

Learning and Reasoning on Graph for Recommendation

Slides in <https://next-nus.github.io/>

Xiang Wang

National University of
Singapore



Xiangnan He

University of Science and
Technology of China



Tat-Seng Chua

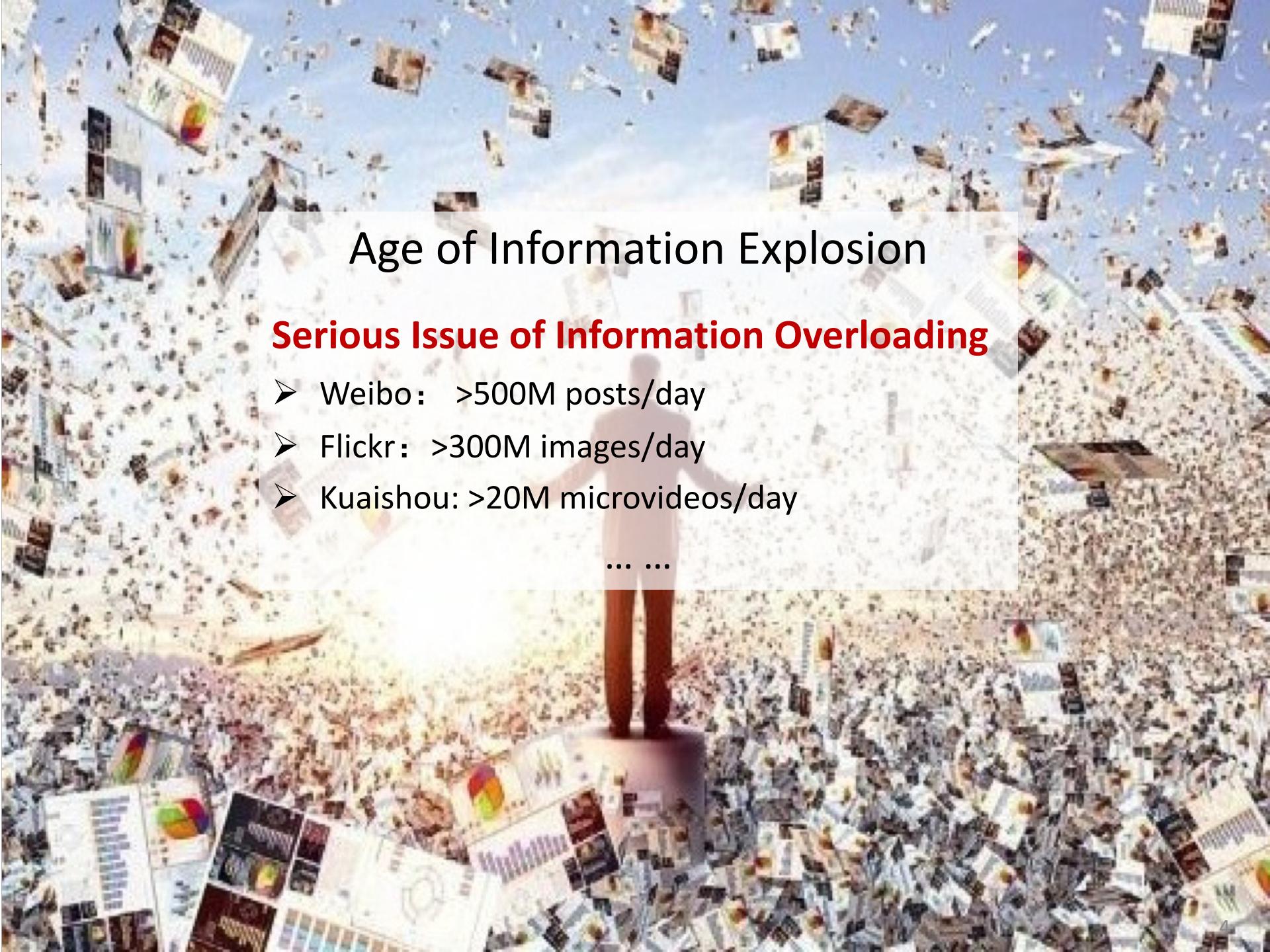
National University of
Singapore



OUTLINE

- **Introduction**
- Part I: Preliminary of Recommendation (~40mins)
- Part II: Random Walk for Recommendation (~20mins)
- Part III: Network Embedding for Recommendation (~20 mins)
- Part III: Graph Neural Networks for Recommendation (~100 mins)

Slides in <https://next-nus.github.io/>



Age of Information Explosion

Serious Issue of Information Overloading

- Weibo: >500M posts/day
- Flickr: >300M images/day
- Kuaishou: >20M microvideos/day

....

Ubiquitous Personalized Recommendation

Recommendation has been widely applied in online services:

- **E-commerce, Content Sharing, Social Networking, Forum ...**



淘宝网
Taobao

ebay



... Search results of Taobao

Ad & Product
Recommendation

Ubiquitous Personalized Recommendation

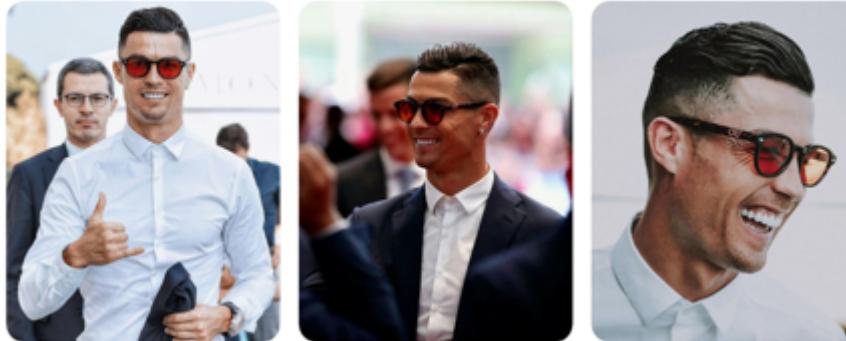
Recommendation has been widely applied in online services:

- E-commerce, **Content Sharing**, Social Networking, Forum ...



Image & Video
Recommendation

More like this

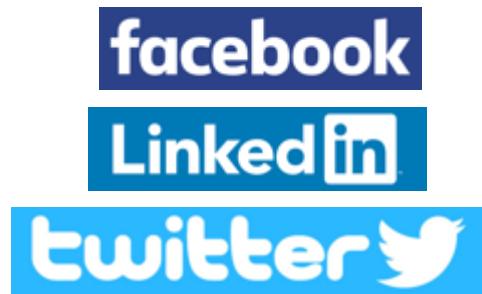


Search results of Pinterest

Ubiquitous Personalized Recommendation

Recommendation has been widely applied in online services:

- E-commerce, Content Sharing, **Social Networking**, Forum ...



People You May Know

A screenshot of a Facebook page titled "People You May Know". It lists four users with their profile pictures, names, and the number of mutual friends. To the right of the third and fourth entries, the text "Friend Recommendation" is written in red. A large yellow arrow points from the social media logos on the left towards this section.

User Profile	Name	Mutual Friends
	(Cheryl)	and 2 other mutual friends
		3 mutual friends
		and 4 other mutual friends
		and 5 other mutual friends

Friend
Recommendation

Screenshot of Facebook

Ubiquitous Personalized Recommendation

Recommendation has been widely applied in online services:

- E-commerce, Content Sharing, Social Networking, **Forum ...**



Explore Beijing

The screenshot shows a mobile application interface for "Explore Beijing". It features four main sections: 1) "Best Seller" with a picture of a lion statue and a "Beijing Forbidden City Admission Ticket" for \$14 per adult, with a "Book Now" button. 2) "FULL-DAY TOURS" with a picture of people at the Great Wall and a "Mutianyu Great Wall Small-Group Tour from Beijing including Lunch" for \$63 per adult, with a "Book Now" button. 3) "THEATER, SHOWS & MUSICALS" with a picture of performers and a "Shaolin Kung Fu Show in Beijing Red Theater" for \$14 per adult, with a "Book Now" button. 4) "HALF-DAY TOURS" with a picture of a woman and a "4-Hour Private Beijing Walking Tour of the Forbidden City" for \$63 per adult, with a "Book Now" button. Each listing includes a heart icon and a review count (e.g., 97 reviews, 249 reviews, 39 reviews).

POI & Post
Recommendation

Screenshot of TripAdvisor

Values of Recommender System (RecSys)

RecSys has become a major **monetization** tool for customer-oriented online services:

- E-commerce, Content Sharing, Social Networking, Forum ...

Ad systems are technically supported by recommendation solutions:

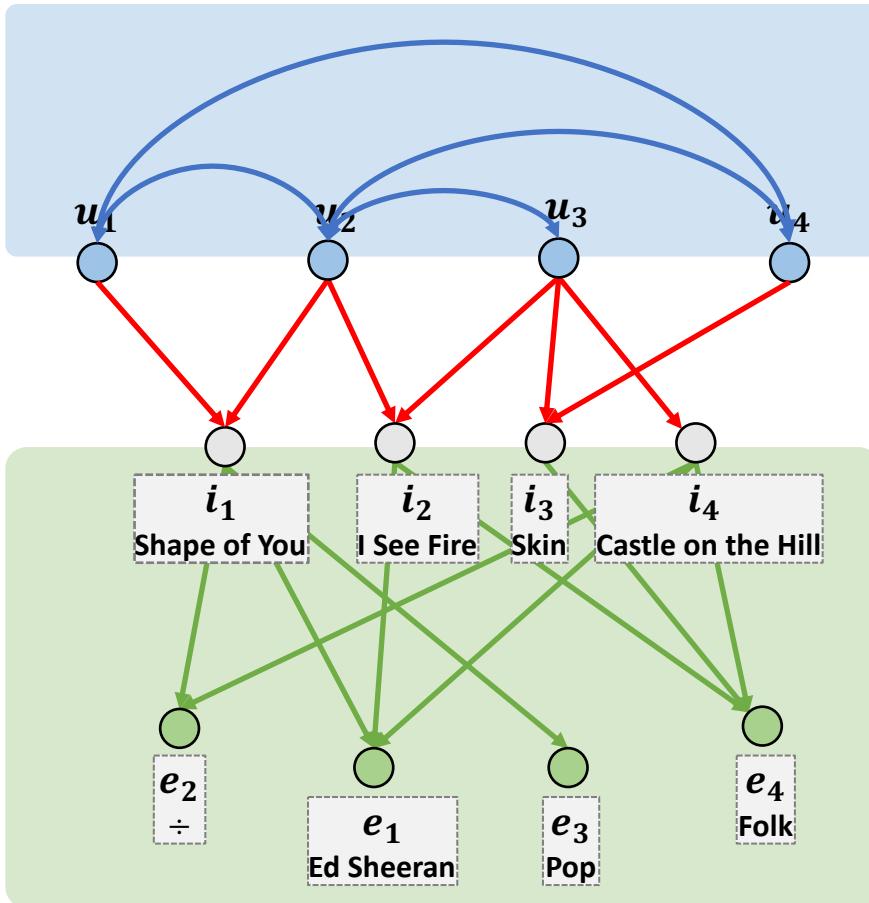
- The key is Click-Through Rate (CTR) prediction.

Some Statistics:

- YouTube Homepage: 60%+ clicks [Davidson et al. 2010]
- Netflix: 80%+ movie watches [Gomze-Uribe et al 2016]
- Amazon: 30%+ page views [Smith and Linden, 2017]

The Era of **Connected** World

The world is more **closely connected** than you might think!



User-User Connections

- Social Relations
- Same Profiles ...

User-Item Interactions

- Implicit Feedback
- Explicit Feedback ...

Item-Item Connections

- Same Attributes
- External Knowledge ...

OUTLINE

- Introduction
- **Part I: Preliminary of Recommendation**
 - **Problem Formulation**
 - **Unified View for Recommendation Paradigm**
 - **Limitations of Previous Works**
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- Part III: Graph Neural Networks for Recommendation

Slides in <https://next-nus.github.io/>

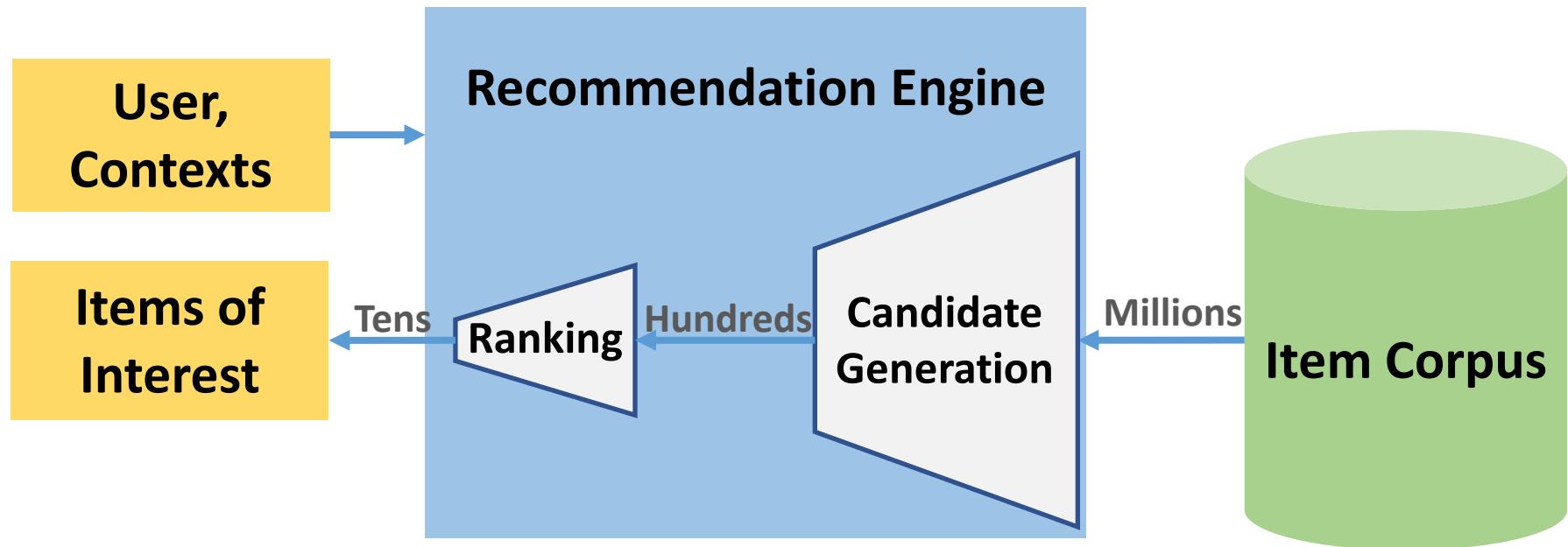
Overview of Recommendation Engine

User Interest is implicitly reflected in:

- Interaction history
- Demographics ...
- Contexts

Items can be:

Products, News, Movies,
Videos, Friends ...

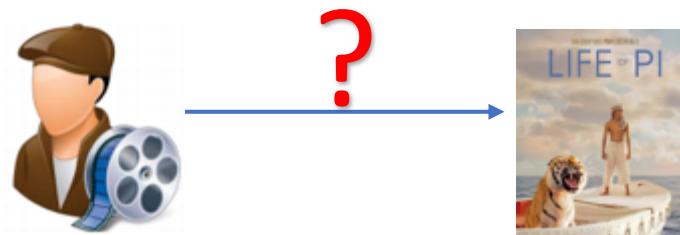


Key challenge: user-item semantic gap

- user and item are two **different types of entities** and are represented by different features.

Problem Formulation

- **Input:** historical user-item interactions or additional side information (e.g., user profile, item profile)
- **Output:** given a target Item (e.g., movie, song, product), how likely a user would interact with it (e.g., click, view, or purchase)



User Profile:

- User ID
- Rating history
- Age, Gender
- Clicks
- Income level

.....

Item Profile:

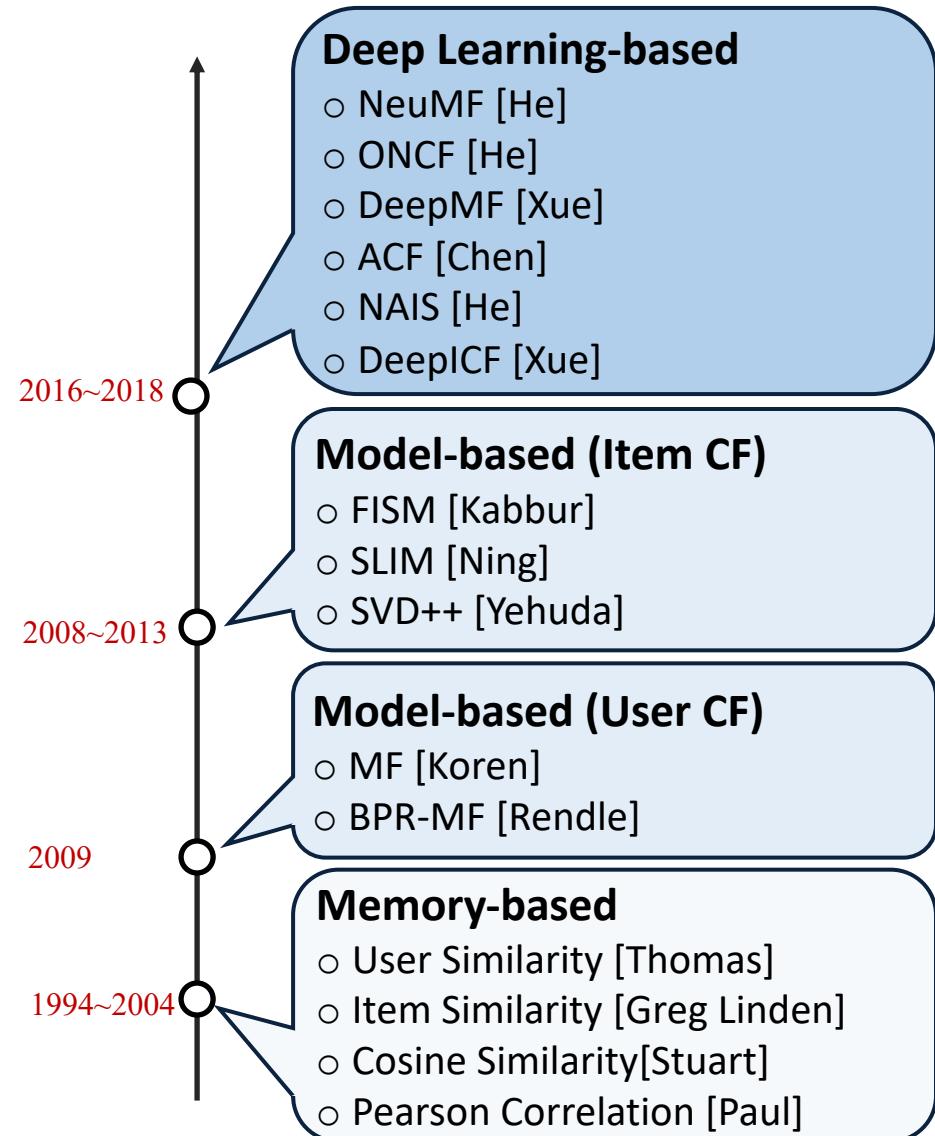
- Item ID
- Description
- Image
- Category
- Price

.....

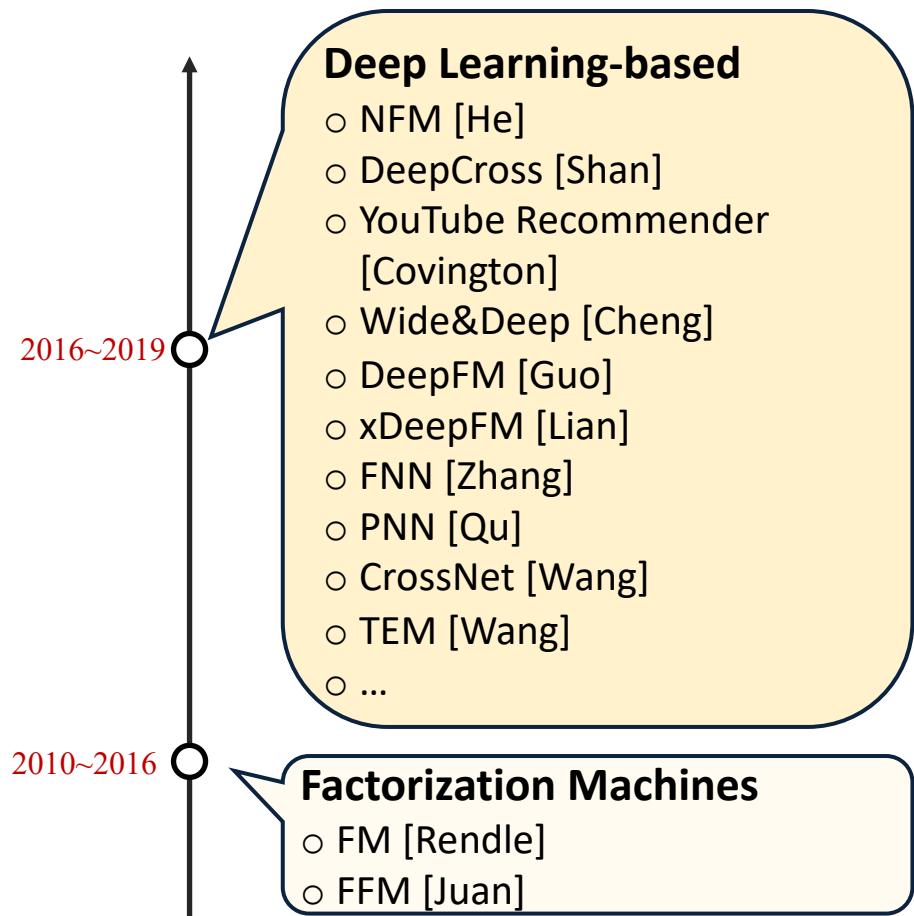
There may be **no overlap** between user features and item features.

Research on Prevalent RecSys

Collaborative Filtering Models

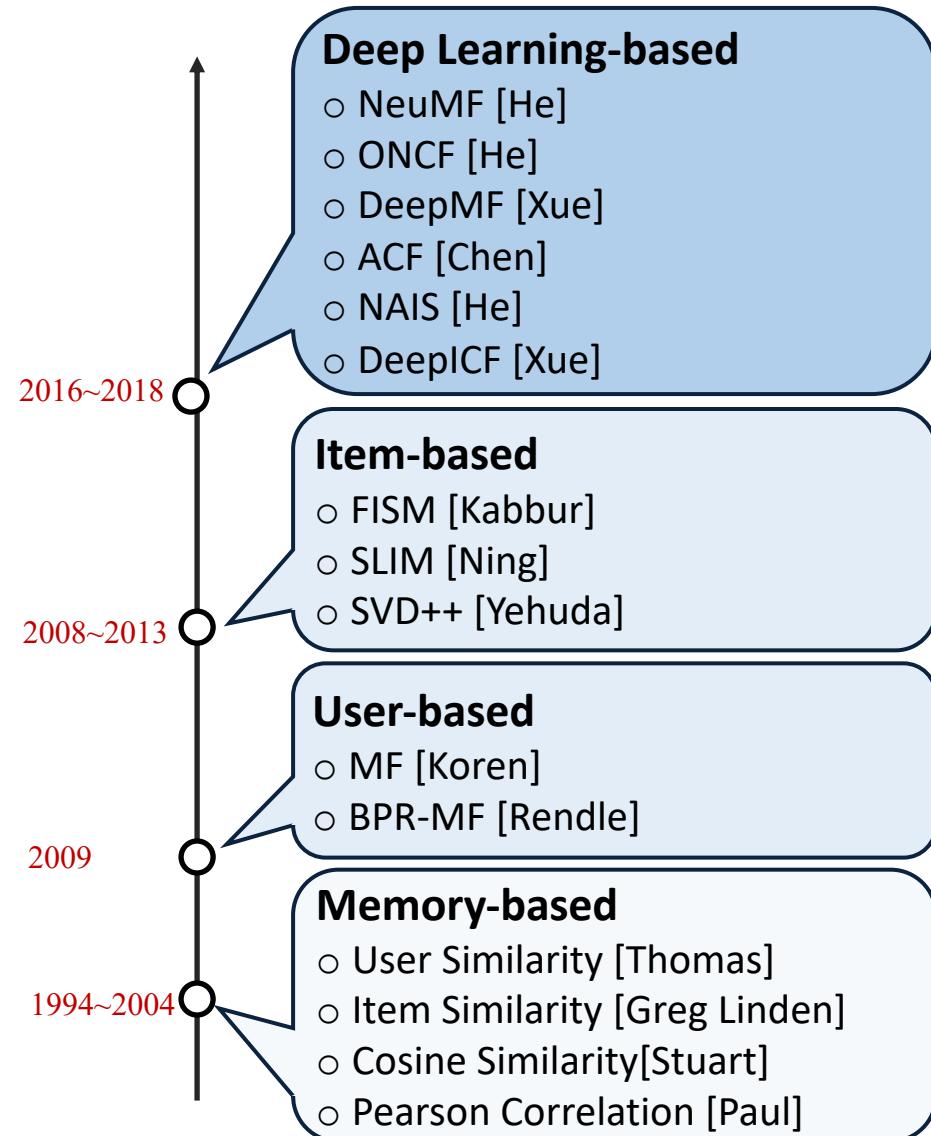


Generic Feature-based Models



Research on Collaborative Filtering Models

Collaborative Filtering Models



Input Data:

- User-Item Interaction Data
 - Explicit Feedback (e.g., rating)
 - Implicit Feedback (e.g., clicks)

$$(u_1 \circledcirc i_1 \circledcirc) \quad y_{u_1 i_1}$$

$$(u_2 \circledcirc i_1 \circledcirc) \quad y_{u_1 i_1}$$

$$(u_2 \circledcirc i_2 \circledcirc) \quad y_{u_1 i_1}$$

$$(u_3 \circledcirc i_2 \circledcirc) \quad y_{u_1 i_1}$$

$$(u_3 \circledcirc i_3 \circledcirc) \quad y_{u_1 i_1}$$

... ...

Collaborative Filtering (CF)

- CF is the most well-known technique for recommendation.
 - “CF makes predictions (**filtering**) about a user’s interest by collecting preferences information from many users (**collaborating**)” ---Wikipedia
- Collaborative Signals → Behavior Similarity of Users
 - Similar users would have similar preference on items.

The diagram illustrates the process of creating an interaction matrix from user-item rating data. On the left, a list of user-item pairs with their ratings is shown:

$(u_1 \circlearrowleft i_1 \circlearrowright)$	5
$(u_2 \circlearrowleft i_1 \circlearrowright)$	3
$(u_2 \circlearrowleft i_2 \circlearrowright)$	4
$(u_3 \circlearrowleft i_2 \circlearrowright)$	1
$(u_3 \circlearrowleft i_3 \circlearrowright)$	2
$(u_3 \circlearrowleft i_4 \circlearrowright)$	4
...

An arrow points from this list to the right, where an "Interaction Matrix" is shown as a grid:

		item				
		1	2	3	4	...
user	1	5	?	?	?	...
	2	3	4	?	?	...
3	?	1	2	4	...	
	

The matrix has "user" labeled vertically on the left and "item" labeled horizontally at the top. The first row and column are empty. The matrix contains numerical values (5, 3, 4, 1, 2, 4) and question marks (?) representing known and unknown ratings respectively. Ellipses indicate more users and items.

Interaction Matrix

1. **Memory-based CF:**
Predict by **memorizing** similar users’ (or items’) ratings
2. **Model-based CF:**
Predict by **inferring** from an underlying model.

Memory-based CF

Problem: predict user u 's rating on item i .

- User-based CF leverages the ratings of u 's **similar users** on the target item i .

$$\hat{y}_{ui} = \sum_{u' \in S_u(u)} sim(u, u') \cdot y_{u'i}$$

Rating of a similar user on i

Similar users of u

- Item-based CF leverages the ratings of u on other **similar items** of i .

$$\hat{y}_{ui} = \sum_{i' \in S_i(i)} sim(i, i') \cdot y_{ui'}$$

Rating of u on a similar item

Similar items of i

- Many similarity measures can be used, e.g., Jaccard, Cosine, Pearson Correlation. Recent advance learns the similarity from data.

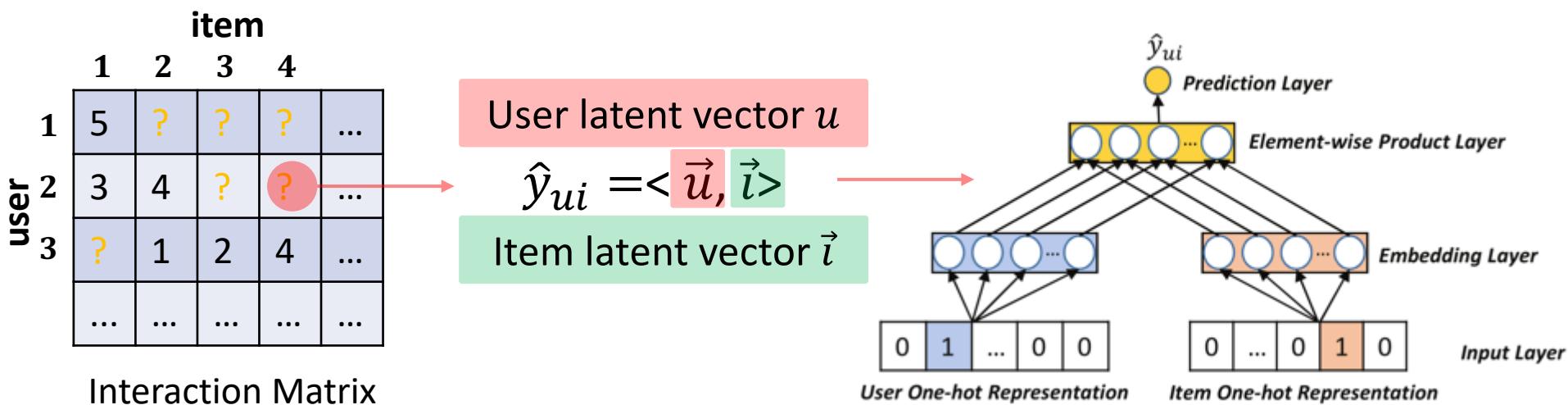
		item					
		1	2	3	4		
user	1	5	?	?	?	...	
	2	3	4	?	?	...	
3	?	1	2	4	...		
...		

Interaction Matrix

Model-based CF

Matrix Factorization (MF) is the most popular and effective model-based CF method.

- It represents a user and an item as a vector of latent factors.
- The score is estimated as the **inner product** of user latent vector and item latent vector.



- Optimizing a loss to minimize the prediction error on training data can get the latent vectors.

Item-based CF

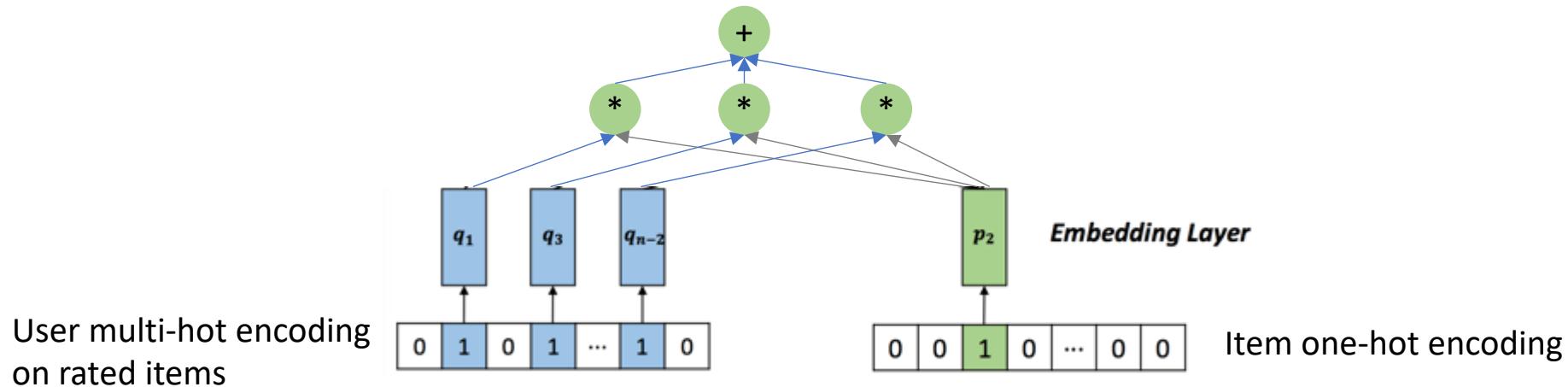
Instead of only using an ID to encode a user, we can make the encoding more meaningful by using the user's rated items.

- This can be interpreted as an item-based CF model.

$$\hat{y}_{ui} = \sum_{i' \in S_i(i)} sim(i, i') \cdot y_{ui'} \approx \sum_{j \in \mathcal{R}_u} \mathbf{q}_j^T \mathbf{p}_i$$

Use all items as neighbors

Factorize item similarity in the latent space



- E.g., FISM [Kabbur], SLIM [Ning]

Fusing User-based & Item-based CF

- MF (user-based CF) represents a user as her ID.
 - Directly projecting the ID into latent space
- FISM (item-based CF) represents a user as her interacted items.
 - Projecting interacted items into latent space
- SVD++ fuses the two types of models in the latent space:

$$\hat{y}_{ui} = (\mathbf{v}_u + \sum_{j \in \mathcal{R}_u} \mathbf{q}_j)^T \mathbf{p}_i$$

User representation in latent space

- This is the best single model for rating prediction in the Netflix challenge [Koren, KDD '08].

Two Widely-Used Loss

Pointwise loss → e.g., log loss

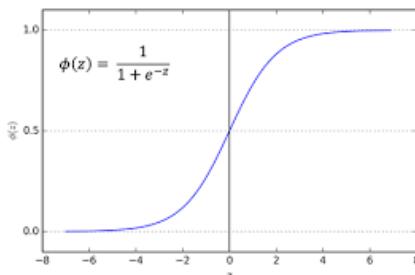
- Cast the recommendation task as a **classification** problem
- Rating Prediction, CTR Prediction ...

$$\mathcal{L} = \sum_{(u, i, x) \in O} -y_{ui} \log \sigma(\hat{y}_{ui}) - (1 - y_{ui}) \log (1 - \sigma(\hat{y}_{ui}))$$

Force the prediction scores to be close to the target scores

Pairwise loss → e.g., Bayesian Personalized Ranking (BPR) loss

- Cast the recommendation task as a **ranking** problem
- Top-N Recommendation, Preference Ranking ...

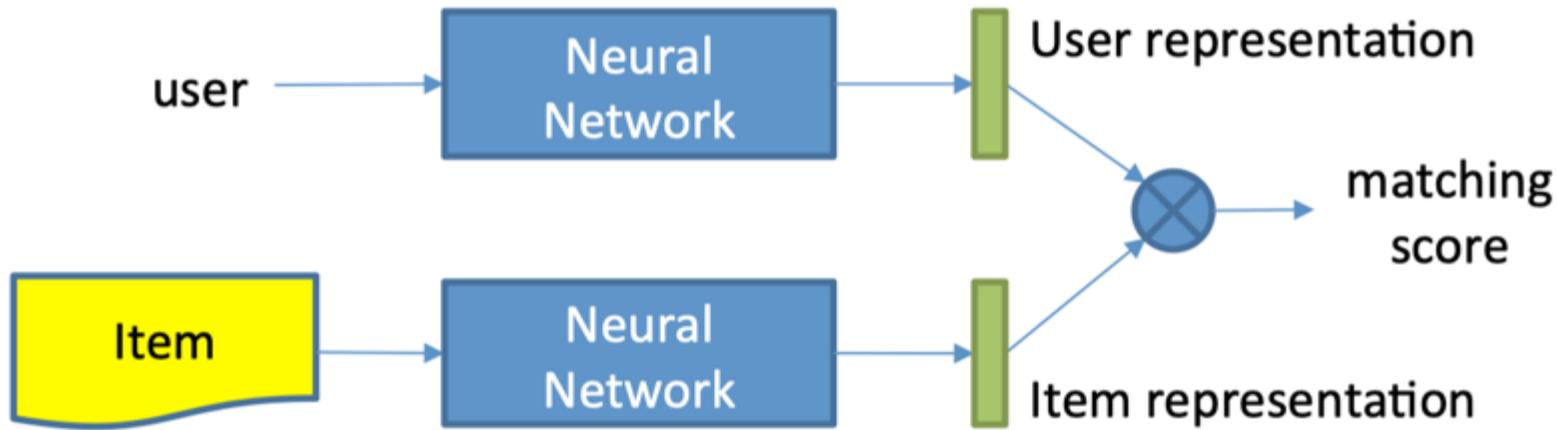


$$Loss = \sum_{(u, i, j) \in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2$$

Relative order between observed & unobserved interactions

Deep Learning Meets CF (1)

- **Methods of representation learning**
 - Enhance representation ability/expressiveness of models

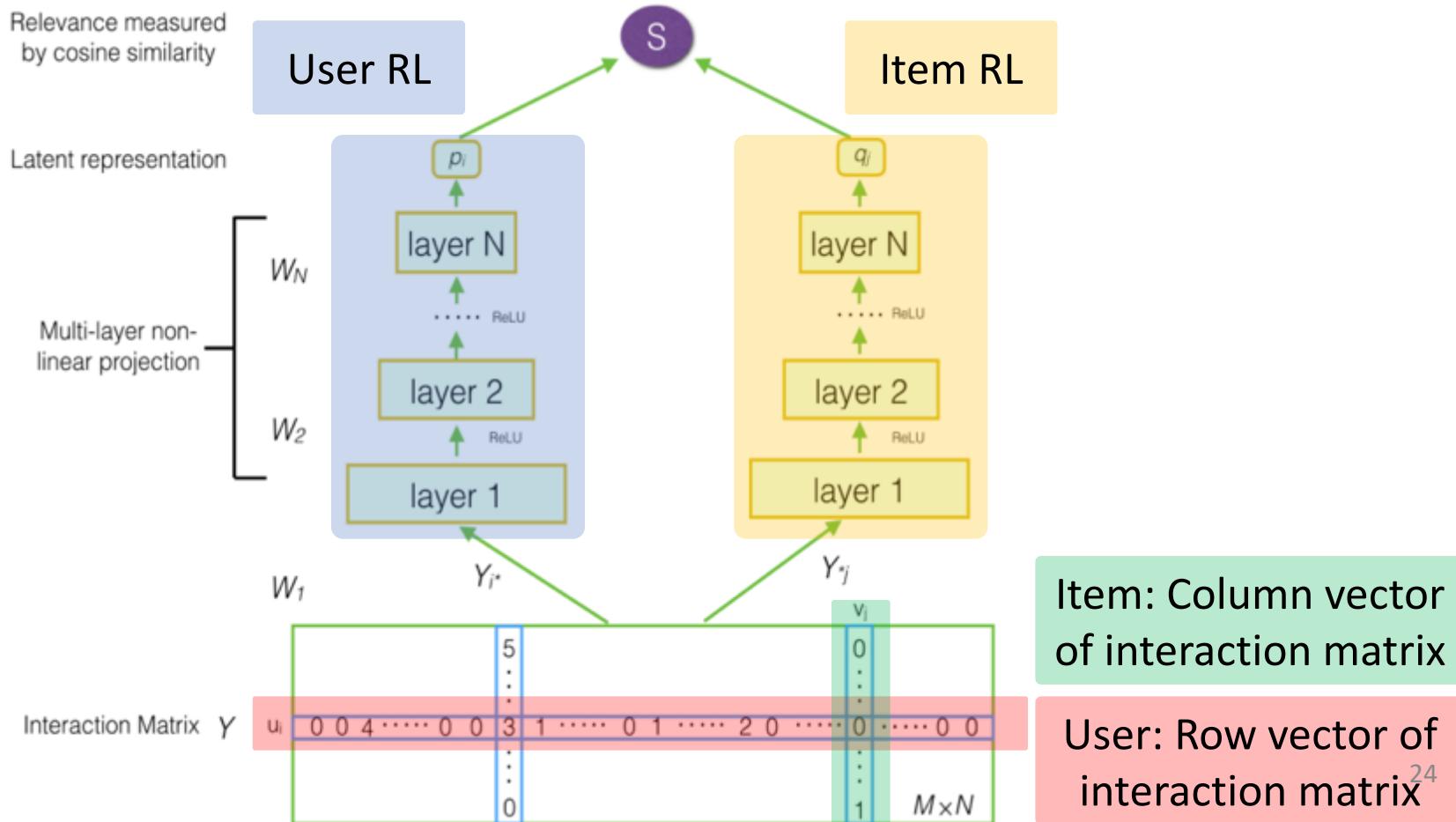


Model	Input Data	Representation Learning	Interaction Learning
DeepMF [Xue, IJCAI'17]	User: Historical items Item: User group	Multi-Layer Perceptron	Inner product
AutoRec [Sedhain, WWW'15]	User: Historical items Item: ID	Multi-Layer Perceptron	Inner Product
CDAE [Wu, WSDM'16]	User: Historical items + ID Item: ID	Multi-Layer Perceptron	Inner Product

Deep Matrix Factorization (Xue, IJCAI'17)

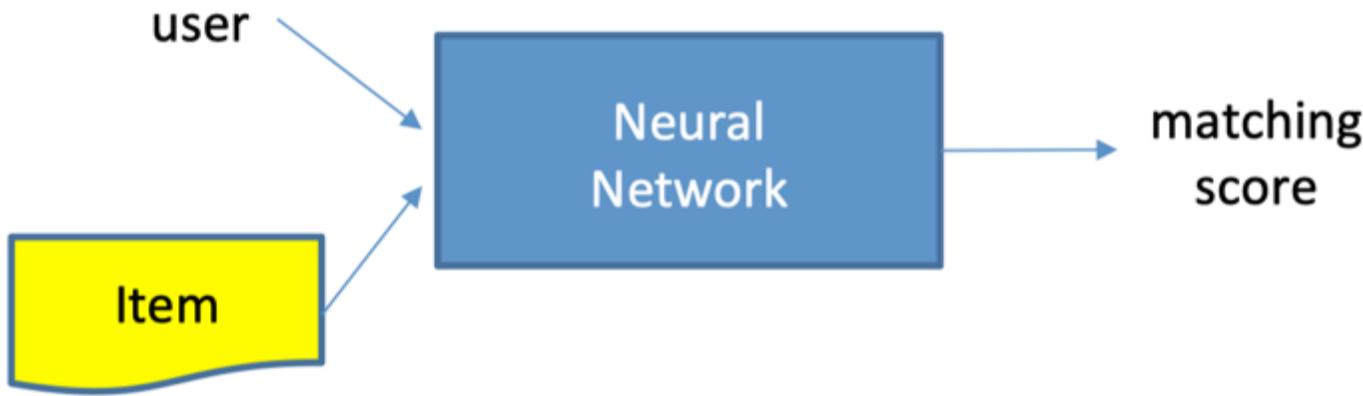
Representation Learning → Multi-layer perceptron

- Deep Neural Networks are adopted to learn representations of users & items



Deep Learning Meets CF (2)

- **Methods of interaction function learning**
 - Capture complex patterns of user-item relationships

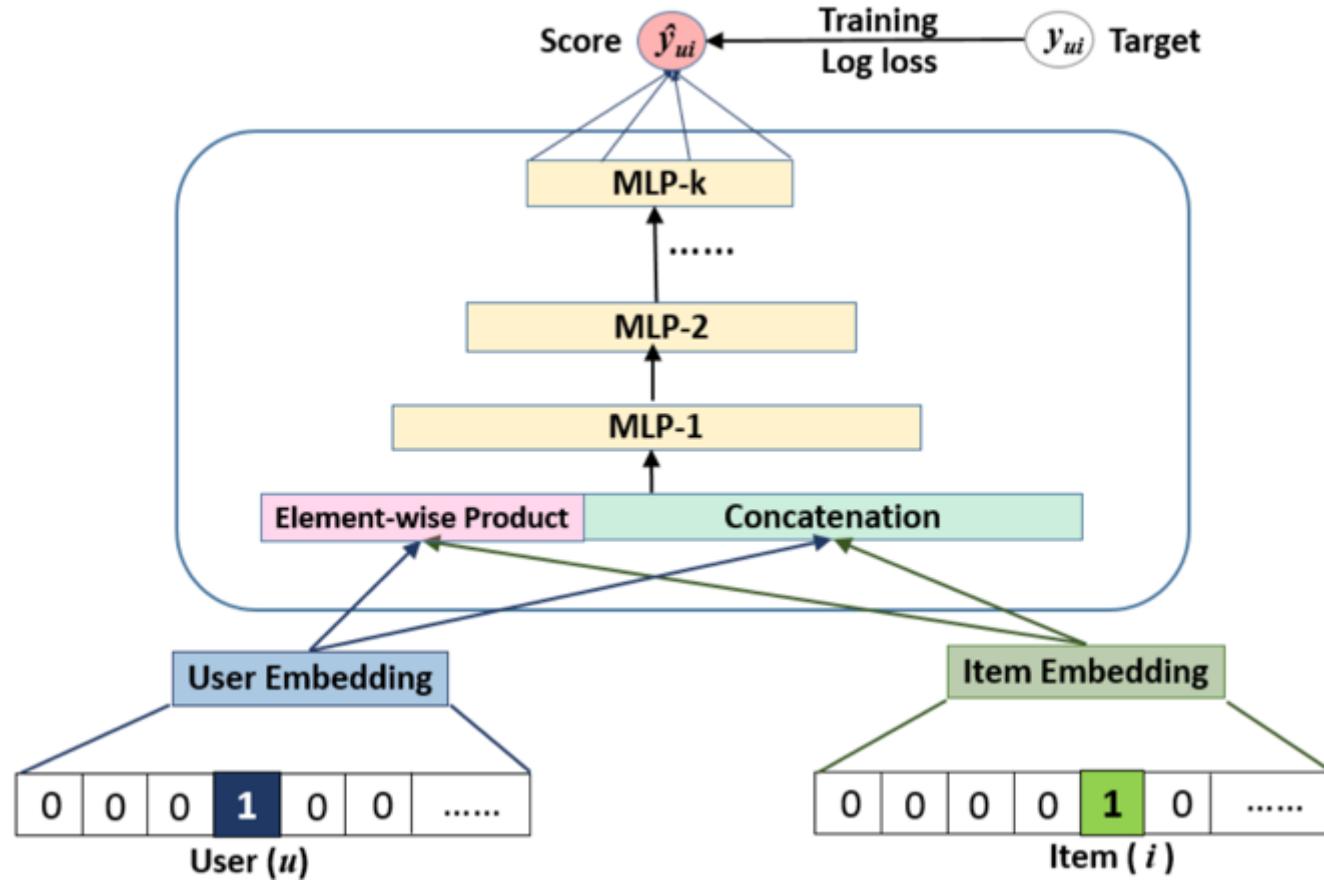


Model	Input Data	Representation Learning	Interaction Learning
NCF [He, WWW'17]	User: ID Item: ID	ID embedding	Multi-Layer Perceptron
NNCF [Bai, CIKM'17]	User: User neighbors Item: Item neighbors	Embeddings	Multi-Layer Perceptron
ONCF [He, IJCAI'28]	User: ID Item: ID	ID embedding	Convolutional Neural Network

Neural Matrix Factorization (He, WWW'17)

Interaction Modeling → MF + MLP over users and items

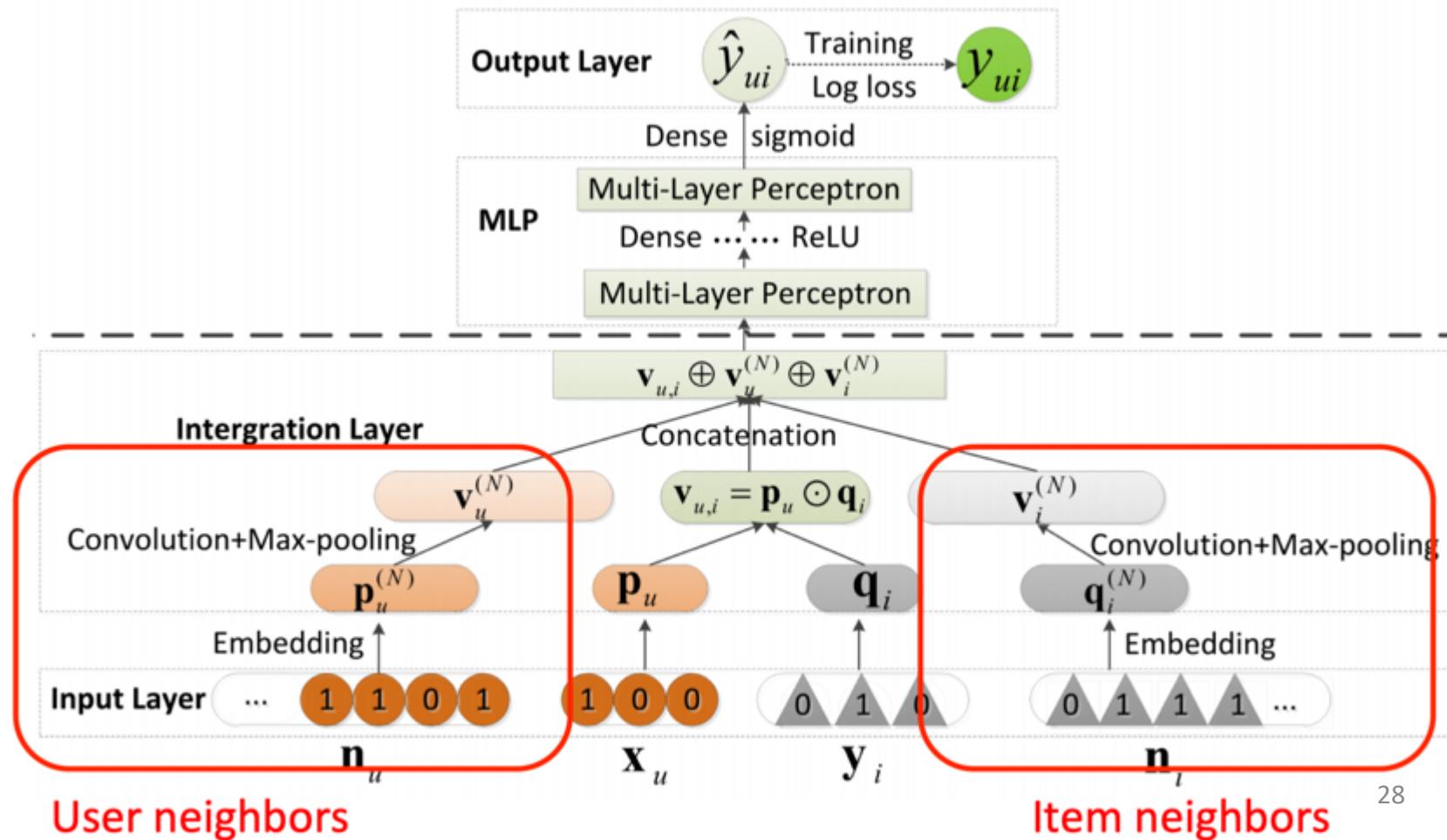
- MF uses inner product to capture the low-rank relation
- MLP is more flexible in using DNN to learn the matching function.



NNCF: Neighbor-based NCF (Bai, CIKM'17)

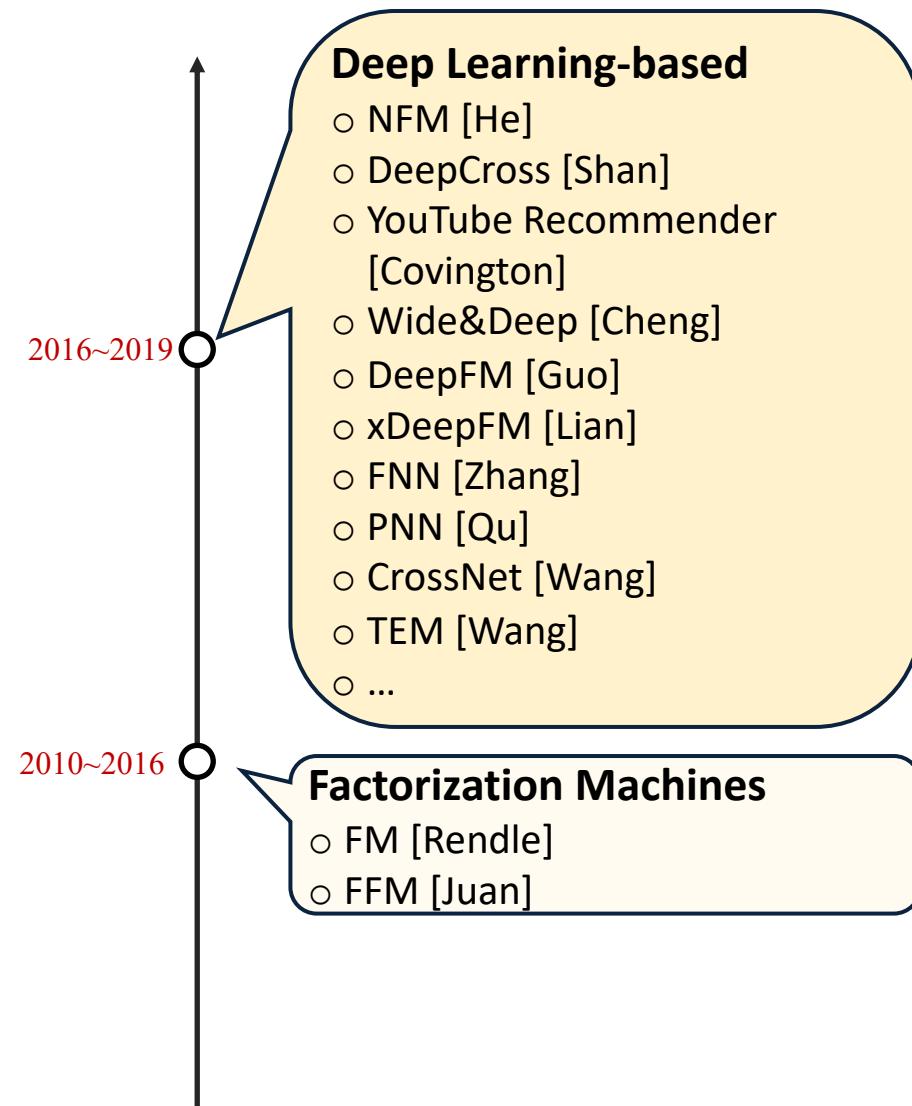
Interaction Modeling → MF + MLP over user and item neighbors

- Feeding user and item neighbors into the NCF framework



Research on Feature-based Models

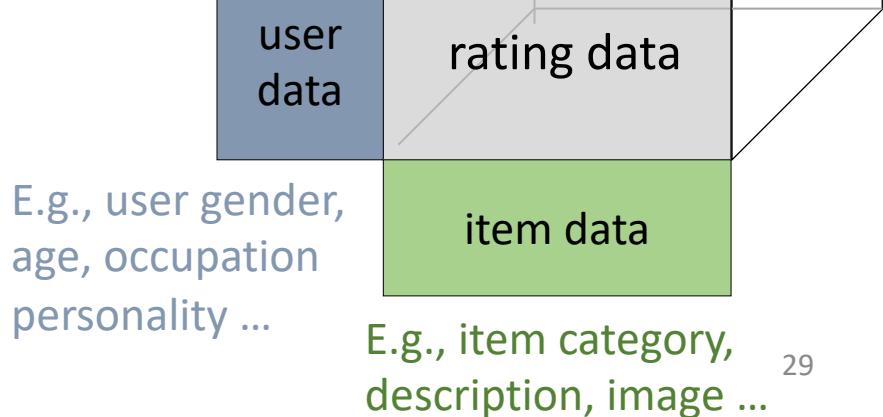
Generic Feature-based Models



Input Data:

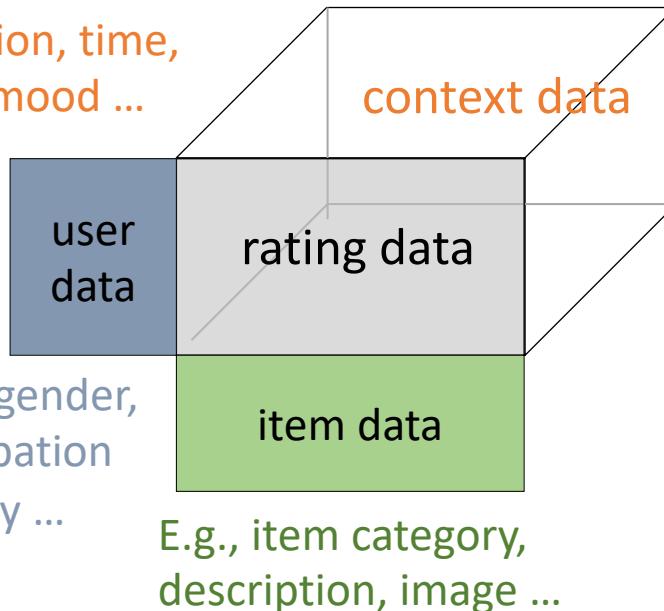
- User-Item Interaction Data
- Other Information
 - User Data
 - Item Data
 - Context Data

E.g., location, time, weather, mood ...



Feature-based Models

E.g., location, time,
weather, mood ...



E.g., user gender,
age, occupation
personality ...

E.g., item category,
description, image ...

Feature vector \mathbf{x}

$\mathbf{x}^{(1)}$	1 0 0 ...	1 0 0 0 ...	0.3 0.3 0.3 0 ...
$\mathbf{x}^{(2)}$	1 0 0 ...	0 1 0 0 ...	0.3 0.3 0.3 0 ...
$\mathbf{x}^{(3)}$	1 0 0 ...	0 0 1 0 ...	0.3 0.3 0.3 0 ...
$\mathbf{x}^{(4)}$	0 1 0 ...	0 0 1 0 ...	0 0 0.5 0.5 ...
$\mathbf{x}^{(5)}$	0 1 0 ...	0 0 0 1 ...	0 0 0.5 0.5 ...
$\mathbf{x}^{(6)}$	0 0 1 ...	1 0 0 0 ...	0.5 0 0.5 0 ...
$\mathbf{x}^{(7)}$	0 0 1 ...	0 0 1 0 ...	0.5 0 0.5 0 ...
A B C ...	TI NH SW ST ...	TI NH SW ST ...	Other Movies rated ...
User	Movie		

Raw Features:

- **Categorical Features:**
 - One-hot encoding on ID features
- **Continuous Features:**
 - E.g., time, frequency. Need feature normalization

Transformed Features:

- **Categorical Features:**
 - Cross Features are important (e.g., AND(A=True, B=True))
- **Continuous Features:**
 - E.g., outputs of other models like visual embeddings.

Factorization Machine (FM)

- FM is inspired from previous factorization models
- It represents each feature as a latent vector (embedding), and models the second-order **feature interactions**:

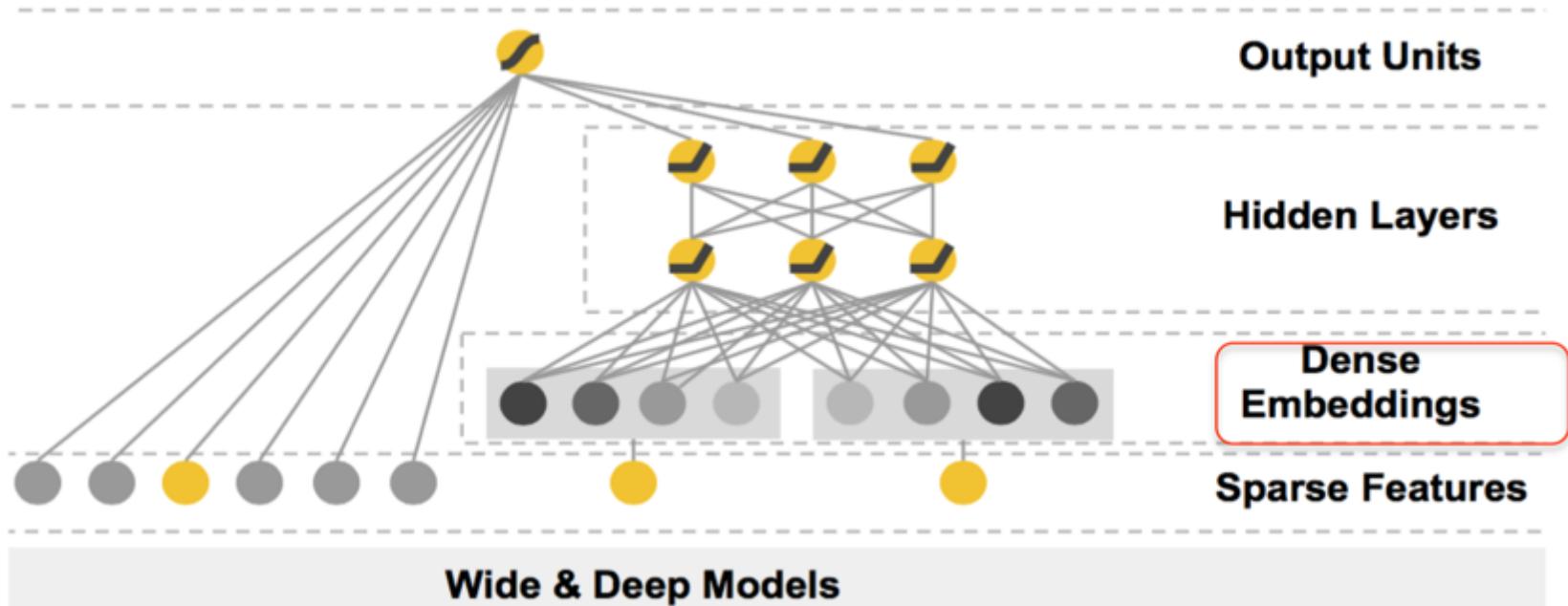
$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^p w_i x_i + \sum_{i=1}^p \sum_{j>i}^p \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

First-order:
Linear Regression

Second-order:
Pair-wise interactions between features

- FM allows easy feature engineering for recommendation, and can mimic many existing models (that are designed for a specific task) by inputting different features.
 - E.g., MF, SVD++, timeSVD [Koren, KDD'09], PIFT [Rendle, WSDM'10] etc.

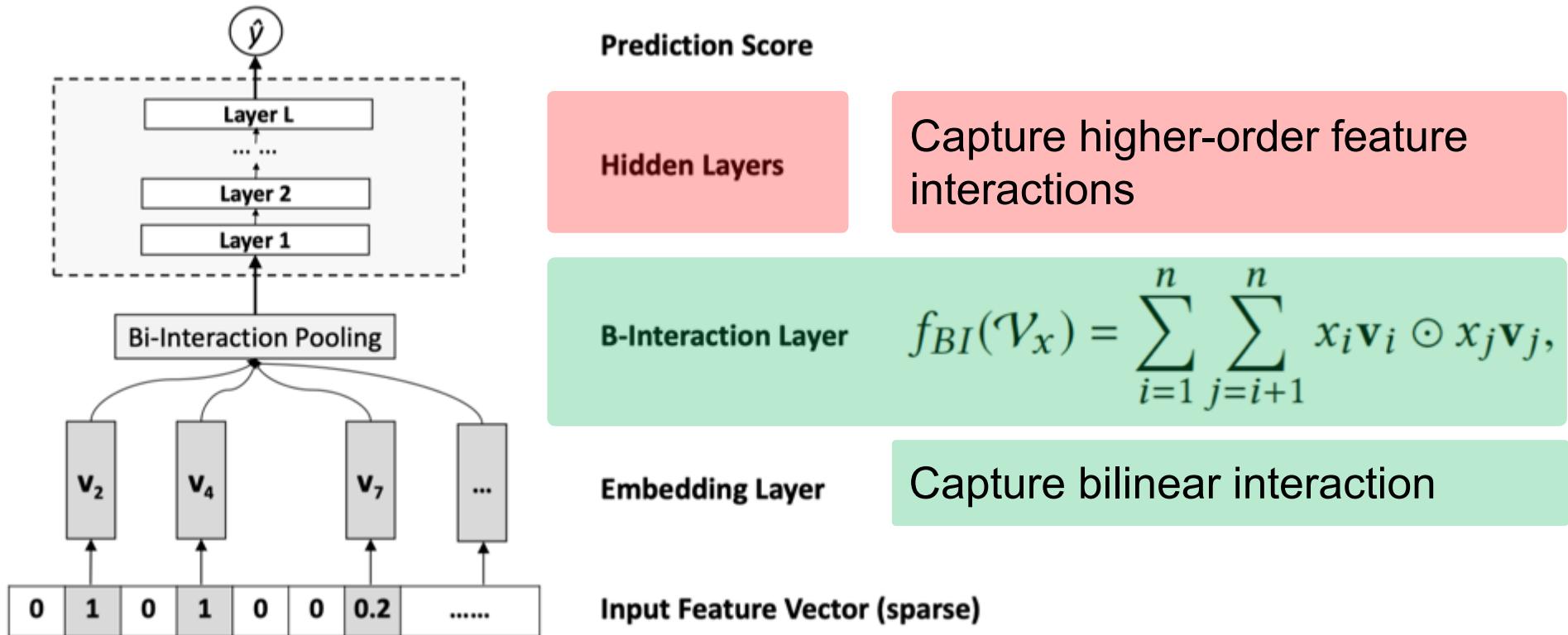
Wide&Deep (Cheng et al, RecSys'16)



- The **wide part** is linear regression for memorizing seen feature interactions, which requires careful engineering on cross features.
 - E.g., $\text{AND}(\text{gender=female}, \text{language=en})$ is 1 if both single features are 1
- The **deep part** is DNN for generalizing to unseen feature interactions.
 - Cross feature effects are captured in an implicit way.

Neural Factorization Machine (He et al, SIGIR'17)

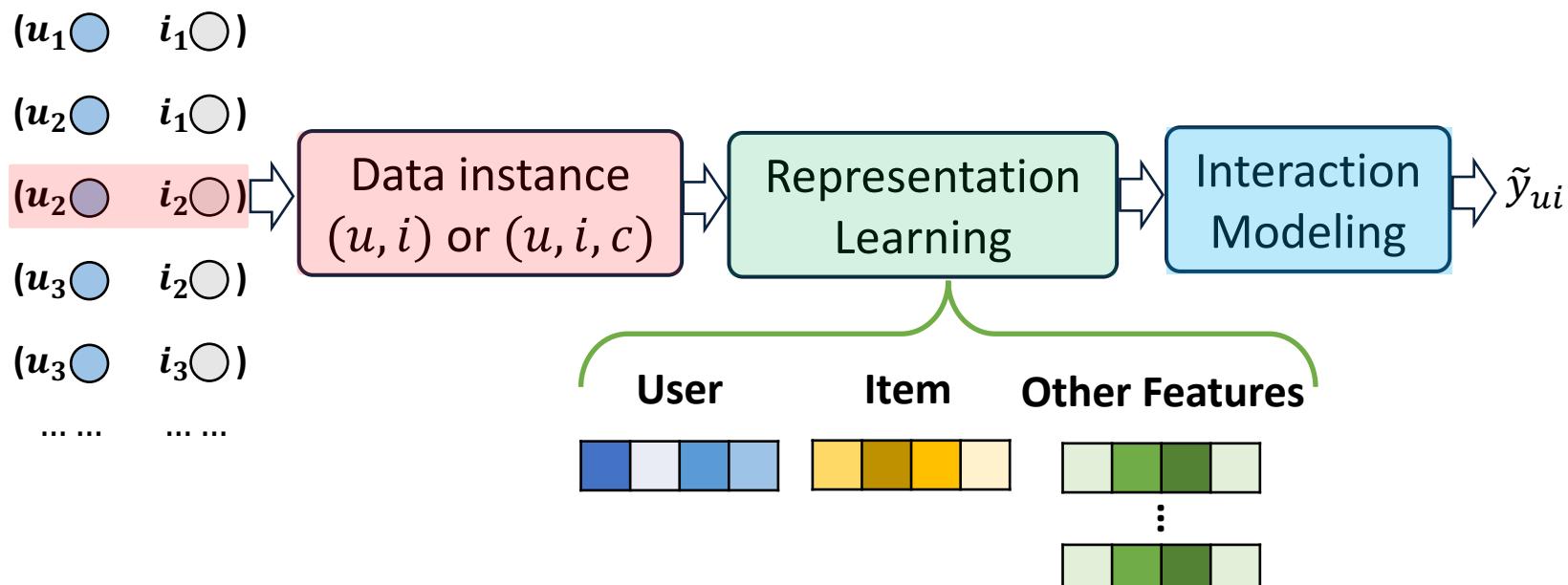
- Inspired by FM, NFM models **pairwise interactions** between feature embeddings with multiplication.



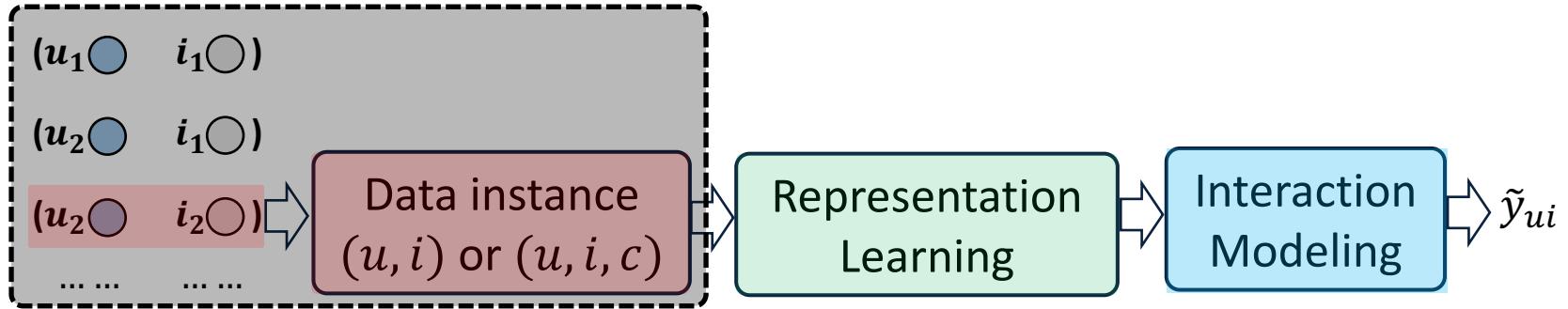
A General Paradigm

Transform each observation, a user-item pair (u, i) or with side information (u, i, c) , into a separate data instance

- Initiate representations for each feature → Representation Learning
 - Design whatever features as you want
- Perform predictions based on interactions → Interaction Modeling
 - Design whatever networks as you like

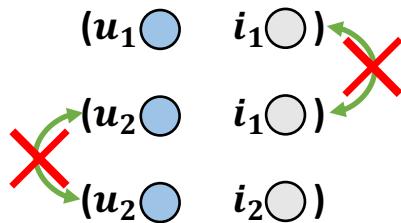


Information Isolated Island Issue (1)



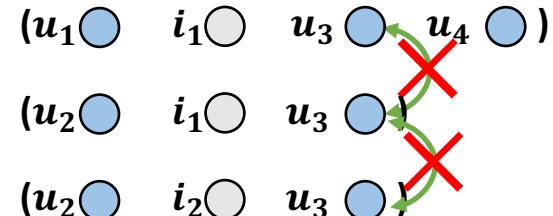
Treating each observation as an independent instance

- Forgoing relationships among instances



User-Item Interactions

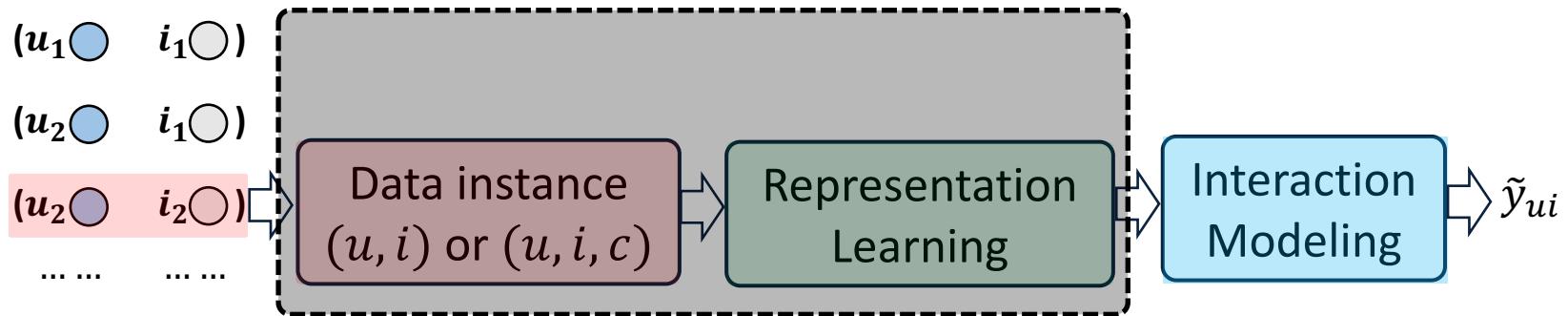
- Behavior similarity among users
- Audience similarity among items



User-Item Interactions + Social Ties

- Shared friends as bridge among users
→ mouth marketing

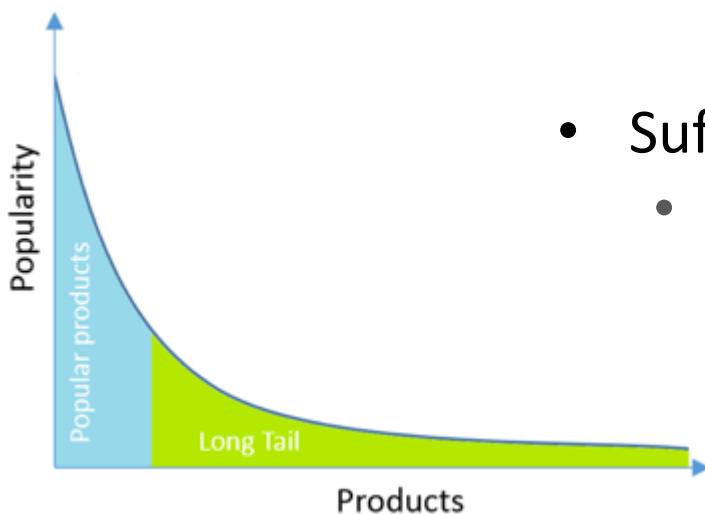
Information Isolated Island Issue (2)



Treating each observation as an independent instance

- **Limited Representation Ability**

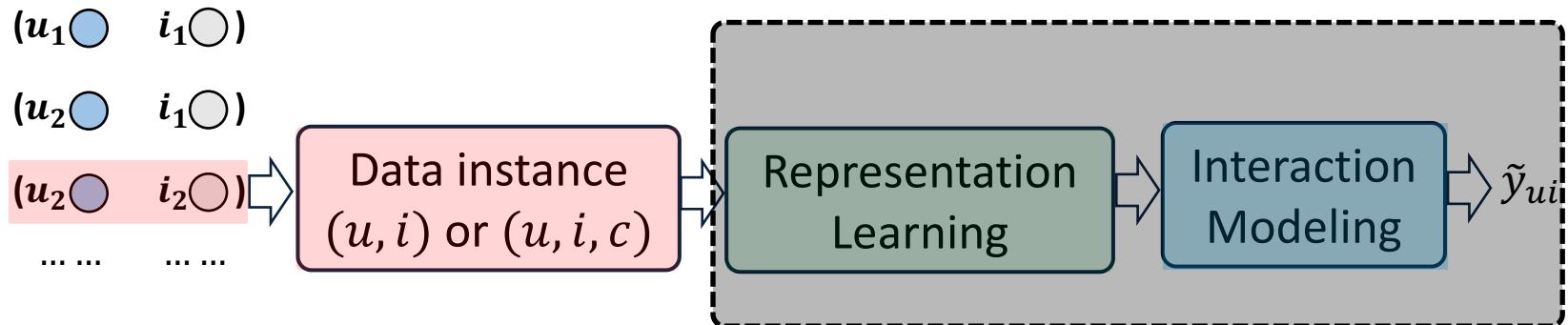
- Instance representation is dependent on its own features merely
 - SVD++, NAIS: CF with neighbors as input are more expressive



- Suffer from sparsity issue
 - inactive users, unpopular items, infrequent features → insufficient information to learn optimal representation

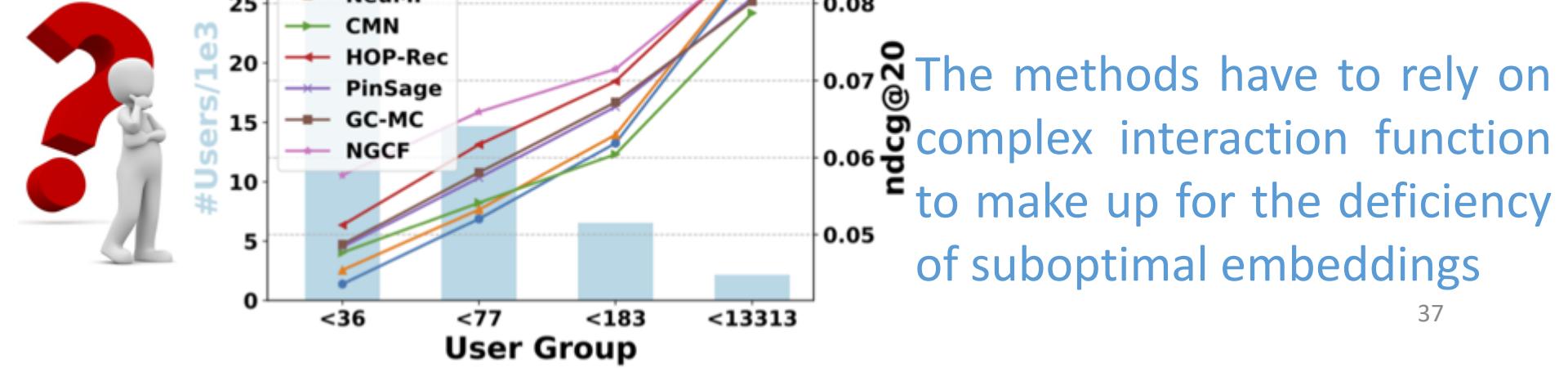


Information Isolated Island Issue (3)

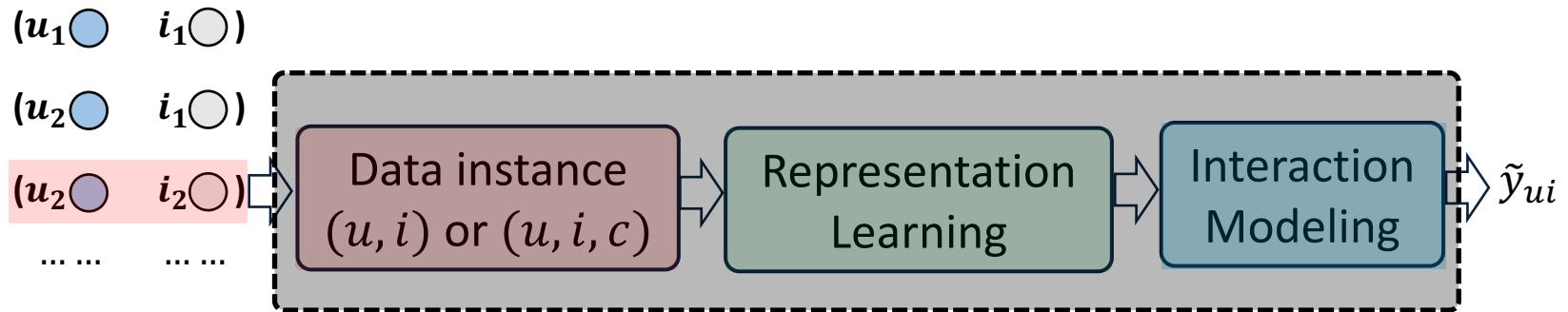


Treating each observation as an independent instance

- Suboptimal Model Capacity
 - Suboptimal representations lead to unsatisfactory interaction model, especially for unseen (user-item or feature) interactions



Information Isolated Island Issue (4)



Treating each observation as **an independent instance**

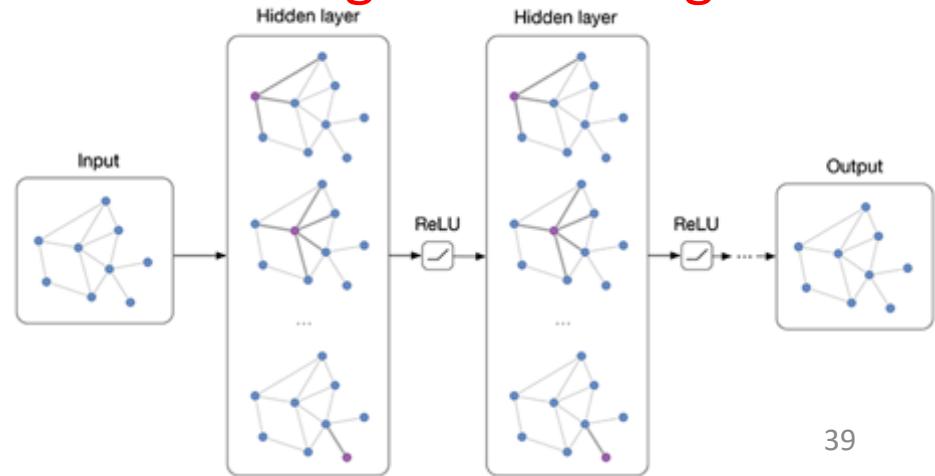
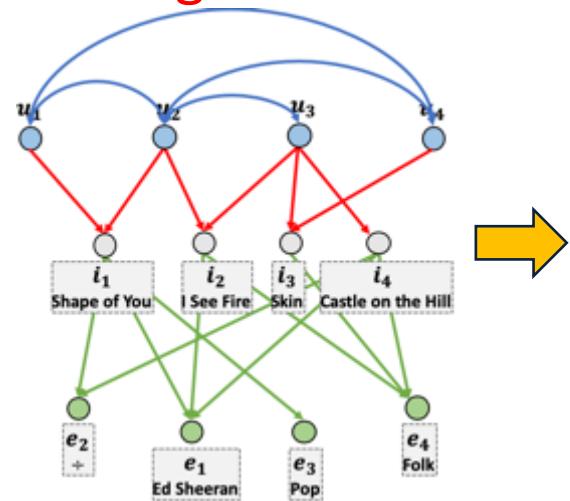
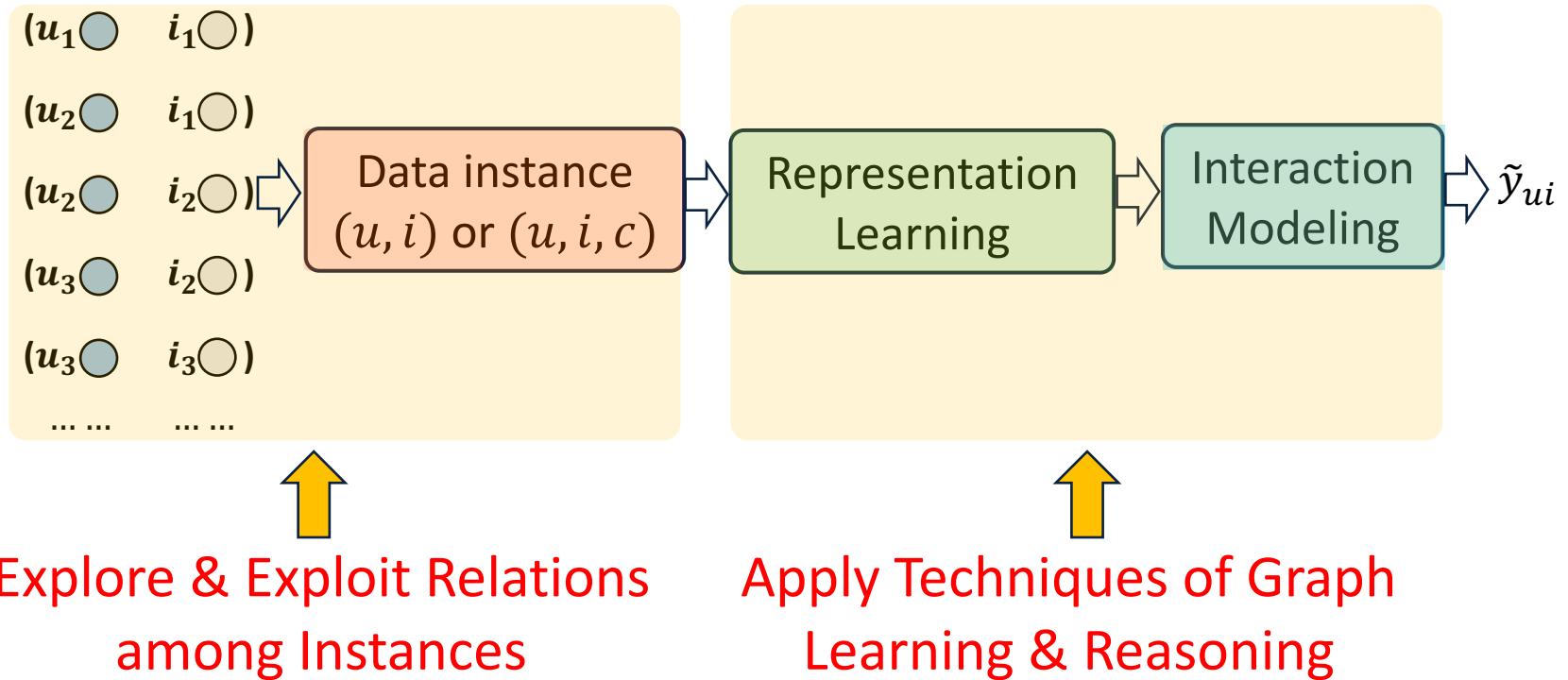
- Components work as a **black-box**
 - hardly exhibit the reasons behind a recommendation
 - Make the decision-making process opaque to understand

Why i_1 is recommended to u_1 ? Which one is more important?

- Collaborative Signals?
- Mouth Marketing?
- Item Knowledge?



How to Solve Such Issue?



References

- Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*
- Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.
- Stuart James Fitz-Gerald and Bob Wiggins. Introduction to modern information retrieval, 3rd ed., C.G. chowdhury. facit publishing, london (2010). *Int J. Information Management*, 30(6):573–574, 2010
- Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.
- Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, no. 8, pp. 3037, 2009.
- S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, ser. *UAI ’09*, 2009, pp. 452–461.
- S. Kabbur, X. Ning, and G. Karypis, “FISM: factored item similarity models for top-n recommender systems,” in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, 2013, pp. 659–667.
- Xia Ning, George Karypis: SLIM: Sparse Linear Methods for Top-N Recommender Systems. *ICDM 2011*: 497-506
- Yehuda Koren: Factorization meets the neighborhood: a multifaceted collaborative filtering model. *KDD 2008*: 426-434
- S. Rendle, “Factorization machines,” in *IEEE 10th International Conference on Data Mining*, 2010, pp. 995–1000.
- Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, Chih-Jen Lin: Field-aware Factorization Machines for CTR Prediction. *RecSys 2016*: 43-50
- X., X. Du, X. Wang, F. Tian, J. Tang, and T. Chua, “Outer product-based neural collaborative filtering,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018, pp. 2227–2233.
- H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, “Deep matrix factorization models for recommender systems,” in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. *IJCAI ’17*, 2017, pp. 3203–3209.
- X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th International Conference on World Wide Web*, ser. *WWW ’17*, 2017, pp. 173–182.

References

- J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, “Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention,” SIGIR ’17, 2017
- X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, “Nais: Neural attentive item similarity model for recommendation,” TKDE, 2018.
- F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, “Deep itembased collaborative filtering for top-n recommendation,” TOIS, vol. 37, no. 3, p. 33, 2019.
- H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir et al., “Wide & deep learning for recommender systems,” DLRS ’16, 2016, pp. 7–10.
- H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “Deepfm: A factorization-machine based neural network for CTR prediction,” IJCAI, 2017, pp. 1725–1731.
- J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, “xdeepfm: Combining explicit and implicit feature interactions for recommender systems,” KDD, 2018, pp. 1754–1763.
- Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao, “Deep crossing: Web-scale modeling without manually crafted combinatorial features,” KDD pp. 255–262.
- P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” RecSys ’16, 2016, pp. 191–198.
- X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, “Tem: Treeenhanced embedding model for explainable recommendation,” WWW, 2018, pp. 1543–1552.
- R. Wang, B. Fu, G. Fu, and M. Wang, “Deep & cross network for ad click predictions,” in Proceedings of the ADKDD’17, ser. ADKDD’17, 2017, pp. 12:1–12:7.
- Weinan Zhang, Tianming Du, Jun Wang: Deep Learning over Multi-field Categorical Data - - A Case Study on User Response Prediction. ECIR 2016: 45-57
- Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, Jun Wang: Product-Based Neural Networks for User Response Prediction. ICDM 2016: 1149-1154

OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- **Part II: Random Walk for Recommendation**
 - Random Walk
 - Absorption, ItemRank, TriRank, Pixie, RecWalk
- Part III: Network Embedding for Recommendation
- Part III: Graph Neural Networks for Recommendation

Slides in <https://next-nus.github.io/>

Random Walk

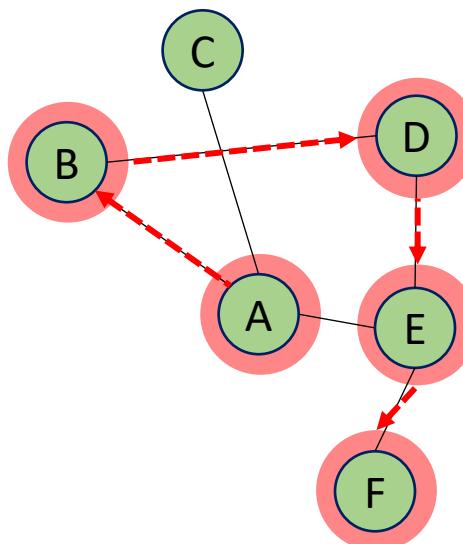
Graph Data G :

- V is the vertex set
- $E = V \times V$ is the edge set

$$p_{ij} = P(v_{k+1} = j | v_k = i) = \begin{cases} \frac{1}{|N_i|}, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases}$$

Random Walk → exhibit high-order proximity among nodes

1. Given an initial vertex (node) v_0 , select randomly an adjacent node v_1 ;
2. Move to this neighbor v_1 and treat v_1 as the starting node;
3. Repeat Steps 1& 2.



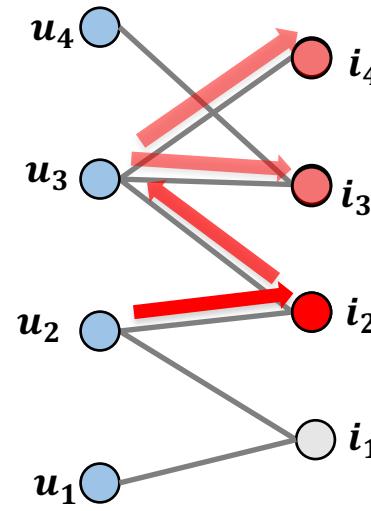
$A \rightarrow B \rightarrow D \rightarrow E \rightarrow F$

Motivation —— Preference Propagation

$(u_1 \circlearrowleft i_1 \circlearrowright)$	5
$(u_2 \circlearrowleft i_1 \circlearrowright)$	3
$(u_2 \circlearrowleft i_2 \circlearrowright)$	4
$(u_3 \circlearrowleft i_2 \circlearrowright)$	1
$(u_3 \circlearrowleft i_3 \circlearrowright)$	2
$(u_3 \circlearrowleft i_4 \circlearrowright)$	4
....

Interaction Data

Target User u_2



Interaction Graph

Item ranking $i_2 > i_3 \approx i_4$

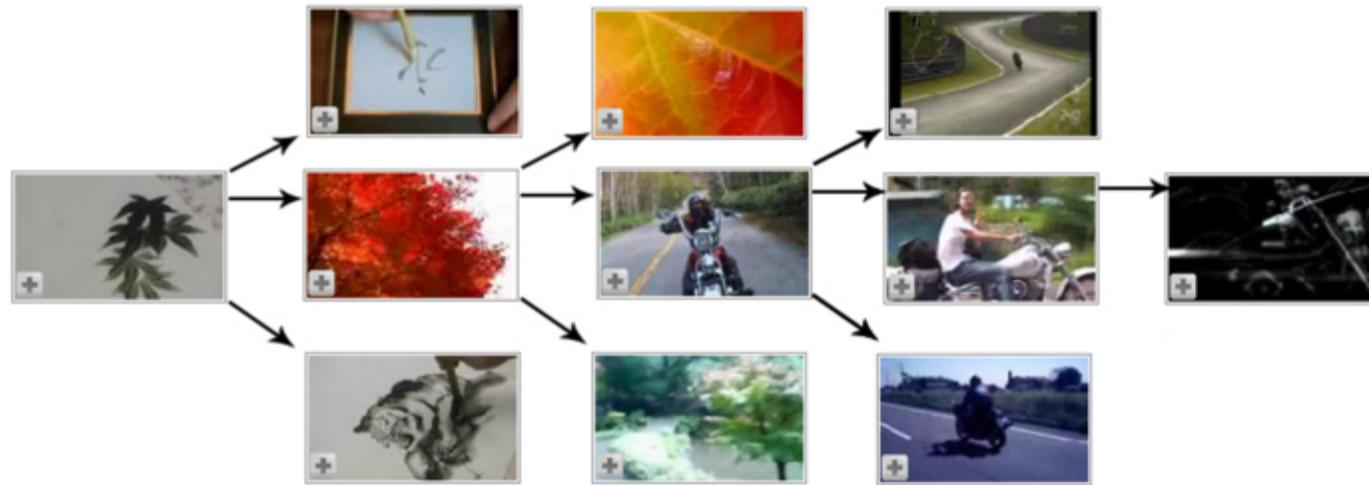
High-order Proximity → Label Propagation → Preference Distribution

- Label (preference) propagation from the target user's historical item nodes assigns unseen items with expected labels.

Absorption: Random Walk Through View Graph

Absorption from [Baluja et al, WWW'2008]:

- Interactions → **item-item co-viewed graph** or **user-item graph**
 - Edges → two video items are often co-viewed



Take a starting node v for a **random walk** & output a **label distribution**

$$L_v(\ell) = \sum_w \sum_u N_v(u) N_u(w) L_w(\ell)$$

The probability of reaching u from v in one random walk step

The probability of picking a neighbor w of u

ItemRank: A Random-Walk Scoring Algorithm

ItemRank from [Gori et al, IJCAI'2007]:

- Interactions → **item-item correlation graph**
 - Edges → the shared user groups
- Inspired by Classic **PageRank** [Kamvar et al., 2003a]:

Importance score
for every node

$$\mathbf{PR} = \alpha \cdot \mathbf{C} \cdot \mathbf{PR} + (1 - \alpha) \cdot \frac{1}{|\mathcal{V}|} \cdot \mathbf{1}_{|\mathcal{V}|}$$

Normalized connectivity
matrix for graph

Restart

- **ItemRank**

Preference score
for an item node
& user profile

$$\mathbf{IR}_{u_i} = \alpha \cdot \mathbf{C} \cdot \mathbf{IR}_{u_i} + (1 - \alpha) \cdot \mathbf{d}_{u_i}$$

User preference recorded
in training set → bias

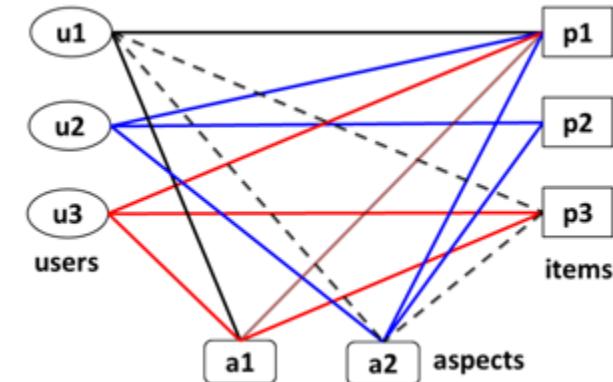
TriRank: Ranking over Tripartite Graph

TriRank from [He et al, CIKM'2015]:

- User-Item Interactions + **Item Aspects** → **Tripartite Graph**
 - User u previously rated item p with mentioning aspect a
- **Ranking score** for all nodes of a node
 - User-User → user similarity;
 - User-Aspect → interests on aspects
 - User-Item → preference on items
- **Smoothness Constraint**

$$Q_{UP}(f) = \sum_{i=1}^{|U|} \sum_{j=1}^{|P|} r_{ij} \left(\frac{f(u_i)}{\sqrt{d_i^u}} - \frac{f(p_j)}{\sqrt{d_j^p}} \right)^2$$

Inputs:
 $\langle u_1, p_1, a_1 \rangle$
 $\langle u_2, p_1, a_2 \rangle$
 $\langle u_2, p_2, a_2 \rangle$
 $\langle u_3, p_1, a_1 \rangle$
 $\langle u_3, p_3, a_1 \rangle$



Local consistency → ranking scores of nearby nodes should not vary too much

- **Fitting Constraint**

$$Q_P(f) = \sum_{j=1}^{|P|} (f(p_j) - p_j^0)^2$$

Ranking scores should adhere to the observations (i.e., initial values).

Pixie Random Walk

Pixie from [Eksombatchai et al, WWW'2018]:

- **Undirected Pin-Board Graph**
 - An edge between a pin p and a board b if a user saved p to b
- **Input:** a user-specific input query pin q
- **Output:** relevant pin p

Basic Random Walk:

- Simulate many random walks on G , starting from q
- record visit count for each candidate pin p
- The more often p is visited \rightarrow More related it is to q



More like this



Pixie Random Walk:

- Bias the random walk towards user-specific pins (q, U) \rightarrow personalized results for even the same query set
- Perform queries based on multiple pins $(q \in Q, U)$ each with a different importance \rightarrow consider the history of users

RecWalk: Nearly Uncoupled Random Walks

RecWalk from [Nikolakopoulos et al, WSDM'2019]:

- **Interactions → User-Item Bipartite Graph**

Adjacency matrix

$$\mathbf{A}_G \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix}$$

Interaction matrix

Transition probability matrix to govern the random sampler

$$\mathbf{P} \triangleq \alpha \mathbf{H} + (1 - \alpha) \mathbf{M}$$

1. **One based on bipartite graph**

$$\mathbf{H} \triangleq \text{Diag}(\mathbf{A}_G \mathbf{1})^{-1} \mathbf{A}_G.$$

2. **The other designed to capture item relations**

$$\mathbf{M} \triangleq \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_I \end{pmatrix}$$

Recommendation Strategies

$$\boldsymbol{\pi}_u^\top \triangleq \mathbf{e}_u^\top \mathbf{P}^K$$

- the probability the random walker lands on nodes after steps.

Summary: Random Walk for Recommendation

	Graph Data	Random Walk
Absorption	Item-item co-viewed graph	Basic
ItemRank	Item-item correlation graph	User-specific transition probability
TriRank	User-item- aspect tripartite graph	Smoothness & fitting constrains
Pixie	Pin-board graph	User-specific multi-pin transition probability
RecWalk	User-item bipartite graph	Basic + item relation-guided transition probability

Limitations

- **Efficiency Issue:**
 - for every user, generate ranking scores on all items each step → hard to apply large-scale graphs
- **Lack model parameters to optimize recommendation objective**
 - heuristic-based, rather than learning-based paradigm



OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- **Part III: Network Embedding for Recommendation**
 - **Network Embedding**
 - **HPE, HOP-Rec, CES**
- Part III: Graph Neural Networks for Recommendation

Slides in <https://next-nus.github.io/>

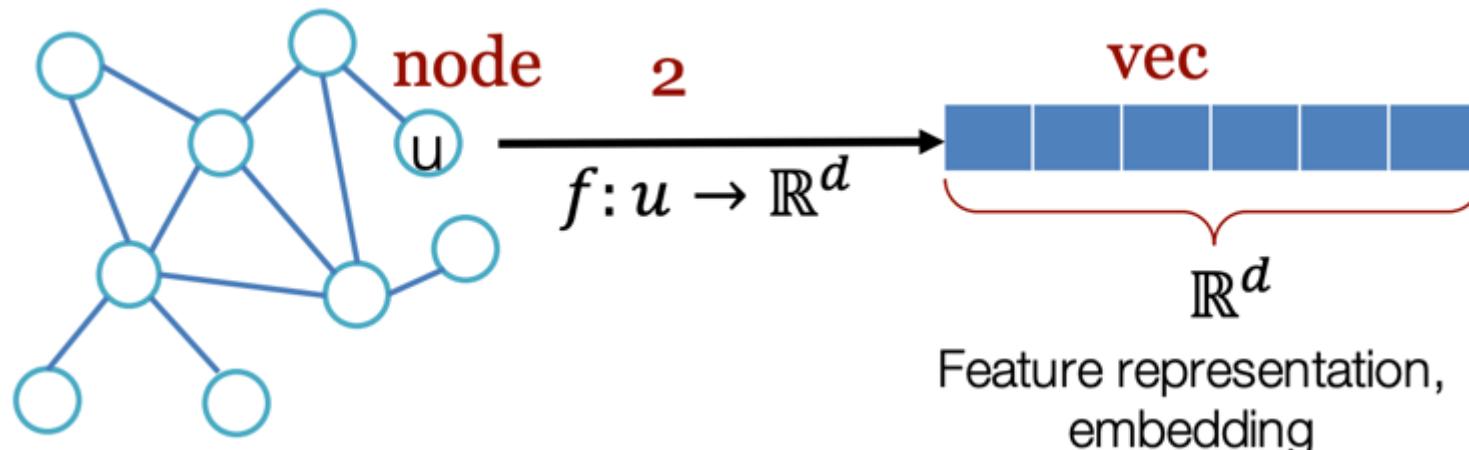
Network Embedding

Also well known as graph representation learning, node embedding, graph embedding.

Input: graph Data G

- V is the vertex set
- $E = V \times V$ is the edge set

Output: $Z \in R^{|V| \times d}$ latent feature representation matrix



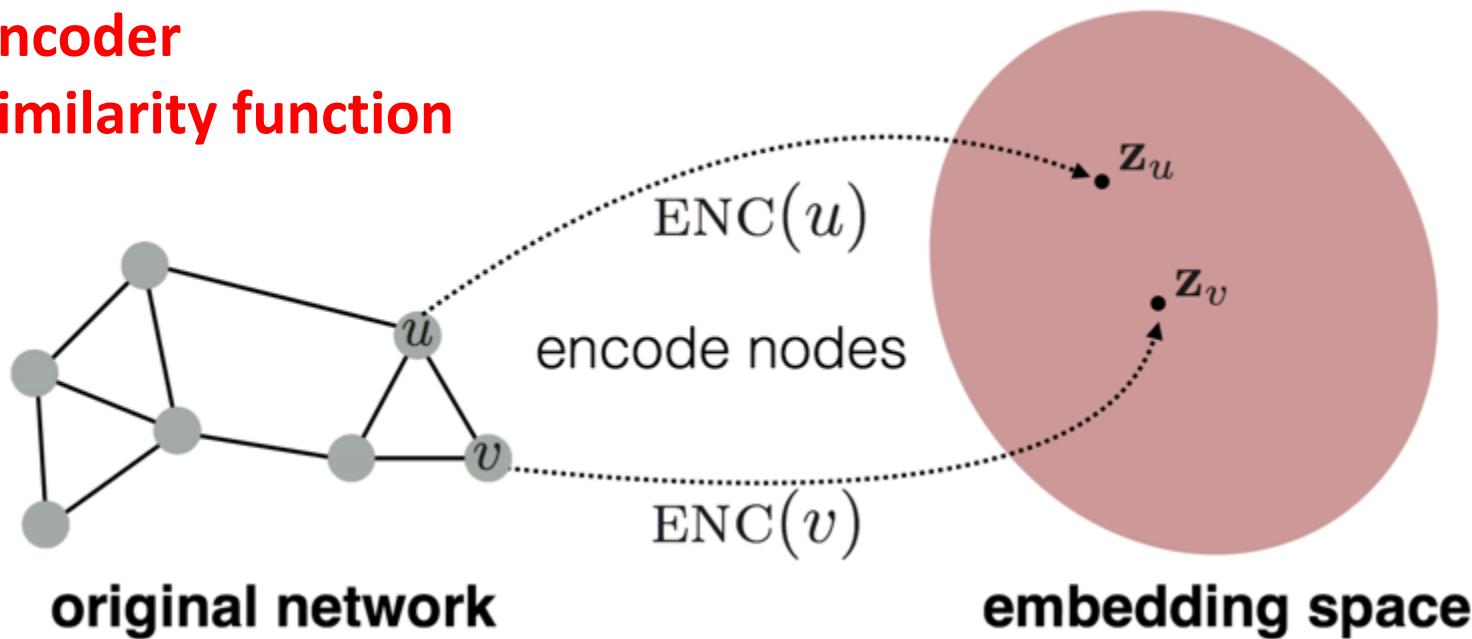
Intuition

Goal:

- Find embedding of nodes to d -dimensions
- **Similarity in the embedding space** approximates **similarity in the original network**

Need to define:

- **Encoder**
- **Similarity function**



Two Main Components

- **Encoder**

- To embed each node to a low-dimension vector representation

$$\text{ENC}(v) = z_v \quad \text{Node in input graph}$$

- **Similarity Function**

- To specify how relationships in the embedding space map to relationships in the original network.

$$\text{similarity}(v, u) = z_u^T z_v$$

Similarity in the original network

- Are connected?
- Share neighbors?
- Have similar structural roles?

Similarity in the embedding space

One- & Multi-hop Similarity

Similarity Function

- the edge weight between v and u in the original network.
- **One-hop Similarity** → Adjacency Similarity
 - e.g., [Ahmed et al. WWW'2013]

$$\mathcal{L} = \sum_{(u,v) \in E \times E} ||z_u^T z_v - A_{uv}||^2$$

All node pairs

(Weighted) adjacency matrix for input graph

- Only consider the existence of **direct connections**

- **Multi-hop Similarity** → overlap between node multi-hop neighbors
 - e.g., [Cao et al. CIKM'2015], [Ou et al. KDD'2016]

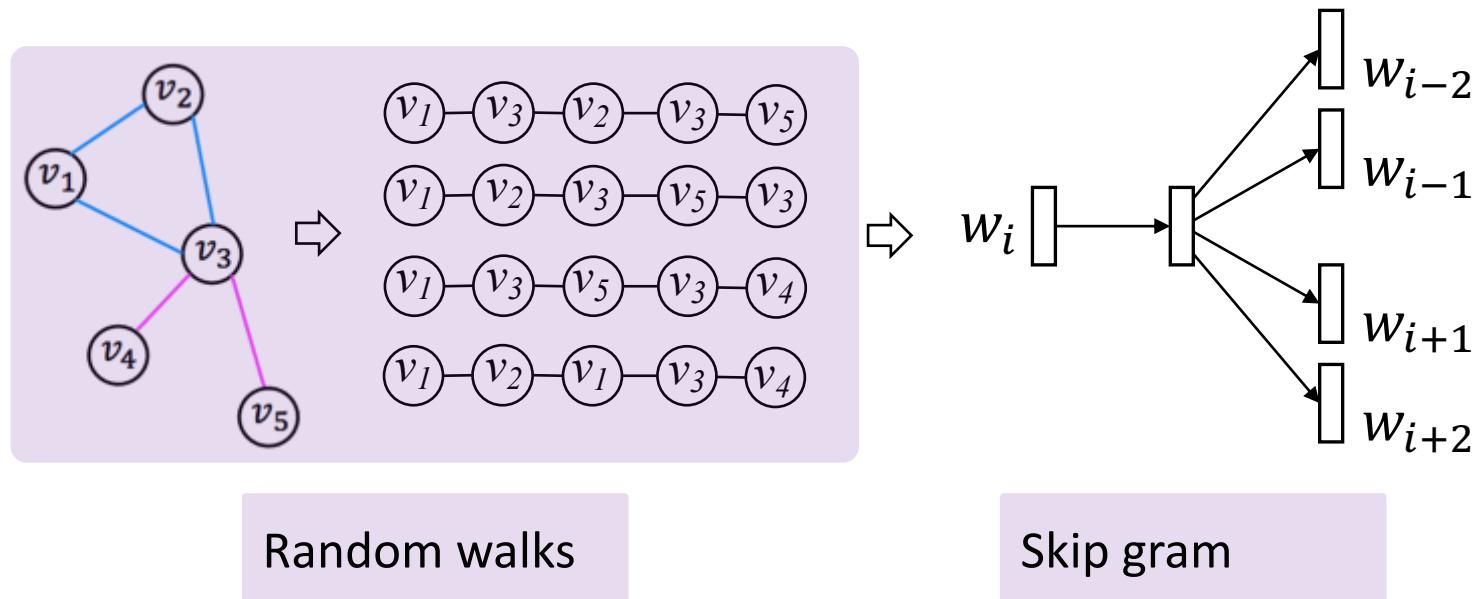
$$\mathcal{L} = \sum_{(u,v) \in E \times E} ||z_u^T z_v - S_{uv}||^2$$

Neighborhood overlap between nodes

Random Walk-based Similarity

Similarity Function

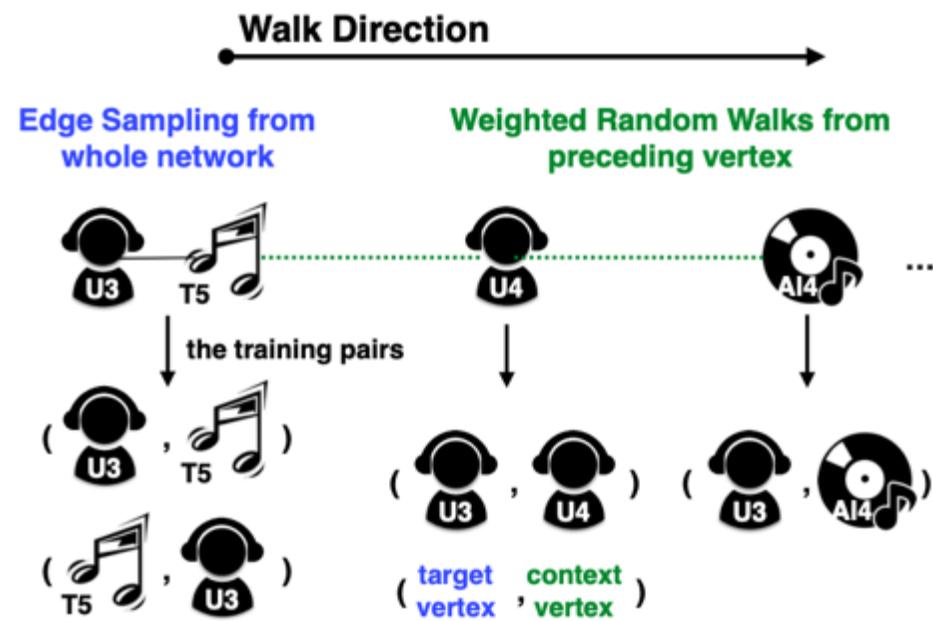
- **Probability that u and v co-occur on a random walk over the graph**
- E.g., DeepWalk, node2vec ...



Heterogeneous Preference Embedding (HPE)

HPE from [Yang et al, RecSys'2016]

- Interactions + Side information → heterogeneous graph
- **Random walk similarity**
 - treat indirect user-item interactions as user context



$$Pr(v_j|\Phi(v_i)) = \begin{cases} 1 & \text{if } v_j \in \text{Context}(v_i) \\ 0 & \text{otherwise} \end{cases}$$

Indirect connected nodes as the contextual information of ego node

$$- \sum_{(i,j) \in S} w_{i,j} \log p(v_j|\Phi(v_i)) + \lambda \sum_i \|\Phi(v_i)\|^2$$

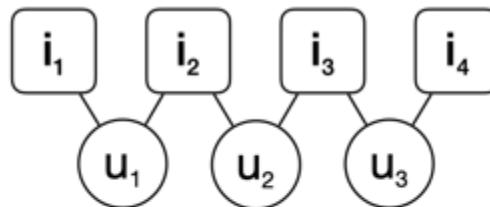
Random walk steps

Learn the preference embeddings via contextual nodes

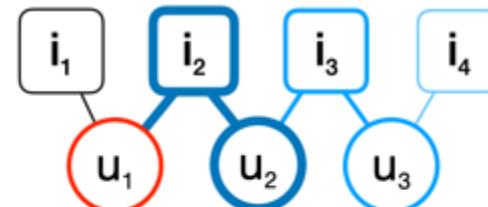
HOP-Rec: High-Order Proximity for Recommendation

HOP-Rec from [Yang et al, RecSys'2018]

- Interactions → user-item bipartite graph
- **Random walk similarity**
 - In a path, nodes with different orders → different confidence
 - Involve indirect user-item interactions into user preference



(a)



(b)

	i_1	i_2	i_3	i_4
U_1	1st	1st		
U_2		1st	1st	
U_3			1st	1st

(c)

	i_1	i_2	i_3	i_4
U_1	1st	1st	2nd	3rd
U_2	2nd	1st	1st	2nd
U_3	3rd	2nd	1st	1st

(d)

HOP-Rec: High-Order Proximity for Recommendation

HOP-Rec from [Yang et al, RecSys'2018]

$$\mathcal{L}_{HOP} = \sum_{\substack{1 \leq k \leq K \\ u, (i, i')}} \overbrace{C(k) \mathbb{E}_{\substack{i \sim P_u^k \\ i' \sim P_N}} \left[\mathcal{F}(\theta_u^\top \theta_{i'}, \theta_u^\top \theta_i) \right]}^{\text{graph model}} + \lambda_\Theta \|\Theta\|_2^2$$

A random walk with a decay factor for
confidence weighting $C(k)$

- For a given walk sequence, item with order k that the user potential prefers → **treated as positive instances**

factorization model

$$\mathcal{F}(\theta_u^\top \theta_{i'}, \theta_u^\top \theta_i)$$

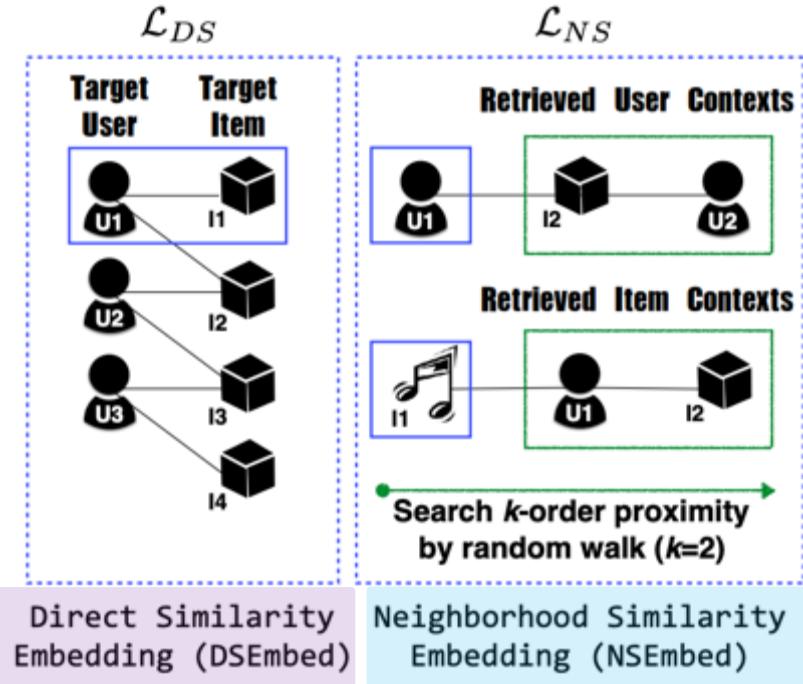
Matrix Factorization with BPR loss

- Random walk **enriches the positive observations**

Collaborative Similarity Embedding (CES)

CES from [Chen et al, WWW'2019]

- Interactions → user-item bipartite graph



$$\arg \max_{\Phi} \sum_{(v_i, v_j) \in E} \log p(v_i, v_j | \Phi)$$

Maximize the likelihood of observed pairs

$$\begin{aligned} & \arg \max_{\Phi, \Phi^{UC}, \Phi^{IC}} \prod_{(v_i, v_j) \in S_U} p(v_j | v_i; \Phi; \Phi^{UC}) \\ & + \prod_{(v_i, v_j) \in S_I} p(v_j | v_i; \Phi; \Phi^{IC}) \end{aligned}$$

- Conduct k-step random walk
- Get neighborhoods of a user (or item) pair
- Neighborhood proximity → user (or item) similarity

Summary: Network Embedding for Recommendation

	Graph Data	Connectivity/Proximity
HPE	Heterogeneous graph	Indirect connections
HOP-Rec	User-item bipartite graph	<ul style="list-style-type: none">• Direct connections• Indirect connections
CES	User-item bipartite graph	<ul style="list-style-type: none">• Direct similarity• Neighborhood proximity similarity

Limitations

- **Not end-to-end Learning:**
 - Random walk is conducted first to get multi-hop neighbors
- **Not fully explore high-order connectivity**
 - Multi-hop neighbors are used to enrich the training data, rather than directly contributing to the representation learning

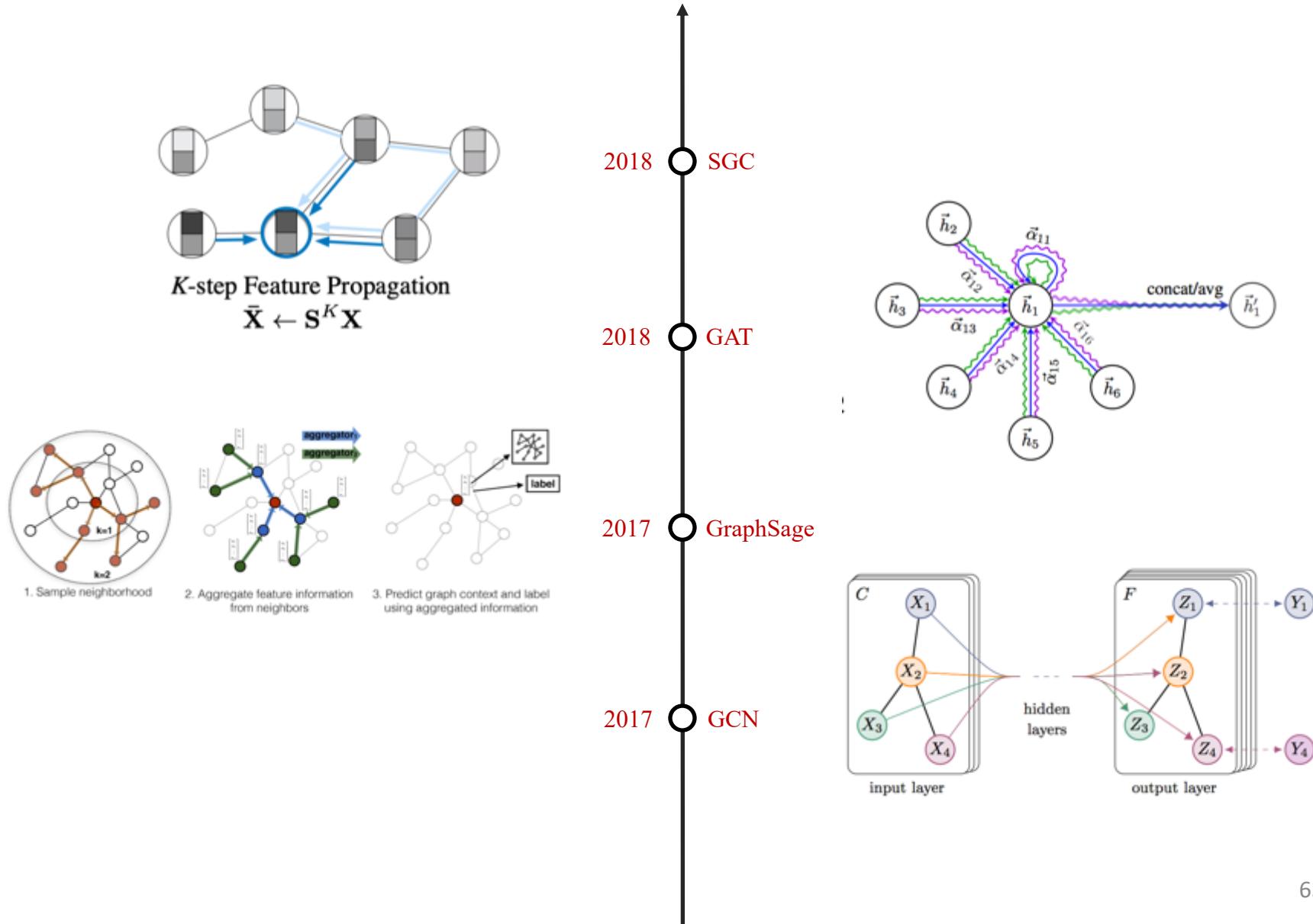


OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- **Part III: Graph Neural Networks for Recommendation**
 - **Graph Neural Networks**

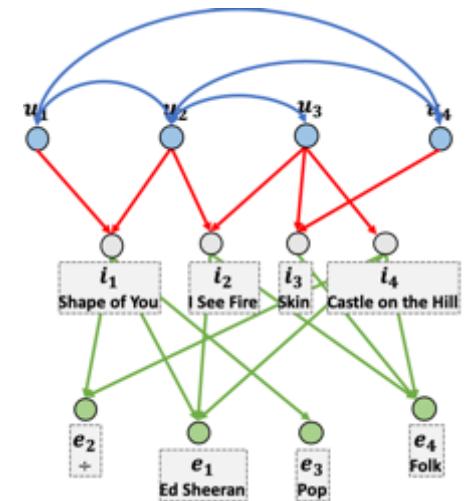
Slides in <https://next-nus.github.io/>

Recent Graph Neural Network (GNN) Research



Graph Data G

- V is the vertex set
- $E = V \times V$ is the edge set
 - Undirected: social relations, user-item interactions ...
 - Directed: triplets in knowledge graph
- $A \in R^{|V| \times |V|}$ is the adjacency matrix
 - $A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$
- $X \in R^{d \times |V|}$ is a matrix of node features
 - Categorical attributes, text, image data
 - Node degrees, clustering coefficient, ...
 - Indicator vectors (i.e., one-hot encoding of each node)
- $X' \in R^{d' \times |E|}$ is a matrix of edge features
 - Relations



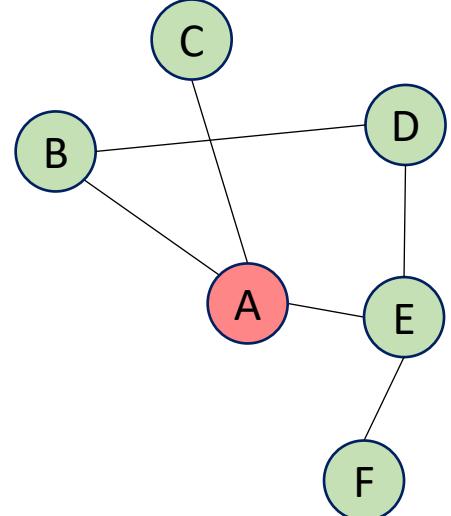
Graph Convolution Network (GCN)

- At the core of GCN
 - 1. Model a local structural information (neighborhood) of a node as the receptive field
 - 2. Apply the **graph convolution** operation
 - Spectral domain:
 - Laplacian eigen-decomposition [Bruna et al. ICLR'2014]
 - Chebyshev polynomials [Defferrard et al, NeurIPS'2016]
 - **However, computationally expensive**
 - Spatial domain:
 - Node (Neighborhood) aggregation [William et al, NeurIPS'2017]
 - 3. Update its representation.
 - $Z \in R^{d \times |V|}$ latent feature representation matrix

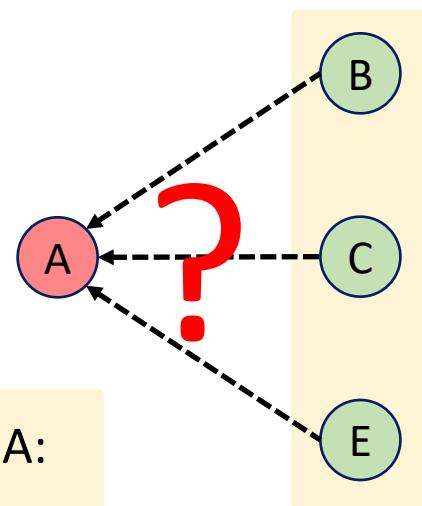
Neighborhood Aggregation (1)

Key Idea:

- Generate node embeddings based on local neighbors
- Neighborhood defines a computation graph.



Computation Graph of
node A

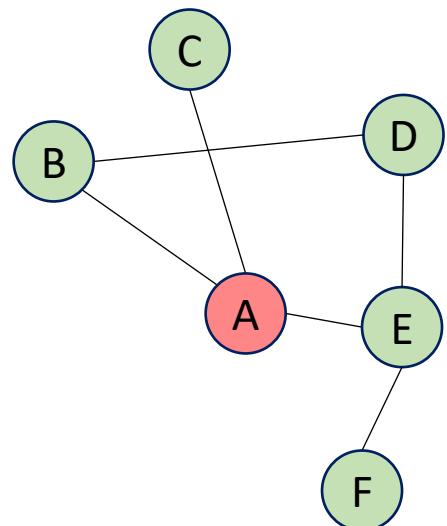


One-hop Neighborhood of node A:
 $N_A = \{B, C, E\}$

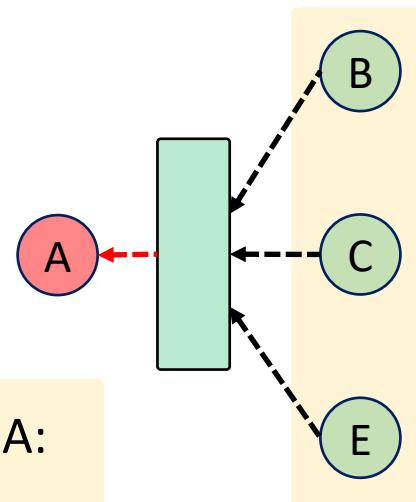
Neighborhood Aggregation (2)

Message Passing/ Information Propagation:

- A node aggregates information from their neighbors via neural networks



Computation Graph of
node A



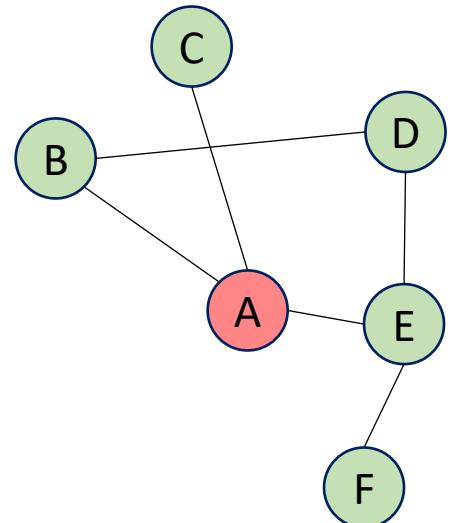
One-hop Neighborhood of node A:
 $N_A = \{B, C, E\}$

One-hop Neighborhood Aggregation:
 $\mathbf{h}'_A = f(\mathbf{h}_B, \mathbf{h}_C, \mathbf{h}_E)$

Neighborhood Aggregation (3)

Moreover:

- Every neighboring node has its own computation graph!

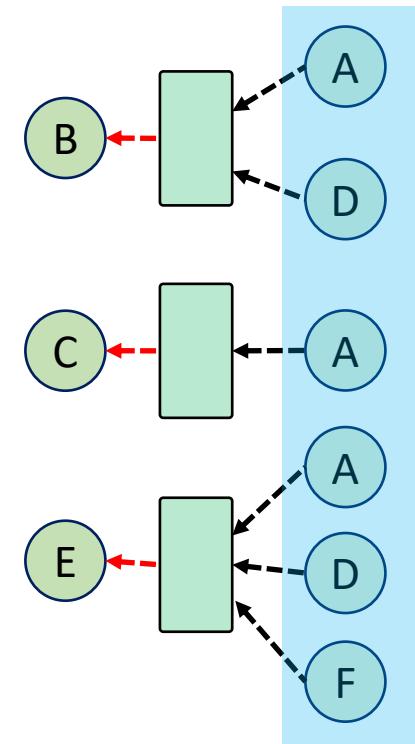


Computation Graph of
each node

Two-hop Neighborhood of node A:

$$N_A = \{D, F\}$$

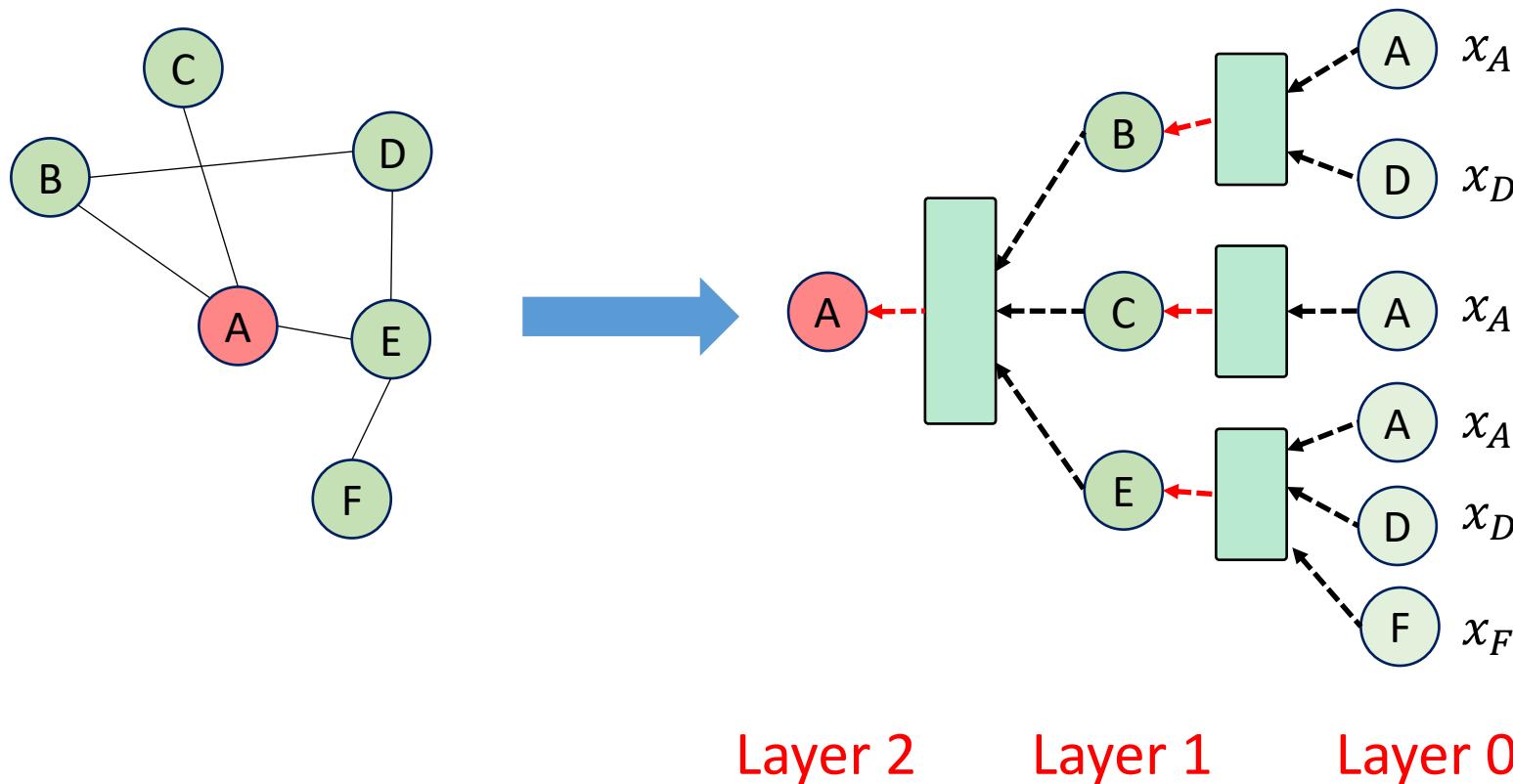
Unseen in one-hop
neighborhood



Neighborhood Aggregation (4)

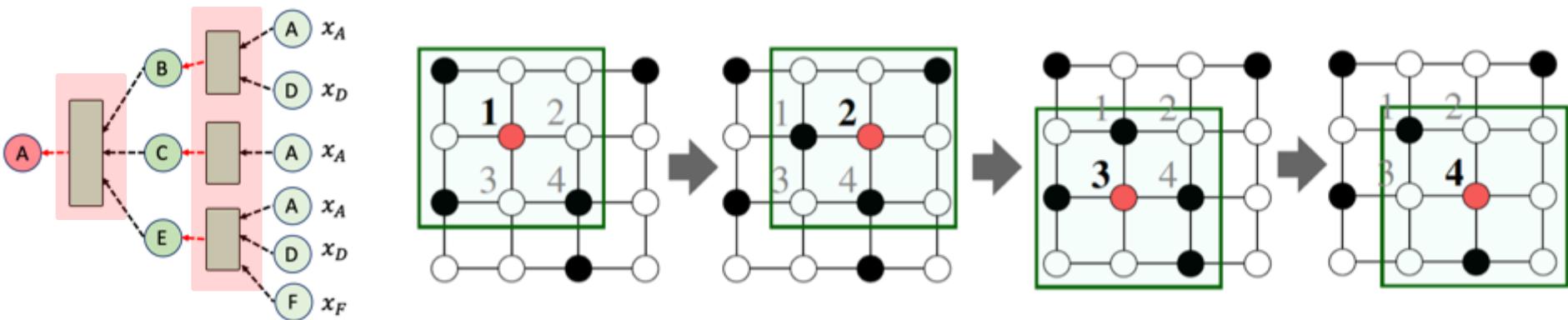
Stacking more neighborhood aggregation layers

- Nodes have embeddings at each layer
- Model can be arbitrary depth
- At Layer 0, embedding of node $v \in V$ is its input feature, i.e., x_v .



Graph Convolution

Neighborhood aggregation can be viewed as a center-surround filter in convolutional neural network (CNN).



Mathematically related to **spectral graph convolutions** (Bronstein et al., 2017)

Now

- How to aggregate information across layers!
- i.e., how to design the neural networks!

Component 1: Information Construction

Generally speaking, the **first** main component:

1. Information Construction

Construct the information being propagated from one neighboring node to the target node.

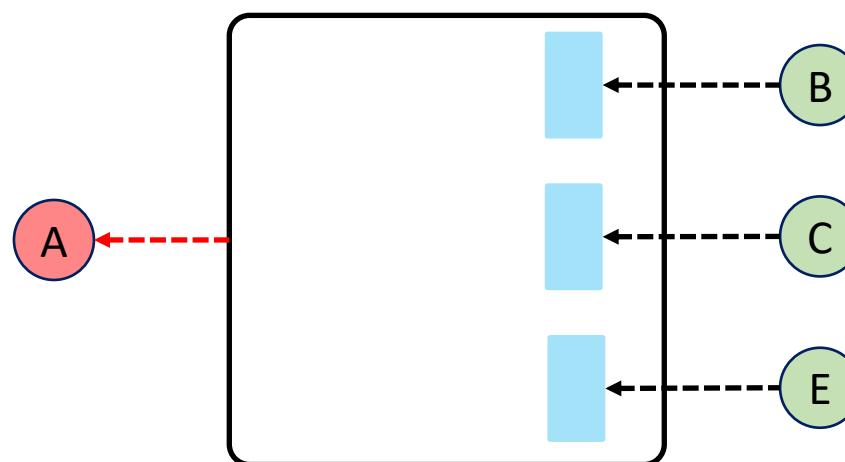
$$m_{v \rightarrow u}^{(k-1)} = f_1^{(k-1)}(h_v^{(k-1)}, h_u^{(k-1)}, p_{vu}^{(k-1)})$$

The information being propagated from node v to the target u

Neural network

Nodes' previous layer embeddings

Decay factor or normalization term



Component 2: Neighborhood Aggregation

Generally speaking, the **second** main component:

2. Neighborhood Aggregation

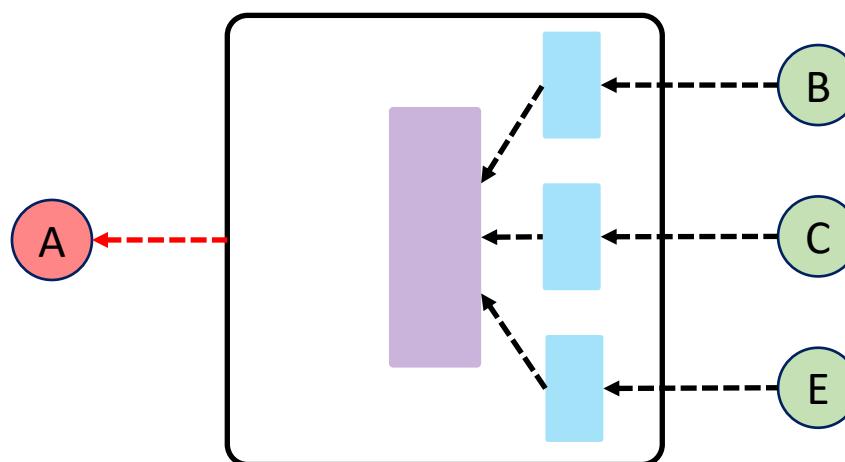
Aggregate the information from the whole neighborhood.

$$h_{\mathcal{N}_u}^{(k-1)} = f_2^{(k-1)} (\{ h_v^{(k-1)} \}, \text{for } v \in \mathcal{N}_u)$$

The information
combining neighbors'
information

Aggregation
Function

The information
obtained from the first
component



Component 3: Representation Update

Generally speaking, the **third** main component:

3. Representation Update

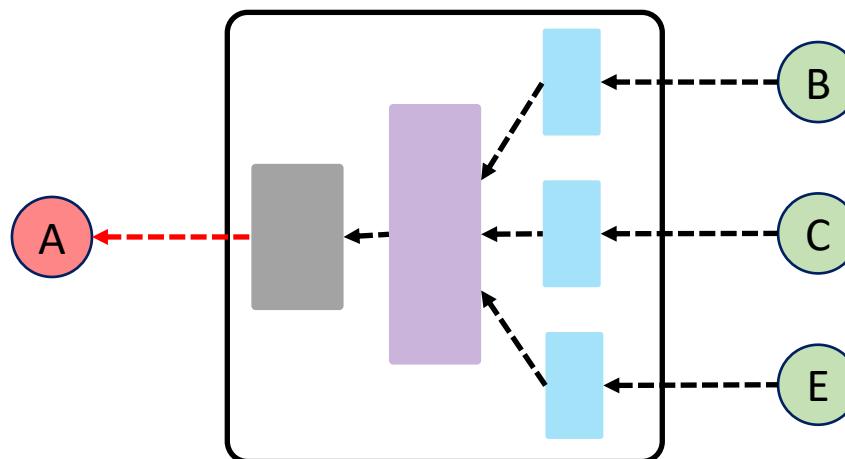
Integrate the neighborhood information with its own representation.

$$h_u^{(k)} = f_3^{(k-1)}(h_{\mathcal{N}_u}^{(k-1)}, h_u^{(k-1)})$$

The updated representation of u at the k -th layer

Integration Function

The representation of the target node u at the $(k - 1)$ -th layer



Graph Convolutional Network (GCN)

GCN from [Kipf et al., ICLR'2017]:

Nonlinear
activation function

$$h_u^{(k)} = \sigma \left(W_k \sum_{v \in \mathcal{N}_u \cup u} \frac{h_v^{(k-1)}}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_v|}} \right)$$

The same neural network
for self and neighbor
embeddings in **Comp. 3**

- More parameter sharing

Weighted sum:
Aggregation in
Comp. 2

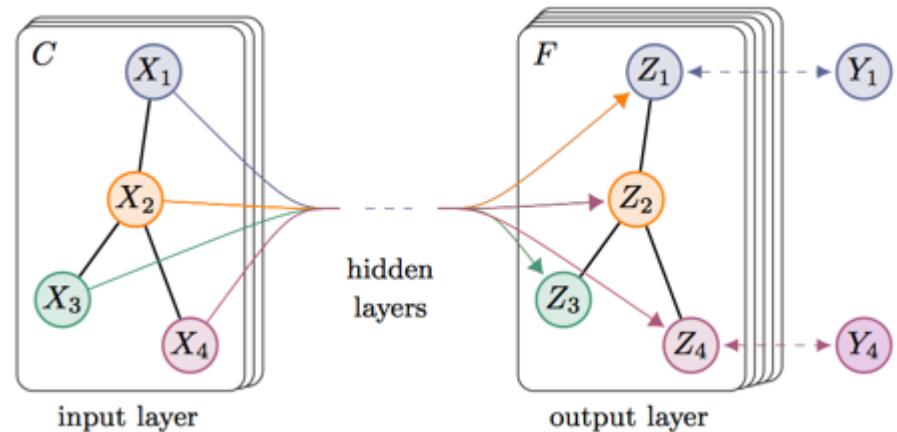
per-neighbor normalization:
 $p_{uv}^{(k-1)}$ in **Comp. 1**

- Normalization varies across neighbors
- Down-weights high degree neighbors

GCN in Matrix Form

Can be rewritten in the matrix form

- Which is efficiently implemented using sparse batch operations
- Time complexity is $O(|E|)$



Representation Matrix
at the k -th layer

$$H^{(k)} = \sigma(D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}H^{(k-1)}W_k) \rightarrow Z = H^{(K)}$$

Laplacian Matrix for Graph G

Final representation
matrix

GraphSage

GraphSage from [Hamilton et al., NeurIPS'2017]:

- The most distinction is the **generalized aggregation**

$$h_{\mathcal{N}_u}^{(k-1)} = f_2^{(k-1)} (\{ h_v^{(k-1)} \}, \text{for } v \in \mathcal{N}_u)$$

Generalized Aggregation Function in **Comp. 2**

→ Any differentiable function that maps a set of vectors to a single vector.



$$h_u^{(k)} = \sigma ([W_1^{(k)} \cdot \text{AGG} (\{ h_v^{(k-1)}, \forall v \in \mathcal{N}_u \}), W_2^{(k)} h_u^{(k-1)}])$$

Integration Function in **Comp.3** → concatenate
neighbors & self embeddings, instead of sum

Generalized Aggregation in GraphSage

- Mean

$$\text{AGG} = \sum_{\forall v \in \mathcal{N}_u} \frac{h_v^{(k-1)}}{|\mathcal{N}_v|}$$

- Pool

- Transform neighbor vectors and apply symmetric vector function.

$$\text{AGG} = \gamma(Q h_v^{(k-1)}, \forall v \in \mathcal{N}_u)$$

- LSTM

Element-wise mean/max

- Apply LSTM to random permutation of neighbors.

$$\text{AGG} = \text{LSTM}([h_v^{(k-1)}, \forall v \in \mathcal{N}_u])$$

Graph Attention Network (GAT)

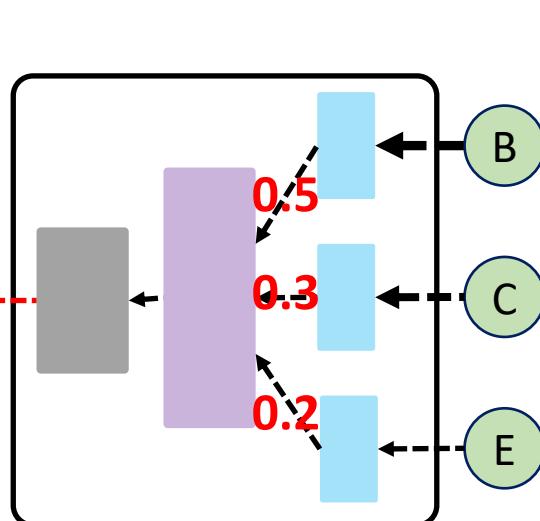
GAT from [Velickovic et al., ICLR'2018]:

- The most distinction is the **attentive neighborhood aggregation**

$$h_{\mathcal{N}_u}^{(k-1)} = f_2^{(k-1)} (\{ h_v^{(k-1)} \}, \text{for } v \in \mathcal{N}_u)$$

Attentive Aggregation Function in Comp. 1&2

- Different neighbors have varying contributions when propagating information.
- Instead of fixed heuristic-based decay factor like GCN, GraphSage



$$h_u^{(k)} = \sigma \left(\sum_{v \in \mathcal{N}_u} \alpha_{u,v} W^{(k)} h_v^{(k-1)} \right)$$

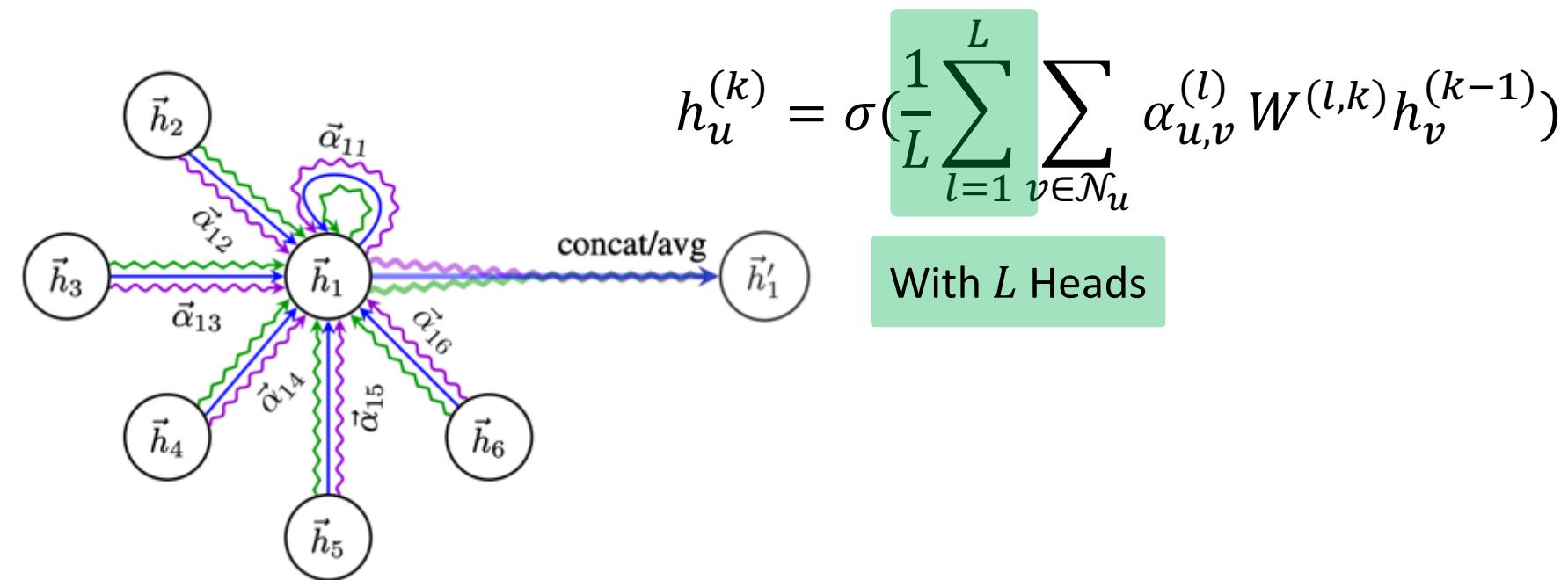
Learned attentive weights

Attention Weights in GAT

- Attention Network

$$\alpha_{v,u} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]))}{\sum_{u' \in N(v) \cup \{v\}} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]))}$$

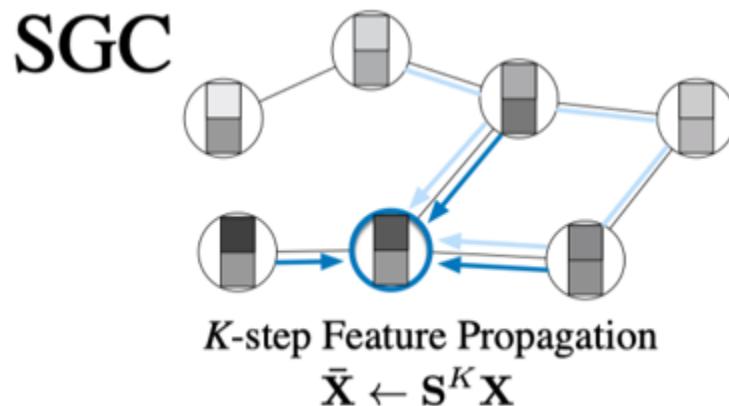
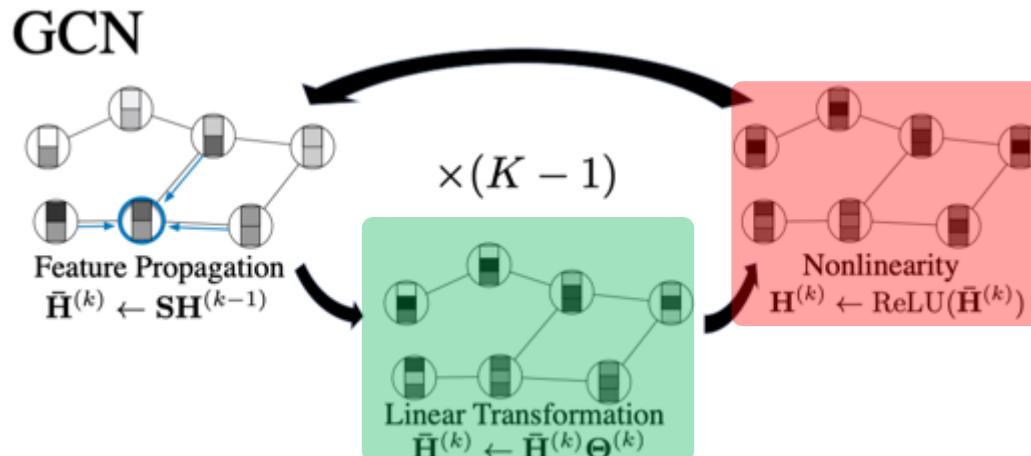
- Multi-head Attention



Simple Graph Convolution (SGC)

SGC from [Wu et al., ICML'2019]:

- **Unnecessary** complexity & redundant computation of GCN



Remove the nonlinearities between GCN layers

Use a single linear transformation

SGC in Matrix Form

- SGC improves the **efficiency** of GCN largely **without sacrificing accuracy**, & even outperforms GCN on some tasks.

GCN: $H^{(k)} = \sigma(D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}H^{(k-1)}W_k)$ $\rightarrow Z = H^{(K)}$

SGC: $H^{(k)} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}H^{(k-1)}$ $\rightarrow Z = H^{(K)}$

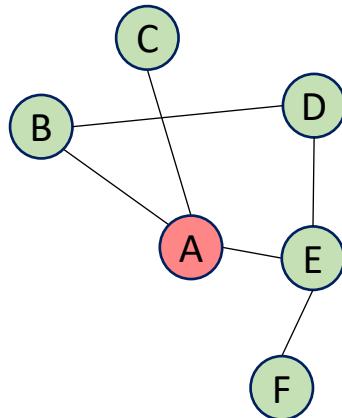
Only linear feature propagation is remained

Training

- After K graph convolution layers (e.g., GCN, GraphSage, GAT, SGC), we get output embeddings for each node.

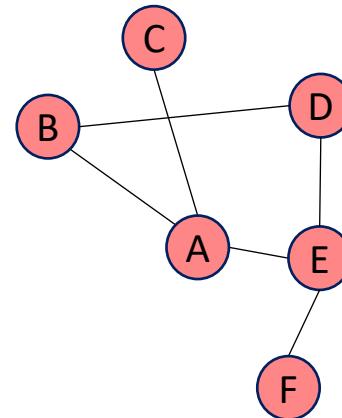
$$Z = H^{(K)}$$

- Upon these embeddings, we can define a loss function for a specific task:



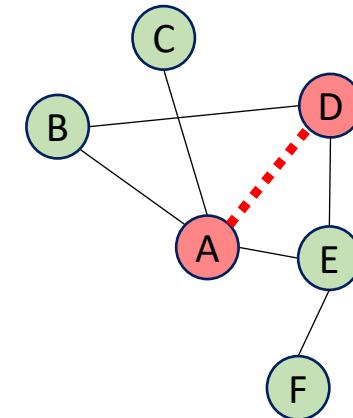
Node classification

$$\mathcal{L}(Z_u)$$



Graph classification

$$\mathcal{L}(Z_G)$$

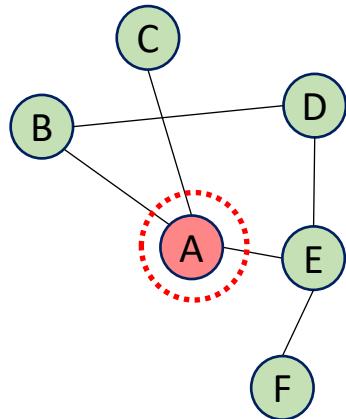


Link prediction

$$\mathcal{L}(Z_u, Z_v)$$

- Run stochastic gradient descent to train the aggregation parameters

e.g., Node Classification Task



Estimate the label of the target node:

- positive or negative?
- Belonging to one of C classes

$$\mathcal{L} = \sum_{v \in V} y_v \log(\sigma(z_v^T \theta)) + (1 - y_v) \log(1 - \sigma(z_v^T \theta))$$

Ground-truth label

Trainable weights in the classifier

GNN embeddings can be plug-and-play & serve other semi-supervised & unsupervised & supervised tasks.

Recent Research on GNN

- More Details in **Previous Tutorials**:
 - Jure Leskovec et al. Representation Learning on Networks, Tutorial@WWW2018
 - Hamilton & Jie Tang. Graph Representation Learning, Tutorial@AAAI2019
 - Jie Tang et al. Representation Learning on Networks, Tutorial@WWW2019
- More Details in **Survey Papers**:
 - Zhou et al., Graph Neural Networks: A Review of Methods and Applications
 - Zhang et al., Deep Learning on Graphs: A Survey
 - Wu et al., A Comprehensive Survey on Graph Neural Networks
- More **Paper Collections** in Github:
 - <https://github.com/thunlp/GNNPapers#survey-papers>
 - <https://github.com/naganandy/graph-based-deep-learning-literature>

OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- **Part III: Graph Neural Networks for Recommendation**
 - **Collaborative Filtering: GC-MC, SpectralCF, NGCF**

Slides in <https://next-nus.github.io/>

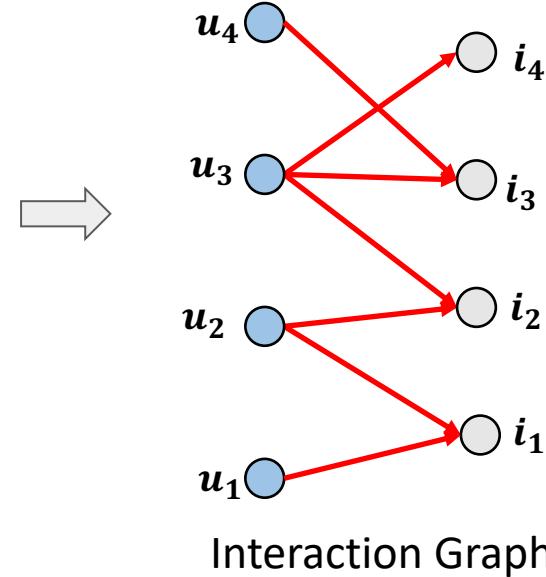
Recap Collaborative Filtering (CF)

- Collaborative Signals → Behavior Similarity of Users
 - Similar users would have similar preference on items.
- User-Item Interaction Data → User-Item Bipartite Graph
 - Edges indicate the user behaviors.

$(u_1 \bigcirc \quad i_1 \bigcirc) \quad 5$
 $(u_2 \bigcirc \quad i_1 \bigcirc) \quad 3$
 $(u_2 \bigcirc \quad i_2 \bigcirc) \quad 4$
 $(u_3 \bigcirc \quad i_2 \bigcirc) \quad 1$
 $(u_3 \bigcirc \quad i_3 \bigcirc) \quad 2$
 $(u_3 \bigcirc \quad i_4 \bigcirc) \quad 4$
...

		item				
		1	2	3	4	
user	1	5	?	?	?	...
	2	3	4	?	?	...
3	?	1	2	4		...
...

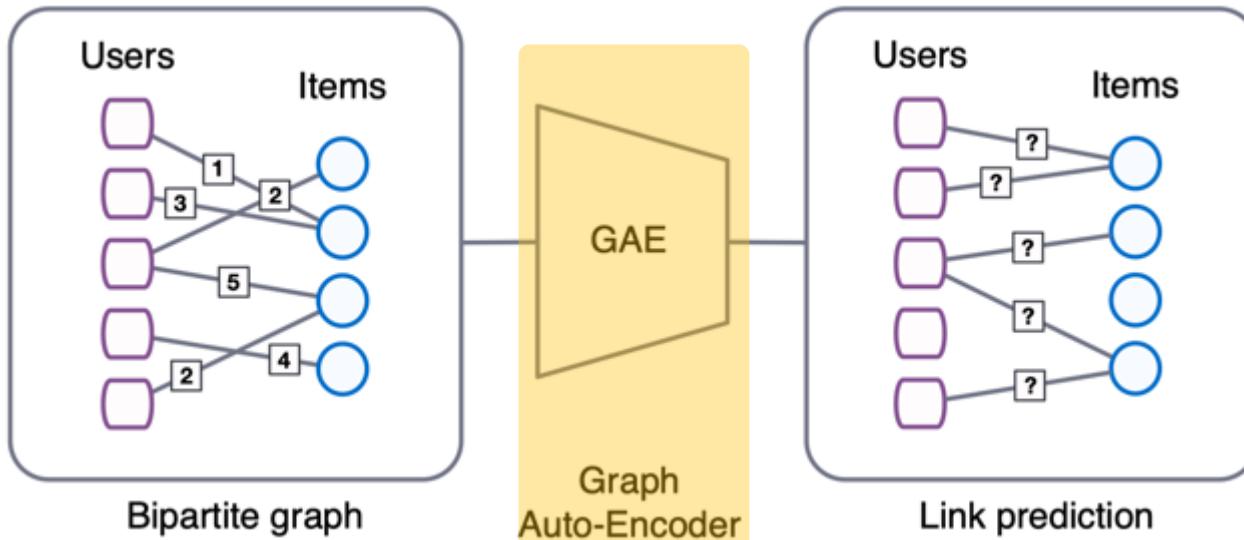
Interaction Matrix



Graph Convolutional Matrix Completion (GC-MC)

GC-MC from [Rianne et al., KDD'2018]

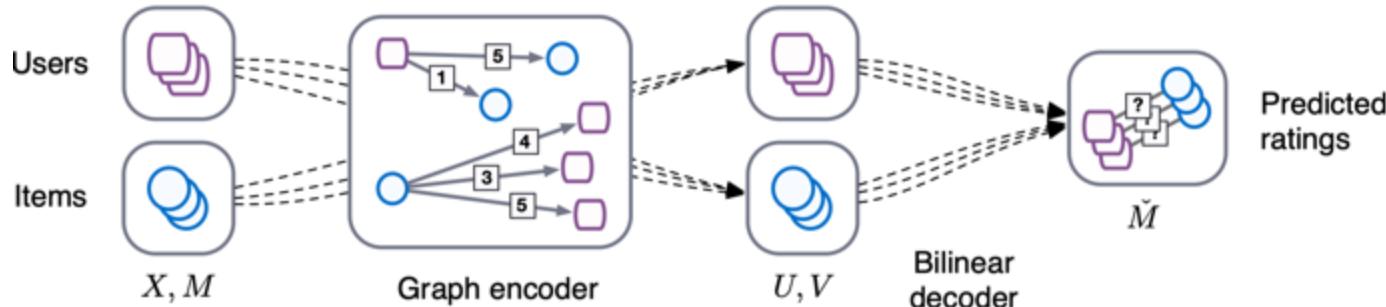
- View matrix completion as link prediction on interaction graph
 - Rating Prediction → predict links in bipartite user-item graph



Generate high-quality embeddings of users and items on the graph **in an end-to-end fashion**

- Previous solutions separate the graph feature model and link prediction model

Graph Convolutional Encoder in GC-MC



- Comp.1: Information Construction:

$$\mu_{j \rightarrow i, r} = \frac{1}{c_{ij}} W_r x_j^v$$

Different W_r are assigned to different rating level r .

- Comp.2: Neighborhood Aggregation:

$$h_i^u = \sigma \left[\text{accum} \left(\sum_{j \in \mathcal{N}_i(u_i)} \mu_{j \rightarrow i, 1}, \dots, \sum_{j \in \mathcal{N}_R(u_i)} \mu_{j \rightarrow i, R} \right) \right]$$

Accumulation operation over neighbors at all rating levels

- Comp.3: Representation Update:

$$z_i^u = \sigma(W h_i^u)$$

Rating Prediction in GC-MC

- Rating Prediction

$$p(\check{M}_{ij} = r) = \frac{e^{(z_i^u)^T Q_r z_j^v}}{\sum_{s=1}^R e^{(z_i^u)^T Q_s v_j}}$$

Trainable weights for different rating levels

$$\check{M}_{ij} = g(u_i, v_j) = \mathbb{E}_{p(\check{M}_{ij}=r)}[r] = \sum_{r \in R} r p(\check{M}_{ij} = r)$$

- Model Training

$$\mathcal{L} = - \sum_{i,j; \Omega_{ij}=1} \sum_{r=1}^R I[M_{ij} = r] \log p(\check{M}_{ij} = r)$$

Negative log likelihood

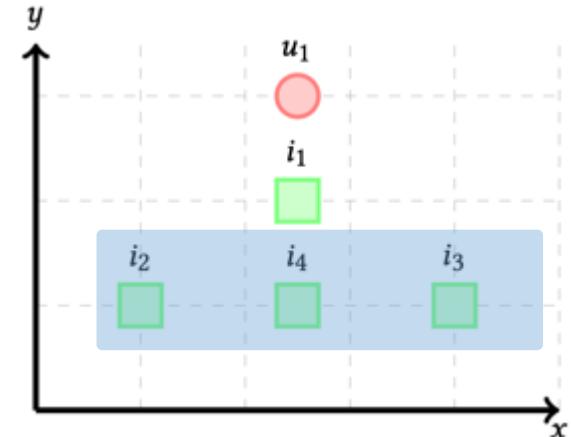
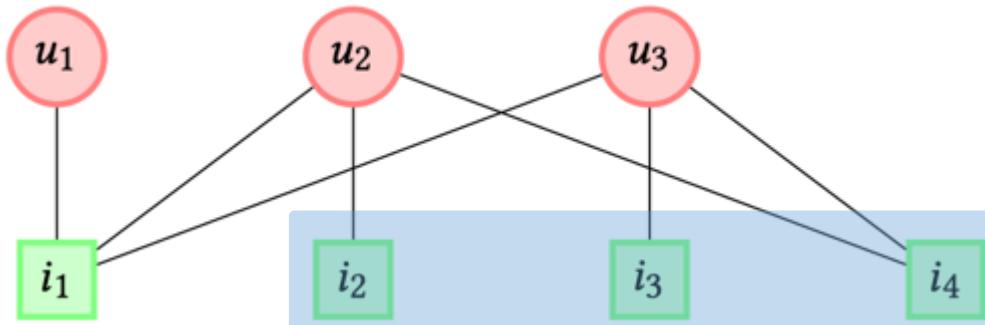
- Others:

- One graph convolution layer achieved the best performance.
- Structural information can be combined with interaction graph
 - Social networks, knowledge graphs, ...

Spectral Collaborative Filtering

SpectralCF from [Zheng et al., RecSys'2018]

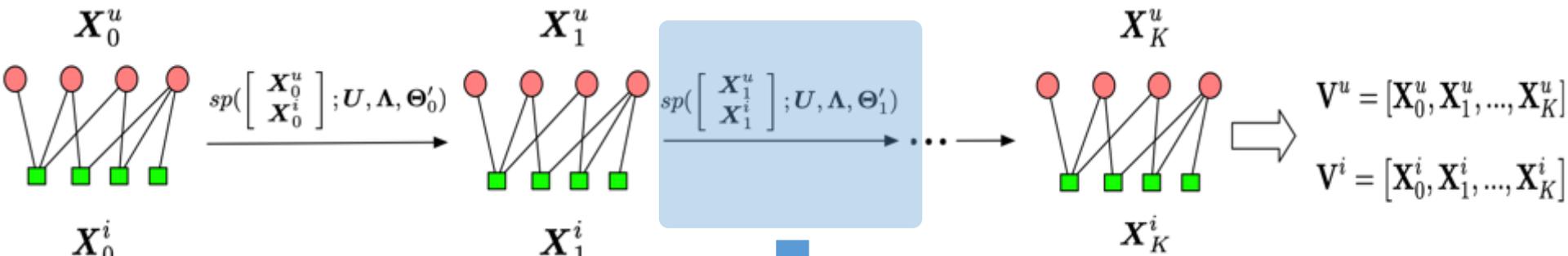
- User-Item Interaction Graph
 - GC-MC: use existing connectivity
 - SpectralCF: discover hidden connectivity in the spectral domain



The connectivity between u_1 and i_2, i_3, i_4

- Uncovered in the frequency domain
- Discovered in the spectral domain

Spectral Convolution Filtering in SpectralCF



Spectral Convolution Filtering

$$\begin{bmatrix} \mathbf{x}_{new}^u \\ \mathbf{x}_{new}^i \end{bmatrix} = \mathbf{U}g_{\theta}(\Lambda) \begin{bmatrix} \hat{\mathbf{x}}^u \\ \hat{\mathbf{x}}^i \end{bmatrix} = \mathbf{U}g_{\theta}(\Lambda)\mathbf{U}^\top \begin{bmatrix} \mathbf{x}^u \\ \mathbf{x}^i \end{bmatrix}$$

Polynomial Approximation

$$\begin{bmatrix} \mathbf{x}_{new}^u \\ \mathbf{x}_{new}^i \end{bmatrix} = \theta'(UU^\top + U\Lambda U^\top) \begin{bmatrix} \mathbf{x}^u \\ \mathbf{x}^i \end{bmatrix}$$

However, eigen-decomposition of graph adjacency matrix is required

- A rather high complexity
- Difficult to support large-scale graphs

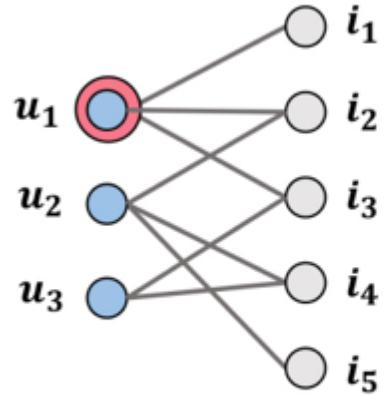
Neural Graph Collaborative Filtering (NGCF)

NGCF from [Wang et al., SIGIR'2019]

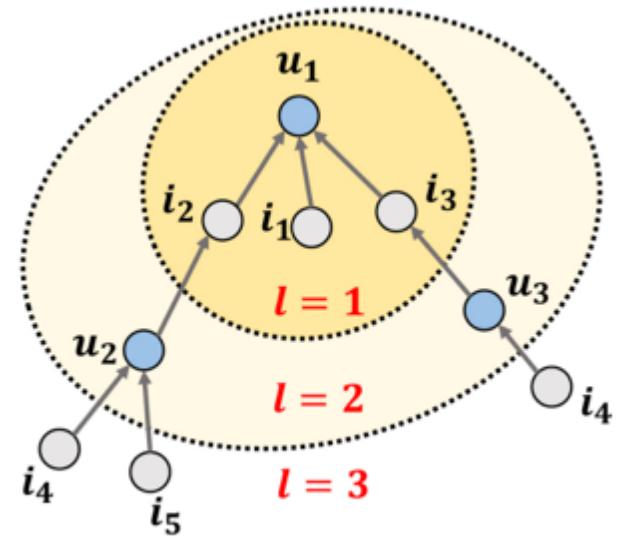
- Revisit CF via **high-order connectivity**
 - The paths that reach u_1 from any node with the path length l larger than 1 → **unseen connectivity argued in SpectralCF!**
 - A natural way to encode collaborative signal in the interaction graph structure

Why u_1 may like i_4

- $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$
- $u_1 \leftarrow i_3 \leftarrow u_3 \leftarrow i_4$



User-Item Interaction Graph



High-order Connectivity for u_1

First-order Connectivity Modeling

Inspired by GNNs

1. Propagate embeddings recursively on the user-item graph
 2. Construct **information flows** in the embedding space
-
- Comp.1: **Information Construction:**

message passed from i to u

$\mathbf{m}_{u \leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} (\mathbf{W}_1 \mathbf{e}_i + \mathbf{W}_2 (\mathbf{e}_i \odot \mathbf{e}_u))$

discount factor

- message dependent on the affinity, distinct from GCN, GraphSage, etc.
 - Pass more information to similar nodes
 - Comp.2 & 3: **Neighbor Aggregation & Representation Update:**

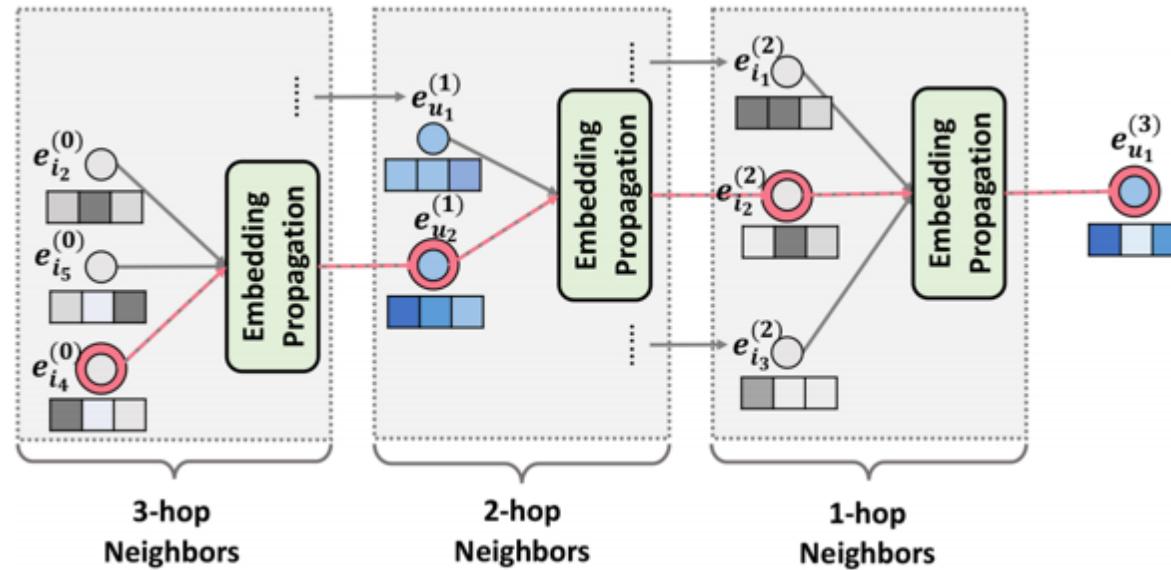
$$\mathbf{e}_u^{(1)} = \text{LeakyReLU} \left(\mathbf{m}_{u \leftarrow u} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i} \right)$$

self-connections

all neighbors of u

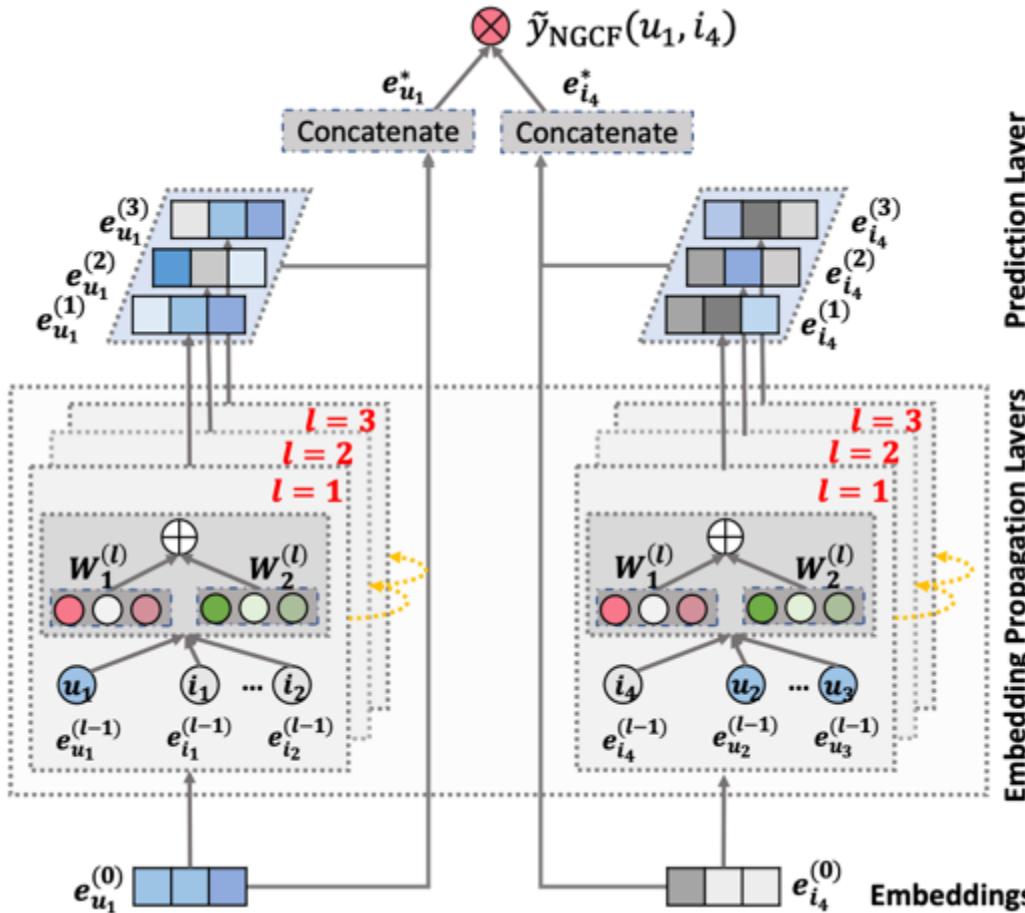
High-order Connectivity Modeling

- Stack more embedding propagation layers to explore the high-order connectivity



- The collaborative signal like $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ can be captured in the embedding propagation process.
- **Collaborative signal can be injected into the representation learning process.**

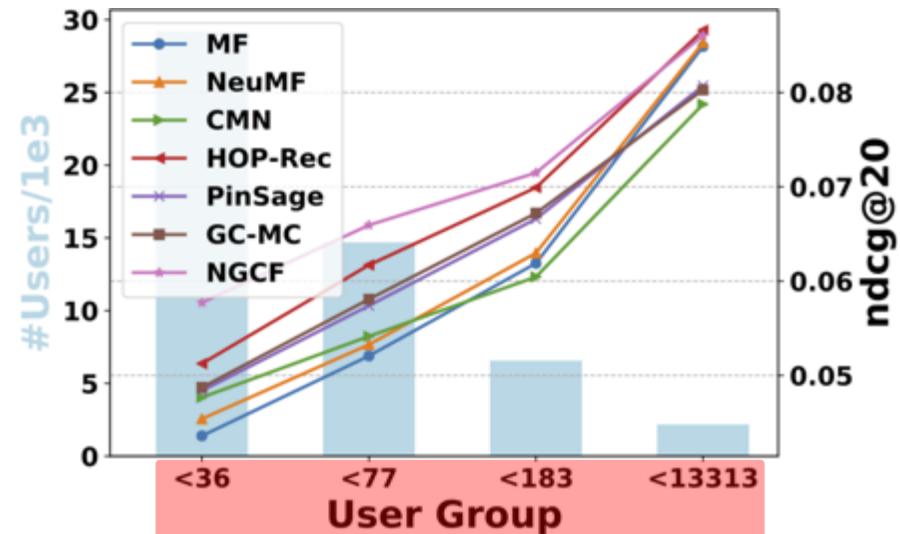
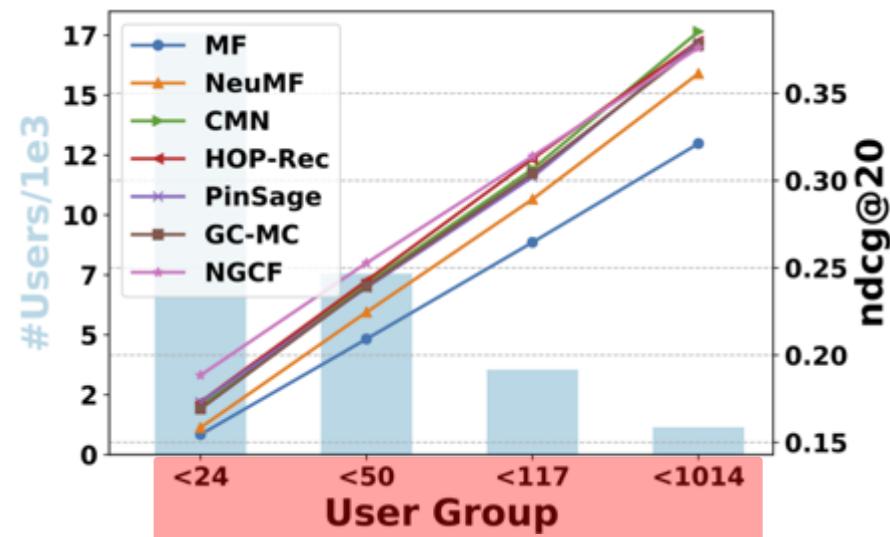
Overall Framework



The representations at different layers

- emphasize the messages passed over different connections
- have different contributions in reflecting user preference

Experiment Results —— Sparsity Issue



user groups with different group sparsity levels

- NGCF consistently outperforms all other baselines on most user groups.
- Exploiting high-order connectivity **facilitates the representation learning for inactive users.**
- **It might be promising to solve the sparsity issue in recommender systems**

OUTLINE

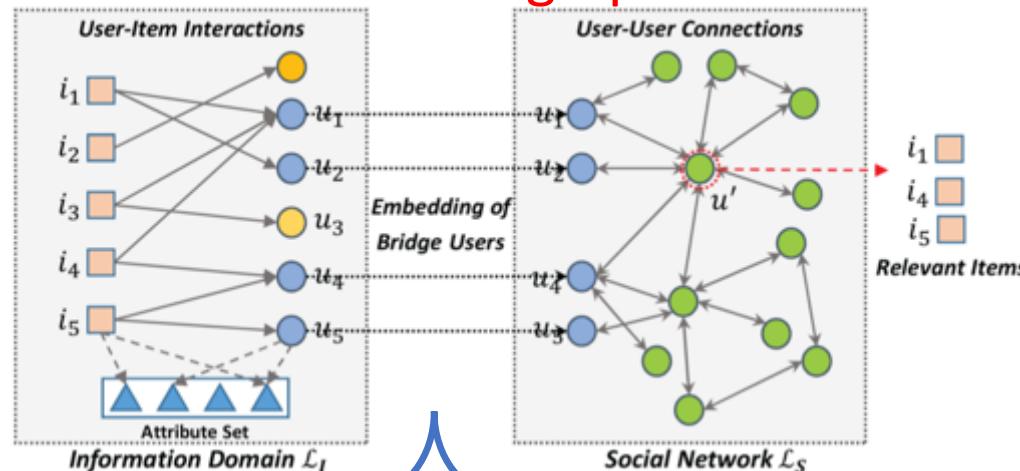
- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- **Part III: Graph Neural Networks for Recommendation**
 - **Social Recommendation: GraphRec, DiffNet, DANSEr**

Slides in <https://next-nus.github.io/>

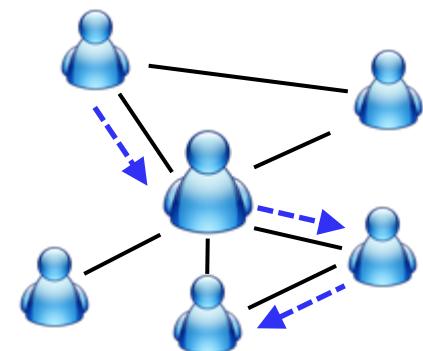
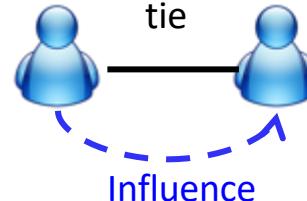
Social Recommendation

Social relation is of importance to help users filter information

- Two graphs → user-item interaction graph + user-user social graph.



Social Network Analysis



- User Profiling
- Behavior Modeling

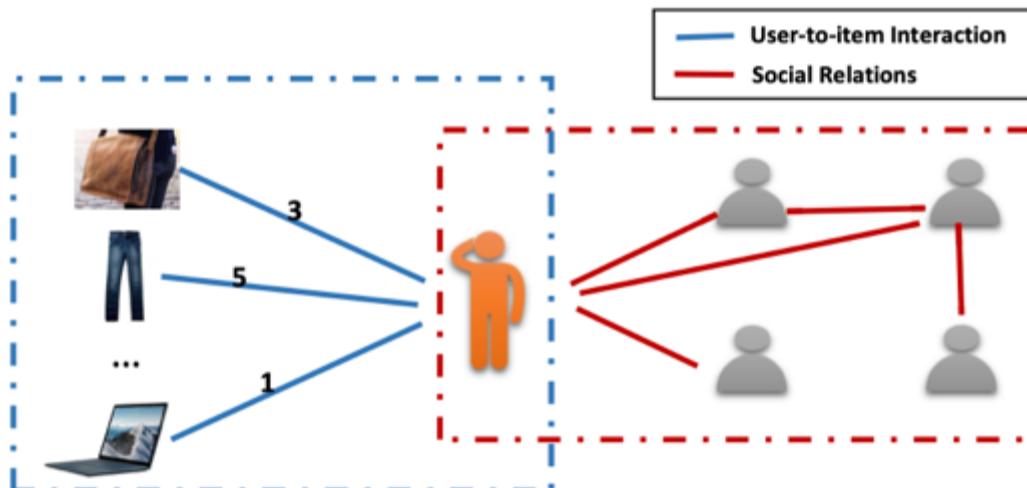
- Social Tie
- Social Influence

- Triad & Group Formation
- Information Diffusion

Graph Neural Networks for Social Recommendation (GraphRec)

GraphRec from [Fan et al, WWW'2019]

- User-Item Graph
 - Interactions between users and items
 - Users' opinions on items (i.e., explicit feedback, ratings)
- User-User Graph
 - Social relations have heterogeneous strengths
 - Strong & weak ties are mixed together
 - Users are likely to share more similar tastes with strong ties than weak ties.



User Modeling in GraphRec

These two graphs provide user information from different angles

- **Item Aggregation**

- Item space: leverage user-item interactions to get user representations

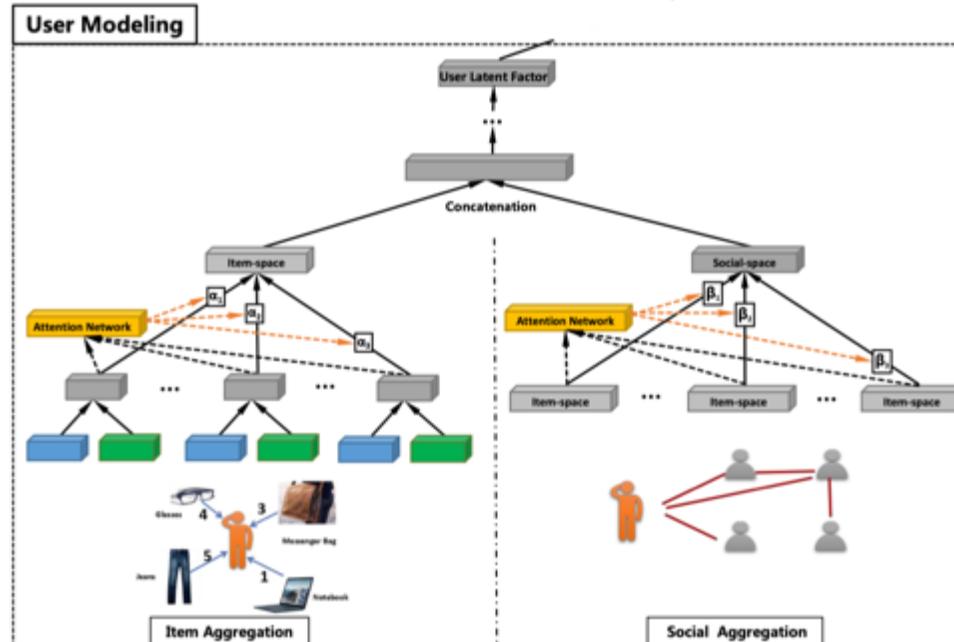
$$\mathbf{h}_i^I = \sigma(\mathbf{W} \cdot \text{Aggre}_{items}(\{\mathbf{x}_{ia}, \forall a \in C(i)\}) + \mathbf{b})$$

- **Social Aggregation**

Opinion-aware representation of an interaction

- Social space: use social relationships to get user representations

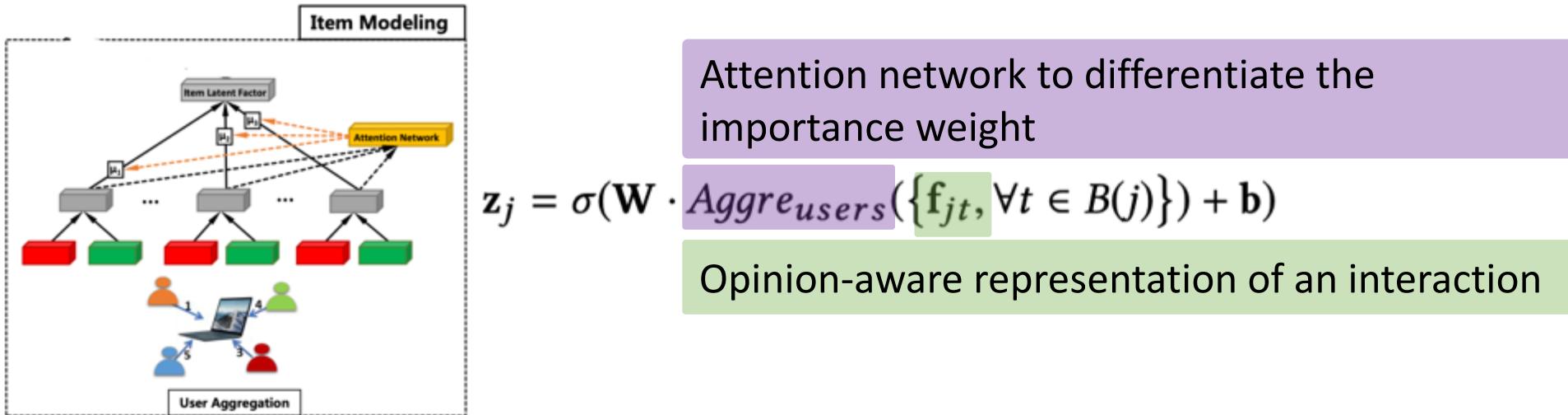
$$\mathbf{h}_i^S = \sigma(\mathbf{W} \cdot \text{Aggre}_{neighors}(\{\mathbf{h}_o^I, \forall o \in N(i)\}) + \mathbf{b})$$



Item Modeling in GraphRec

- **User Aggregation**

- Consider both interactions & opinions to get item representations



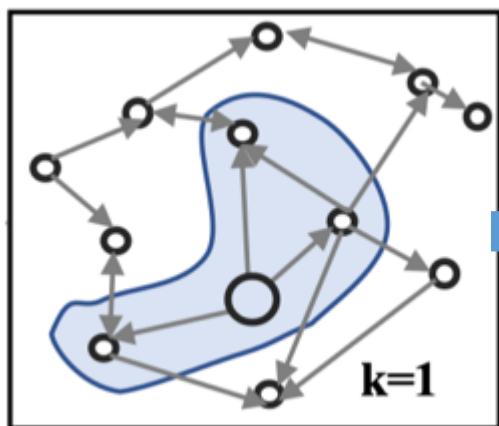
- **Rating Prediction**

- Feed the concatenation of user & item representation into a neural network (MLP) to get predictions.

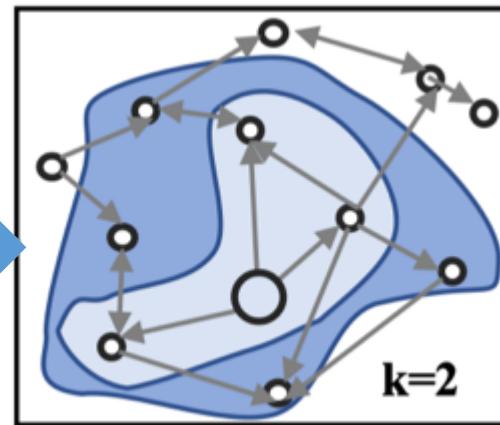
Neural Influence Diffusion Model (DiffNet)

DiffNet from [Wu et al, SIGIR'2019]

- **Social Influence** in Social Recommendation
 - A user's preference is influenced by her trusted users, with these trusted users are influenced by their own social connections → **high-order connectivity**
 - **Social influence recursively propagates & diffuses in social network!**

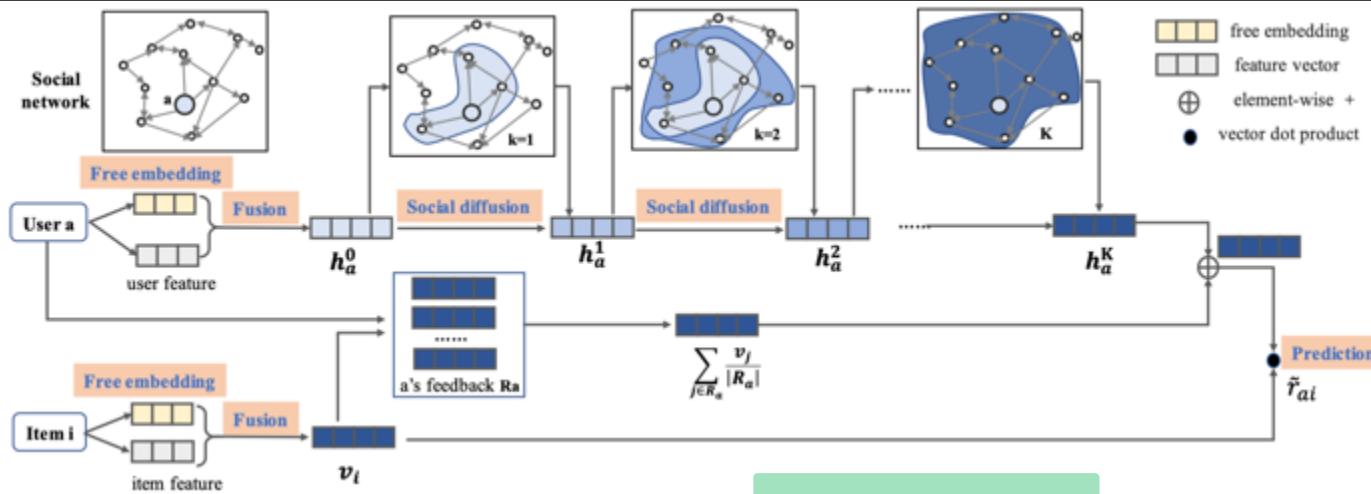


One-hop trusted users



Two-hop trusted users

Diffusion Influence Aggregation in DiffNet



- Comp.1: Information Construction User features

$$\mathbf{h}_a^0 = g(\mathbf{W}^0 \times [\mathbf{x}_a, \mathbf{p}_a])$$
User ID Embeddings

- Comp.2: Diffusion Influence Aggregation

$$\mathbf{h}_{Sa}^{k+1} = Pool(\mathbf{h}_b^k | b \in S_a)$$
Ego social network

- Comp.3: Representation Update

$$\mathbf{h}_a^{k+1} = s^{(k+1)}(\mathbf{W}^k \times [\mathbf{h}_{Sa}^{k+1}, \mathbf{h}_a^k]).$$

DANSER

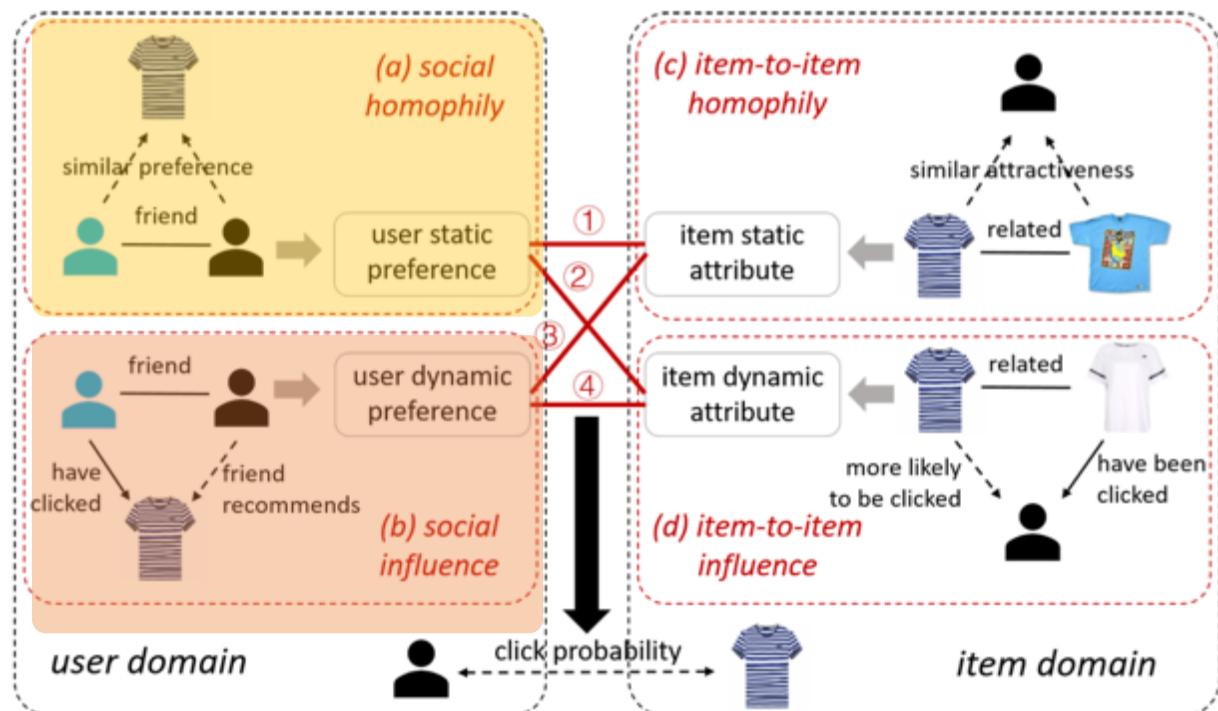
DANSER from [Wu et al, WWW'2019]

Social homophily

- User static preference
- Unchanged & independent of external contexts

Social influence

- User dynamic preference
- Change dynamically with specific contexts



DANSER

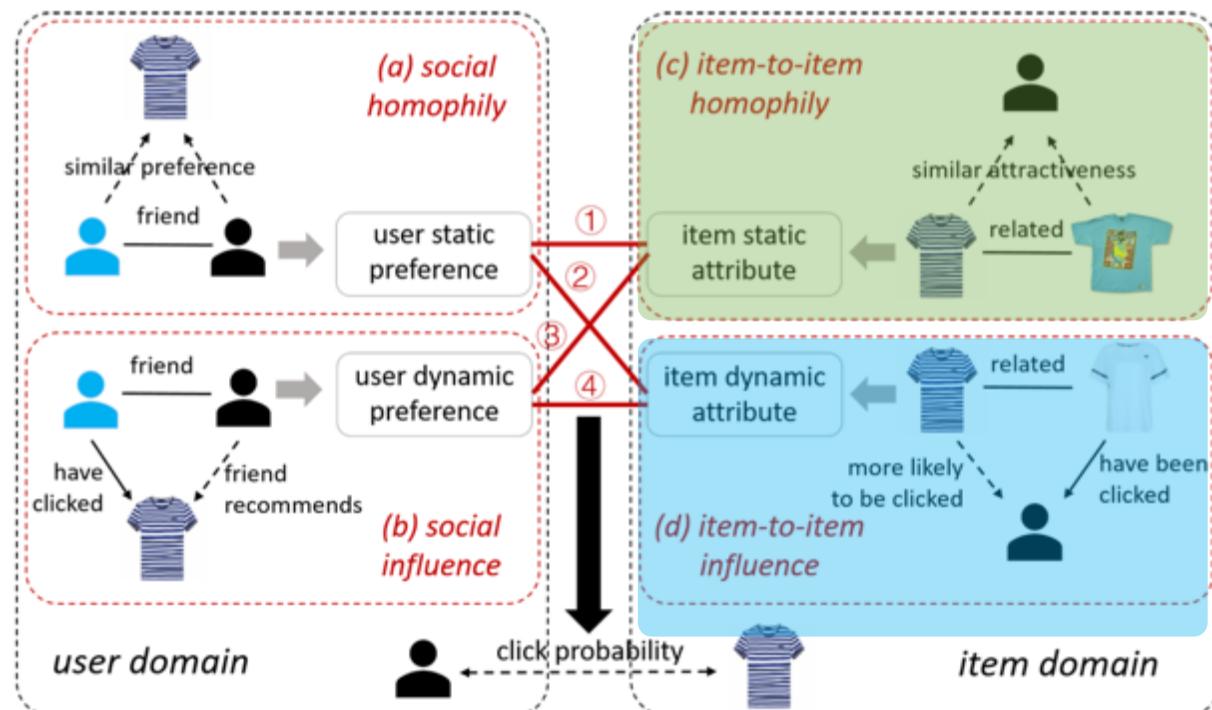
DANSER from [Wu et al, WWW'2019]

Item-to-item homophily

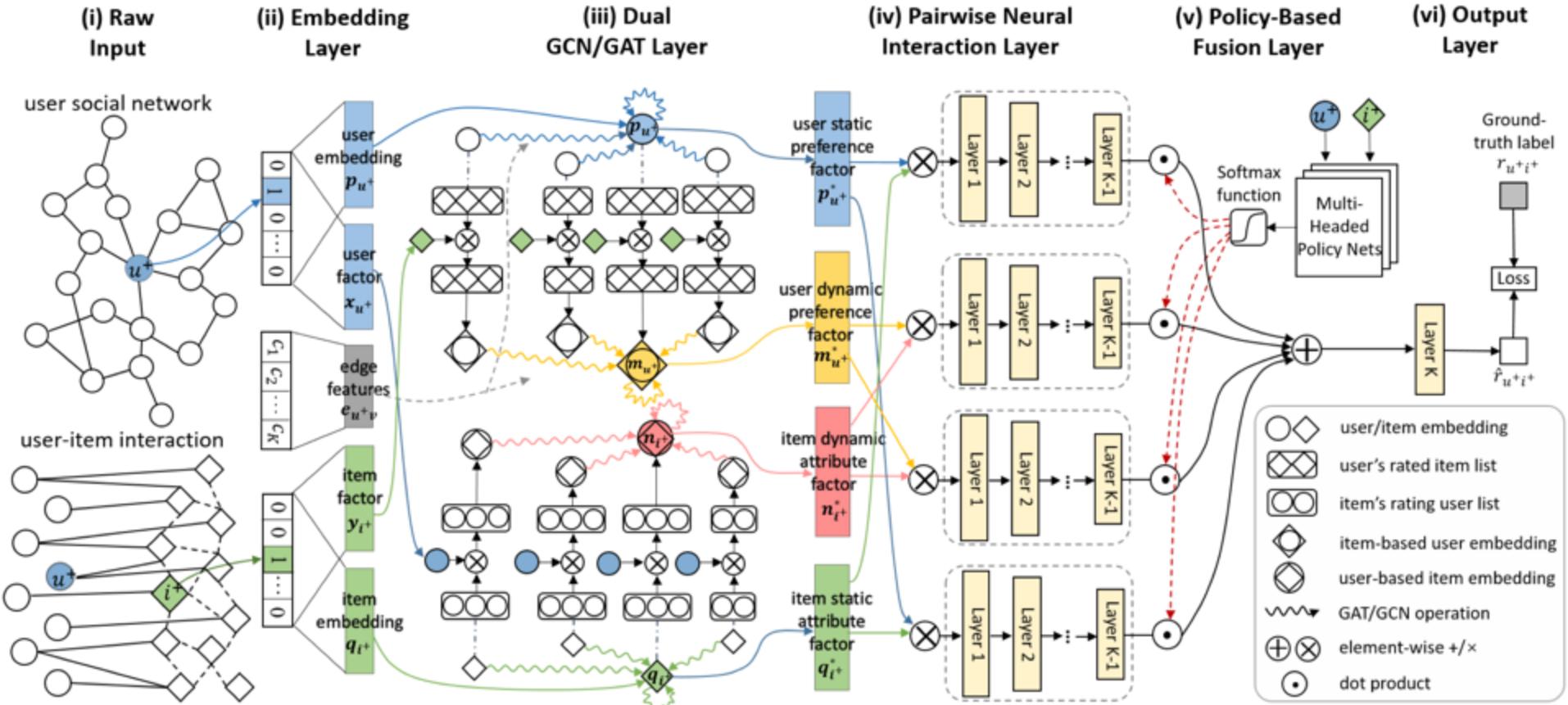
- Item static attribute

Item-to-item influence

- Item dynamic attribute
- Depends on a specific context



Dual GAT in User & Item Domains



Summary: GNN for Social Recommendation

	Graph Data	User Modeling	Item Modeling
GraphRec	<ul style="list-style-type: none">Interaction GraphSocial Network	First-order Connectivity <ul style="list-style-type: none">historical itemssocial relations	First-order Connectivity <ul style="list-style-type: none">user feedback
DiffNet	<ul style="list-style-type: none">Social Network	High-order Connectivity <ul style="list-style-type: none">social influence	First-order Connectivity <ul style="list-style-type: none">user feedback
DANSER	<ul style="list-style-type: none">Interaction GraphSocial Network	First-order Connectivity <ul style="list-style-type: none">social homophily <i>(static)</i>social influence <i>(dynamic)</i>	First-order connectivity <ul style="list-style-type: none">item homophily <i>(static)</i>item influence <i>(dynamic)</i>

Social recommendation needs more guides from social network analysis:

- Behavior modeling, social influence, group formation, information diffusion → from micro to macro!

OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- **Part III: Graph Neural Networks for Recommendation**
 - **Sequential Recommendation: SR-GNN, DGRec**

Slides in <https://next-nus.github.io/>

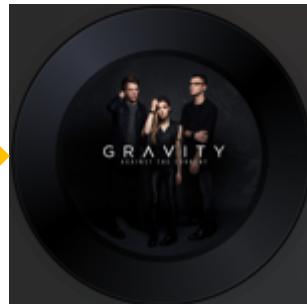
Sequential Recommendation

Sequential (Session-based) recommendation

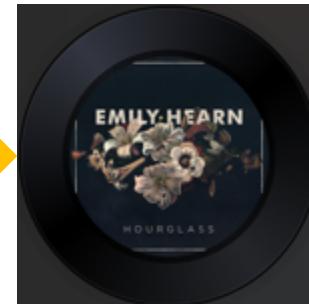
- Given historical interactions, to predict the **successive** items that a user is likely to interact with → **sequential needs of users**



Listen To Your Heart



Dreaming Alone



The Oak Tree



Follow Your Heart

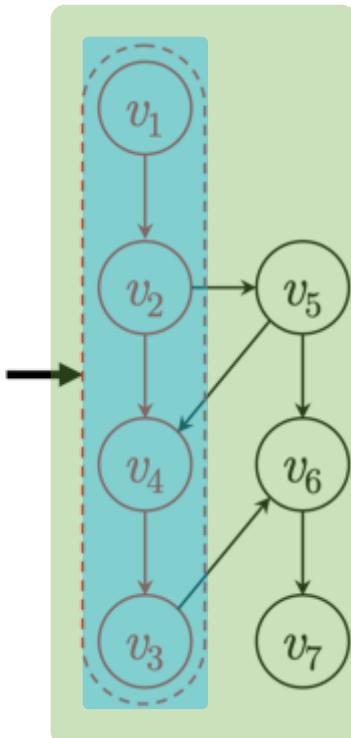
- User interests are dynamic in sessions.
- Sequential pattern is of crucial importance.

Session-based Recommendation with GNNs (SR-GNN)

SR-GNN from [Wu et al, AAAI'2019]

- Sequential pattern of a transition → one session sequence
- Complex patterns of item transitions → all session sequences

$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3$
 $v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7$
 $v_5 \rightarrow v_4 \rightarrow v_3 \rightarrow v_6$
.....



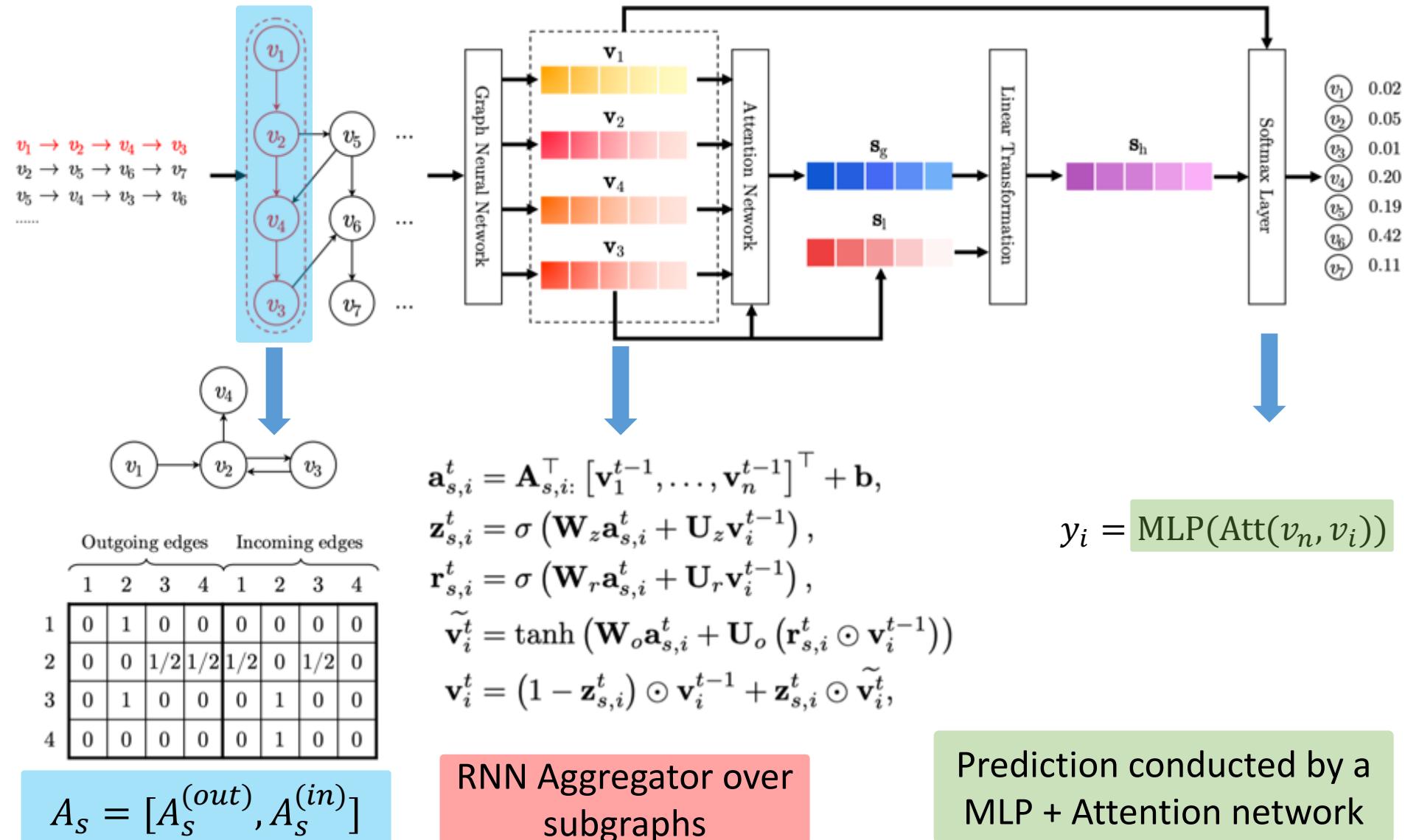
Reorganize all session sequences into graph structured data → session graph

- A directed graph G_s over items
 - Each edge $(v_{s,i-1}, v_{s,i})$ means a user clicks item $v_{s,i}$ after $v_{s,i-1}$ in session s
- Present the global preference in session s

treating a session sequences as a session subgraph

- Present the current interest & sequential needs of the user in session s

Session Graphs with GNNs

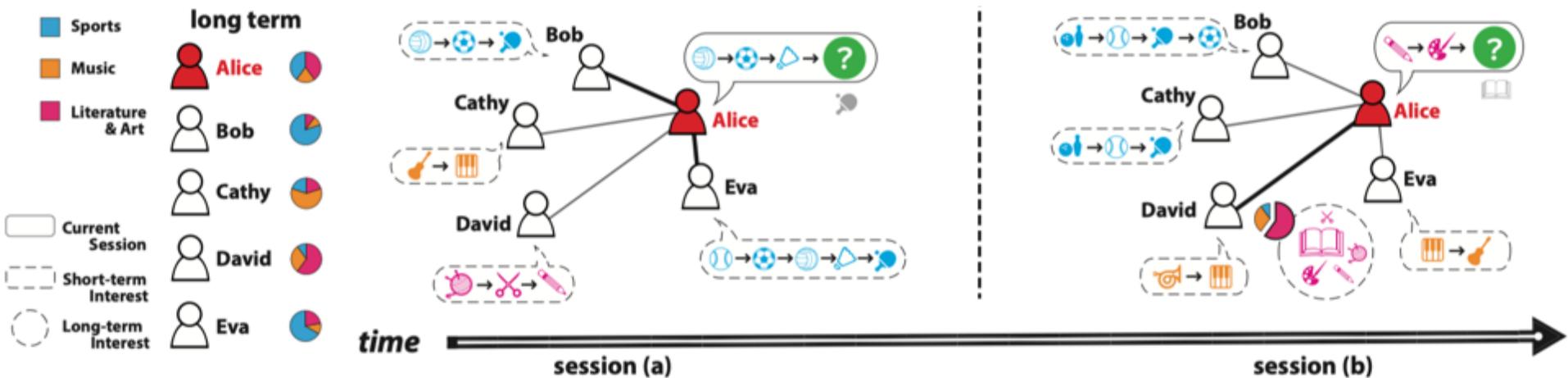


$$A_s = [A_s^{(out)}, A_s^{(in)}]$$

Dynamic Graph Attention Network for Session-based Social Recommendation (DGRec)

DGRec from [Song et al, WSDM'2019]

- Session + Social Recommendation → **social networks**
- User interests change across sessions, due to:
 - **Short-term** preferences of user friends
 - **Long-term** preferences of user friends



Dynamic Social Recommendation

- Dynamic individual interest in **current session** → RNN

$$h_n = f(i_{T+1,n}^u, h_{n-1}), \quad \vec{S}_{T+1}^u = \{i_{T+1,1}^u, \dots, i_{T+1,n}^u\}$$

User session representation LSTM adopted on the session sequence Session sequence of the user

- Friends' interest
 - **Short-term** preferences → a friend's latest online session

$$s_k^s = r_{N_{k,T}} = f(i_{T,N_{k,T}}^k, r_{N_{k,T-1}})$$

- **Long-term** preferences → a friend's average interests, which is no item-sensitive

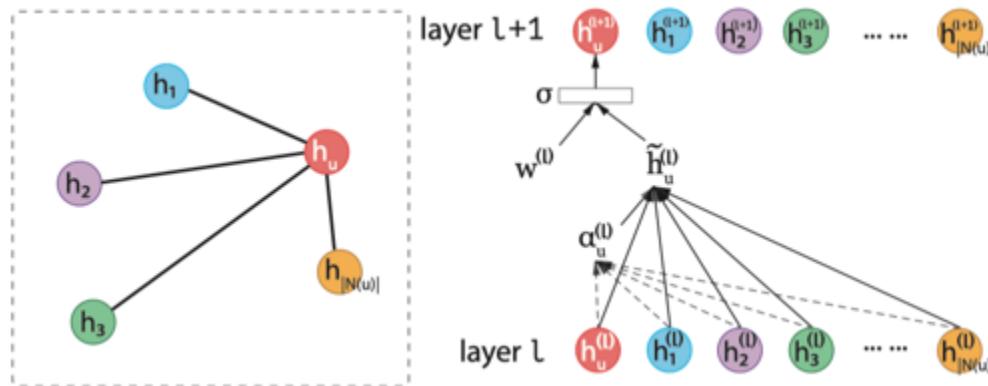
$$s_k^l = \mathbf{W}_u[k,:]$$

User ID embedding

- **Unified representation**

$$s_k = \text{ReLU}(\mathbf{W}_1[s_k^s; s_k^l])$$

Dynamic Graph Attention Network



- **Dynamic feature graph**
 - User + Friends graph
 - Dynamic features → updated whenever a user consumed a new item.
- **Attentive social aggregation**

Level of influence or importance
of a friend to the target user

$$\alpha_{uk}^{(l)} = \frac{\exp(f(h_u^{(l)}, h_k^{(l)}))}{\sum_{j \in N(u) \cup \{u\}} \exp(f(h_u^{(l)}, h_j^{(l)}))}$$

$$\tilde{h}_u^{(l)} = \sum_{k \in N(u) \cup \{u\}} \alpha_{uk}^{(l)} h_k^{(l)}$$

Final representation combining user
session interests & social influence

Summary: GNN for Sequential Recommendation

	Graph Data	User Modeling	Item Modeling
SR-GNN	Directed Session Graph	<ul style="list-style-type: none">• Global preference in all session sequences• Local preference in a session sequence• Without ID information	Graph representations
DGRec	Social Network	<ul style="list-style-type: none">• Short-term preference in the latest session (sequential pattern)• Long-term preference in all sessions (ID Information)• Social influence (graph)	ID embeddings

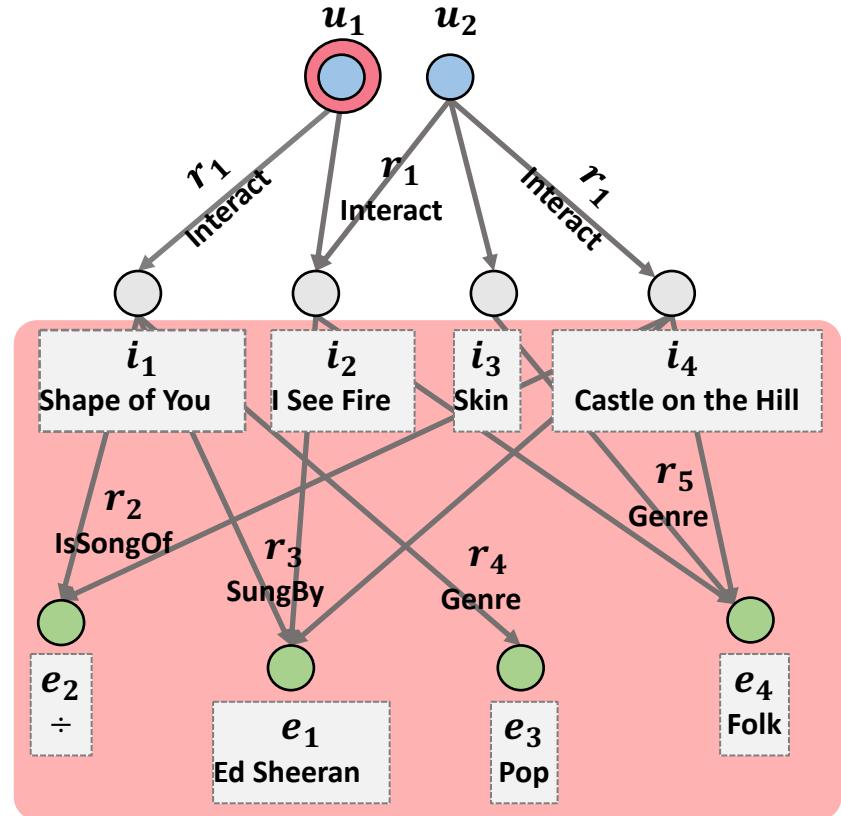
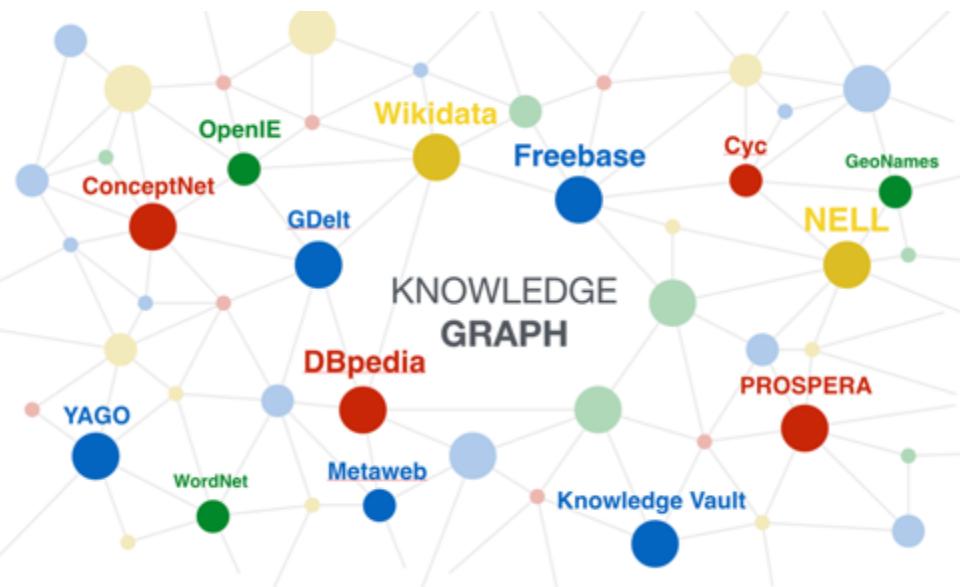
Sequential recommendation needs new & reasonable angles to organize sequence data in the form of graph.

OUTLINE

- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- **Part III: Graph Neural Networks for Recommendation**
 - **Knowledge Graph-based Recommendation: KGCN, KGNN-LS, KGAT**

Slides in <https://next-nus.github.io/>

Knowledge Graph-based Recommendation



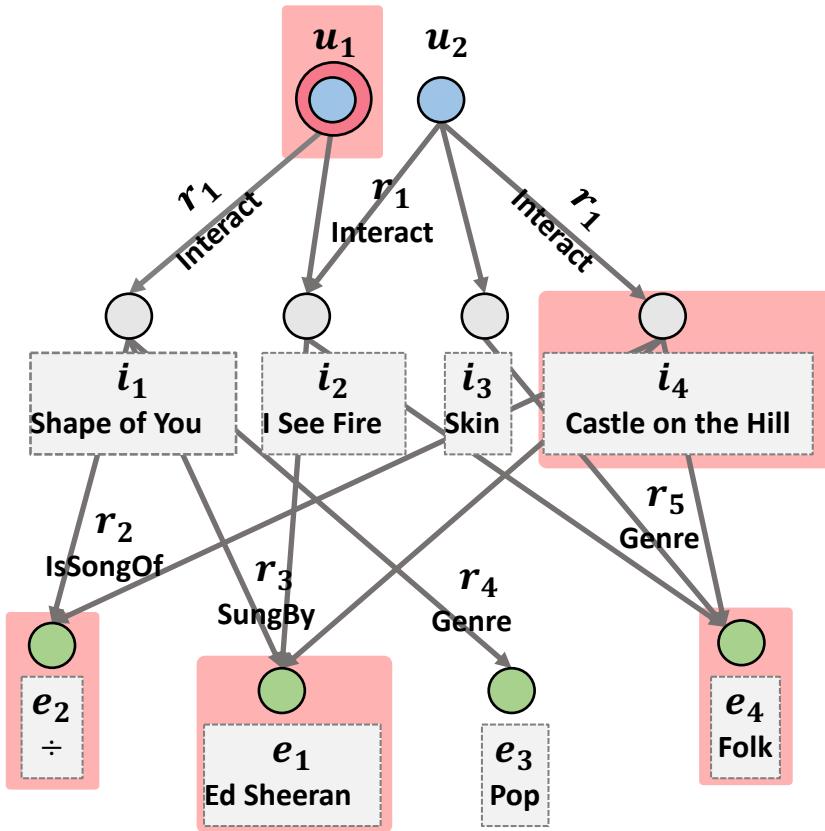
Knowledge Graph (KG):

- Background knowledge on items
- Rich semantics & Relations
- Structural information

Benefit for Recommendation

- Narrow down search space
- Explore user interests reasonably
- Offer explanations

Prior Works: Supervised Learning-based



To estimate u_1 's preference on i_2

Feature Engineering

- $u_1 - i_4$ interaction as a data instance
- transfer item knowledge into a feature vector is $\vec{x} = \langle u_1, i_4, e_1, e_2, e_4 \rangle$

Prediction Modeling

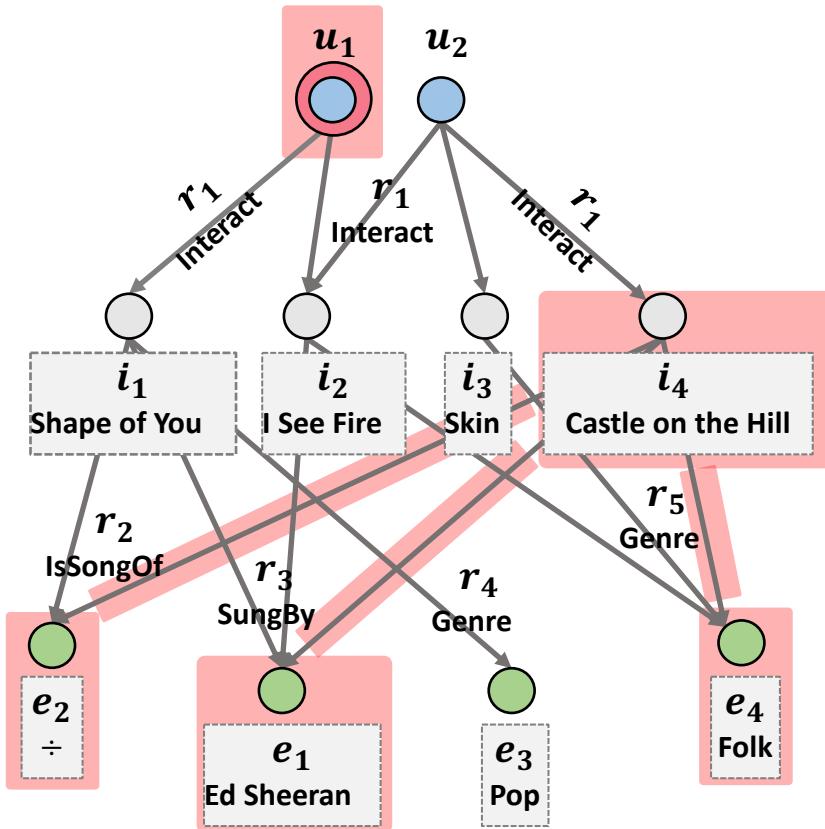
- A supervised learning model
- e.g., FM, NFM, Wide&Deep

Limitations

- Semantic relations are ignored
- Graph structure is ignored
- CF signals are captured in an implicit fashion
- High-order connectivity/relation are ignored



Prior Works: Regularized-based



To estimate u_1 's preference on i_2

Representation Learning

- i_4 -related KG triplets **regularize** the learning of its representation
- Translational Principle
 - Head + Relation \approx Tail

Interaction Modeling

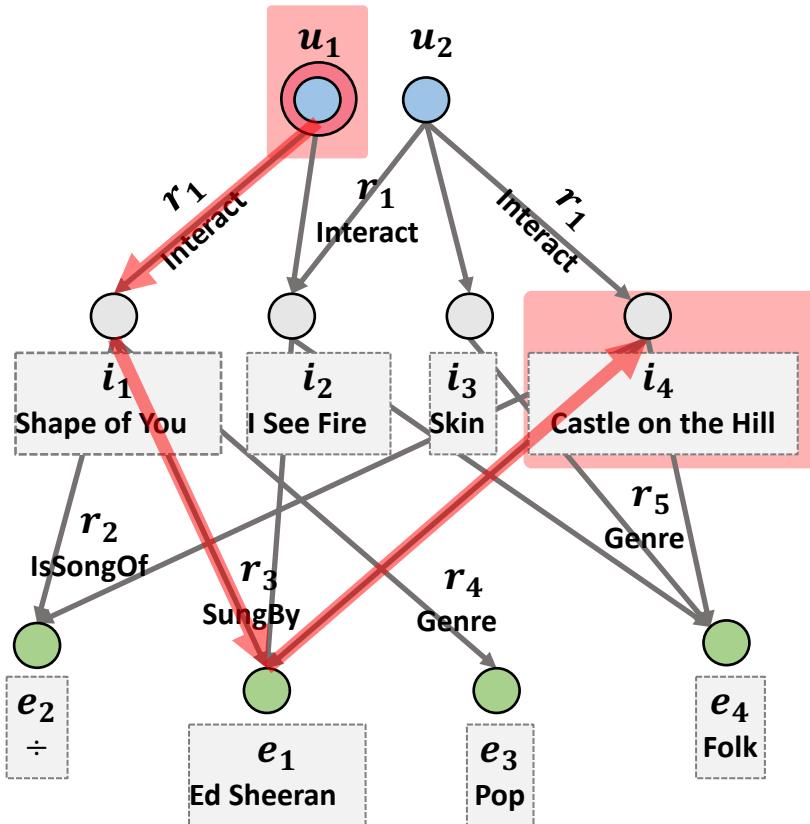
- Reconstruct direct user-item interactions
- e.g., NCF, MF, ...

Limitations

- **High-order connectivity** between user and item nodes are modeled in an **implicit fashion**
- **It fails to synthesize high-order relations.**



Prior Works: Path-based



To estimate u_1 's preference on i_2

Representation Learning

- Paths connecting u_1 and i_4 to represent their connectivity
- $u_1 \rightarrow i_1 \rightarrow e_1 \rightarrow i_2$.

Interaction Modeling

- Information fusion of multiple paths
- A supervised learning model

Limitations

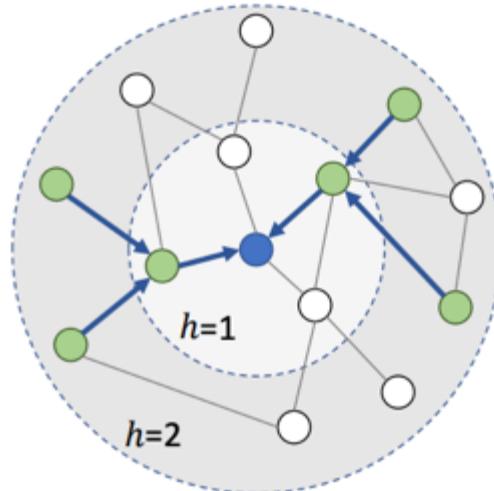
- Require **domain knowledge** to define meta-paths
- Require **labor-intensive feature engineering** to extract qualified paths
- Have rather high complexity



Knowledge Graph Convolution Network (KGCN)

KGCN from [Wang et al, WWW'2019]

- Item graph → KG entities are used to enrich item representation



Comp.1 & 2

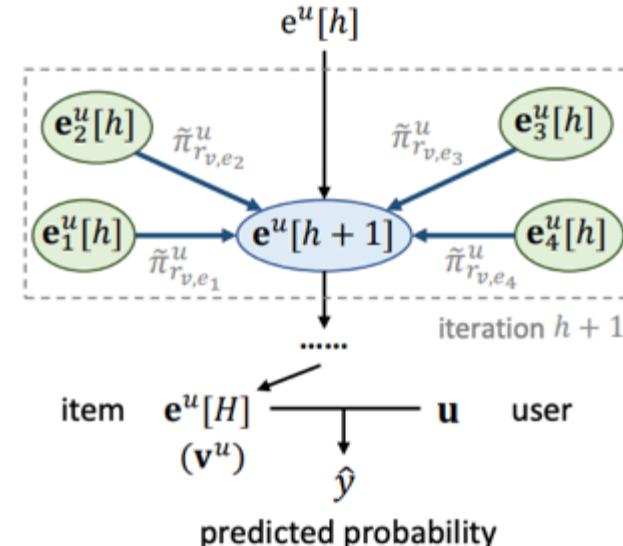
$$\mathbf{v}_{\mathcal{N}(v)}^u = \sum_{e \in \mathcal{N}(v)} \tilde{\pi}_{r_v, e}^u \mathbf{e}_e$$

Attention score
of user-relation

Comp.3

$$aggsum = \sigma \left(\mathbf{W} \cdot (\mathbf{v} + \mathbf{v}_{S(v)}^u) + \mathbf{b} \right)$$

KG entities connected
with the target item



Prediction

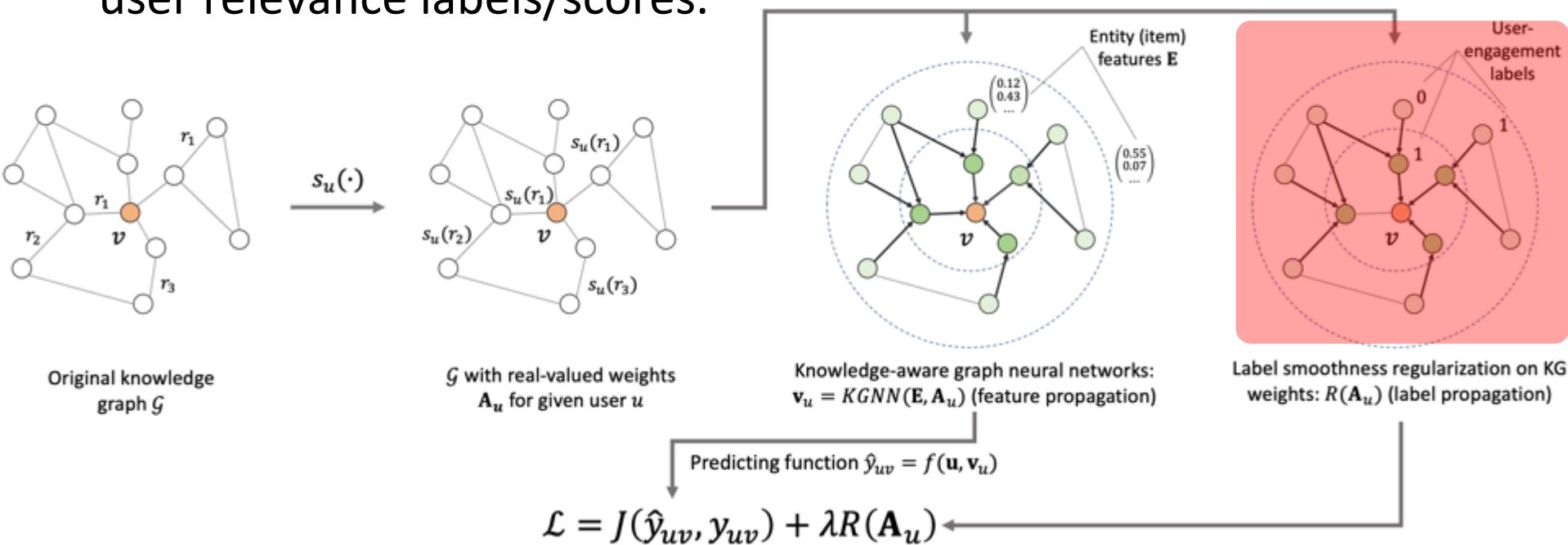
$$\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}^u)$$

User ID embeddings →
users are excluded from
the propagation.

Knowledge Graph Neural Networks with Label Smoothness Regularization (KGNN-LS)

KGNN-LS is an extension of KGCN from [Wang et al, KDD'2019]

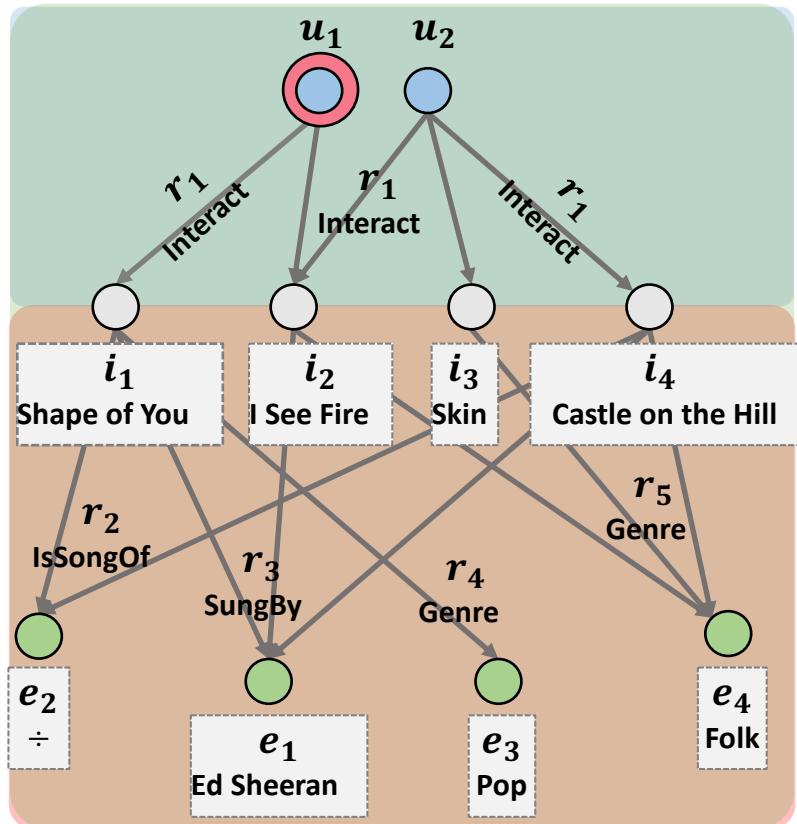
- Label smoothness → adjacent items in KG are likely to have similar user relevance labels/scores.



$$E(l_u, \mathbf{A}_u) = \frac{1}{2} \sum_{e_i \in \mathcal{E}, e_j \in \mathcal{E}} A_u^{ij} (l_u(e_i) - l_u(e_j))^2$$

Knowledge Graph Attention Network (KGAT)

KGAT from [Wang et al, KDD'2019]



User-Item Bipartite Graph

- User-Item Direct Interactions

$$u_1 \xrightarrow{r_1} i_1$$



Knowledge Graph

- Item-Item External Connections

$$i_1 \xrightarrow{r_2} e_1$$



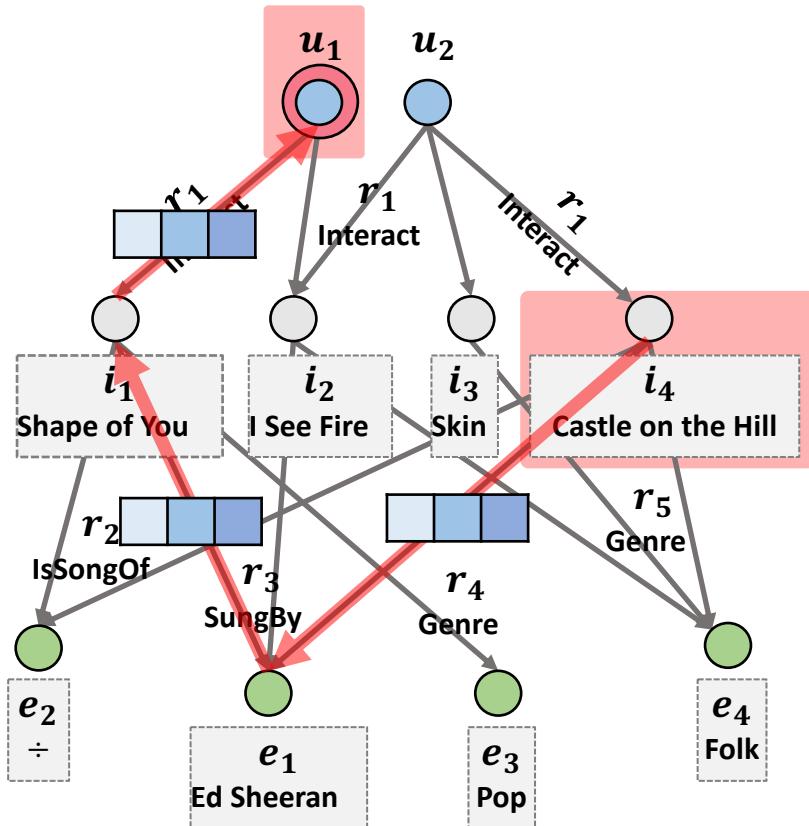
Collaborative Knowledge Graph

- High-order connectivity between users and items

$$u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1 \xrightarrow{-r_2} i_2 \xrightarrow{r_1} u_1$$

- Reasoning ability & Explainability

Attentive Embedding Propagation in KGAT



Attentive Embedding Propagation, inspired by GNNs

- Propagate embeddings recursively on the graph
- Reveal the importance of a high-order connectivity via relation-aware attentions
- Construct information flows in the embedding space

Attentive Embedding Propagation in KGAT

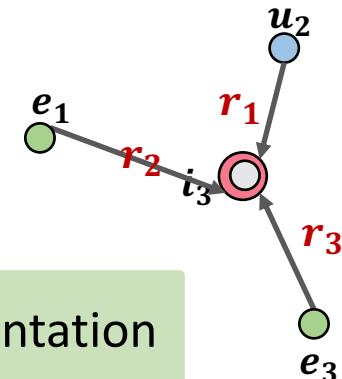
- **Comp.1:** Information Propagation

The messages accounting for first-order connectivity

The set of triples, where the target node is the head entity

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h, r, t) \in \mathcal{N}_h} \pi(h, r, t) \mathbf{e}_t$$

Tail representation



- **Comp.2:** Knowledge-aware Attention Aggregation

decay factor on each propagation

$$\pi(h, r, t) = (\mathbf{W}_r \mathbf{e}_t)^\top \tanh((\mathbf{W}_r \mathbf{e}_h + \mathbf{e}_r))$$

the attention score is dependent on the distance of \mathbf{e}_t and \mathbf{e}_h in r 's space

- **Comp.3:** Representation Update

$$f_{\text{Bi-Interaction}} = \text{LeakyReLU}\left(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\right) + \text{LeakyReLU}\left(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})\right),$$

Similar to NGCF

Model Training

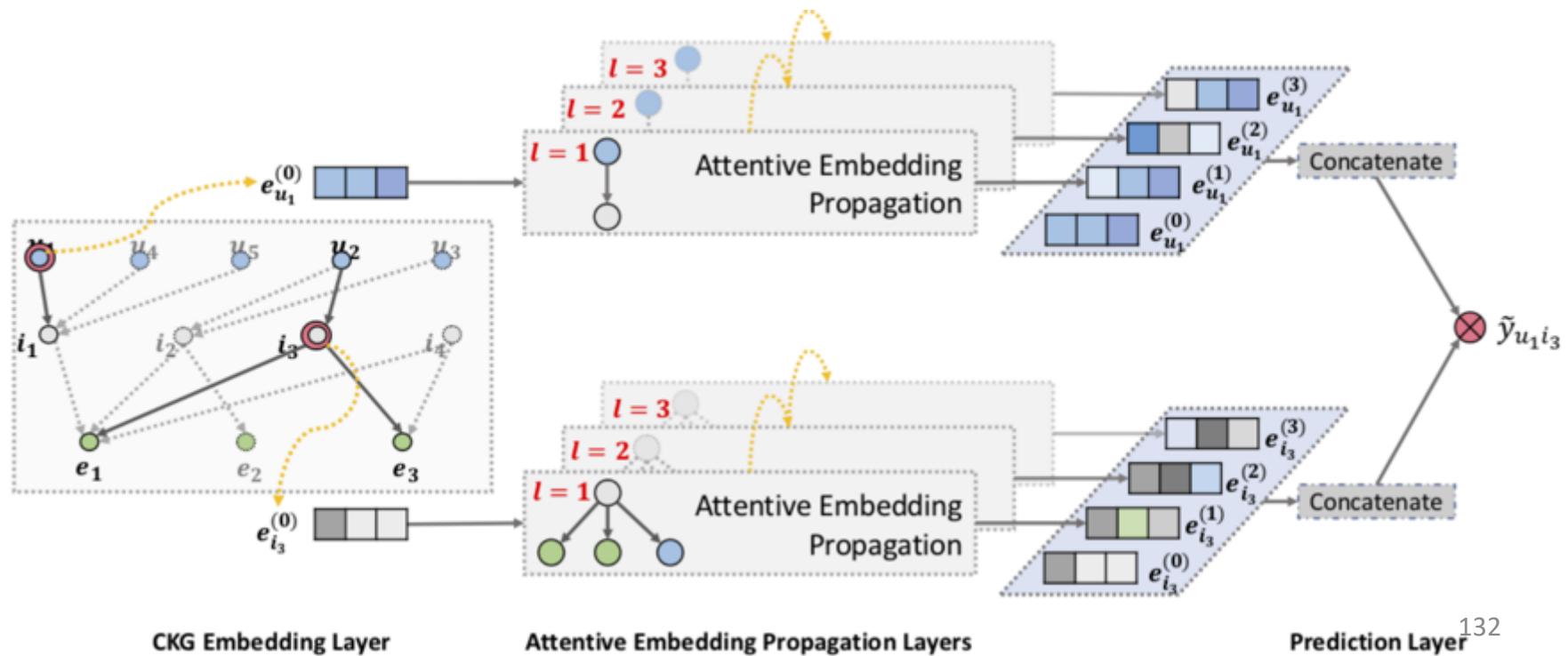
$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \parallel \cdots \parallel \mathbf{e}_u^{(L)}$$

$$\mathbf{e}_i^* = \mathbf{e}_i^{(0)} \parallel \cdots \parallel \mathbf{e}_i^{(L)}$$

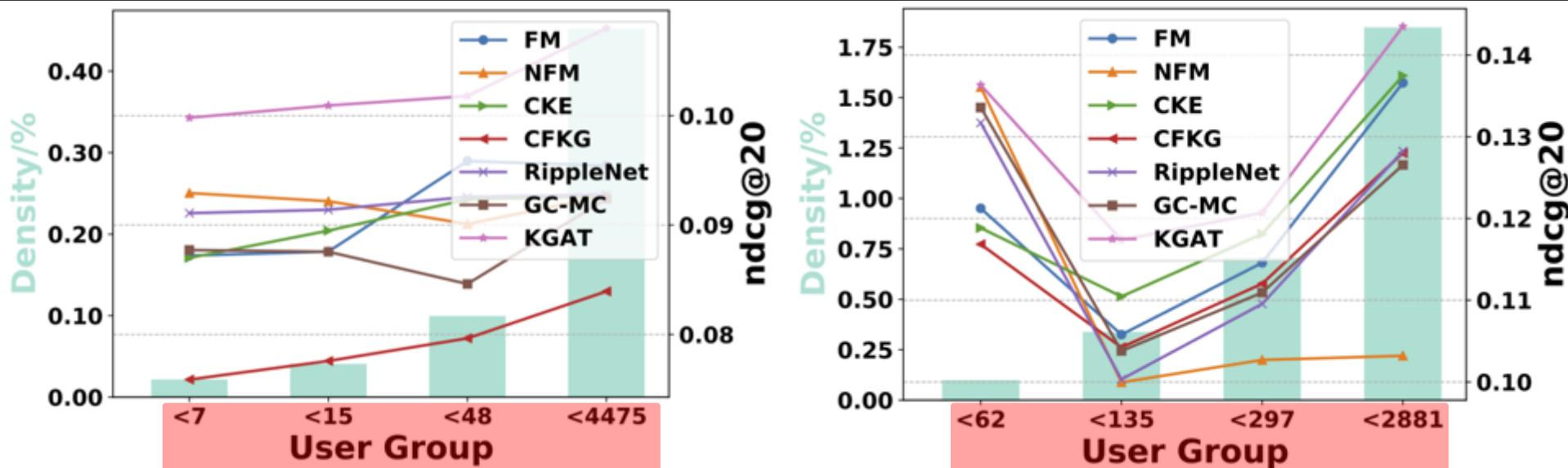
$$\hat{y}(u, i) = \mathbf{e}_u^{*\top} \mathbf{e}_i^*$$

Similar to NGCF, the representations at different layers

- emphasize the messages passed over different connections
- have different contributions in reflecting user preference



Experiment Results —— Sparsity Issue



user groups with different group sparsity levels

- KGAT outperforms the other models in most cases, especially on the two sparsest user groups in Amazon-Book and Yelp2018.
- It again verifies the significance of high-order connectivity modeling:
 - contains the lower-order connectivity
 - enriches the representations of inactive users via recursive embedding propagation

Case Study for Explainable Recommendation

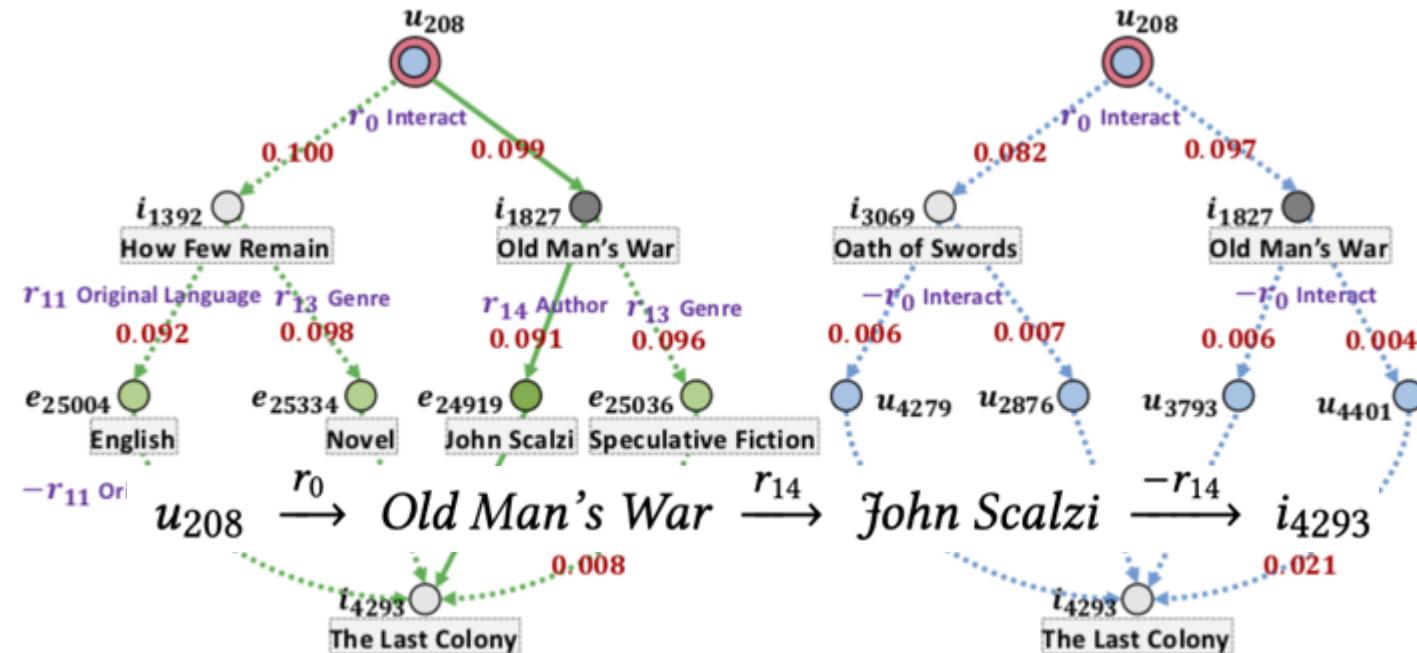


Figure 4: Real Example from Amazon-Book.

- KGAT captures the behavior-based and attribute-based high-order connectivity, which play a key role to infer user preferences.

$$u_{208} \xrightarrow{r_0} Old\ Man's\ War \xrightarrow{r_{14}} John\ Scalzi \xrightarrow{-r_{14}} i_{4293}$$

- The explanation can be "*The Last Colony* is recommended since you have watched *Old Man's War* written by the same author *John Scalzi*."

OUTLINE

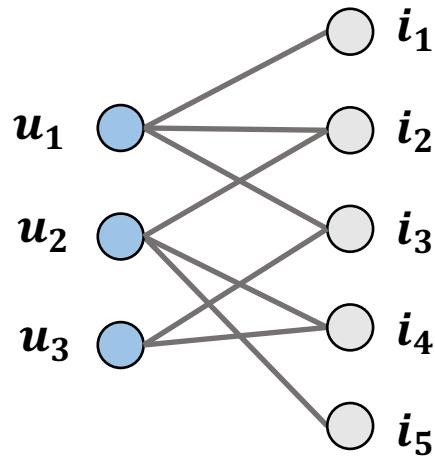
- Introduction
- Part I: Preliminary of Recommendation
- Part II: Random Walk for Recommendation
- Part III: Network Embedding for Recommendation
- Part III: Graph Neural Networks for Recommendation
- **Summary & Future Directions**

Slides in <https://next-nus.github.io/>

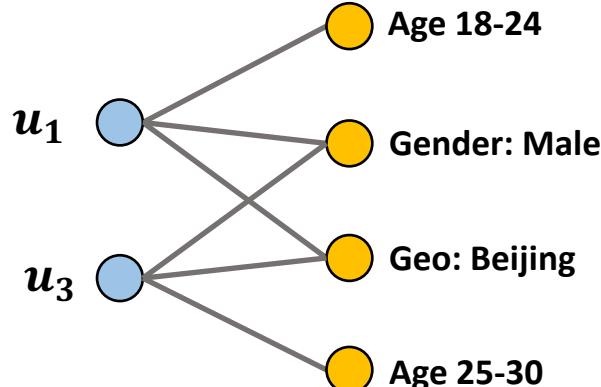
Summary

The data is more **closely connected** than we might think!

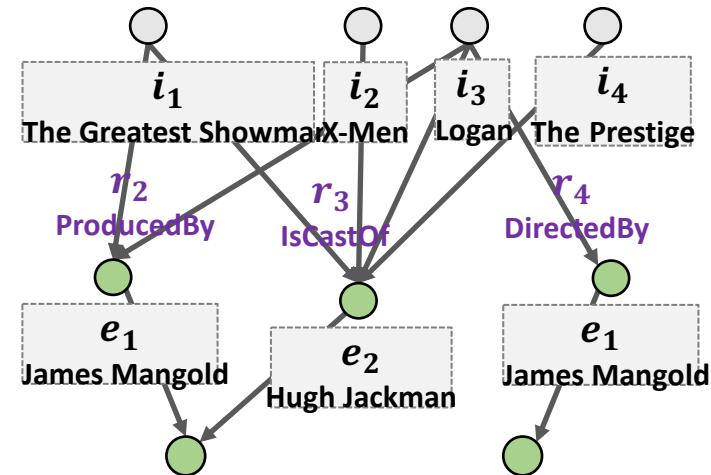
User-Item Interactions



User/Item Profiles



Knowledge Graph



Limited Representation Ability

Information Propagation along with the connections

Suboptimal Model Capacity

High-order connectivity complementary to user-item interactions

Limited Reasoning Ability

High-order connectivity interpreting user intents

Future Direction (1)

- **Dynamic Graphs**
 - Graph for recommendation evolves over time
 - Online User-Item Interactions, Trending of (fashion) items, CTR prediction ...
 - Challenges:
 - How to efficiently & incrementally update representations?
 - How to incorporate edge timing?
 - How to forget old/irrelevant information? ...
- **Adversarial Learning**
 - Attack & Defense
 - Node Features + Edge Features + Graph Structure
 - Applications:
 - Malicious detection, Fraud detection ...

Future Direction (2)

- **Causal Inference**
 - Get intents behind user behaviors
 - What contexts → what behaviors are reasonable
 - Towards explainable recommendation
- **Neural Symbolic Reasoning**
 - Mimic Human reasoning
 - Study & Understand user behaviors



THANK YOU!