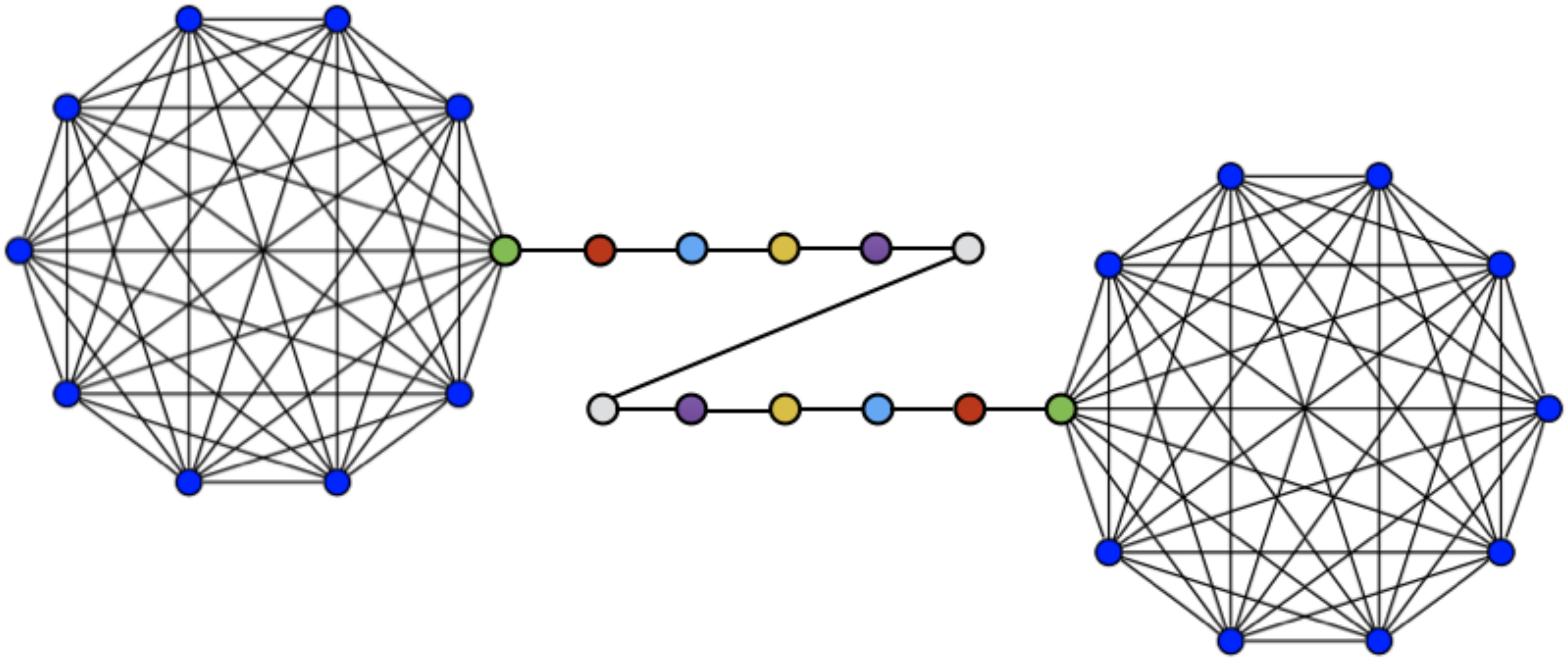


Centrality

vertices share similar properties are treated as similar pairs

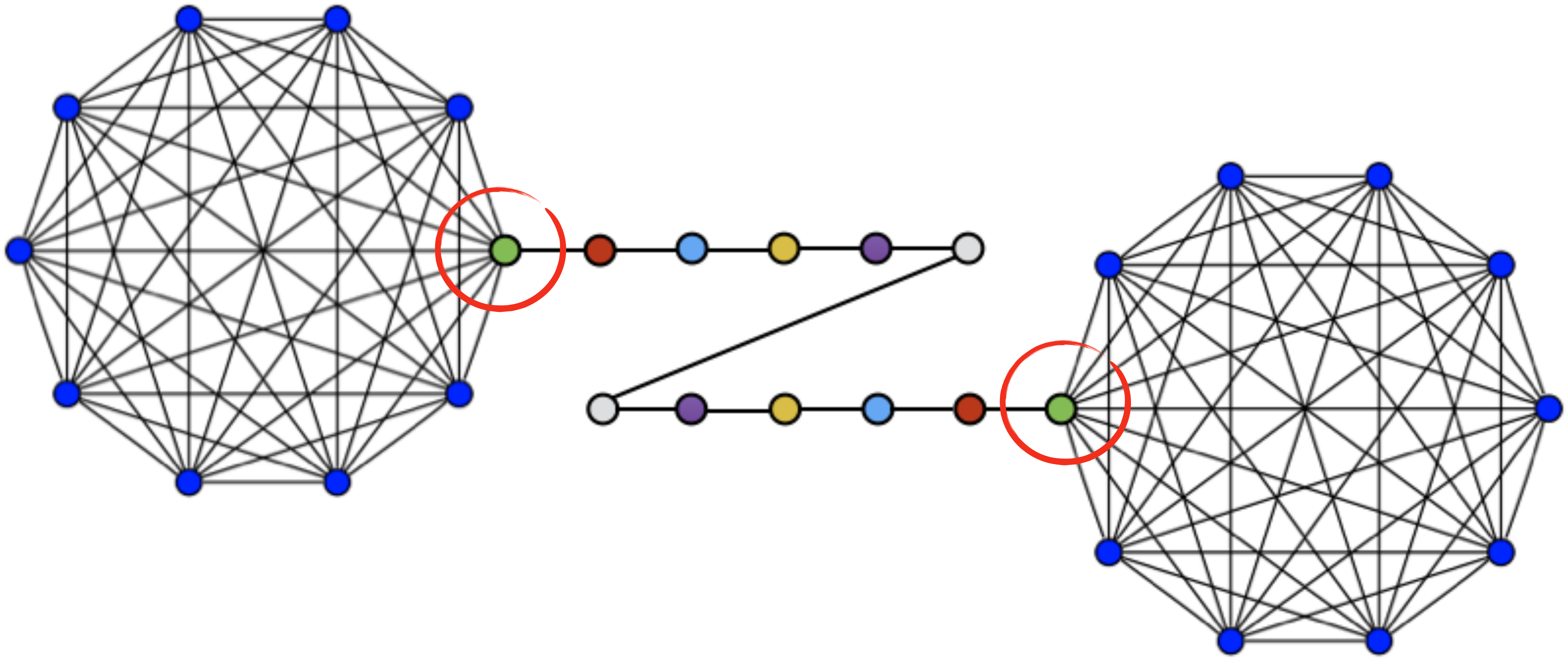


struc2vec *KDD'17*



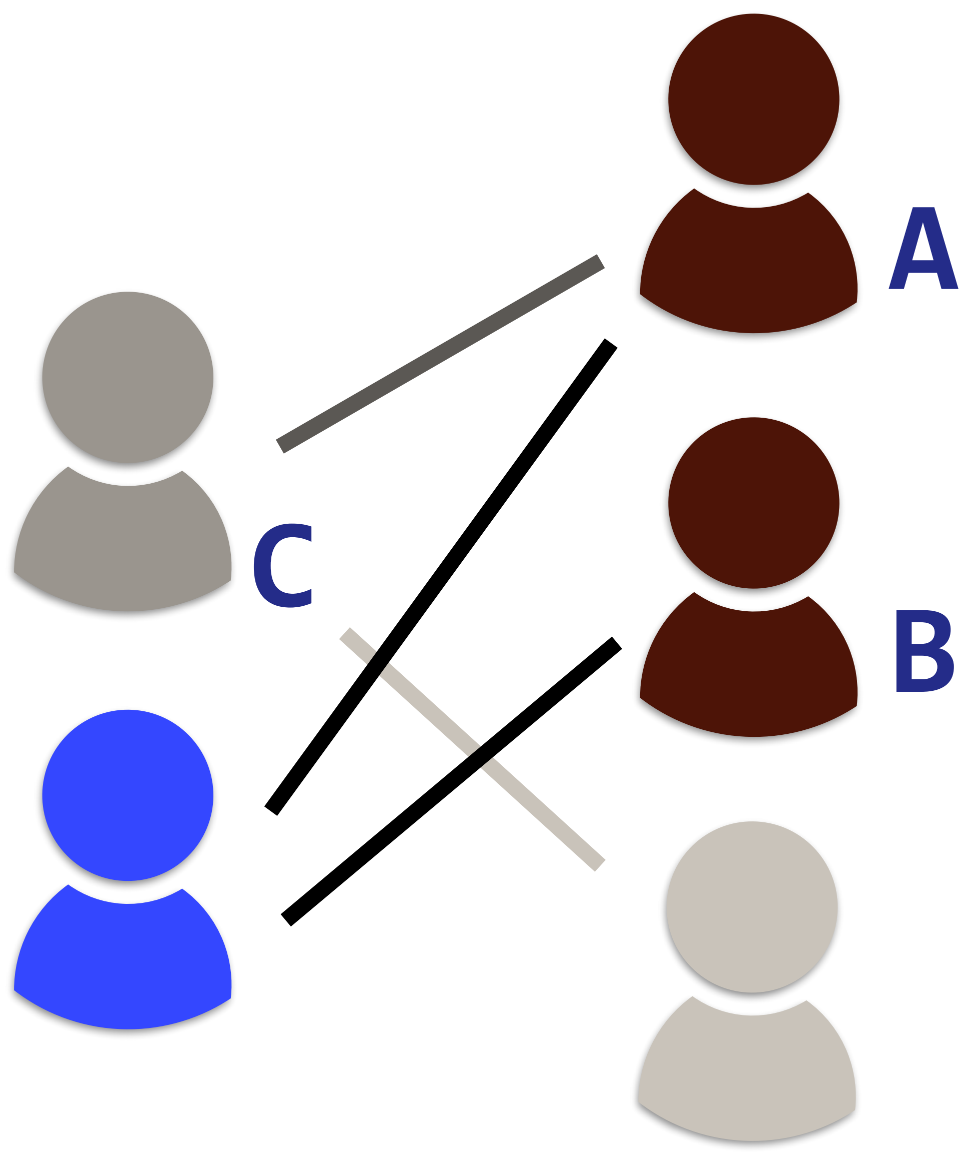
Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, Daniel R. Figueiredo: struc2vec: Learning Node Representations from Structural Identity. KDD 2017: 385-394

struc2vec *KDD'17*



Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, Daniel R. Figueiredo: struc2vec: Learning Node Representations from Structural Identity. KDD 2017: 385-394

struc2vec *KDD'17*



user-user graph

1-step:

fn1(A) = structure1_A

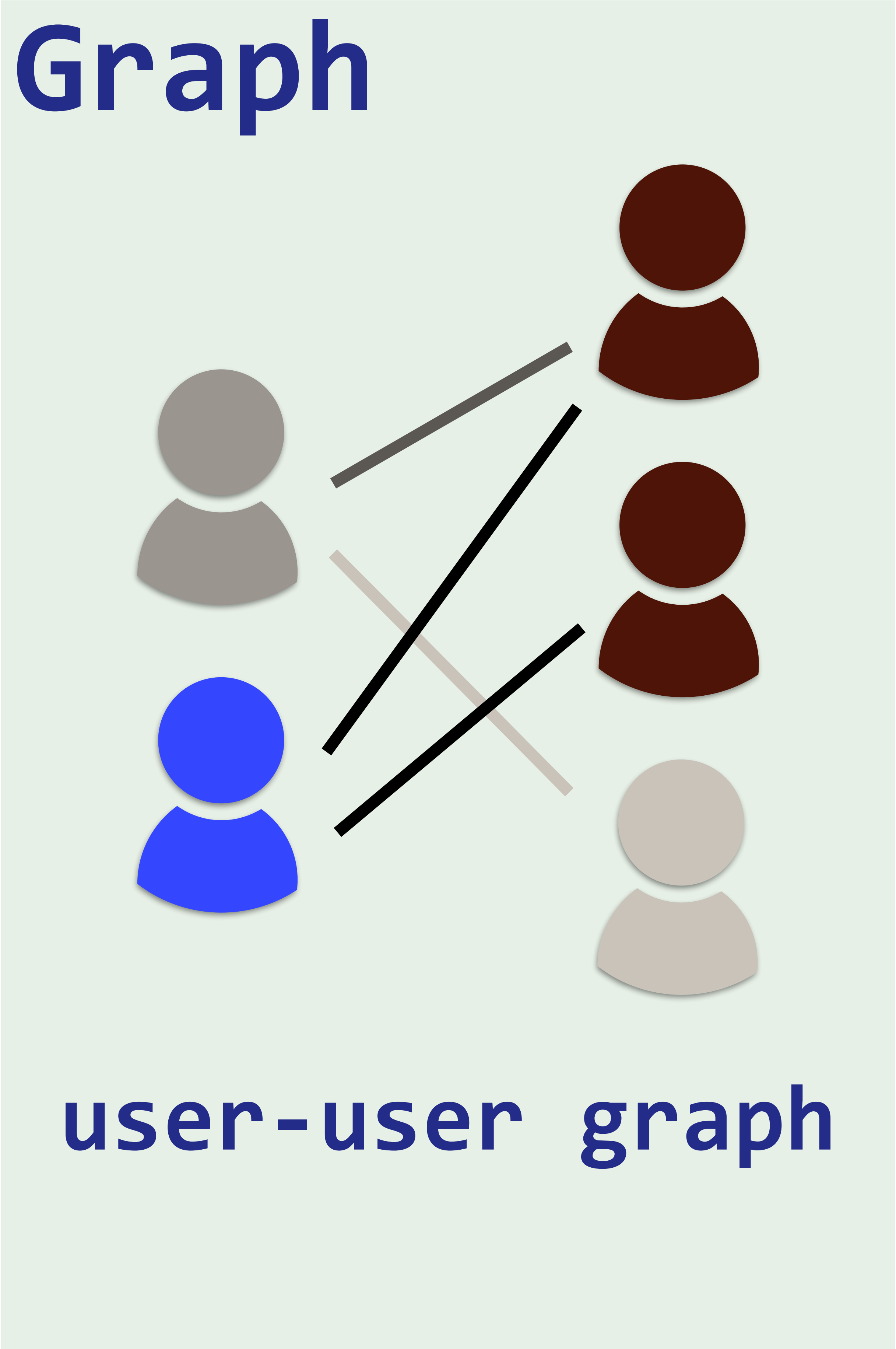
fn1(B) = structure1_B

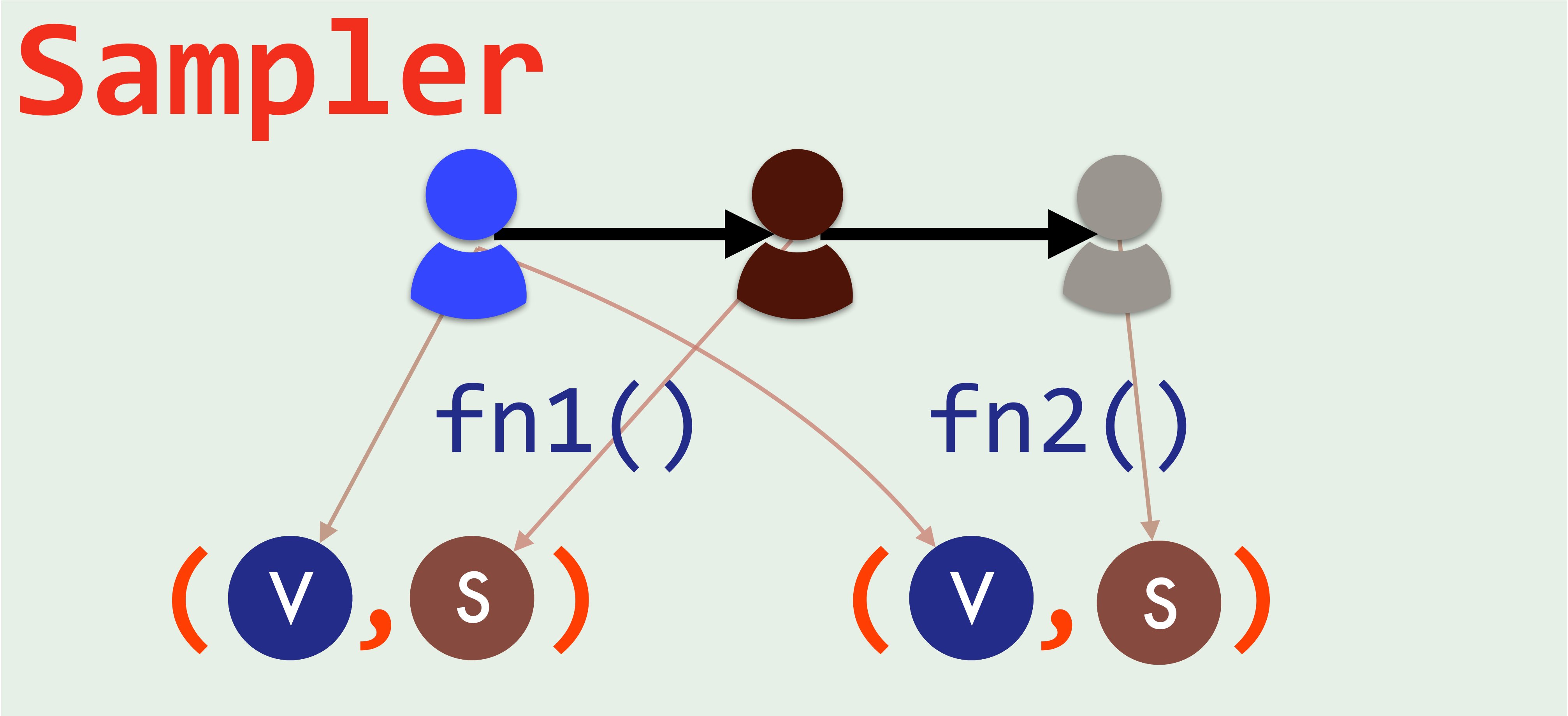
2-steps:

fn2(C) = structure2_C

Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, Daniel R. Figueiredo: struc2vec: Learning Node Representations from Structural Identity. KDD 2017: 385-394

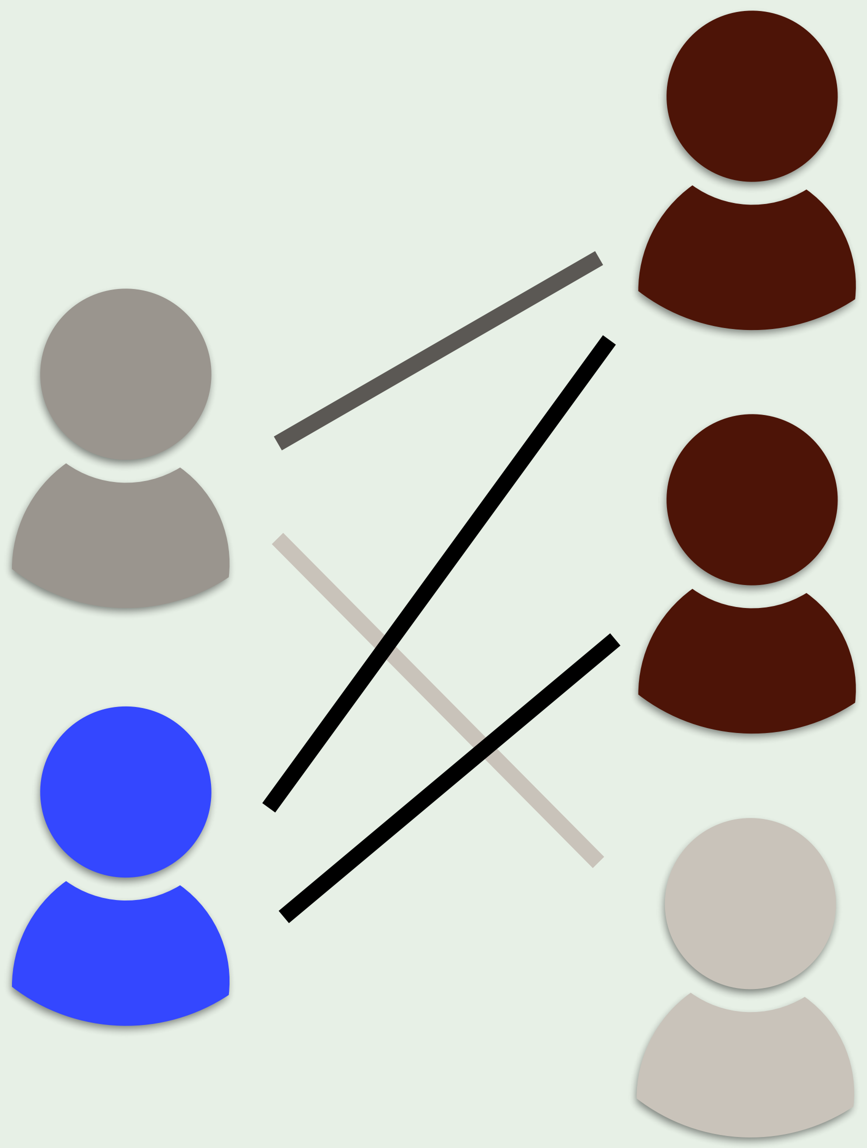
struc2vec in SMORe





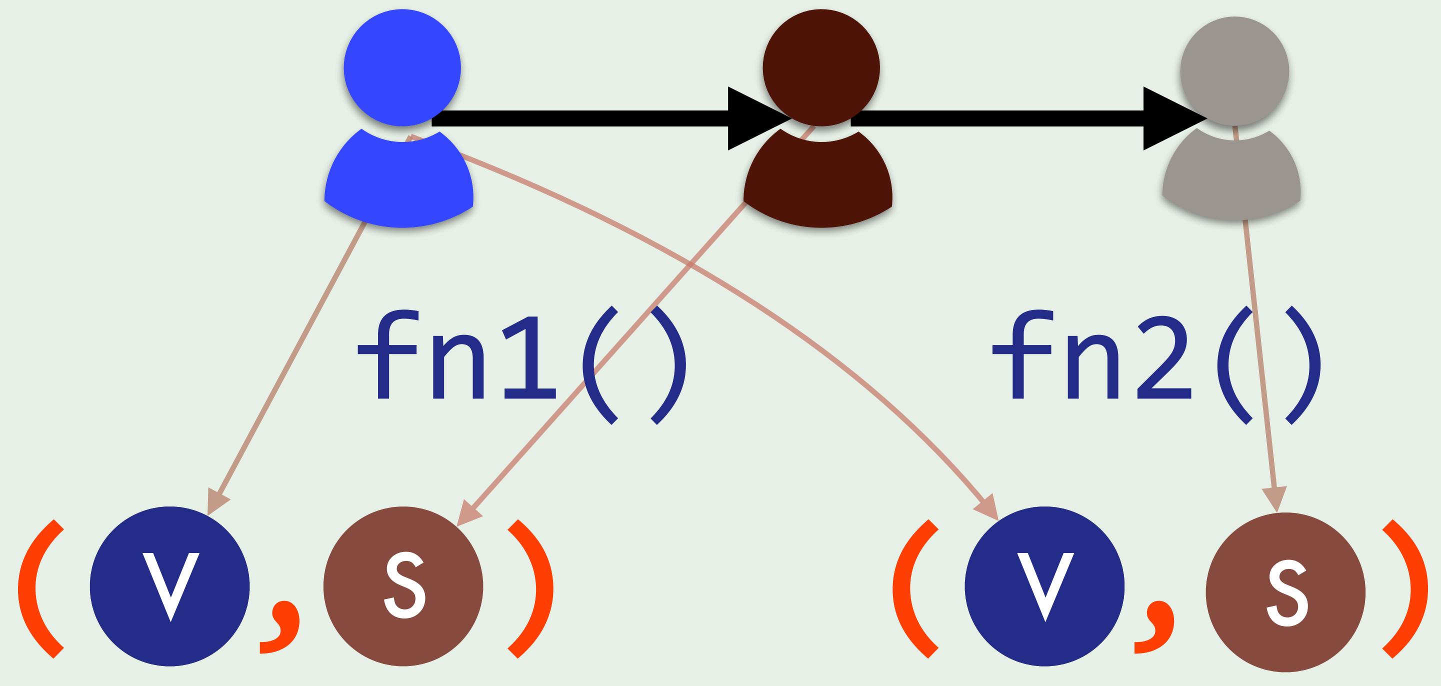
struc2vec in SMORe

Graph



user-user graph

Sampler



Mapper

$$\text{V} \longrightarrow \vec{v}^{\,\text{V}}_{\text{V}}$$

vertex embedding lookup

$$\text{S} \longrightarrow \vec{v}^{\,\text{C}}_{\text{S}}$$

context embedding lookup

Optimizer

$$\log \sigma(\vec{v}^{\,\text{V}}_{\text{V}} \cdot \vec{v}^{\,\text{C}}_{\text{S}})$$

Log Likelihood

note. original work adopts hierarchal softmax for learning

RECSYS 2019

Recap

	Sampler	Mapper	Optimizer
MF	Adjacency	Vertex Embedding	Dot product RMSE
SLIM	Adjacency	Vertex Embedding	Dot product RMSE
item2vec	Neighborhood	Vertex Embedding Context Embedding	Sigmoid Dot Product Log Likelihood
HPE	Community	Vertex Embedding Context Embedding	Sigmoid Dot Product Log Likelihood
struc2vec	Centrality	Vertex Embedding Context Embedding	Sigmoid Dot Product Log Likelihood

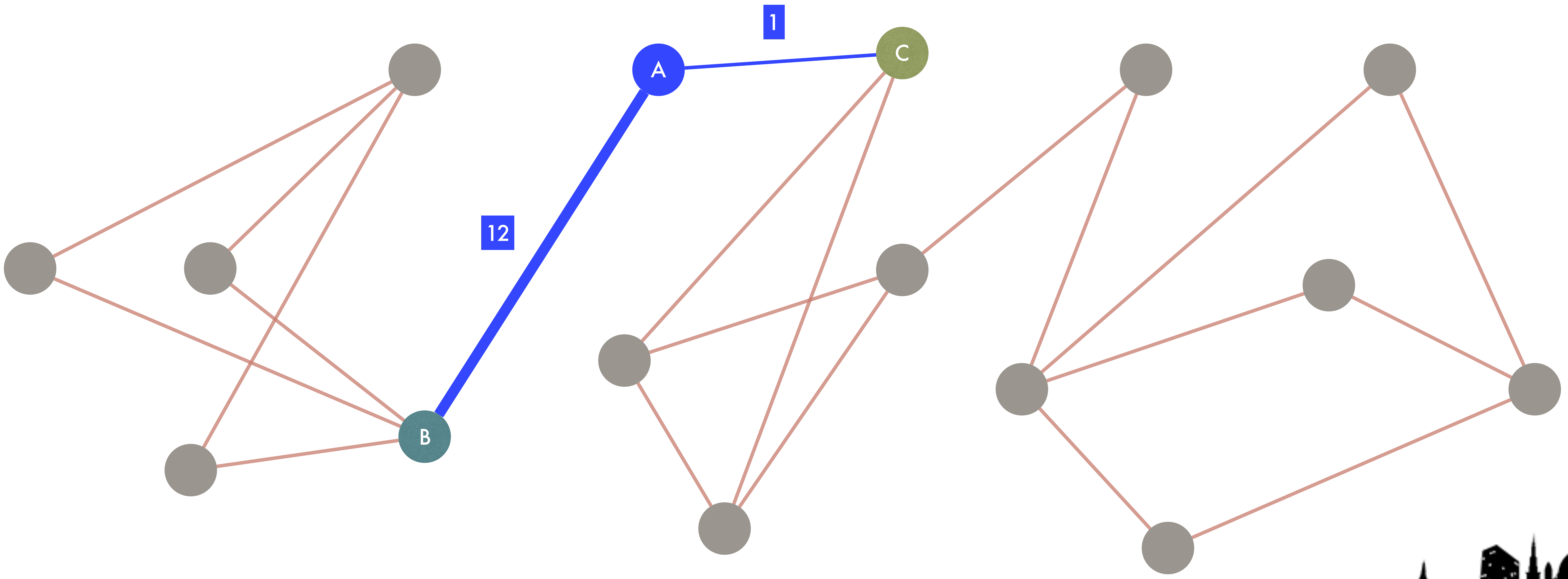
RECSYS 2019

Edge Information

- Weight
  - reflects strength of interaction
- Closeness
  - reflects order of interaction
- Semantics
  - sample with grammar
- Inductiveness
  - information diffuses through edges

RECSYS 2019

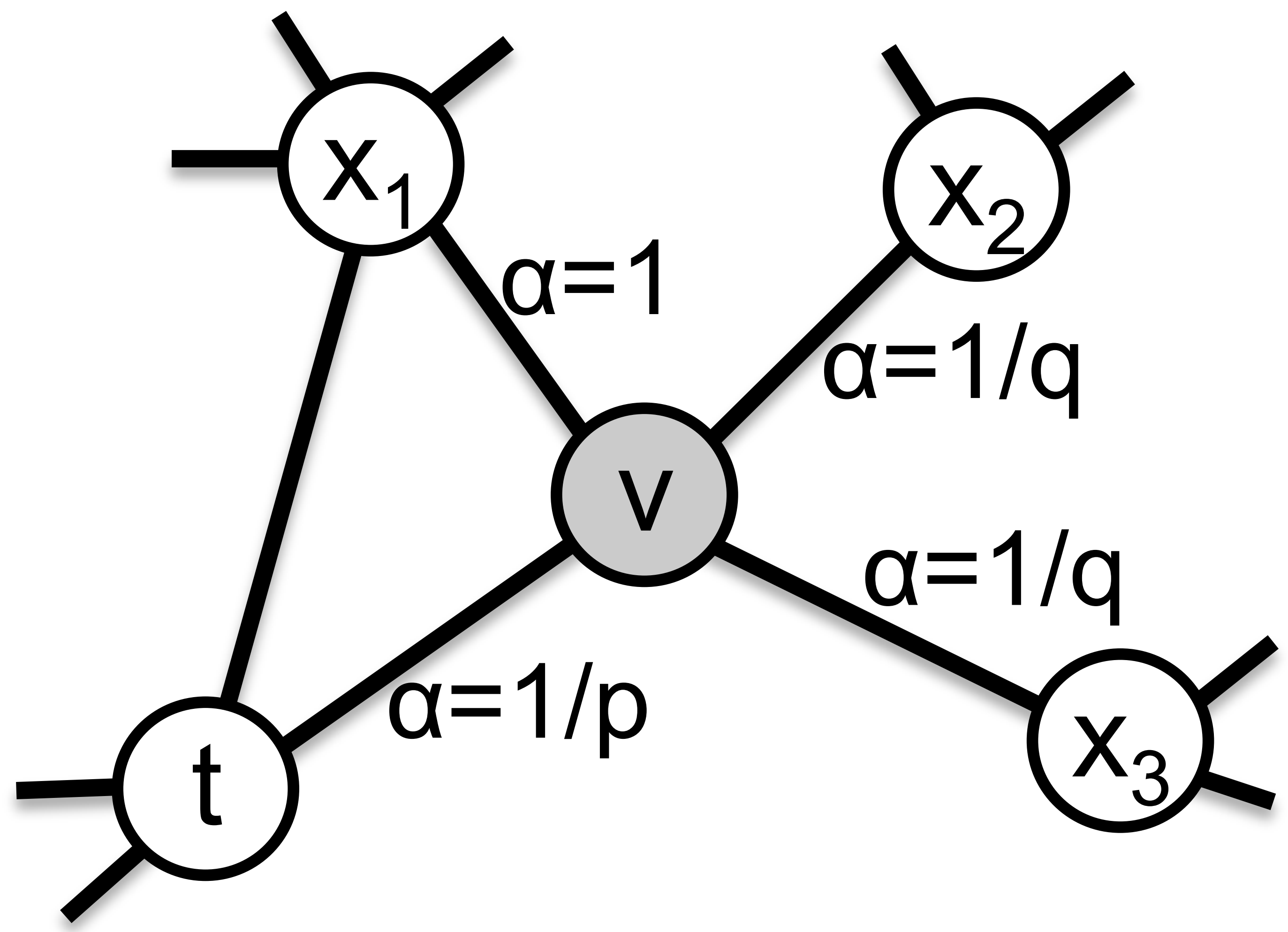
1. Weight

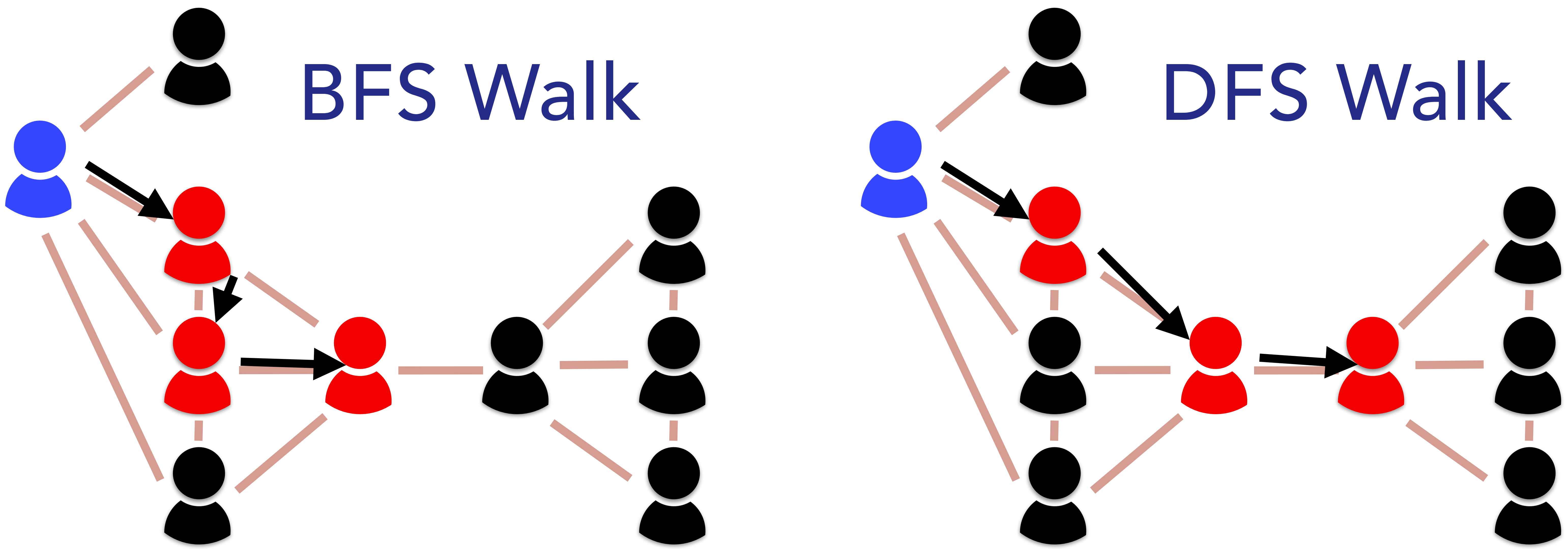


RECSYS 2019

node2vec *KDD'16*

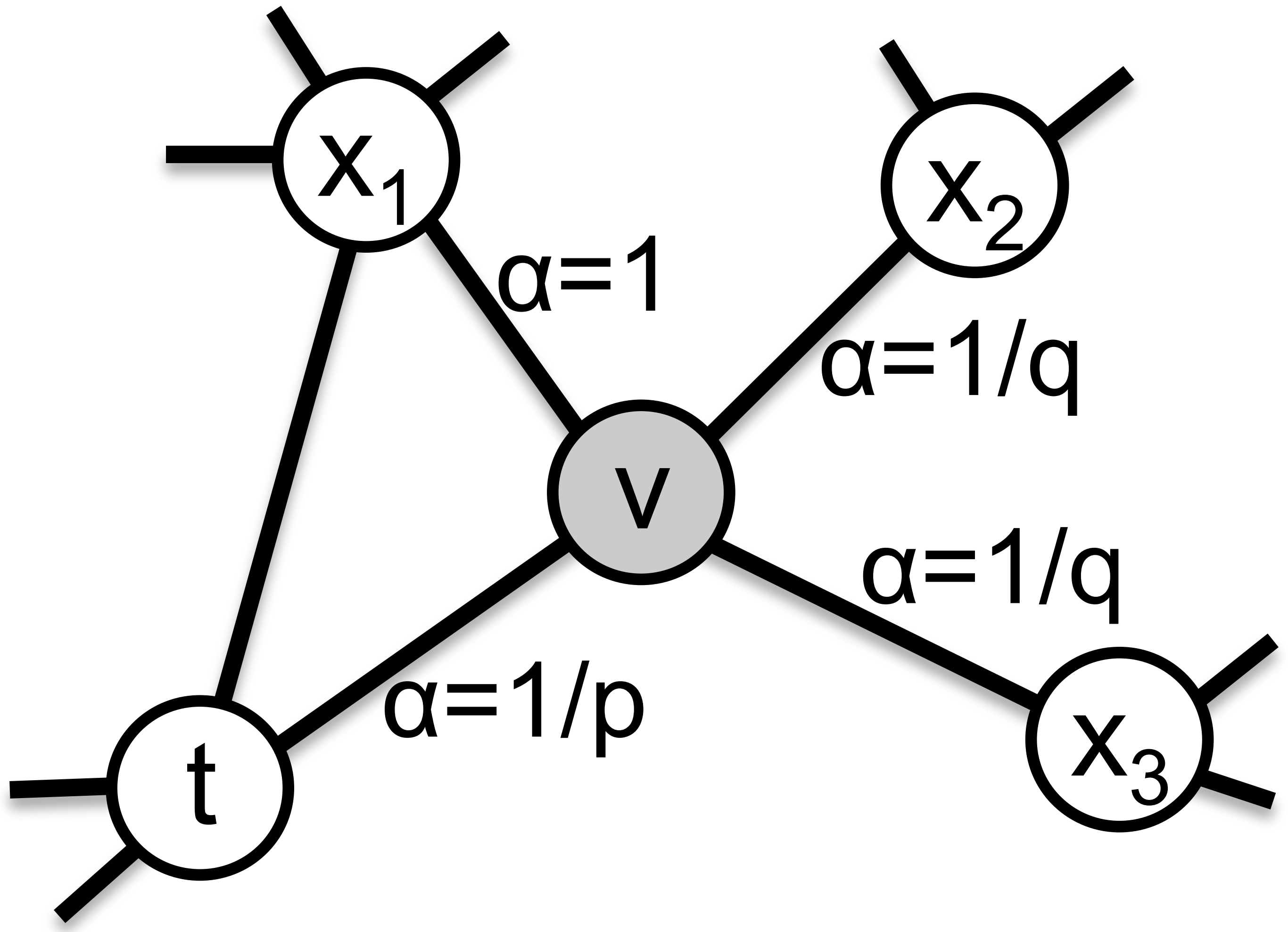
edge weight guides the walk





Aditya Grover, Jure Leskovec: node2vec: Scalable Feature Learning for Networks. KDD 2016: 855-864

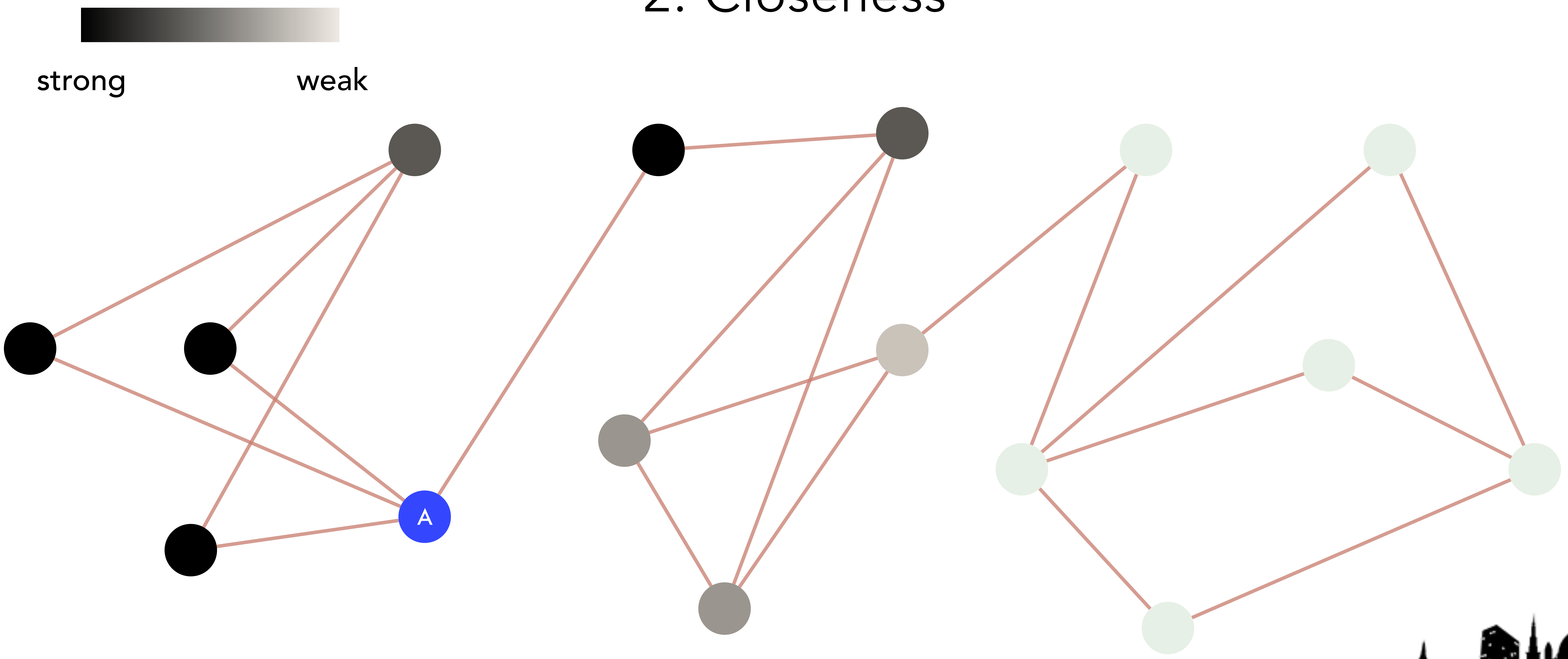
node2vec in SMORe



Sampler

customized random walk

2. Closeness



High-order Proximity for Recommendations (HOP-Rec)

order-aware interaction

RecSys'18

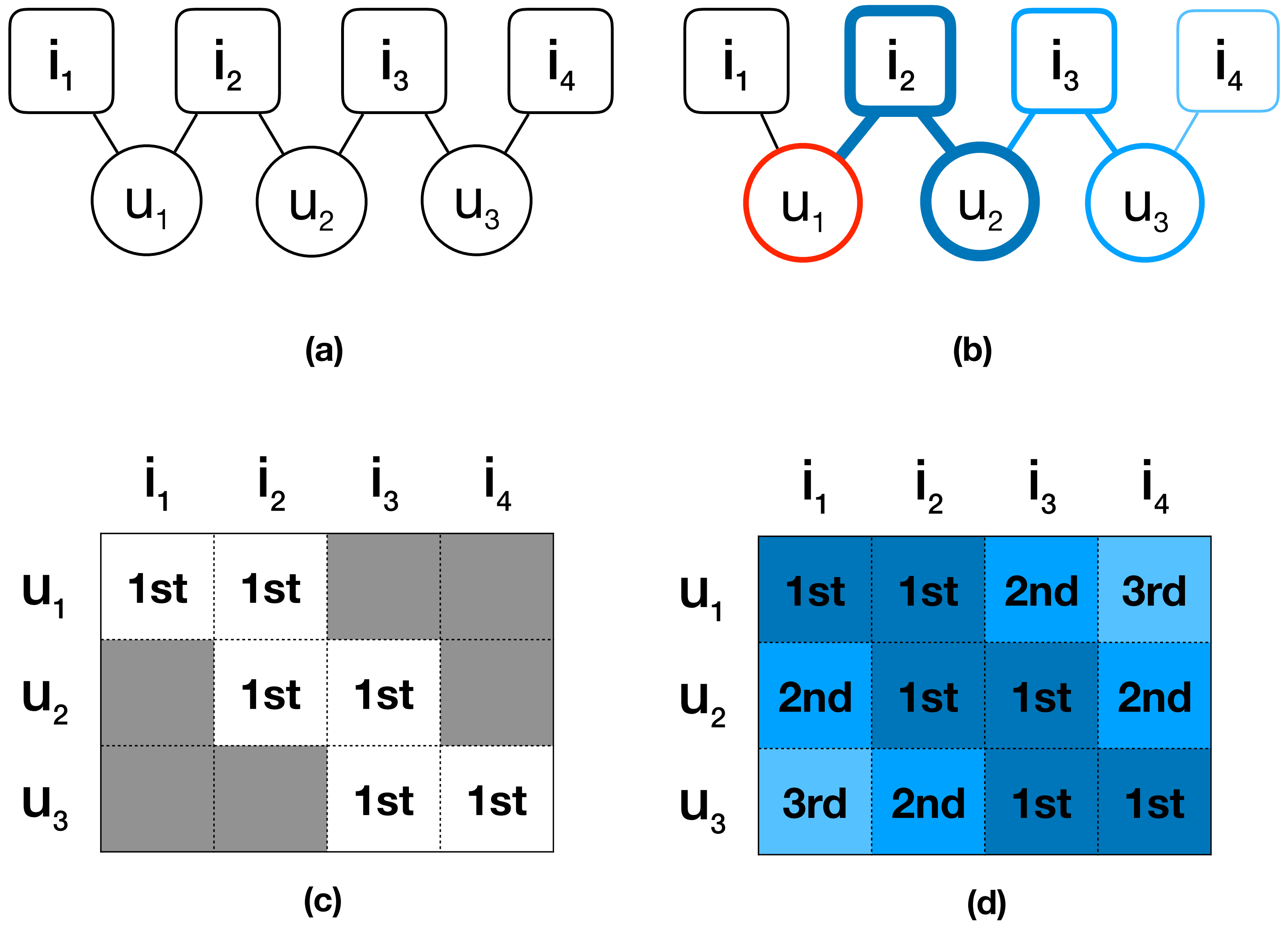


Figure 1: High-order proximity between users and items within observed interactions

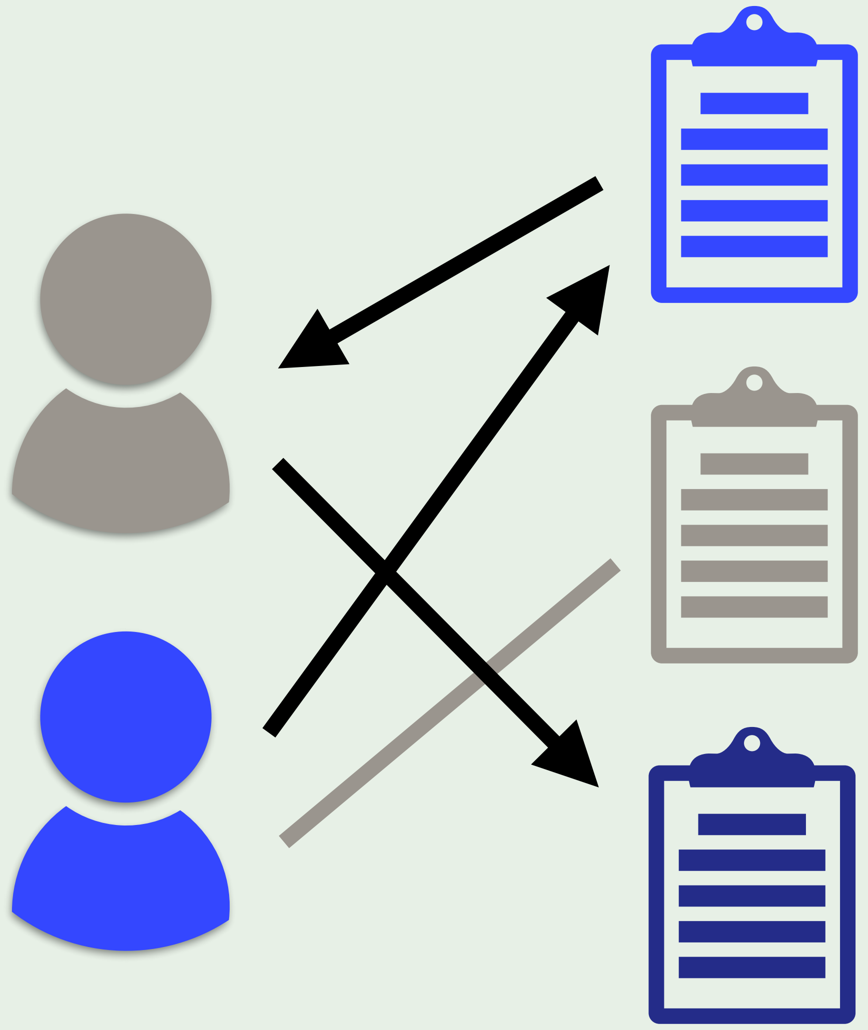
$$\sum_{\substack{1\le k\le K\\ u,(i,i')}} \overbrace{C(k)\,\mathbb{E}_{\substack{i\sim P_u^k\\ i'\sim P_N}}}^{\text{graph model}} \overbrace{\left[\mathcal{F}\left(\theta_u^\top\theta_{i'},\theta_u^\top\theta_i\right)\right]}^{\text{factorization model}}$$

$$\mathbb{1}_{\{\theta_u^\top\theta_{i'}-\theta_u^\top\theta_i>\epsilon_k\}}\log\left[\sigma\left(\theta_u^\top\theta_{i'}-\theta_u^\top\theta_i\right)\right]$$

Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai: HOP-rec: high-order proximity for implicit recommendation. RecSys 2018: 140-144

HOP-Rec in SMORe

Graph



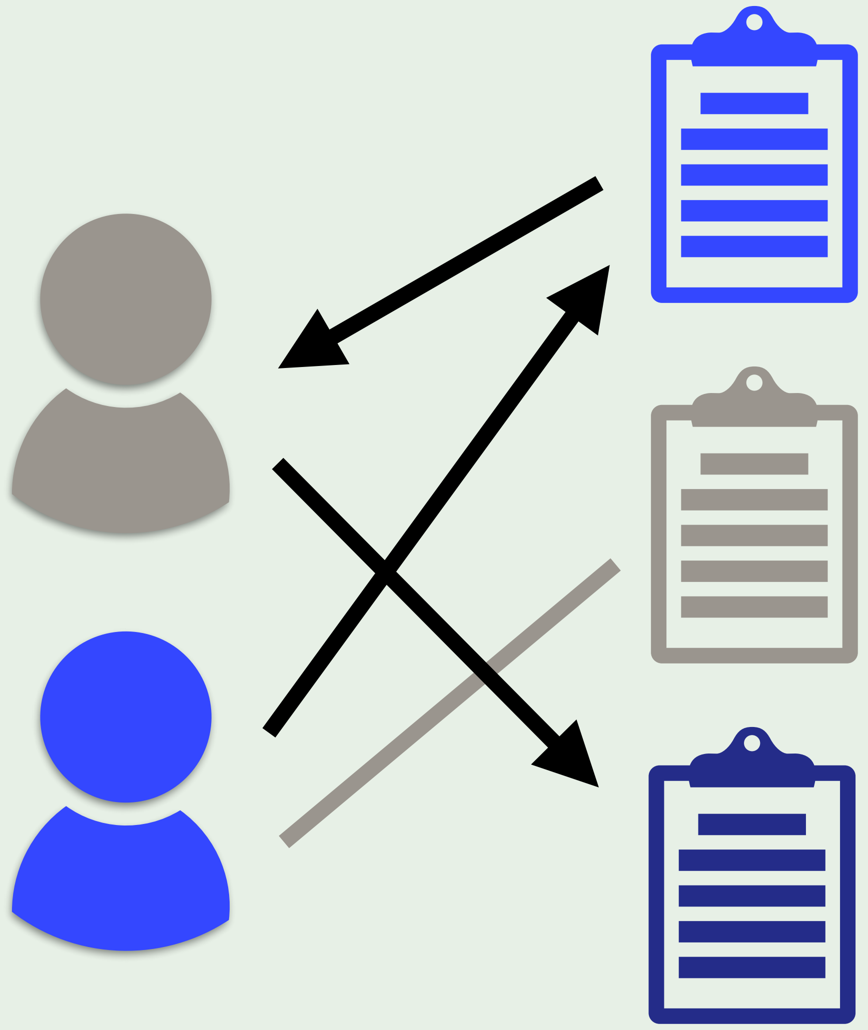
user-item graph

Sampler

$$\mathbb{1}(\vec{v} \cdot \vec{v} - \vec{v} \cdot \vec{v}_{R} < \text{ margin})$$

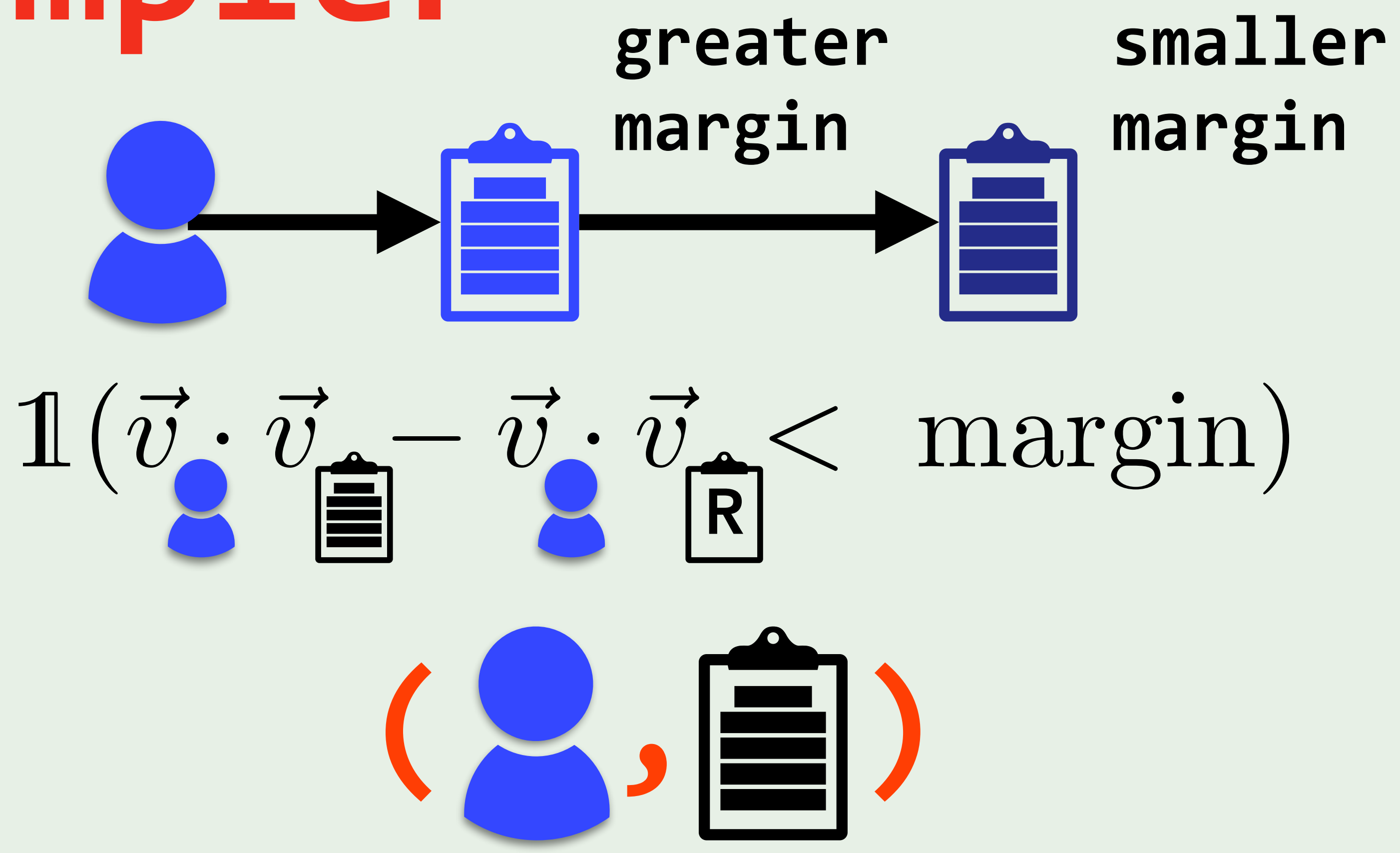
HOP-Rec in SMORe

Graph

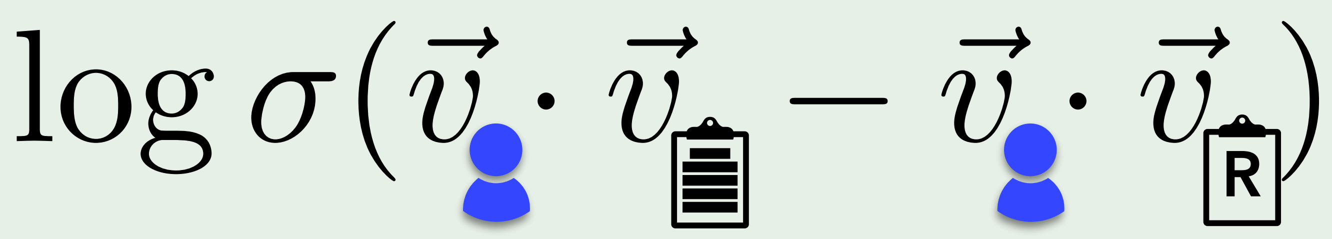


user-item graph

Sampler

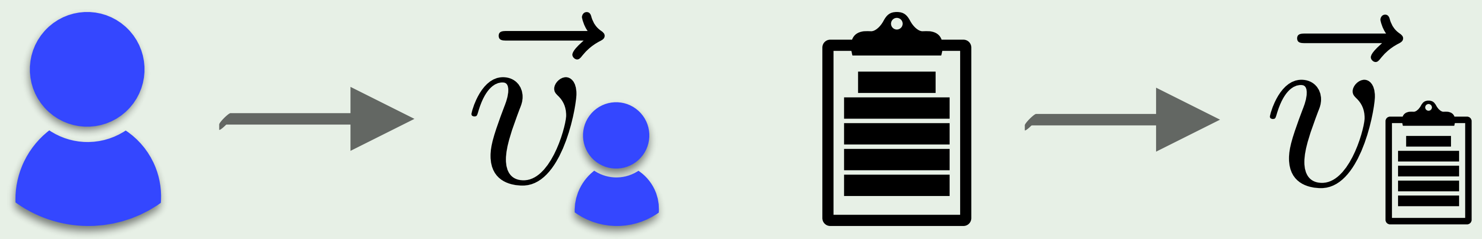


Optimizer



BPRLoss

Mapper

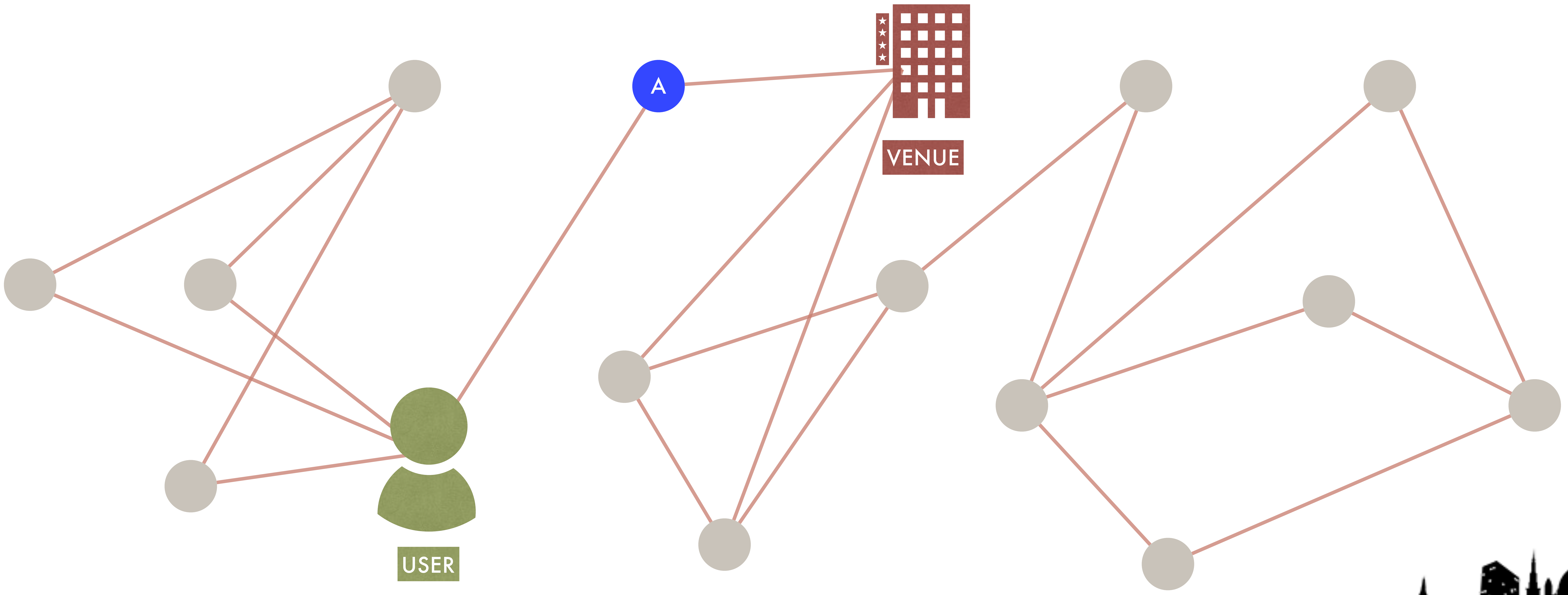


embedding lookup

1. *Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme: BPR: Bayesian Personalized Ranking from Implicit Feedback. UAI 2009: 452-461*

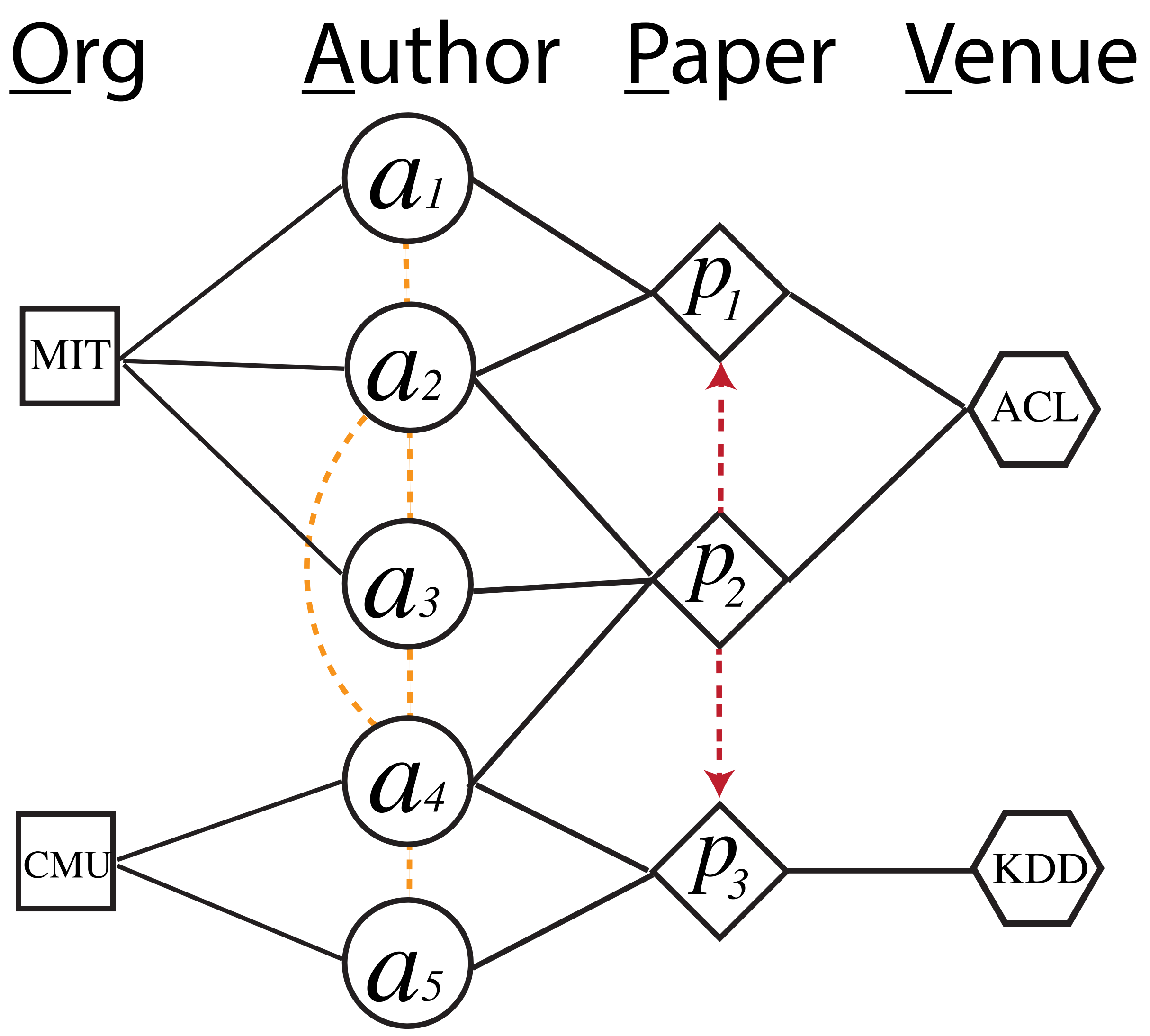
RECSYS 2019

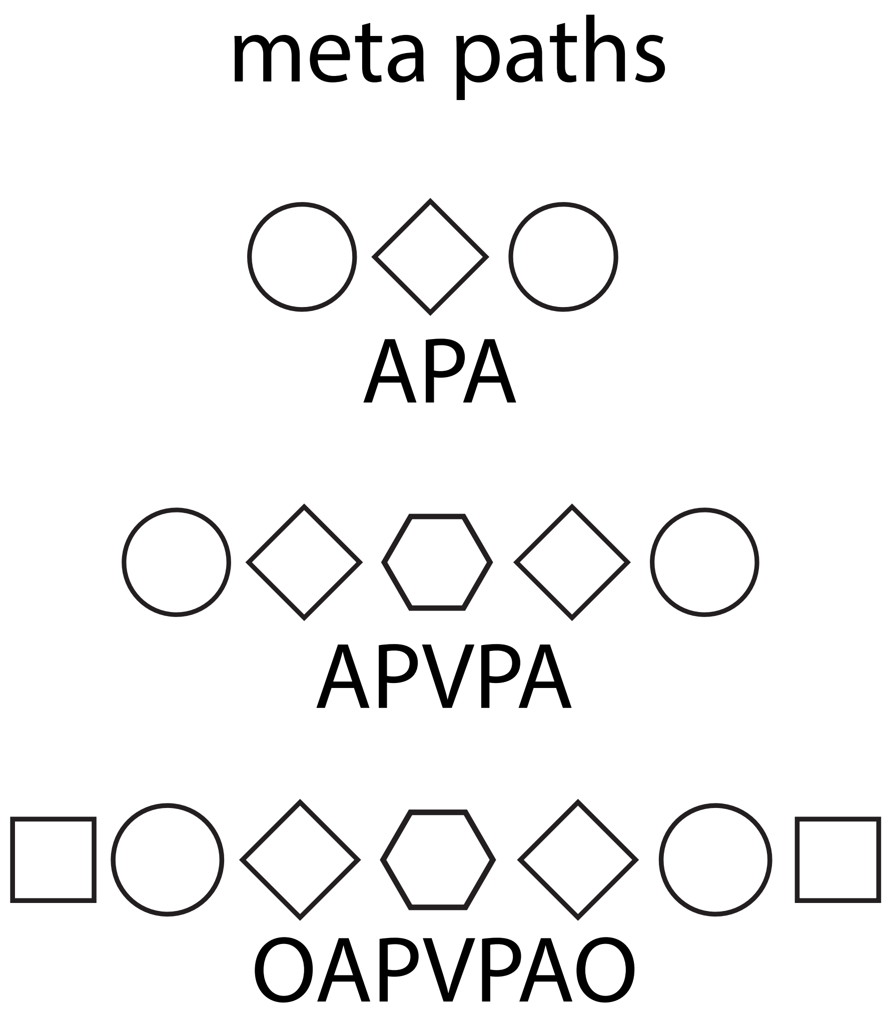
3. Semantics



metapath2vec *KDD'17*

metapath carries semantics





$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log p(c_t | v; \theta)$$

different types

neighbors in given type

Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami: metapath2vec: Scalable Representation Learning for Heterogeneous Networks. KDD 2017: 135-144

RECSYS 2019

metapath2vec in SMORe

Sampler

$$\mathcal{P}: V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$$

grammar1: Paper -> Author -> Paper

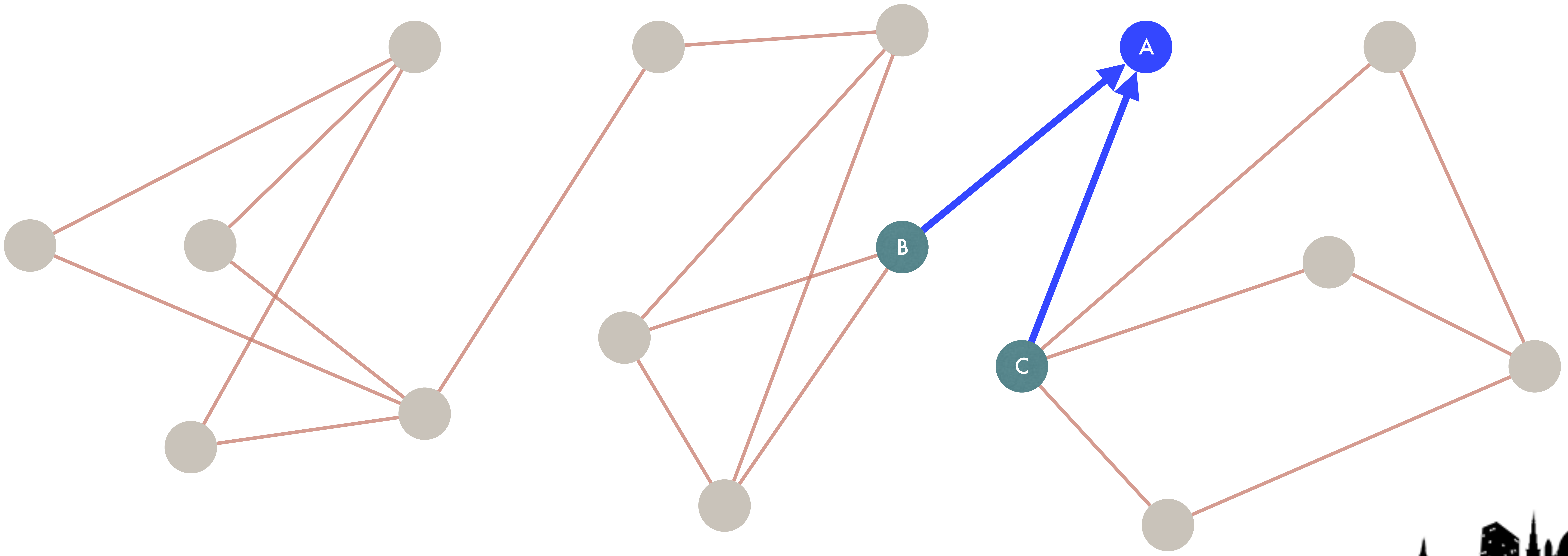
to find context from
the same author

grammar2: Paper -> Venue -> Paper

to find context from
the same conference

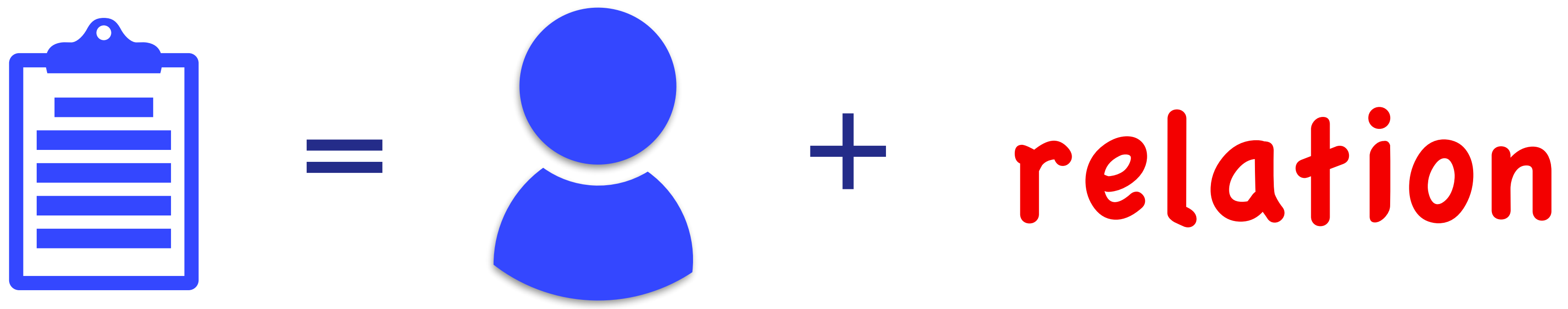
Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami: metapath2vec: Scalable Representation Learning for Heterogeneous Networks. KDD 2017: 135-144

4. Inductiveness

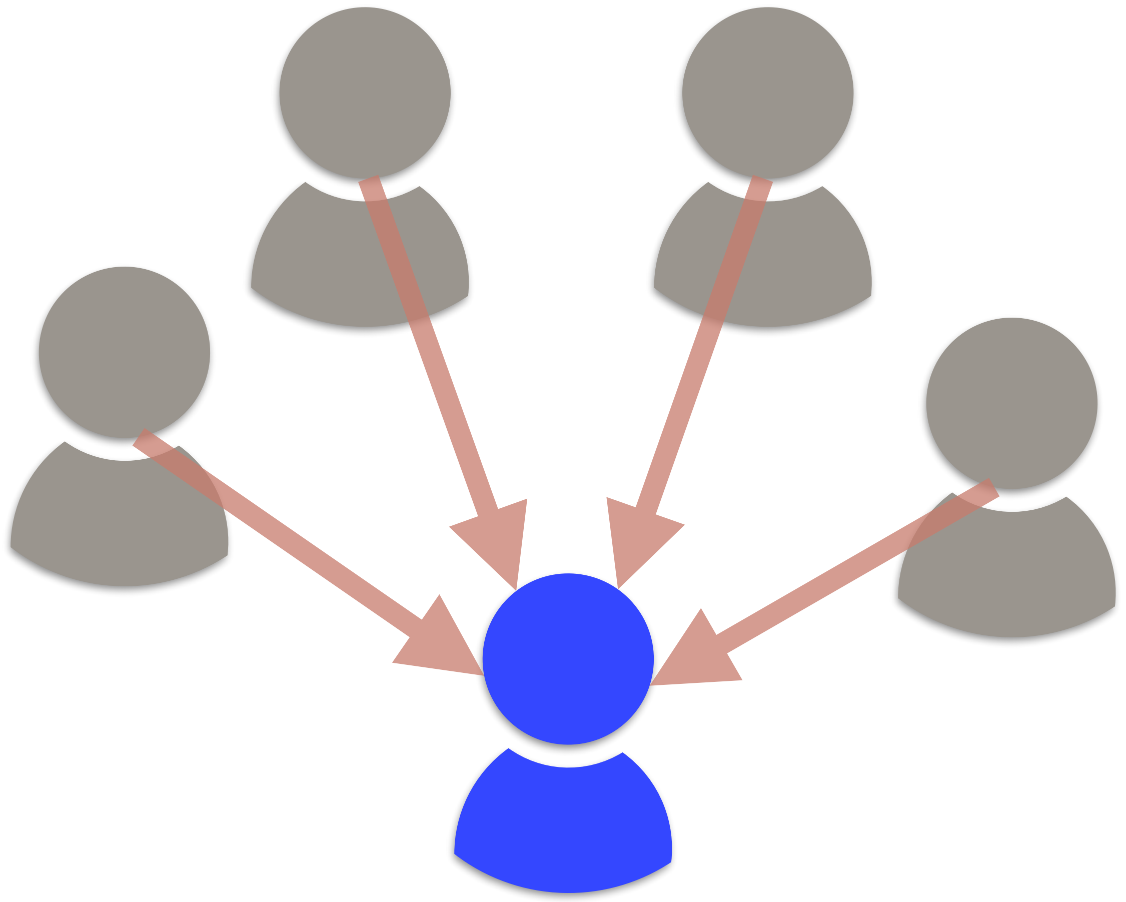


Inductiveness

1. vertex = another vertex + relation

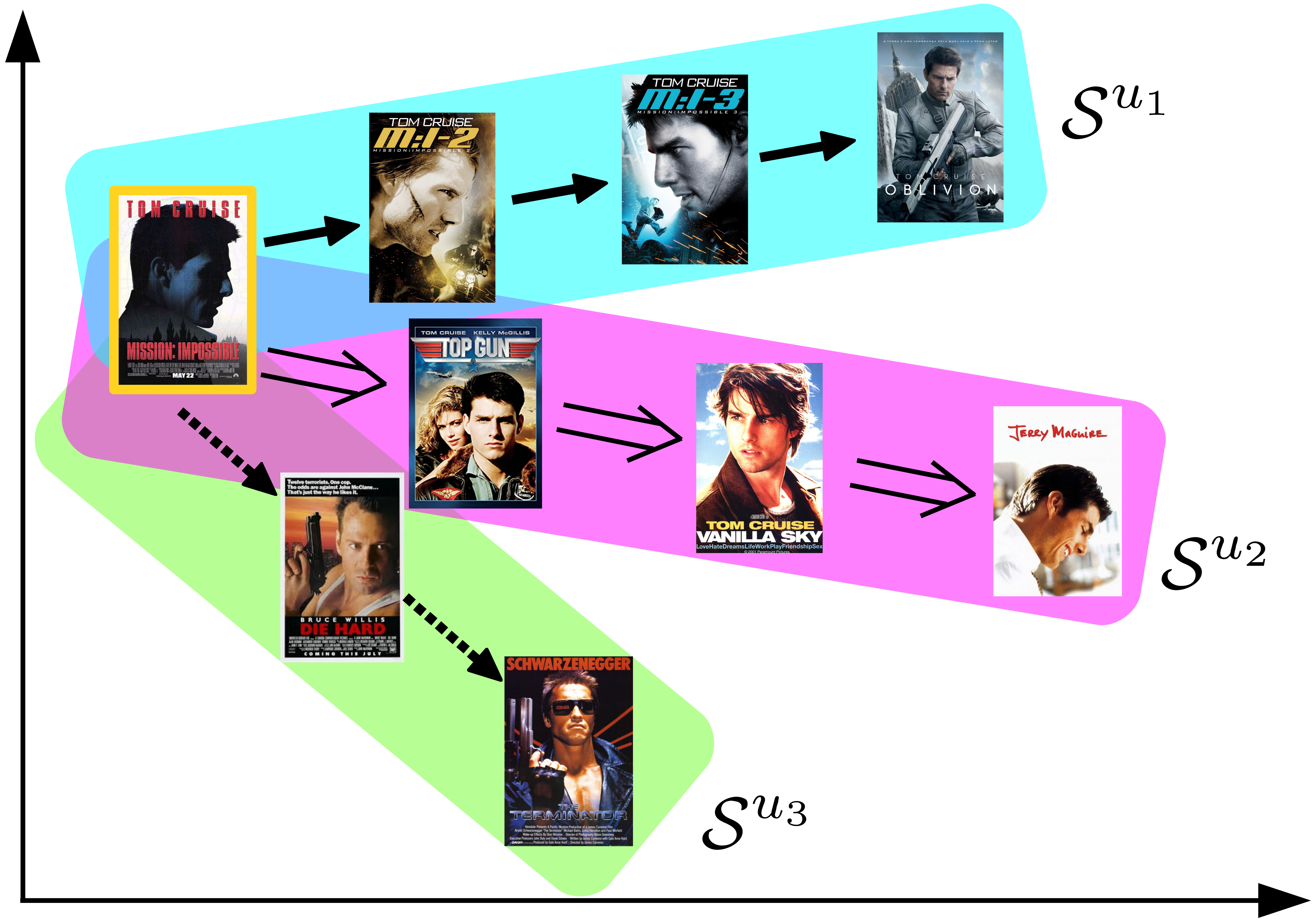


2. vertex = pooling(neighbors)



RECSYS 2019

Translation-based Recommendations (TransRec) *RecSys'18*



$\longrightarrow$	User u_1
$\Longrightarrow$	User u_2
$\dashrightarrow$	User u_3

Translation operation:

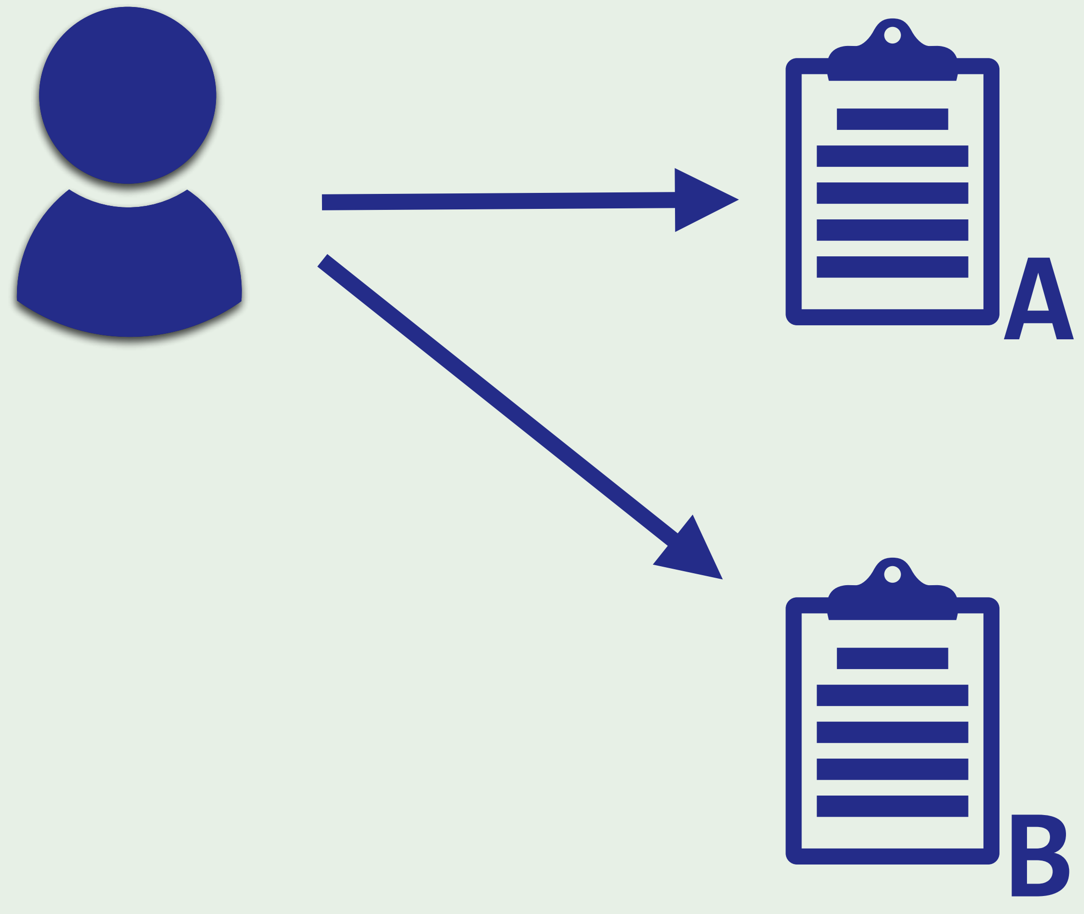
$$\overrightarrow{\textit{prev. item}} + \overrightarrow{\textit{user}} \approx \overrightarrow{\textit{next item}}$$

Rajiv Pasricha, Julian J. McAuley: Translation-based factorization machines for sequential recommendation. RecSys 2018: 63-71

RECSYS 2019

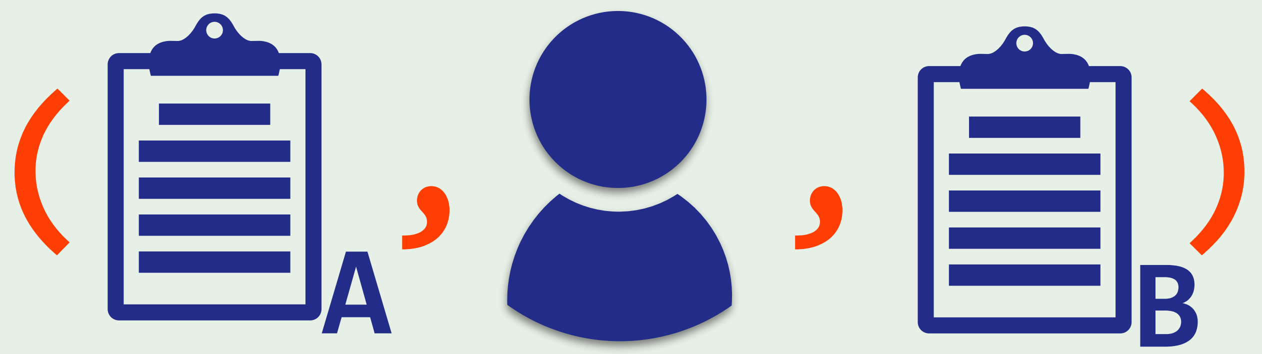
TransRec in SMORe

Graph



user-item graph

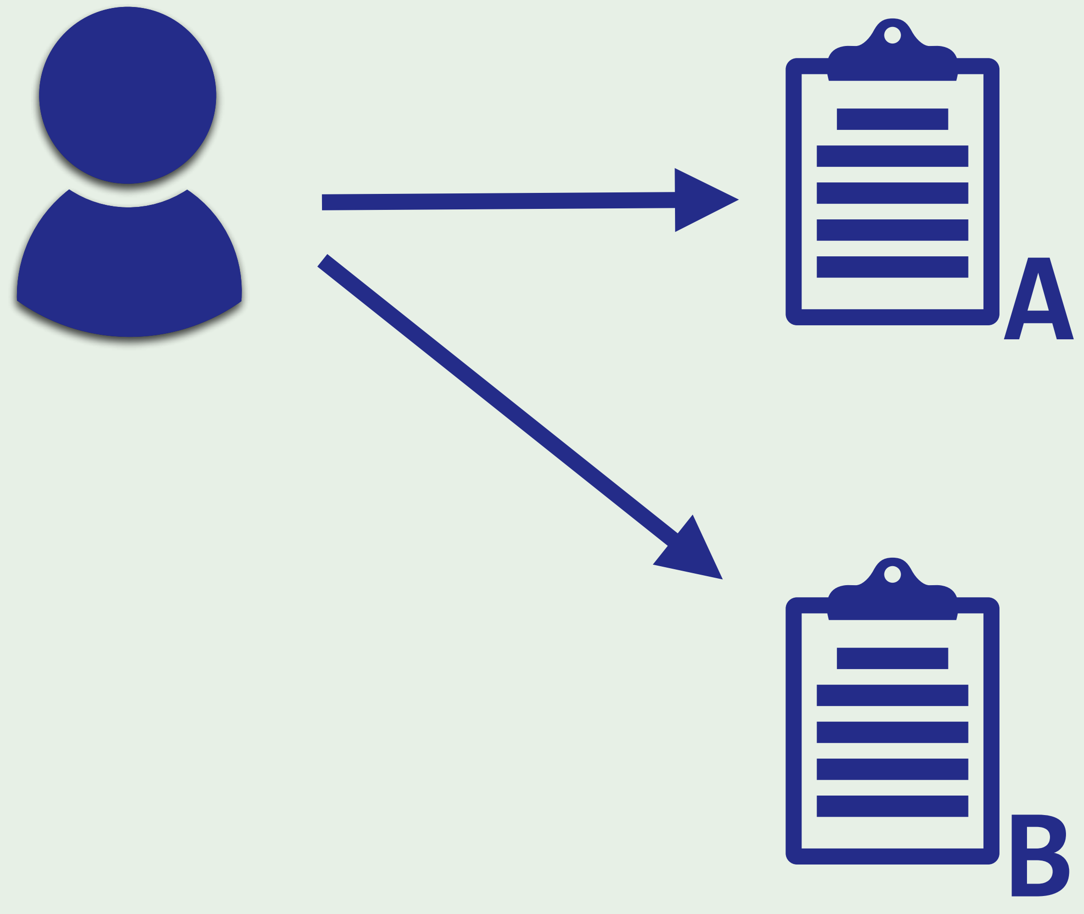
Sampler



(itemA, user, itemB)

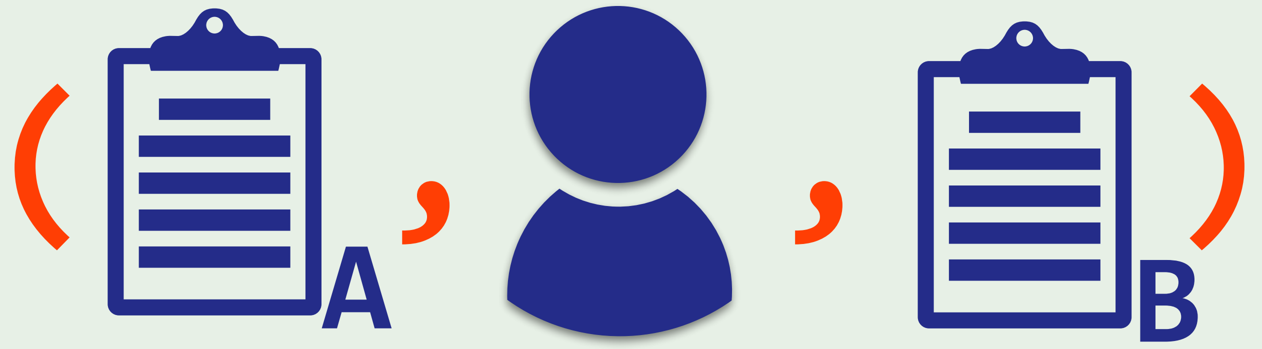
TransRec in SMORe

Graph



user-item graph

Sampler



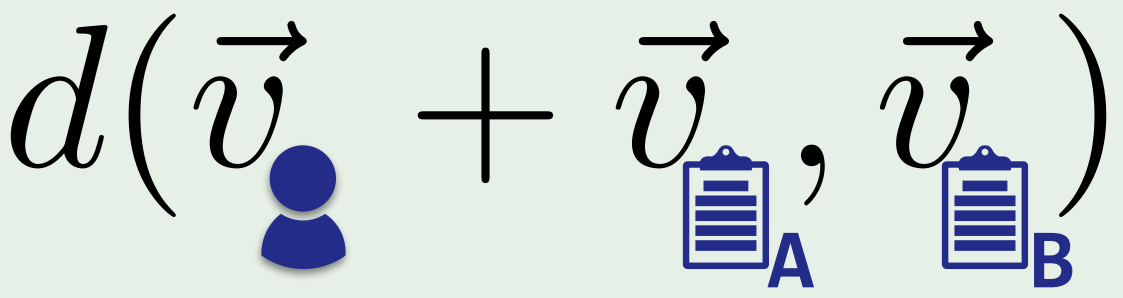
(itemA, user, itemB)

Mapper

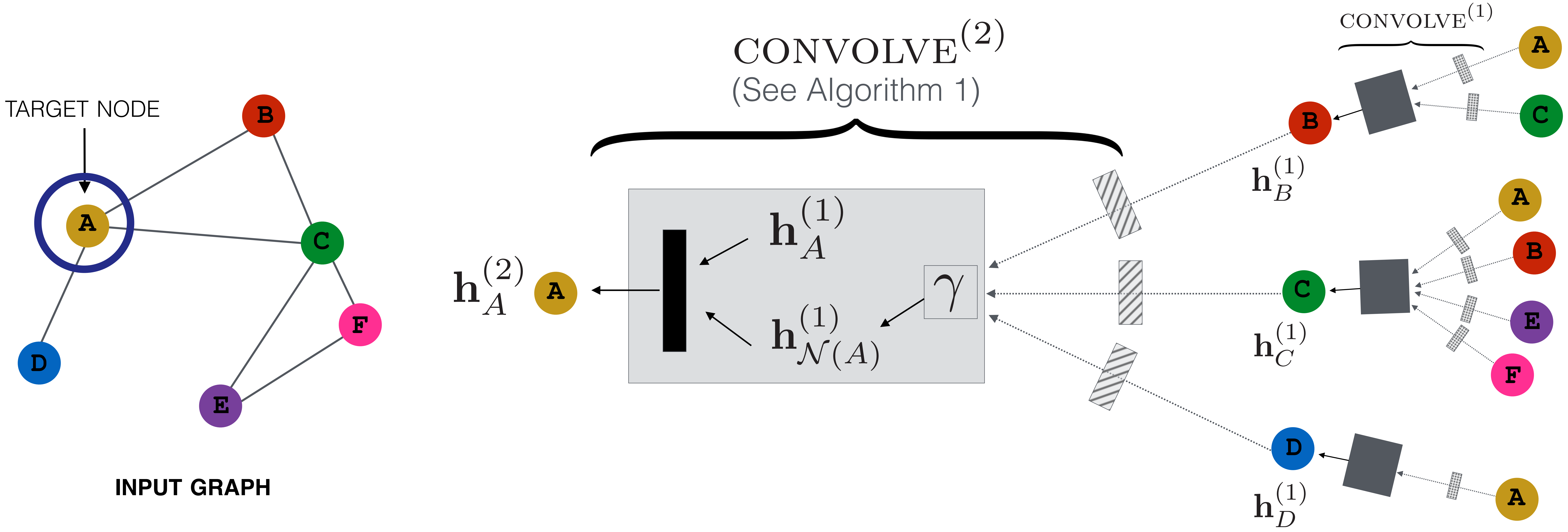
Embedding LookUp

Optimizer

minimize
the distance

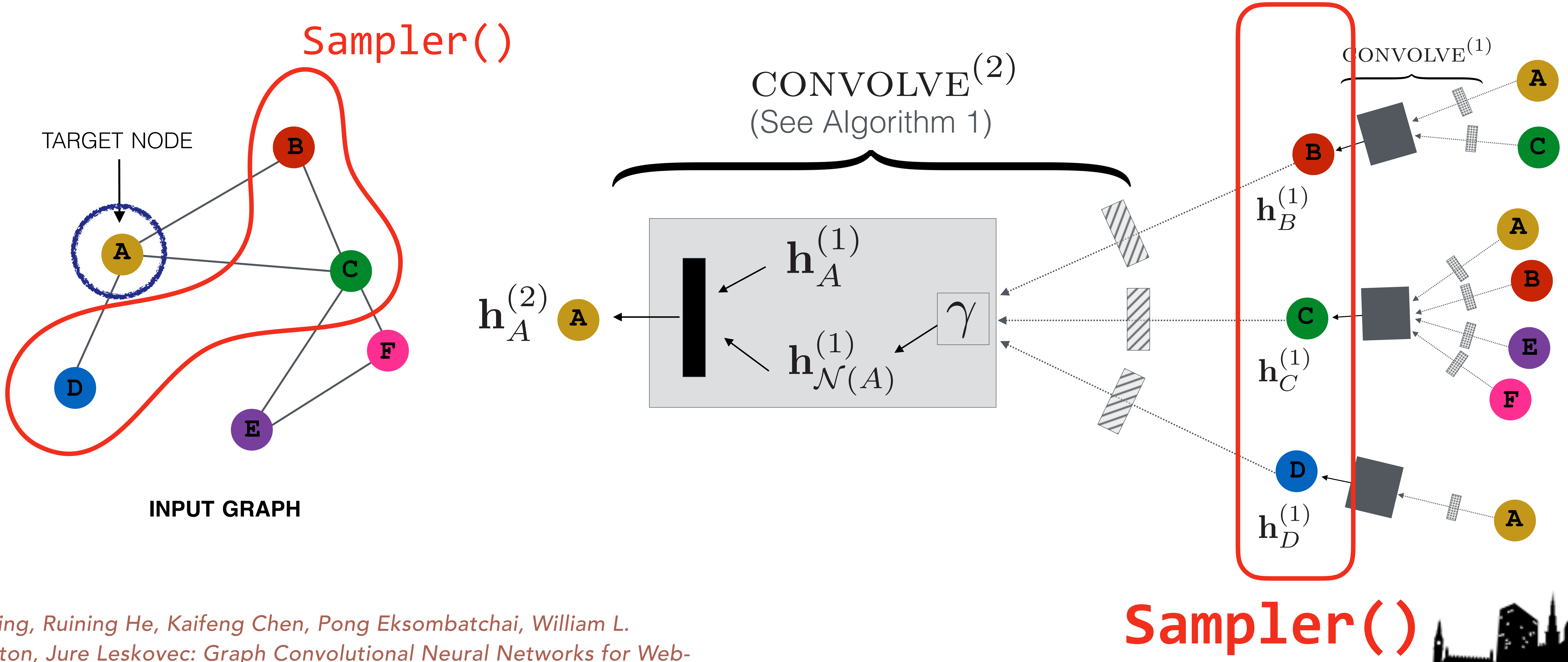


Graph Convolution for Recommendation (Pinsage) *KDD'18*



Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec: Graph Convolutional Neural Networks for Web-Scale Recommender Systems. KDD 2018: 974-983

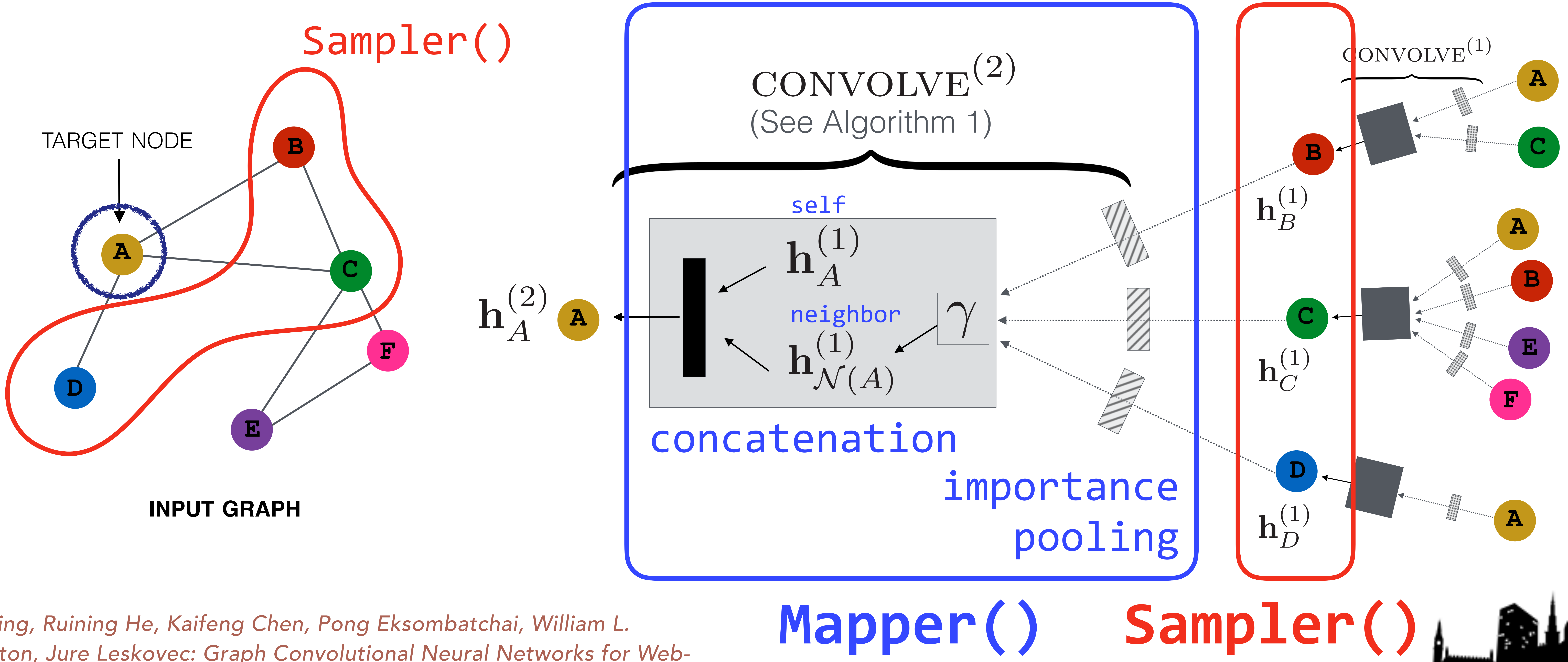
Graph Convolution for Recommendation (Pinsage) *KDD'18*



Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec: Graph Convolutional Neural Networks for Web-Scale Recommender Systems. KDD 2018: 974-983

RECSYS 2019

Graph Convolution for Recommendation (Pinsage) *KDD'18*



Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec: Graph Convolutional Neural Networks for Web-Scale Recommender Systems. KDD 2018: 974-983

Recap

	Sampler	Mapper	Optimizer
node2vec	Community (Weight)	Vertex Embedding Context Embedding	Sigmoid Dot Product Log Likelihood
HOP-Rec	Community (Closeness)	Vertex Embedding	Dot Product BPR
metapath2vec	Community (Semantics)	Vertex Embedding Context Embedding	Sigmoid Dot Product Log Likelihood
TransRec	Adjacency (Inductiveness)	Vertex Embedding	Dot product BPR
Pinsage	Adjacency (Inductiveness)	Vertex / Context Embedding Pooling + Concatenation	Fully Connected Layer

Lecture Agenda

(Sean) **Q0. Recommendation** (REC) and challenges

(Sean) **Q1.** Why **graph embedding** (GE) for REC

(Sean) **Q2. SMORe** modularization of GE and benefits

(CM) **Q3.** Exemplar **structural modeling** for REC

(CM) **Q4.** REC using **SMORe**

RECSYS 2019

Ting-Hsiang Wang, Texas A&M University

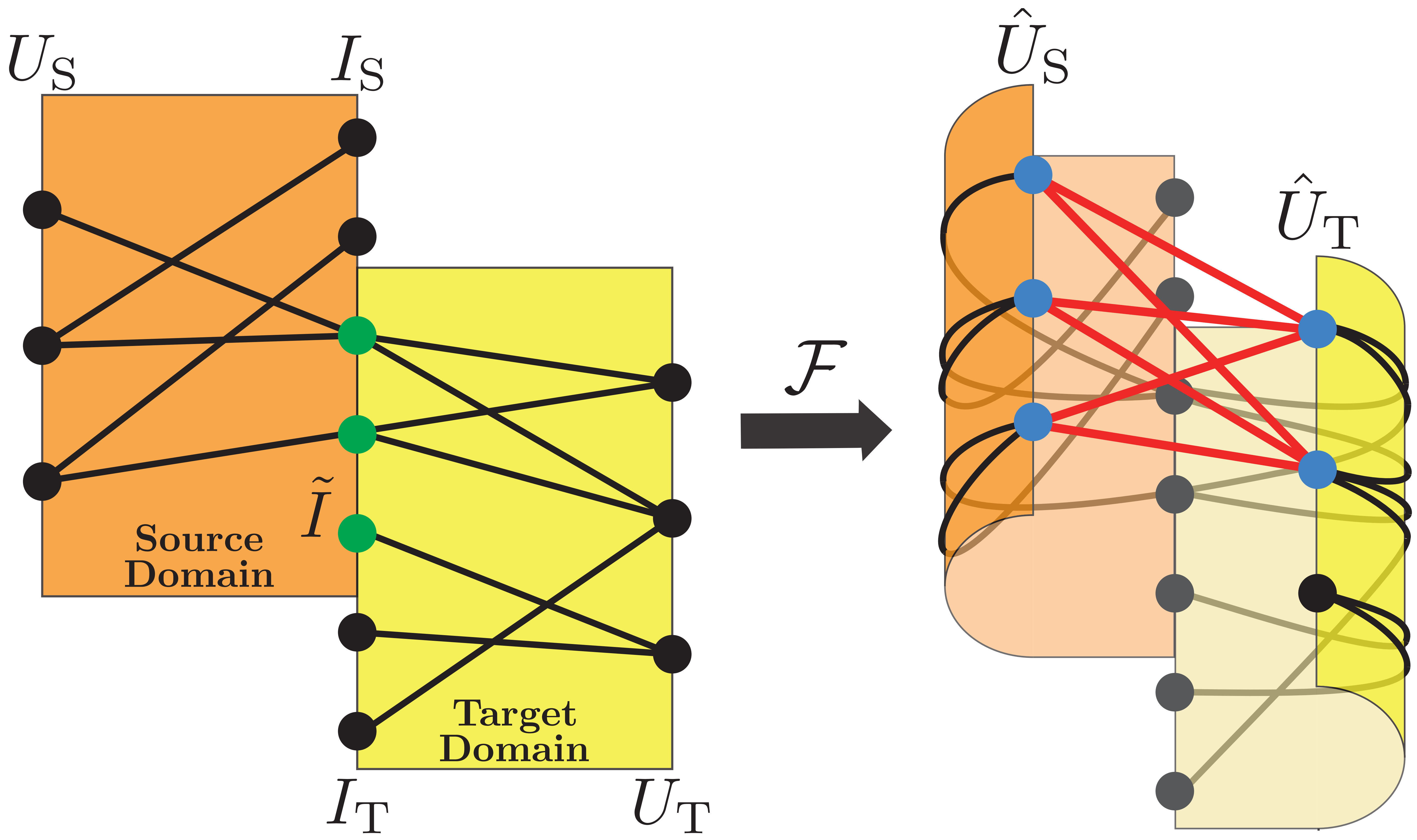
Graph Manipulation

- Connectivity
  - modify the edge weights
- Augmentation
  - extend the connections from external knowledge

RECSYS 2019

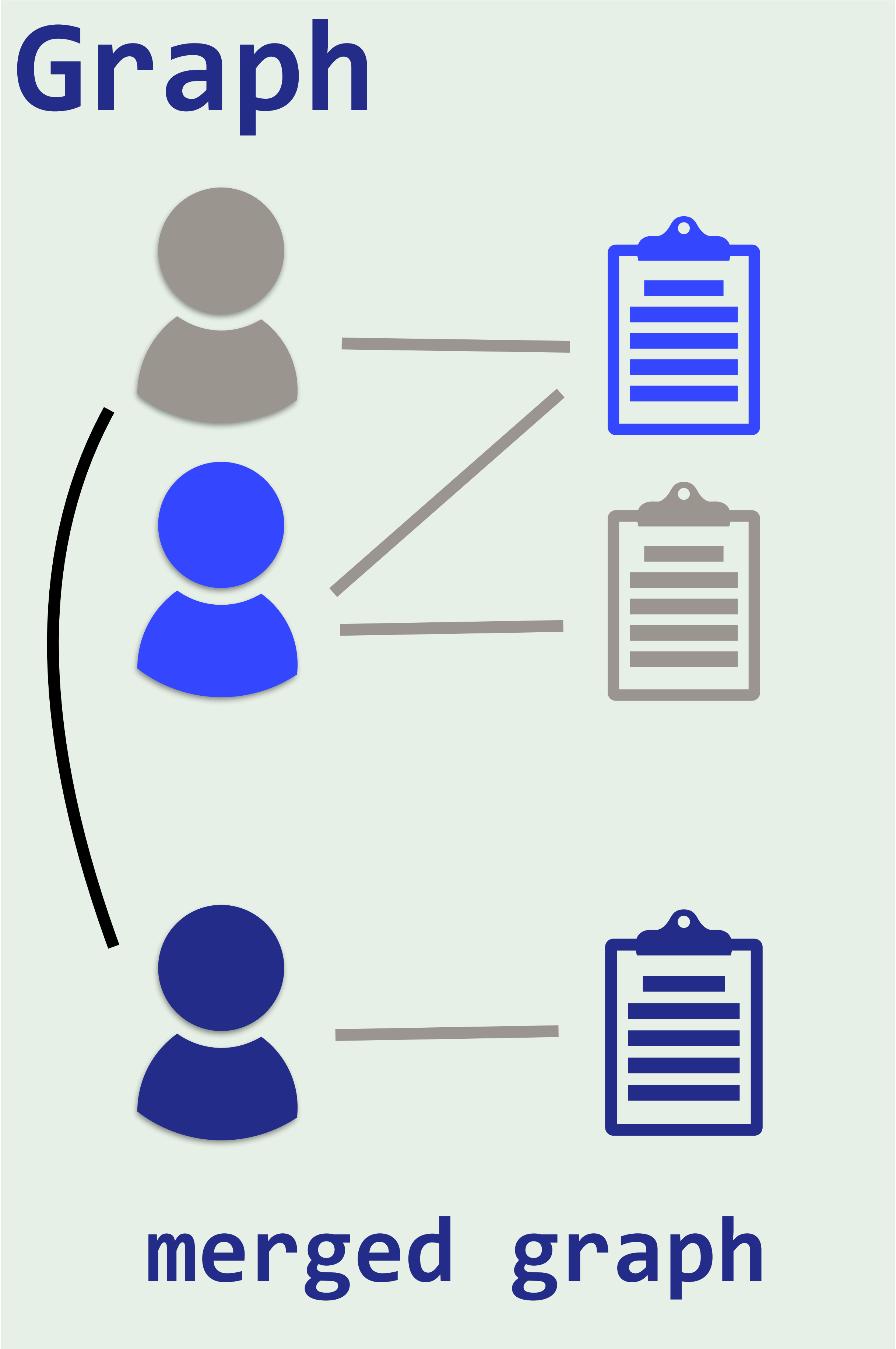
Superhighway for Cross-Domain CF

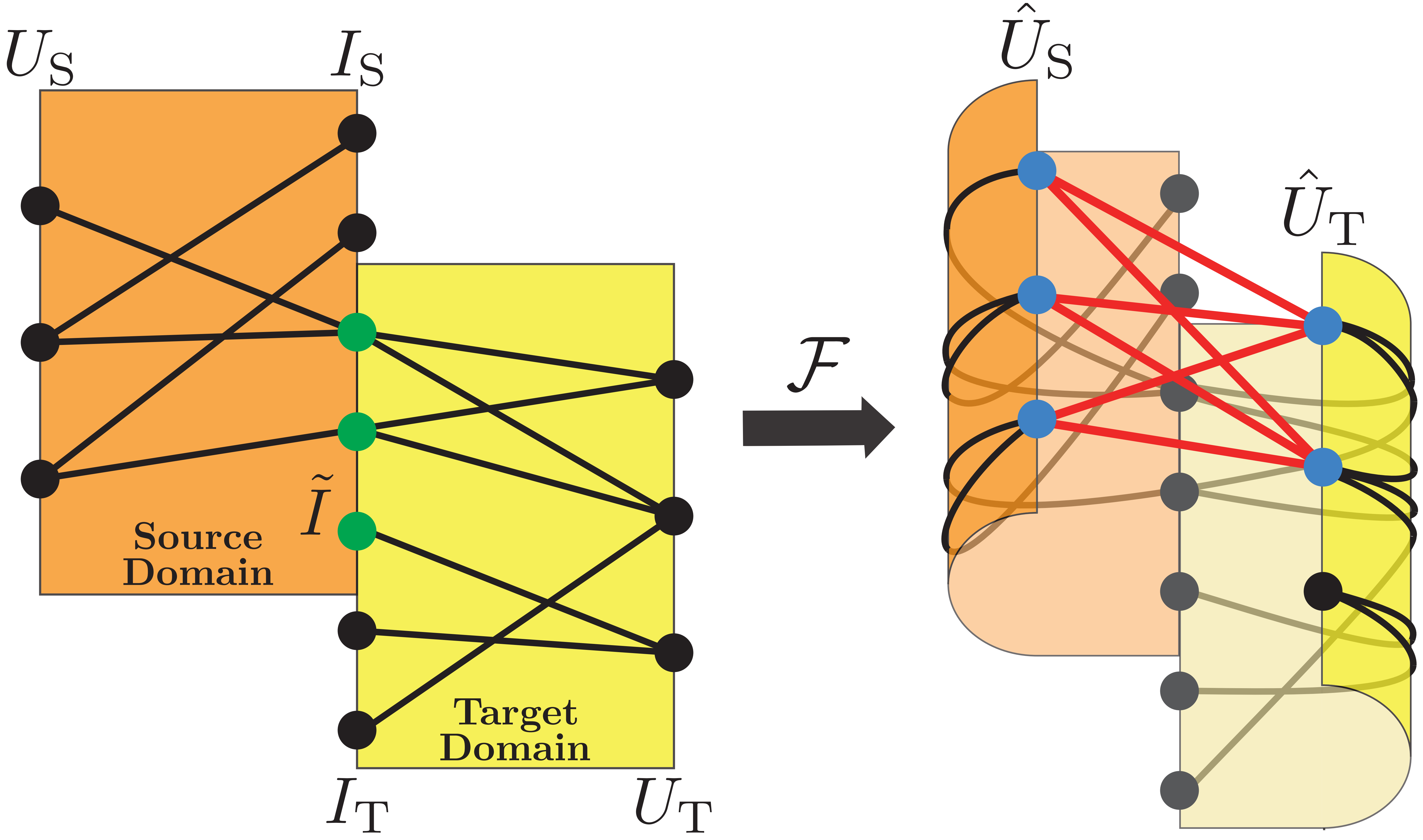
RecSys'18



Kwei-Herng Lai, Ting-Hsiang Wang, Heng-Yu Chi, Yian Chen, Ming-Feng Tsai, Chuan-Ju Wang: Superhighway: Bypass Data Sparsity in Cross-Domain CF. CoRR abs/1808.09784 (2018)

Superhighway in **SMORe**

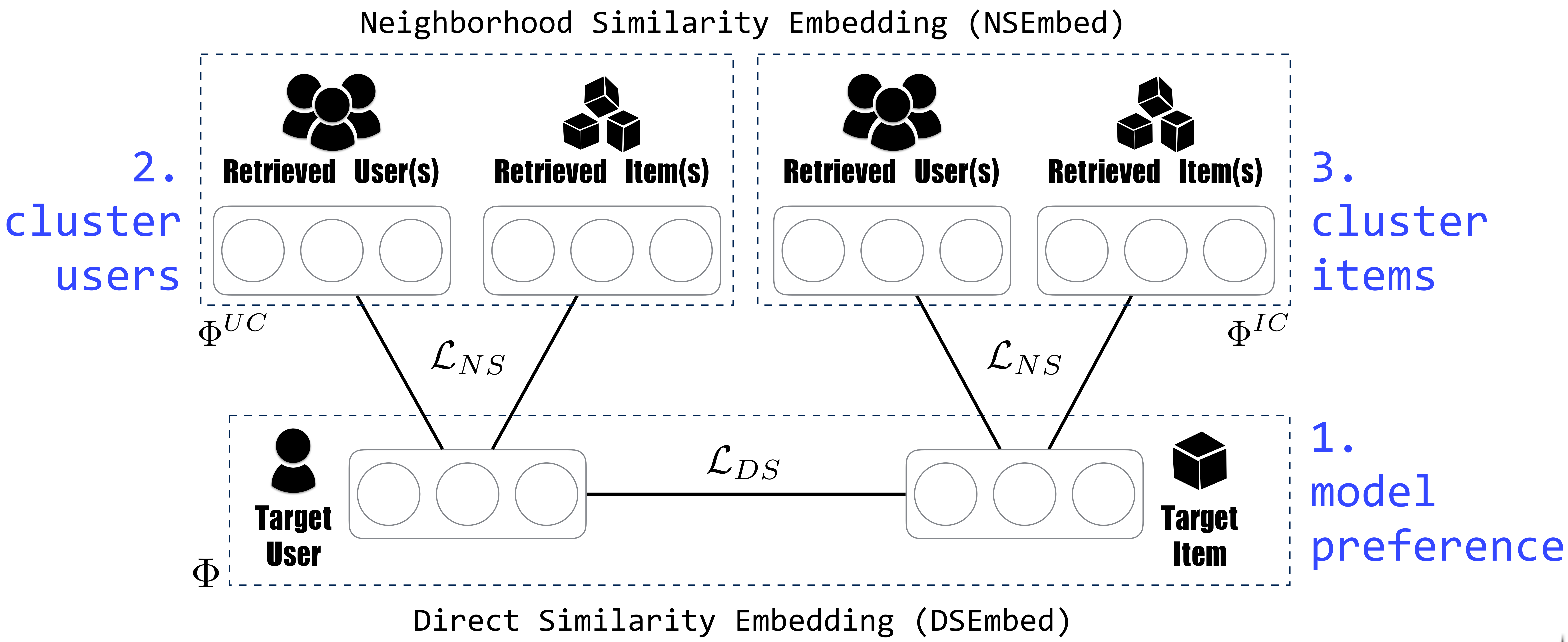




- Multi-Task Learning
  - shared representations for multi-tasks

Collaborative Similarity Embedding (CSE)

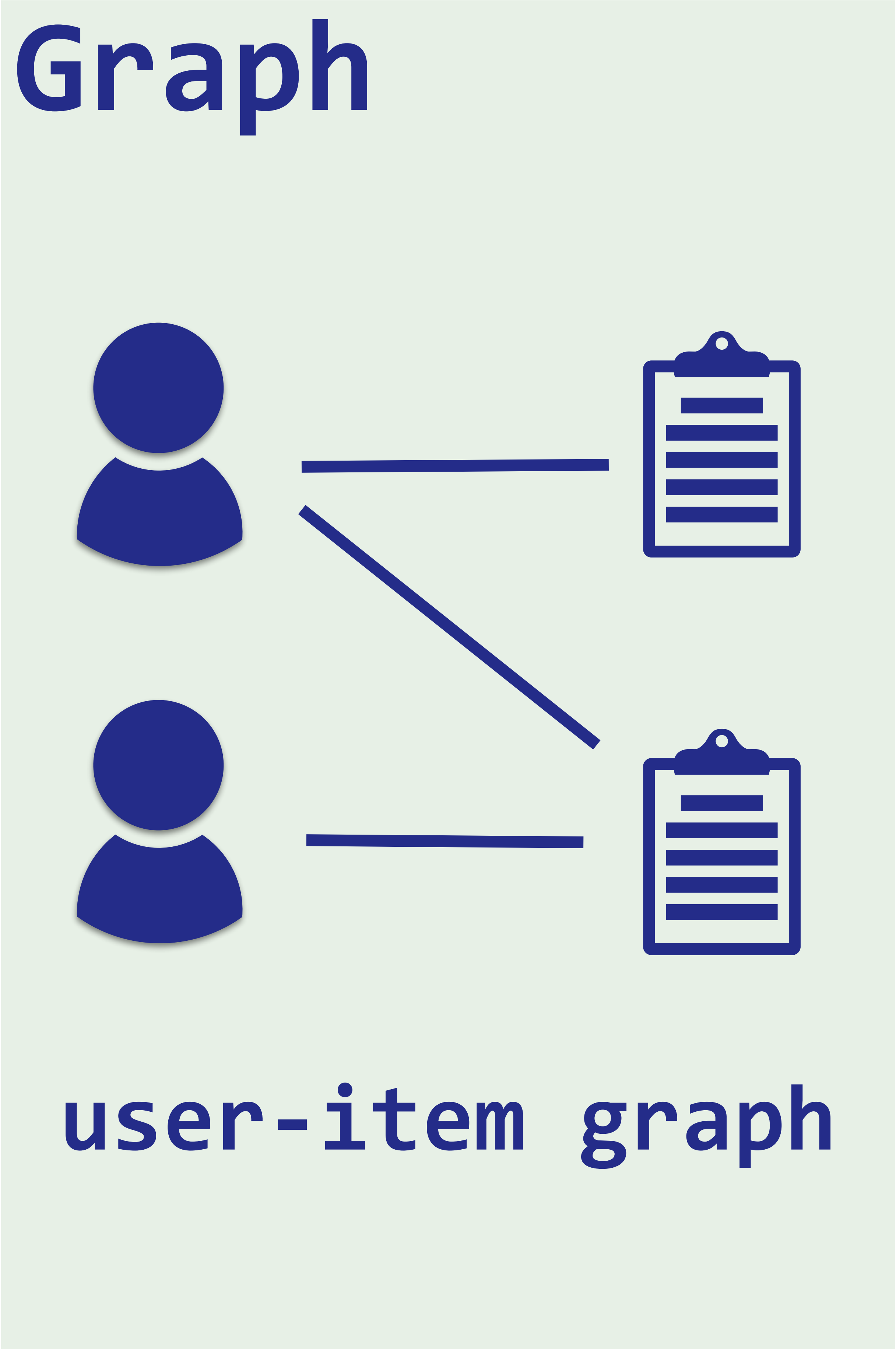
WWW'19



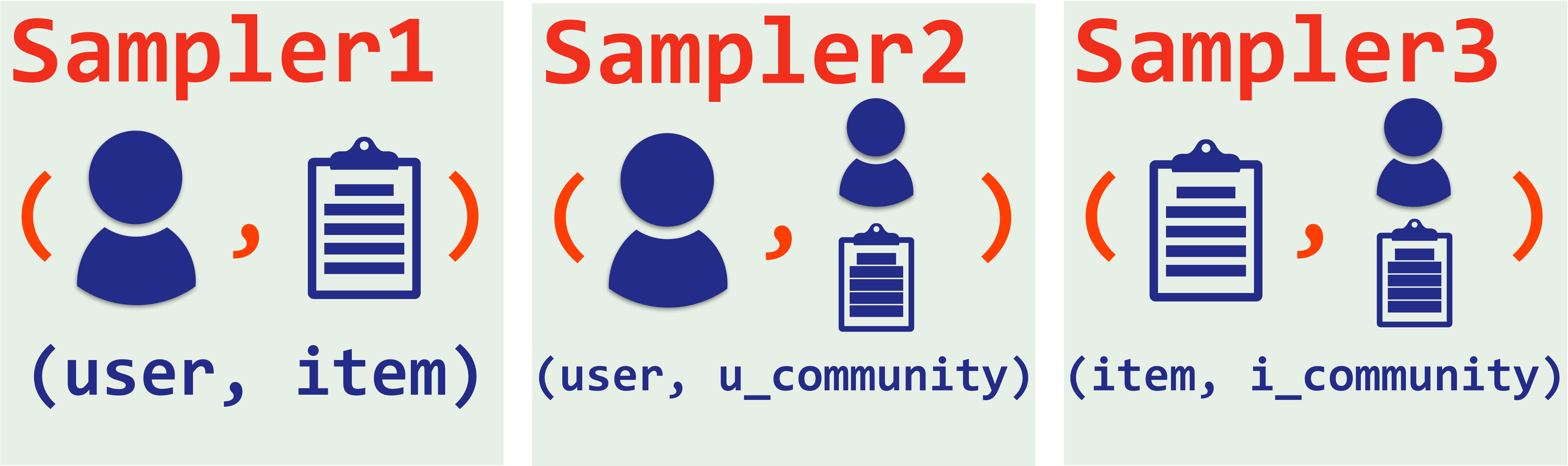
Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, Yi-Hsuan Yang: Collaborative Similarity Embedding for Recommender Systems. WWW 2019: 2637-2643

RECSYS 2019

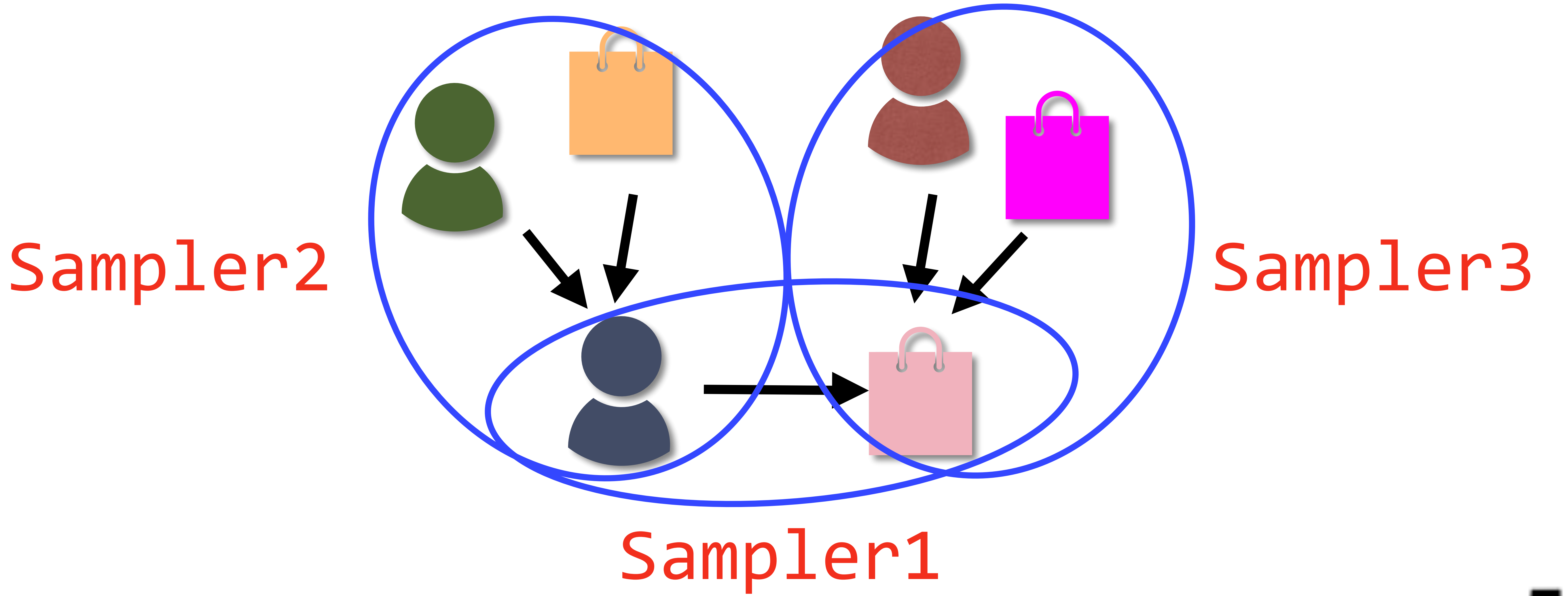
CSE in SMORe



Graph
user-item graph

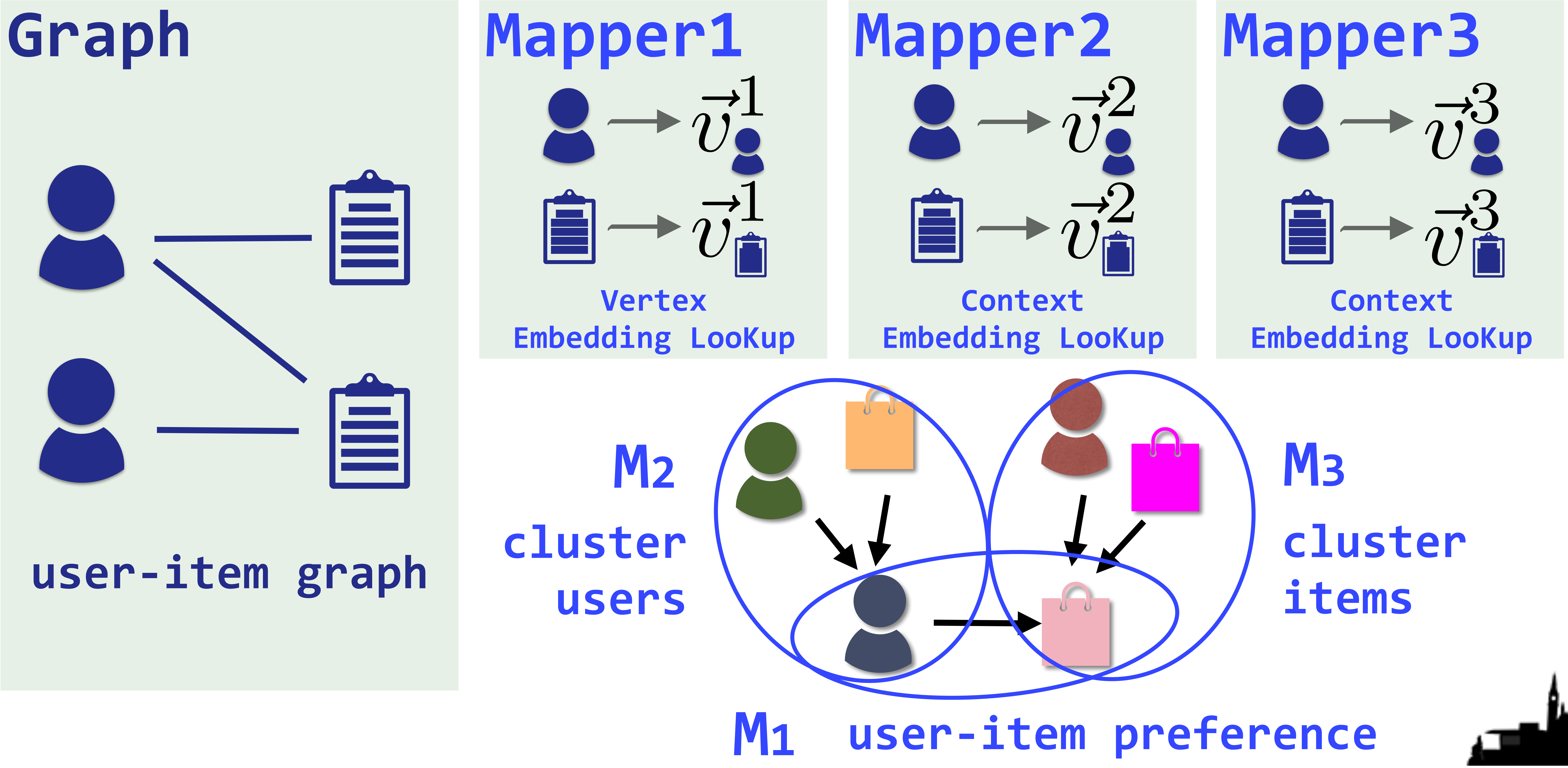


Sampler1
(user, item)
Sampler2
(user, u_community)
Sampler3
(item, i_community)



Sampler2
Sampler3
Sampler1

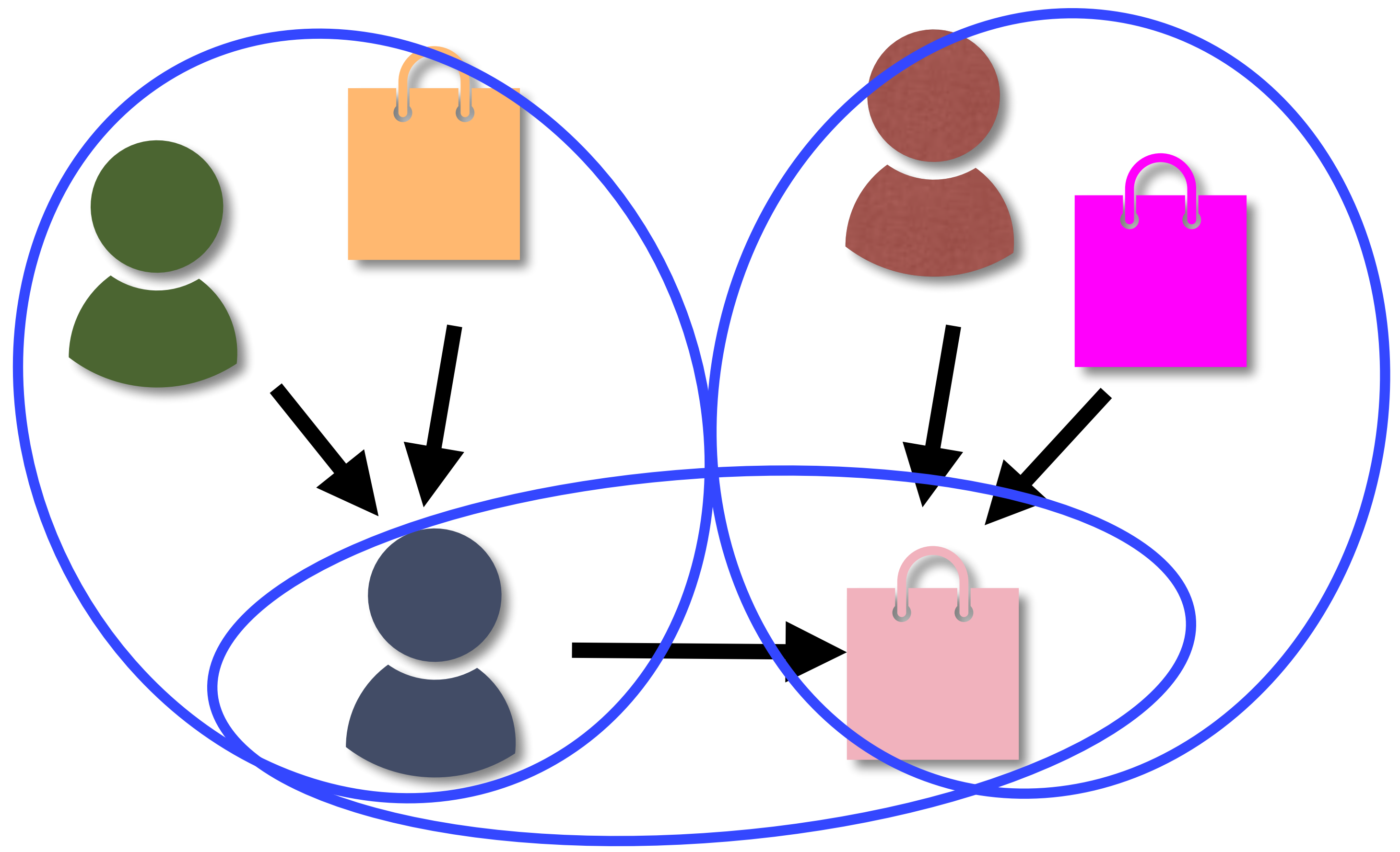
CSE in SMORe



Graph
user-item graph
Mapper1
Vertex
Embedding LooKup
Mapper2
Context
Embedding LooKup
Mapper3
Context
Embedding LooKup
M2
cluster
users
M3
cluster
items
M1 user-item preference

CSE in SMORe

Optimizer1		Optimizer2		Optimizer3
$\log \sigma(\vec{v}^1 \cdot \vec{v}^1)$	$+$	$\log \sigma(\vec{v}^1 \cdot \vec{v}^2)$	$+$	$\log \sigma(\vec{v}^1 \cdot \vec{v}^3)$
Log Likelihood		Log Likelihood		Log Likelihood



Coming Next ...

Lecture (Sean & CM, 65 minutes)

Hands-on (CM, 15 minutes)

Q&A (Sean & CM, 10 minutes)

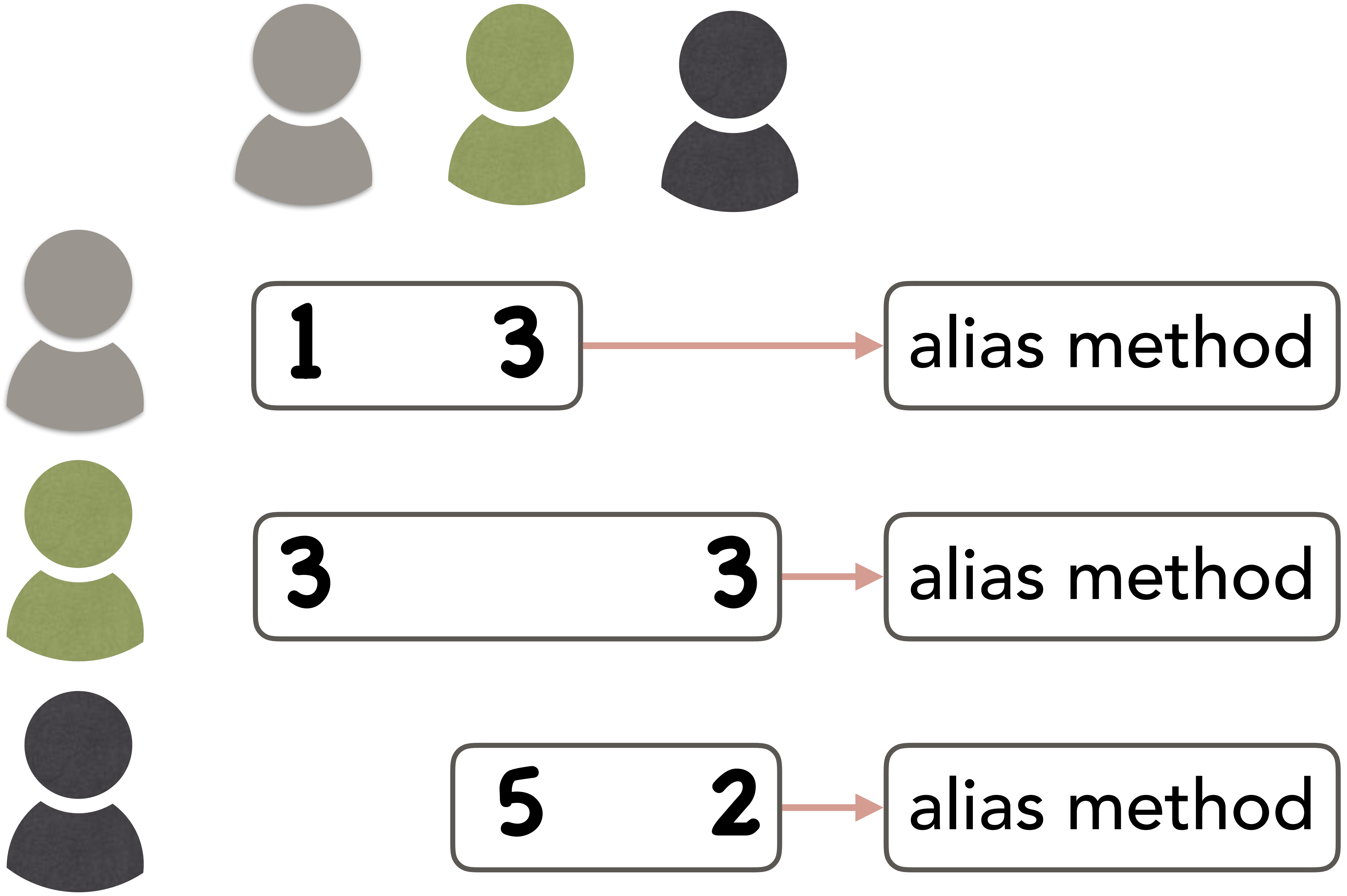
QR to Slides, Codes, Abstract

Ting-Hsiang Wang, Texas A&M University

RECSYS 2019

SMORe

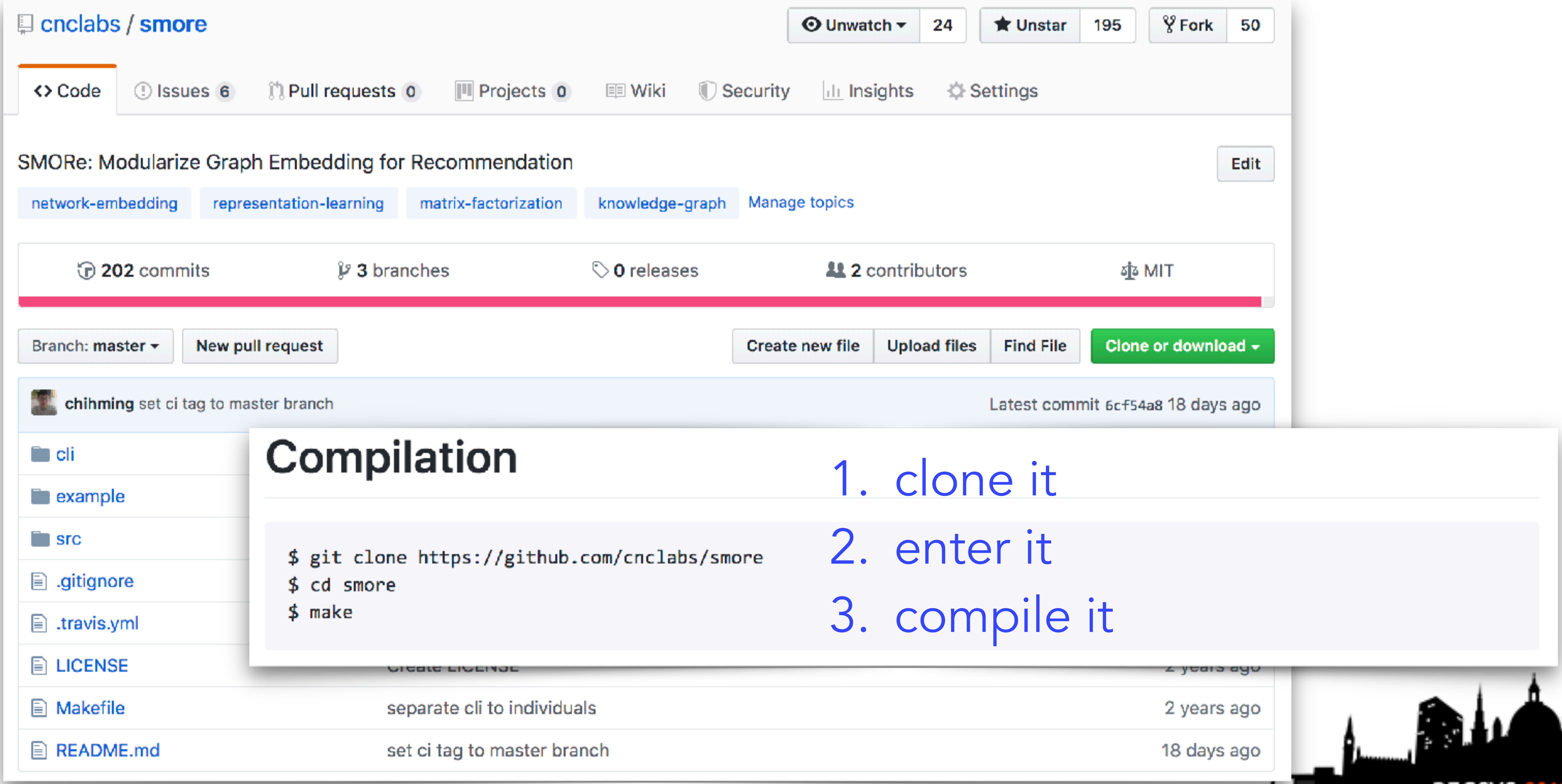
https://github.com/cnclabs/smore



Compressed Sparse Row (CSR)
+ Alias Method

RECSYS 2019

SMORe



cnclabs / smore

Unwatch 24 | Unstar 195 | Fork 50

Code | Issues 6 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

SMORe: Modularize Graph Embedding for Recommendation

Edit

network-embedding | representation-learning | matrix-factorization | knowledge-graph | Manage topics

202 commits | 3 branches | 0 releases | 2 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

chihming set ci tag to master branch — Latest commit 6cf54a8 18 days ago

cli		
example		
src		
.gitignore		
.travis.yml		
LICENSE	Create LICENSE	2 years ago
Makefile	separate cli to individuals	2 years ago
README.md	set ci tag to master branch	18 days ago

Compilation

```
$ git clone https://github.com/cnclabs/smore
$ cd smore
$ make
```

1. clone it
2. enter it
3. compile it

RECSYS 2019

SMORe

- DeepWalk
  - DeepWalk: online learning of social representations
- Walklets
  - Don't Walk, Skip! Online Learning of Multi-scale Network Embeddings
- LINE(**L**arge-scale **I**nformation **N**etwork **E**mbedding)
  - LINE: Large-scale Information Network Embedding
- HPE (**H**eterogeneous **P**reference **E**mbedding)
  - Query-based Music Recommendations via Preference Embedding
- APP (**A**symmetric **P**roximity **P**reserving graph embedding)
  - Scalable Graph Embedding for Asymmetric Proximity
- MF (**M**atrix **F**actorization)
- BPR (**B**ayesian **P**ersonalized **R**anking)
  - BPR: Bayesian personalized ranking from implicit feedback
- WARP-like
  - WSABIE: Scaling Up To Large Vocabulary Image Annotation
  - Learning to Rank Recommendations with the k-Order Statistic Loss
- HOP-REC
  - HOP-Rec: High-Order Proximity for Implicit Recommendation
- CSE (named nemf & nerank in cli)
  - Collaborative Similarity Embedding for Recommender Systems

RECSYS 2019

SMORe for Most End Users

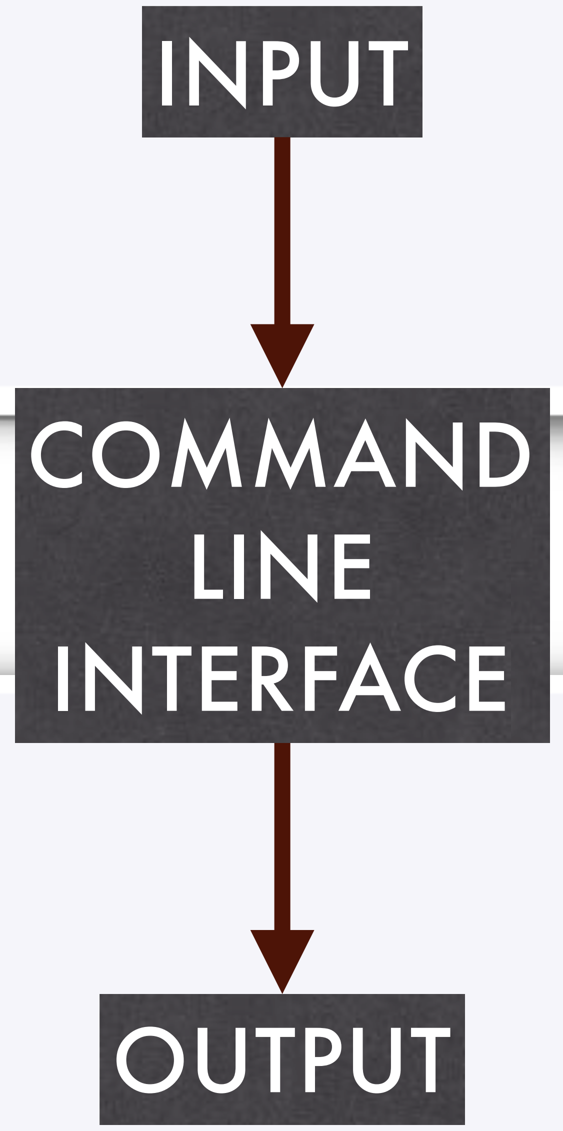
```
Options Description:
        -train <string>
                Train the Network data
        -save <string>
                Save the representation data
        -dimensions <int>
                Dimension of vertex representation; default is 64
        -undirected <int>
                Whether the edge is undirected; default is 1
        -negative_samples <int>
                Number of negative examples; default is 5
        -window_size <int>
                Size of skip-gram window; default is 5
        -walk_times <int>
                Times of being staring vertex; default is 10
        -walk_steps <int>
                Step of random walk; default is 40
        -threads <int>
                Number of training threads; default is 1
        -alpha <float>
                Init learning rate; default is 0.025
Usage:
./deepwalk -train net.txt -save rep.txt -undirected 1 -dimensions 64 -walk_times 10 -walk_steps 40 -window_size 5 -n
```

RECSYS 2019

SMORe for Most End Users

Graph as input (edge list)

```
userA itemA 3
userA itemC 5
userB itemA 1
userB itemB 5
userC itemA 4
```



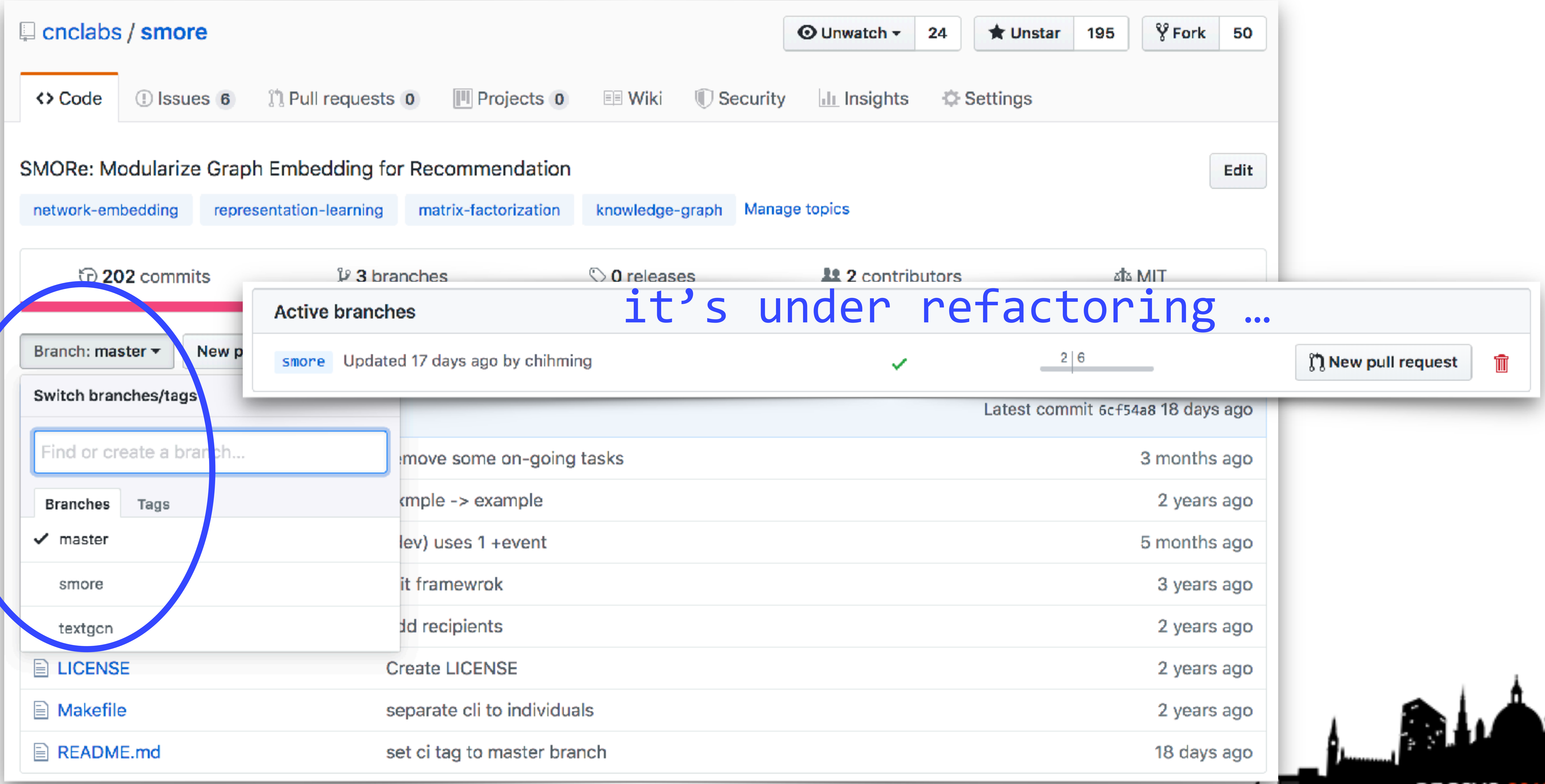
Embeddings as output

```
6 5
userA 0.0815412 0.0205459 0.288714 0.296497 0.394043
itemA -0.207083 -0.258583 0.233185 0.0959801 0.258183
itemC 0.0185886 0.138003 0.213609 0.276383 0.45732
userB -0.0137994 -0.227462 0.103224 -0.456051 0.389858
itemB -0.317921 -0.163652 0.103891 -0.449869 0.318225
userC -0.156576 -0.3505 0.213454 0.10476 0.259673
```

RECSYS 2019

- On-Going Work

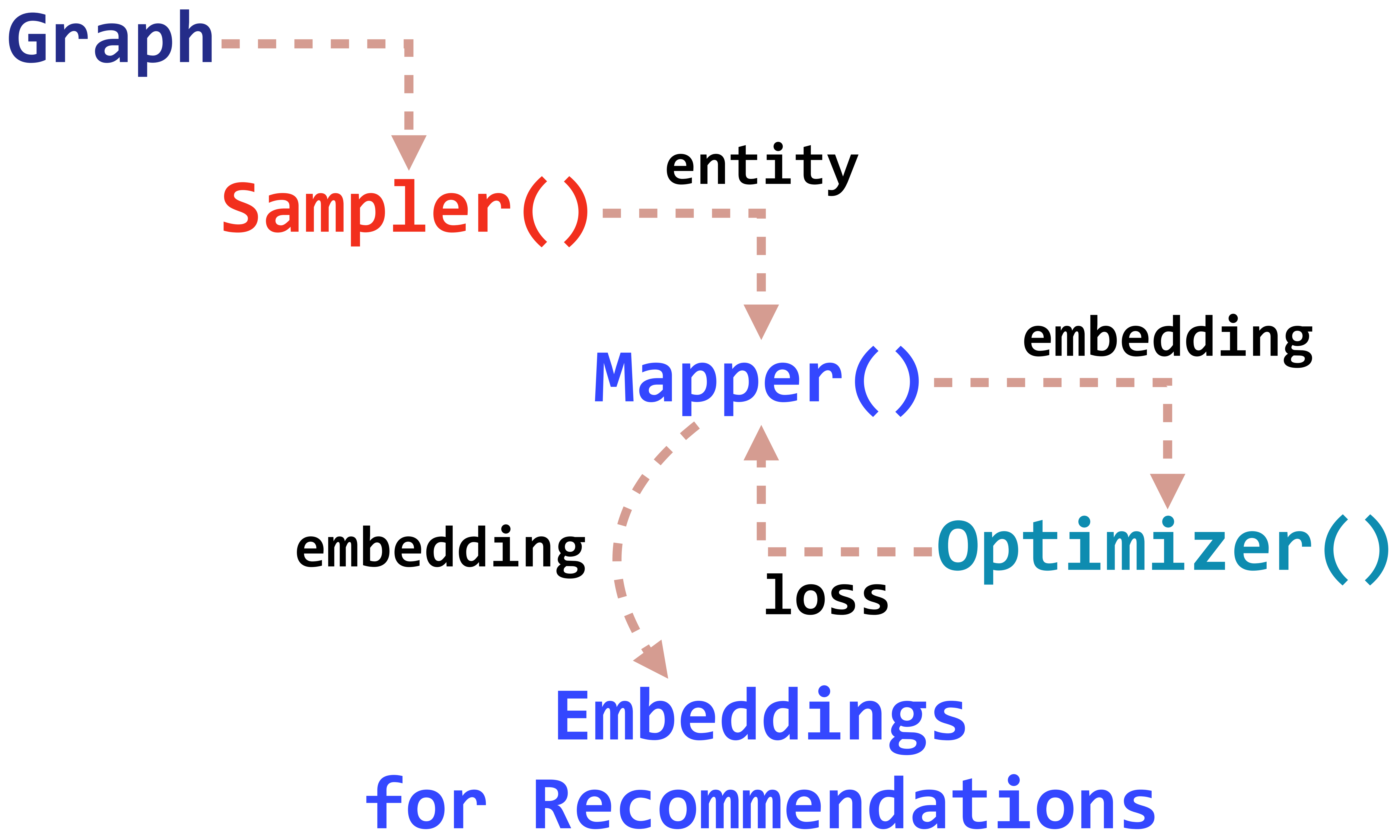
SMORe (another modularized version)



find
the
branch
smore

RECSYS 2019

SMORe for Developers



SMORe Example Codes (in smore branch)

```
// main
// 0. [Graph] read from file-based graph
FileGraph *file_graph = new FileGraph(path, 0);

// 1. [Sampler] determine what sampler to be used
VCSampler sampler(file_graph);

// 2. [Mapper] define what embedding mapper to be used
LookupMapper mapper(sampler.vertex_size, dimension);

// 3. [Optimizer] claim the optimizer
PairwiseOptimizer optimizer;
```

Graph()

Sampler()

Mapper()

Optimizer()

RECSYS 2019

SMORe Factorization (in smore branch)

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

RECSYS 2019

SMORe Factorization (in smore branch)

run it until conditions hold

```
            while (update < worker_update_times)
            {
                // 4.0 reset user batch loss
                user_batch_loss.assign(dimension, 0.0);
                item_loss.assign(dimension, 0.0);

                // 4.1 sample positive (user, item) pair, feed the loss, update
                user = sampler.draw_a_vertex();
                item = sampler.draw_a_context(user);
                optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
                mapper.update_with_l2(item, item_loss, alpha, 0.001);
                item_loss.assign(dimension, 0.0);

                // 4.2 sampler negative (user, item) pair, feed the loss, update
                for (int n=0; n<negative; n++)
                {
                    item = sampler.draw_a_negative();
                    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
                    mapper.update_with_l2(item, item_loss, alpha, 0.001);
                    item_loss.assign(dimension, 0.0);
                }
                mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

SMORe Factorization (in smore branch)

run it until conditions hold

sample positive (user, item) pair

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

RECSYS 2019

SMORe Factorization (in smore branch)

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

run it until conditions hold

sample positive (user, item) pair

map user/item to its embedding

RECSYS 2019

SMORe Factorization (in smore branch)

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

run it until conditions hold

sample positive (user, item) pair

map user/item to its embedding

estimate the loss from log likelihood

RECSYS 2019

SMORe Factorization (in smore branch)

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

run it until conditions hold

sample positive (user, item) pair

map user/item to its embedding

estimate the loss from log likelihood

update embedding

SMORe Factorization (in smore branch)

```
while (update < worker_update_times)
{
    // 4.0 reset user batch loss
    user_batch_loss.assign(dimension, 0.0);
    item_loss.assign(dimension, 0.0);

    // 4.1 sample positive (user, item) pair, feed the loss, update
    user = sampler.draw_a_vertex();
    item = sampler.draw_a_context(user);
    optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 1.0, dimension, user_batch_loss, item_loss);
    mapper.update_with_l2(item, item_loss, alpha, 0.001);
    item_loss.assign(dimension, 0.0);

    // 4.2 sampler negative (user, item) pair, feed the loss, update
    for (int n=0; n<negative; n++)
    {
        item = sampler.draw_a_negative();
        optimizer.feed_loglikelihood_loss(mapper[user], mapper[item], 0.0, dimension, user_batch_loss, item_loss);
        mapper.update_with_l2(item, item_loss, alpha, 0.001);
        item_loss.assign(dimension, 0.0);
    }
    mapper.update_with_l2(user, user_batch_loss, alpha, 0.01);
```

run it until conditions hold

sample positive (user, item) pair

map user/item to its embedding

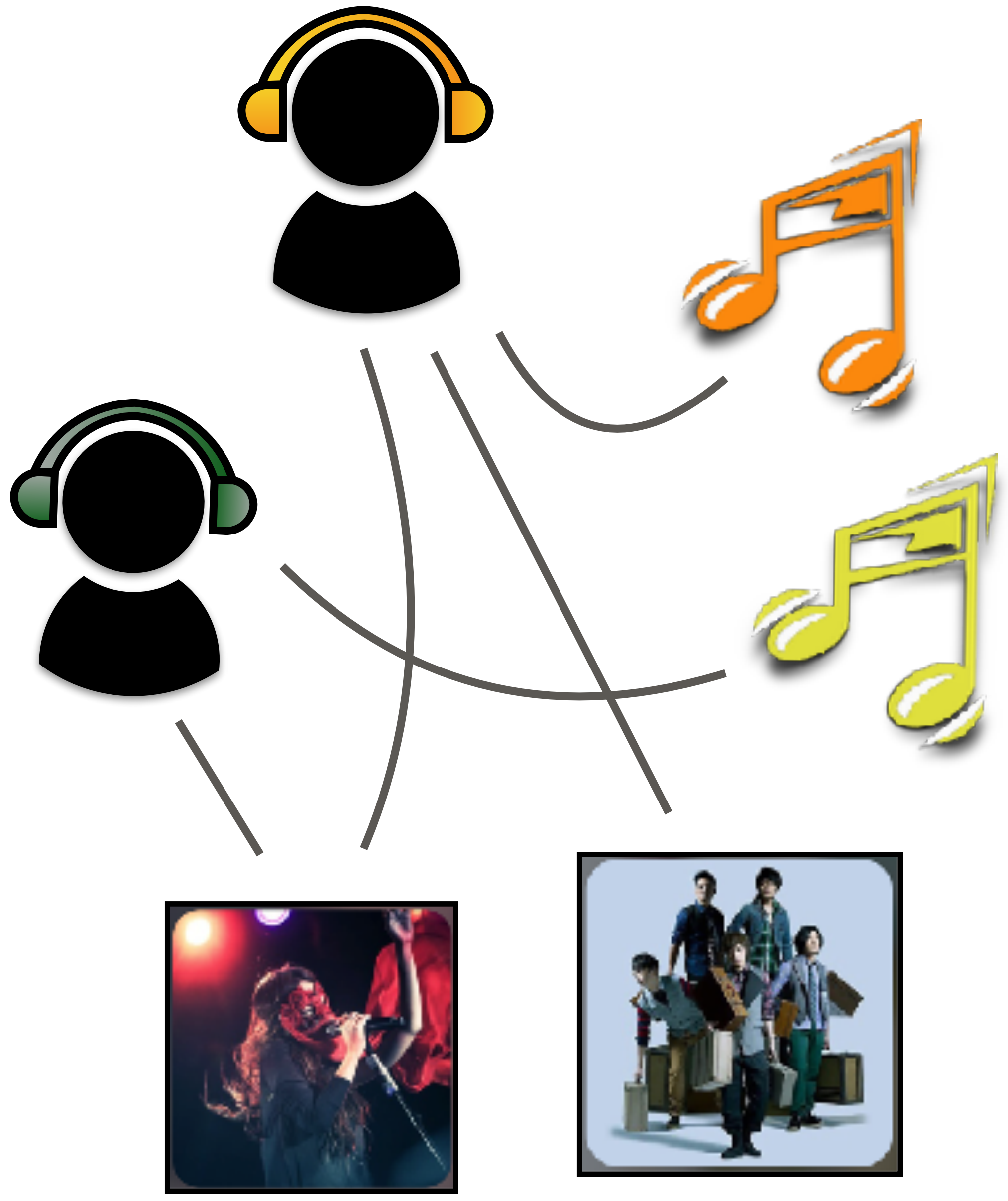
estimate the loss from log likelihood

update embedding

SMO

mode negative (user, item) pair

RECSYS 2019



Graph

Sampler()

Mapper()

Optimizer()

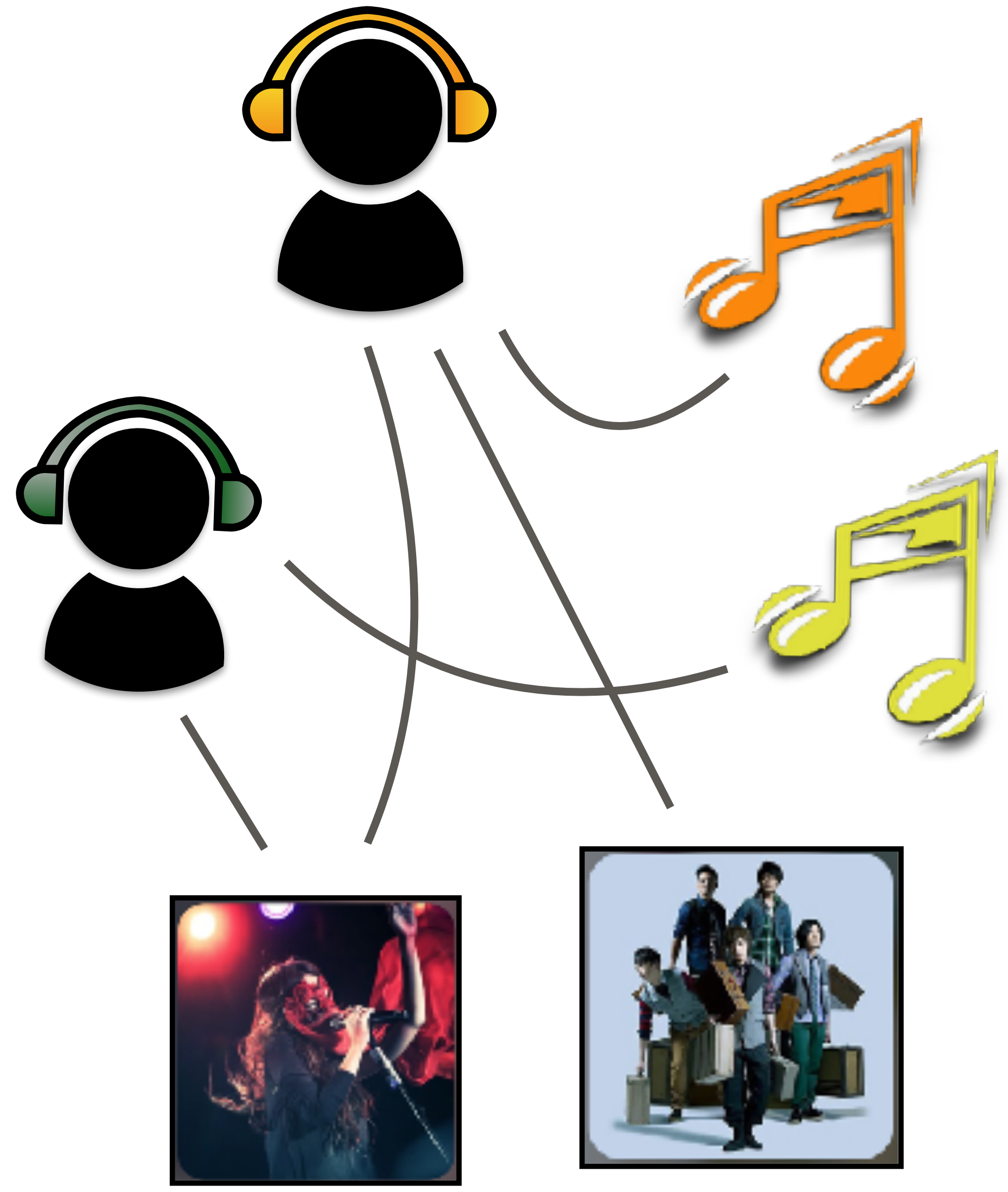
Embeddings

[0.08 0.02 0.28]

[-0.31 -0.1 0.1]

[-0.15 -0.3 0.2]

RECSYS 2019



Graph

unit1

Sampler()
Mapper()
Optimizer()

unit2

Sampler()
Mapper()
Optimizer()

unit3

Sampler()
Mapper()
Optimizer()

unit4

Sampler()
Mapper()
Optimizer()

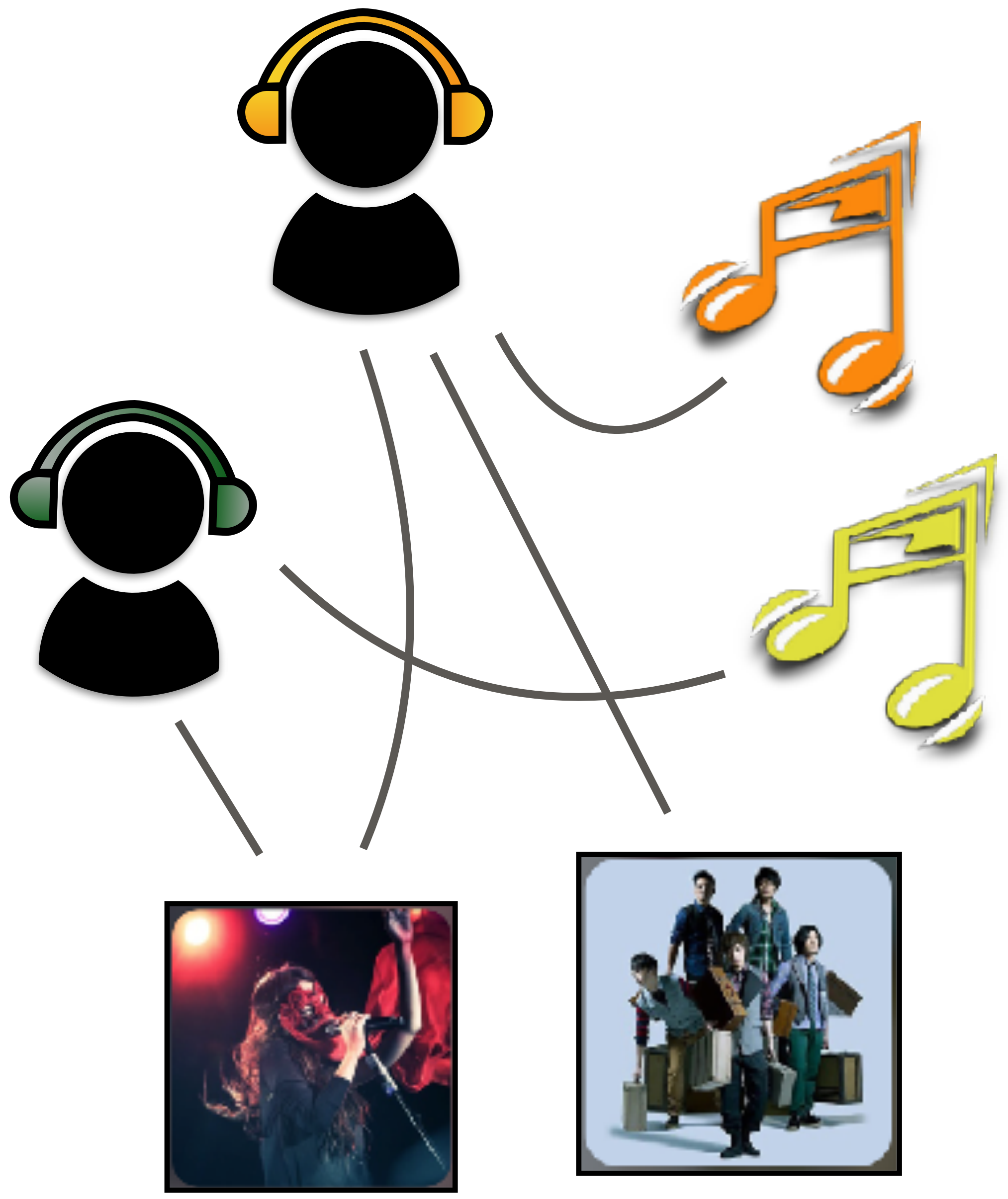
Embeddings

[0.08 0.02 0.28]

[-0.31 -0.1 0.1]

[-0.15 -0.3 0.2]

RECSYS 2019



Graph

https://www.reddit.com/r/aww/comments/2oagj8/multithreaded_programming_theory_and_practice/

Embeddings

[0.08 0.02 0.28]

[-0.31 -0.1 0.1]

[-0.15 -0.3 0.2]

HOGWILD!

Benjamin Recht, Christopher Ré, Stephen J. Wright, Feng Niu: Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. NIPS 2011: 693-701

Characteristics of SMORe

Holistic

Handles complex systems of interactions as a unified graph structure, allowing **joint mining** of diverse information to address core problems for REC such as **data sparsity** and **cold start**

Structural

Generalizes relations as **graph structures** composed of **vertices** and **edges**. Models can explore a spectrum of complex structures and their combinations for any given REC tasks

Modular

Breaks GE into **sampler**, **mapper**, and **optimizer**; which extracts interactions as structures, converts entities into spatial features, and preserves relatedness as spatial properties, respectively

Ting-Hsiang Wang, Texas A&M University

Benefits of SMORe

Speedy

Speeds development by reusing codes; **provides model toolkit** for REC and fair baseline comparison; and **accelerates training process** using CSAR and HOGWILD!

Effective

Adapts to different REC needs on module-level for embedding and structure-level for relations; also **opens to deep methods**, which continue to churn out SOTA models over the past years

Multi-task

Jointly captures different relations from HINs by selecting graph structures using sampler and combining embeddings using mapper

Coming Next ...

Lecture (Sean & CM, 65 minutes)

Hands-on (CM, 15 minutes)

Q&A (Sean & CM, 10 minutes)

QR to Slides, Codes, Abstract

RECSYS 2019

Ting-Hsiang Wang, Texas A&M University

SMORe :

Ting-Hsiang Wang, Texas A&M University

SMORe Members at CFDA & CLIP Labs : https://cfda.csie.org/

Chih-Ming Chen	Ting-Hsiang Wang	Chuan-Ju Wang	Ming-Feng Tsai
104761501@nccu.edu.tw	thwang1231@tamu.edu	cjwang@citi.sinica.edu.tw	mftsai@nccu.edu.tw

Special Thanks to :

政治大學 NATIONAL CHENGCHI UNIVERSITY

中央研究院 ACADEMIA SINICA

KKBOX

Link to Codes, Slides, Abstract :
https://github.com/cnclabs/smore/

RECSYS 2019