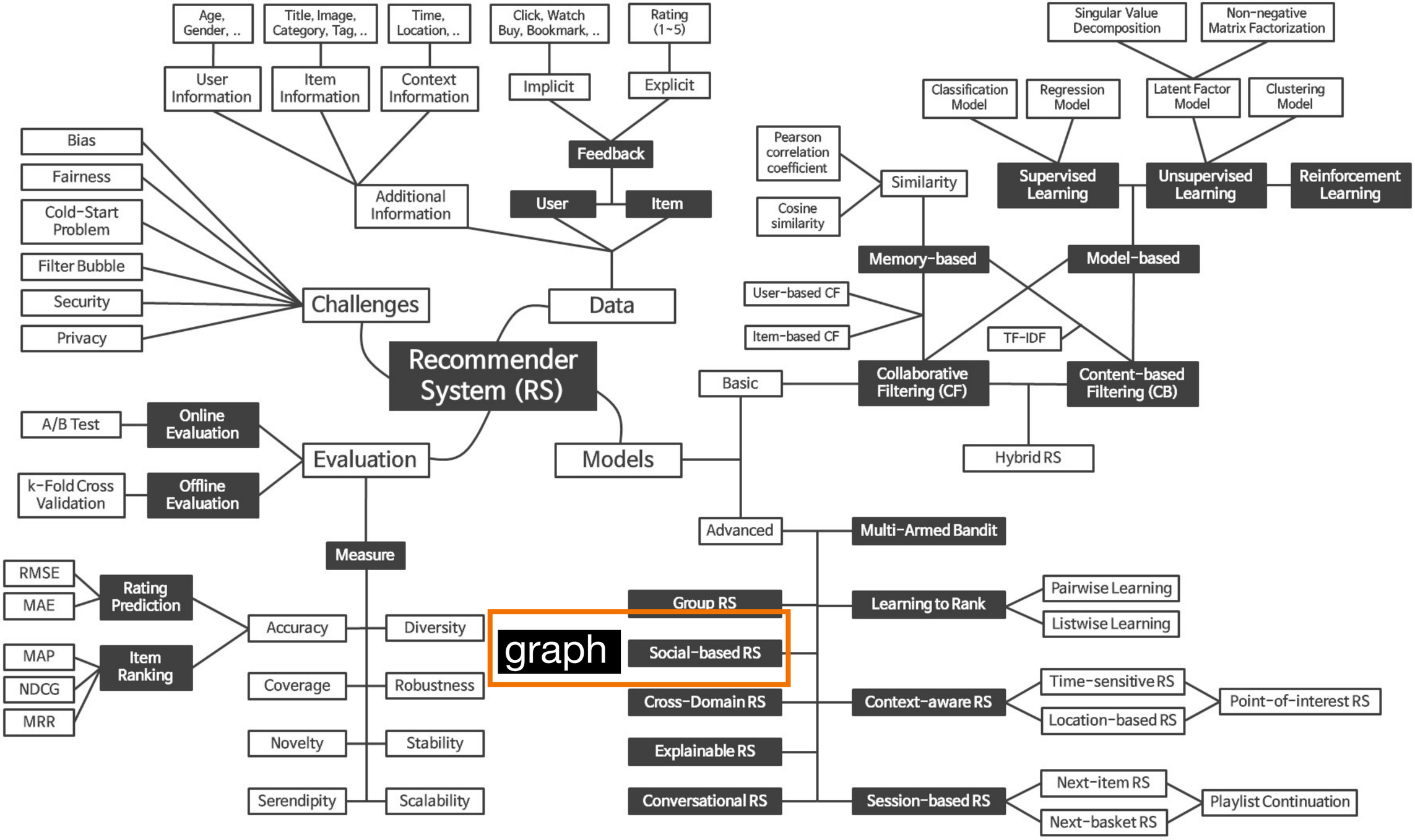
Графовые сети в рекомендательных системах

Краснов Александр, 12.05.2025, AI masters


Recommender System (RS)
Challenges
Bias
Fairness
Cold-Start Problem
Filter Bubble
Security
Privacy
Data
Additional Information
User Information
Item Information
Context Information
Feedback
Implicit
Explicit
User
Item
Age, Gender, ..
Title, Image, Category, Tag, ..
Time, Location, ..
Click, Watch Buy, Bookmark, ..
Rating (1~5)
Evaluation
Online Evaluation
A/B Test
Offline Evaluation
k-Fold Cross Validation
Measure
Rating Prediction
RMSE
MAE
Item Ranking
MAP
NDCG
MRR
Accuracy
Diversity
Coverage
Robustness
Novelty
Stability
Serendipity
Scalability
Models
Basic
Collaborative Filtering (CF)
Memory-based
Model-based
Similarity
Pearson correlation coefficient
Cosine similarity
User-based CF
Item-based CF
Content-based Filtering (CB)
TF-IDF
Hybrid RS
Supervised Learning
Classification Model
Regression Model
Unsupervised Learning
Latent Factor Model
Clustering Model
Singular Value Decomposition
Non-negative Matrix Factorization
Reinforcement Learning
Advanced
Multi-Armed Bandit
Learning to Rank
Pairwise Learning
Listwise Learning
Context-aware RS
Time-sensitive RS
Location-based RS
Point-of-interest RS
Session-based RS
Next-item RS
Next-basket RS
Playlist Continuation
Group RS
Social-based RS
Cross-Domain RS
Explainable RS
Conversational RS
graph

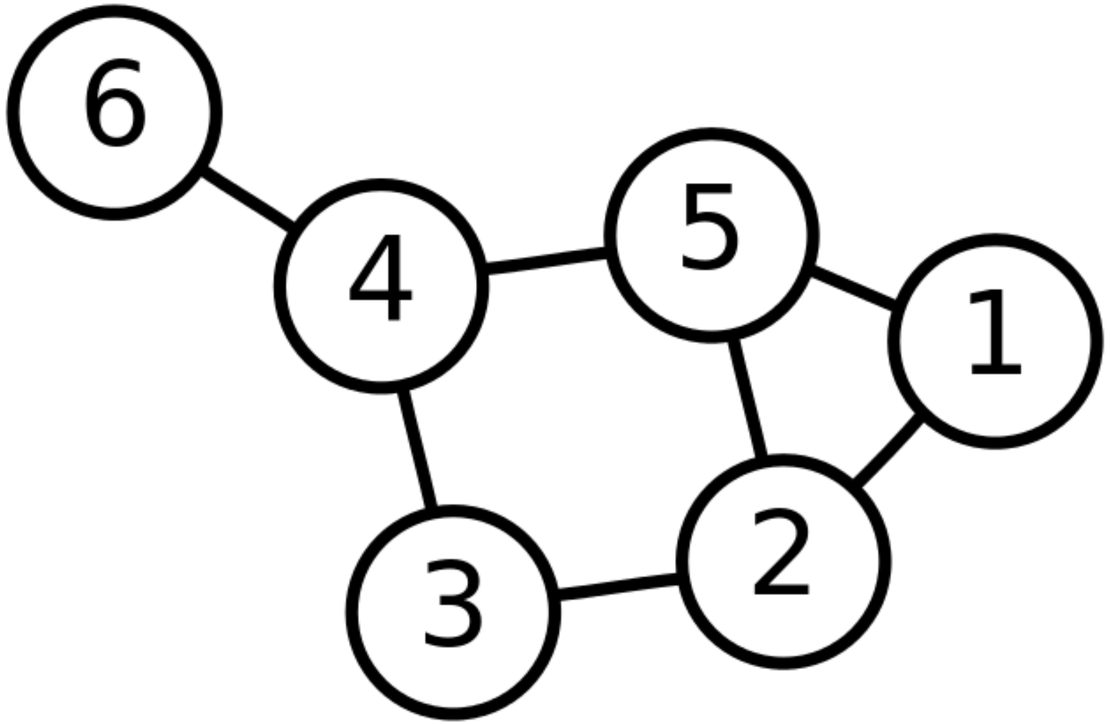
Графы

Введение

Определение. *Простой граф* $G(V, E)$ есть совокупность двух множеств — непустого множества V и множества E неупорядоченных пар различных элементов множества V . Множество V называется *множеством вершин*, множество E называется *множеством рёбер*

$$G(V, E) = \langle V, E \rangle, \quad V \neq \varnothing, \quad E \subseteq V \times V, \quad \{v, v\} \notin E, \quad v \in V,$$

то есть множество E состоит из 2-элементных подмножеств множества V.



Графы

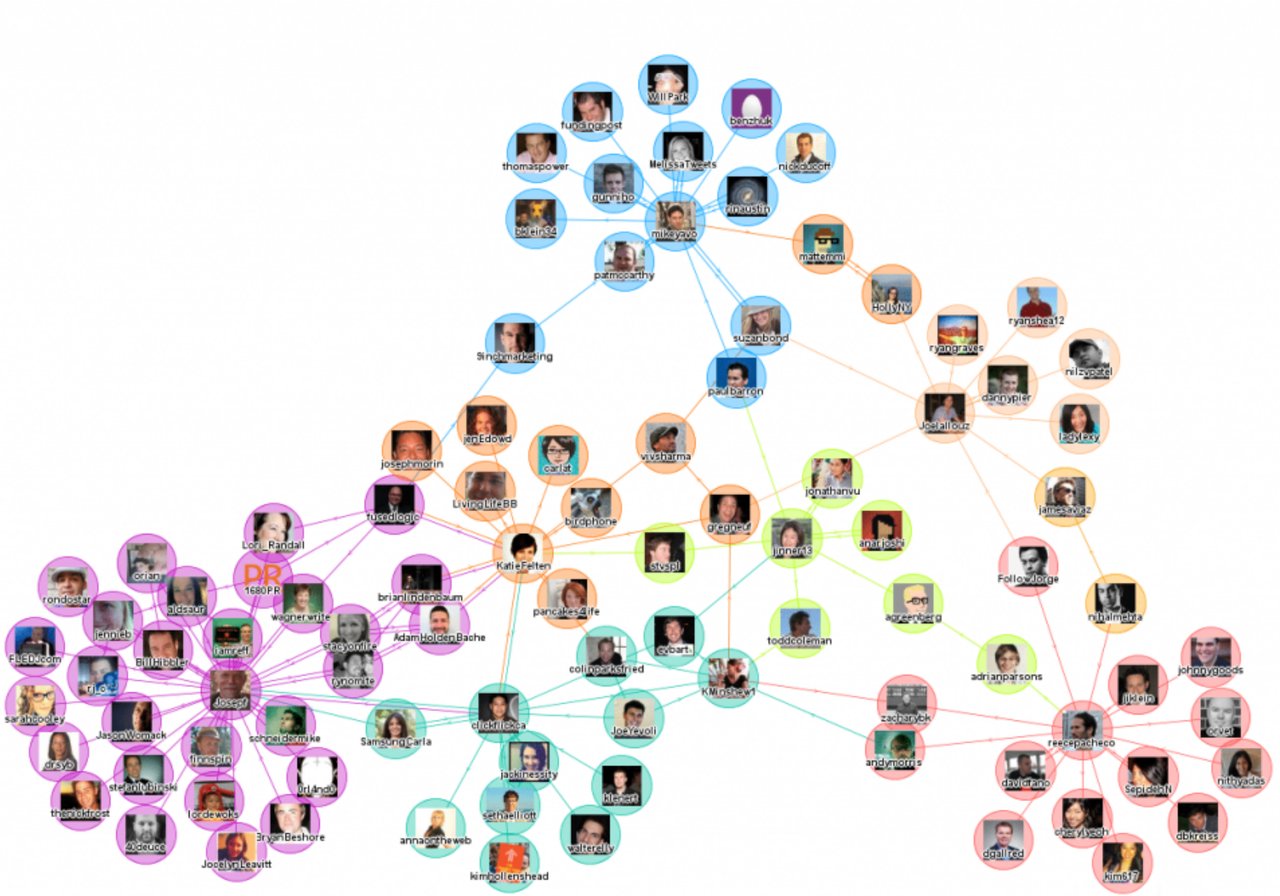
Введение

- Вершины - пользователи, товары, категории, запросы и т.д.
- Ребра - подписка, покупка, клик, прослушивание, рейтинг или любое другое действие

Графы

User-user graph

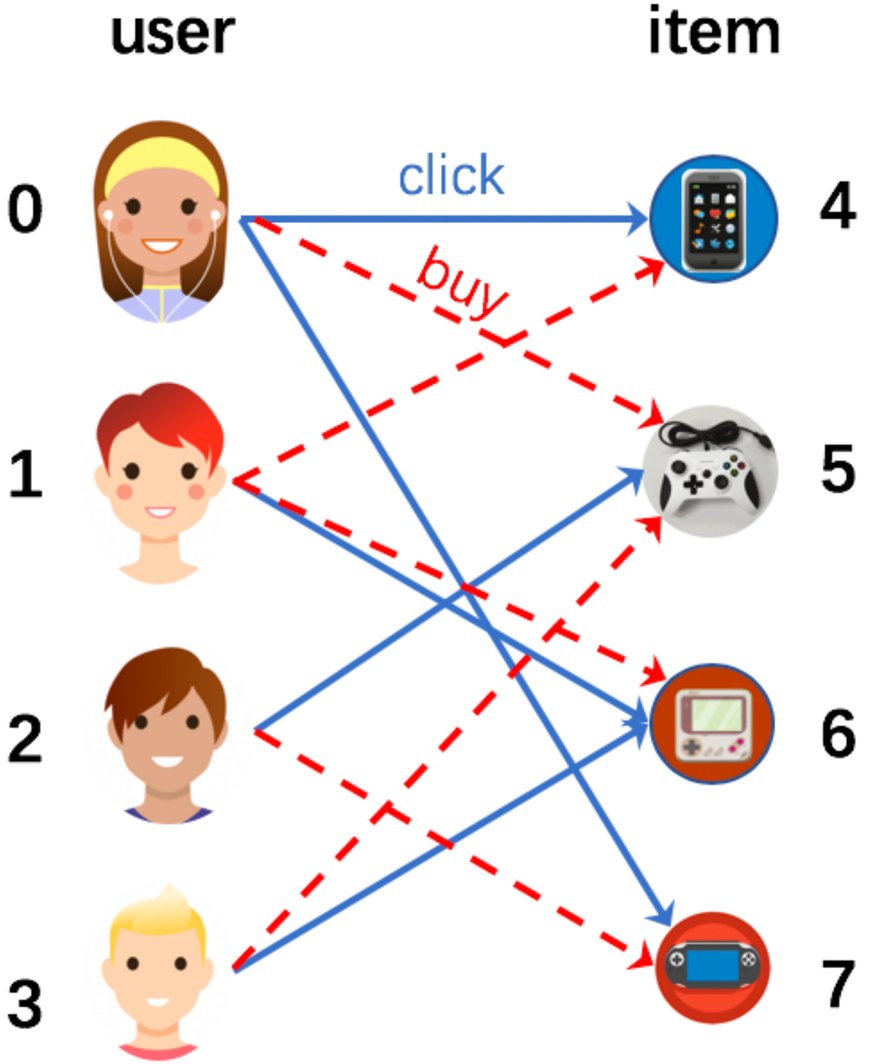
- V - множество пользователей
- E - взаимодействия между пользователями (дружба)
- Пример - рекомендация новых друзей



Графы

User-item graph

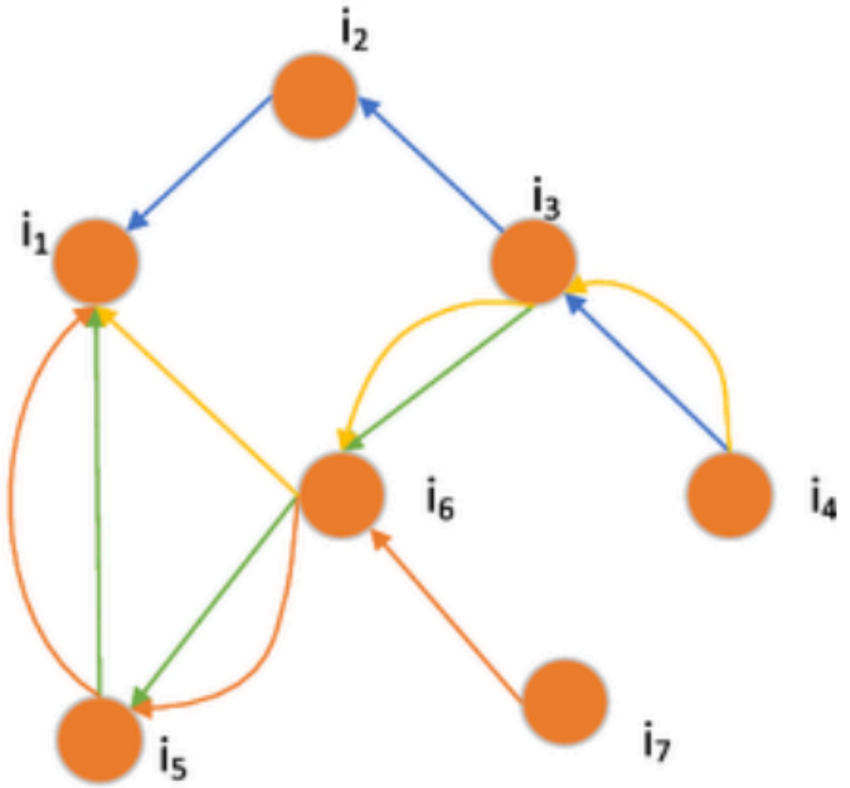
- V - множество пользователей и айтемов
- E - взаимодействия между пользователями и айтемами (покупка, прослушивание)
- Пример - рекомендации товаров/ музыки



Графы

Item-item graph

- V - множество айтемов
- E - взаимодействия между товарами (были в одном заказе/ сессии) или похожесть
- Пример - рекомендации похожих/ сопутствующих товаров

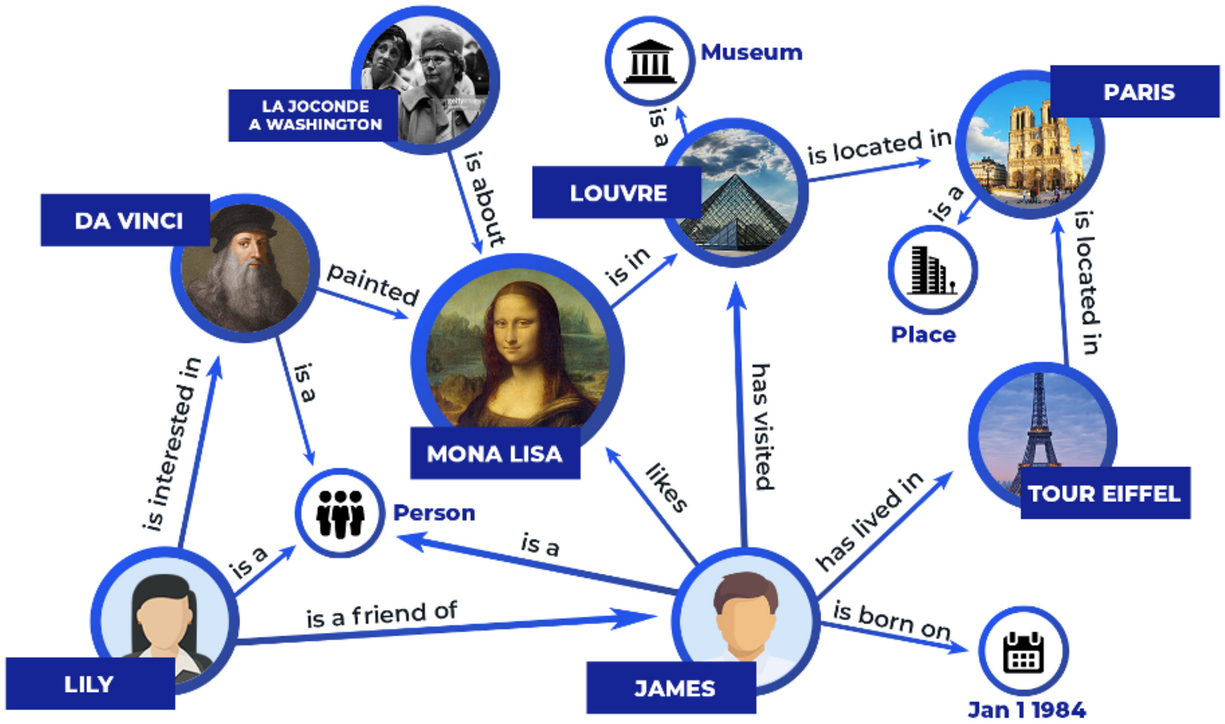


(c) The item-item sequential graph

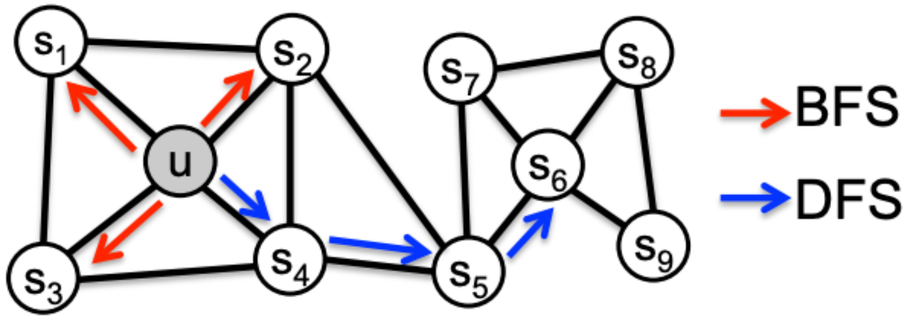
Графы

Knowledge graph

- V - множество различный объектов
- E - отношения между объектами
- Атрибут - дополнительная информация об объектах или ребрах



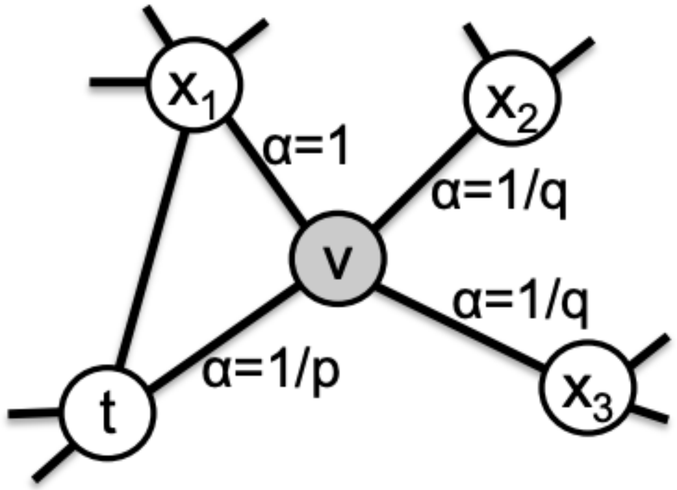
Random walk



→BFS

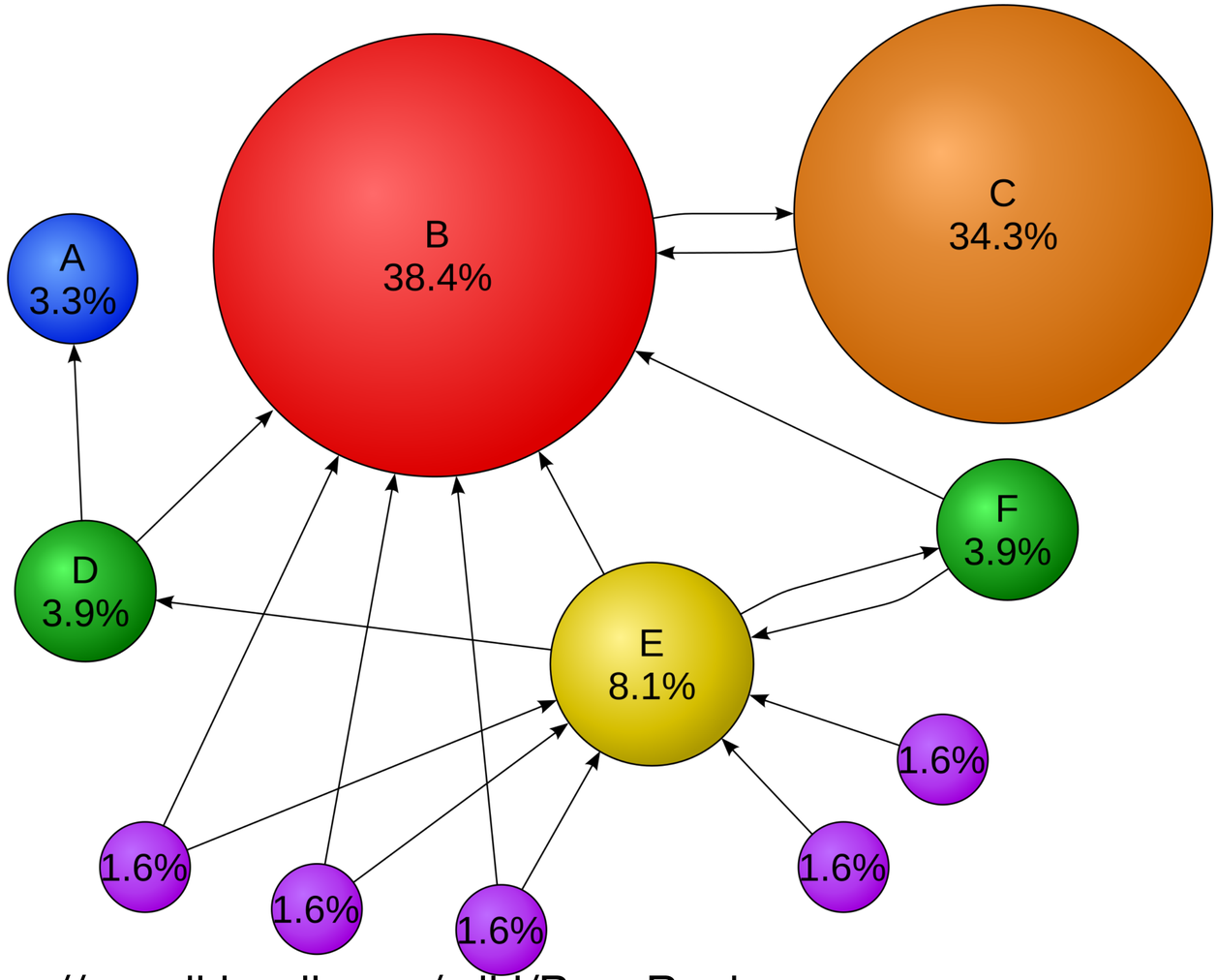
→DFS

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



node2vec: Scalable Feature Learning for Networks

Page rank



https://en.wikipedia.org/wiki/PageRank

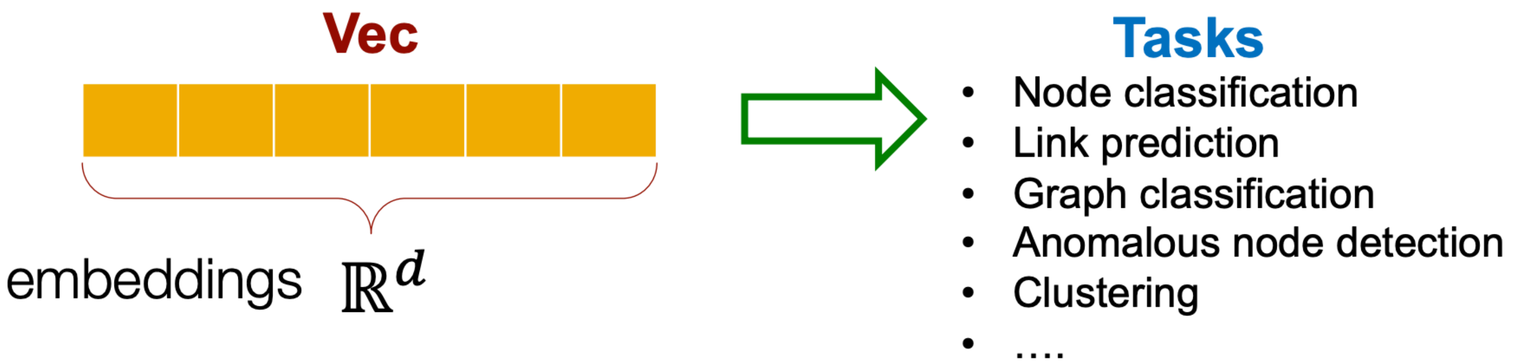
$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

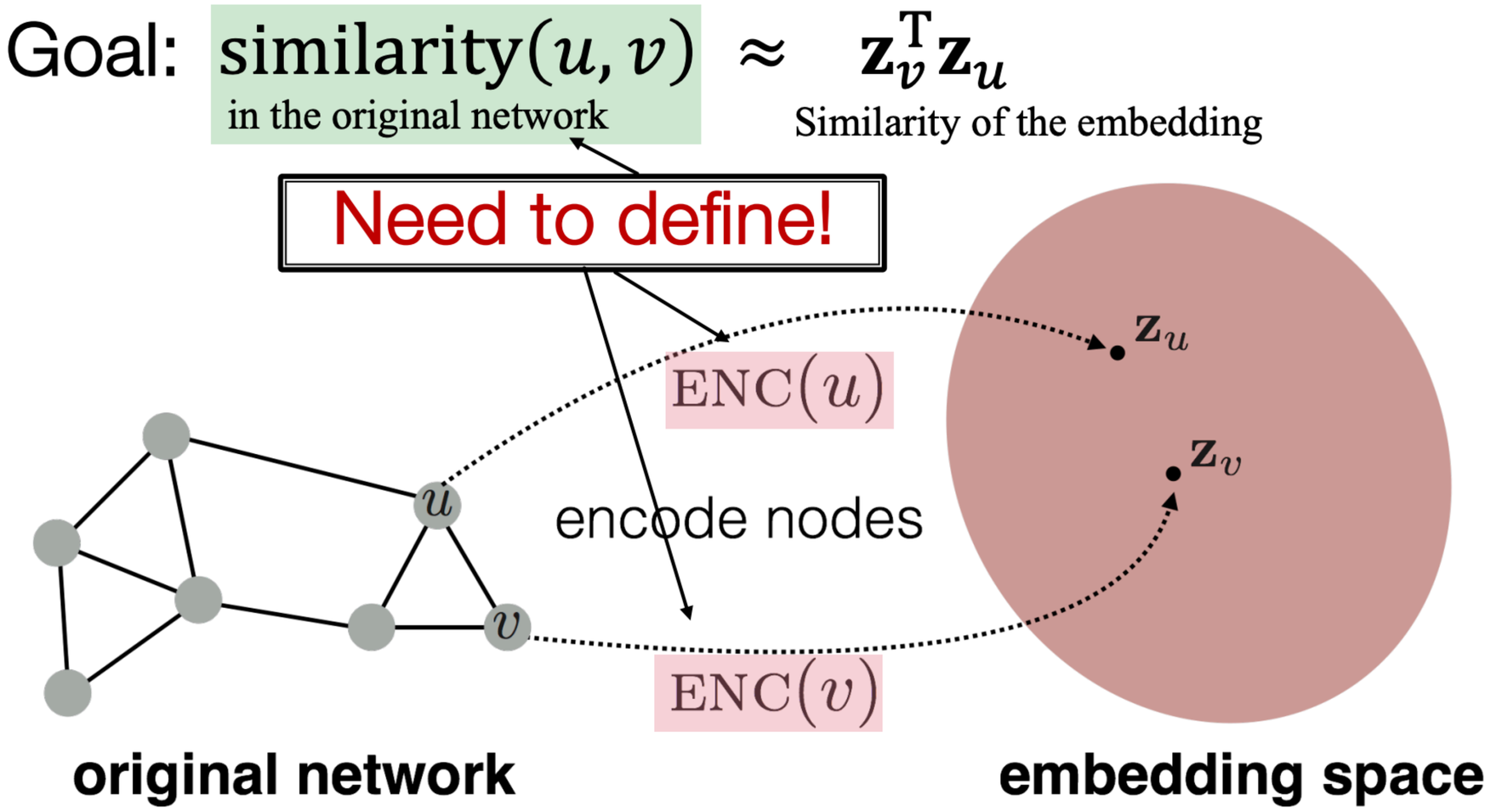
Вопросы

Embedding

- Задача: перевести ноды в эмбеддинги
- Embedding similarity = похожесть нод в графе (например есть ребро)
- Нужно построить энкодер ноды в эмбеддинг

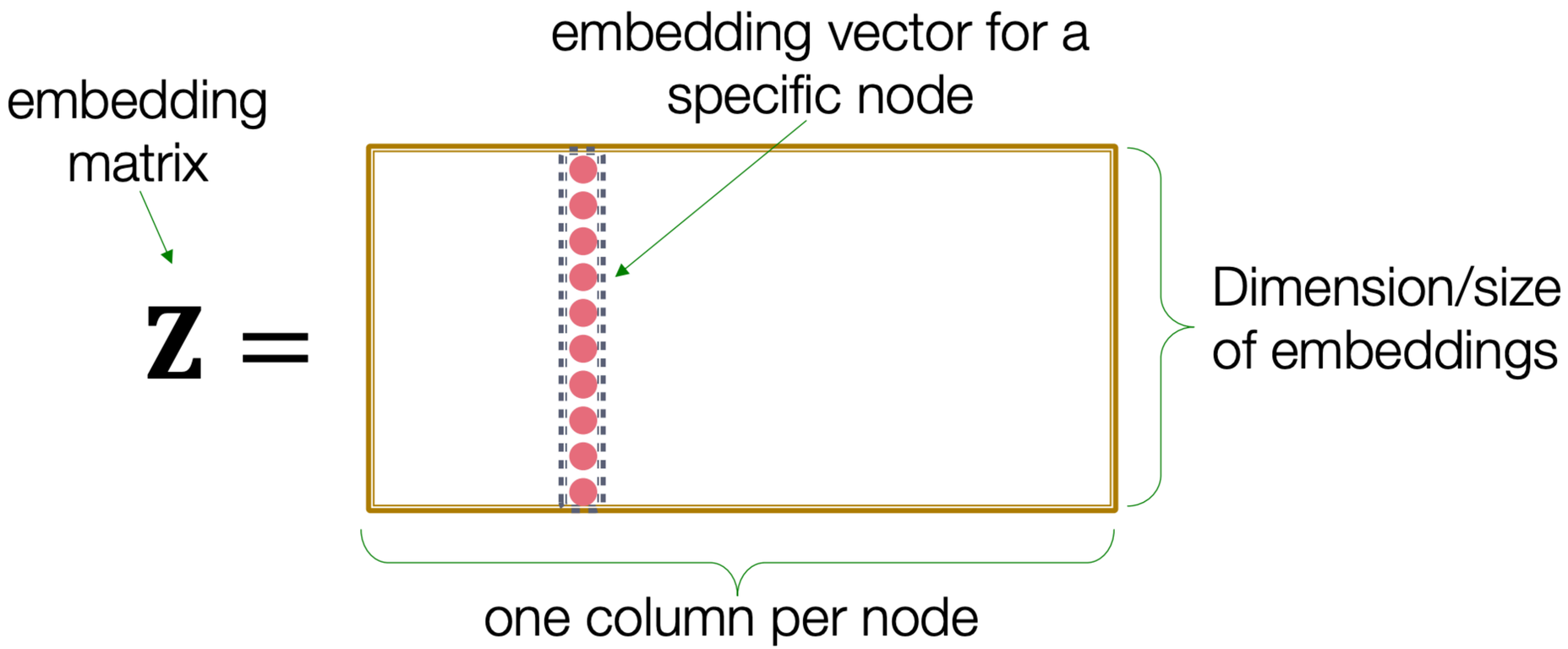


Embedding


Goal: similarity(u, v) $\approx$ $\mathbf{z}_v^{\mathrm{T}}\mathbf{z}_u$
in the original network
Similarity of the embedding
Need to define!
ENC(u)
ENC(v)
encode nodes
u
v
$\mathbf{z}_u$
$\mathbf{z}_v$
original network
embedding space

Embedding

Simplest encoding approach: **encoder is just an embedding-lookup**



Embedding

- **Encoder:** maps each node to a low-dimensional vector

$$\text{ENC}(v) = \mathbf{z}_v$$

d-dimensional embedding

node in the input graph

- **Similarity function:** specifies how the relationships in vector space map to the relationships in the original network

$$\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$$

Decoder

Similarity of u and v in the original network

dot product between node embeddings

Node similarity

Как определить похожесть node?

- Есть ребро
- Есть 1, 2... общих соседа
- Встречаемость при random walk

Random walk -> node2vec

1. Run **short fixed-length random walks** starting from each node u in the graph using some random walk strategy R.
2. For each node u collect $N_R(u)$, the multiset* of nodes visited on random walks starting from u.
3. Optimize embeddings according to: Given node u, predict its neighbors $N_R(u)$.

$$\arg\max_z \sum_{u \in V} \log \mathrm{P}(N_R(u) | \mathbf{z}_u)$$ ⟹ Maximum likelihood objective

*$N_R(u)$ can have repeat elements since nodes can be visited multiple times on random walks

$$\arg\min_z \mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}$$

$$\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}\right)$$

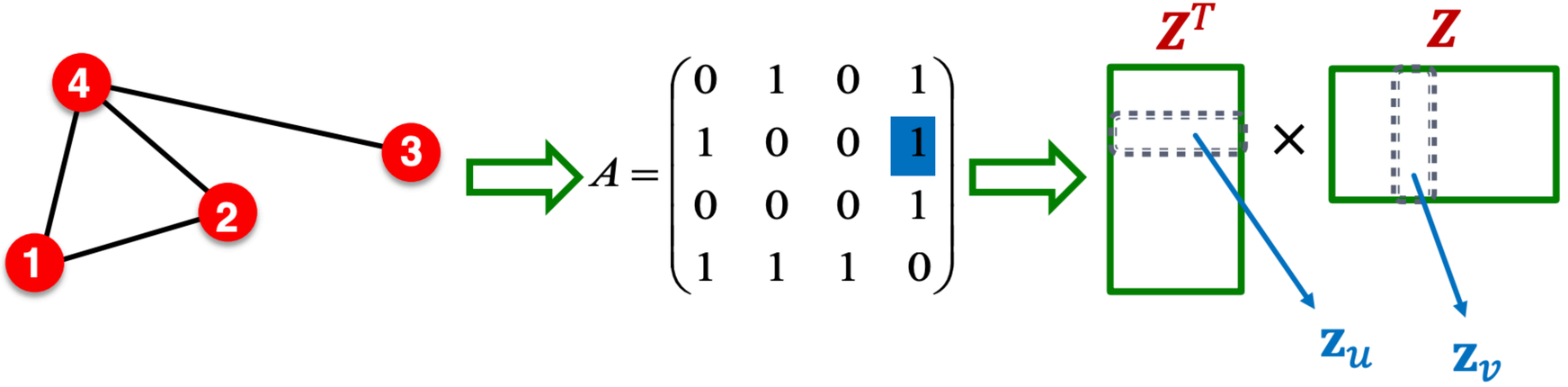
random distribution over nodes

$$\approx \log\left(\sigma(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)\right) + \sum_{i=1}^{k} \log\left(\sigma(-\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_{n_i})\right), n_i \sim P_V$$

node2vec: Scalable Feature Learning for Networks

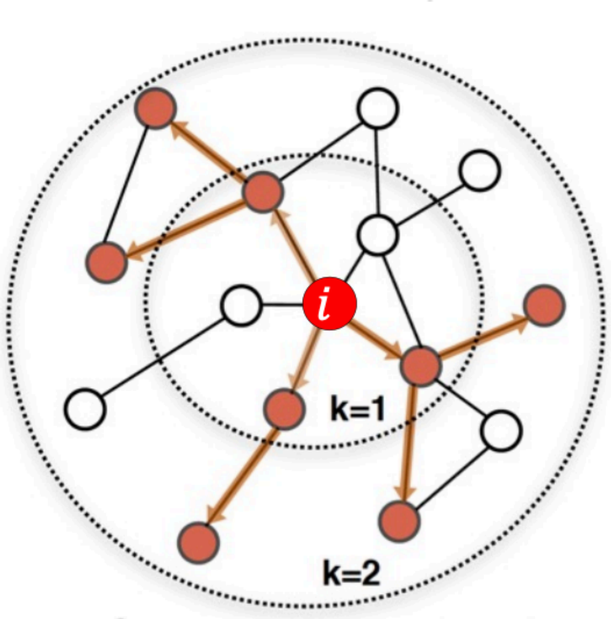
Matrix factorization

- Simplest **node similarity**: Nodes u, v are similar if they are connected by an edge
- This means: $\mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u = A_{u,v}$ which is the (u, v) entry of the graph adjacency matrix A
- Therefore, $\boldsymbol{Z}^T \boldsymbol{Z} = A$

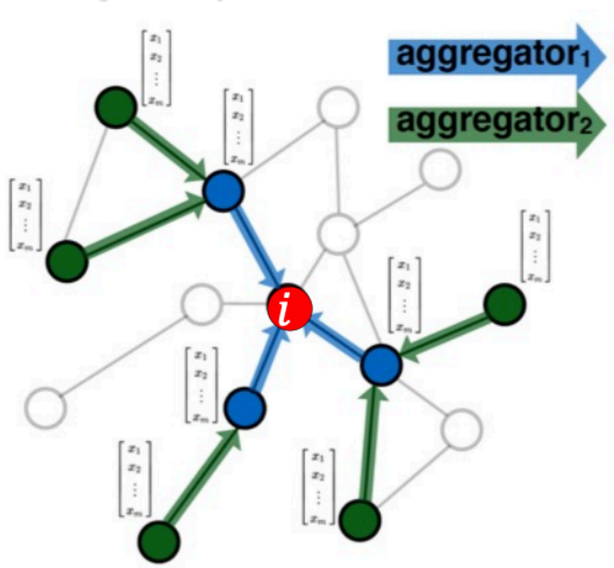


Вопросы

Свертки

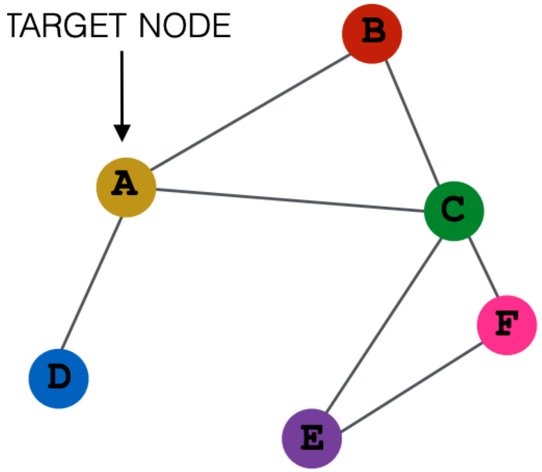


Determine node computation graph

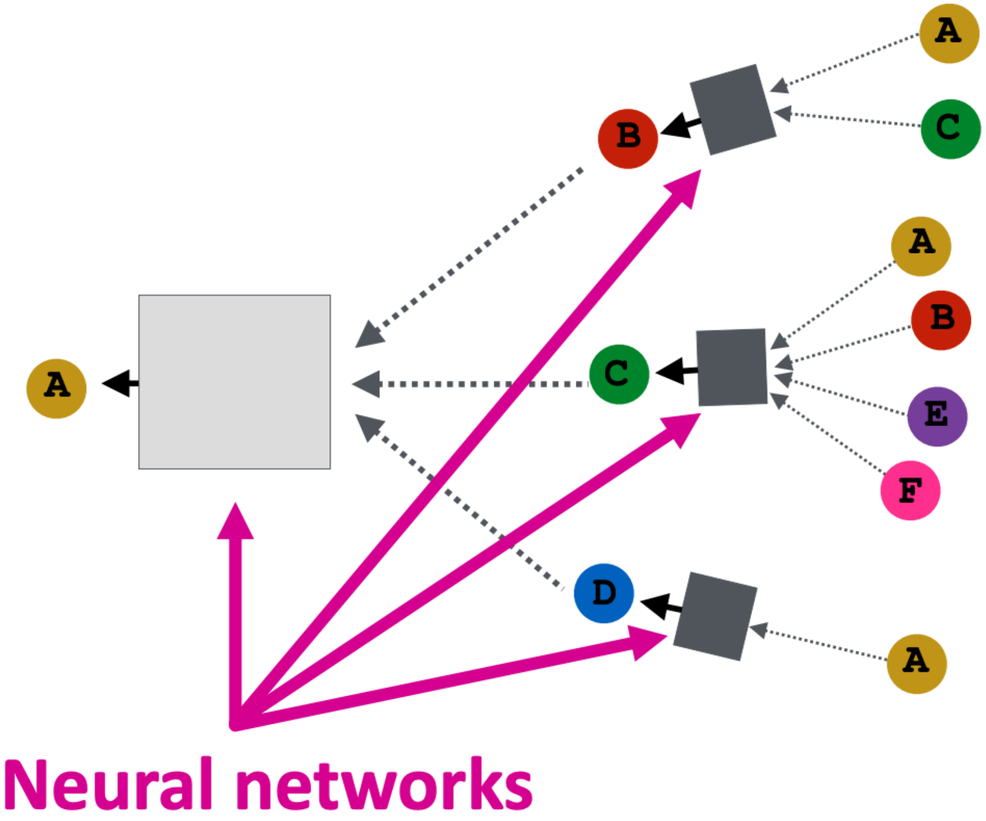


Propagate and transform information

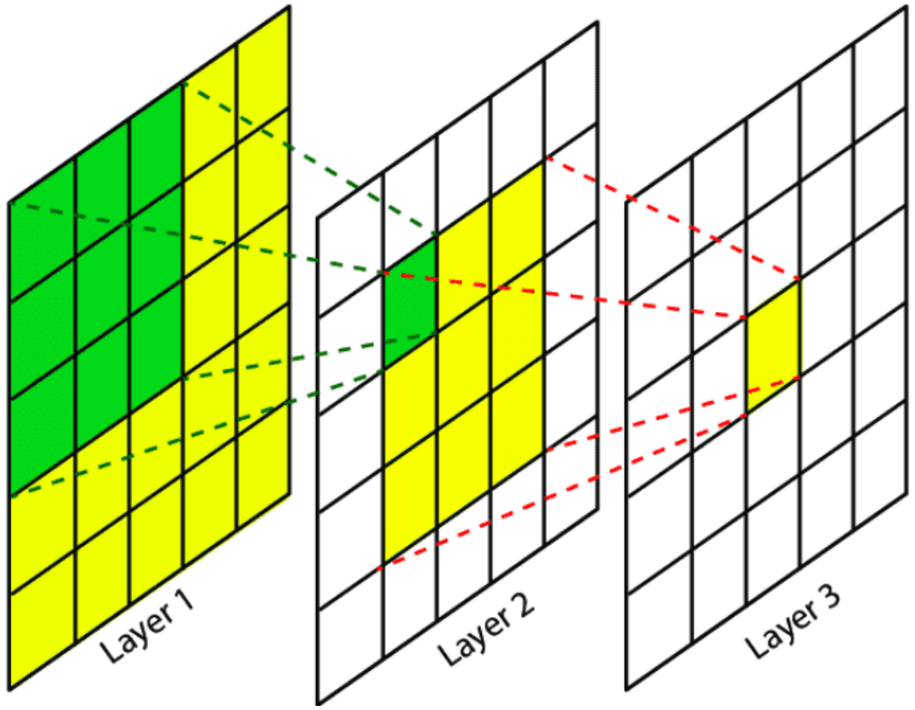
Свертки

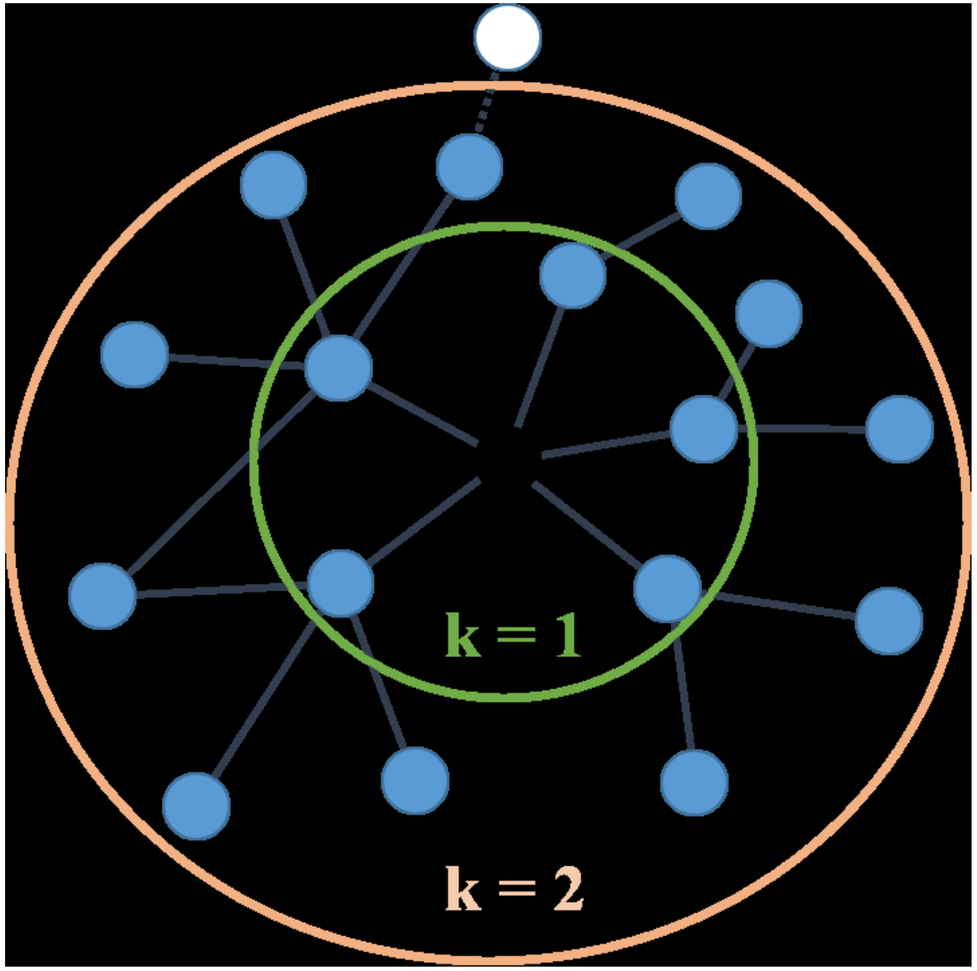


INPUT GRAPH



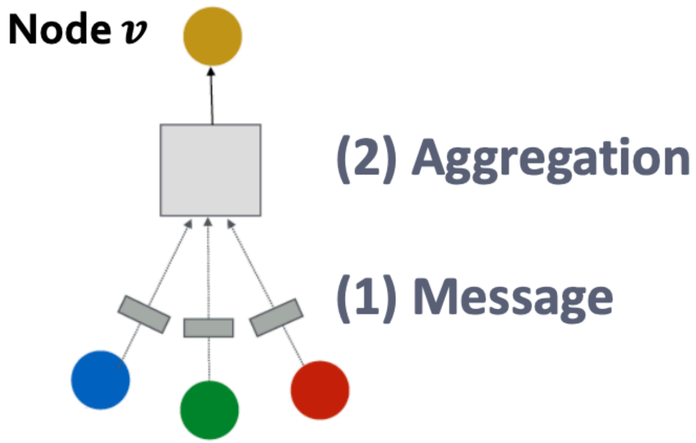
Receptive fields

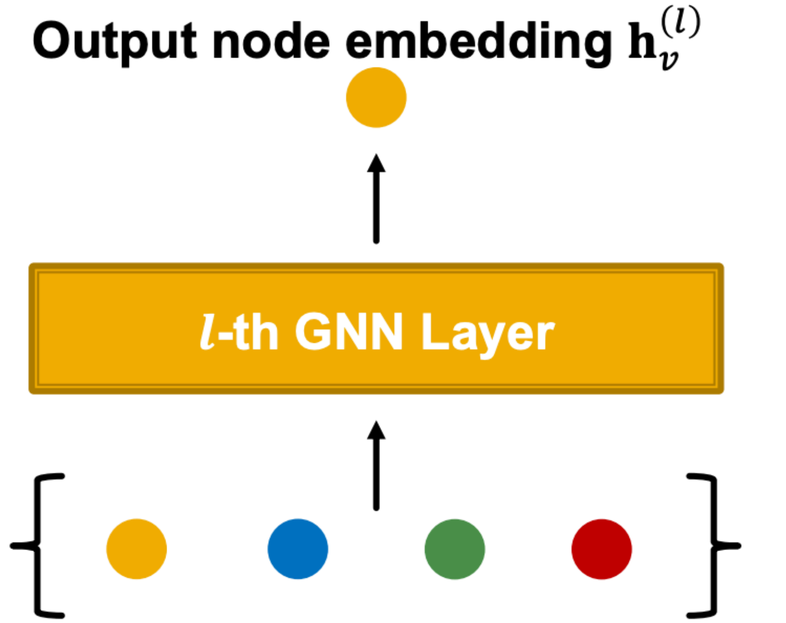




Свертки

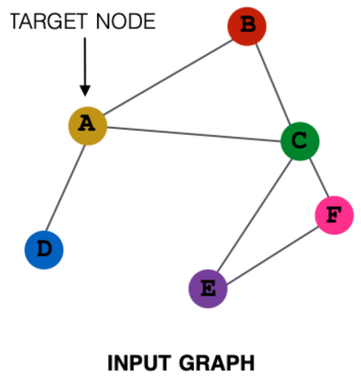
- **Idea of a GNN Layer:**
  - Compress a set of vectors into a single vector
  - **Two-step process:**
    - **(1) Message**
    - **(2) Aggregation**



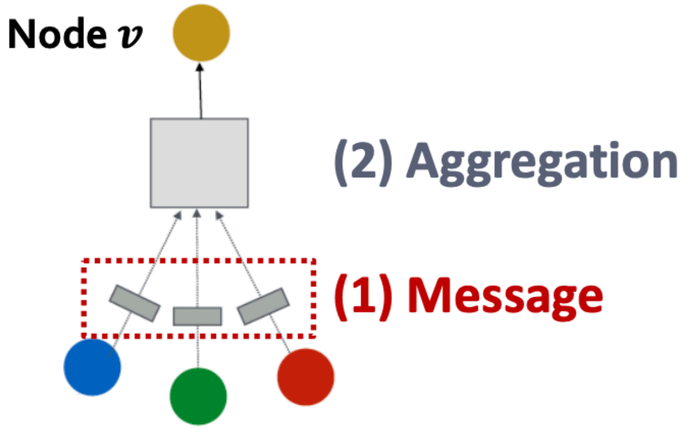


Свертки, общий пайплайн

- **(1) Message computation**
  - **Message function:** $\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$
    - **Intuition:** Each node will create a message, which will be sent to other nodes later
    - **Example:** A Linear layer $\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$
      - Multiply node features with weight matrix $\mathbf{W}^{(l)}$



INPUT GRAPH

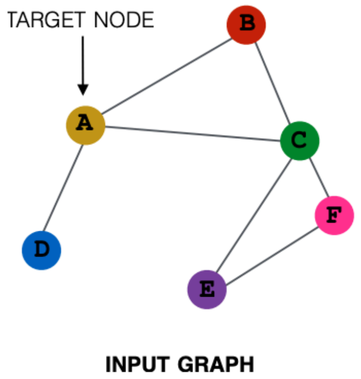


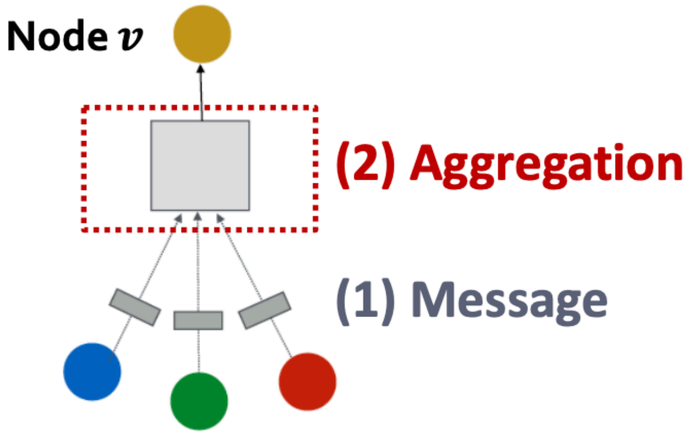
Свертки, общий пайплайн

- **(2) Aggregation**
  - **Intuition:** Node v will aggregate the messages from its neighbors u:

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

  - **Example:** $\mathrm{Sum}(\cdot)$, $\mathrm{Mean}(\cdot)$ or $\mathrm{Max}(\cdot)$ aggregator
    - $\mathbf{h}_v^{(l)} = \mathrm{Sum}(\{\mathbf{m}_u^{(l)}, u \in N(v)\})$





GCN (2017)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding.

... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Mean of v's neighbour's embeddings at step $k-1$.

Node v's embedding at step $k-1$.

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

Semi-Supervised Classification with Graph Convolutional Networks

GraphSage (2017)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding.

... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\underset{u \in \mathcal{N}(v)}{\mathrm{AGG}}(\{h_u^{(k-1)}\}),\ h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Aggregation of v's neighbour's embeddings at step $k-1$ ...

... Node v's embedding at step $k-1$.

... concatenated with ...

Inductive Representation Learning on Large Graphs

GraphSage (2017)

The original GraphSAGE paper considers the following choices for $\underset{u \in \mathcal{N}(v)}{\mathrm{AGG}}(\{h_u^{(k-1)}\})$:

- Mean (similar to the GCN):

$$W_{\text{pool}}^{(k)} \cdot \frac{h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{1 + |\mathcal{N}(v)|}$$

- Dimension-wise Maximum:

$$\max_{u \in \mathcal{N}(v)} \{\sigma(W_{\text{pool}}^{(k)} h_u^{(k-1)} + b)\}$$

- LSTM (after ordering the sequence of neighbours).

Inductive Representation Learning on Large Graphs

GraphSage (2017)

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right)$$

Inductive Representation Learning on Large Graphs

GAT (2018)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding.

... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k-1)} h_u^{(k-1)} + \alpha_{vv}^{(k-1)} h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Weighted mean of v's neighbour's embeddings at step $k-1$.

Node v's embedding at step $k-1$.

where the attention weights $\alpha^{(k)}$ are generated by an attention mechanism $A^{(k)}$, normalized such that the sum over all neighbours of each node v is 1:

$$\alpha_{vu}^{(k)} = \frac{A^{(k)}(h_v^{(k)}, h_u^{(k)})}{\sum_{w \in \mathcal{N}(v)} A^{(k)}(h_v^{(k)}, h_w^{(k)})} \quad \text{for all } (v, u) \in E.$$

Graph Attention Networks

GAT (2018)

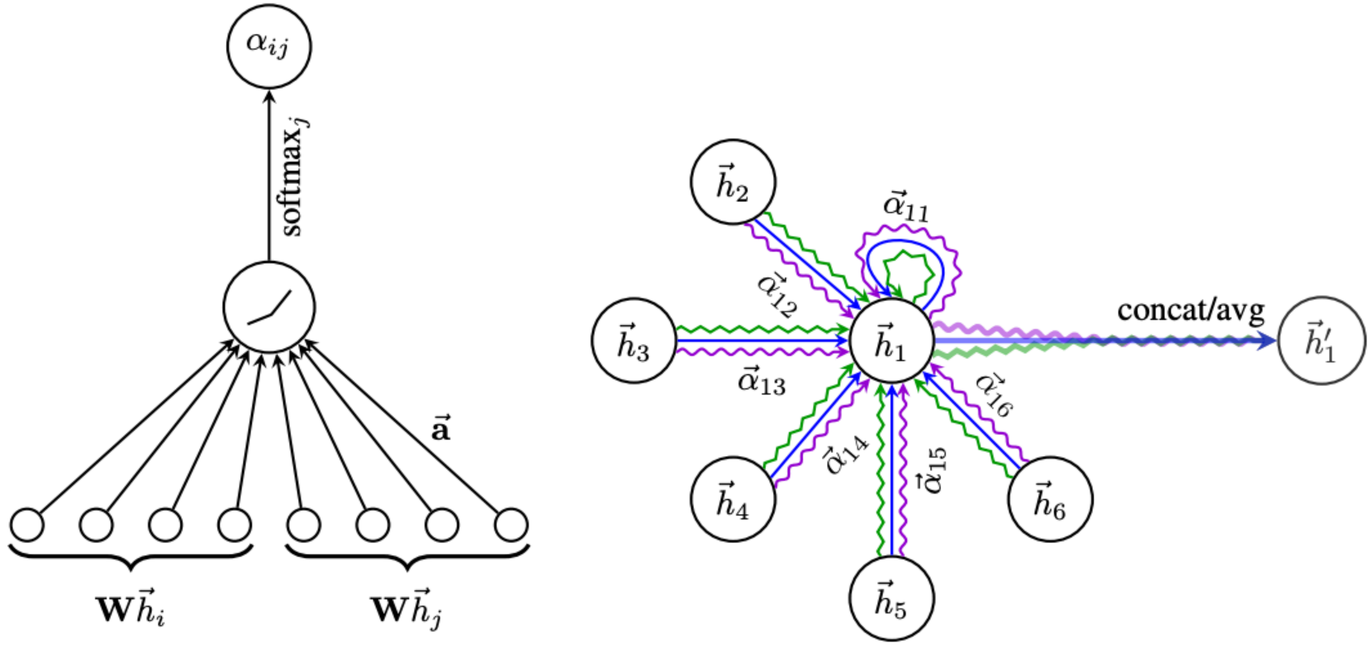


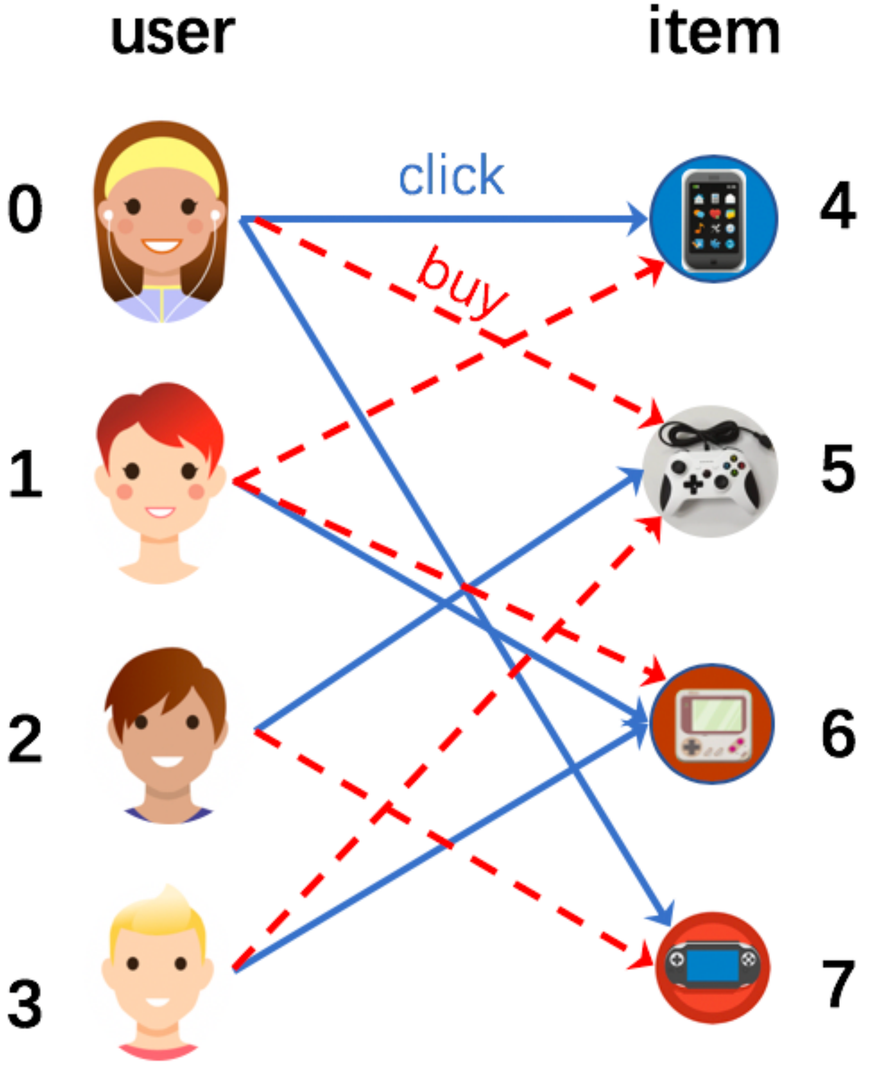
Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Graph Attention Networks

Вопросы

Рекомендации

- двудольный граф user-item
- задача link prediction
- $score = f(u, i)$



NGCF (2019)

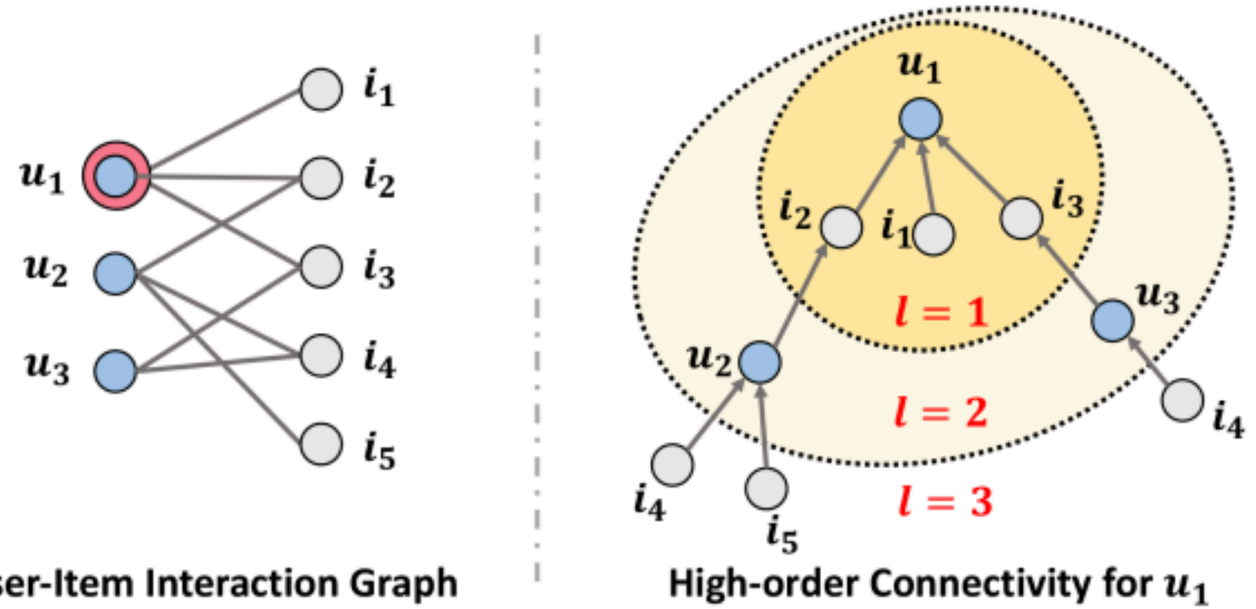


Figure 1: An illustration of the user-item interaction graph and the high-order connectivity. The node u_1 is the target user to provide recommendations for.

Neural Graph Collaborative Filtering

NGCF (2019)

$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right), \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

Following the graph convolutional network [18], we set p_{ui} as the graph Laplacian norm $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$, where $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the first-hop neighbors of user u and item i. From the viewpoint of representation learning, p_{ui} reflects how much the historical item contributes the user preference. From the viewpoint of message passing, p_{ui} can be interpreted as a discount factor, considering the messages being propagated should decay with the path length.

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

Neural Graph Collaborative Filtering

NGCF (2019)

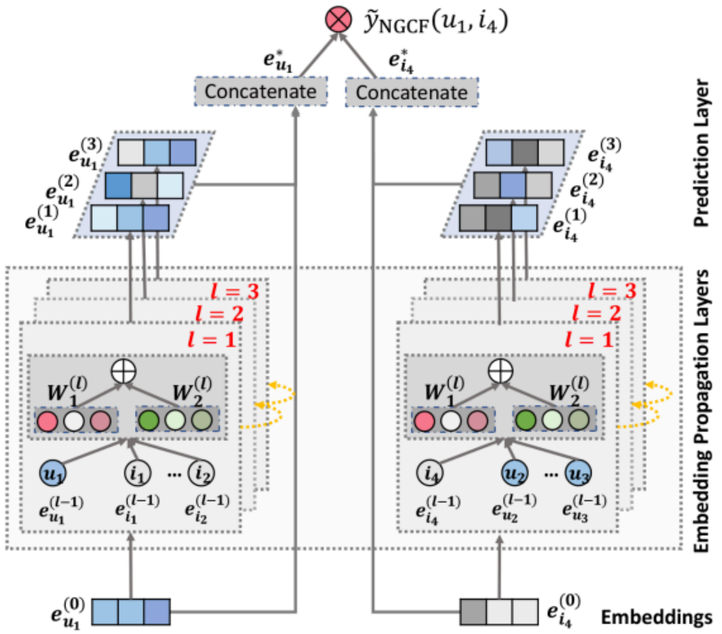


Figure 2: An illustration of NGCF model architecture (the arrowed lines present the flow of information). The representations of user u_1 (left) and item i_4 (right) are refined with multiple embedding propagation layers, whose outputs are concatenated to make the final prediction.

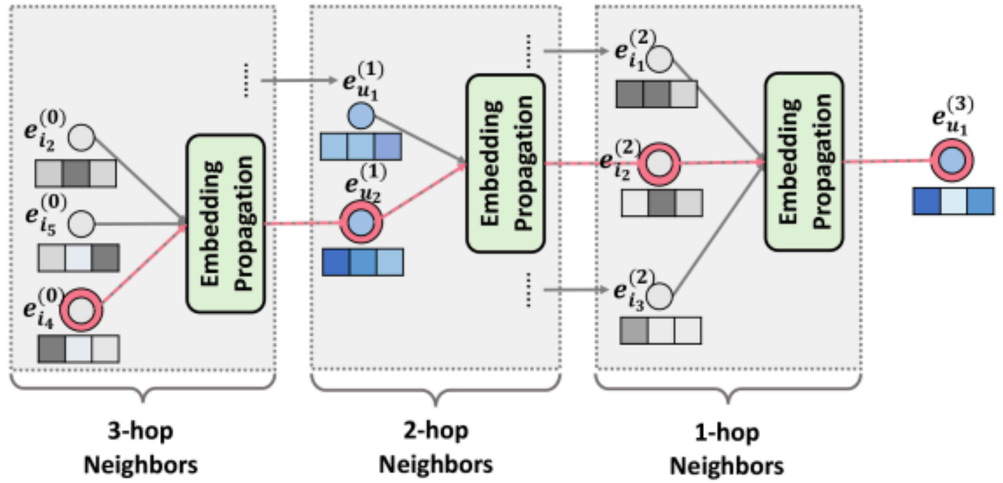
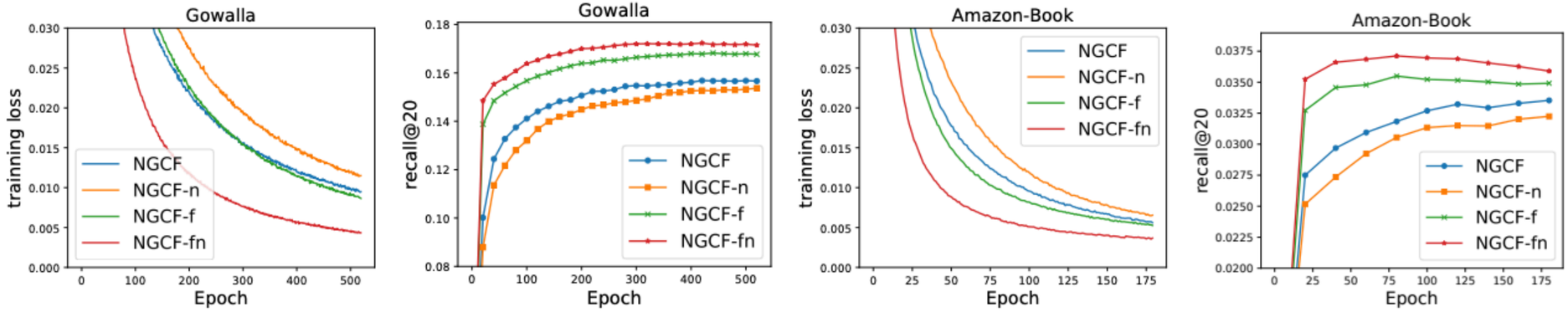


Figure 3: Illustration of third-order embedding propagation for user u_1. Best view in color.

Neural Graph Collaborative Filtering

LightGCN (2020)



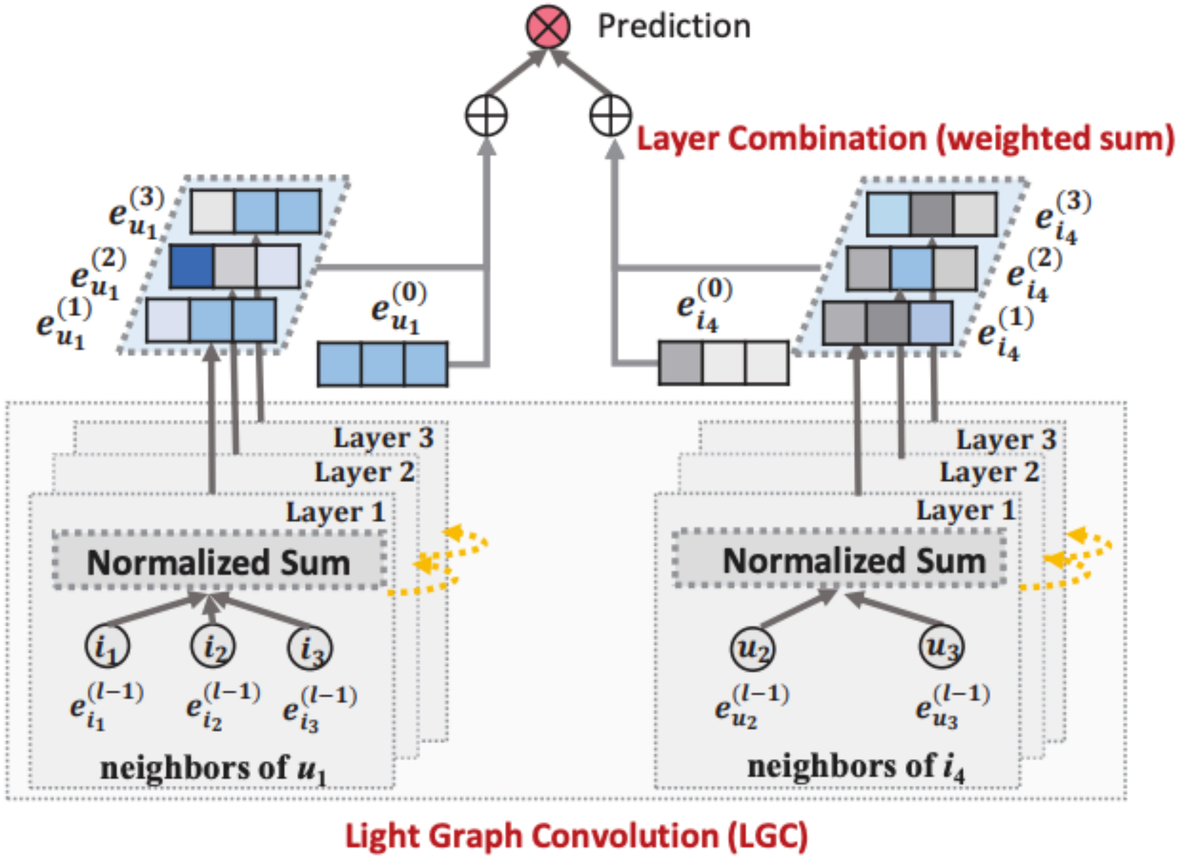
(a) Training loss on Gowalla (b) Testing recall on Gowalla (c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

Figure 1: Training curves (training loss and testing recall) of NGCF and its three simplified variants.

- NGCF-f, which removes the feature transformation matrices $\mathbf{W}_1$ and $\mathbf{W}_2$.
- NGCF-n, which removes the non-linear activation function σ.
- NGCF-fn, which removes both the feature transformation matrices and non-linear activation function.

LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

LightGCN (2020)



LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

LightGCN (2020)

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||\mathbf{E}^{(0)}||^2$$

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)},$$

LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

LightGCN (2020)

Table 3: Performance comparison between NGCF and LightGCN at different layers.

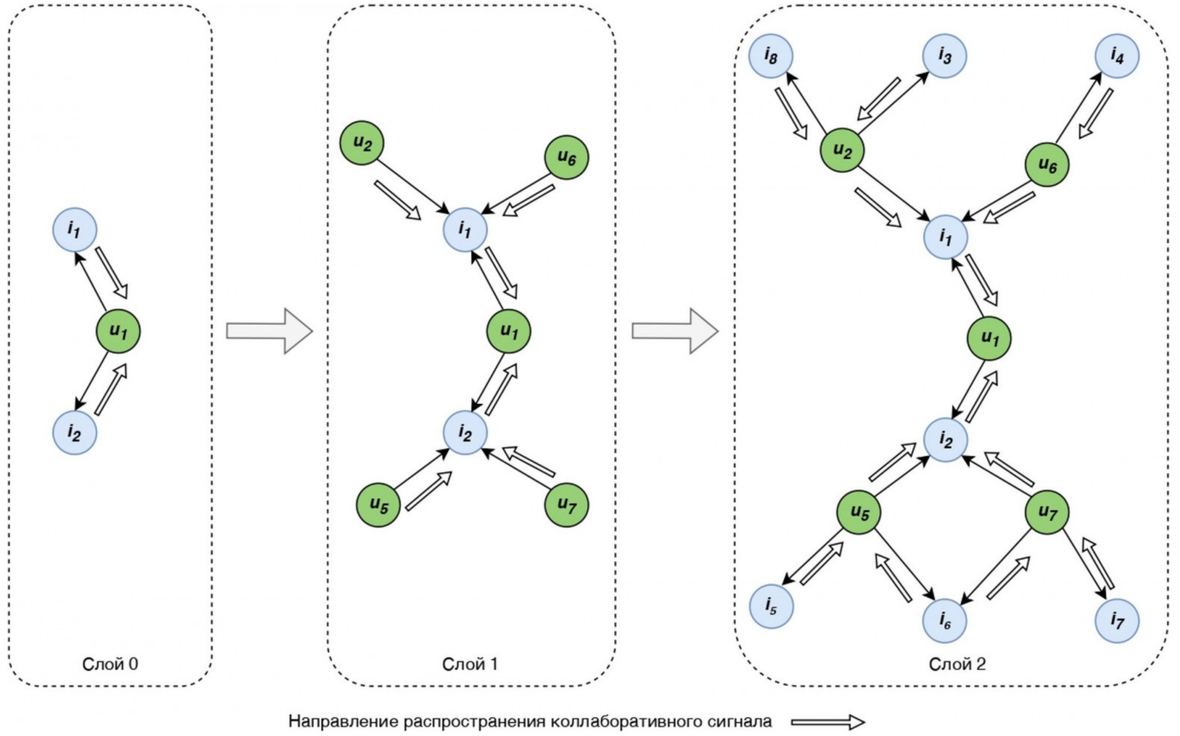
Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	**Method**	**recall**	**ndcg**	**recall**	**ndcg**	**recall**	**ndcg**
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)

LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

LightGCN (2020)

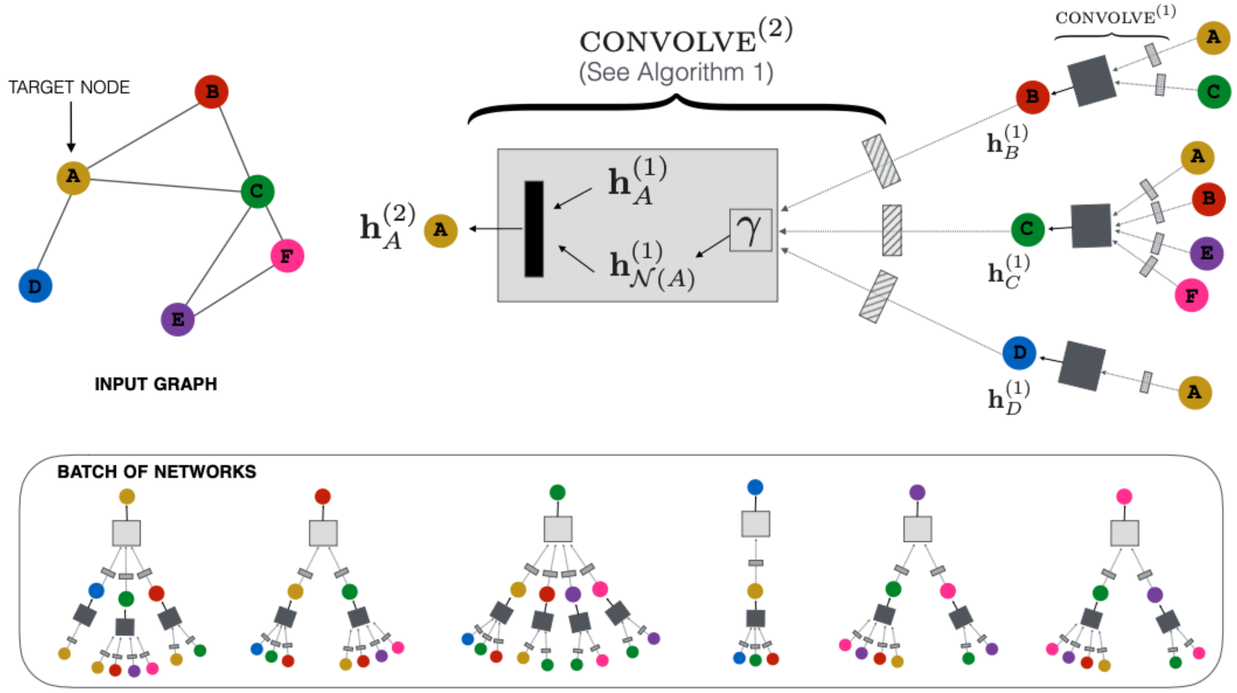
От user-item к item-item



$$e_i^{(k)} = \sum_{u \in N_i} \frac{1}{|N_i|^r |N_u|^{1-r}} e_u^{(k-1)} = \sum_{u \in N_i} \frac{1}{|N_i|^r |N_u|^{1-r}} \left(\sum_{i \in N_u} \frac{1}{|N_u|^r |N_i|^{1-r}} e_i^{(k-2)} \right).$$

https://habr.com/ru/companies/wildberries/articles/826422/

PinSage (2018)



Algorithm 1: CONVOLVE

Input : Current embedding $\mathbf{z}_u$ for node u; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$

Output: New embedding $\mathbf{z}_u^{\text{NEW}}$ for node u

$\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

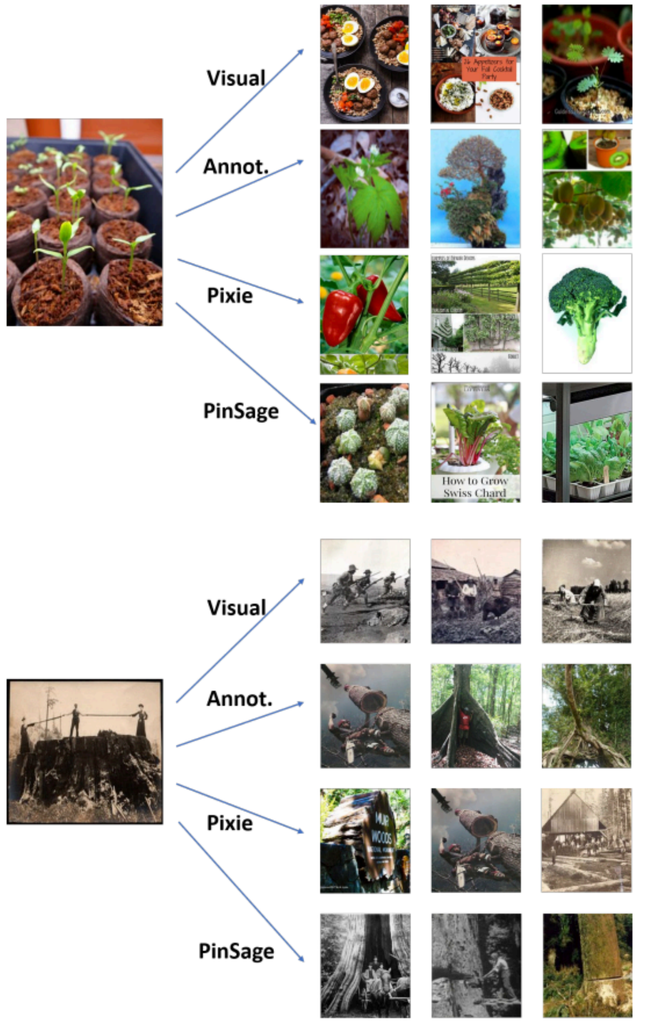
Graph Convolutional Neural Networks for Web-Scale Recommender Systems

PinSage (2018)

user deems to be thematically related. Altogether, the Pinterest graph contains 2 billion pins, 1 billion boards, and over 18 billion edges (*i.e.*, memberships of pins to their corresponding boards).

being 11B.

Features used for learning. Each pin at Pinterest is associated with an image and a set of textual annotations (title, description). To generate feature representation $\mathbf{x}_q$ for each pin q, we concatenate visual embeddings (4,096 dimensions), textual annotation embeddings (256 dimensions), and the log degree of the node/pin in the graph. The visual embeddings are the 6-th fully connected layer of a classification network using the VGG-16 architecture [28]. Textual annotation embeddings are trained using a Word2Vec-based model [23], where the context of an annotation consists of other annotations that are associated with each pin.



Graph Convolutional Neural Networks for Web-Scale Recommender Systems

PinSage (2018)

Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
max-pooling	39%	0.37
mean-pooling	41%	0.51
mean-pooling-xent	29%	0.35
mean-pooling-hard	46%	0.56
PinSage	67%	**0.59**

We also conduct ablation studies and consider several variants of PinSage when evaluating performance:

- **max-pooling** uses the element-wise max as a symmetric aggregation function (i.e., γ = max) without hard negative samples;
- **mean-pooling** uses the element-wise mean as a symmetric aggregation function (i.e., γ = mean);
- **mean-pooling-xent** is the same as mean-pooling but uses the cross-entropy loss introduced in [18].
- **mean-pooling-hard** is the same as mean-pooling, except that it incorporates hard negative samples as detailed in Section 3.3.
- **PinSage** uses all optimizations presented in this paper, including the use of importance pooling in the convolution step.

Graph Convolutional Neural Networks for Web-Scale Recommender Systems

PinSage (2018)

Query

Positive Example

Random Negative

Hard Negative

Graph Convolutional Neural Networks for Web-Scale Recommender Systems

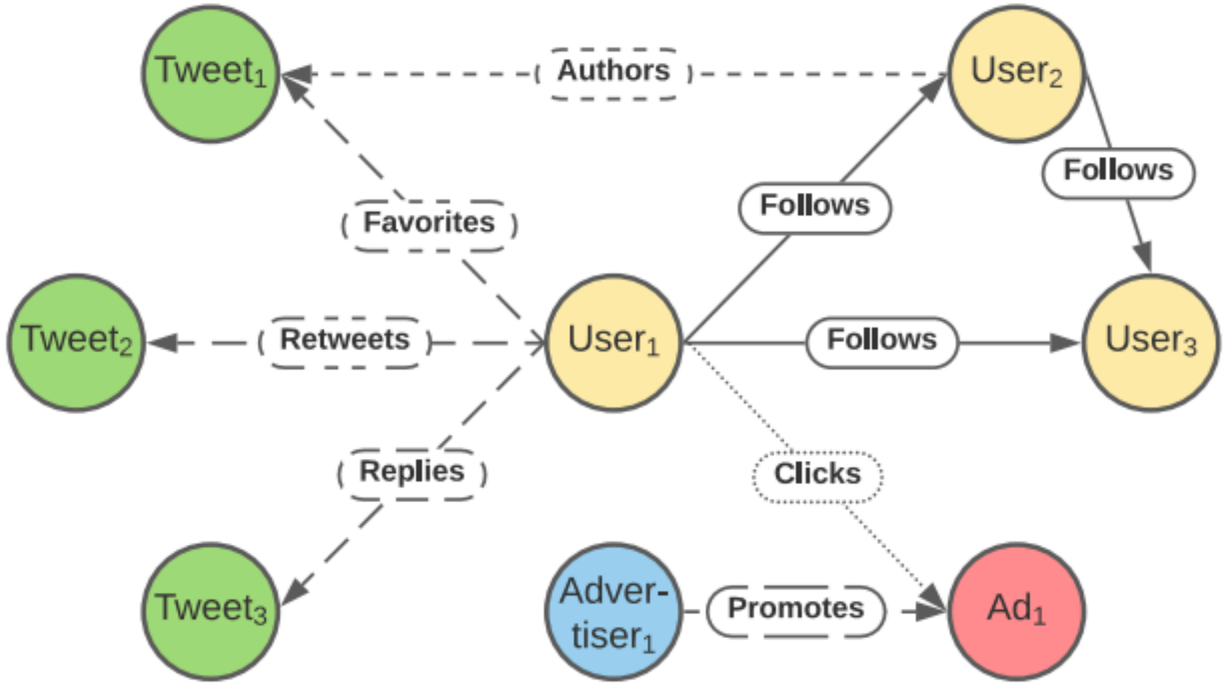
PinSage (2018)

- **Key insight**: It is effective **to make the negative samples *gradually harder* in the process of training**.
- At n-th epoch, we add $n-1$ hard negative items.
  - #(Hard negatives) gradually increases in the process of training.
- The model will gradually learn to make finer-grained predictions.

- **For each user node**, the **hard negatives** are item nodes that are close (but not connected) to the user node in the graph.
- Hard negatives for user $u \in \boldsymbol{U}$ are obtained as follows:
  - Compute random walks from user u.
    - Run random walk with restart from u, obtain visit counts for other items/nodes.
  - Sort items in the descending order of their visit count.
  - Randomly sample items that are ranked high but not too high, e.g., 2000^{th} —5000^{th} .
    - Item nodes that are close but not too close (connected) to the user node.
- The hard negatives for each user are used in addition to the shared negatives.

Graph Convolutional Neural Networks for Web-Scale Recommender Systems

TwHIN (2022)



TwHIN: Embedding the Twitter Heterogeneous Information Network for Personalized Recommendation

TwHIN (2022)

For model simplicity, we apply translating embeddings (TransE) to embed entities and edges in a HIN [1]. For an edge $e = (s, r, t)$, this operation is defined by:

$$f(e) = f(s, r, t) = (\theta_s + \theta_r)^\intercal \theta_t \tag{1}$$

$$\arg\max_{\theta} \sum_{e \in G} \left[\log \sigma(f(e)) + \sum_{e' \in N(e)} \log \sigma(-f(e')) \right]$$

where: $N(s, r, t) = \{(s, r, t') : t' \in V\} \cup \{(s', r, t) : s \in V\}$.

TwHIN: Embedding the Twitter Heterogeneous Information Network for Personalized Recommendation

TwHIN (2022)

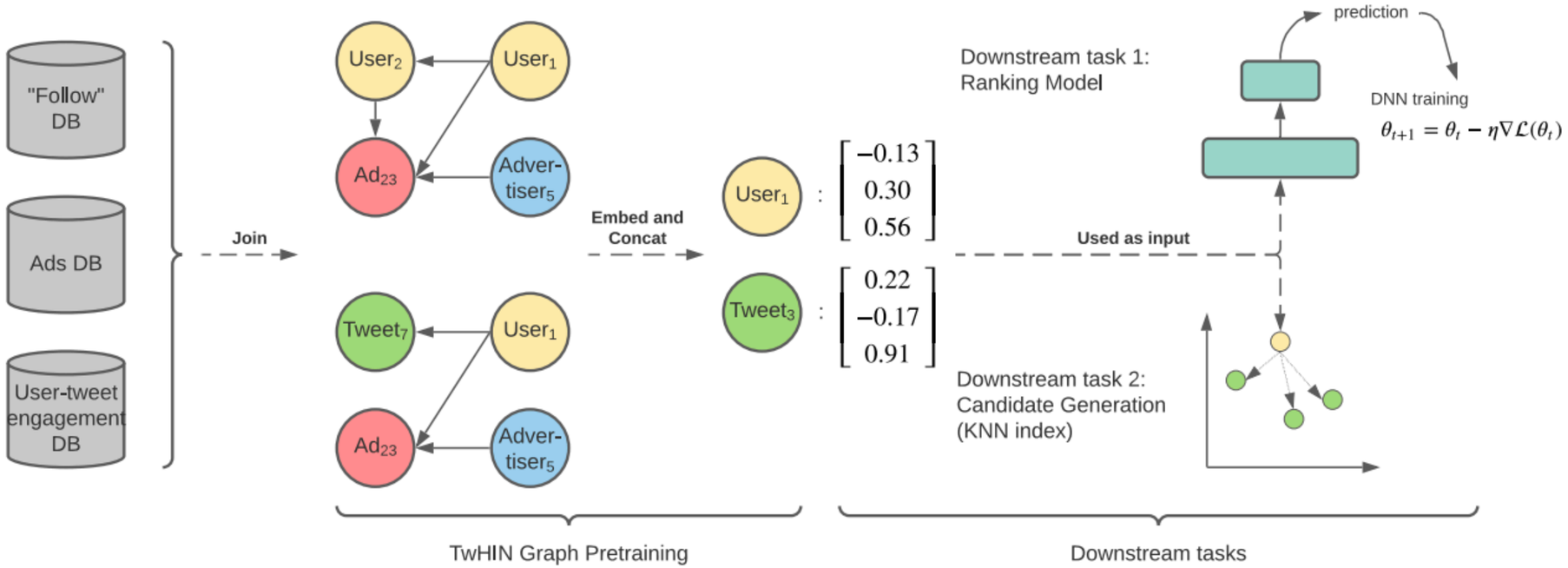
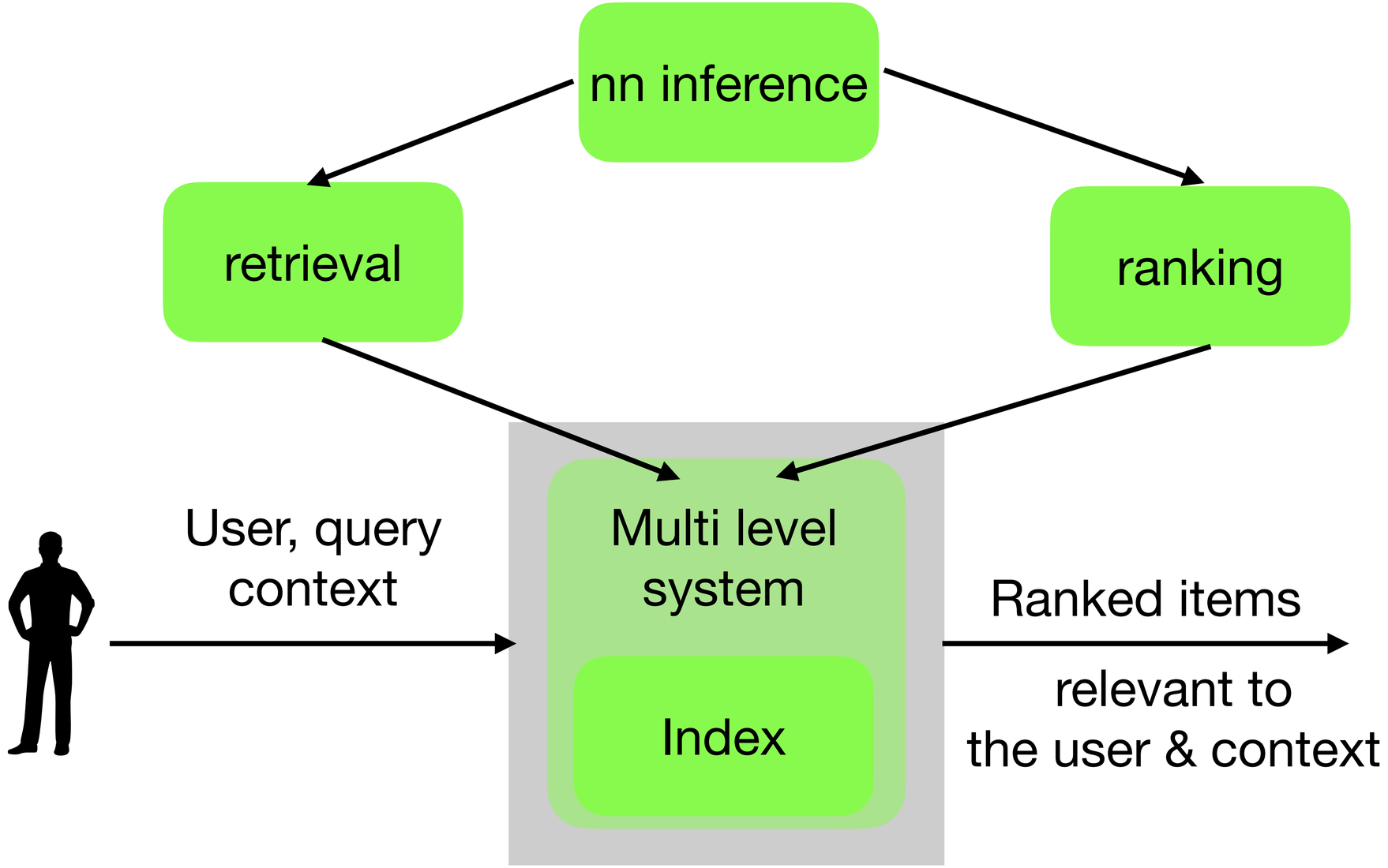


Figure 3: The end-to-end framework aggregates disparate network data to construct TwHIN, joint-embedding is performed and embeddings are consumed in downstream tasks and ML models.

TwHIN: Embedding the Twitter Heterogeneous Information Network for Personalized Recommendation

Наша система




GNN+ATTENTION+
DYNAMIC GRAPHS
ПРОСТО РЕКОМЕНДОВАТЬ
ТО ЖЕ, ЧТО УЖЕ ПОКУПАЛИ

Вопросы