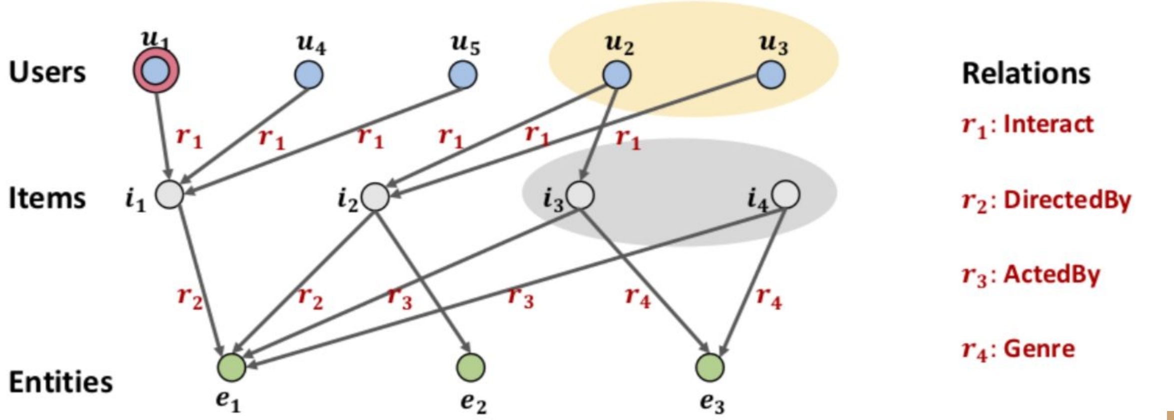
GNN in KG-based Recommender System

Jingyue Shen, Haochen Li, Boyuan He

Introduction

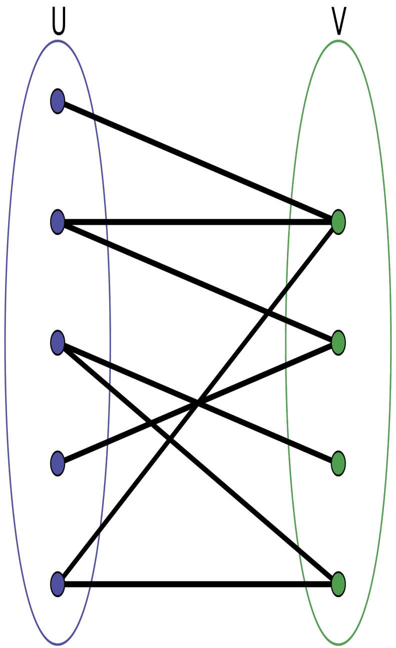
- Traditional Recommender System
  - Collabrotive Filtering (CF) based approach
  - Content-based approach - Factorization Machine
- Issue:
  - CF-based: perform poorly in sparse situations
  - Content-based: do not explore “high-order relations”



Introduction

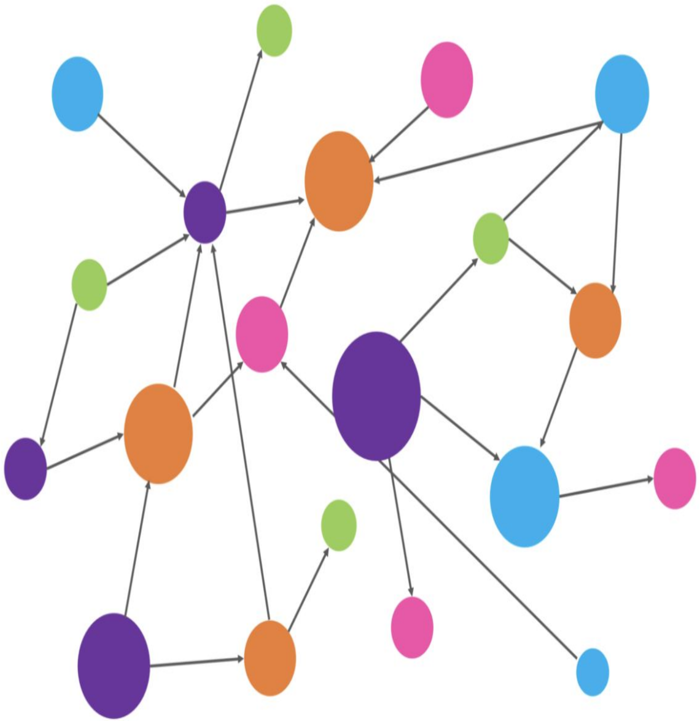
- KG-based Recommender System
    - Goal & challenges: find ways to represent KG information to improve recommendations
    - One recent trend: use GNN to capture information in KG

User/Item Interaction



Assists

Knowledge Graph



Bipartite Graph by MistWiz - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1814874

Approaches

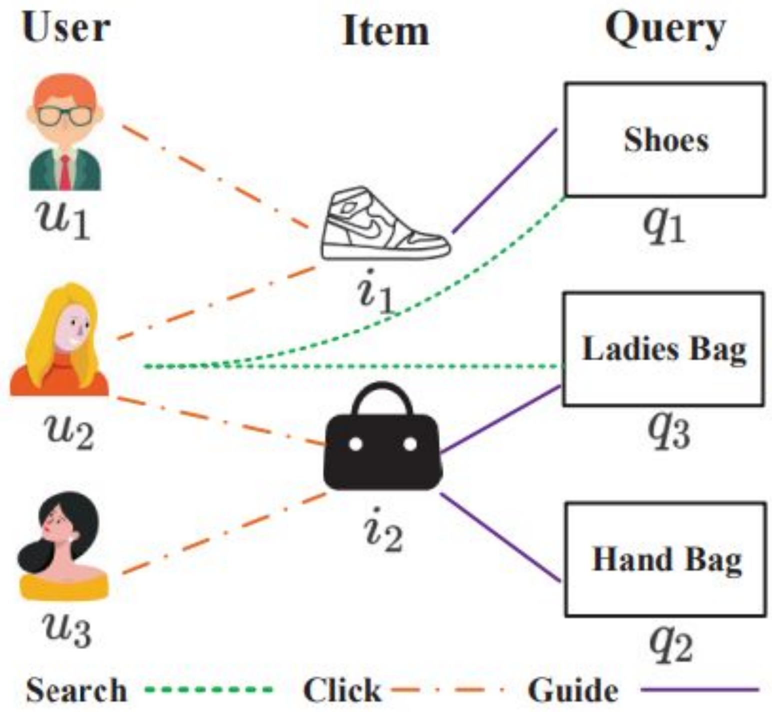
- How to get context embedding and aggregate with node embedding?
    - Relation-unaware
    - Relation-aware
        - Subgraph aggregation
        - Attentive aggregation
        - Explicit relation aggregation

Relation-unaware

Doesn’t distinguish between different types of edges

Focus on how to pick and aggregate node embedding

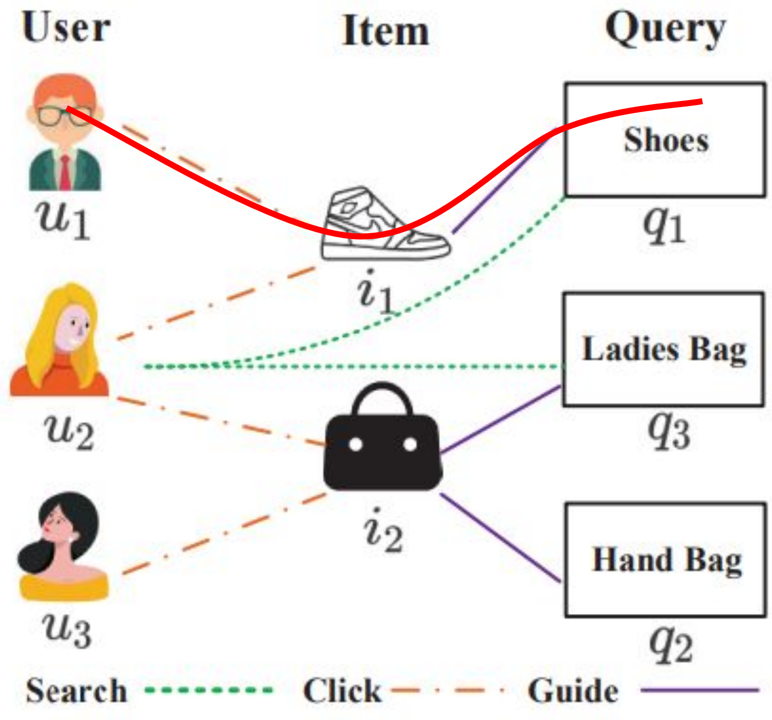
Metapath-guided Embedding method for Intent Recommendation



A GNN of user, item, query interactions

- User search query
- Query guide to items
- User click item

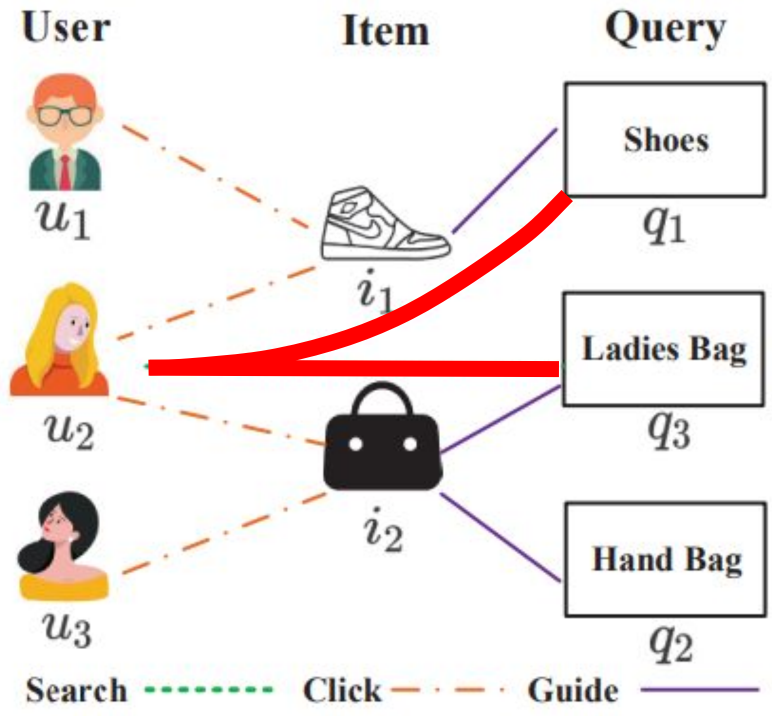
Metapath-guided Embedding method for Intent Recommendation



Metapath

- User-item-query
- Query-user-item

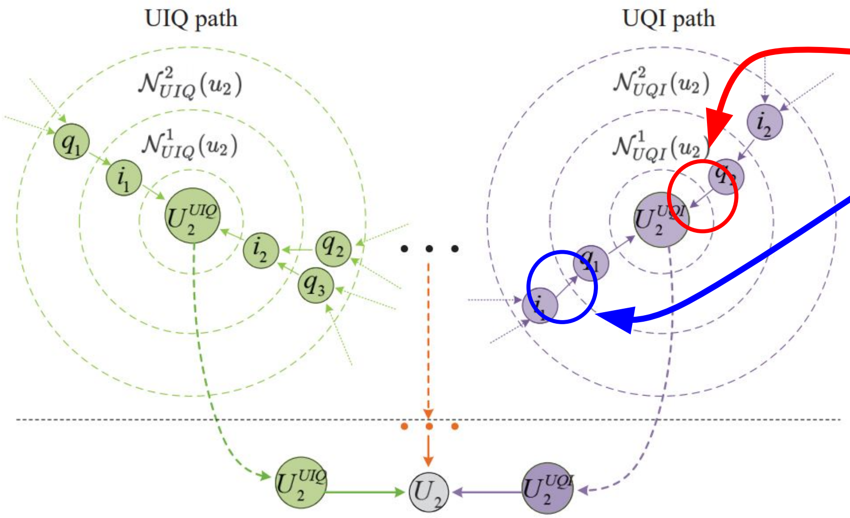
Metapath-guided Embedding method for Intent Recommendation



Metapath

- User-item-query
- Query-user-item

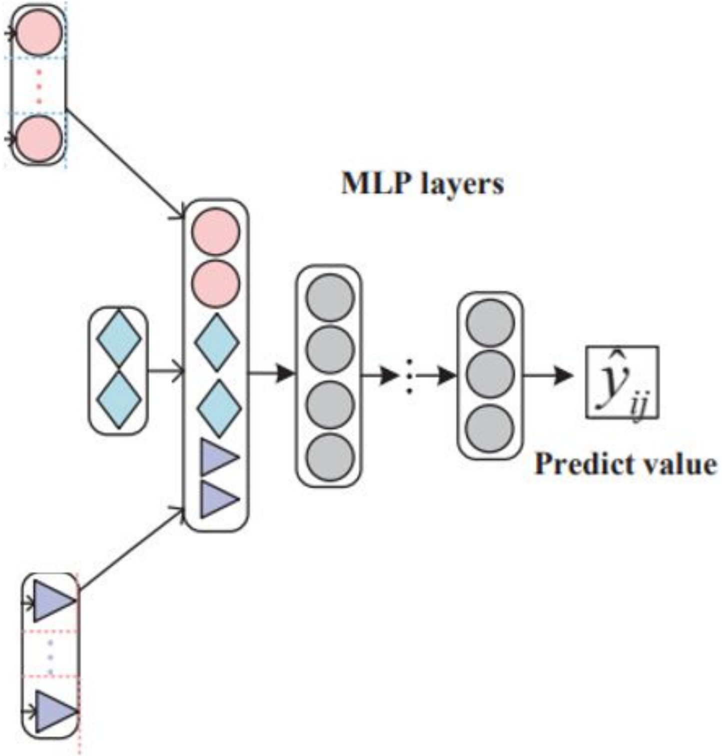
Metapath-guided Embedding method for Intent Recommendation



Aggregate embedding for u2

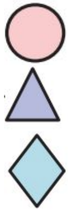
- Use LSTM to get embedding for user
- User average function for other

Metapath-guided Embedding method for Intent Recommendation



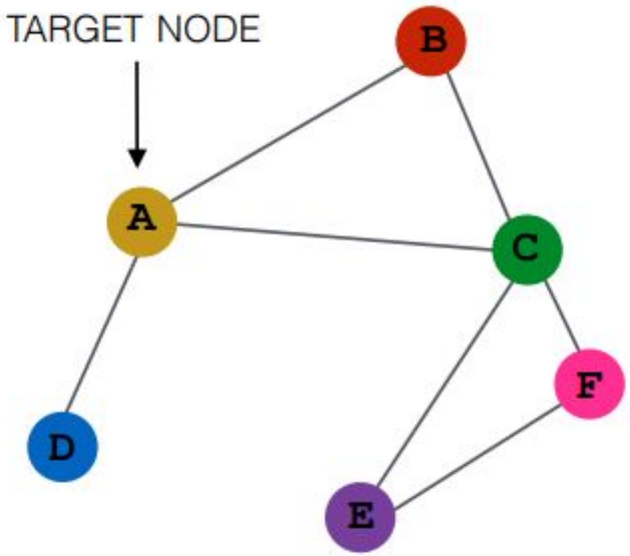
Throw

- user embeddings
- query embedding
- statics features



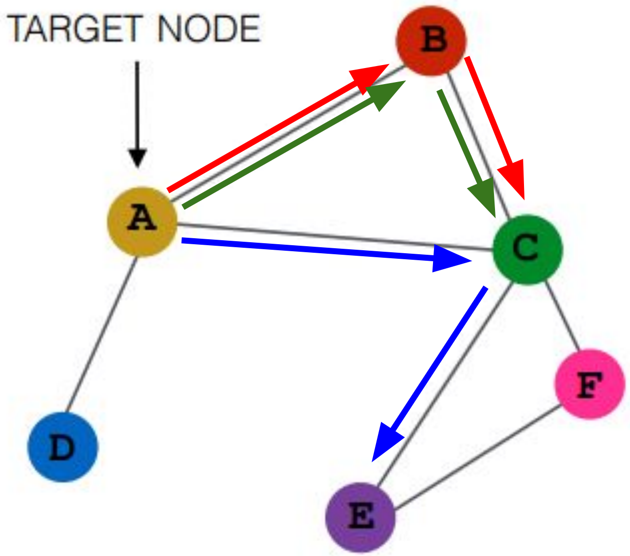
into MLP layer to predict how likely the user will search the query

Graph Convolutional Neural Networks for Web-Scale Recommender Systems



Use random walk algorithm to assign importance score

Graph Convolutional Neural Networks for Web-Scale Recommender Systems



A -> B ->C

B:1, C:1

A -> C ->E

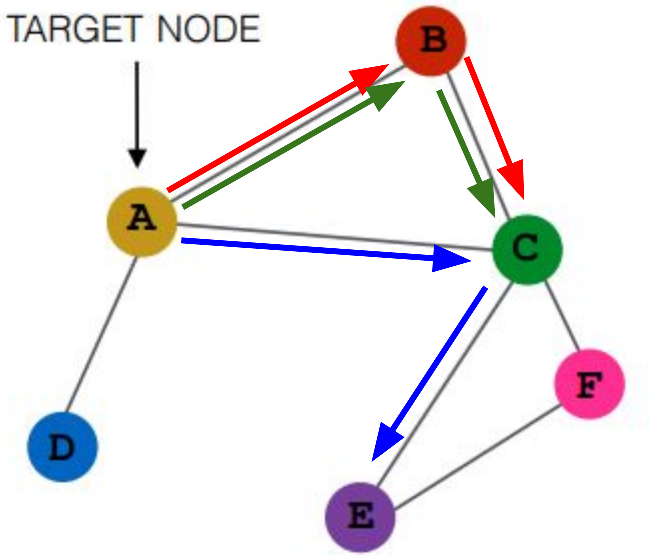
B:1, C:2, E:1

A -> B -> C

B:2, C:3, E:1

...

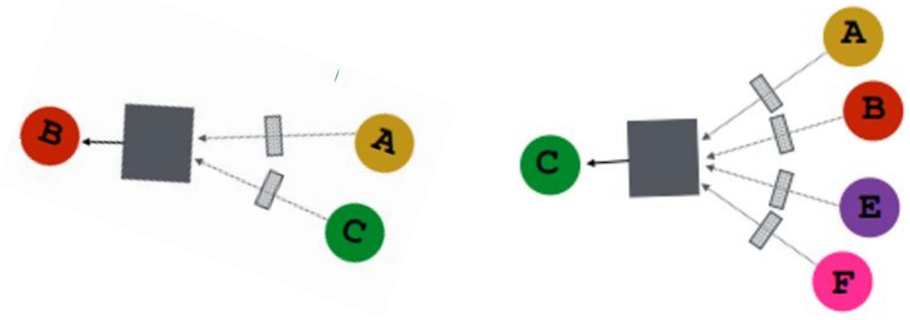
Graph Convolutional Neural Networks for Web-Scale Recommender Systems



B:2, C:3, E:1

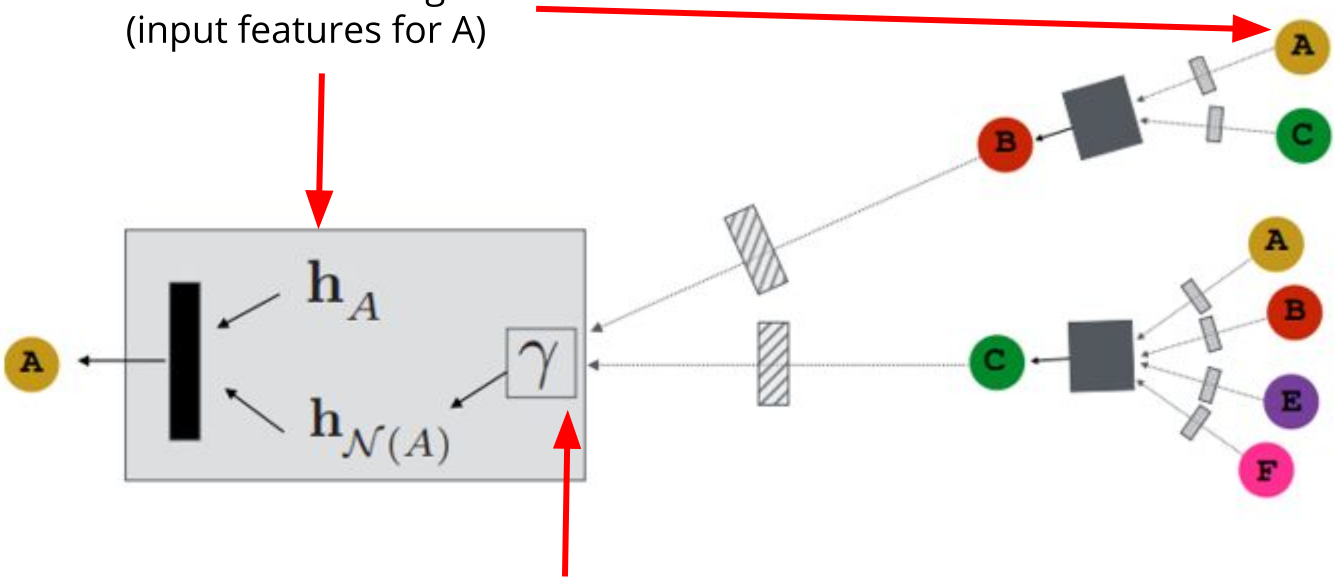
Pick top n important node

For example top 2 : **B**, **C**



Graph Convolutional Neural Networks for Web-Scale Recommender Systems

A’s old embedding
(input features for A)



Element-wise mean

Relation-aware

Distinguish between different type of edges

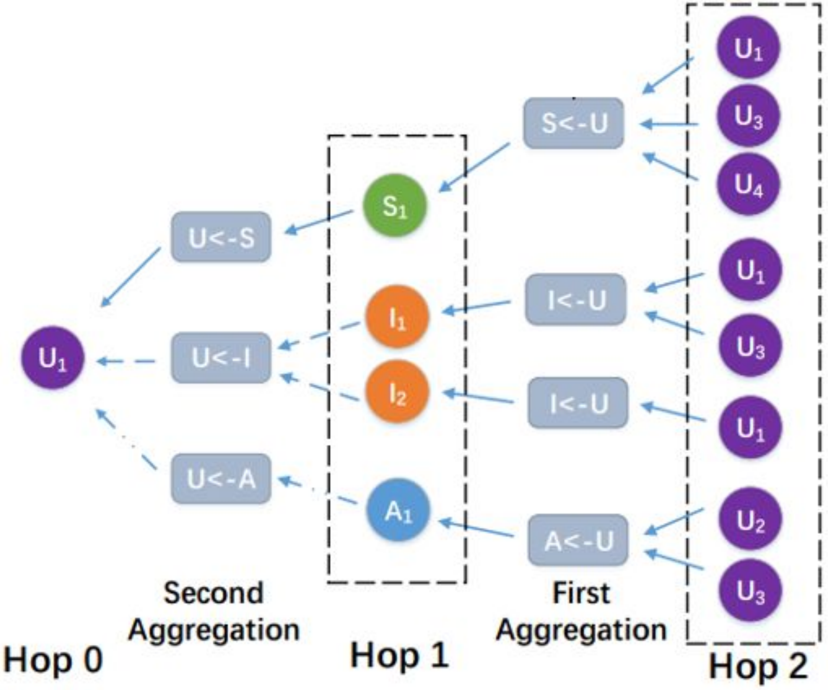
1. Subgraph aggregator:
    a. split the neighborhood graph into multiple subgraphs
    b. all edges of a subgraph belong to only one of the relation types in R
    c. Each subgraph has its own parameters

Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation

Similarities with previous paper

- Metapath based approach
- When updating target node embedding, concatenate neighbor aggregate result with old target node embedding

Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation



One aggregator for each type of edge

for sell-agent to user edge

For user to item edge

Item to user edge

…

Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation

Aggregation result from neighbor are multiplied with a weight score

weight score for node n from neighbor v:

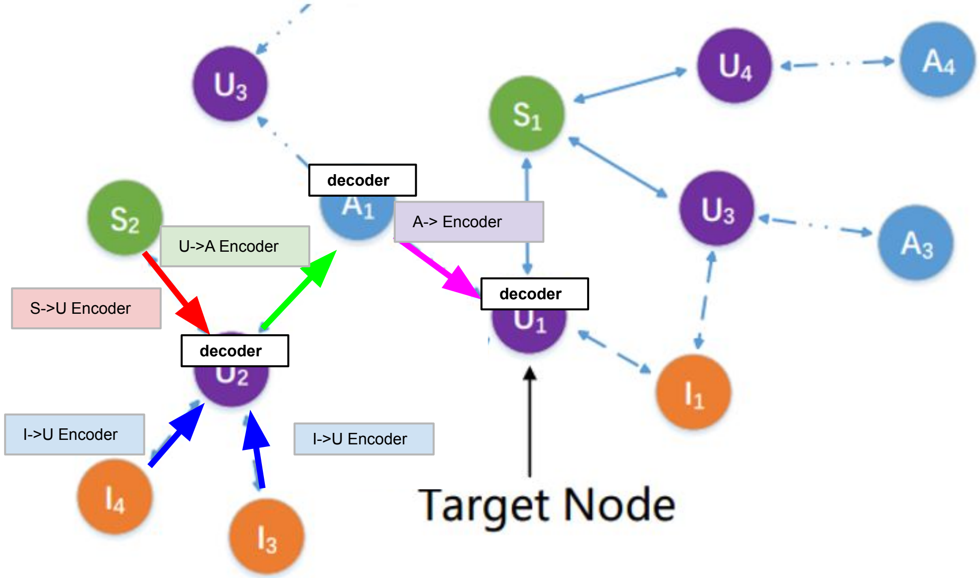
1. feed v's embedding and n's embedding into two learnable matrices that output two vectors
2. Dot product these two vectors together
3. Put dot product into activation function

STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems

To get the embedding for node n

1. Use learnable matrix (edge type specific) to map n's neighbor v to lower dimensional representation
2. Multiple neighbor's low dimensional representation are summed together
3. Use two-layer feedforward neural network to upscale the low dimension representation

STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems



STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems

Tackle cold start problem

1. During training, a portion of the node would be masked, which would have representation of zero vector
2. New node that doesn't have any embedding can be represented as zero vector

IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation

First order proximity (direct interaction)

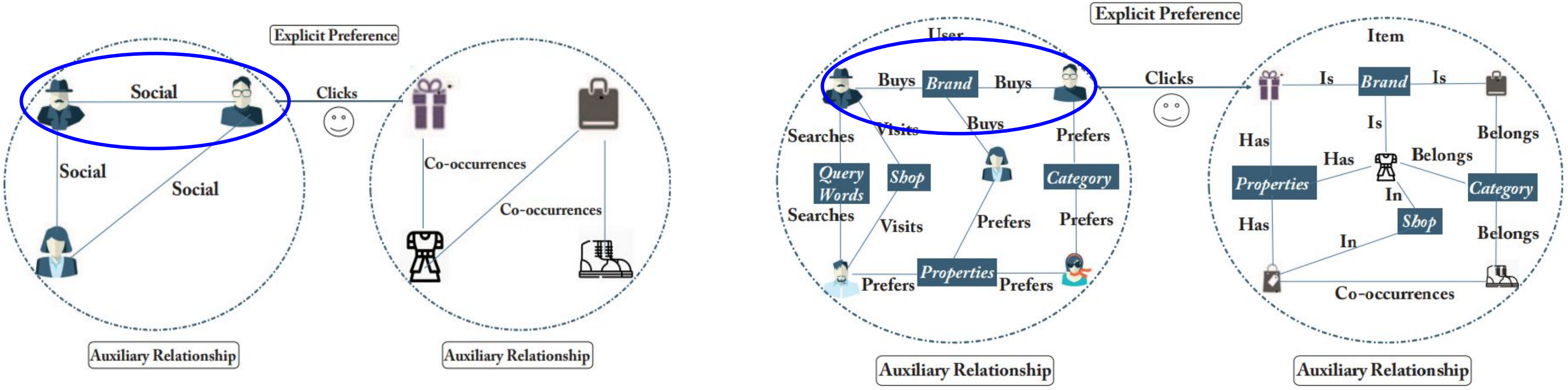
User A submit query on shoes => A - shoes have first order

Second order proximity

User B submit query on shoes and bags => B - A have second order

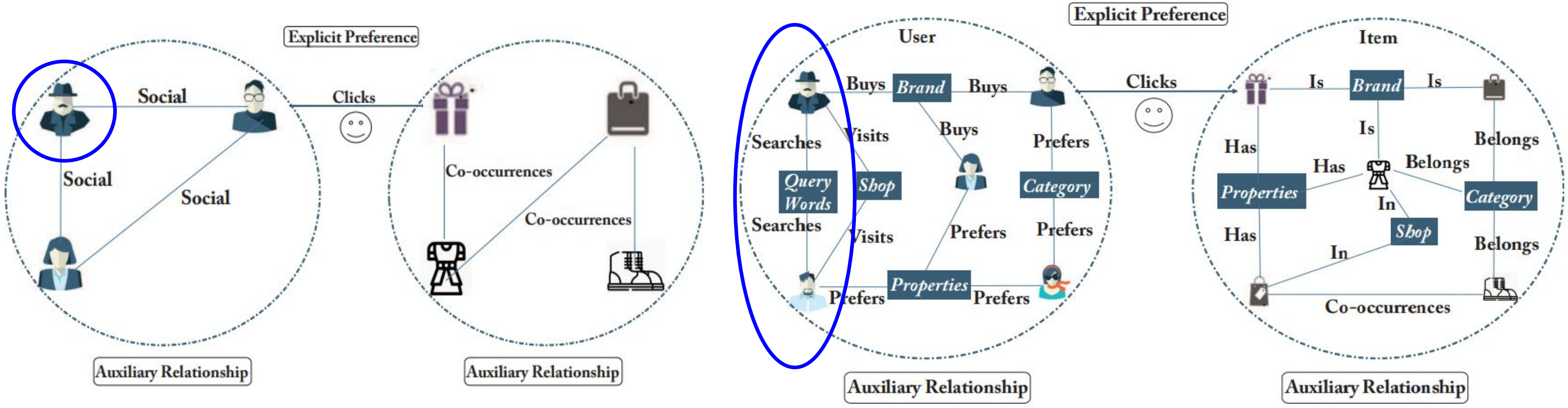
IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation

Use auxiliary information to add weighted edge to node who have second order proximity (weight = number of common auxiliary neighbors)



IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation

Use auxiliary information to add weighted edge to node who have second order proximity (weight = number of common auxiliary neighbors)



IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation

Vector-wise convolution

h_A

3 level dense layer ← Learnable vector ← Concatenate

$h_{N(A)}$

Learnable vector ← h_A

3 level dense layer ← Learnable vector

Learnable vector ← $h_{N(A)}$

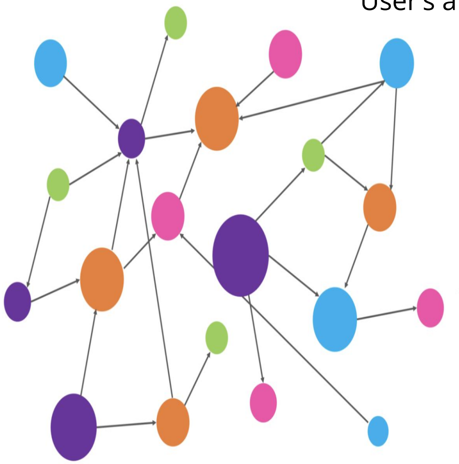
Relation-aware

2. Attentive aggregator:

- Assign each relation with different weights
- Convert into a weighted graph

KGCN/KGNN-LS

Original KG

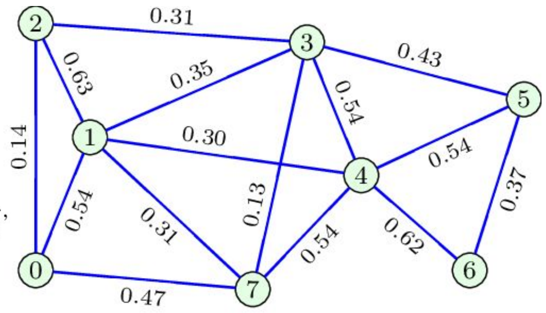


User's attention on each relation

$$\pi^u_r = g(\mathbf{u}, \mathbf{r}),$$

$$\tilde{\pi}^u_{r_{v,e}} = \frac{\exp(\pi^u_{r_{v,e}})}{\sum_{e \in \mathcal{N}(v)} \exp(\pi^u_{r_{v,e}})},$$

Weighted Graph



GNN Layers

$$agg_{sum} = \sigma\left(\mathbf{W} \cdot (\mathbf{v} + \mathbf{v}^u_{\mathcal{S}(v)}) + \mathbf{b}\right)$$

$$agg_{concat} = \sigma\left(\mathbf{W} \cdot concat(\mathbf{v}, \mathbf{v}^u_{\mathcal{S}(v)}) + \mathbf{b}\right)$$

$$agg_{neighbor} = \sigma\left(\mathbf{W} \cdot \mathbf{v}^u_{\mathcal{S}(v)} + \mathbf{b}\right)$$

Final item embedding $\mathbf{v}^u$

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

Trained end-to-end

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \left(\sum_{v: y_{uv}=1} \mathcal{J}(y_{uv}, \hat{y}_{uv}) - \sum_{i=1}^{T^u} \mathbb{E}_{v_i \sim P(v_i)} \mathcal{J}(y_{uv_i}, \hat{y}_{uv_i}) \right) + \lambda \|\mathcal{F}\|_2^2,$$

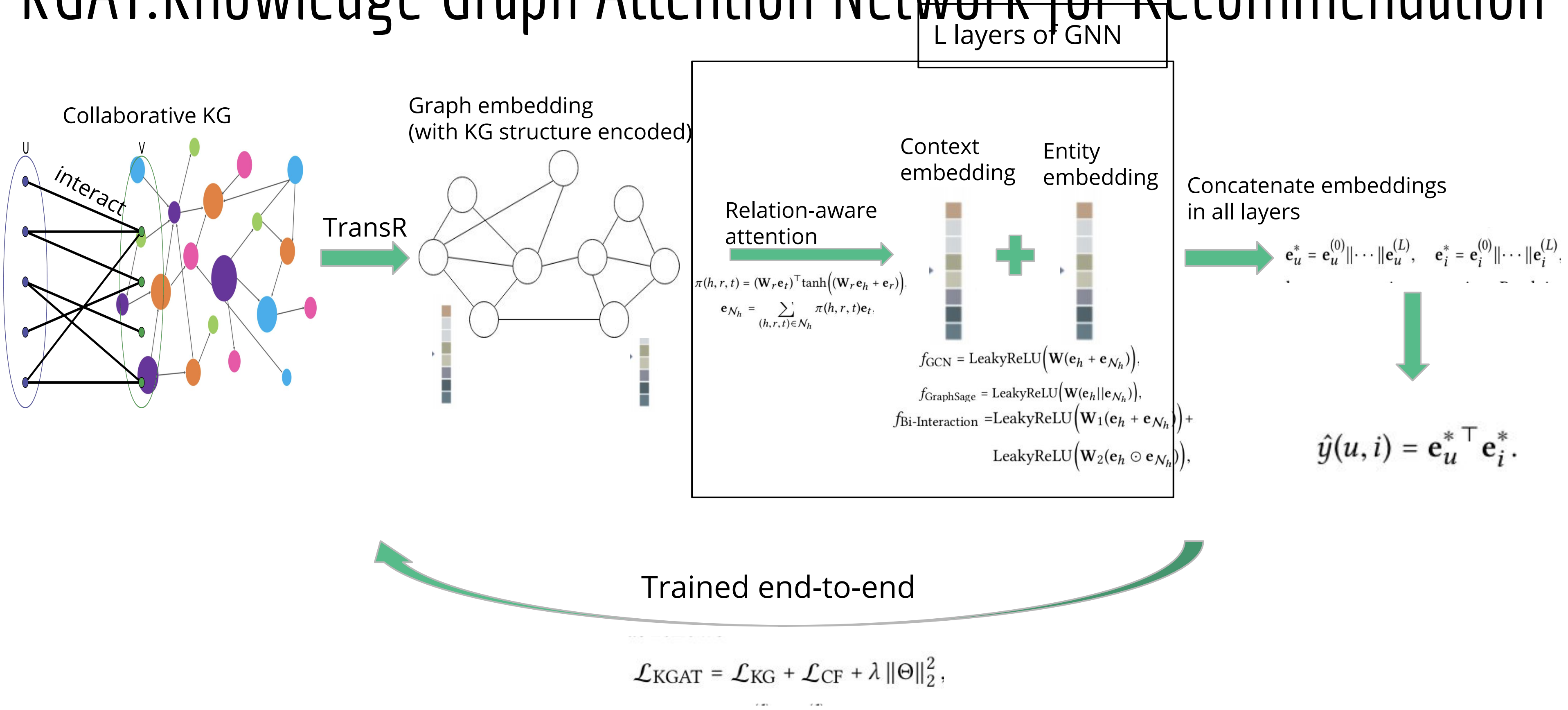
KGCN/KGNN-LS

- Can alleviate cold-start scenarios where user-item interactions are sparse

r	20%	40%	60%	80%	100%
SVD	0.882	0.913	0.938	0.955	0.963
LibFM	0.902	0.923	0.938	0.950	0.959
LibFM+TransE	0.914	0.935	0.949	0.960	0.966
PER	0.802	0.814	0.821	0.828	0.832
CKE	0.898	0.910	0.916	0.921	0.924
RippleNet	0.921	0.937	0.947	0.955	0.960
KGNN-LS	**0.961**	**0.970**	**0.974**	**0.977**	**0.979**

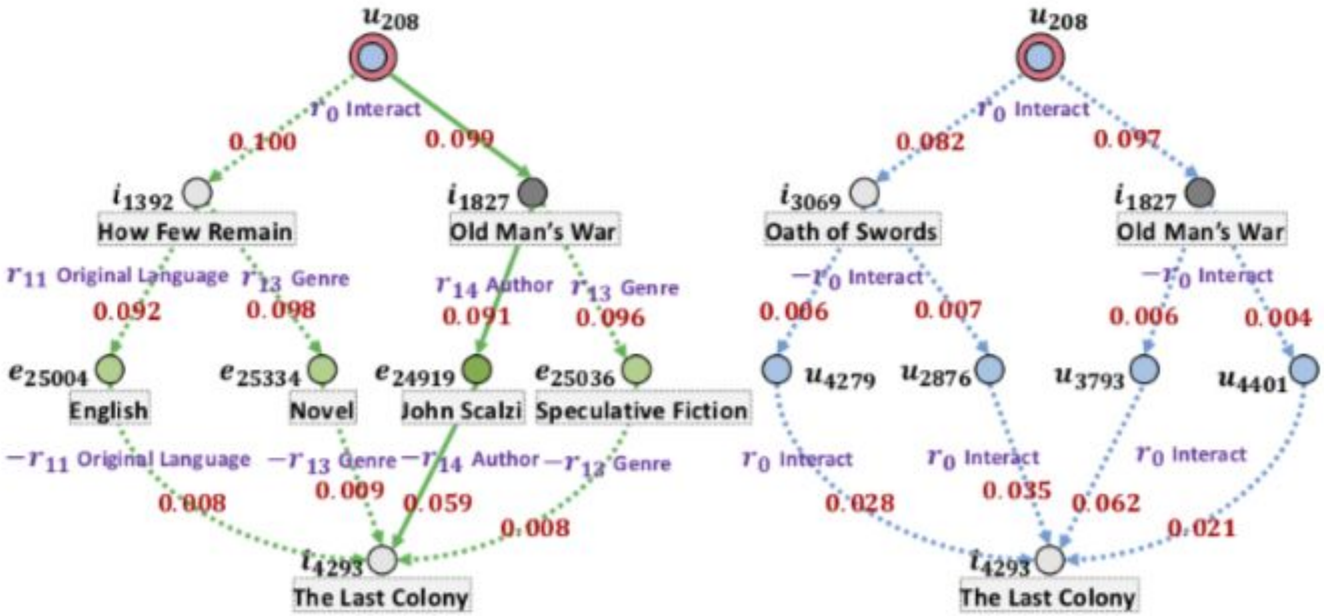
Table 5: *AUC* of all methods w.r.t. the ratio of training set *r*.

KGAT:Knowledge Graph Attention Network for Recommendation



KGAT

- Can reason on high-order connectivity to infer the user preferences on the target item, offering explanations.

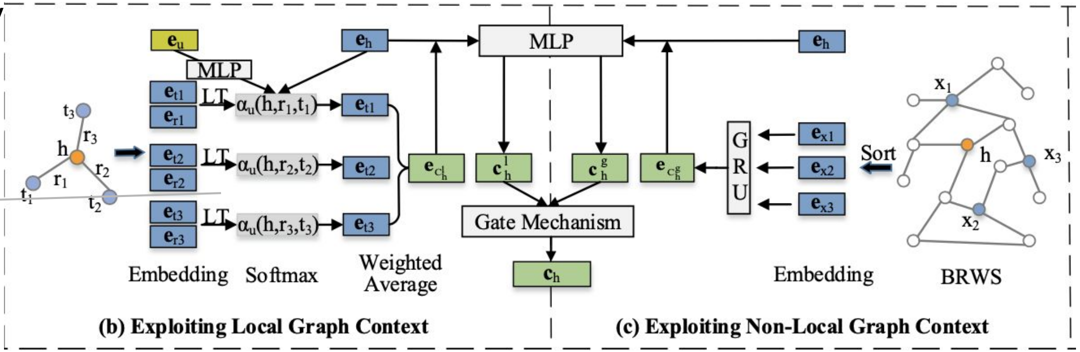


CGAT: Contextualized Graph Attention Network for Recommendation with Item Knowledge Graph

- Used user-specific graph attention mechanism in KG to capture each entity's local graph context in knowledge graph
  - Local graph context: a set of its first order neighbors
- Include a biased random walk based GRU module to capture non-local context in knowledge graph
  - Non-local context: the set of its most related high order neighbors
  - repeated random walk from h to obtain M path with length L
  - Sort entities based on frequency

$$e_{\mathcal{C}_h^l} = \sum_{t \in \mathcal{C}_h^l} \alpha_u(h, r, t) e_t.$$

$$\alpha_u(h, r, t) = \frac{\exp\left[\pi_u(h, r, t)\right]}{\sum_{(h, \tilde{r}, \tilde{t}) \in \mathcal{D}} \exp\left[\pi_u(h, \tilde{r}, \tilde{t})\right]}$$



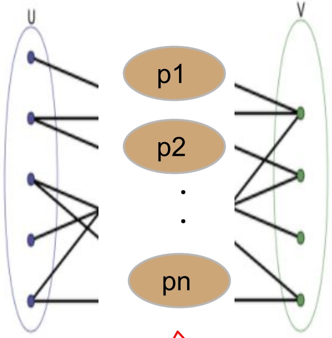
Relation-aware

3. Explicit relation aggregation

- Explicitly incorporate relation embedding in context embedding

KGIN:Knowledge Graph-based Intent Network

Intent Graph



Model intent into user embedding

$$\mathbf{e}_u^{(1)} = \frac{1}{|\mathcal{N}_u|} \sum_{(p,i)\in\mathcal{N}_u} \beta(u,p)\mathbf{e}_p \odot \mathbf{e}_i^{(0)},$$

$$\beta(u,p) = \frac{\exp(\mathbf{e}_p^\top \mathbf{e}_u^{(0)})}{\sum_{p'\in\mathcal{P}} \exp(\mathbf{e}_{p'}^\top \mathbf{e}_u^{(0)})},$$

L layers

$$\mathbf{e}_u^{(l)} = f_{\text{IG}}\Big(\{(\mathbf{e}_u^{(l-1)}, \mathbf{e}_p, \mathbf{e}_i^{(l-1)})|(p,i)\in\mathcal{N}_u\}\Big),$$

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

Model relation embedding into item embedding

$$\mathbf{e}_i^{(1)} = \frac{1}{|\mathcal{N}_i|} \sum_{(r,v)\in\mathcal{N}_i} \mathbf{e}_r \odot \mathbf{e}_v^{(0)},$$

L layers

$$\boxed{\mathbf{e}_i^{(l)} = f_{\text{KG}}\Big(\{(\mathbf{e}_i^{(l-1)}, \mathbf{e}_r, \mathbf{e}_v^{(l-1)})|(r,v)\in\mathcal{N}_i\}\Big)}$$

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

$$\mathbf{e}_i^{(l)} = \sum_{s\in\mathcal{N}_i^l} \frac{\mathbf{e}_{r_1}}{|\mathcal{N}_{s_1}|} \odot \frac{\mathbf{e}_{r_2}}{|\mathcal{N}_{s_2}|} \odot \cdots \odot \frac{\mathbf{e}_{r_l}}{|\mathcal{N}_{s_l}|} \odot \mathbf{e}_{s_l}^{(0)},$$

$$\hat{y}_{ui} = \mathbf{e}_u^{*\top}\mathbf{e}_i^*.$$

attentive combination of relation embeddings + independence constraint

$$\mathbf{e}_p = \sum_{r\in\mathcal{R}} \alpha(r,p)\mathbf{e}_r,$$

$$\alpha(r,p) = \frac{\exp(w_{rp})}{\sum_{r'\in\mathcal{R}} \exp(w_{r'p})},$$

$$\mathcal{L}_{\text{IND}} = \sum_{p,p'\in\mathcal{P},\, p\neq p'} dCor(\mathbf{e}_p, \mathbf{e}_{p'}),$$

Trained end-to-end

$$\mathcal{L}_{\text{KGIN}} = \mathcal{L}_{\text{BPR}} + \lambda_1 \mathcal{L}_{\text{IND}} + \lambda_2 \|\Theta\|_2^2,$$

KGIN

- KGIN creates instance-wise explanations for each interaction — the personalization of a single user

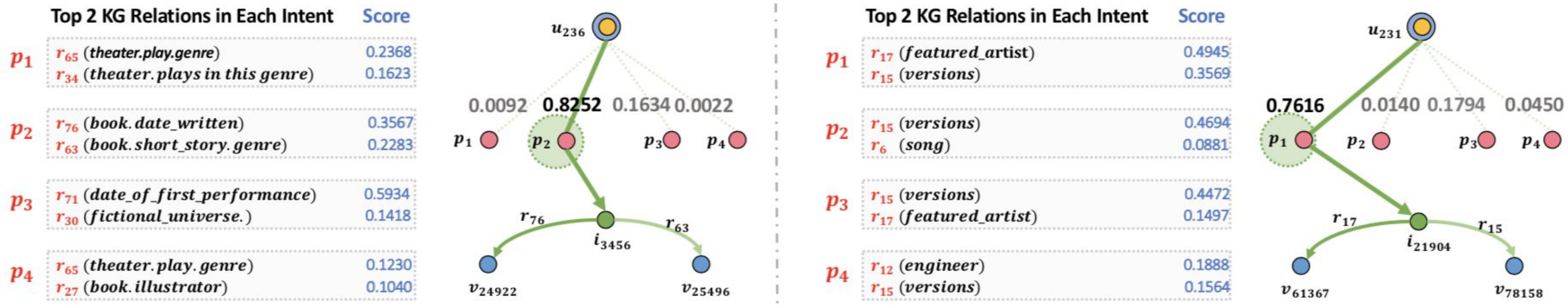


Figure 5: Explanations of user intents and real cases in Amazon-Book (left) and Last-FM (right). Best viewed in color.

Social Network Domain

GraphRec: Graph Neural Network for Social Recommendation

(3) normalizing via softmax function to get final latent score

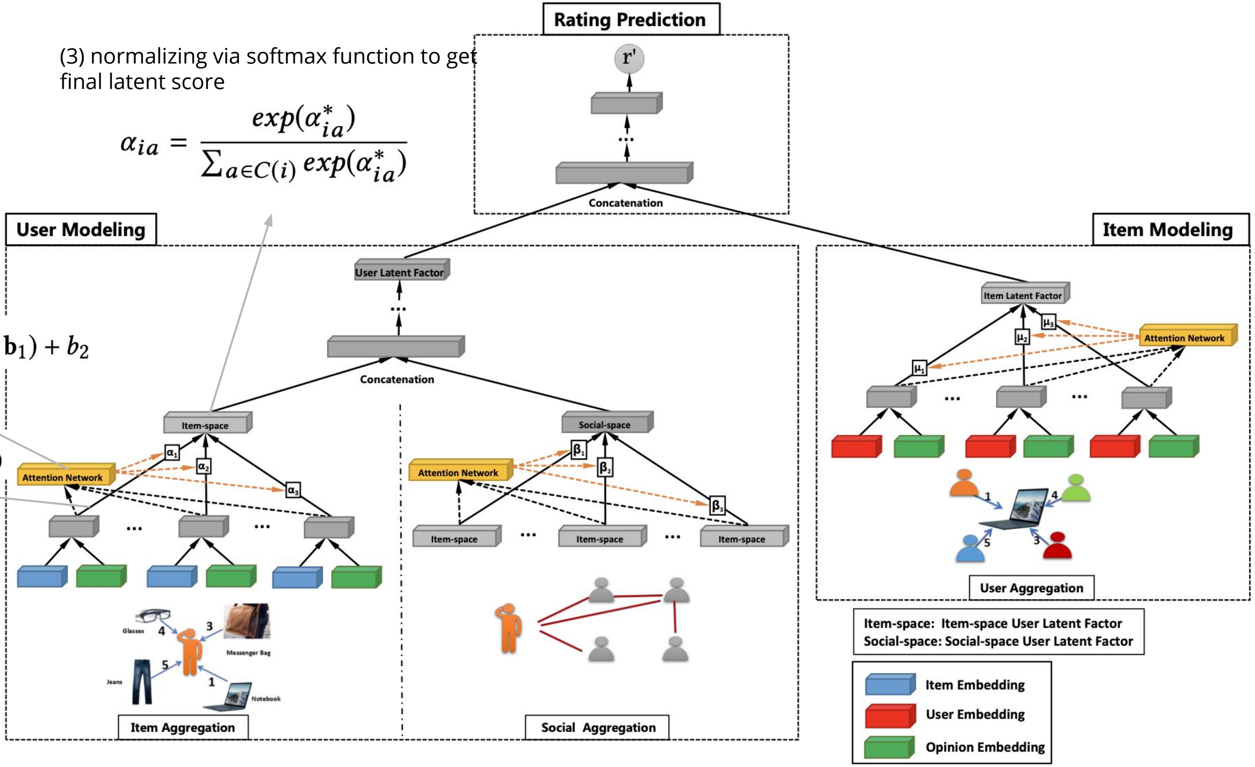
$$\alpha_{ia} = \frac{exp(\alpha^*_{ia})}{\sum_{a \in C(i)} exp(\alpha^*_{ia})}$$

(2) Attention mechanism to alleviate drawbacks of mean based aggregator

$$\alpha^*_{ia} = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{x}_{ia} \oplus \mathbf{p}_i] + \mathbf{b}_1) + b_2$$

(1) Aggregation to obtain latent factor

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



DGRec: Session Based Social Recommendation via Dynamic Graph Attention Networks

The number of layers L of the networks corresponds to the number of iterations of message passing

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

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

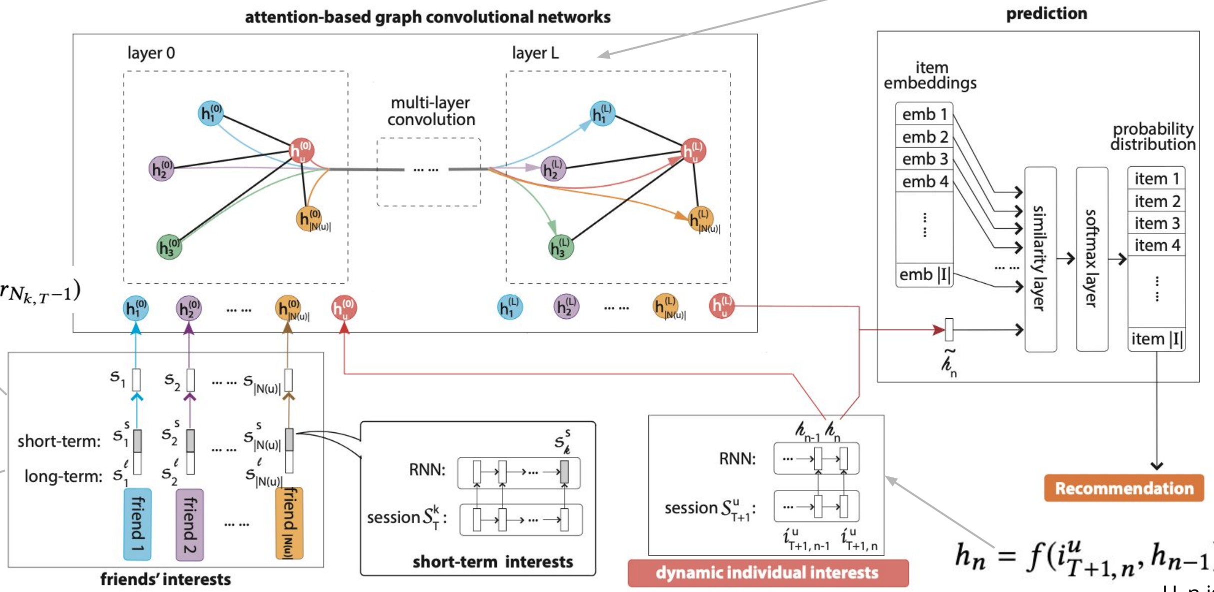


Figure 2: A schematic view of our proposed model for dynamic social recommendation.

$$h_n = f(i^u_{T+1,n}, h_{n-1}),$$

H_n is representation of user's interest
S represents a session which includes all user's actions (clicks)
f is LSTM

DGRec: Session Based Social Recommendation via Dynamic Graph Attention Networks

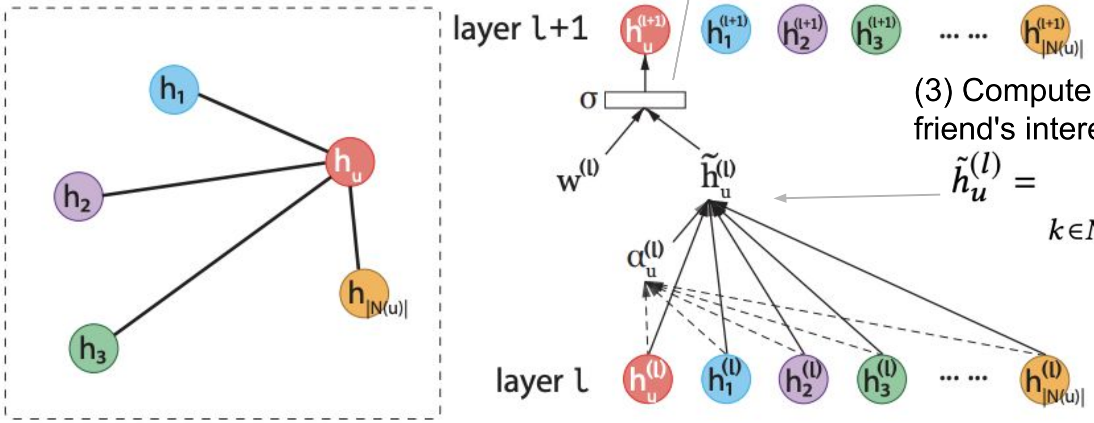
(4) non-linear transformation

$$h_u^{(l+1)} = ReLU(\mathbf{W}^{(l)} \tilde{h}_u^{(l)})$$

(1) similarity between the target user's node representation h (l) u and all of its neighbors' representations h (l) k

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

(2) include self-connection edge to preserve a user's revealed interests



(3) Compute mixture of user u's friend's interest at layer K

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

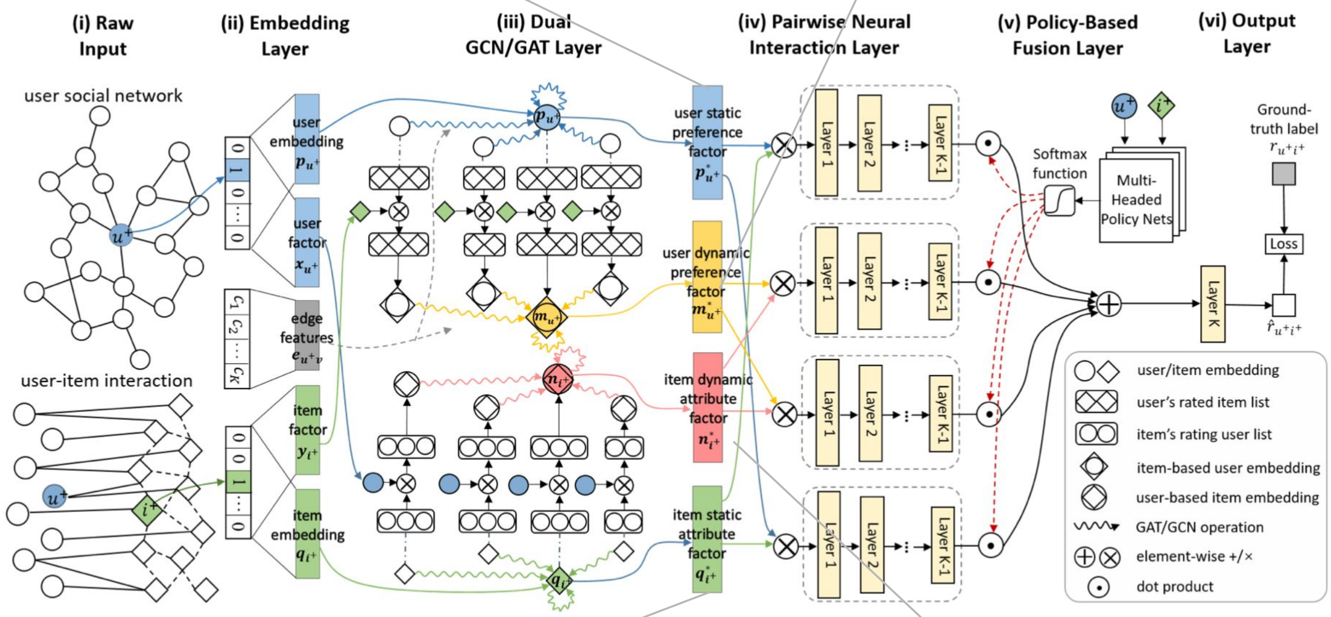
Dual Graph Attention Networks

$$\alpha^P_{uv} = \frac{attn_U(\mathbf{W}_P\mathbf{p}_u, \mathbf{W}_P\mathbf{p}_v, \mathbf{W}_E\mathbf{e}_{uv})}{\sum_{w\in\Gamma_U(u)} attn_U(\mathbf{W}_P\mathbf{p}_u, \mathbf{W}_P\mathbf{p}_w, \mathbf{W}_E\mathbf{e}_{uv})}, v \in \Gamma_U(u),$$

$$\mathbf{P}^* = \sigma(\mathbf{A}_P(G_U)\mathbf{P}\mathbf{W}_P^T + \mathbf{b}_P),$$

$$\mathbf{M}^*_{i^+} = \sigma(\mathbf{A}_M(G_U)\mathbf{M}\mathbf{W}_M^T + \mathbf{b}_M), \mathbf{A}_M(G_U) = \{\alpha^M_{uv,i^+}\}_{M\times M},$$

$$\alpha^M_{uv,i^+} = \frac{attn_U(\mathbf{W}_M\mathbf{m}_u^{i^+}, \mathbf{W}_M\mathbf{m}_v^{i^+}, \mathbf{W}_E\mathbf{e}_{uv})}{\sum_{w\in\Gamma_U(u)} attn_U(\mathbf{W}_M\mathbf{m}_u^{i^+}, \mathbf{W}_M\mathbf{m}_w^{i^+}, \mathbf{W}_E\mathbf{e}_{uv})},$$



$$\mathbf{Q}^* = \sigma(\mathbf{A}_Q(G_I)\mathbf{Q}\mathbf{W}_Q^T + \mathbf{b}_Q), \mathbf{A}_Q(G_I) = \{\alpha^Q_{ij}\}_{N\times N},$$

$$\alpha^Q_{ij} = \frac{attn_I(\mathbf{W}_Q\mathbf{q}_i, \mathbf{W}_Q\mathbf{q}_j)}{\sum_{k\in\Gamma_I(i)} attn_I(\mathbf{W}_Q\mathbf{q}_i, \mathbf{W}_Q\mathbf{q}_k)}, j \in \Gamma_I(i),$$

$$\mathcal{X}_i^{u^+} = \{\mathbf{x}_v \otimes \mathbf{x}_{u^+} | v \in R_I(i)\},$$

$$n^{u^+}_{id} = \max_{v\in R_U(i)} \{x_{vd} \cdot x_{u^+d}\} \quad \forall d = 1, \ldots, D,$$

Common Datasets

Scenario	Dataset	# Entities	# Connections	# Relation Types
Book	Amazon-books	95,594	847,733	39
	Book-crossing	25,787	60,787	18
Movie	Douban	46,423	331,315	5
	Flixter	1,049,000	26,700,000	-
	MovieLens	102,569	499,474	32
Music	Last-FM	9,336	15,518	60
POI	Delicious	5,932	15,328	-
	Yelp	159,426	6,818,026	6
	Dianping	28,115	160,519	7
Social Network	Epinions	175,000	508,000	-

Common Metrics

- Precision@K: (# of recommended items @k that are relevant) / (# of recommended items @k)
- Recall@K: (# of recommended items @k that are relevant) / (total # of relevant items)
- MRR@K (Mean Reciprocal Rank): $$\mathrm{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathrm{rank}_i}.$$
- NDCG@k (Normalized Discounted Cumulative Gain@K): $$NDCG@k = \frac{DCG@k}{IDCG@k}$$
- AUC
- RMSE

Future Directions

- Dynamic Graphs
  - How to efficiently & incrementally update representations?

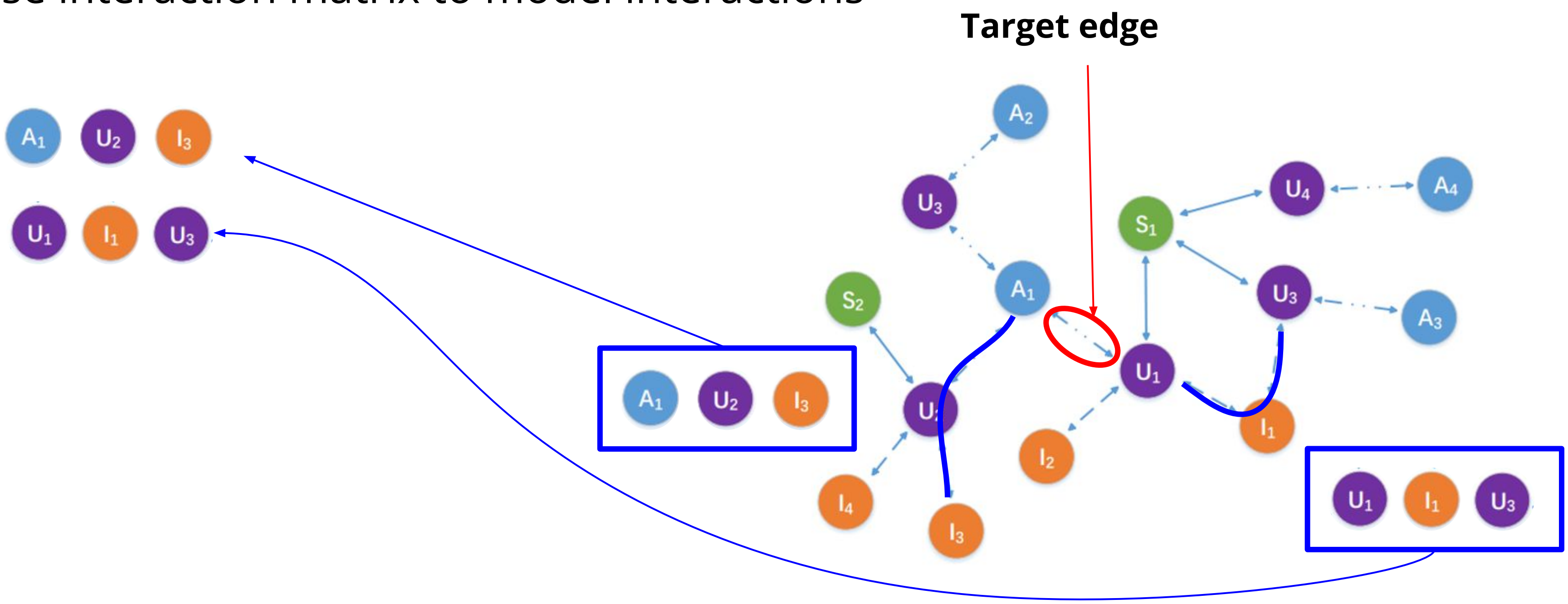
- Explainable Recommendation

Thank you!

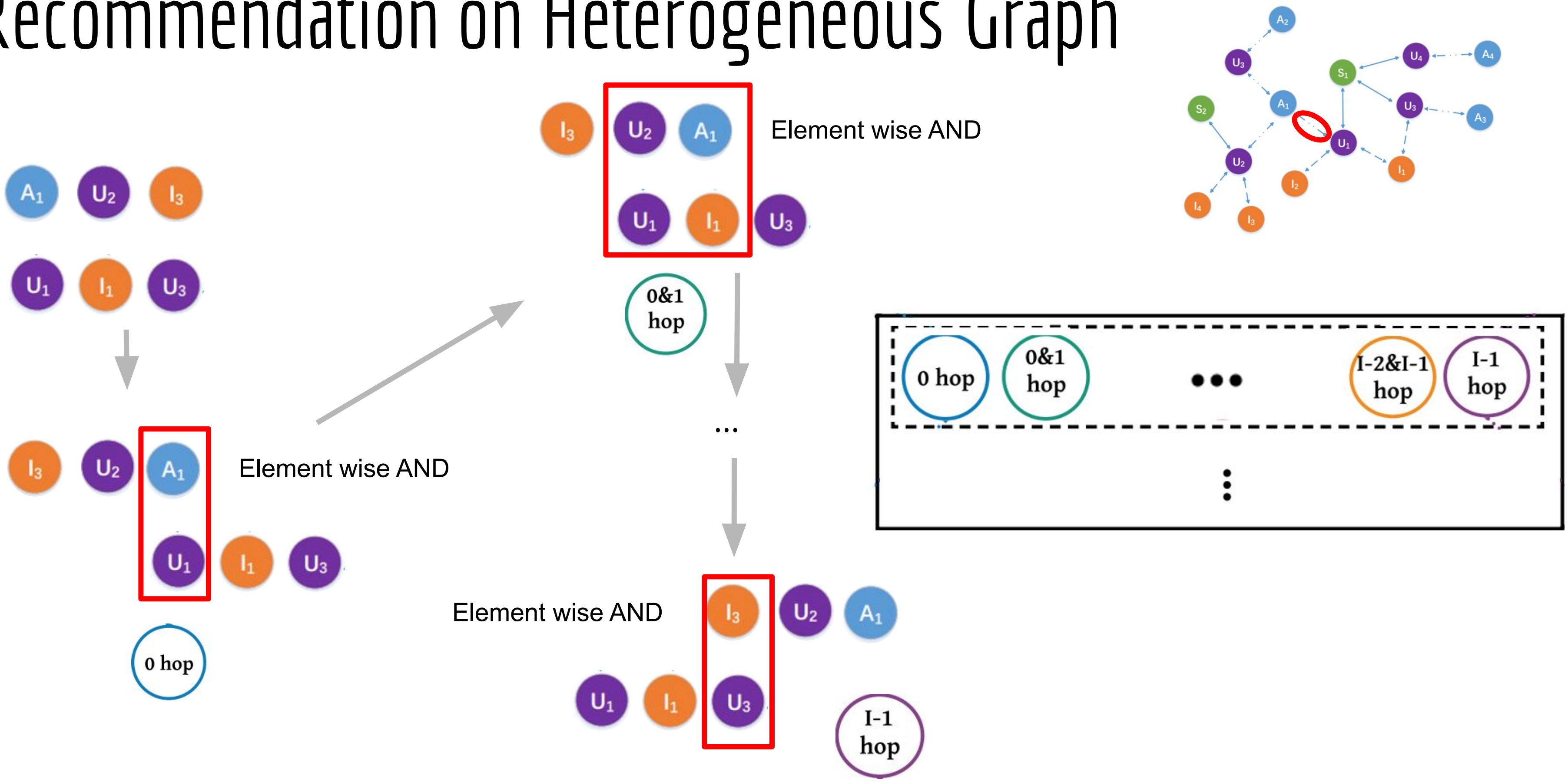
Embed edge instead of node

An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph

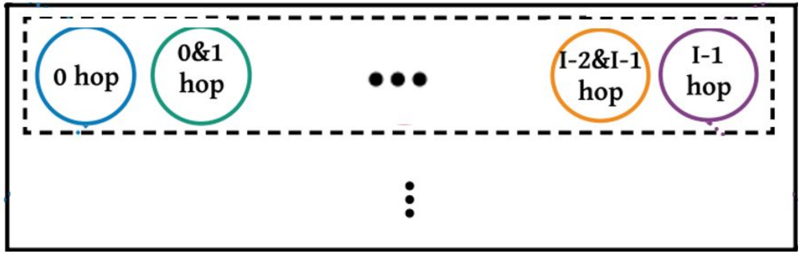
Use interaction matrix to model interactions



An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph



An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph



Measure importance of each sub-interaction trainable matrices

Importance value and sub-interaction embedding used to train matrix that gives the embedding of the overall interaction