Representation Learning on Networks

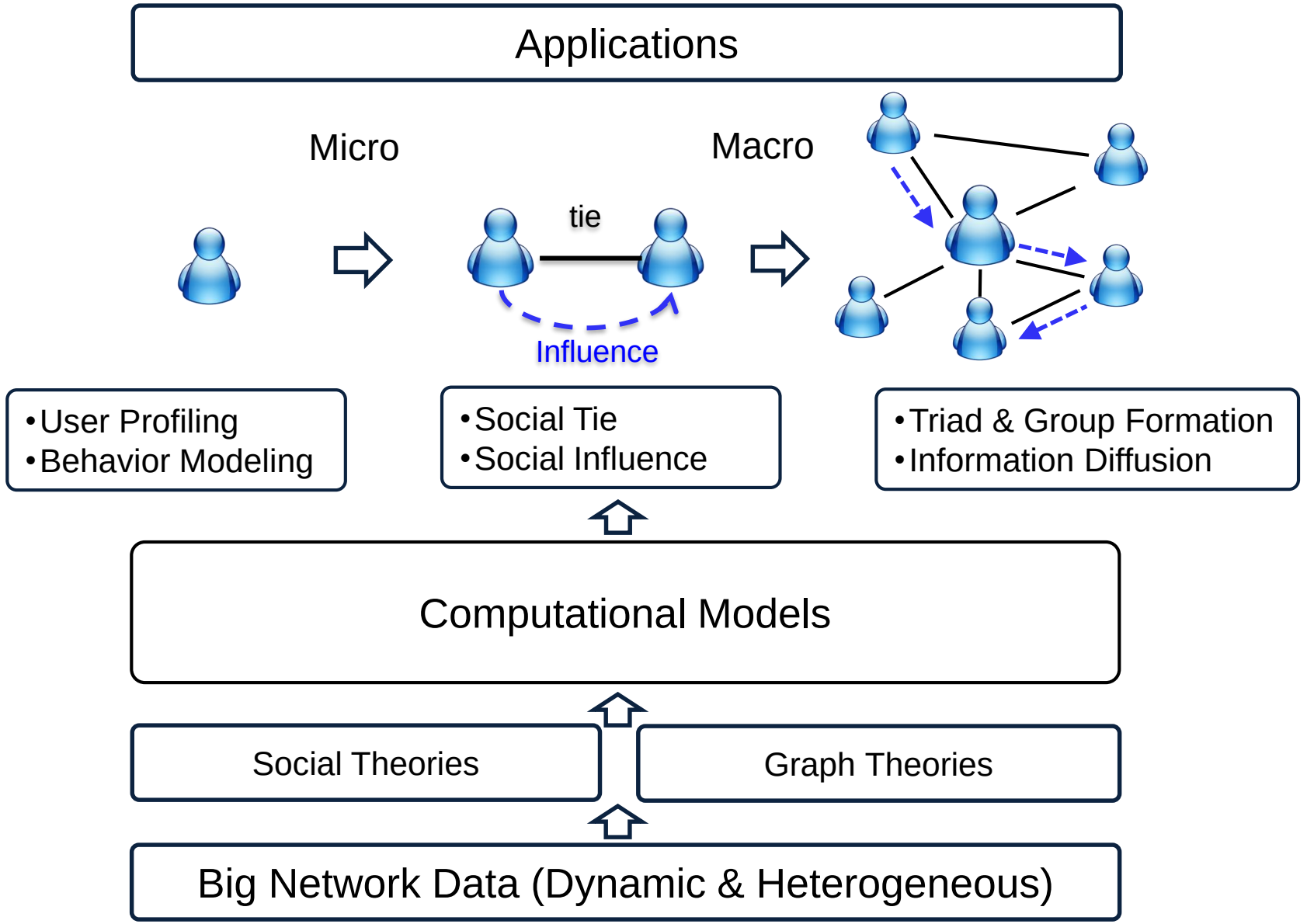
Algorithms, Theory, and Applications

Jie Tang
Tsinghua University

Yuxiao Dong
Microsoft Research, Redmond

Microsoft Research

Social & Information Network Analysis

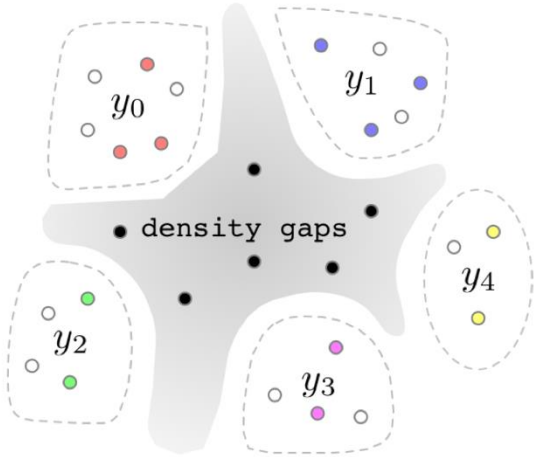


Representation Learning on Networks

- **The first part:**
  - Conventional network analysis
    - Node classification
    - Social tie & link prediction
  - Network embeddings
    - Embedding models
    - Theoretical understanding
    - Large-scale embedding

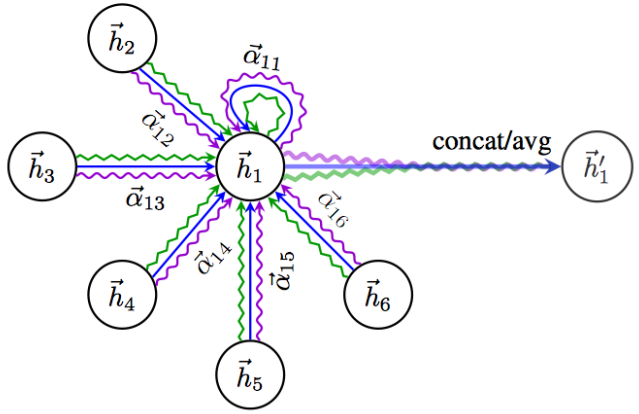
- **The second part:**
  - Graph neural networks
    - Graph convolution
    - Graph GAN
    - Dynamic Representation
    - Heterogeneous Representation
  - Large-scale applications
    - Knowledge graph linking
    - Recommendation in E-commerce
    - Online-to-offline recommendations
    - Social influence in gaming

Recent GCN Research



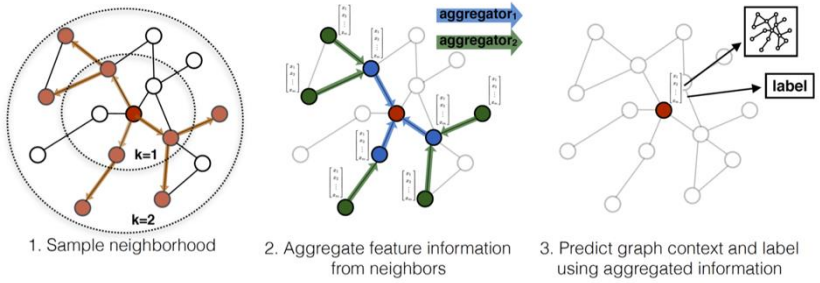
2018 GraphSGAN

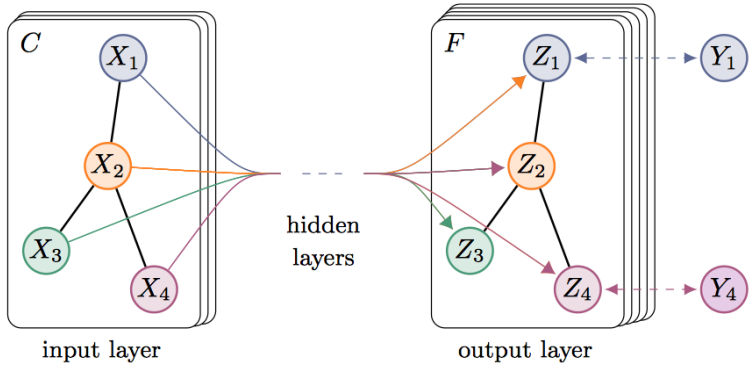
FastGCN 2018



2018 GAT

GraphSAGE 2017

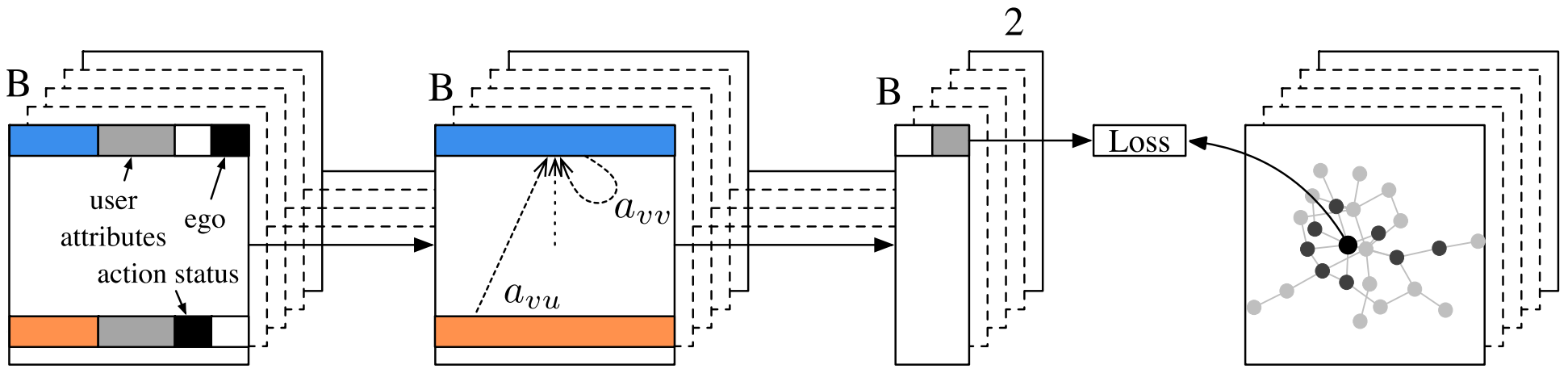




2017 GCN

GCN

- GCNs can be considered as a simplification of the traditional graph spectral methods
- The common strategy is to model a node's neighborhood as the receptive field and then apply the *graph convolution* operation



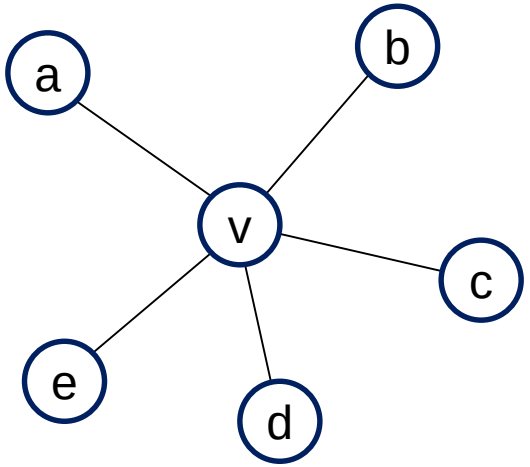
Graph Neural Networks

- Input: an undirected weighted network $G = (V, E)$ with $|V| = n$ & $|E| = \mathrm{m}$
  - Adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{n \times \mathrm{n}}$

$$A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

  - Degree matrix $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$
  - Node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times q}$
- Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n \times k}$
  - Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$

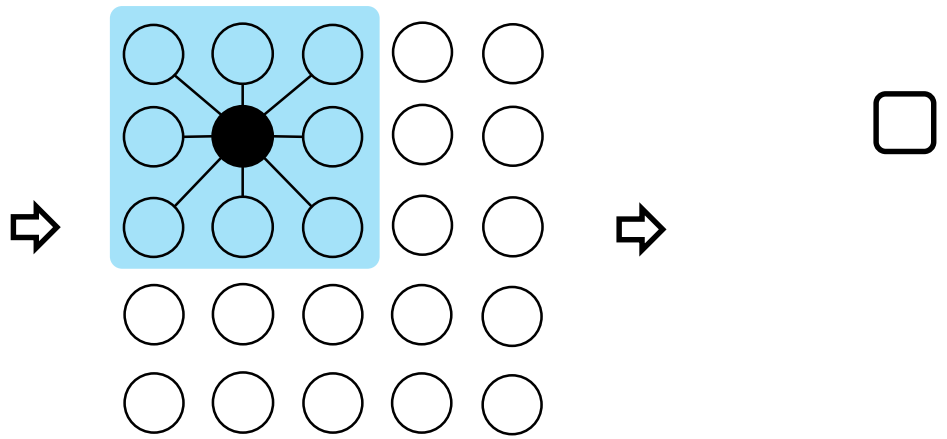
The Core of Graph Neural Networks



$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

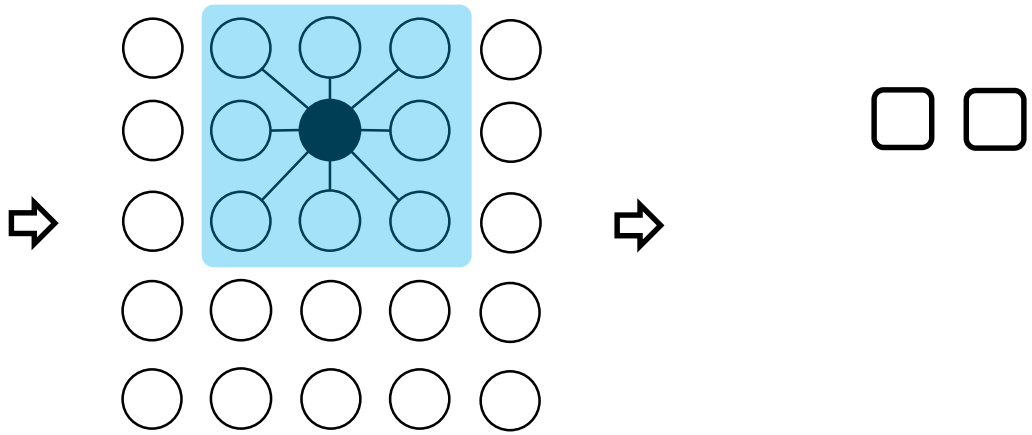
- **Neighborhood Aggregation:**
  - Aggregate neighbor information and pass into a neural network

Convolutional neural network



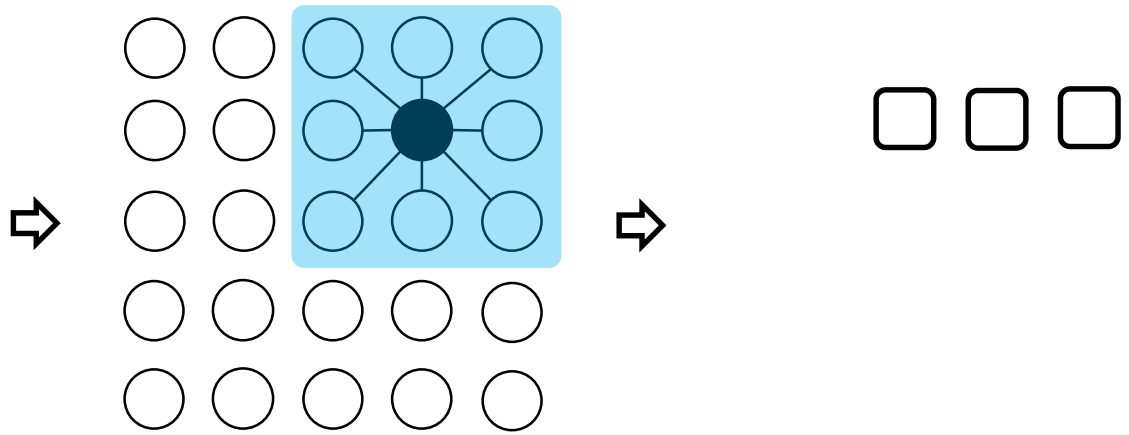
$$\sum wx$$

Convolutional neural network



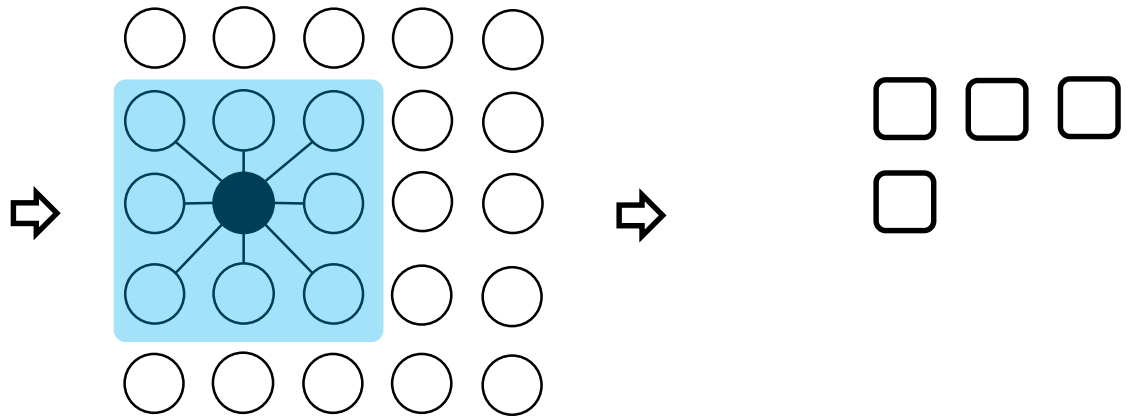
$$\sum wx$$

Convolutional neural network



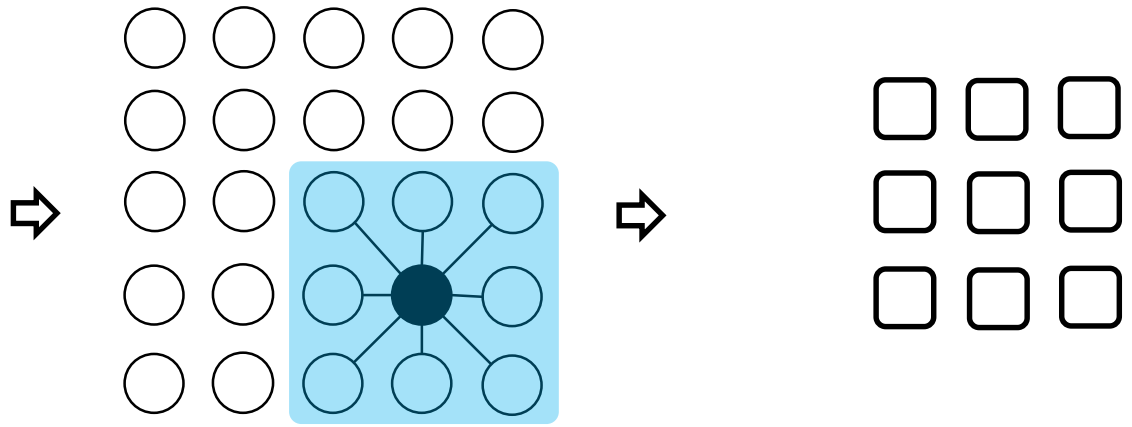
$$\sum wx$$

Convolutional neural network



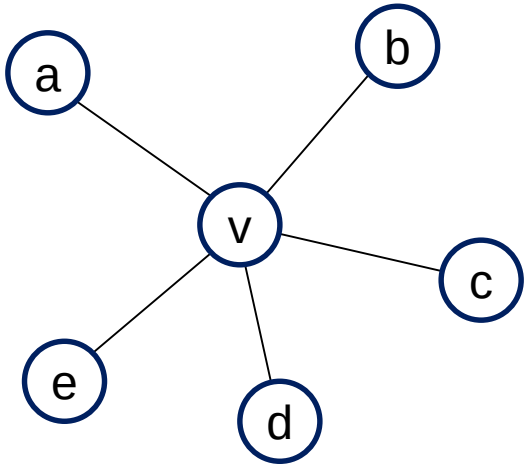
$$\sum wx$$

Convolutional neural network



$$\sum wx$$

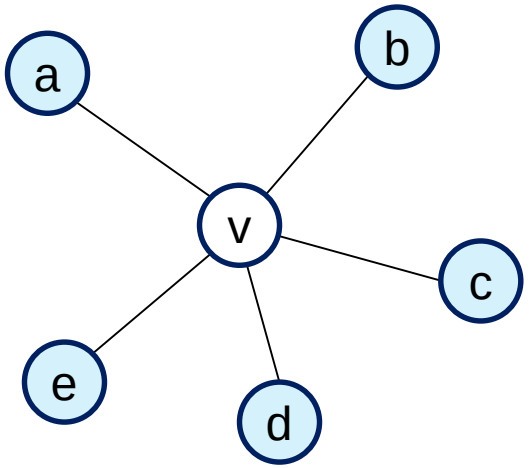
Graph Neural Networks



$$\boldsymbol{h}_a = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

- **Neighborhood Aggregation:**
  - Aggregate neighbor information and pass into a neural network
  - It can be viewed as a center-surround filter in CNN---graph convolutions!

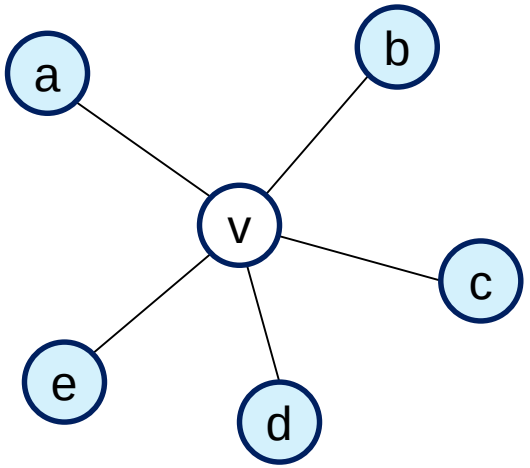
GNN: Graph Convolutional Networks



$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks



parameters in layer k

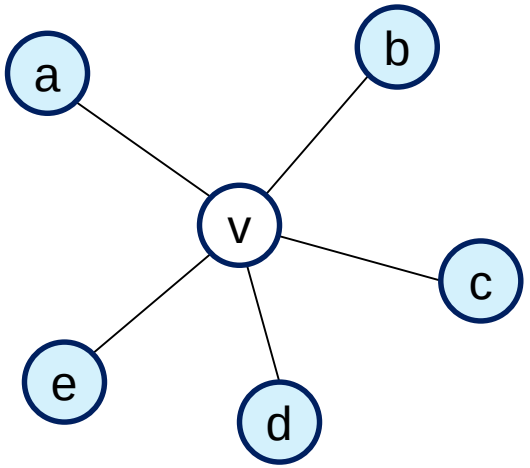
Non-linear activation function (e.g., ReLU)

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

node v's embedding at layer k

the neighbors of node v

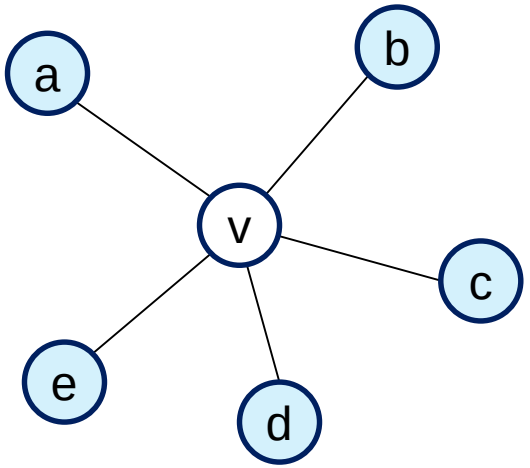
GNN: Graph Convolutional Networks



$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks



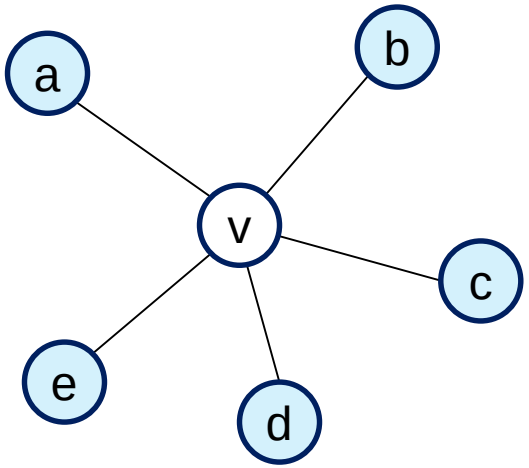
Aggregate from v's neighbors

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v)} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}_k \sum_{v} \frac{h_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

Aggregate from itself

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks

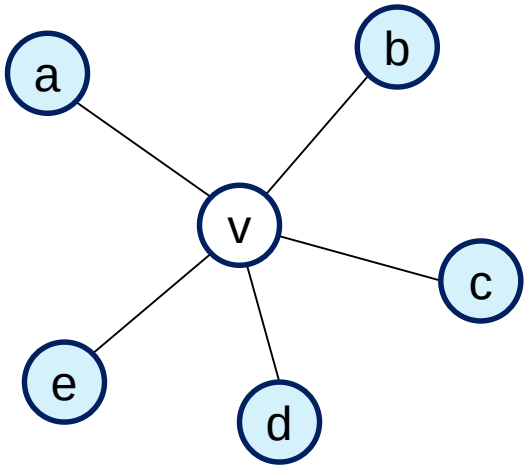


The same parameters for both its neighbors & itself

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}_k \sum_{v} \frac{h_v^{k-1}}{\sqrt{|N(v)||N(v)|}}\right)$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks



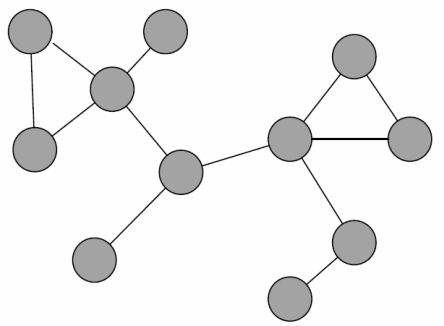
$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}$$

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v)} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}_k \sum_{v} \frac{h_v^{k-1}}{\sqrt{|N(v)||N(v)|}}\right)$$

$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{I}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

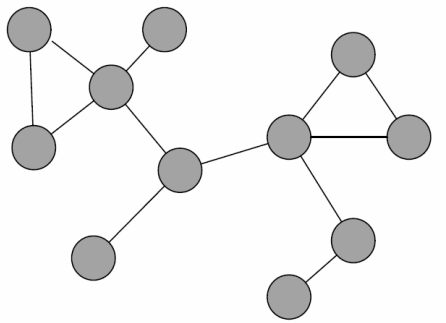
$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

Input

Output

GNN: Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

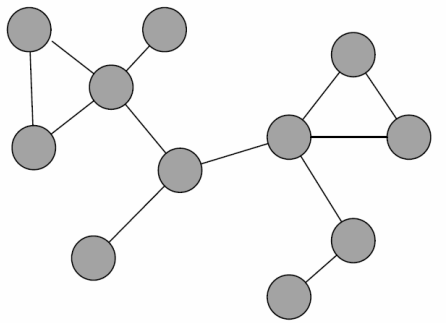
Input

Model training

- The common setting is to have an end to end training framework with a supervised task
- That is, define a loss function over $\boldsymbol{Z}$

Output

GNN: Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

Input

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

Benefits: Parameter sharing for all nodes

- #parameters is subline in |V|
- Enable inductive learning for new nodes

Output

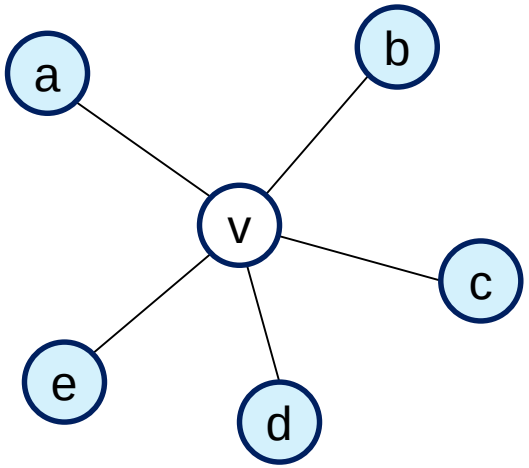
GNN: Graph Convolutional Networks

$$\boldsymbol{H}^{k} = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$

GCN is one way of neighbor aggregations

- **GraphSage**
- Graph Attention
- … …

GraphSage



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

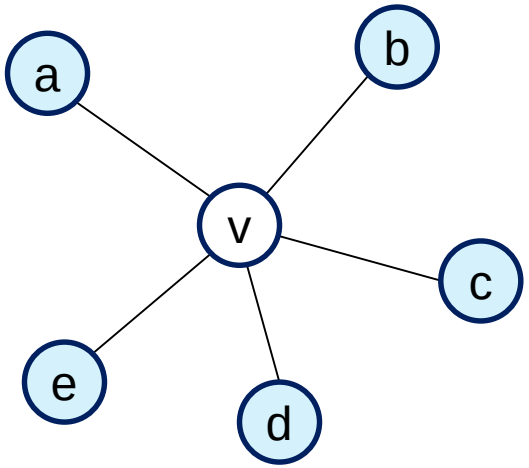
GraphSage

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}_k \cdot \mathrm{AGG}(\{h_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}_k h_v^{k-1}])$$

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning

GraphSage



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GraphSage

Instead of summation, it concatenate neighbor & self embeddings

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}_k \cdot \mathrm{AGG}(\{h_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}_k h_v^{k-1}])$$

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning

GraphSage

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}_k \cdot \mathrm{AGG}(\{h_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}_k h_v^{k-1}])$$

- Mean:

$$\mathrm{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool
  - Transform neighbor vectors and apply symmetric vector function.

element-wise mean/max

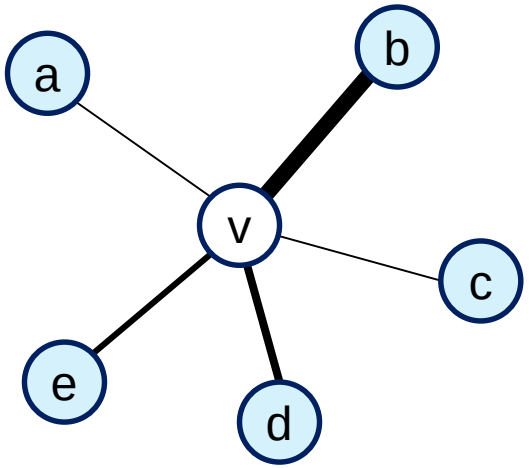
$$\mathrm{AGG} = \gamma(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$$

- LSTM:
  - Apply LSTM to random permutation of neighbors.

$$\mathrm{AGG} = \mathrm{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

Graph Neural Networks



Realistically, neighbors are of different importance to each node

Velickovic et al. Graph Attention Networks. ICLR 2018

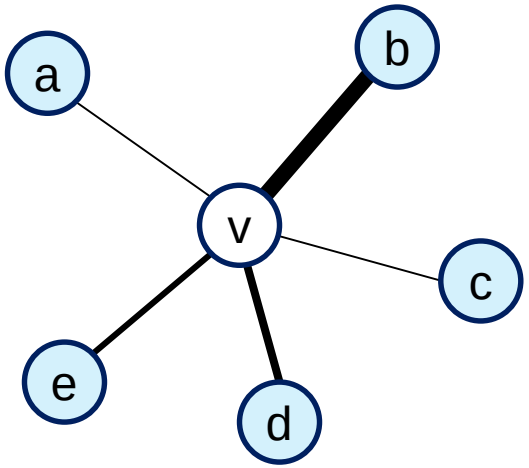
Graph Neural Networks

$$\boldsymbol{H}^{k} = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$

GCN is one way of neighbor aggregations

- GraphSage
- **Graph Attention**
- … …

GNN: Graph Attention



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

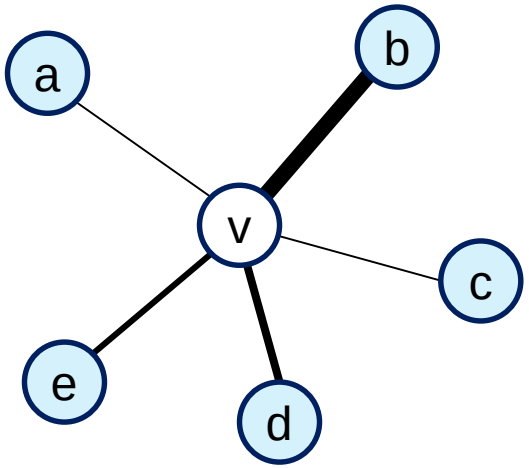
Graph Attention

$$\boldsymbol{h}_v^k = \sigma(\sum_{u \in \mathrm{N}(v) \cup v} \alpha_{v,u} \boldsymbol{W}^k h_u^{k-1})$$

Learned attention weights

Velickovic et al. Graph Attention Networks. ICLR 2018

GNN: Graph Attention

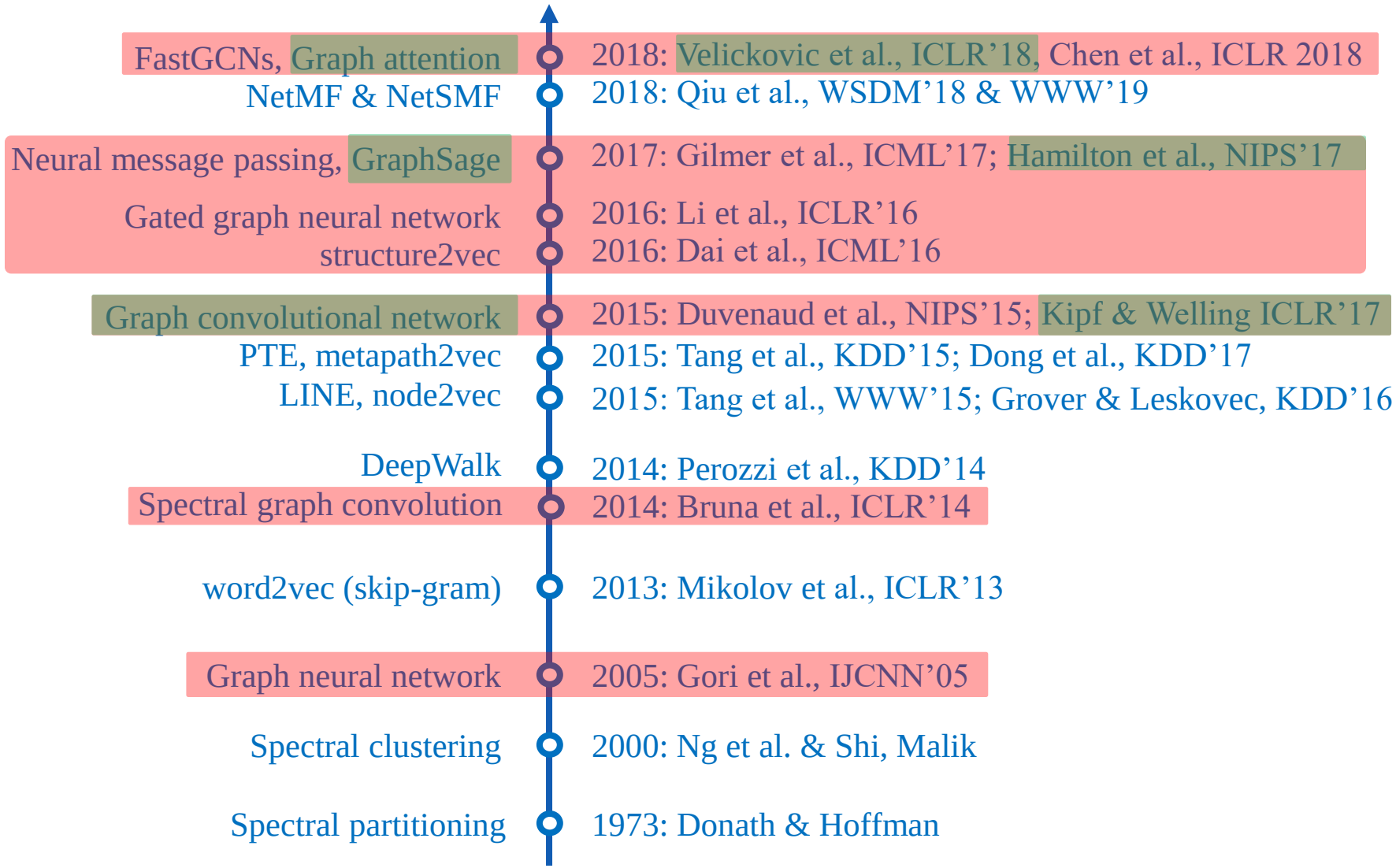


$$\alpha_{v,u} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]\right)\right)}{\sum_{u' \in N(v) \cup \{v\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]\right)\right)}$$

Various ways to define attention!

Velickovic et al. Graph Attention Networks. ICLR 2018

Network Representation Learning / Network Embedding

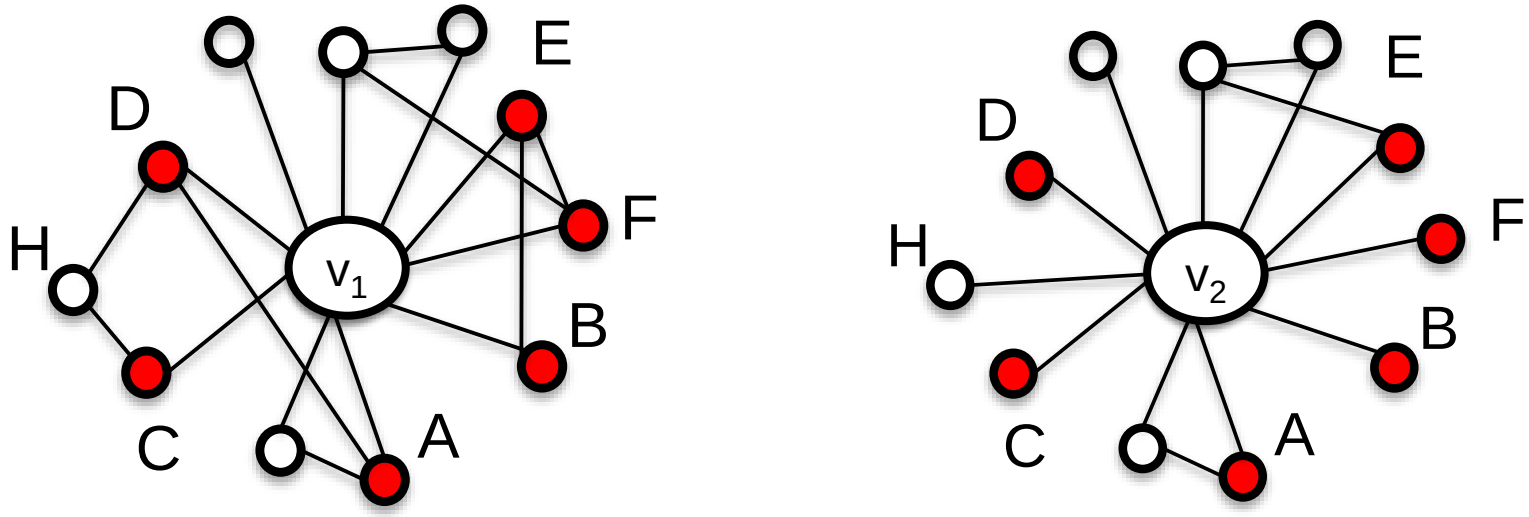


Example Application: Social Influence

- Social influence occurs when one's opinions, emotions, or behaviors are affected by others, intentionally or unintentionally.[1]
  - **Informational social influence**: to accept information from another;
  - **Normative social influence**: to conform to the positive expectations of others.

1. http://en.wikipedia.org/wiki/Social_influence
2. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Structural Social Influence

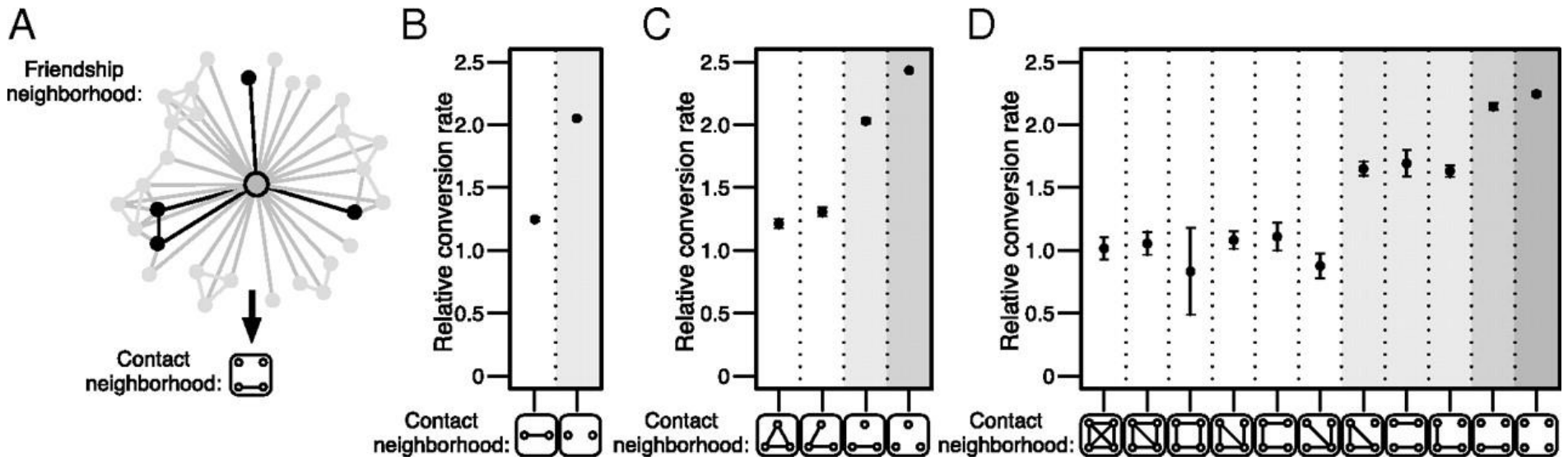


Who are more likely to be "active", v_1 or v_2?

e.g., active = "to attend CAAI AIDL"

Active neighbor Inactive neighbor v User to be influenced

Structural Social Influence



Invite user to join FB by existing users

(J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, PNAS'12)

Previous Solution

- Hand craft features + prediction model. E.g.,

Name	Description
Vertex	Coreness [3].
	Pagerank [30].
	Hub score and authority score [8].
	Eigenvector Centrality [5].
	Clustering Coefficient [46].
	Rarity (reciprocal of ego user's degree) [1].
	Network embedding (DeepWalk [31], 64-dim).
Ego	The number/ratio of active neighbors [2].
	Density of subgnetwork induced by active neighbors [40].
	#Connected components formed by active neighbors [40].

+ Logistic Regression

Network Representation Learning

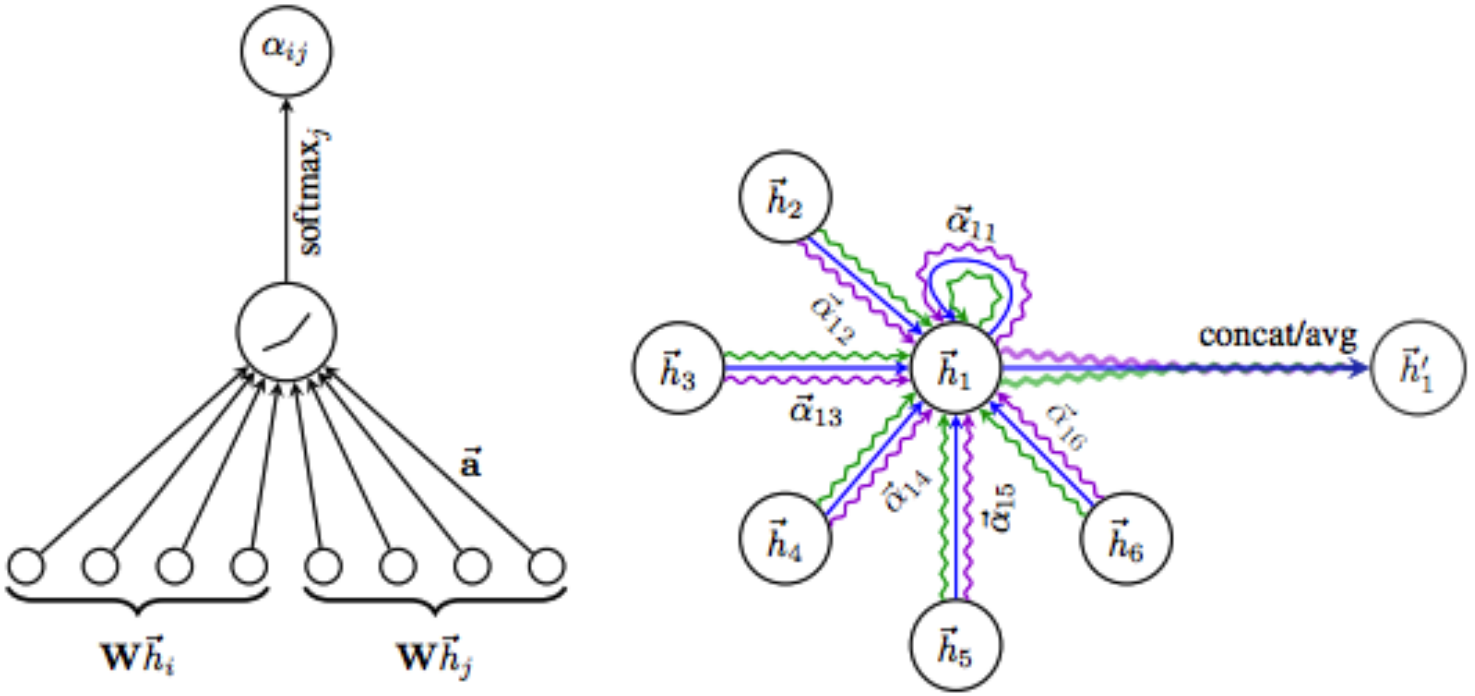
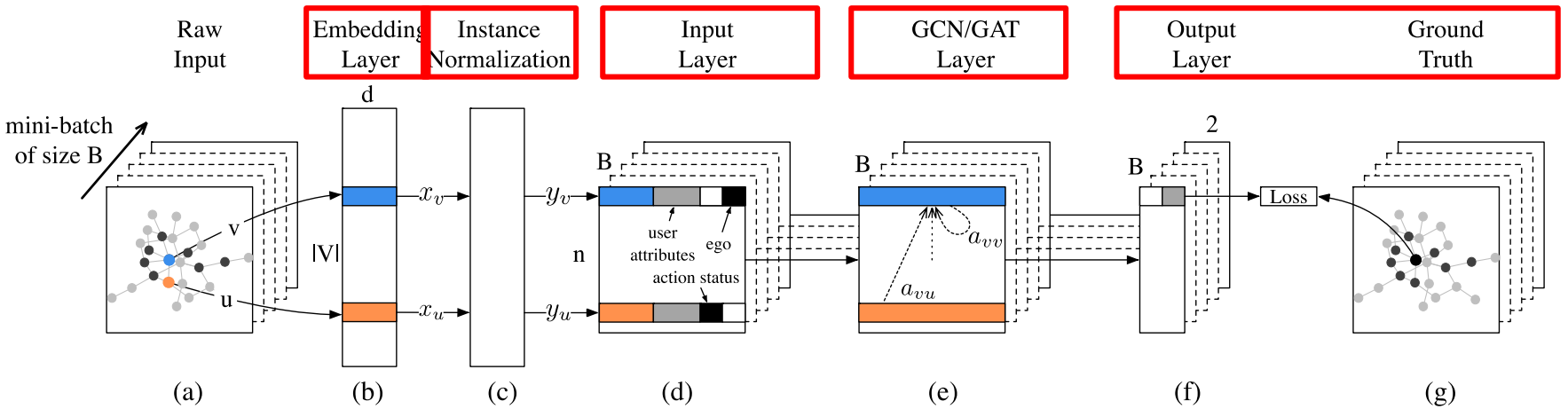


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 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



A pre-trained embedding layer

N layer to avoid overfitting

Construct Batch Normalization layers

Stacked Multi-head GCN/GAT layers

Compare model output with ground truth

1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Experiments --- Datasets

Table 1: Summary of datasets. $|V|$ and $|E|$ indicates the number of vertices and edges in graph $G = (V, E)$, while N is the number of social influence locality instances (observations) as described in Section 2.

	OAG	Digg	Twitter	Weibo
$\|V\|$	953,675	279,630	456,626	1,776,950
$\|E\|$	4,151,463	1,548,126	12,508,413	308,489,739
N	499,848	24,428	499,160	779,164

- OAG (Open Academic Graph): predict paper citations
- Digg (a Reddit-like platform): predict vote-up
- Twitter & Weibo: predict retweet

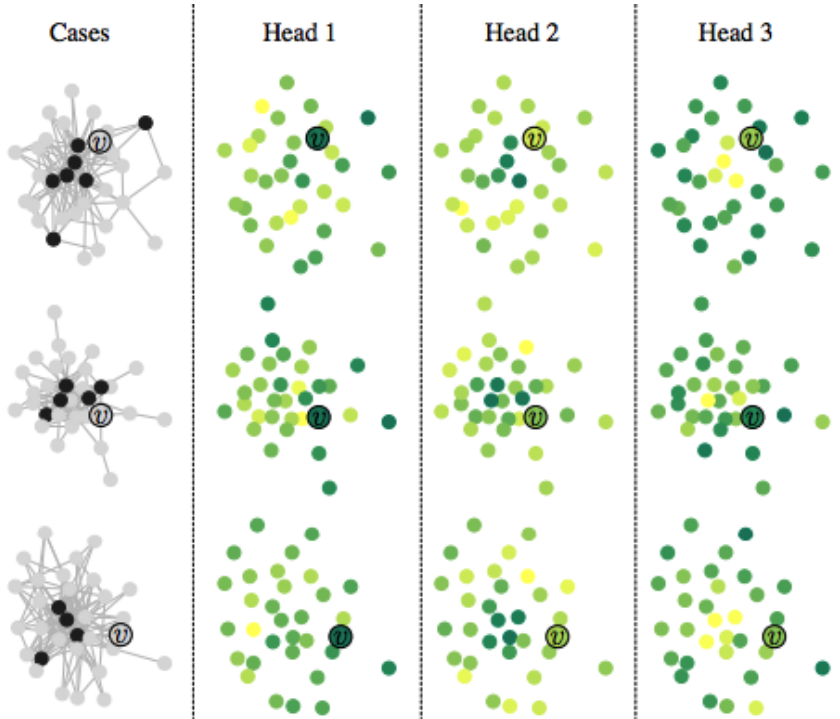
Experiments --- Results

- LR/SVM: Logistic regression/Support vector machine
- PSCN: SOTA deep graph kernel model (ICML'16)
- DeepInf-GAT: Our solution with GAT as bulding blocks

Data	Model	AUC	Prec.	Rec.	F1
OAG	LR	65.55	32.26	69.97	44.16
	SVM	65.48	32.17	**69.82**	44.04
	PSCN	67.70	36.24	60.46	45.32
	DeepInf-GAT	**70.59**	**38.93**	61.29	**47.61**
Digg	LR	84.72	56.78	73.12	63.92
	SVM	86.01	63.42	67.34	65.32
	PSCN	83.96	62.16	67.34	64.65
	DeepInf-GAT	**88.97**	**68.80**	**73.79**	**71.21**
Twitter	LR	78.07	45.86	**69.81**	55.36
	SVM	79.42	49.12	67.31	56.79
	PSCN	79.40	48.43	68.06	56.59
	DeepInf-GAT	**80.01**	**49.39**	67.47	**57.03**
Weibo	LR	77.10	42.34	72.88	53.56
	SVM	77.11	43.27	70.79	53.71
	PSCN	79.54	44.89	73.48	55.73
	DeepInf-GAT	**82.75**	**48.86**	**74.13**	**58.90**

Case Study

- How different graph attention heads highlight different areas of the network.
- Head 1: Focus on the ego-user
- Head 2: Highlight active users
- Head 3: Highlight inactive users



We applied the learned influence
to help real applications
—in very big networks

Big Data Analytics in Game Data

- Online gaming is one of the largest industries on the Internet…
- Facebook
  - 250 million users play games monthly
  - 200 games with more than 1 million active users
  - 12% of the company's revenue is from games
- Tencent (Market Cap: ~150B $)
  - More than 400 million gaming users
  - 50% of Tencent's overall revenue is from games

[1] Zhanpeng Fang, Xinyu Zhou, Jie Tang, Wei Shao, A.C.M. Fong, Longjun Sun, Ying Ding, Ling Zhou, and Jarder Luo. Modeling Paying Behavior in Online Social Networks. **CIKM'14.**

Two games: DNF

- Dungeon & Fighter Online (DNF)
  - A game of melee combat between users and large number of underpowered enemies
  - 400+ million users, the 2nd largest online game in China
  - Users in the game can fight against enemies by individuals or by groups

Two games: QQ Speed

- QQ Speed
  - A racing game that users can partake in competitions to play against other users
  - 200+ million users
  - Users can race against other users by individuals or forma a group to race together
  - Some users may pay…

Task

- Given behavior log data and paying logs of online game users, build a model to help

item recommendation

- Will social influence play an important role in this task?

Online Test

- Test setting
  - Two groups: *test group* and *control group*
  - Send msgs to invite the user to attend a promotion activity.

	Online Test 1 2013.12.27 - 2014.1.3		Online Test 2 2014.1.24 - 2014.1.27		
Group name	test group	control group	test group	test group2	control group
Group size	600K	200K	400K	400K	200K
#Message read	345K	106K	229K	215K	106K
Message read rate	57.50%	53.00%	57.25%	53.75%	53.00%
#Message clicked	47584	7466	23325	20922	6299
Message clicked rate	7.93%	3.73%	5.83%	5.23%	3.15%
Lift_Ratio	196.87%	0%	123.63%	73.40%	0%

Dropout Prediction

- Dropout prediction with influence
  - Problem: Call back of users
  - Game data: *King of Glory (王者荣耀)*

	With Influence			w/o influence		
Top k	#message	#success	ratio	#message	#success	ratio
1	3996761	1953709	**48.88%**	6617662	2732167	41.29%
2	2567279	1272037	**49.55%**	9756330	4116895	42.20%
3	1449256	727728	**50.21%**	10537994	4474236	42.46%
4	767239	389588	**50.78%**	9891868	4255347	43.02%
5	3997251	2024859	**50.66%**	15695743	6589022	41.98%

Dropout Prediction

- Deployed to 9 Games
  - 王者荣耀
  - 欢乐斗地主
  - QQ飞车手游
  - 穿越火线手游龙之谷
  - 英雄联盟
  - 逆战
  - NBA
  - CF
- In 2017, the total #callback: 129,000,000
- In 2018, up to now, the number is 52,890,000

Revisiting GCN

—Neighborhood Aggregation and Network Sampling

GCN can be viewed as multi-layer graph convolution network

- with the following propagation rule

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

where $\hat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\,\widetilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, $H^{(l)}$ is the $D^{(l)}$-dimensional hidden representation in the l layer.

Two Improvements

- Node Ranking-Aware Rescaling
- Network Sampling based GCN

NRGCN: Node Ranking-aware GCN

- Intuition: Complex propagation or diffusion processes over a network are driven by the diversity of nodes' status and influence— as measured by their relative **rankings** in the network.

- NRGCN incorporates the node ranking into neighborhood aggregation step in GCN.

NRGCN: Node Ranking-aware GCN

- Neighborhood aggregation in GCN:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

- Neighborhood aggregation in NRGCN (for node *i*):

$$H_i^{(l+1)} = \sigma_i^{(l)}[(P^{(l)}\hat{A}Q^{(l)}H^{(l)}W^{(l)})_i]$$

  - *P* and *Q*: diagonal matrix, node ranking-aware rescaling
  - σ_i : node ranking-aware encoder
  - $P_{ii}, Q_{ii},$ and σ_i are adaptively adjusted according to the network ranking of node i

NRGCN: Node Ranking-aware GCN

- Network state $\vec{s}_1$
- Current node ranking:

$$r_i = \vec{s}_1^T (HW)_i$$

- Node ranking-aware rescaling:

$$Q_{ii} = \text{sigmoid}[r_i] \qquad P_{ii} = \frac{1}{\sum_j \hat{A}_{ij} Q_{jj}}$$

- Current node ranking:

$$r_i = \vec{s}_2^T (P\hat{A}QHW)_i$$

- Node ranking-aware encoder:

$$\sigma_i(x) = \begin{cases} x & , x > 0 \\ \text{sigmoid}(r_i)(e^x - 1) & , x \leq 0 \end{cases}$$

NRGCN: Node Ranking-aware GCN

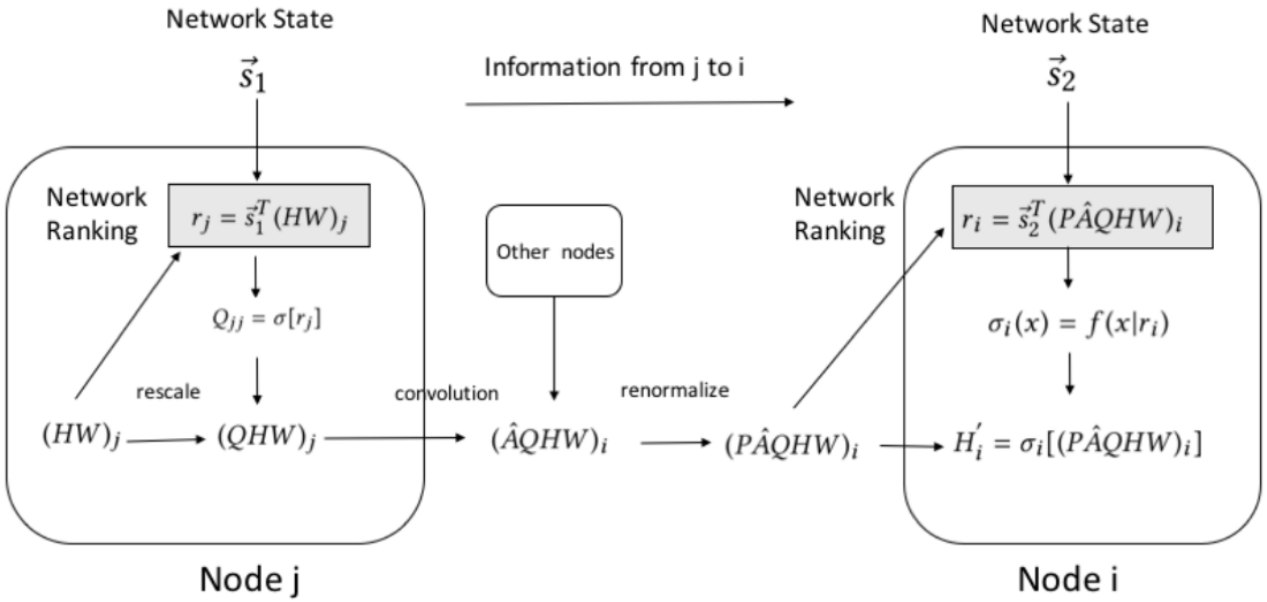


Figure 1: The NRGCN model. The signal propagation from source j to the target neighbor i involves j's rescaling function according to its node ranking, i's convolutional operation which sums over the signals in its neighborhood and again rescales (renormalizes) the resulting signal, and finally i choosing an encoder according to its current network ranking.

NRGCN: Model Enhancements

- NRGCN layer:

$$H_i^{'} = \sigma_i[(P\hat{A}QHW)_i]$$

- Multi-head Propagation mechanism:

$$H_i^{'} = \Big\Vert_{k=1}^{K} \sigma_i^{(k)}[(P^{(k)}\hat{A}Q^{(k)}HW^{(k)})_i]$$

- Multi-hop variants:

$$H_i^{'} = \sigma_i[(P\hat{A}^k QHW)_i]$$

Generalization to Graph Attention

- When *P* normalizes signals in different range, the model can function as node, edge, and path attention mechanisms

(Node attention)

$$\begin{cases} S = \hat{A}Q \\ \vec{p}_i = P_{ii} = \dfrac{1}{\vec{q}^T \vec{1}} \end{cases}$$

(Edge attention)

$$\begin{cases} S = \hat{A}Q \\ \vec{p} = \dfrac{1}{S\vec{1}} \end{cases}$$

(K-hop edge attention)

$$\begin{cases} S = \hat{A}^k Q \\ \vec{p} = \dfrac{1}{S\vec{1}} \end{cases}$$

(Path attention)

$$\begin{cases} S = \hat{A}Q_k \hat{A}Q_{k-1} \ldots \hat{A}Q_1 \\ \vec{p} = \dfrac{1}{S\vec{1}} \end{cases}$$

GAT can be considered as a special case of NRGCN

NSGCN: Network Sampling GCN

- Observation: structural dependency and information redundancy in graph-structured data.
- Sampling can help to explore the network information!!!

$NSGCN_{half}$: a test model

- Sample half of the nodes as set C

$$\begin{cases} Pr(X_i' = X_i) = 0.5, & i \in C. \\ Pr(X_i' = \vec{0}) = 0.5, & i \notin C. \end{cases}$$

- Recover the feature matrix by propagation

$$\widetilde{X} = \frac{1}{k} \sum_{i}^{k} \hat{A}^i X'$$

- $NSGCN_{half}$ (GCN model with partial nodes' features) can generate very close performance to the ordinary GCN.

$NSGCN_{dp}$: Exponential ensemble

- Different sampled subnetworks may provide different views of the original network.
- Dropout-like ensemble model $NSGCN_{dp}$:
  - Exponential ensemble of $NSGCN_{half}$
  - Sample half of the nodes and do the $NSGCN_{half}$ training process at each training epoch
  - Inference with the entire feature matrix and a discount factor 0.5
- Training Loss

$$Loss_{c1} = -\sum_{i \in V^L} \sum_{l}^{|\mathcal{Y}|} Y_{i,l} \ln Z_{i,l}$$

- Inference

$$\hat{Z} = GCN\left(\frac{0.5}{k} \sum_{i}^{k} \hat{A}^i X\right)$$

NSGCN$_{dm}$: Disagreement Minimization

- Minimize the disagreement of two complementary NSGCN$_{dp}$s (on C and V-C at each training epoch)
- The objective to minimize the disagreement at a certain epoch can be obtained by Jensen-Shannon divergence:

$$Loss_{JS} = \frac{1}{2}\sum_{i\in V}\sum_{l}^{|\mathcal{Y}|}\left(Z'_{i,l}\ln\frac{Z'_{i,l}}{Z''_{i,l}} + Z''_{i,l}\ln\frac{Z''_{i,l}}{Z'_{i,l}}\right)$$

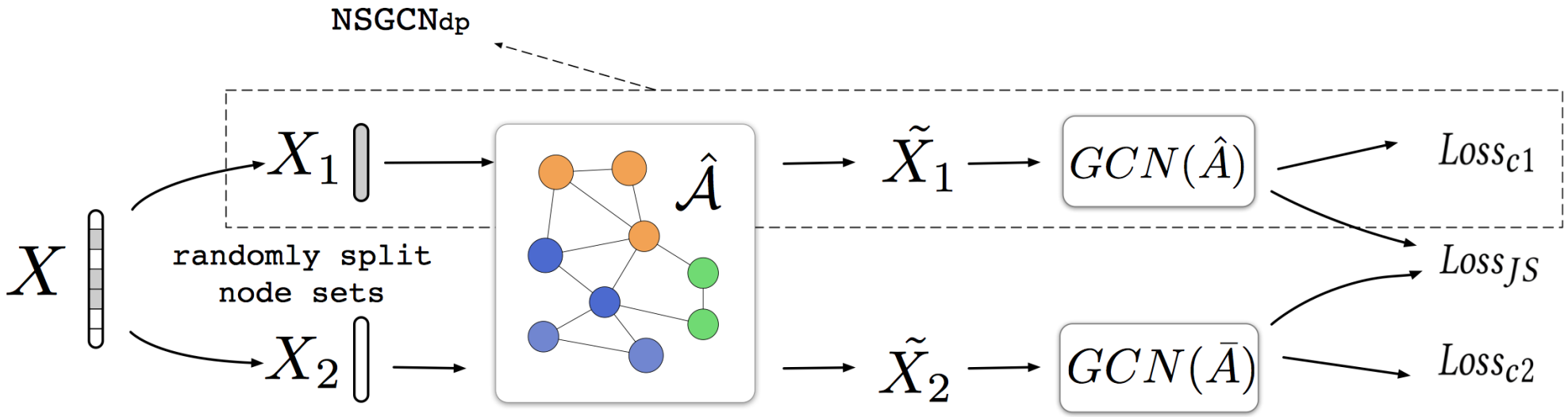
- Training Loss

$$Loss_{dm} = (Loss_{c1} + Loss_{c2})/2 + \lambda Loss_{JS}$$

- Inference

$$\hat{Z} = (\hat{Z}' + \hat{Z}'')/2$$

NSGCN: Network Sampling GCN



Experiments

- Datasets

	Cora	Citeseer	Pubmed	PPI
Nodes	2,708	3,327	19,717	56,944
Edges	5,429	4,732	44,338	818,716
Classes	7	6	3	121
Features	1,433	3,703	500	50
Training Nodes	140	120	60	44,906
Validation Nodes	500	500	500	6,514
Test Nodes	1,000	1,000	1,000	5,524
Label Rate	0.052	0.036	0.003	0.789

Experiments (transductive)

Table 2: Summary of classification accuracy (%).

Category	Method	Cora	Citeseer	Pubmed
Non-Graph Convolution	MLP	55.1	46.5	71.4
	ManiReg	59.5	60.1	70.7
	SemiEmb	59.0	59.6	71.1
	LP	68.0	45.3	63.0
	DeepWalk	67.2	43.2	65.3
	ICA	75.1	69.1	73.9
	Planetoid	75.7	64.7	77.2
	GraphSGAN	83.0±1.3	73.1±1.8	–
Graph Convolution	Chebyshev	81.2	69.8	74.4
	GCN	81.5	70.3	79.0
	MoNet	81.7±0.5	–	78.8±0.3
	DPFCNN	83.3±0.5	72.6±0.8	–
	GAT	83.0±0.7	72.5±0.7	79.0±0.3
	GAT-16*	83.3±0.6	72.4±0.7	78.8±0.3
	NRGCN (ours)	**83.6±0.6**	**73.2±0.6**	**79.1±0.3**
	NRGCN (max)	85.8	74.6	79.7

Experiments (transductive)

Table 3: Summary of classification accuracy (graph sampling-based GCNs) (%).

Method	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
FastGCN	81.4±0.5	68.8±0.9	77.6±0.5
GraphSAGE (transductive)	78.9±0.8	67.4±0.7	77.8±0.6
NSGCN$_{half}$ (ours)	79.4±1.2	67.4±1.3	78.2±0.8
NSGCN$_{dp}$ (ours)	82.4±0.5	72.1±0.6	**79.4±0.4**
NSGCN$_{dm}$ (ours)	**84.5±0.5**	**72.7±0.4**	79.2±0.3

Experiments (inductive)

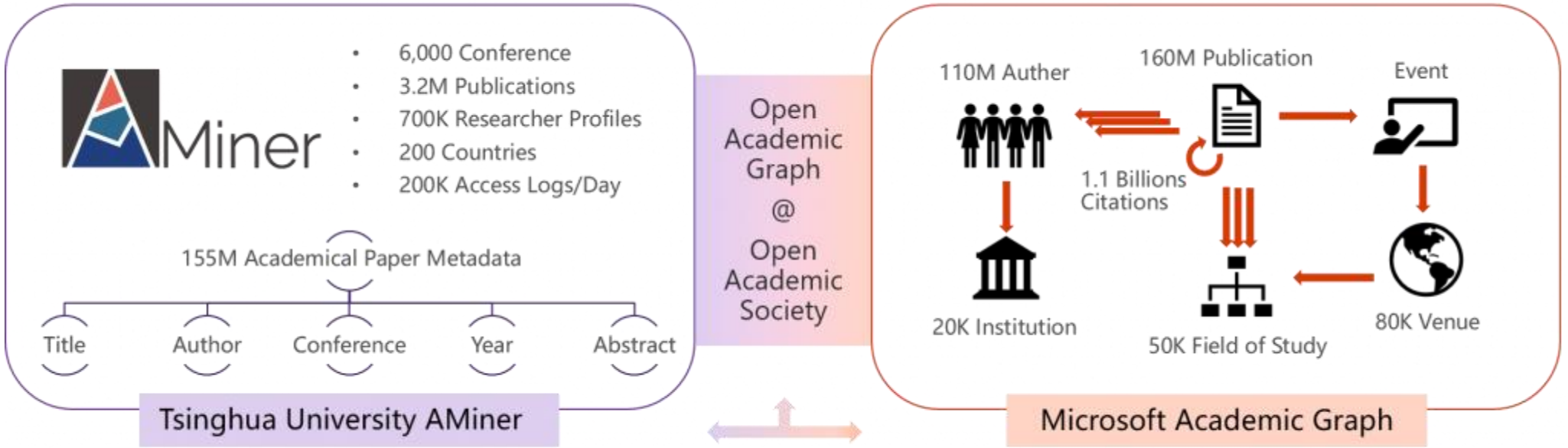
Table 4: Summary of classification accuracy (inductive learning) (%).

Method	PPI
Random	39.6
MLP	42.2
GraphSAGE-GCN	50.0
GraphSAGE-mean	59.8
GraphSAGE-LSTM	61.2
GraphSAGE-pool	60.0
GraphSAGE	76.8
GAT	97.3 ±0.2
NRGCN (ours)	**98.5 ±0.1**

OAG: Toward Linking Large-scale Heterogeneous Entity Graphs

Zhang et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. KDD'19

OAG overview



Linking *large-scale heterogeneous* academic graphs

Problem Definition

- **Heterogeneous Entity Graph (HEG):** $HG = \{E, R\}$ where each entity $e \in E$ and each relation $r \in R$ are associated with different types.
- **Input**: two heterogeneous entity graphs HG_1 and HG_2.
- **Output**: entity linkings $L = \{(e_1, e_2) | e_1 \in HG_1, e_2 \in HG_2\}$ such that e_1 and e_2 represent exactly the same entity.

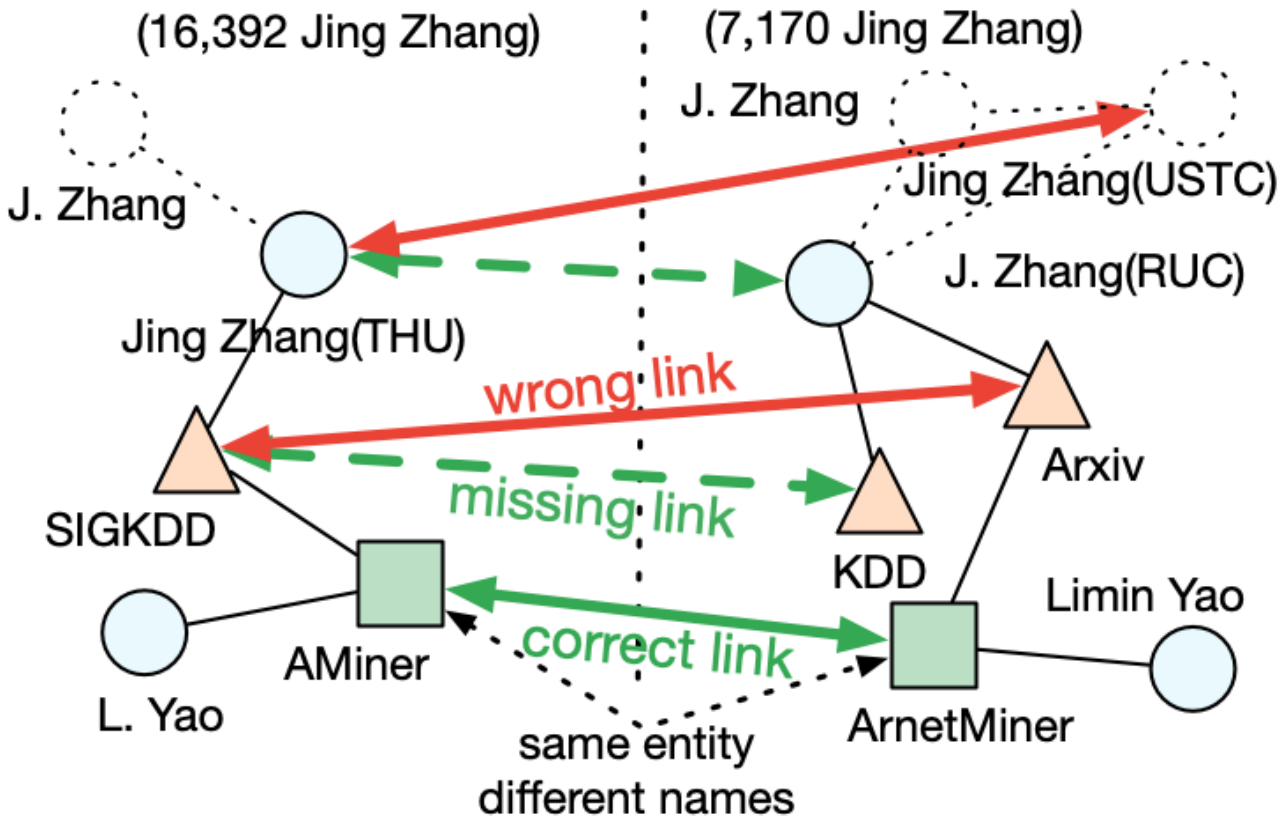
Challenges

large-scale

	AMiner	MAG
venue	*69,397*	*52,678*
paper	*172,209,563*	*208,915,369*
author	*113,171,945*	*253,144,301*

ambiguity

heterogeneity

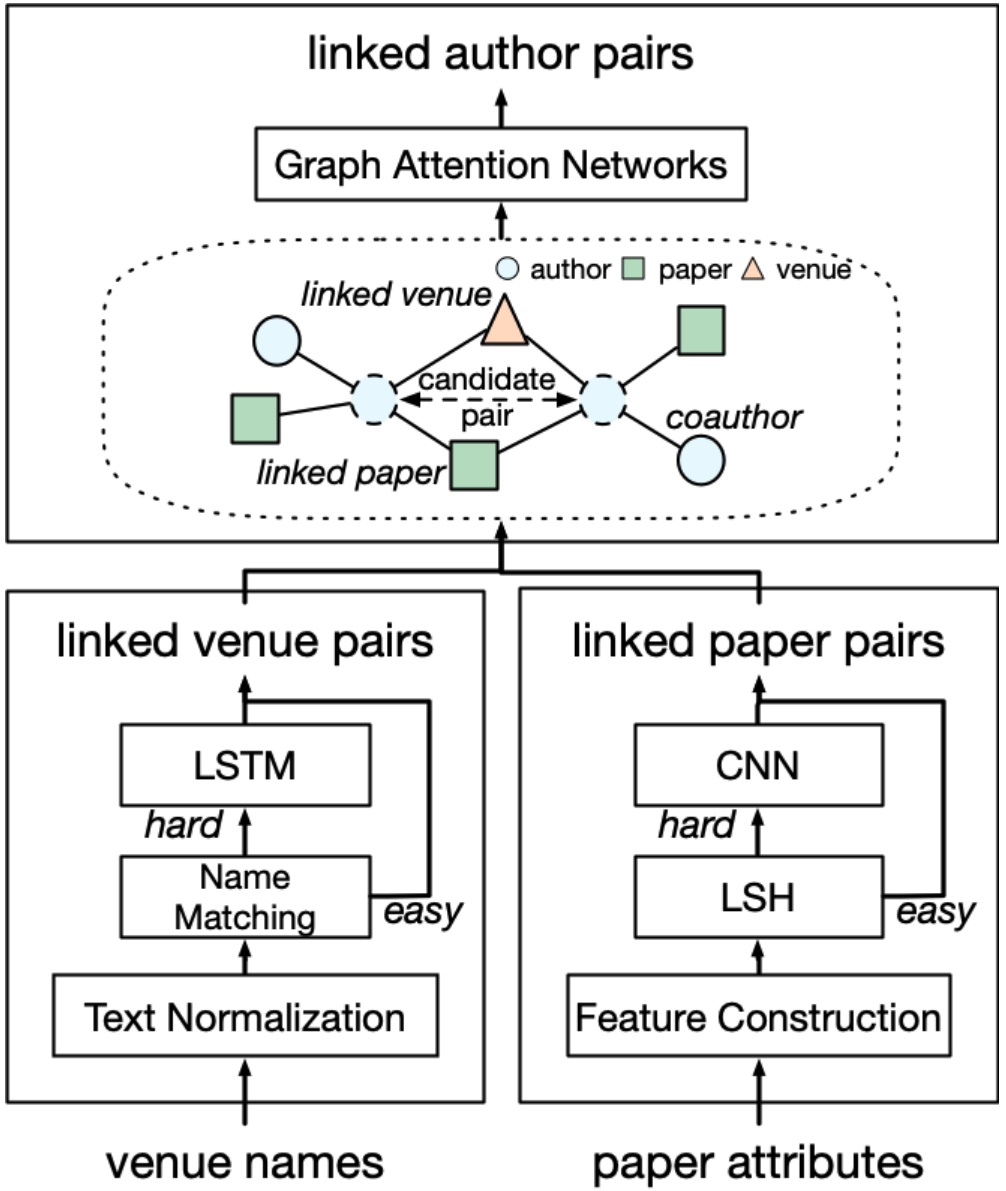


Solution -- LinKG

Author linking module

Venue linking module

Paper linking module

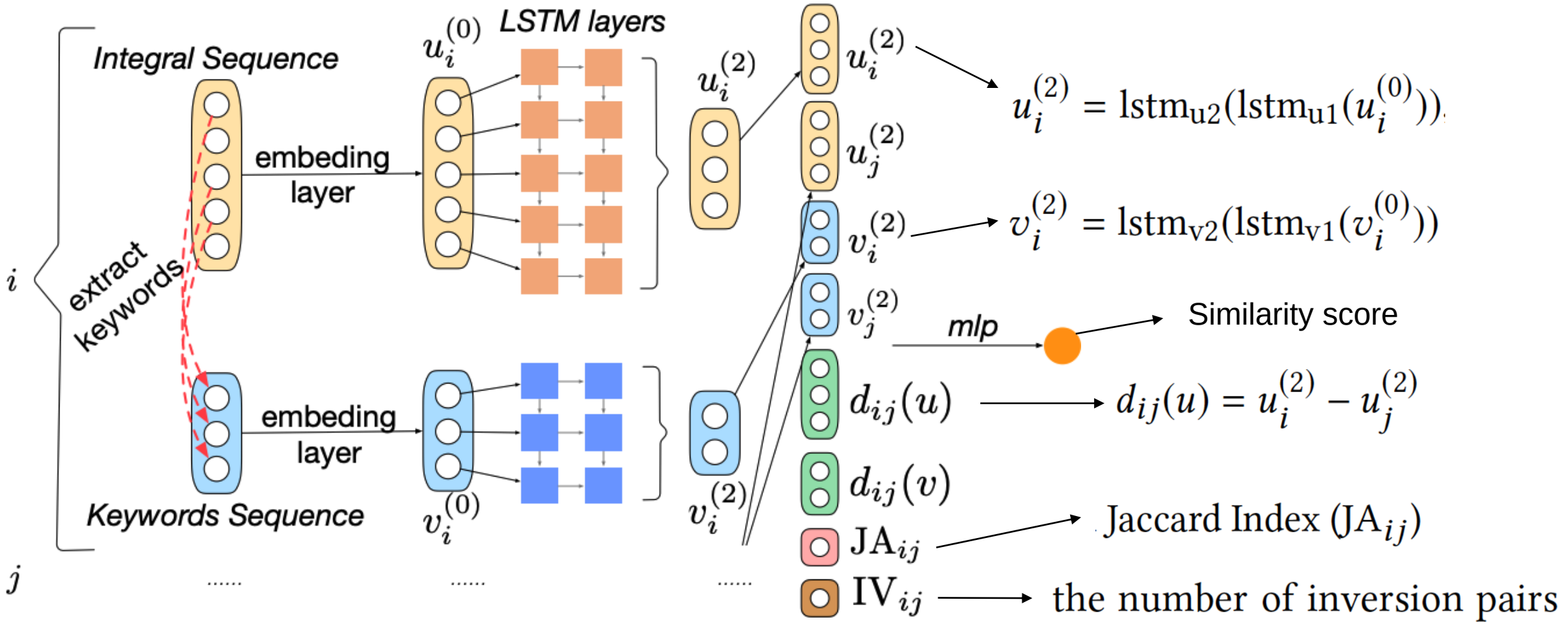


Linking Venues — Sequence-based Entities

- **Problem setting**: given the full names of venues in each graph, the goal is to link the same venues from both graphs.

- **Idea**: use direct name matching to link easy cases, and use LSTM to perform fuzzy-sequence linking.

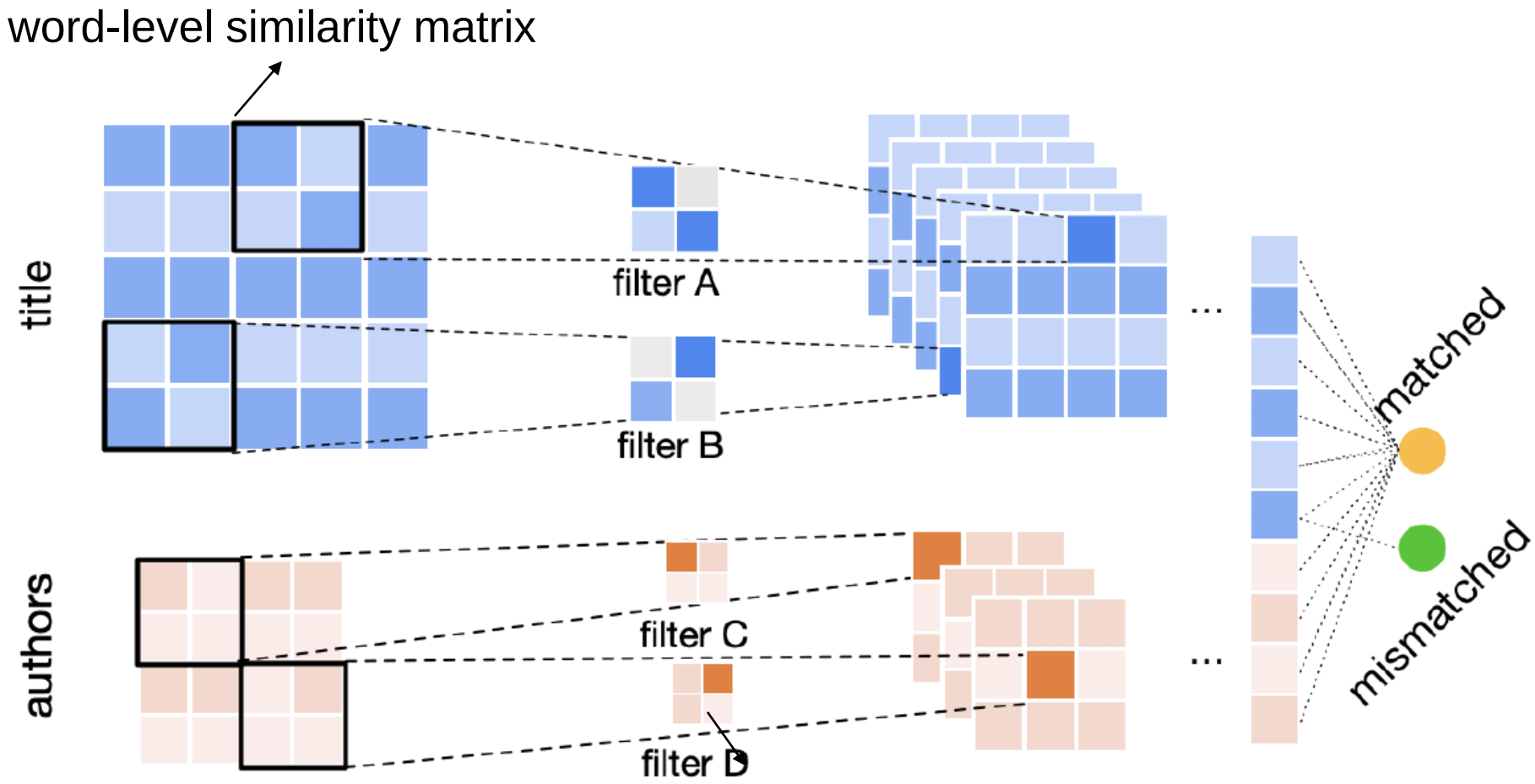
Venue Linking model



Linking Papers — Large-scale Entities

- **Problem setting**: To link paper entities, we fully leverage the heterogeneous information, including a paper's title and publication year, authors and venues.
- Leverage the **hashing technique** (LSH) for fast processing
  - Adopt Doc2Vec to transform titles to real-valued vectors
  - Use LSH to map real-valued paper features to binary codes.
- And the **convolutional neural network** for effective linking.

Paper linking model — CNN model

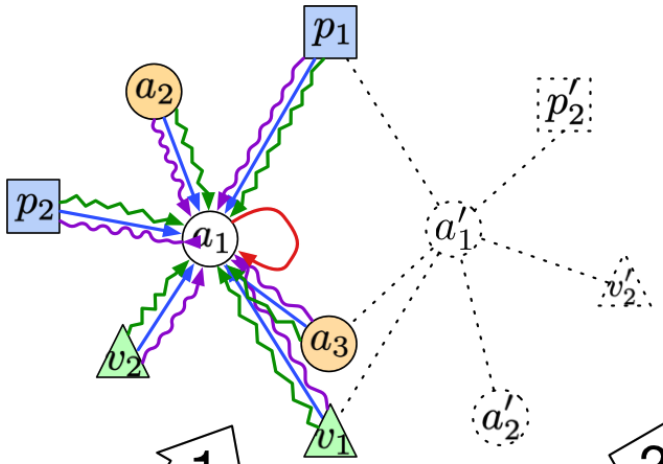


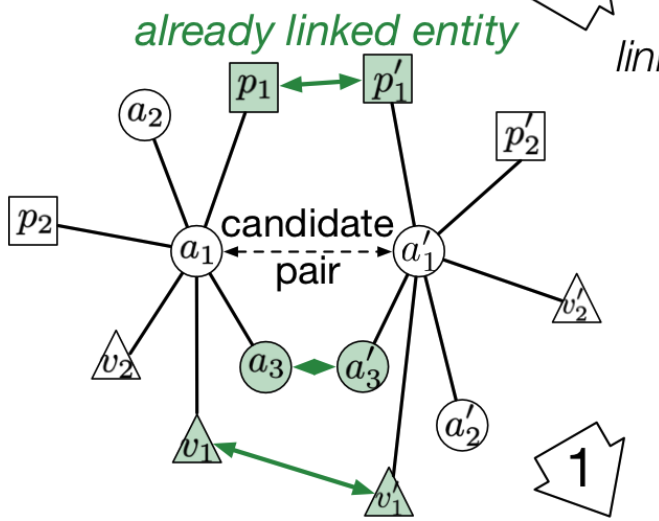
Linking Authors — Ambiguous Entities

- **Problem setting**: To link author entities, we generate a heterogeneous subgraph for each author. One author's subgraph is composed of his or her coauthors, papers, and publication venues.
- Also incorporate the **venue and paper linking results**.
- Present a heterogeneous graph attention network based technique for linking author entities.

Author linking model — Heterogenous Graph Attention

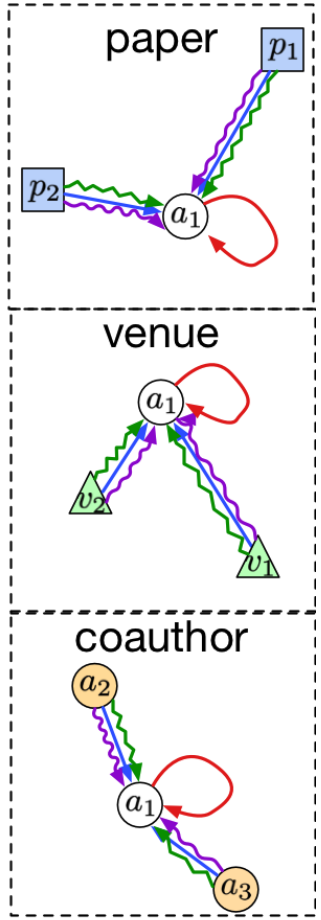
Heterogeneous subgraph
for a candidate author pair





Split

2



Different attention
parameters for
different entity types

Attention coefficient

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} W h_i + c_{\tau(e_j)}^{\top} W h_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} W h_i + c_{\tau(e_k)}^{\top} W h_k))}$$

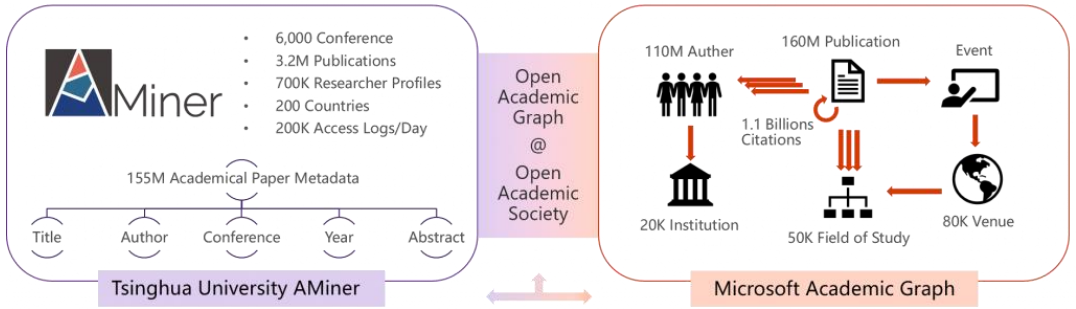
Experimental Results

Table 1: Results of linking heterogeneous entity graphs. "–" indicates the method does not support the entity linking.

Methods		Keyword	SVM	Dedupe	COSNET	MEgo2Vec	LinKG$_C$	LinKG$_L$	LinKG
Venue	Prec.	80.15	81.69	84.25	–	–	84.67	**91.16**	**91.16**
	Rec.	83.76	83.45	80.92			85.81	**87.58**	**87.58**
	F1	81.91	82.56	82.55			85.23	**89.33**	**89.33**
Paper	Prec.	91.01	96.93	**99.30**	–	–	98.68	86.72	98.68
	Rec.	80.53	96.78	87.09			**98.10**	86.59	**98.10**
	F1	85.45	96.86	92.80			**98.39**	86.66	**98.39**
Author	Prec.	44.48	84.70	50.65	91.73	91.03	81.30	84.92	**95.37**
	Rec.	80.63	92.22	85.46	85.33	90.82	84.95	**94.75**	93.48
	F1	57.33	88.30	63.60	88.42	90.92	83.09	89.57	**94.42**
Overall	Prec.	74.80	92.36	82.26	91.73	91.03	92.38	86.21	**97.36**
	Rec.	80.64	94.89	86.38	85.33	90.82	93.29	89.41	**96.26**
	F1	77.61	93.61	84.27	88.42	90.92	92.83	87.78	**96.81**

OAG: Open Academic Graph

https://www.openacademic.ai/oag/



Data set	#Pairs/Venues	Date
Linking relations	29,841	2018.12
AMiner venues	69,397	2018.07
MAG venues	52,678	2018.11

Table 1: statistics of OAG venue data

Data set	#Pairs/Papers	Date
Linking relations	91,137,597	2018.12
AMiner papers	172,209,563	2019.01
MAG papers	208,915,369	2018.11

Table 2: statistics of OAG paper data

Data set	#Pairs/Authors	Date
Linking relations	1,717,680	2019.01
AMiner authors	113,171,945	2018.07
MAG authors	253,144,301	2018.11

Open Academic Graph

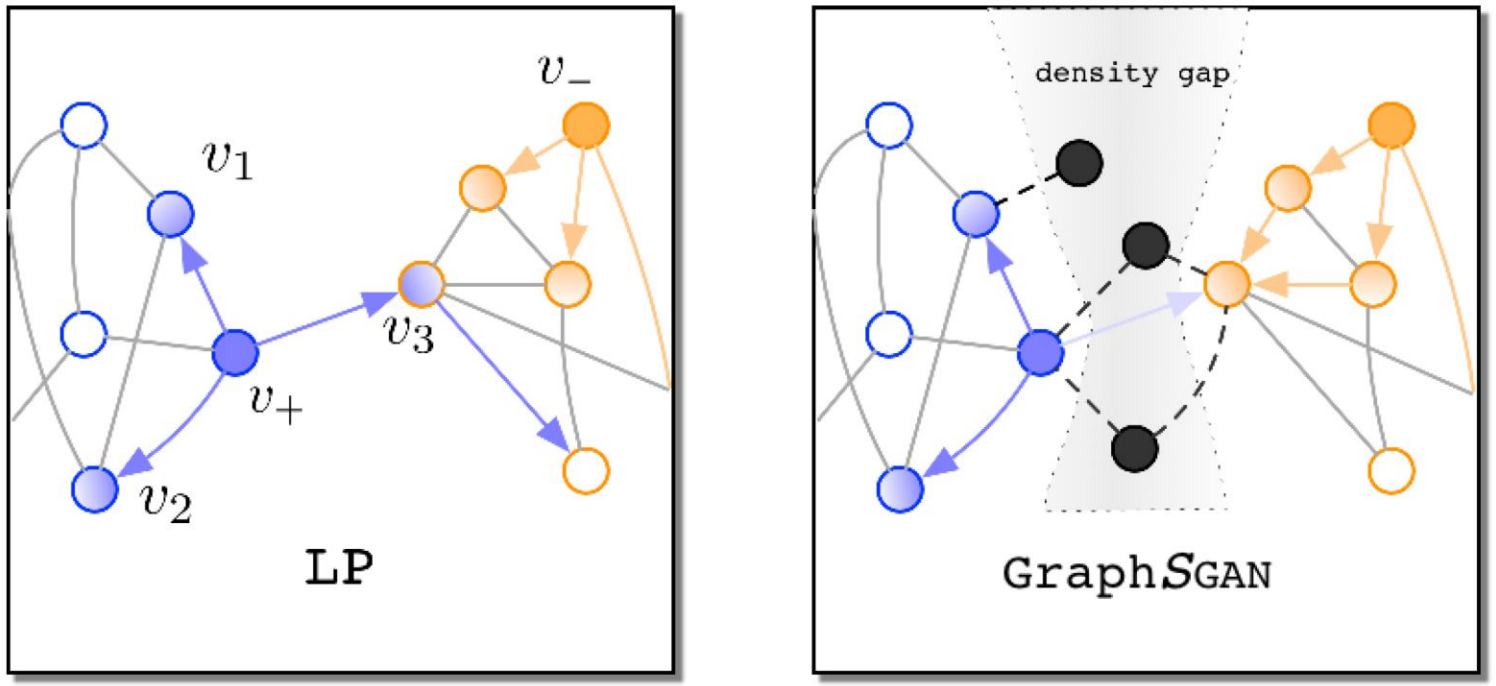
Open Academic Graph (OAG) is a large knowledge graph unifying two billion-scale academic graphs: Microsoft Academic Graph (MAG) and AMiner. In mid 2017, we published OAG v1, which contains 166,192,182 papers from MAG and 154,771,162 papers from AMiner (see below) and generated 64,639,608 linking (matching) relations between the two graphs. This time, in OAG v2, author, venue and newer publication data and the corresponding matchings are available.

Overview of OAG v2

The statistics of OAG v2 is listed as the three tables below. The two large graphs are both evolving and we take MAG November 2018 snapshot and AMiner July 2018 or January 2019 snapshot for this version.

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

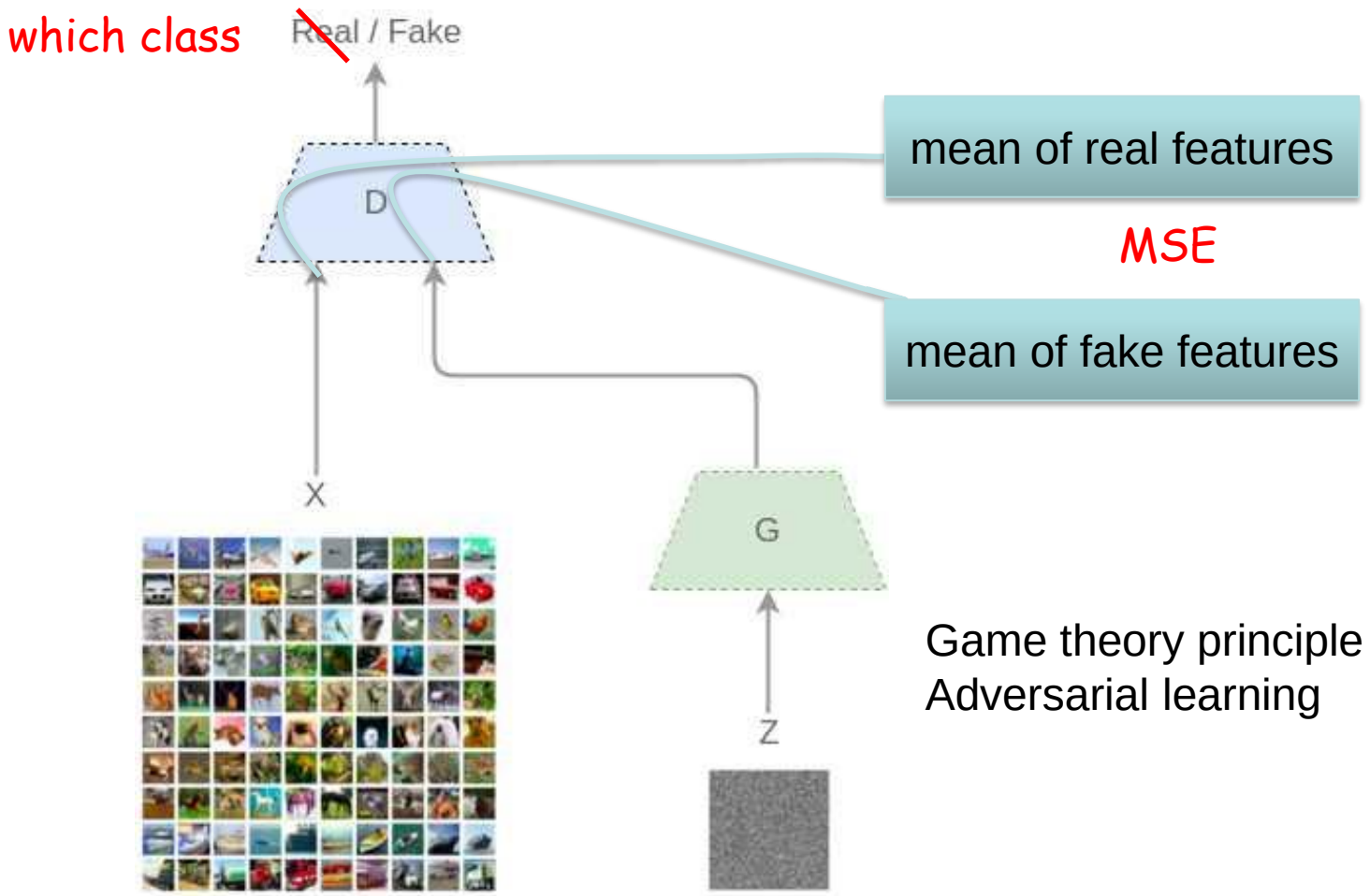
Semi-supervised Learning on Graphs using GANs



- Generate samples in low-density areas
- Reduce influence across these areas

1. Ming Ding, Jie Tang, and Jie Zhang. Semi-supervised Learning on Graphs with Generative Adversarial Nets. In CIKM'18.

Semi-GAN in CV



Semi-supervised Problem Formulation

- Labeled nodes: $L = \{(v_i, l_i)\}$
- Unlabeled nodes: $U = \{u_i\}$
- Graph: $G = (L + U, E)$
- Feature vectors of nodes: $P = \{p_i | p_i \in \mathbb{R}^k\}$

- *Transductive learning:*
  - Predict labels of *U*

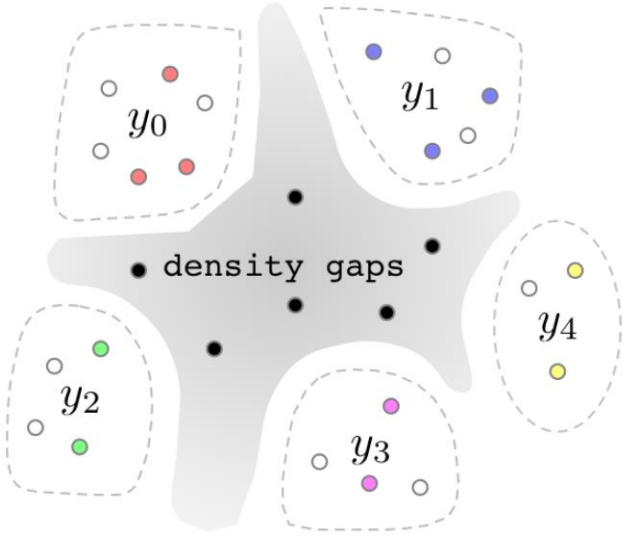
1. Ming Ding, Jie Tang, and Jie Zhang. Semi-supervised Learning on Graphs with Generative Adversarial Nets. In CIKM'18.

How to generate samples in density gaps?

- Original GAN —— zero-sum game
  - Only mimics real data

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \log[1 - D(G(\mathbf{z}))]$$

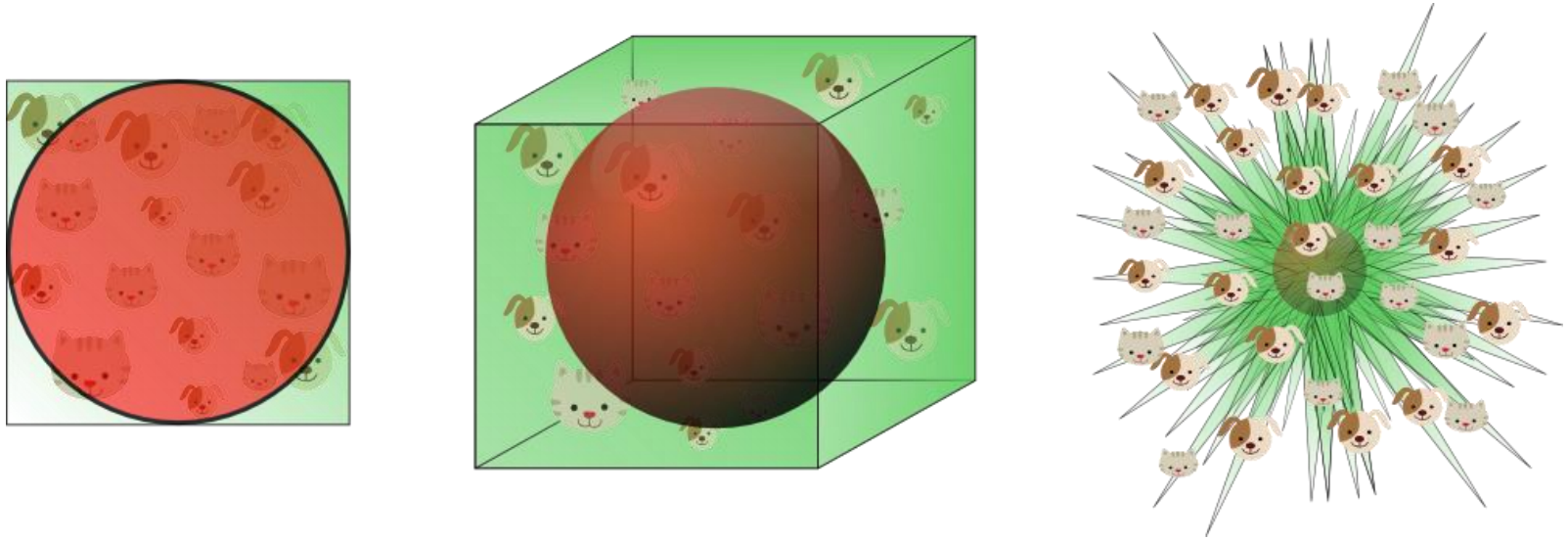
- Expected Nash Equilibrium



In final representation layer:

1. Density gaps lie in the center
2. Both labeled and unlabeled samples are classified into distinct clusters

Curse of dimensionality



Training data in central area are easy to become hubs, which frequently occurs as the nearest neighbor of data from other classes.

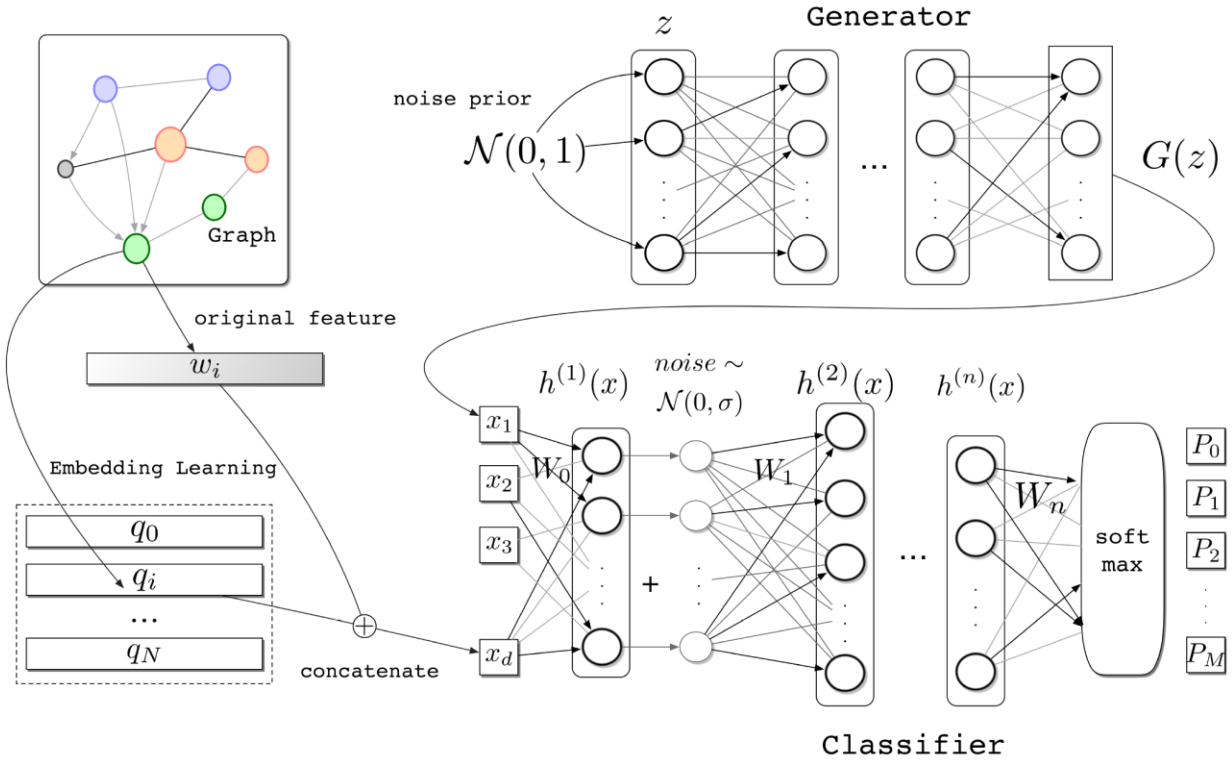
Nash Equilibrium

- “No player can change their strategy to reduce his loss unilaterally ”
- For D:
  - Nodes from different classes should be mapped into different clusters.
  - Both labeled and unlabeled nodes should not be mapped into central area so as to make it a density gap.
  - Every unlabeled node should be mapped into one cluster representing a particular label.
  - Different clusters should be far enough.
- For G:
  - G generates samples which are mapped into the central area.
  - Generated samples should not overfit at the only center point.

$$\mathcal{L}_D = loss_{sup} + \lambda_0 loss_{un} + \lambda_1 loss_{ent} + loss_{pt}$$

$$\mathcal{L}_G = loss_{fm} + \lambda_2 loss_{pt}$$

GraphSGAN Model



$$loss_{ent} = -\mathbb{E}_{\mathbf{x}_i \in X^U} \sum_{y=0}^{M-1} P(y|\mathbf{x}_i, y_i < M) \log P(y|\mathbf{x}_i, y_i < M)$$

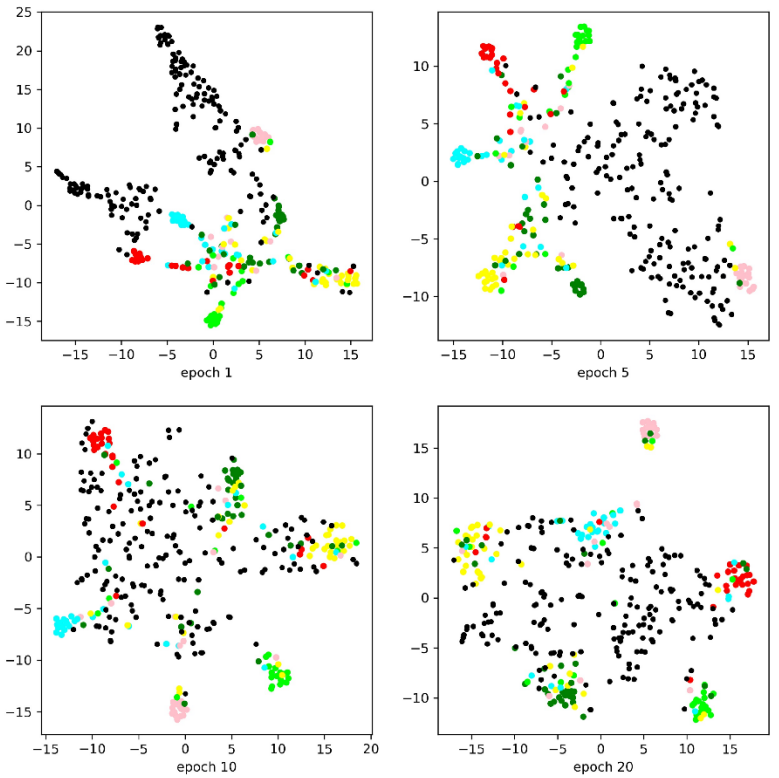
$$loss_{sup} = -\mathbb{E}_{\mathbf{x}_i \in X^L} \log P(y_i|\mathbf{x}_i, y_i < M)$$

$$loss_{un} = -\mathbb{E}_{\mathbf{x}_i \in X^U} \log[1 - P(M|\mathbf{x}_i)] - \mathbb{E}_{\mathbf{x}_i \sim G(\mathbf{z})} \log P(M|\mathbf{x}_i)$$

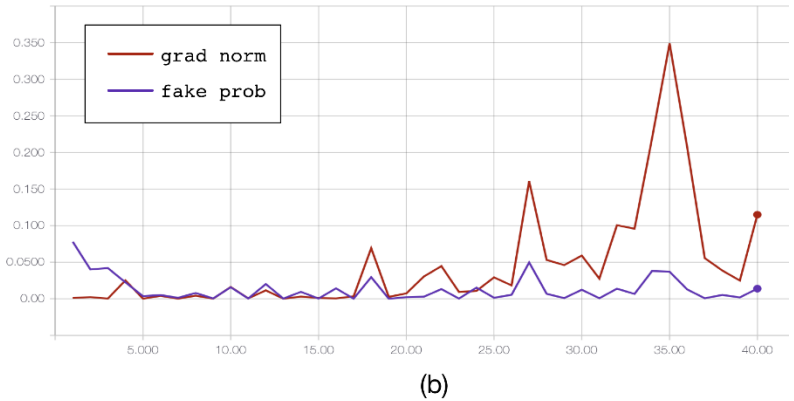
$$loss_{fm} = ||\mathbb{E}_{\mathbf{x}_i \in X_{batch}} h^{(n)}(\mathbf{x}_i) - \mathbb{E}_{\mathbf{x}_j \sim G(\mathbf{z})} h^{(n)}(\mathbf{x}_j)||_2^2$$

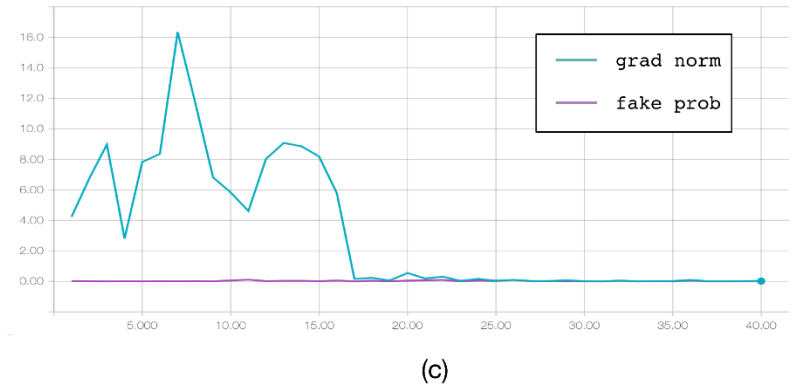
$$loss_{pt} = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \frac{h^{(n)}(\mathbf{x}_i)^\top h^{(n)}(\mathbf{x}_j)}{||h^{(n)}(\mathbf{x}_i)||||h^{(n)}(\mathbf{x}_j)||}^2$$

Verification



Visualization of final layer





Trend of unsmoothness and fake probability

Experiments

Method	Cora %	Citeseer %
ManiReg	59.5	60.1
SemiEmb	59.0	59.6
LP	68.0	45.3
DeepWalk	67.2	43.2
ICA	75.1	69.1
Planetoid	75.7	64.7
GCN	80.1±0.5	67.9± 0.5
AGNN	81.0±0.3	69.8±0.35
GAT	**83.0±0.7**	72.5± 0.7
GraphSGAN	**83.0±1.3**	**73.1± 1.8**

Citation Networks

- Cora:
  - 7 classes
  - 2708 nodes
  - 5429 links
  - 1433 set-of-words features
  - 120 labeled nodes
- Citeseer:
  - 6 classes
  - 3327 nodes
  - 4732 links
  - 3703 set-of-words features
  - 120 labeled nodes

Experiments

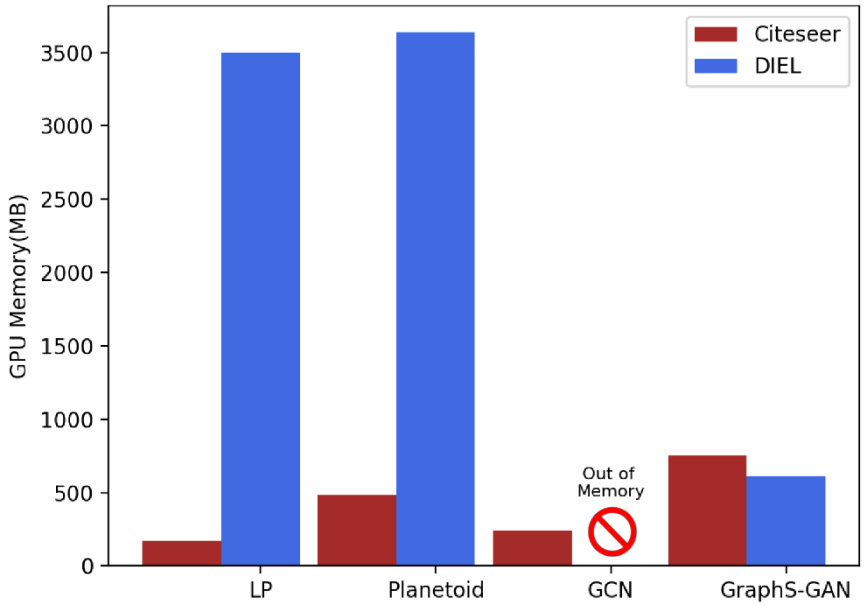
Results on DIEL dataset (medical entity extraction):

- 4,373,008 nodes (3414 labeled)
- 4,464,261 edges
- 1,233,597 features

Category	Method	Recall@K
Regularization	Label propagation	16.2
	ManiReg	47.7
Embedding	SemiEmb	25.8
	Planetoid	50.1
Original	DIEL	40.5
Our	GraphSGAN	**51.8**
	Upper bound	61.7

Space Efficiency

GPU memory consumption:



Mini-batch Training:

- Planetoid
- GraphSGAN

Full-batch Training:

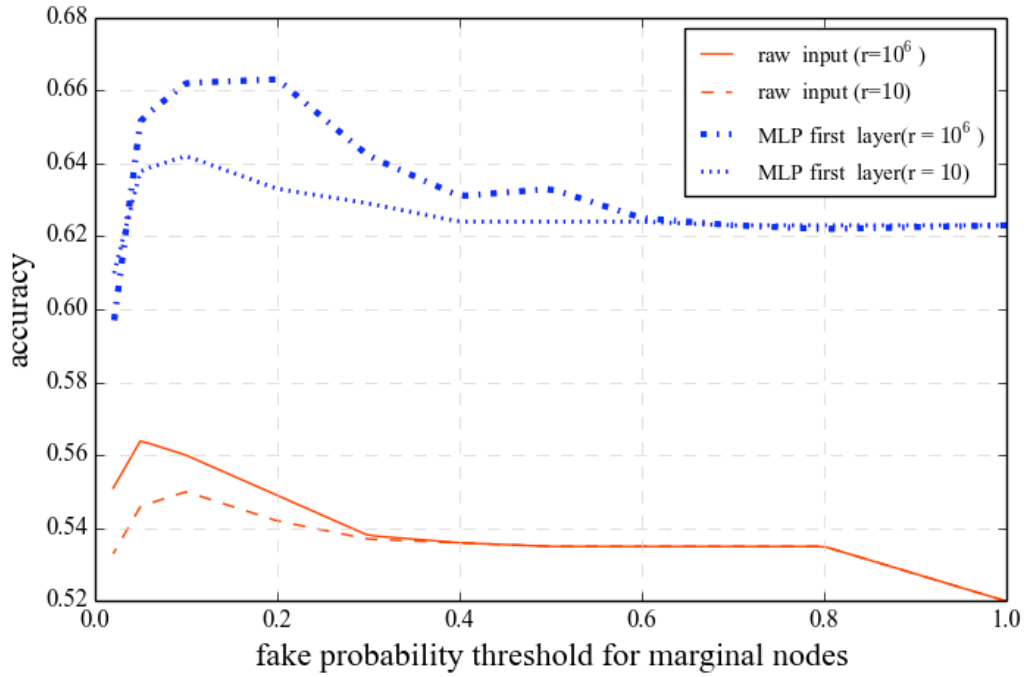
- GCN

CPU training:

- Label propagation

Adversarial Label Propagation

- Theoretical analysis
  - We proved that with limited generated samples in density gaps, we can get a perfect classification under Laplacian Regularization.
- Verification experiments
  - Use generated nodes to help Label Propagation



Summary

- We propose a GAN-based algorithm GraphSGAN to solve semi-supervised learning problem on graphs.
- The model not only reaches start-of-art performance and but also enjoys scalability.
- A new game theory framework is proposed to explain the algorithm.
- The generated samples can be used to optimized label propagation with theoretical guarantee.

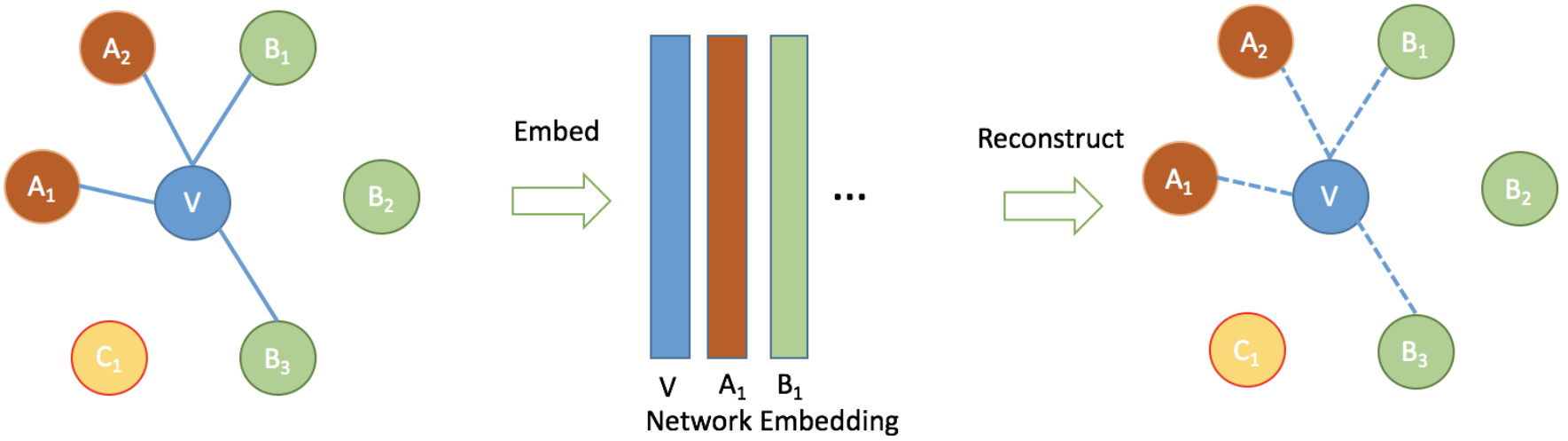
Dynamic Network Embedding with Burst Detection

Zhao et al. Large Scale Evolving Graphs with Burst Detection. IJCAI'19

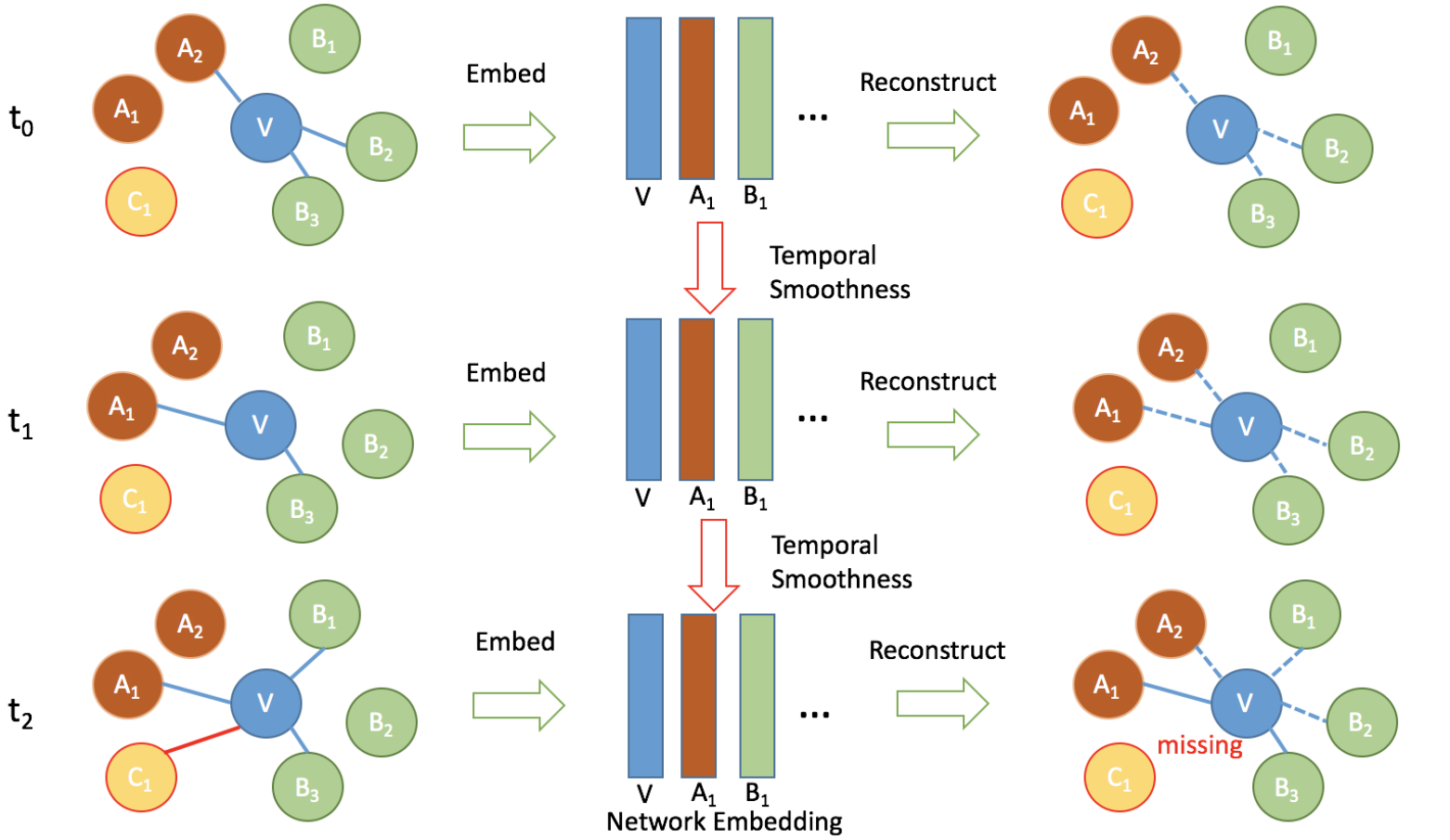
Problem

- Give a dynamic network $\{G_1, G_2, \cdots, \mathrm{G_T}\}$, this task is to learn a series of functions $f_\Phi = (f_{\Phi_1}, f_{\Phi_1}, \cdots, f_{\Phi_T})$.
- Each function $f_{\Phi_t}: V_t \to \mathbb{R}^d$ is to capture both vanilla evolution and bursty evolution in dynamics.

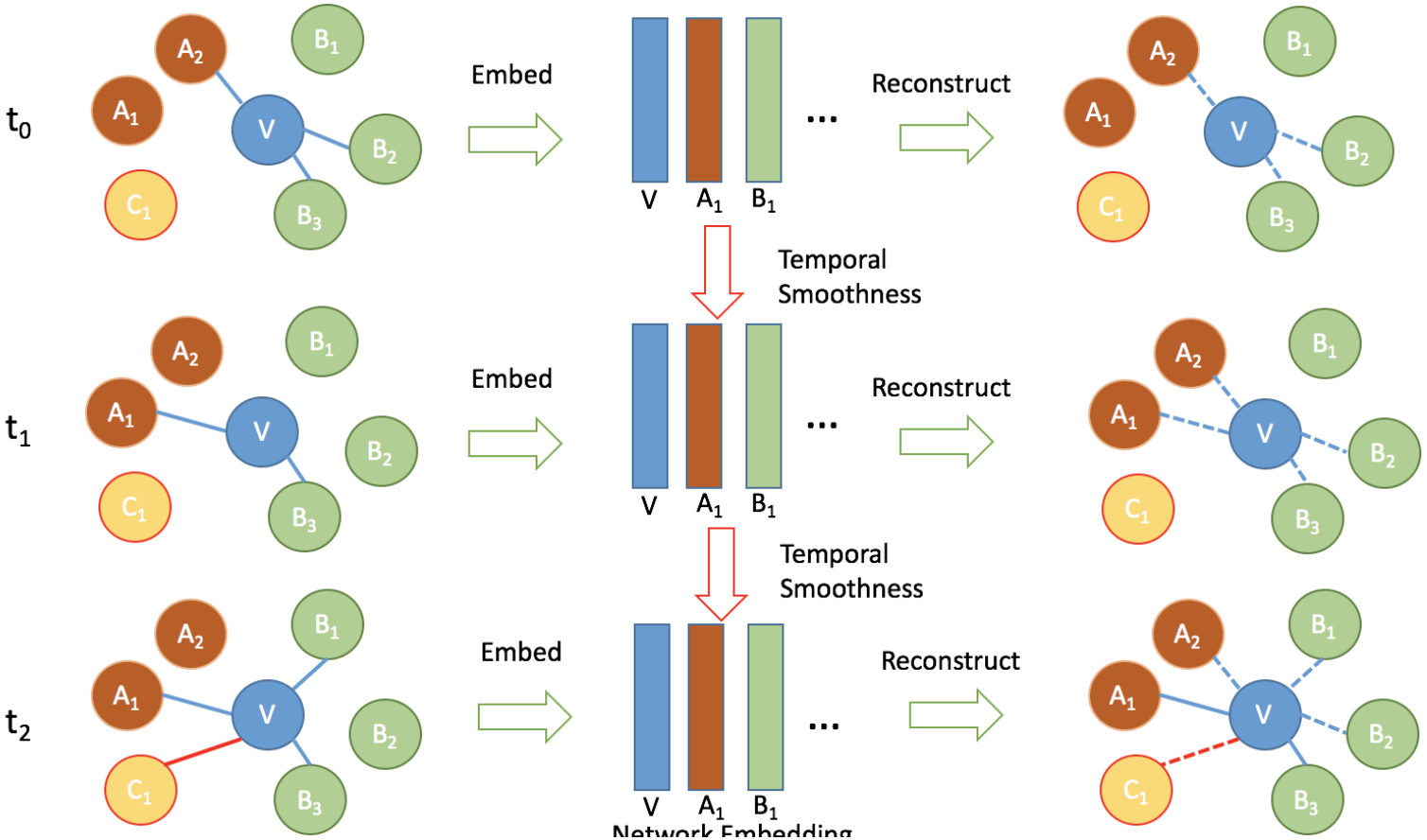
Static Network Embedding



Dynamic Network Embedding



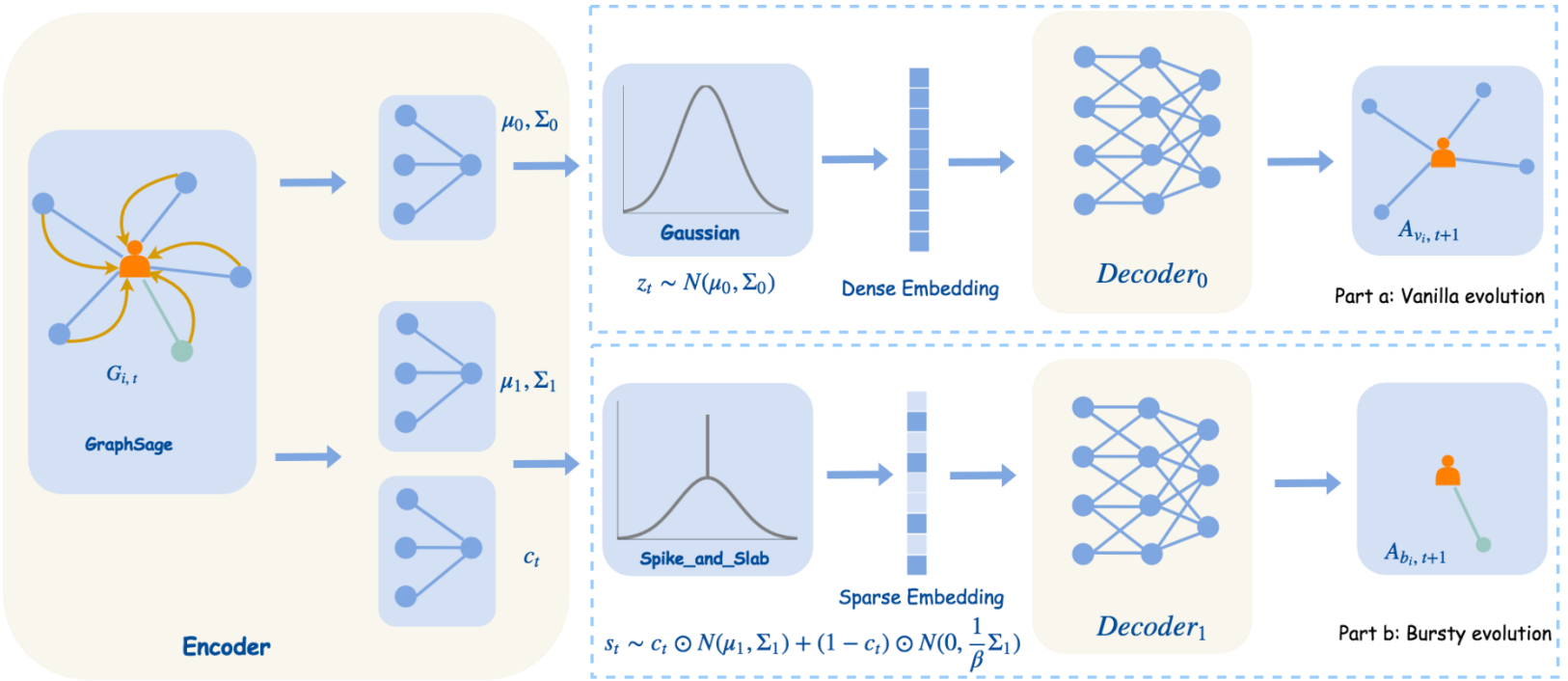
Dynamic Network Embedding with Burst Detection



- **Bursty Dynamics**. For each vertex, the factors underlying bursty links may change over time. For example, in E-commerce network, customers may purchase rarely noticed products due to the weather (e.g., rainy day). Compared to vanilla evolution, bursty evolution is more sparse and discrete.

- **Correlations with Vanilla and Bursty Evolution**. The vanilla and bursty evolution independently represent the characteristics of the vertices, but they still have mutual promotion on each other.

The BurstGraph Framework



- GraphSage aggregates structural information for each snapshot of dynamic networks.
- VAE reconstructs the vanilla evolution and bursty evolution.
- RNN Structure extends the model for dynamic networks.

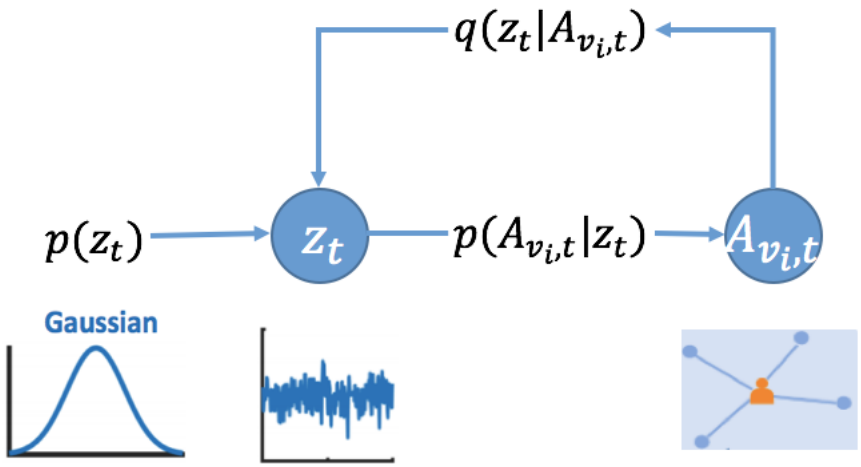
Information Aggregation

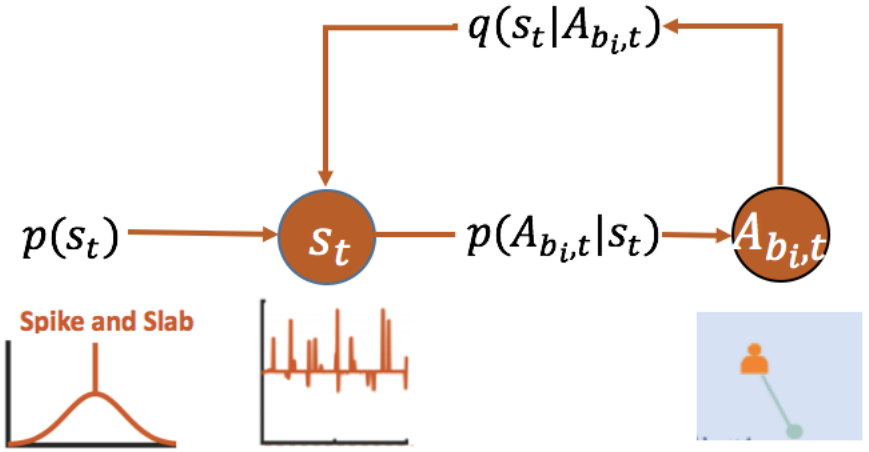
- To get the feature information of each snapshot, we utilize GraphSage to aggregate the hidden representation from vertices' neighbors.
- In each network snapshot, GraphSAGE samples neighbors via uniform distribution.
- Then GraphSage uses simple max-pooling operator to aggregate information from neighbors.

Vanilla Evolution

- Based on the feature information $G_{i,t}$ aggregated by GraphSAGE, vanilla and bursty evolution both choose VAE to reconstruct the structure of network snapshot;
- The random variable $\boldsymbol{z}_t$ of vanilla evolution follows a normal Gaussian distribution:

$$\boldsymbol{z}_t | G_{i,t} \sim \mathcal{N}(\mu_0(G_{i,t}), \Sigma_0(G_{i,t}))$$

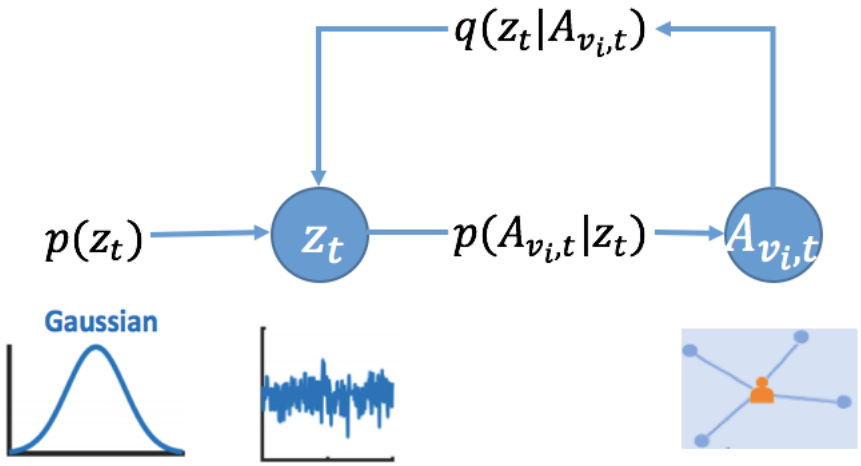


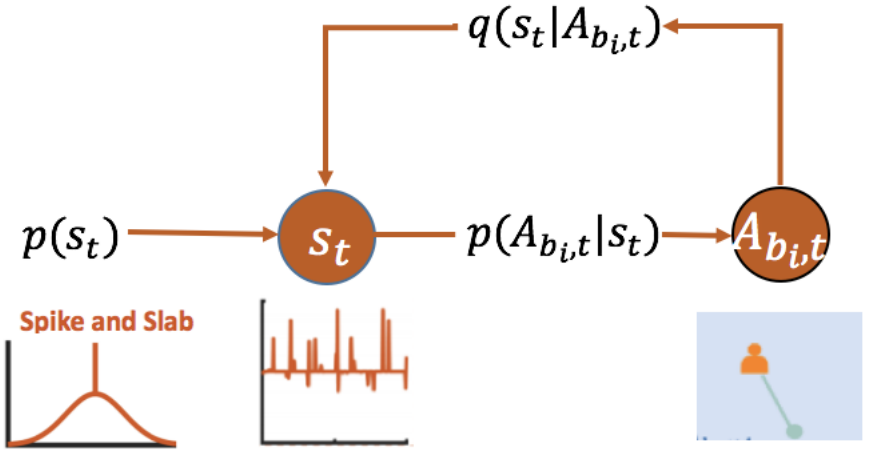


Bursty Evolution

- Due to the sparsity and contingency of bursty evolution, the random variable $\boldsymbol{s}_t$ of bursty evolution follows a Spike-and-Slab distribution:

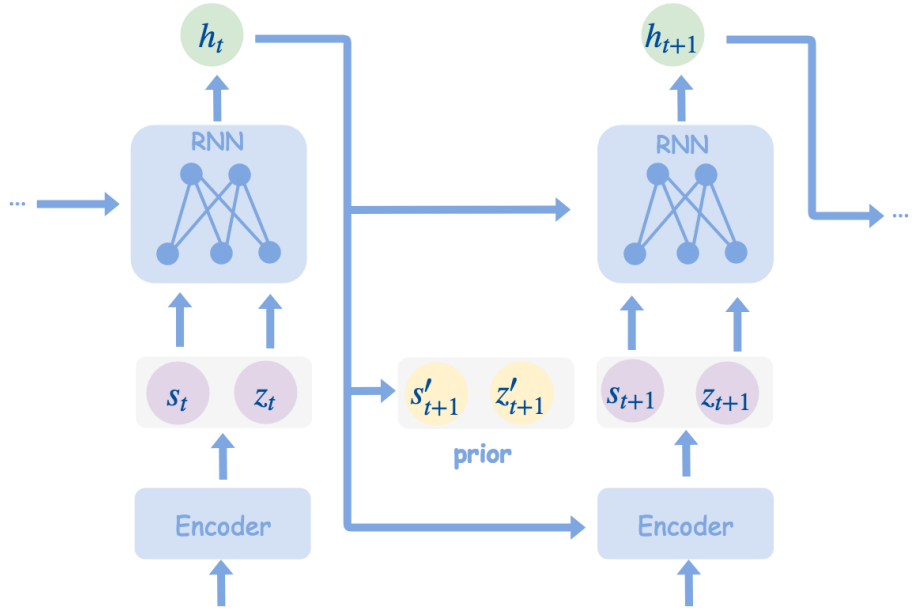
$$\boldsymbol{c}_t | G_{i,t} \overset{iid}{\sim} \text{Bernoulli}(\psi(G_{i,t}))$$
$$\boldsymbol{r}_{t,1} | G_{i,t} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\beta}\Sigma_1(G_{i,t}))$$
$$\boldsymbol{r}_{t,2} | G_{i,t} \sim \mathcal{N}(\mu_1(G_{i,t}), \cdot\Sigma_1(G_{i,t}))$$
$$\boldsymbol{s}_t = (\mathbf{1} - \boldsymbol{c}_t) \odot \boldsymbol{r}_{t,1} + \boldsymbol{c}_t \odot \boldsymbol{r}_{t,2},$$





Temporal Consistence

- Taking the temporal structure of the dynamic networks into account, we introduce an RNN structure into our framework.



$$
\begin{aligned}
\boldsymbol{z}_t | G_{i,t}, \boldsymbol{h}_{t-1} &\sim \mathcal{N}(\mu_0(G_{i,t}, \boldsymbol{h}_{t-1}), \Sigma_0(G_{i,t}, \boldsymbol{h}_{t-1})) \\
\boldsymbol{r}_{t,1} | G_{i,t}, \boldsymbol{h}_{t-1} &\sim \mathcal{N}(\boldsymbol{0}, \frac{1}{\beta} \cdot \Sigma_1(G_{i,t}, \boldsymbol{h}_{t-1})) \\
\boldsymbol{r}_{t,2} | G_{i,t}, \boldsymbol{h}_{t-1} &\sim \mathcal{N}(\mu_1(G_{i,t}, \boldsymbol{h}_{t-1}), \Sigma_1(G_{i,t}, \boldsymbol{h}_{t-1})) \\
\boldsymbol{c}_t | G_{i,t}, \boldsymbol{h}_{t-1} &\overset{iid}{\sim} \text{Bernoulli}(\psi(G_{i,t}, \boldsymbol{h}_{t-1})), \\
\boldsymbol{h}_t &= f_h(\boldsymbol{h}_{t-1}, \boldsymbol{z}_t, \boldsymbol{s}_t)
\end{aligned}
$$

Objective Function

- Following the VAE framework, the ELBO of our model can be written as:

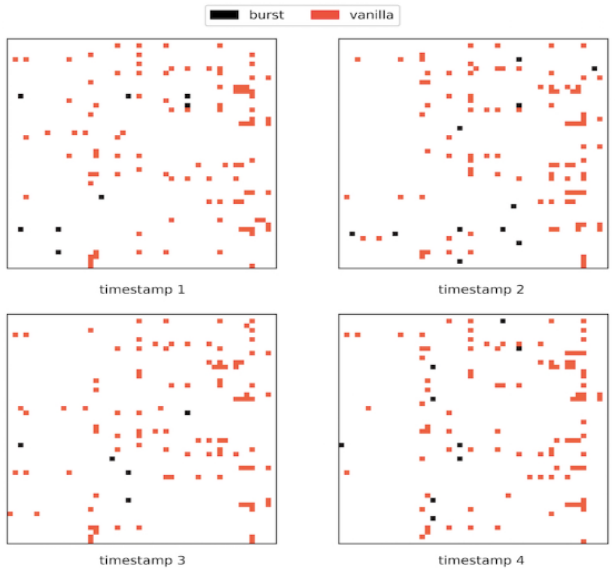
$$Loss = \sum_{t=1}^{T} E_{\boldsymbol{z}_t \sim q_\phi(\boldsymbol{z}_t|G_{i,t})}[\ln \frac{p_\theta(\boldsymbol{A}_{v_i,t}|\boldsymbol{z}_t)p(\boldsymbol{z}_t)}{q_\phi(\boldsymbol{z}_t|G_{i,t})}] + \lambda \cdot E_{\boldsymbol{s}_t \sim q_\phi(\boldsymbol{s}_t|G_{i,t})}[\ln \frac{p_\theta(\boldsymbol{A}_{b_i,t}|\boldsymbol{s}_t)p(\boldsymbol{s}_t)}{q_\phi(\boldsymbol{s}_t|G_{i,t})}]$$

- The edge evolution is always sparse and unbalanced. It brings trouble to identity the positive instances to achieve better performance.
- Instead of traditional sigmoid cross entropy loss function, we use inter-and-intra class balanced loss as reconstruction loss:

$$L_{rec} = \underbrace{\frac{1}{2}(\frac{L_{pos}^{nod}}{n_{pos}^{nod}} + \frac{L_{neg}^{nod}}{n_{neg}^{nod}})}_{inter-class\ loss} + \underbrace{\frac{1}{2}(\frac{L_{pos}^{cls}}{n_{pos}^{cls}} + \frac{L_{neg}^{cls}}{n_{neg}^{cls}})}_{intra-class\ loss}$$

Experiment Setup

- Link Prediction
  - Simulated dataset: A+
  - Static NE: DeepWalk, GraphSage
  - Dynamic NE: TNE, CTDNE



Adjacency matrix heatmap of sub-graphs with four timestamps of the simulated dataset A+

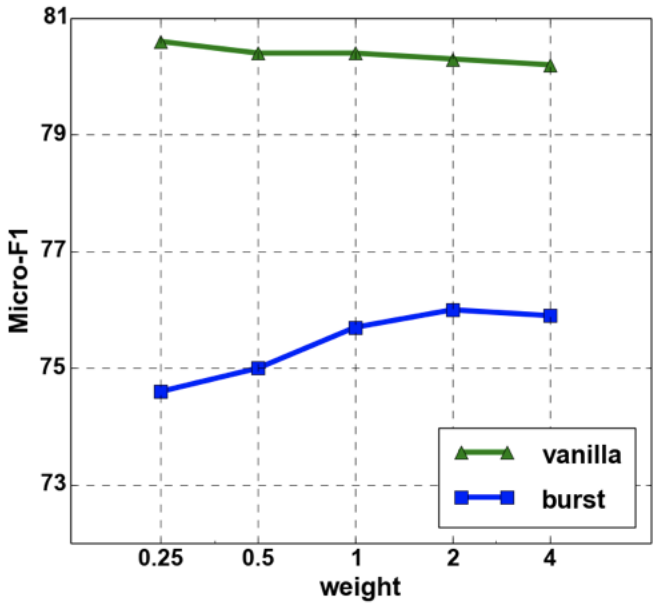
Table: Statistics of datasets.

dataset	#vertices	#edges	#time
Simulate	20,118	527,268	6
A+-Small	19,091	479,441	6
A+-Large	25,432	3,745,819	6

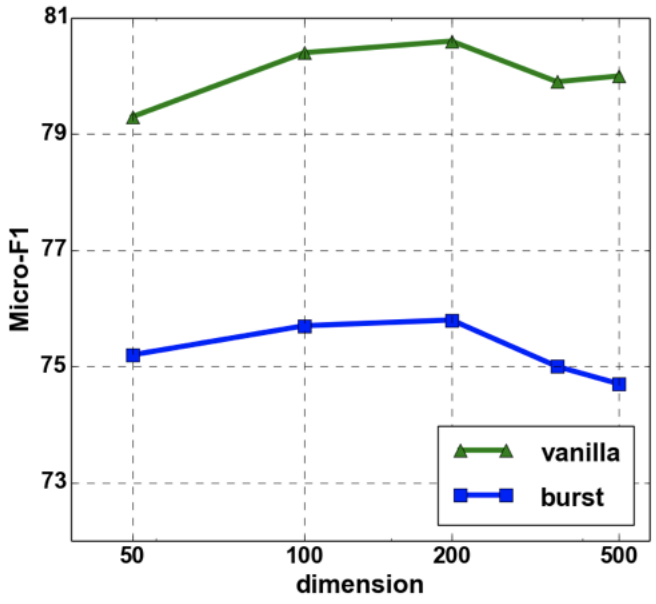
Result

Model	Simulated				A+-Small				A+-Large			
	vanilla evolution		bursty evolution		vanilla evolution		bursty evolution		vanilla evolution		bursty evolution	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
DeepWalk	73.5	71.2	71.7	69.8	79.0	74.7	71.9	70.2	80.7	70.4	68.1	64.6
GraphSAGE	88.3	87.4	70.8	69.2	74.6	74.2	67.8	67.8	76.1	72.8	69.7	68.5
TNE	75.9	74.3	69.1	68.3	77.6	72.1	70.4	68.2	79.9	73.9	69.1	67.2
CTDNE	72.2	69.6	70.9	68.7	79.3	75.0	72.2	70.4	80.5	70.0	67.8	64.3
BurstGraph	**91.5**	**91.4**	**78.8**	**78.4**	**80.4**	**78.4**	**75.7**	**74.8**	**81.4**	**77.7**	**73.3**	**70.8**

Result: Parameter Sensitivity



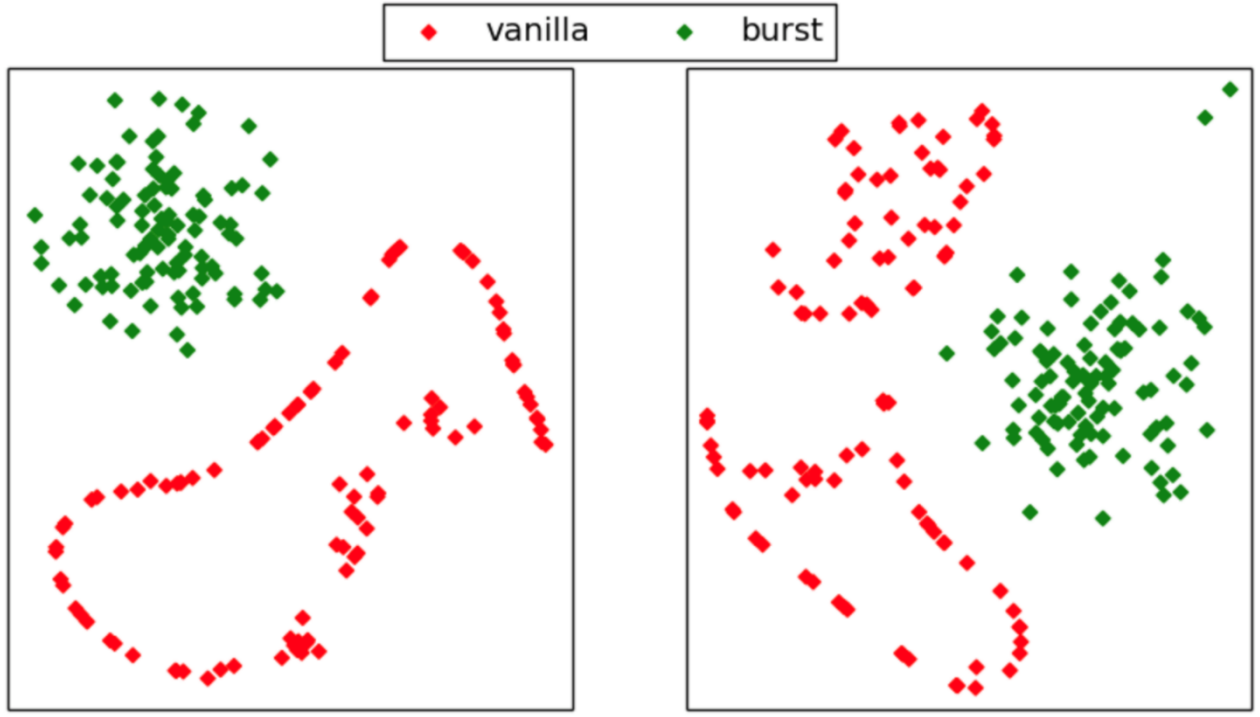
a) importance weight



b) variable dimension

The performance of BurstGraph on A+ small dataset with increasing importance weight λ and variable dimension d

Result: Embedding Visualization



a) Simulated Dataset

b) A+-Small Dataset

2D visualization on embeddings (100 randomly selected vertices) for vanilla evolution and bursty evolution

Dynamic Network Embedding under Partial Monitoring

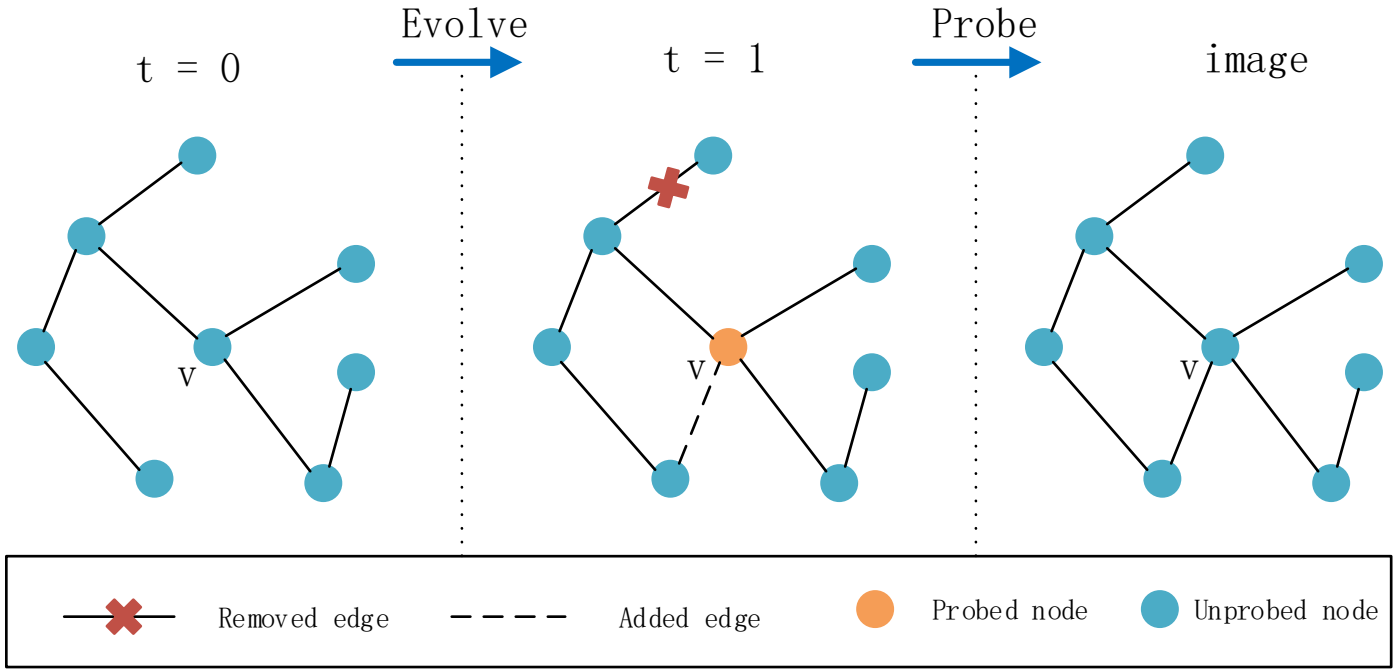
Han et al. Network Embedding under Partial Monitoring for Evolving Networks. IJCAI'19

Motivation & Problem

- Two challenges for network embedding on real world networks
  - Networks are dynamic
  - More importantly, sometimes we have to query the nodes to learn about the changes of the networks, such as the web networks or some large social network platforms. Unfortunately, due to the great cost of the query, we can only probe part of the nodes, i.e. network embedding under partial monitoring.

Problem

What is network embedding under partial monitoring?



We can only probe part of the nodes to perceive the change of the network!

Background

Let’s first revisit traditional network embedding models:

- Key idea of network embedding models: extracting the proximities among the nodes so that the embedding values can preserve and reflect the extracted proximities.

- So one proximity results in one network embedding algorithm.

Background

- We define the proximity matrix.

Definition 1. ***Proximity Matrix.*** *For each kind of node proximity in a network, there is a corresponding proximity matrix. We use* M *to denote the proximity matrix, which is an* $N \times N$ *matrix and* $M_{i,j}$ *represents the value of the corresponding proximity from node* v_i *to* v_j*.*

- Given proximity matrix M, we can perform network embedding with matrix factorization:

$$\mathcal{O}_1 = \min_{X,Y} \|M - XY^T\|_F.$$

X is the embedding table.

Y is the embedding table when the nodes act as context.

Background

- Given graph G = (V, A), any kinds of proximity can be exploited by network embedding models, such as:
  - Adjacency Proximity $M^{(AP)} = A.$
  - Adamic-Adar Proximity $M_{i,j}^{(AA)} = \sum_{z \in Neighbor(v_i) \cap Neighbor(v_j)} \frac{1}{\ln |Neighbor(z)|}.$
  - Katz Proximity
  - Jaccard's Coefficient Proximity
  - SimRank Proximity

Problem

Problem 1. ***Dynamic Network Embedding under Partial Monitoring*** *Given time stamps* $< 0, 1, ..., T >$, *the proximity matrix* M_{t_0} *and the distributed representation of the nodes* X_{t_0} *and* Y_{t_0} *at time* t_0, *we need to figure out a strategy, denoted as* π, *to choose at most* $K < N$ *nodes to probe at each following time stamp* $< 1, 2, ..., T >$, *so that it minimizes the discrepancy between the approximate distributed representation, denoted as* $\hat{f}_t(V)$, *and the potentially best distributed representation* $f_t^*(V)$, *as described by the following objective function.*

$$\mathcal{O}_2 = \min_{\pi} \sum_{t=1}^{T} Discrepancy(f_t^*(V), \hat{f}_t(V))$$

How to figure out the strategy to select the nodes.

Problem

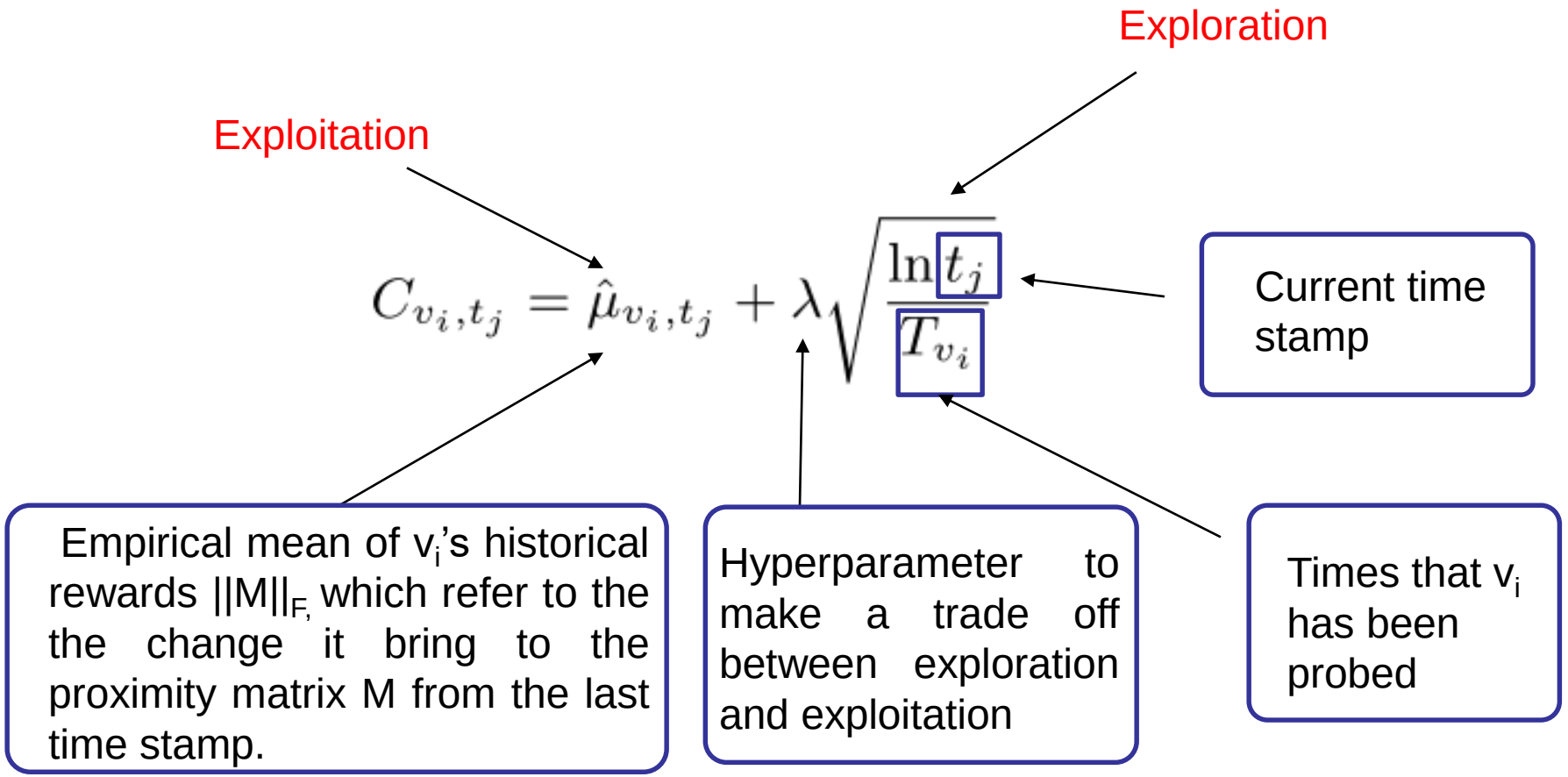
- It is a sequential decision problem
- Obviously, the best strategy is to capture as much “change” as possible with limited “probing budget”.

Credit Probing Network Embedding

- Based on a kind of reinforcement learning problem, namely Multi-armed Bandit (MAB)
- Choose the “productive” nodes according to their historical “rewards”.
- At each time stamp t_j , we maintain a “credit” for each node v_i, which is the consideration for selecting the nodes.
- The “credit” should make a trade-off between exploitation and exploration.

Credit Probing Network Embedding

- The “credit” for each node v_i at time stamp t_j can be defined as:



Credit Probing Network Embedding

Algorithm 1: Credit Probing Network Embedding

Input: An initial network $G = (V, A)$, dimension p, proximity kind, time stamp $< 0, 1, ..., T >$, parameter λ, probing budget K.

Output: approximate node representation matrix X_{t_i} at each time stamp

Initialize $\hat{\mu}_{v_i}, T_{v_i}, C_{v_i}$ for each node.
foreach *time stamp* t_j **do**
    probe the nodes with the highest credit.
    update the approximate network structure $\hat{G}_{t_j}$.
    calculate the approximate proximity matrix $\hat{M}_{t_j}$.
    calculate X_{t_i}.
    foreach *node* v_i **do**
        update T_{v_i}
        update C_{v_i,t_j} according to Eq.4.
    end
end

Credit Probing Network Embedding

- How to evaluate the difference between two embeddings X and X*?
- Obviously, it makes no sense to measure their concrete values with $||X-X^*||_F$.
- So we define two metrics: Magnitude Gap and Angle Gap from their geometric meanings.

Credit Probing Network Embedding

- Magnitude Gap $MG = \|S^* - \hat{S}\|_2$

- Angle Gap

$$AG = \sqrt{\|\sin\Theta\|_F^2 + \|\sin\Phi\|_F^2}$$
$$= \sqrt{\frac{\|P_{U^*} - P_{\hat{U}}\|_F^2 + \|P_{V^*} - P_{\hat{V}}\|_F^2}{2}},$$

where P_{U^*} is the orthogonal projection operator of U^*, which can be achieved by $P_{U^*} = U^*U^{*\dagger} = U^*(U^{*T}U^*)^{-1}U^{*T}$, in which $(\cdot)^{-1}$ is the inverse of the corresponding matrix and $(\cdot)^\dagger$ is Moore-Penrose pseudoinverse. In the same way, we can get $P_{\hat{U}}$, P_{V^*} and $P_{\hat{V}}$ with $\hat{U}$, V^* and $\hat{V}$ respectively.

Credit Probing Network Embedding

- We prove the error bound for loss of magnitude gap and angle gap with matrix perturbation theory and combinatorial multi-armed bandit theory:

$$L_{MG} \leq \sum_{v_i \in V} \frac{4\lambda^2 N^2 \ln T}{\Delta_{v_i}^{min}} + (1 + \sum_{d=1}^{\infty} d^{1-2\lambda^2}) N \Delta^{max}$$

$$L_{AG} \leq \frac{\sqrt{\sum_{v_i \in V} \frac{8\lambda^2 N^2 \ln T}{\Delta_{v_i}^{min}} + 2(1 + \sum_{d=1}^{\infty} d^{1-2\lambda^2}) N \Delta^{max}}}{\delta}$$

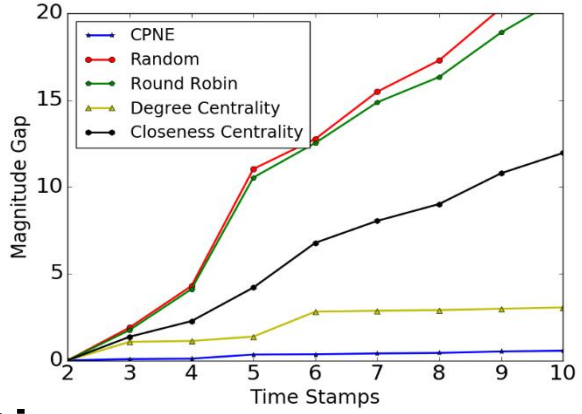
Experimental Setting

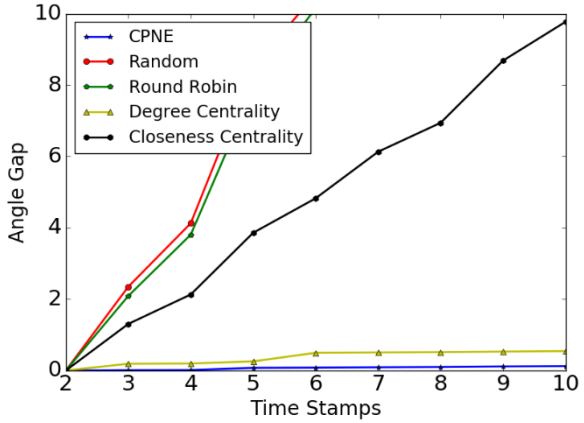
- Approaching the Potential Optimal Values
  - Datasets: CAIDA
  - Baselines: Random, Round Robin, Degree Centrality, Closeness
  - Metrics: Magnitude Gap, Angle Gap
- Link Prediction
  - Datasets: Wechat
  - Baselines: BCGD[1] with the four settings
  - Metrics: AUC

1. Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Transactions on Knowledge and Data Engineering, 28(10):2765–2777, 2016

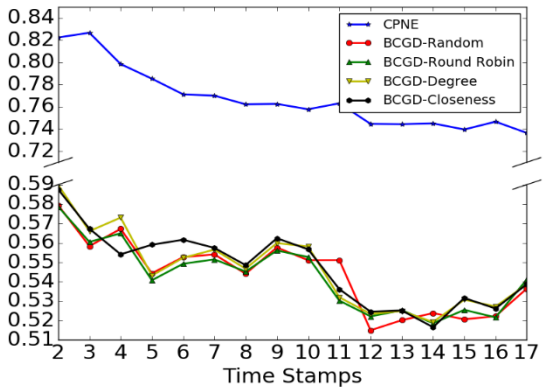
Experimental Results

- Approaching the Potential Optimal Values

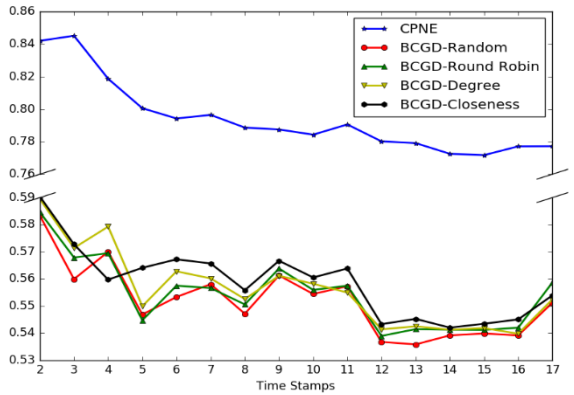




- Link Prediction

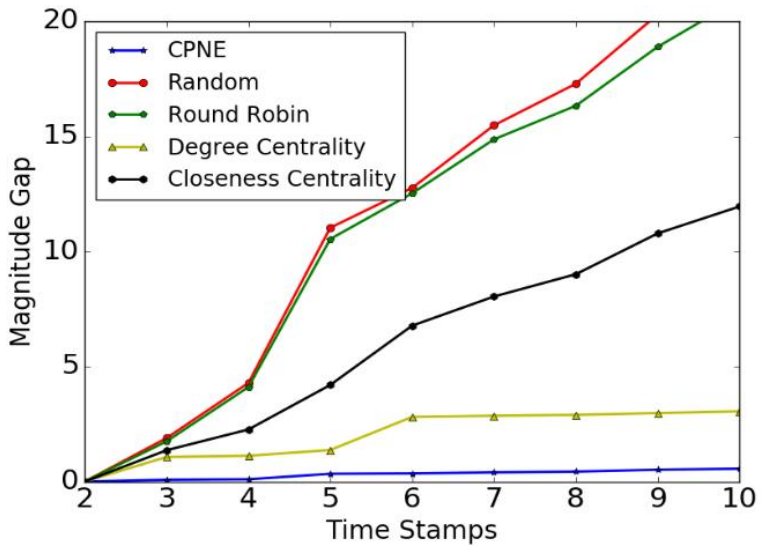


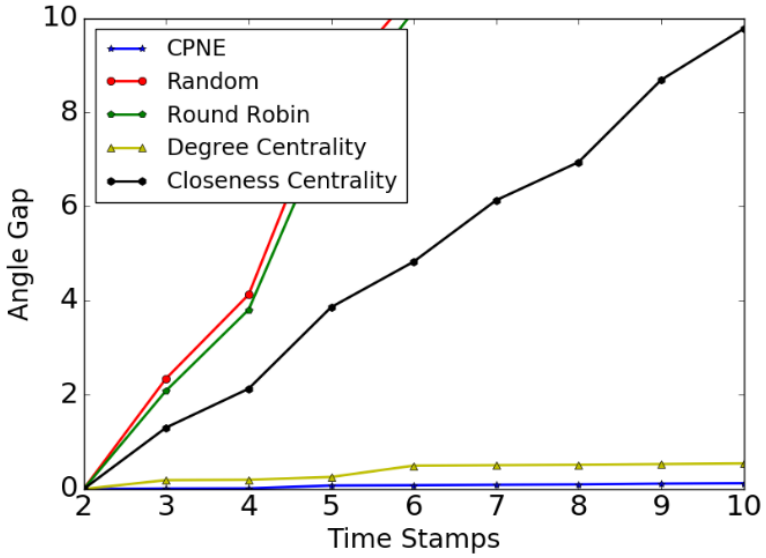
K = 500



K = 1000

Experimental Results





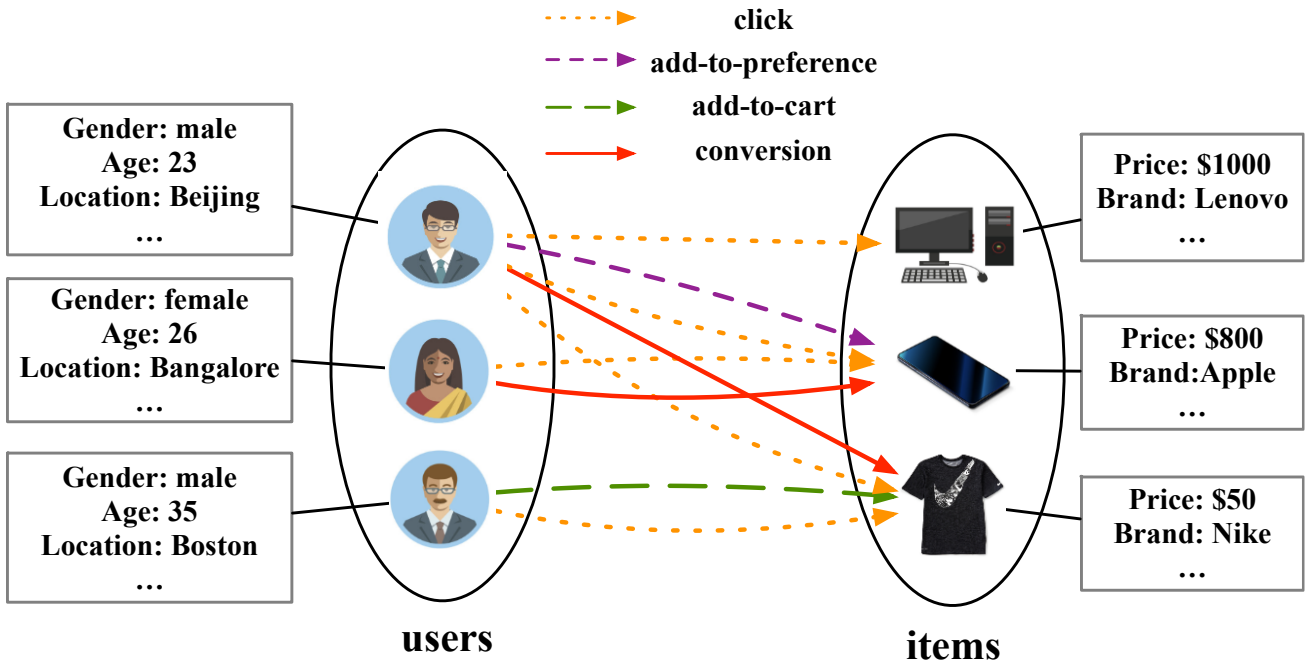
Approaching the Potential Optimal Values

Multiplex Heterogeneous Graph Embedding

Cen et al. Representation Learning for Attributed Multiplex Heterogeneous Network. KDD'19

Multiplex Heterogeneous Graph

- Real-world network-structured applications are much more complicated, comprising not only multi-typed nodes and/or edges but also a rich set of attributes.
- How do we deal with the multiplex edges?



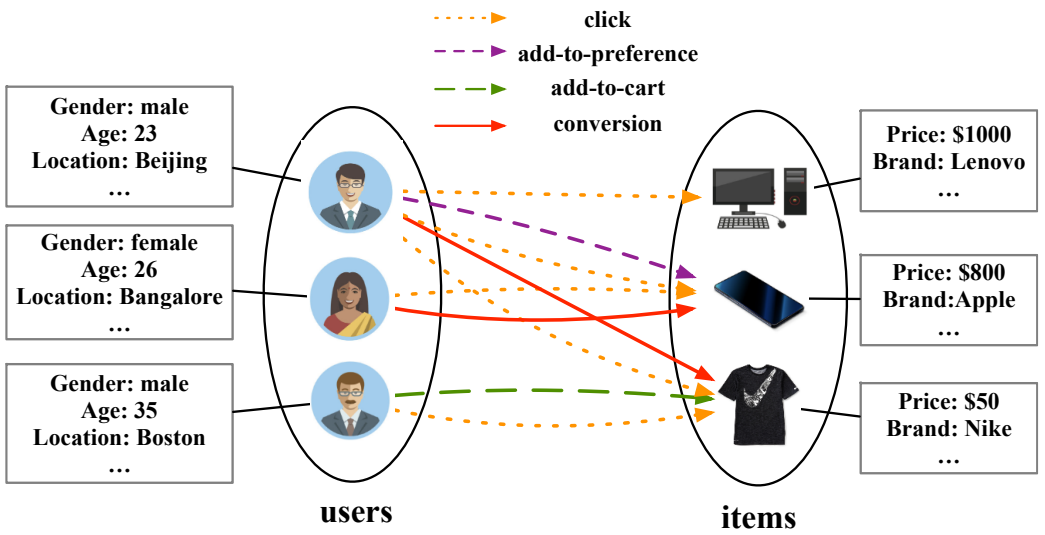
Summary of related network embedding methods

Network Type	Method	Heterogeneity		Attribute
		Node Type	Edge Type	
Homogeneous Network (HON)	DeepWalk [27] LINE [34] node2vec [10] NetMF [28]	Single	Single	/
Attributed Homogeneous Network (AHON)	TADW [41] LANE [16] AANE [15] SNE [20] DANE [9] ANRL [44]	Single	Single	Attributed
Heterogeneous Network (HEN)	PTE [33] metapath2vec [7] HERec [30]	Multi	Single	/
Attributed HEN (AHEN)	HNE [3]	Multi	Single	Attributed
Multiplex Heterogeneous Network (MHEN)	PMNE [22] MVE [29] MNE [43] mvn2vec [31]	Single	Multi	/
	GATNE-T	Multi	Multi	
Attributed MHEN (AMHEN)	*GATNE*-I	Multi	Multi	Attributed

Multiplex Heterogeneous Graph Embedding

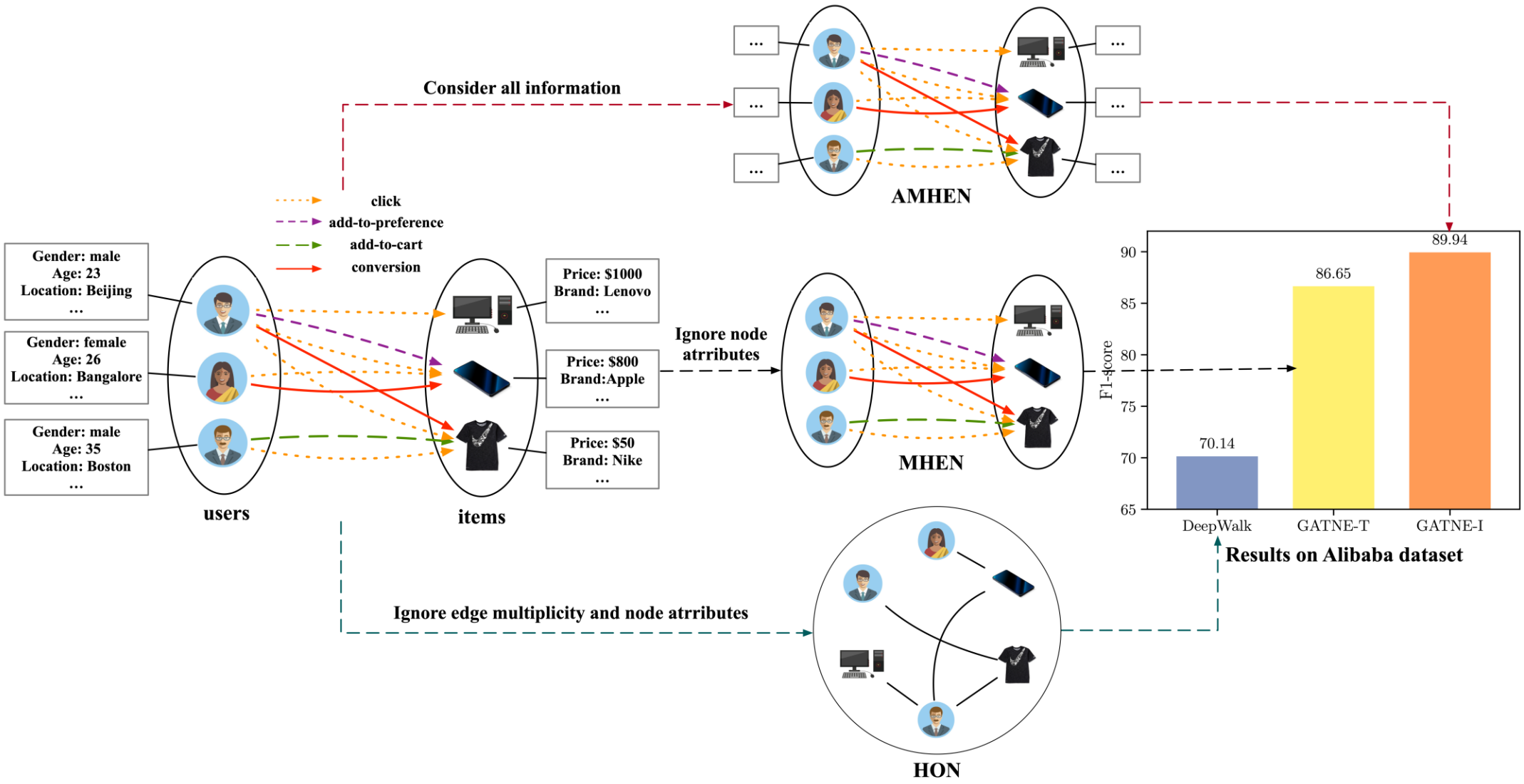
- Input: a multiplex heterogeneous graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{R}$ representing the set of edge types
- Output: vertex embeddings for each edge type

$$\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(|\mathcal{R}|)} \in \mathbb{R}^{|\mathcal{V}| \times d}, d \ll |\mathcal{V}|$$



$\mathcal{R}$ = {click, add-to-preference, add-to-cart, conversion}

The GATNE Framework



Transductive Model: GATNE-T

- In GATNE-T, overall embedding of node v_i on edge type r is split into two parts: base embedding and edge embedding:

$$\mathbf{v}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r},$$

- Base embedding is shared between different edge types
- Edge embedding on edge type r is aggregated from neighbors' edge embedding and then integrated by self-attention mechanism

$$\mathbf{u}_{i,r}^{(k)} = aggregator(\{\mathbf{u}_{j,r}^{(k-1)}, \forall v_j \in \mathcal{N}_{i,r}\}),$$

$$\mathbf{U}_i = (\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,m})$$

$$\mathbf{a}_{i,r} = \text{softmax}(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{U}_i))^T$$

Inductive Model: GATNE-I

- In order to handle unobserved data, GATNE-I is proposed.
- The base embedding and the initial edge embedding are parameterized as the function of node's attributes:

$$\mathbf{b}_i = \mathbf{h}_z(\mathbf{x}_i) \quad \mathbf{u}_{i,r}^{(0)} = \mathbf{g}_{z,r}(\mathbf{x}_i)$$

- And an extra attribute term is added to the overall embedding of node v_i on edge type r:

$$\mathbf{v}_{i,r} = \mathbf{h}_z(\mathbf{x}_i) + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} + \beta_r \mathbf{D}_z^T \mathbf{x}_i$$

Model Training

- GATNE-T:

$$\mathbf{v}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r},$$

- GATNE-I:

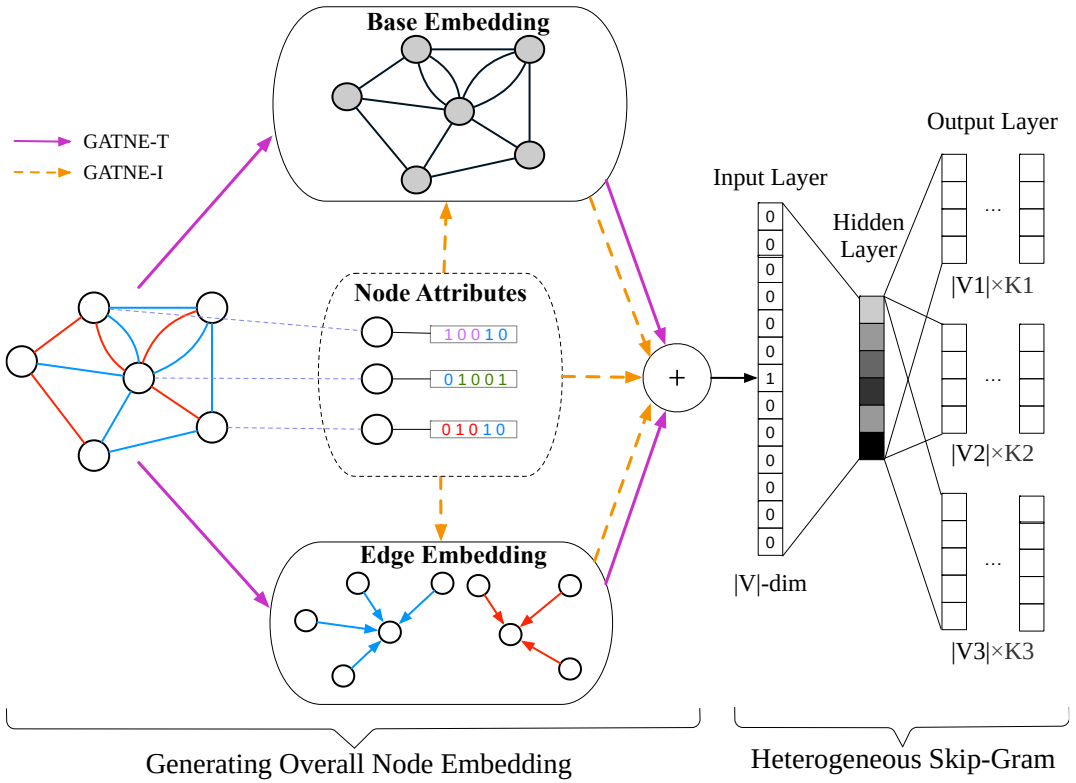
$$\mathbf{v}_{i,r} = \mathbf{h}_z(\mathbf{x}_i) + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} + \beta_r \mathbf{D}_z^T \mathbf{x}_i$$

- Meta-path-based random walk:

$$p(v_j|v_i, \mathcal{T}) = \begin{cases} \frac{1}{|\mathcal{N}_{i,r} \cap \mathcal{V}_{t+1}|} & (v_i, v_j) \in \mathcal{E}_r, v_j \in \mathcal{V}_{t+1}, \\ 0 & (v_i, v_j) \in \mathcal{E}_r, v_j \notin \mathcal{V}_{t+1}, \\ 0 & (v_i, v_j) \notin \mathcal{E}_r, \end{cases}$$

- Heterogeneous negative sampling:

$$E = -\log \sigma(\mathbf{c}_j^T \cdot \mathbf{v}_{i,r}) - \sum_{l=1}^{L} \mathbb{E}_{v_k \sim P_t(v)} [\log \sigma(-\mathbf{c}_k^T \cdot \mathbf{v}_{i,r})]$$



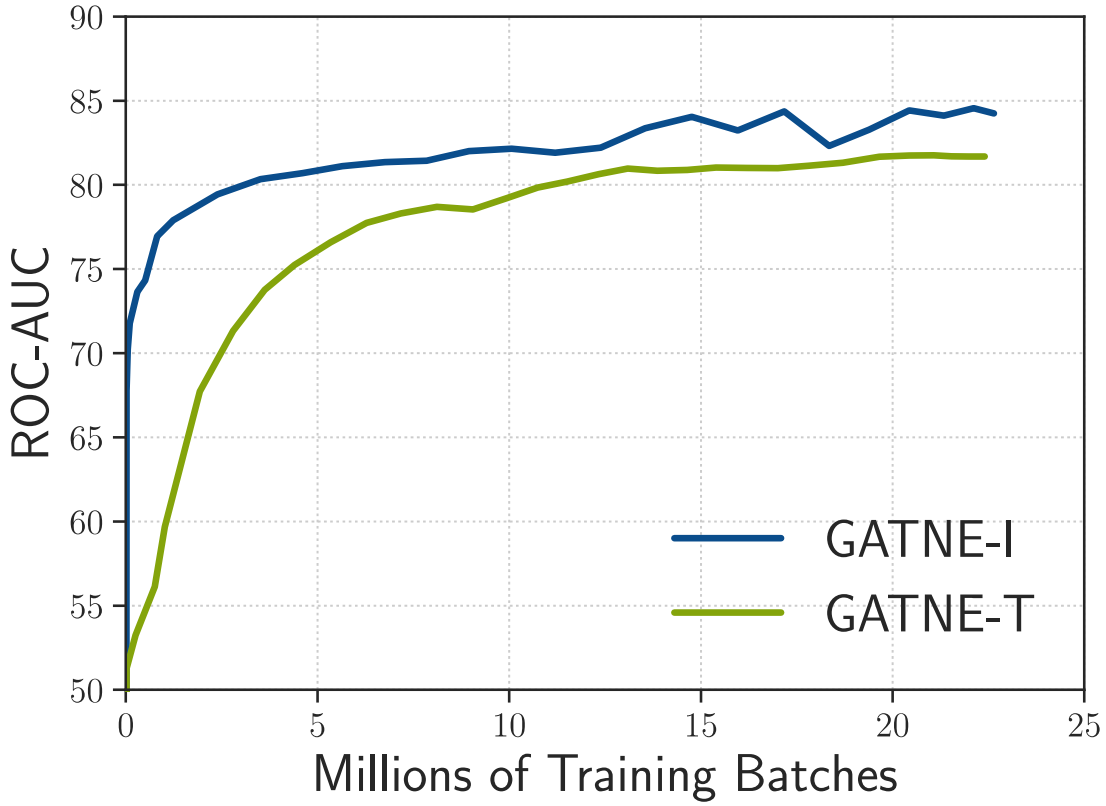
Link Prediction Task

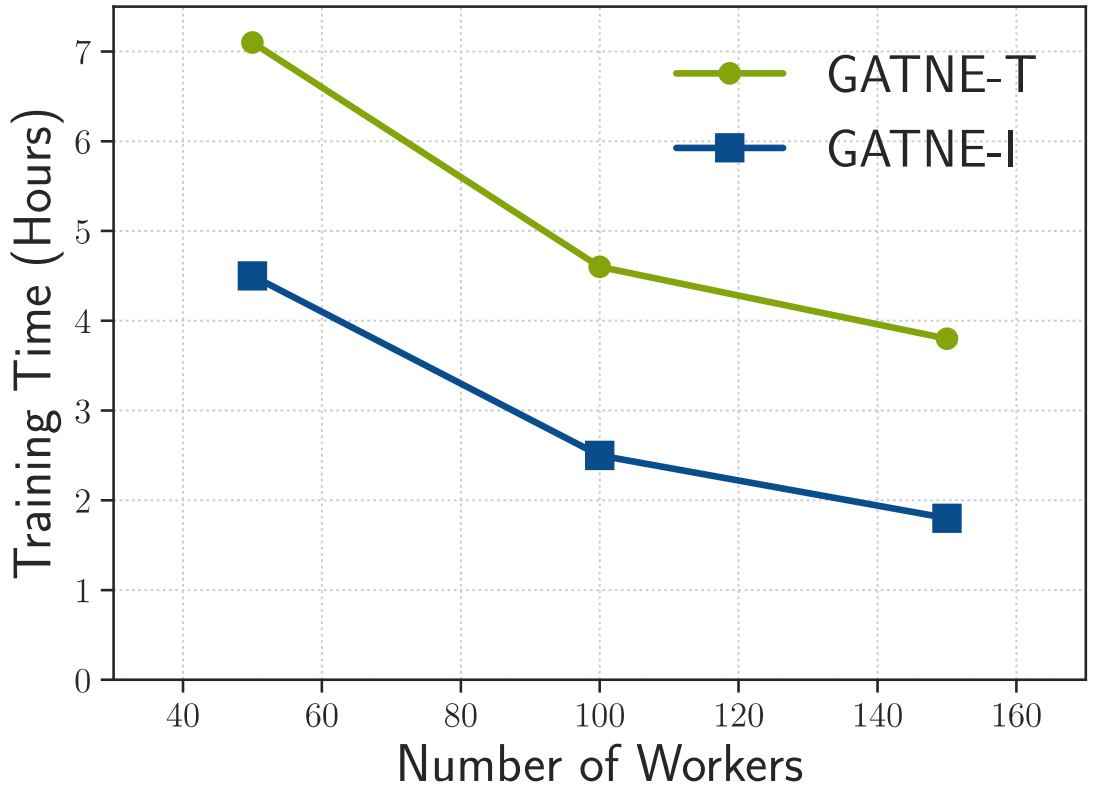
Table 4: Performance comparison of different methods on four datasets.

	Amazon			YouTube			Twitter			Alibaba-S		
	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1
DeepWalk	94.20	94.03	87.38	71.11	70.04	65.52	69.42	72.58	62.68	59.39	60.62	56.10
node2vec	94.47	94.30	87.88	71.21	70.32	65.36	69.90	73.04	63.12	62.26	63.40	58.49
LINE	81.45	74.97	76.35	64.24	63.25	62.35	62.29	60.88	58.18	53.97	54.65	52.85
metapath2vec	94.15	94.01	87.48	70.98	70.02	65.34	69.35	72.61	62.70	60.94	61.40	58.25
ANRL	95.41	94.19	89.60	75.93	73.21	70.65	70.04	67.16	64.69	64.76	61.67	61.37
PMNE(n)	95.59	95.48	89.37	65.06	63.59	60.85	69.48	72.66	62.88	62.23	63.35	58.74
PMNE(r)	88.38	88.56	79.67	70.61	69.82	65.39	62.91	67.85	56.13	55.29	57.49	53.65
PMNE(c)	93.55	93.46	86.42	68.63	68.22	63.54	67.04	70.23	60.84	51.57	51.78	51.44
MVE	92.98	93.05	87.80	70.39	70.10	65.10	72.62	73.47	67.04	60.24	60.51	57.08
MNE	90.28	91.74	83.25	82.30	82.18	75.03	91.37	91.65	84.32	62.79	63.82	58.74
GATNE-T	**97.44**	**97.05**	**92.87**	**84.61**	81.93	**76.83**	**92.30**	91.77	**84.96**	66.71	67.55	62.48
GATNE-I	96.25	94.77	91.36	84.47	**82.32**	**76.83**	92.04	**91.95**	84.38	**70.87**	**71.65**	**65.54**

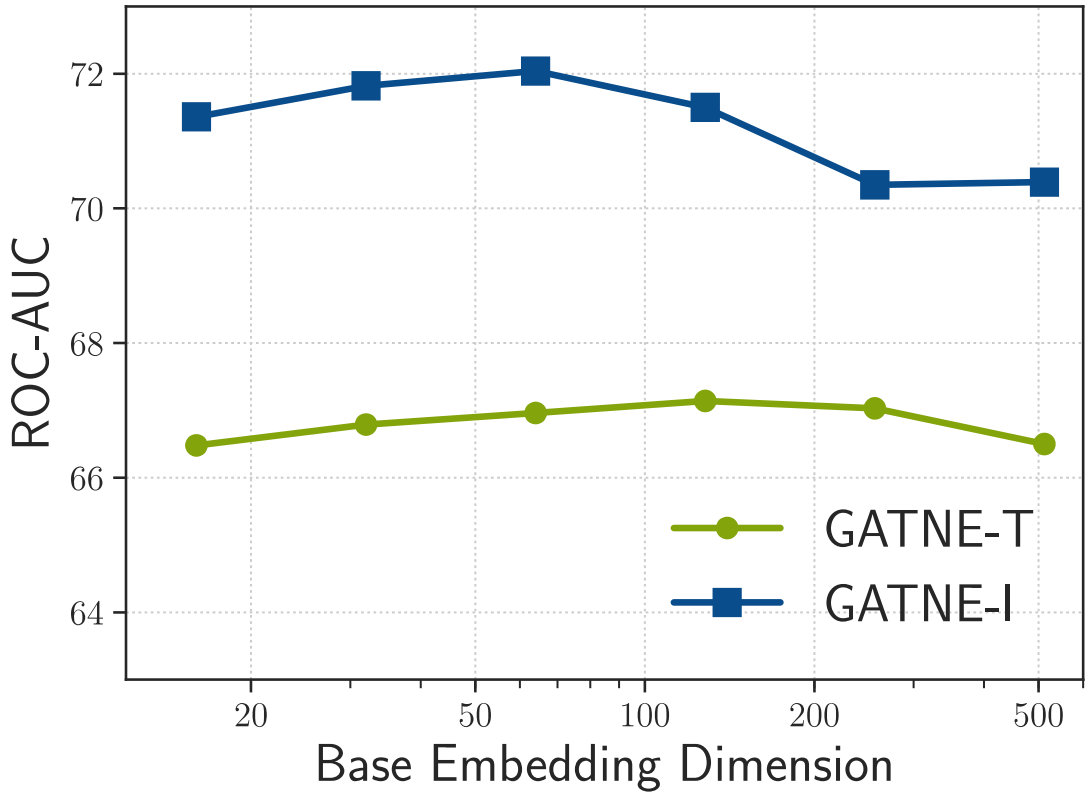
GATNE outperforms all sorts of baselines in the various datasets.

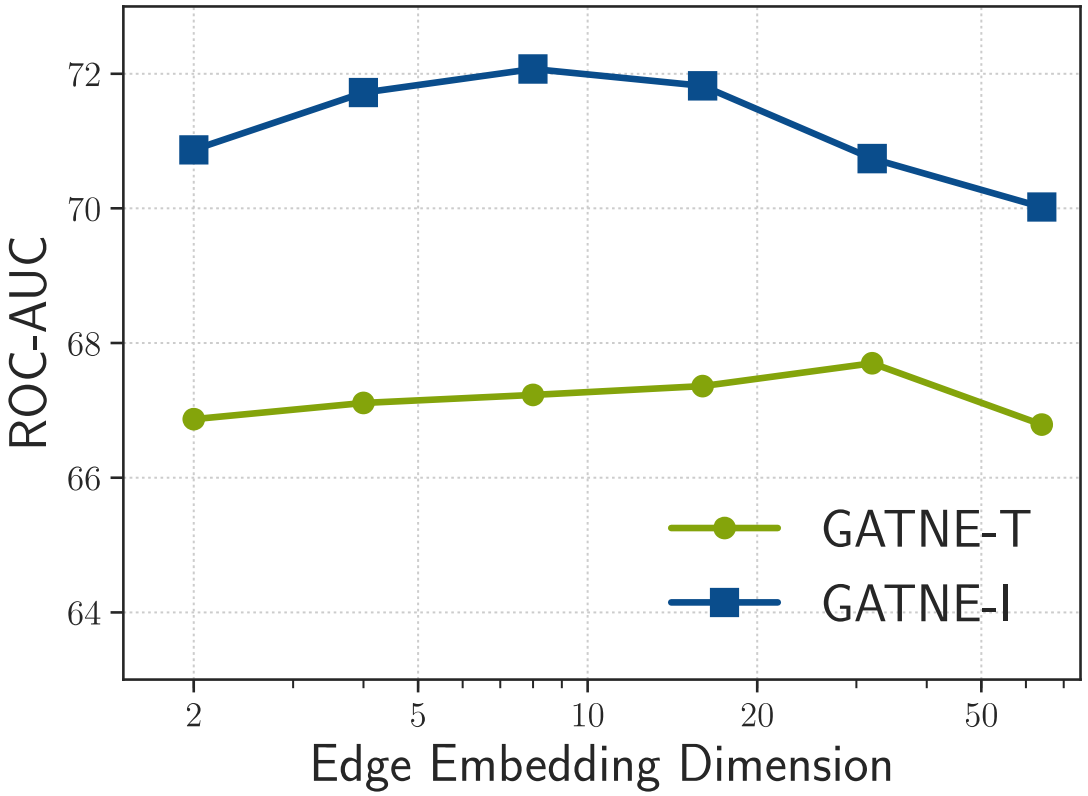
Convergence & Scalability Analysis





Parameter Analysis





Alibaba A/B Tests

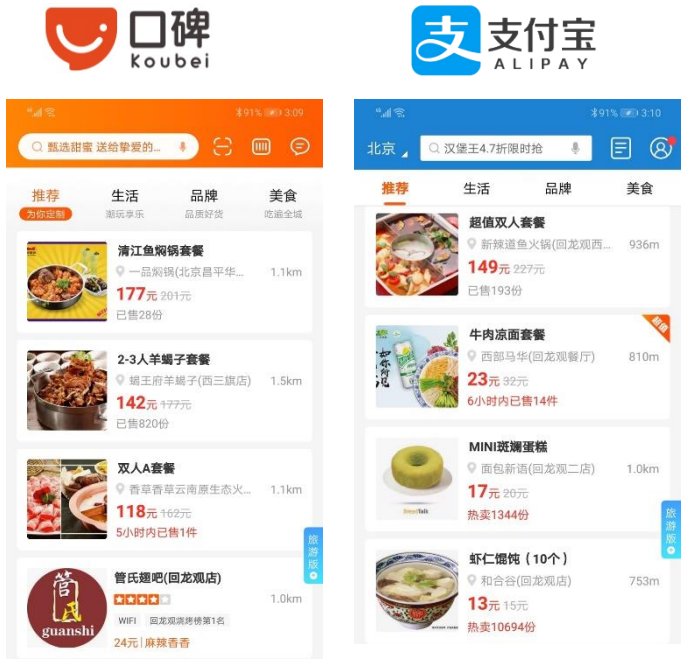
- GATNE-I is deployed on Alibaba's distributed cloud platform for its recommendation system. The training dataset has about 100 million users and 10 million items with 10 billion interactions between them.

- Under the framework of A/B tests, an offline test is conducted on GATNE-I, MNE and DeepWalk. The experimental goal is to maximize Hit-Rate. The results demonstrate that GATNE-I improves Hit-Rate by 3.26% and 24.26% compared to MNE and DeepWalk respectively.

Network Embedding for Online-to-Offline Recommendation

Ding et al. Infer Implicit Contexts in Real-time Online-to-Offline Recommendation. KDD'19

Context-Aware Online-to-Offline Real-time Recommendation



Restaurants

Hair salon

SPA

Online Purchase on mobile devices
Deals, coupons, etc.

Offline Consumption
Scan QR code in the store to complete transaction.

Key Challenges

Context-Aware Online-to-Offline Real-time Recommendation

1. User’s needs vary dynamically in different contexts
2. Explicit context information is very limited (date, time, and location)
3. Implicit context (user’s purpose, status, etc.) is difficult to observe directly

Key Challenges

8:00 am | **1:00 pm** | **4:00 pm** | **7:00 pm**

Scene
Breakfast nearby alone
$-$$

Scene
Lunch at work
Group of people
Dine-in
$$

Scene
Afternoon tea
Quite
Good for talking

Scene
Dinner
Recreation
Family or kids,
$$-$$$

Modeling Implicit Context

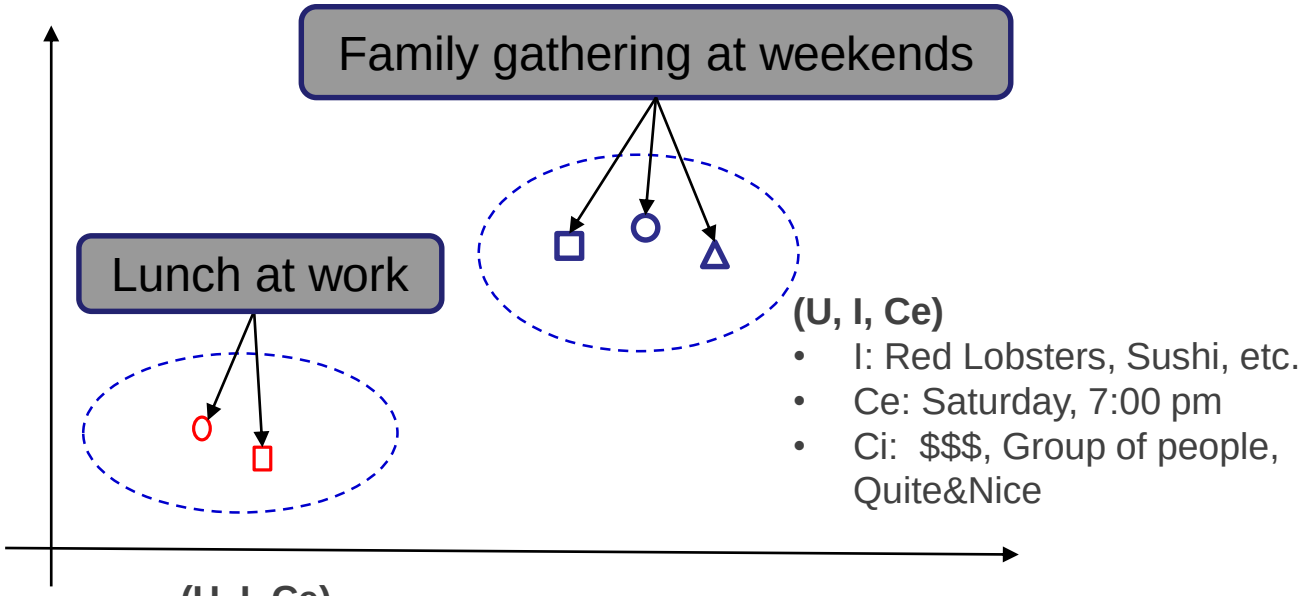
- Approach 1
  - Enumerate possible combination of contextual factors (e.g. date, time, location, people, etc.)
  - Not feasible when the factor number is too large.

Modeling Implicit Context

- Approach 1
  - Enumerate possible combination of contextual factors (e.g. date, time, location, people, etc.)
  - Not feasible when the factor number is too large.
- Approach 2
  - Latent representation of implicit contexts [X1, X2, X3,…];
  - For example, the attributes may include:
    - X1: User status (Alone/With Group of People)
    - X2: User's purposes: Finding Restaurants/Deals/Coupons/etc.
    - X3: Price Preference (low/medium/high)
    - X4: Distance Preference (nearby, at home, at work,…)
    - X5: Environment Preference (quite, fancy, etc.)

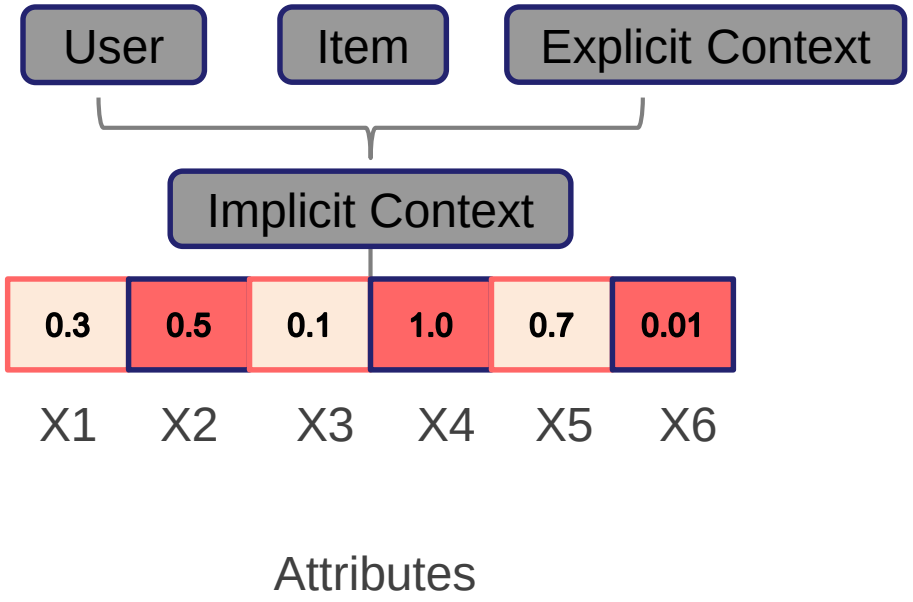
Modeling Implicit Context

Latent representation of Implicit Context
From interaction of <User, Item, Explicit Context>



(U, I, Ce)
- I: Red Lobsters, Sushi, etc.
- Ce: Saturday, 7:00 pm
- Ci: $$$, Group of people, Quite&Nice

(U, I, Ce)
- I: Pizza, Burrito, Burgers
- Ce: Weekday, 1:00 pm
- Ci: $, Alone, Nearby

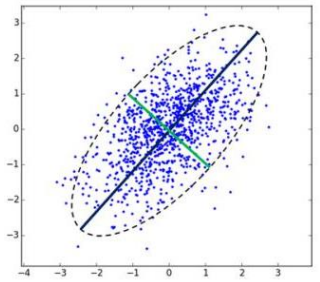


Attributes

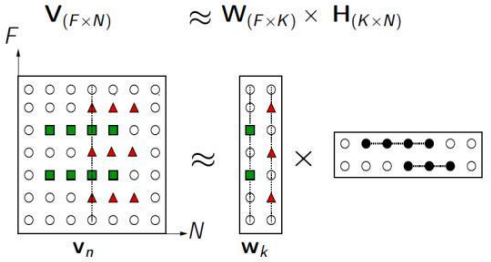
Modeling Implicit Context(Cont.)

- Latent representation modeling methods
  - Latent factor models:
    - Matrix/Tensor factorization based: SVD, PCA, NMF.
    - Probabilistic graphical models based: LDA, pLSA, etc.
  - Deep learning models:
    - AutoEncoder, Denoise Autoencoder, etc.
    - Variational Autoencoder;

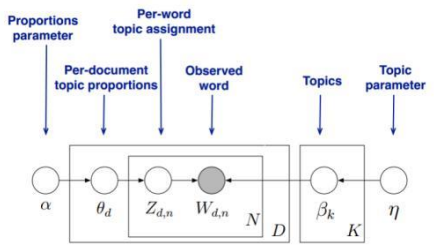
PCA



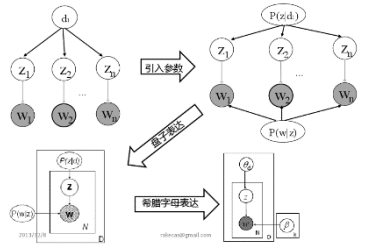
NMF



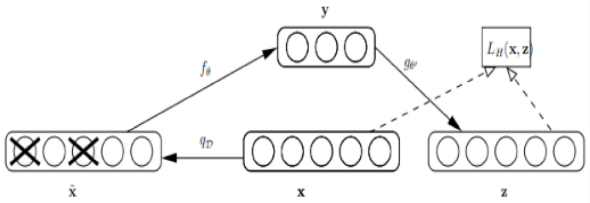
LDA



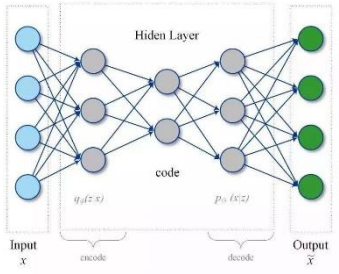
pLSA



DAE



VAE



Problem Definition in O2O Recommendation

- Two-stage recommendation framework
  - Step 1: Infer implicit contexts from interaction of tuple <User, Item, Explicit Context>

$$C_i = \sum_K \mu_k C_{ik}$$

$$C_{ik} = g_k(U, I, C_e)$$

gk(.) can be instantiated by many generative models during pre-training, e.g. DAE, VAE, etc.

  - Step 2: Include the implicit contexts representation in an end-to-end model, e.g. Wide&Deep, DeepFM, etc.

$$y_{ui} = f(U, I, C_e, C_i)$$

Multi-Head DAE/VAE

- Multi-Head structure to represent multiple components of implicit multi-contexts
  - Multi-Head DAE (Denoise AutoEncoder)
    - Hidden state representation h is concatenation of K heads h_k;

$$h = h_1 \oplus h_2 \oplus \ldots \oplus h_K,\ k \in K \tag{21}$$

$$h_k = \sigma(W_k \tilde{x} + b_k),\ k \in K \tag{22}$$

$$x^{'} = \sigma(W^{'} h + b^{'}) \tag{23}$$

$$L_{reconstruct} = \frac{1}{N} \sum_{N} ||x - x^{'}||^2 \tag{24}$$

  - Multi-Head VAE (Variational AutoEncoder)
    - Hidden State z is concatenation of K normally distributed vector z_k;

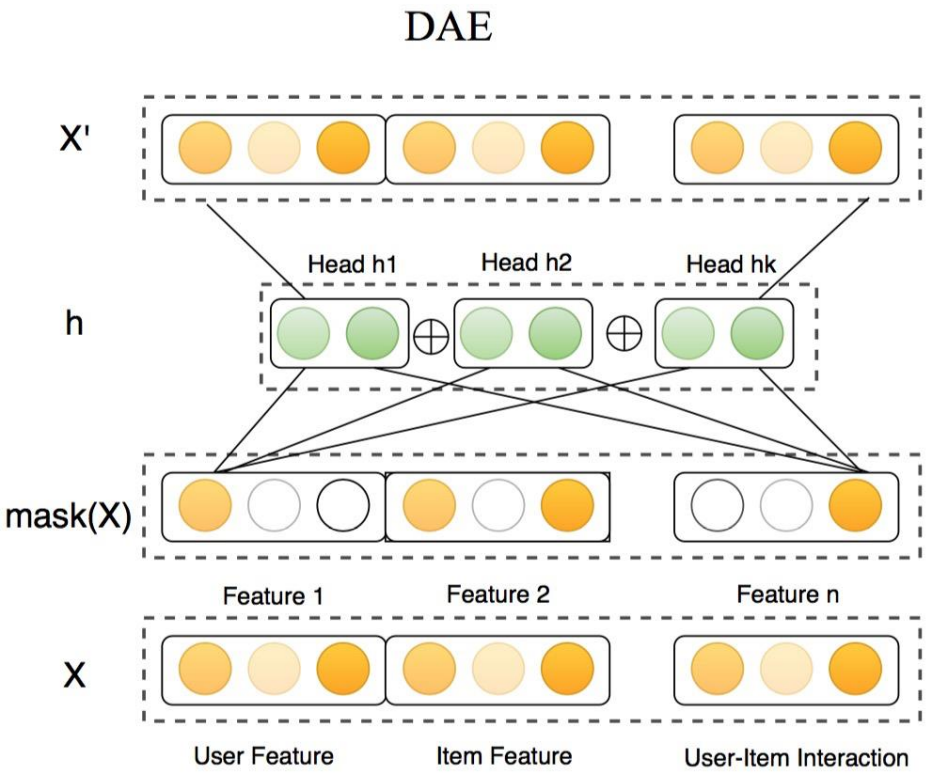
$$z = z_1 \oplus z_2 \oplus \ldots \oplus z_K,\ k \in K \tag{25}$$

$$z_k \sim N(\mu_k(x), \Sigma_k(x)),\ k \in K \tag{26}$$

$$x^{'} = \sigma(W^{'} z + b^{'}) \tag{27}$$

Multi-Head DAE/VAE

- Multi-Head structure to represent multiple components of implicit multi-contexts
  - Multi-Head DAE (Denoise AutoEncoder)
    - Hidden state representation h is concatenation of K heads h_k;



$$h = h_1 \oplus h_2 \oplus \ldots \oplus h_K,\ k \in K \tag{21}$$

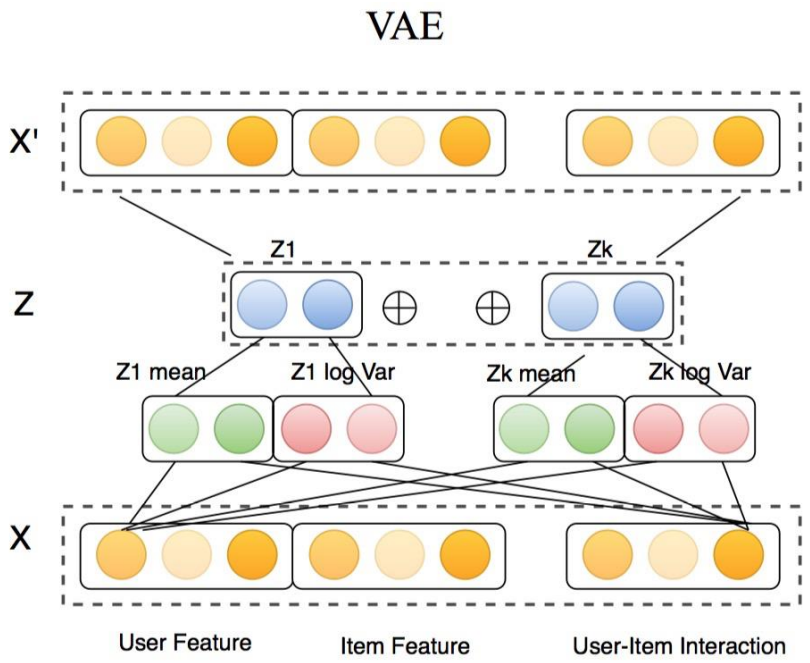
$$h_k = \sigma(W_k \tilde{x} + b_k),\ k \in K \tag{22}$$

$$x^{'} = \sigma(W^{'} h + b^{'}) \tag{23}$$

$$L_{reconstruct} = \frac{1}{N} \sum_{N} ||x - x^{'}||^2 \tag{24}$$

Multi-Head DAE/VAE

- Multi-Head structure to represent multiple components of implicit multi-contexts
  - Multi-Head VAE (Variational AutoEncoder)
    - Hidden State z is concatenation of K normally distributed vector z_k;



$$z = z_1 \oplus z_2 \oplus \ldots \oplus z_K,\ k \in K \tag{25}$$

$$z_k \sim N(\mu_k(x), \Sigma_k(x)),\ k \in K \tag{26}$$

$$x^{'} = \sigma(W^{'} z + b^{'}) \tag{27}$$

MACDAE: Mixture Attentional Constrained DAE

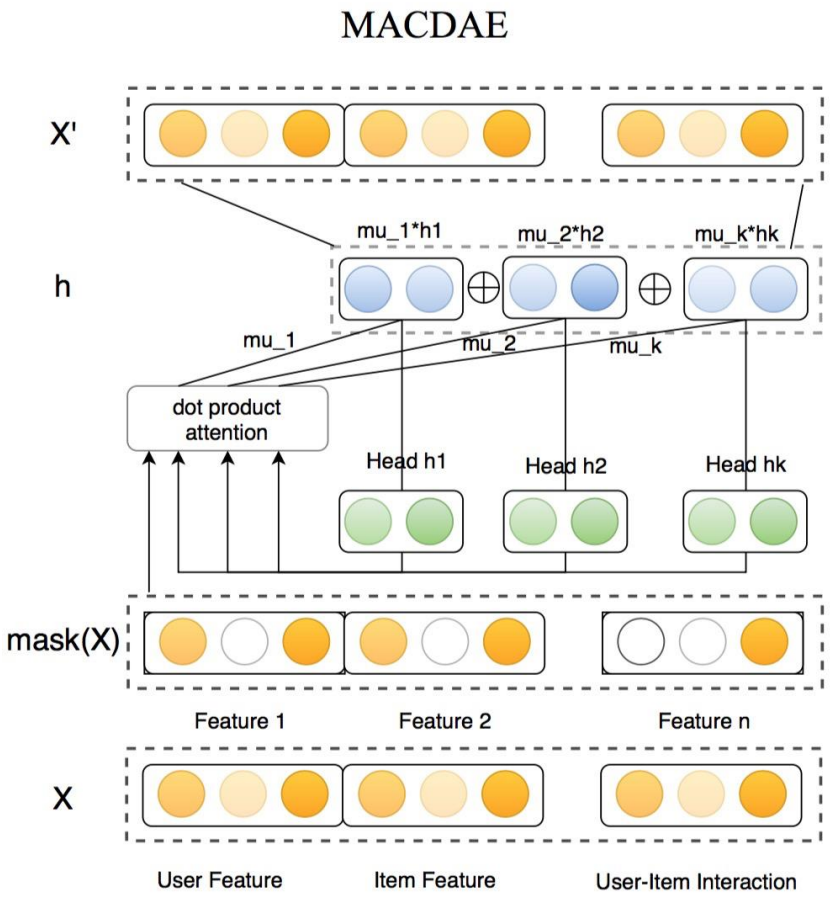
Different implicit components contribute differently to the representation…

- MACDAE
  - Multi-contexts representation h is the concatenation of weighted heads h_k;
  - Attention (Q,K,V): Query is the input X transformed to $W_a X$;
  - Keys, Values are the same: K heads h_k, packed into H;

$$h = \mu_1 h_1 \oplus \mu_2 h_2 \oplus \ldots \oplus \mu_K h_K,\ k \in K \quad (28)$$

$$Q = (W_a x)^T \quad (29)$$

$$[\mu_1, \mu_2, \ldots, \mu_K] = softmax(QH^T), k \in K \quad (30)$$



MACDAE: Mixture Attentional Constrained DAE

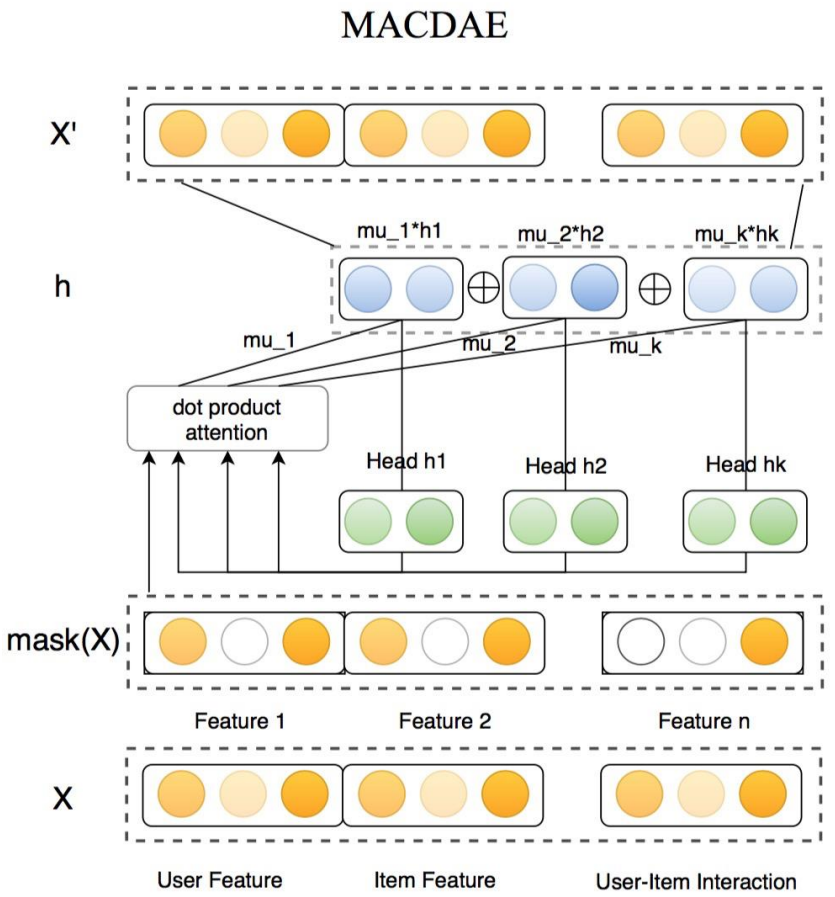
Different implicit components contribute differently to the representation…

- MACDAE
  - Multi-contexts representation h is the concatenation of weighted heads h_k;
  - Attention (Q,K,V): Query is the input X transformed to $W_a X$;
  - Keys, Values are the same: K heads h_k, packed into H;

$$h = \mu_1 h_1 \oplus \mu_2 h_2 \oplus \ldots \oplus \mu_K h_K,\ k \in K \quad (28)$$

$$Q = (W_a x)^T \quad (29)$$

$$[\mu_1, \mu_2, \ldots, \mu_K] = softmax(QH^T), k \in K \quad (30)$$



MACDAE: Mixture Attentional Constrained DAE

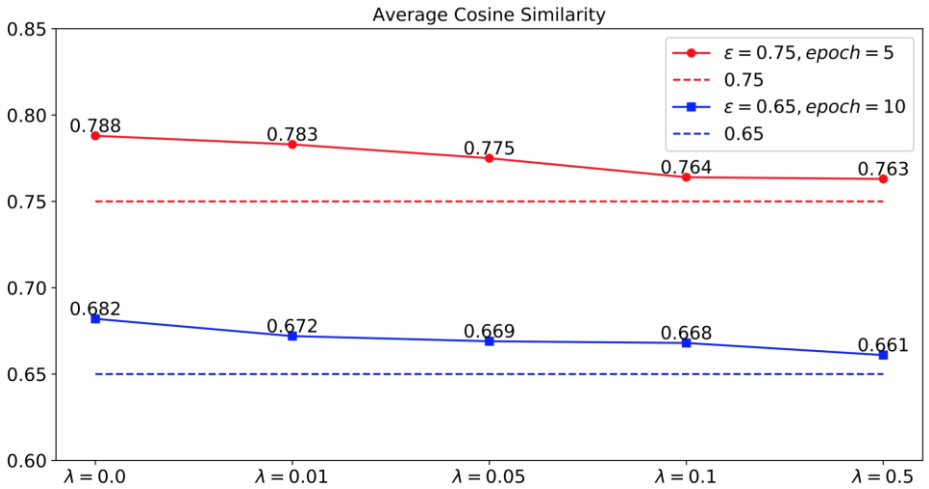
Different implicit components contribute differently to the representation…

- Constraints on multiple heads to avoid them converging to identical subspaces
  - Adding penalty cost of cosine similarity on multiple heads as regularization term;

$$min\ L = L_{reconstruct} \tag{42}$$

$$s.t.\ cos(h_i, h_j) - \varepsilon \leq 0, \forall i, j \in K \tag{43}$$

$$min\ L_{new} = L_{reconstruct} + \sum_{i,j \in K} \lambda(cos(h_i, h_j) - \varepsilon) \tag{44}$$



Average cosine similarity of each pair of heads, change with hyper-parameter of epsilon

Experiments and Analysis

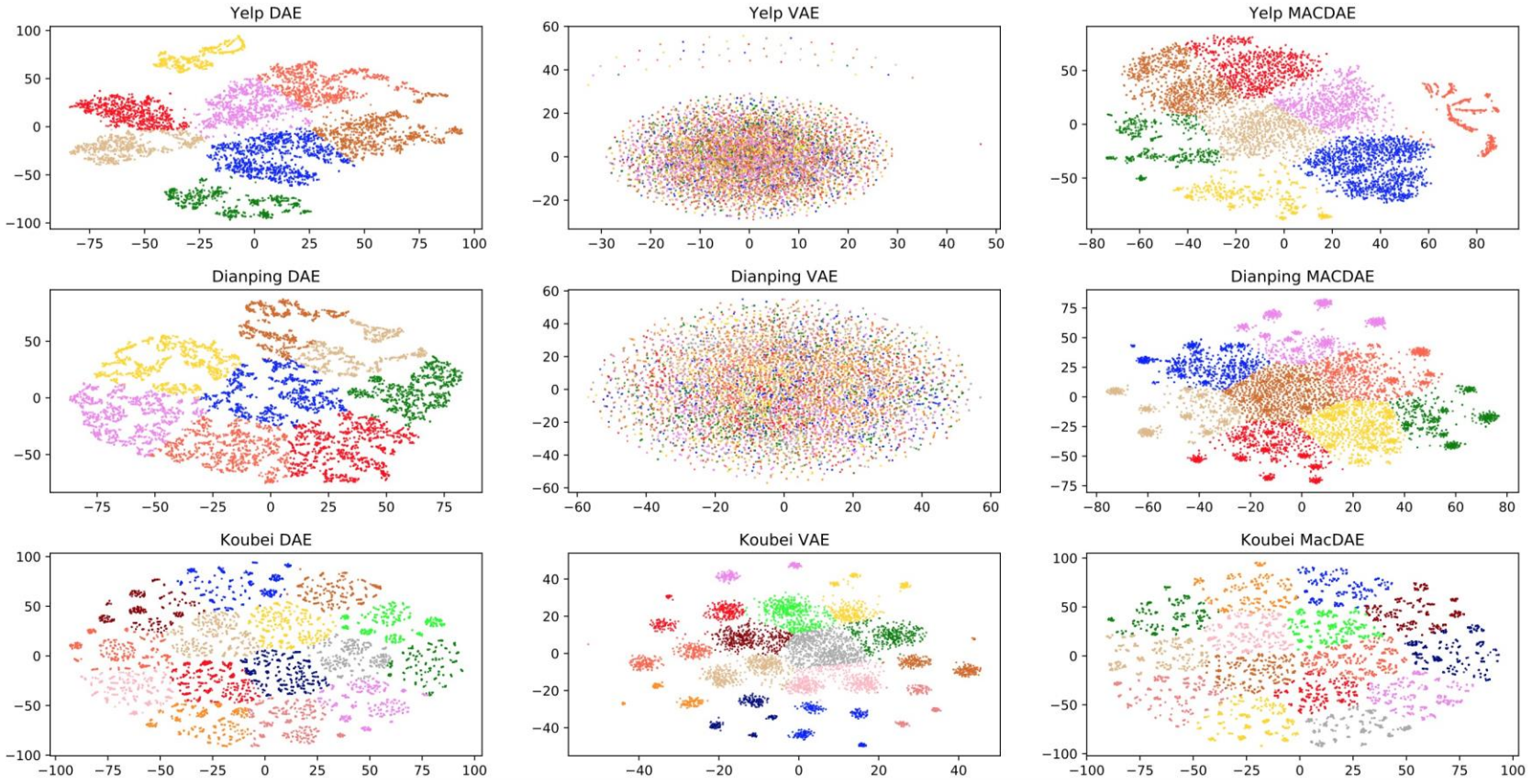
- Offline dataset evaluation
  - Yelp Business Review Dataset*;
  - Dianping Dataset*;
  - Koubei Dataset;
  - Evalution: NDCG@5/10, AUC on different K level;

Experiments and Analysis

Offline evaluation on three datasets

Model	Yelp		Dianping		Koubei
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	AUC
Base(Wide&Deep)	0.3774	0.4272	0.4301	0.4691	0.6754
DeepFM	#0.3893	#0.4414	#0.4564	0.4854	0.6660
NFM	0.3813	0.4242	0.4527	#0.4996	#0.6881
Base+DAE	0.4159	0.4639	0.4507	0.4954	0.6810
Base+VAE	0.4192	0.4661	0.4517	0.4985	0.6790
Base+MACDAE(K=4)	*0.4410	*0.4886	0.4569	0.5024	0.6902
Base+MACDAE(K=8)	0.4136	0.4650	*0.4614	*0.5050	0.6936
Base+MACDAE(K=16)	0.4058	0.4521	0.4470	0.4960	*0.6954

Experiments and Analysis



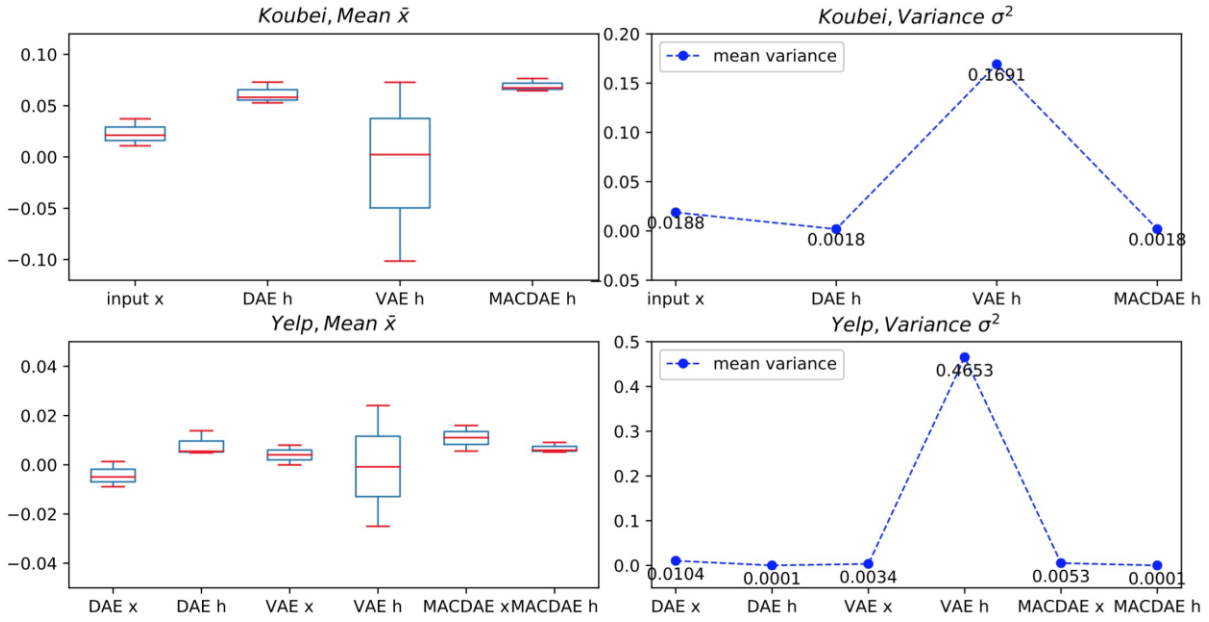
Visualization of latent space learned by DAE, VAE and MACDAE
on Yelp, Dianping and Koubei dataset.

Online A/B Test

- Koubei App's "Guess You Like" recommendation
  - Compare Base+MACDAE with baseline Wide&Deep model
  - Experiments last 7 days in production environment
  - Results: Click-Trough Rate(CTR): +2.9% lift;
  - Conversion Rate(CVR) : +5.6% lift;

Experiments and Analysis

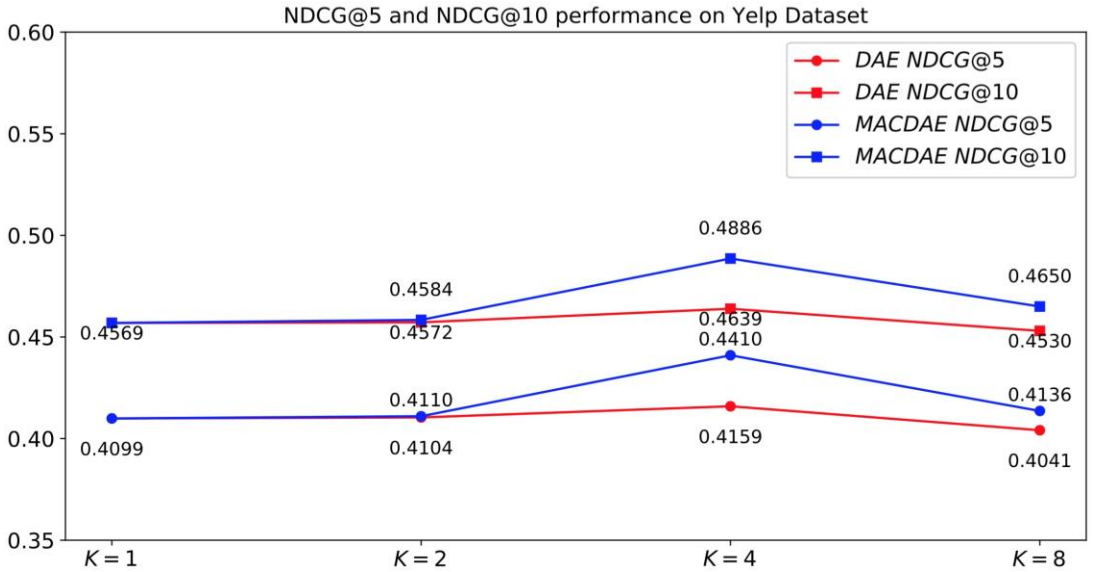
- Distribution of hidden state learned by different models
  - DAE/MACDAE: Increasing mean and reducing the variance…
  - VAE: Hidden state vector has high covariance, low mean;



The mean and variance distribution of the learned hidden states of Yelp and Koubei Dataset

Experiments and Analysis

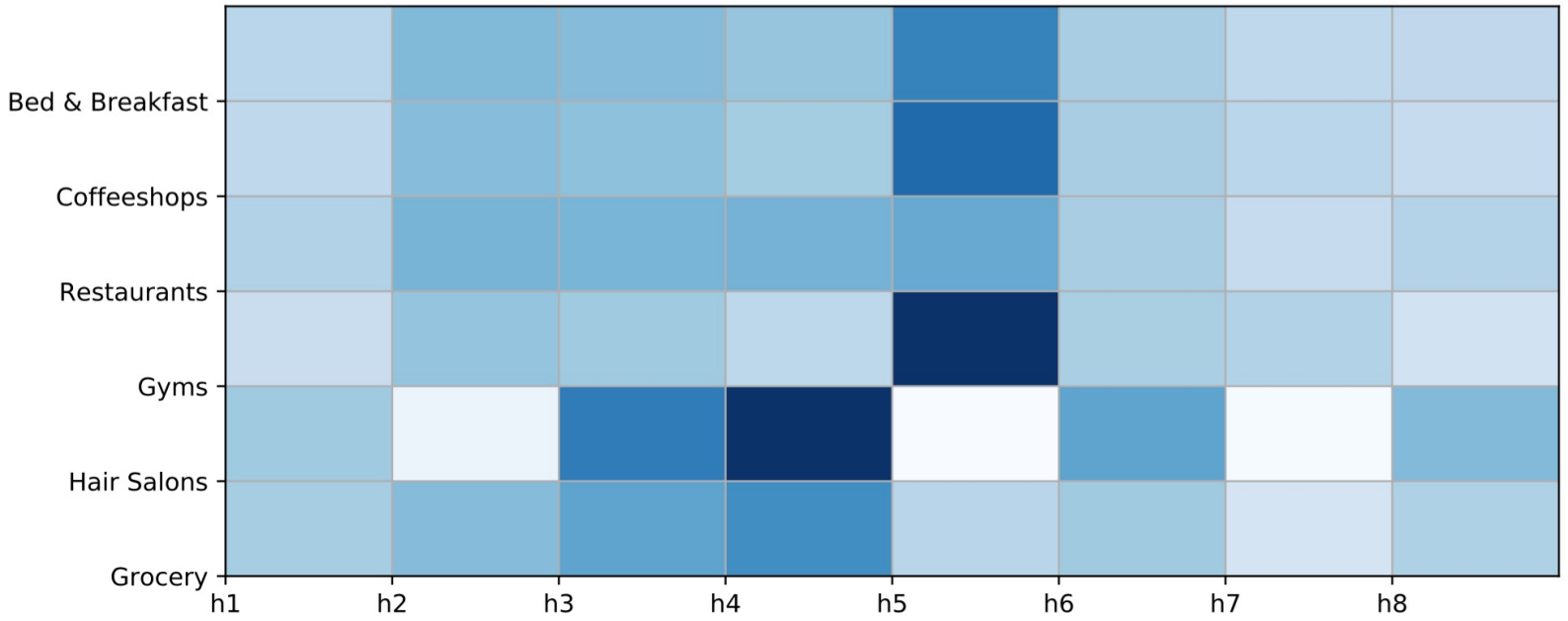
- Impact of multi-head number K
  - Compared different K level: [1, 2, 4, 8]
  - For Yelp dataset, K=4 achieves the best performance;
  - Increasing head number K does not always improve the metrics;



Impact of multi-head number K

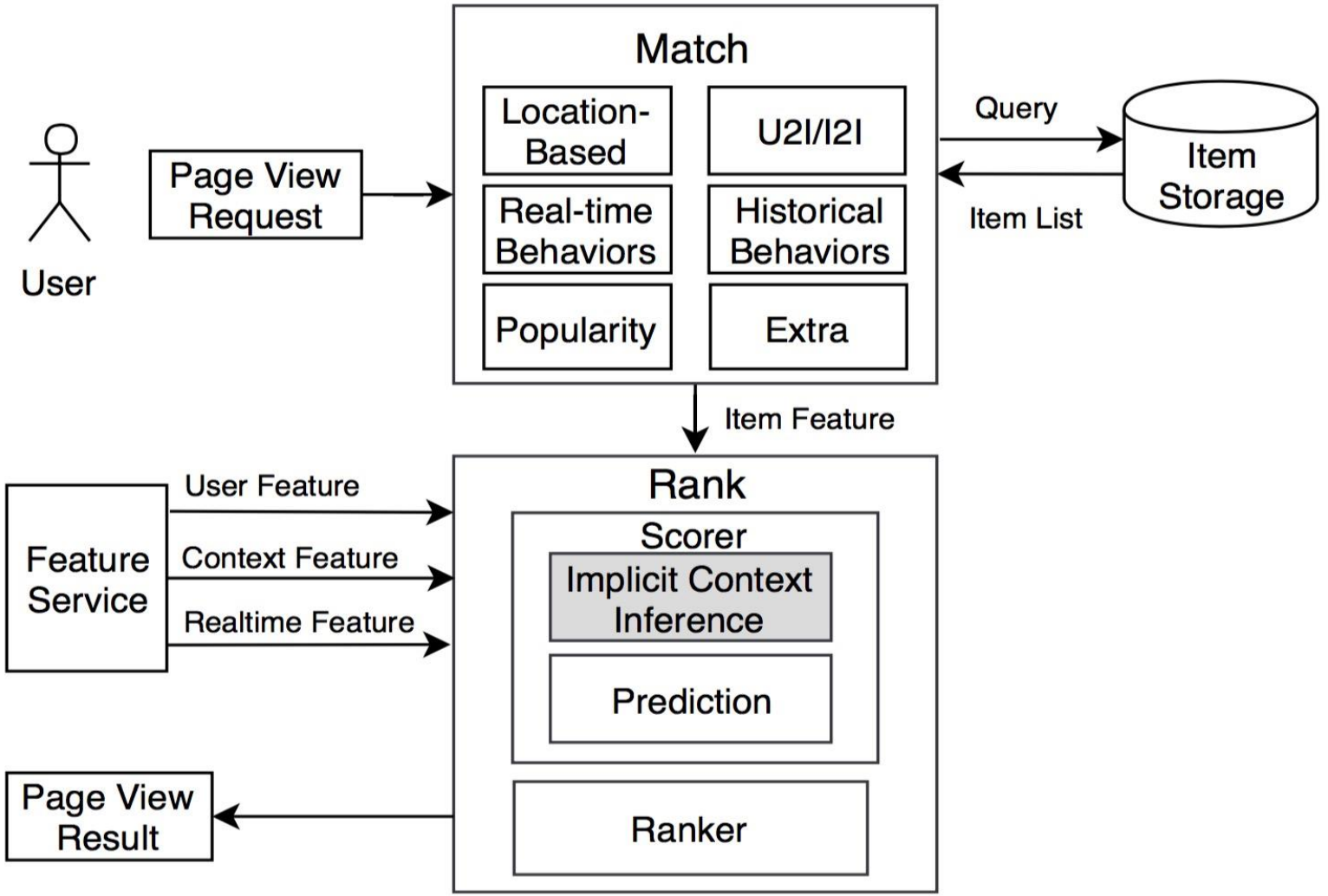
Experiments and Analysis

- Attentional weight illustration



Different shop category of Yelp dataset (K=8):
Different categories of shops have on each contextual components

Online Application System Architecture

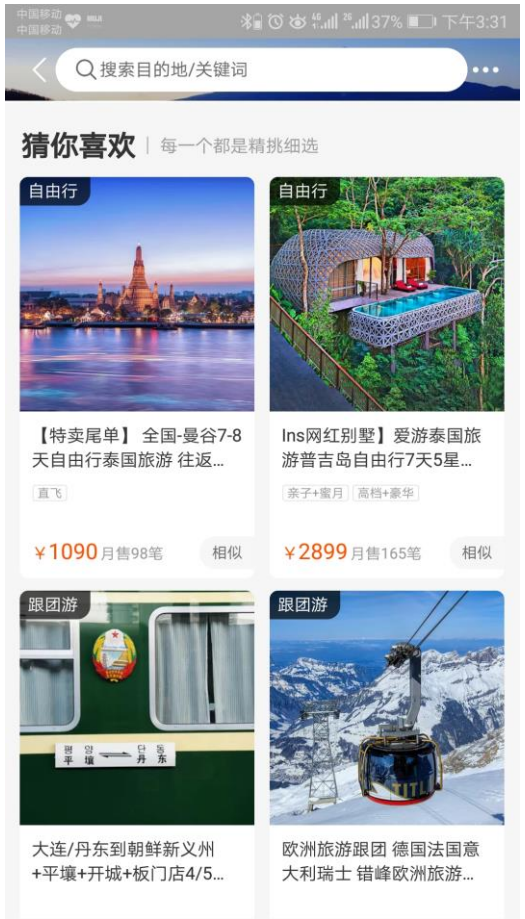


Meta Learning for Online Recommendation

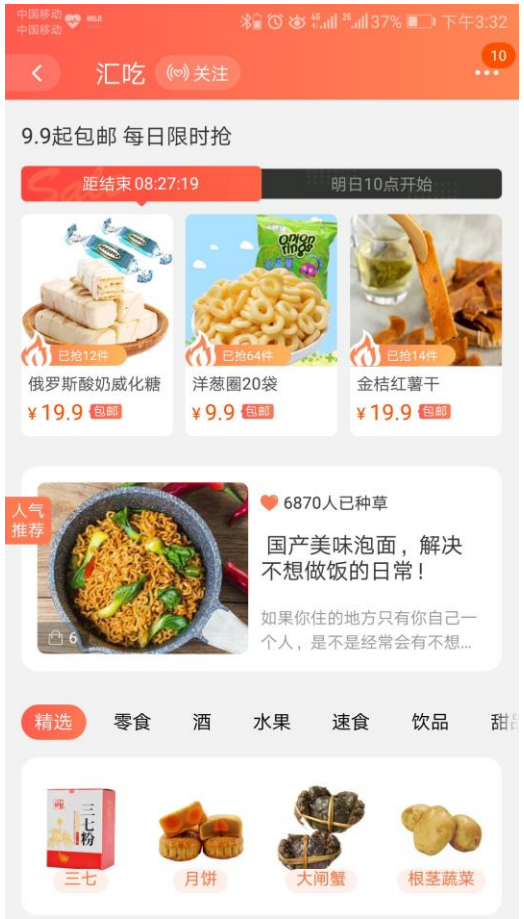
Du et al. Sequential Scenario-Specific Meta Learner for Online Recommendation. KDD'19

Scenarios in RS

Baby

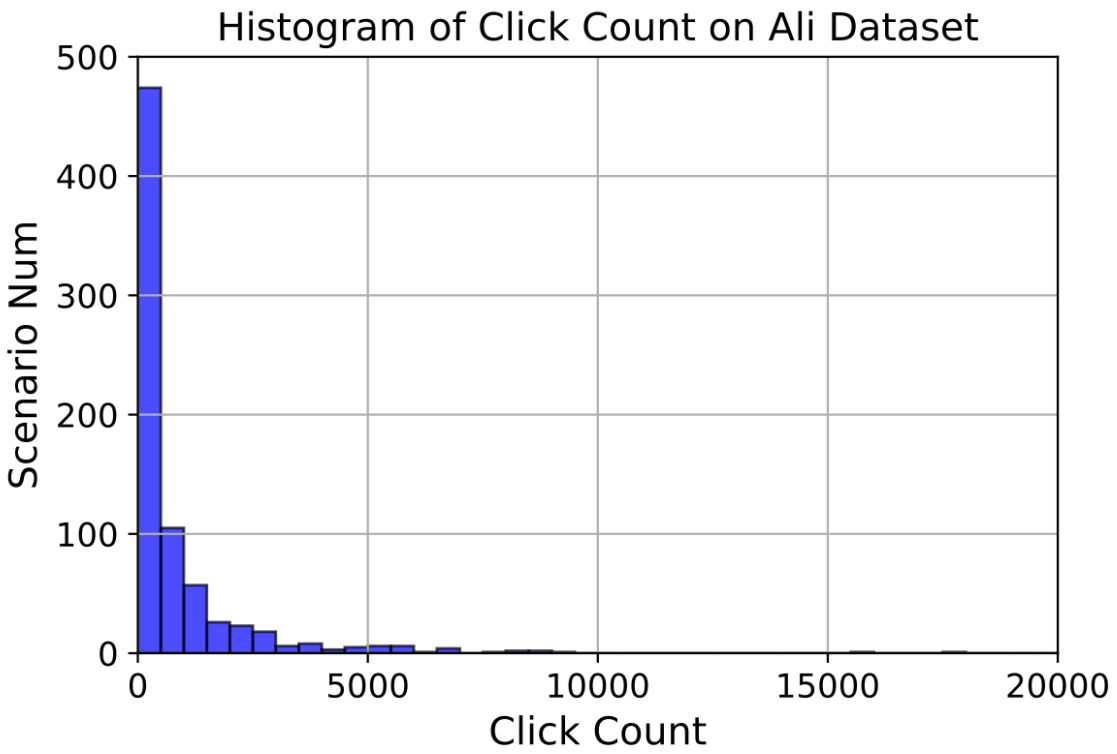


Travel



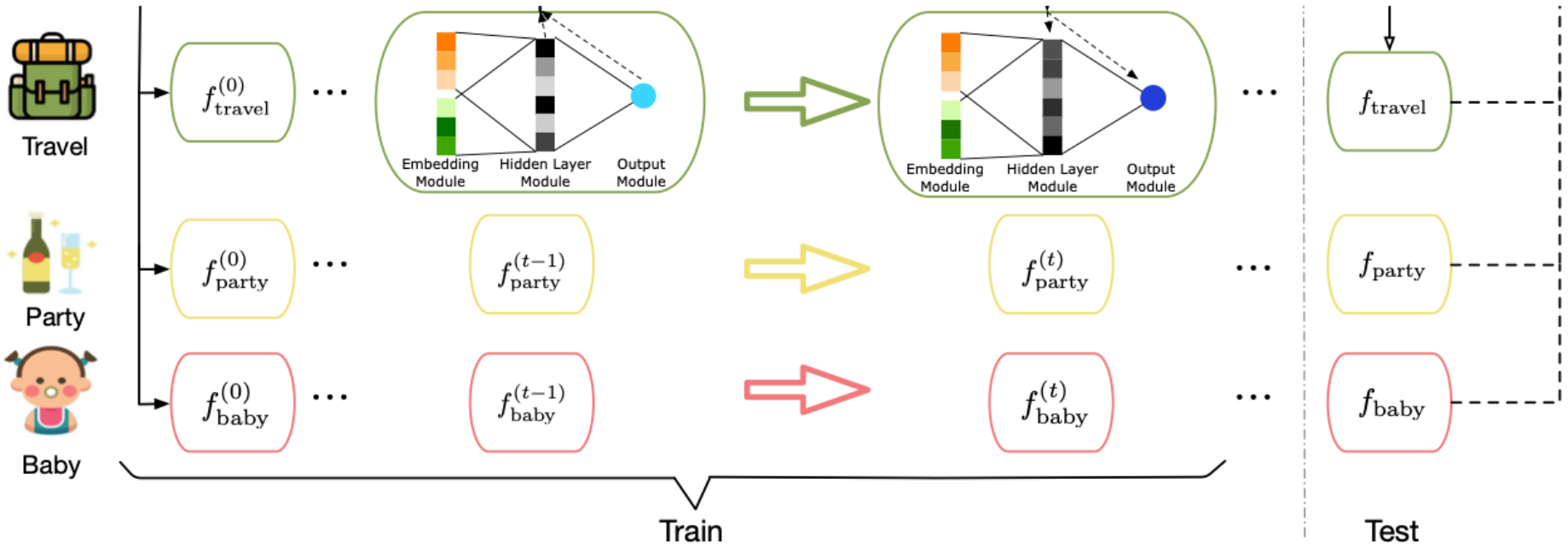
Food

Long-tailed Scenarios



Most scenarios in a recommender system are long-tailed, without enough user feedback

Two-level Learning



- **Scenario-specific learning:** parameters of recommenders are learned from a small amount of user feedback on specific scenario by the meta learner.
- **Meta learning:** the parameters of meta learner are learned across different scenarios and generalize well to unseen scenarios.

Learning as a Sequential Process

- At the beginning of scenario-specific learning, the recommender parameters are initialized as $\theta_c^{(0)}$.
- At each step t, a batch $B^{(t)} = \{u_k, i_k, i_k^-\}_{k=1}^N$ is sampled from D_c^{train}. Then the previous parameters is updated according to the loss on $B^{(t)}$.
- At each step t, the learning process terminates with predicted probability $p^{(t)}$.

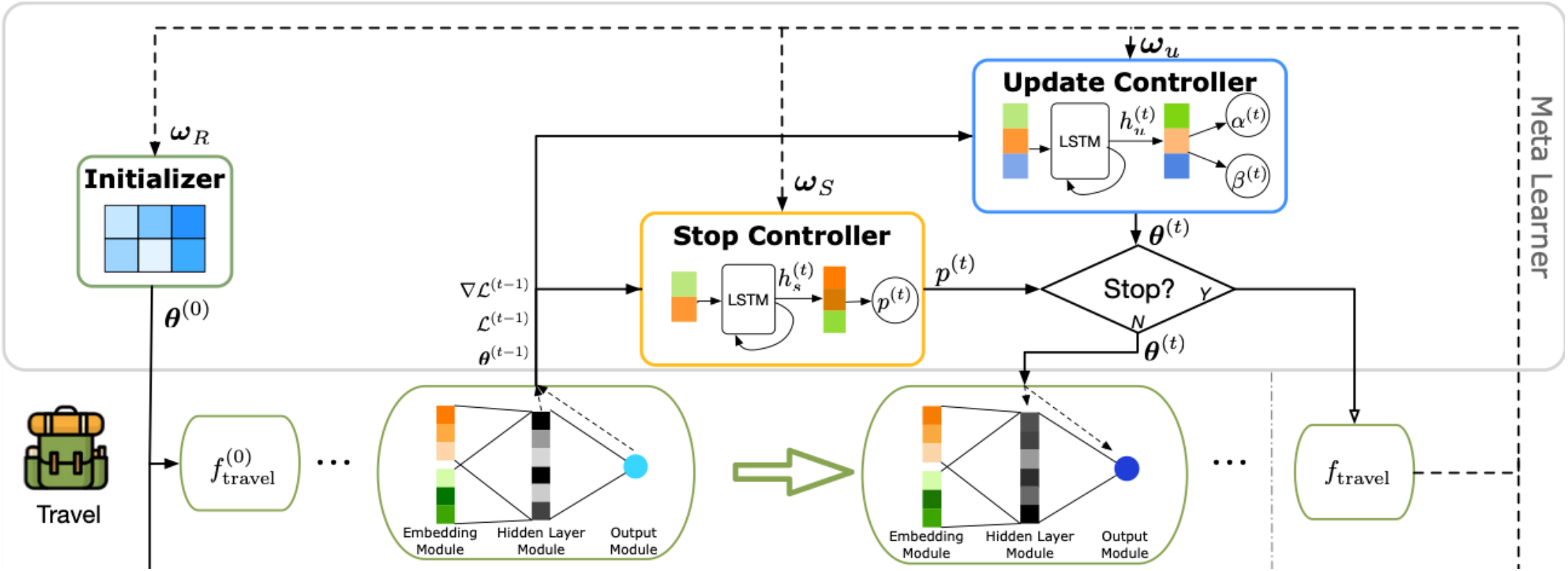
Algorithm

Algorithm 1 Training Algorithm of Meta Learner

Input: Meta-training set $\mathbb{T}_{\text{meta-train}}$, Loss function $\mathcal{L}$

$\boldsymbol{\omega}_u, \boldsymbol{\omega}_s, \boldsymbol{\omega}_R \leftarrow$ Random Initialization
for $d \leftarrow 1, K$ **do** ▷ K is the number of meta-training steps
  $D_c^{\text{train}}, D_c^{\text{test}} \leftarrow$ Random scenario from $\mathbb{T}_{\text{meta-train}}$
  $\boldsymbol{\theta}_c^{(0)} \leftarrow \boldsymbol{\omega}_R, T \leftarrow 0$
  for $t \leftarrow 1, T_{max}$ **do**
    $B^{(t)} \leftarrow$ Random batch from D_c^{train}
    $\mathcal{L}^{(t)} \leftarrow \mathcal{L}(B^{(t)}; \boldsymbol{\theta}_c^{(t-1)})$
    $p^{(t)} \leftarrow M_s(\mathcal{L}^{(t)}, ||\nabla_{\boldsymbol{\theta}_c^{(t-1)}} \mathcal{L}^{(t)}||_2; \boldsymbol{\omega}_s)$ ▷ Equation (9)
    $s^{(t)} \sim \text{Bernoulli}(p^{(t)})$ ▷ Randomly decide to stop or not
    if $s^{(t)} = 1$ **then**
      Break;
    end if
    $\boldsymbol{\theta}_c^{(t)} \leftarrow$ Update $\boldsymbol{\theta}_c^{(t-1)}$ according to Equations (7) and (8)
    $T \leftarrow T + 1$
  end for
  $\mathcal{L}^{\text{test}} \leftarrow \mathcal{L}(D_c^{\text{test}}; \boldsymbol{\theta}_c^{(T)})$
  Update $\boldsymbol{\omega}_u$, $\boldsymbol{\omega}_R$ using $\nabla_{\omega_u} \mathcal{L}^{\text{test}}$, $\nabla_{\omega_R} \mathcal{L}^{\text{test}}$
  $d\boldsymbol{\omega}_s \leftarrow 0$
  for $j \leftarrow 1, T$ **do**
    $d\boldsymbol{\omega}_s \leftarrow d\boldsymbol{\omega}_s + \left(\mathcal{L}^{\text{test}} - \mathcal{L}(D_c^{\text{test}}; \boldsymbol{\theta}_c^{(j)})\right) \nabla_{\boldsymbol{\omega}_s} \ln(1 - p^{(j)})$
  end for
  Update $\boldsymbol{\omega}_s$ using $d\boldsymbol{\omega}_s$ ▷ Equation (11)
end for

Sequential Meta Learner



- **Initializer** initializes the recommender to be broadly suitable to many scenarios.
- **Update controller** finetunes the recommender parameters with flexible strategy.
- **Stop controller** stops learning timely to avoid over-fitting.

Update Strategy

- The most common method to update parameters of a neural network is:

$$\boldsymbol{\theta}_c^{(t)} = \boldsymbol{\theta}_c^{(t-1)} - \alpha \nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)},$$

- Instead, we extend the idea of learning the optimization algorithm[1] to implementing a more flexible update strategy than hand-crafted algorithms:

$$\boldsymbol{\theta}_c^{(t)} = \boldsymbol{\beta}^{(t)} \odot \boldsymbol{\theta}_c^{(t-1)} - \boldsymbol{\alpha}^{(t)} \odot \nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)},$$

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In ICLR.

Stop Policy

- According to early-stop strategy, the learning process is stopped when the performance on the validation set begins to drop. However, this is not proper for small dataset.

- Instead, we propose to learn a stochastic policy to decide whether to stop based on previous loss and gradient scales.

$$\boldsymbol{h}_s^{(t)}, \boldsymbol{c}_s^{(t)} = \mathrm{LSTM}([\mathcal{L}^{(t)}, ||\nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}||_2], \boldsymbol{h}_s^{(t-1)}, \boldsymbol{c}_s^{(t-1)}),$$
$$p^{(t)} = \sigma(\boldsymbol{W}_s \boldsymbol{h}_s^{(t)} + b_s),$$

where $p^{(t)}$ is the stop probability at time step t.

Training

- We assume access to a set of training tasks as the meta-training set, denoted as $\mathbb{T}_{meta-train}$. Each training task $T_c \in \mathbb{T}_{meta-train}$ corresponds to a scenario c.
- The parameters ω of meta learner is optimized as:

$$\min_{\omega} \mathbb{E}_{T_c}\left[\mathcal{L}_{\omega}\left(D_c^{\text{test}} | D_c^{\text{train}}\right)\right]$$

- The parameters of the initializer and update controller are directly updated by ignoring higher-order derivatives.
- The parameters of the stop controller are updated with policy gradient.

Evaluation

Method	Amazon			Movielens			Taobao		
	Recall@10	Recall@20	Recall@50	Recall@10	Recall@20	Recall@50	Recall@20	Recall@50	Recall@100
NeuMF	24.55	35.65	55.19	31.67	51.30	84.98	25.64	42.31	58.84
ItemPop	26.86	32.65	50.42	39.65	54.32	78.12	18.25	28.57	32.44
CDCF	10.27	16.72	32.79	29.19	41.04	65.75	9.81	20.93	32.14
CMF	27.64	38.31	55.24	29.80	46.91	74.37	9.16	16.47	29.31
EMCDR	31.71	42.14	58.73	43.55	60.89	83.54	20.43	31.52	45.67
CoNet	30.17	41.57	56.06	46.62	63.61	87.06	20.27	31.48	44.53
s^2Meta	**34.39**	**46.53**	**64.28**	**47.79**	**66.02**	**89.07**	**27.11**	**44.10**	**59.98**
Improve	8.35	10.42	9.45	2.51	3.79	2.31	5.73	4.23	1.90

Ablation Study

- We remove the initializer, update controller and stop controller respectively.
- Among the three variations, the random initialization hurts the performance most. The effect of removing early-stop policy is relatively small.

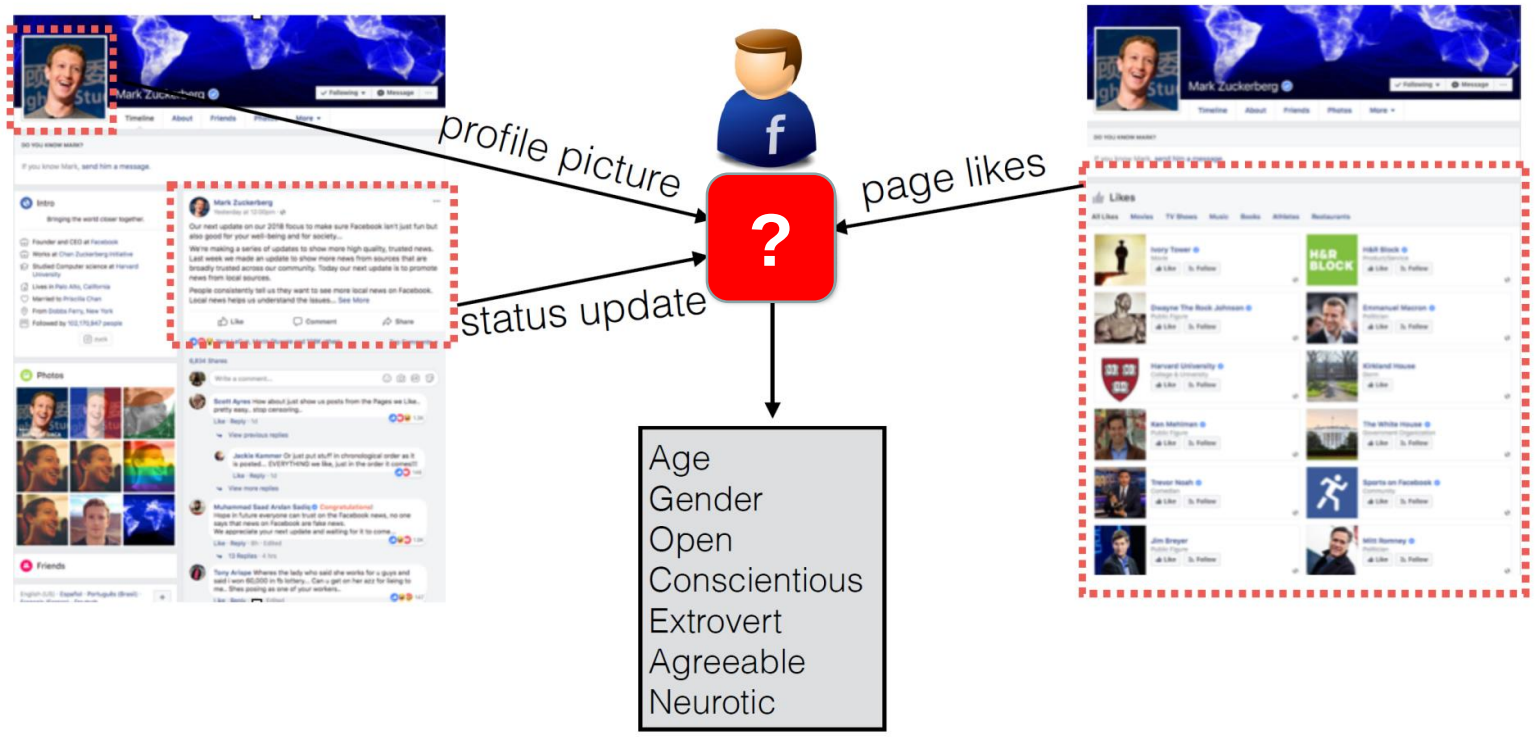
Method	Recall@10	Recall@20	Recall@50
RandInit	33.12	45.44	63.35
FixedLr	33.56	45.90	63.86
FixedStep	33.84	46.13	64.02
Complete	**34.39**	**46.53**	**64.28**

Deep User Profiling

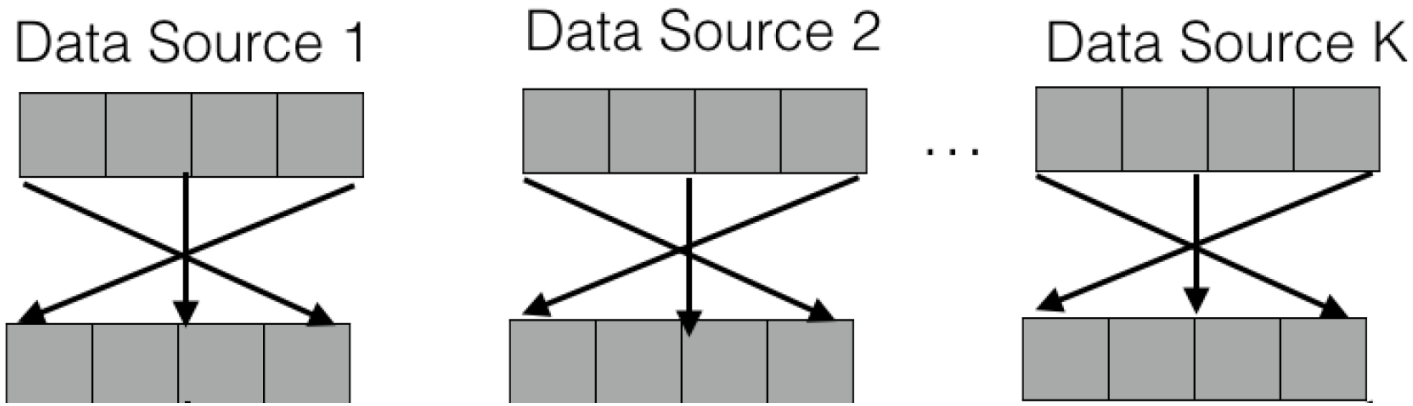
Farnadi et al. User Profiling through Deep Multimodal Fusion. WSDM'18

Deep user profiling

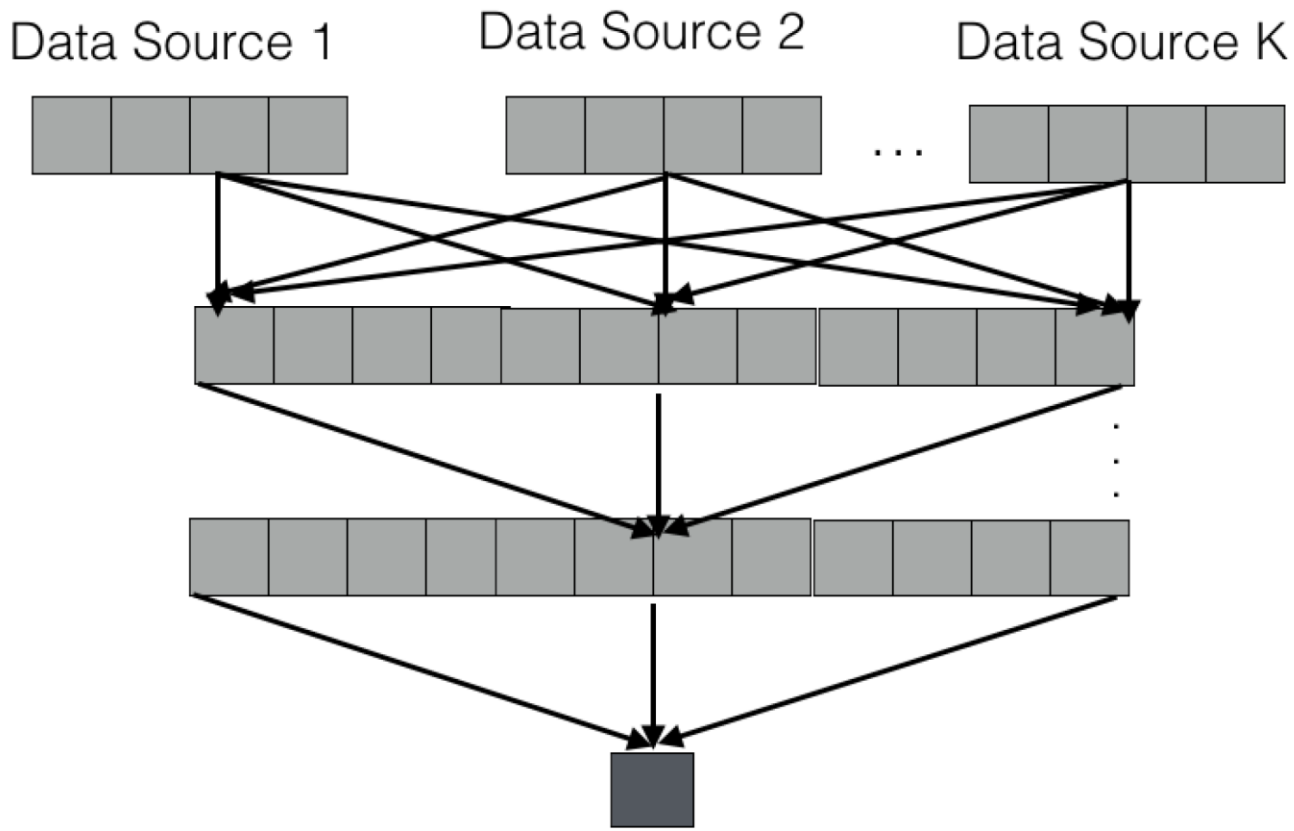
- Similar to previous network integration problem setting, we are given multiple data sources for profiling users in networks.
- The question here is: *how does deep learning shape the problem of user profiling?*



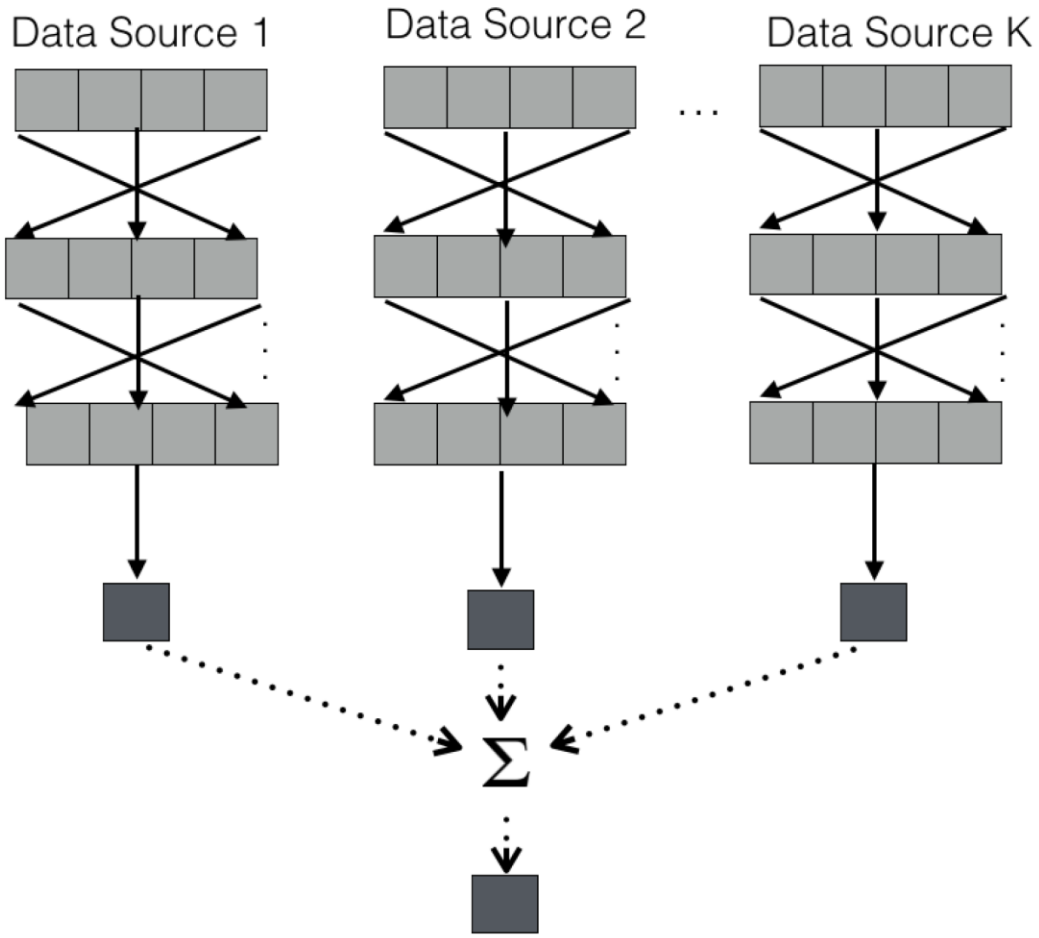
Deep user profiling



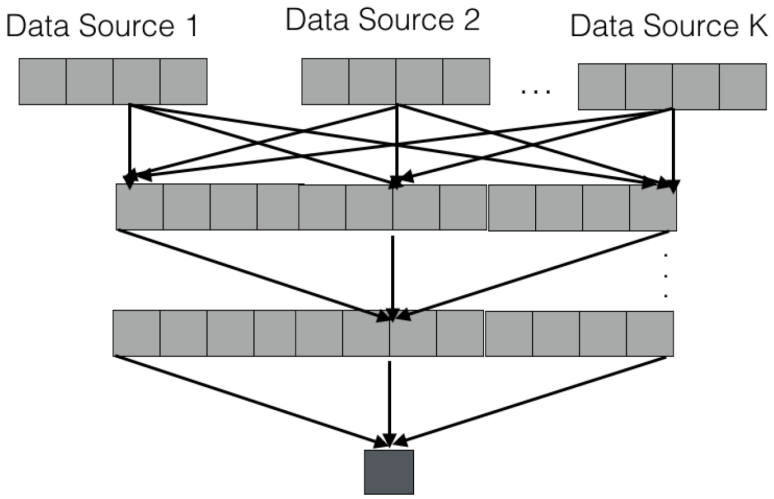
Early deep fusion of multiple data



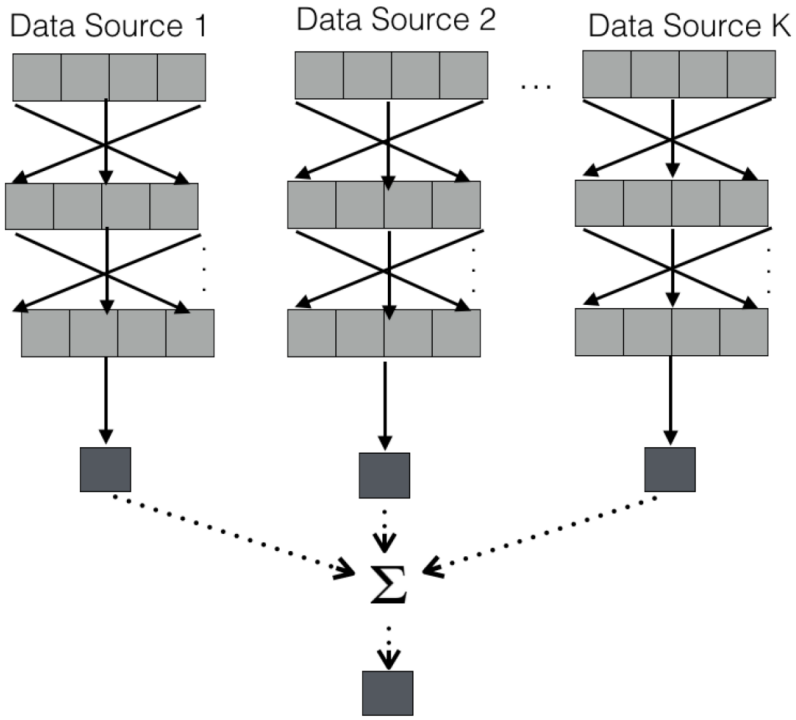
Late deep fusion of multiple data



What is the issue here?

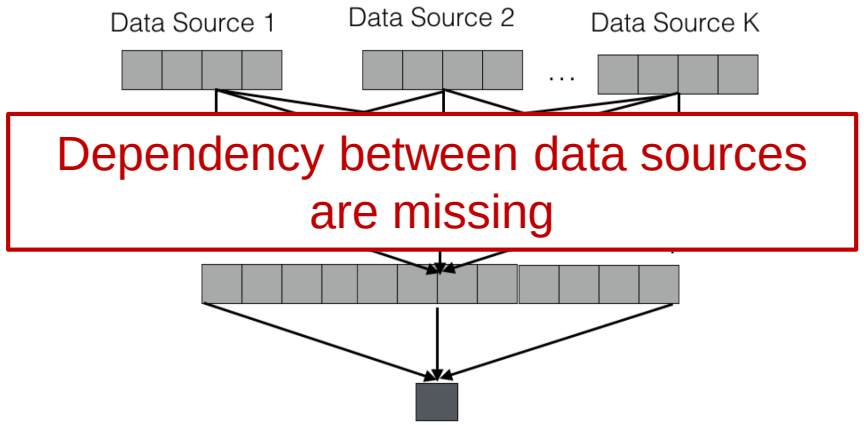


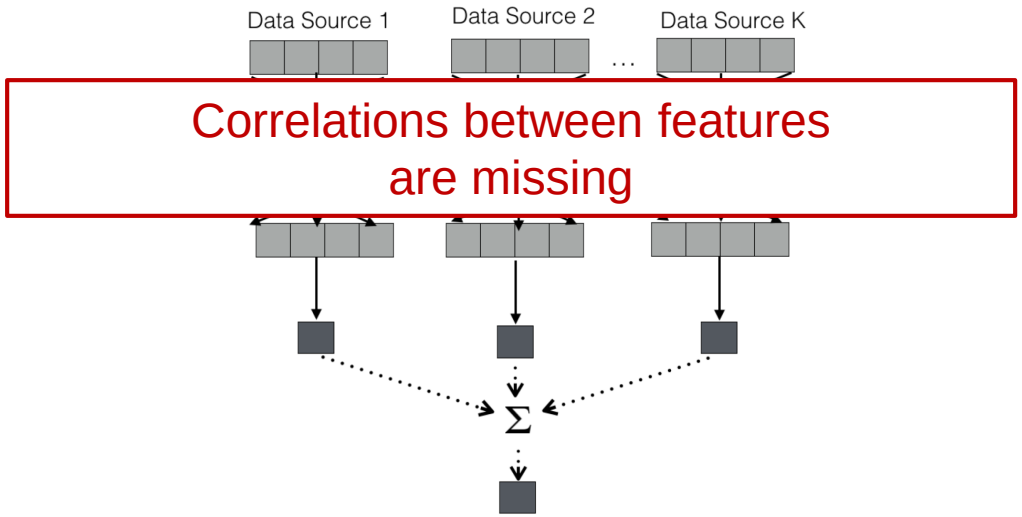
Dependency between data sources are missing

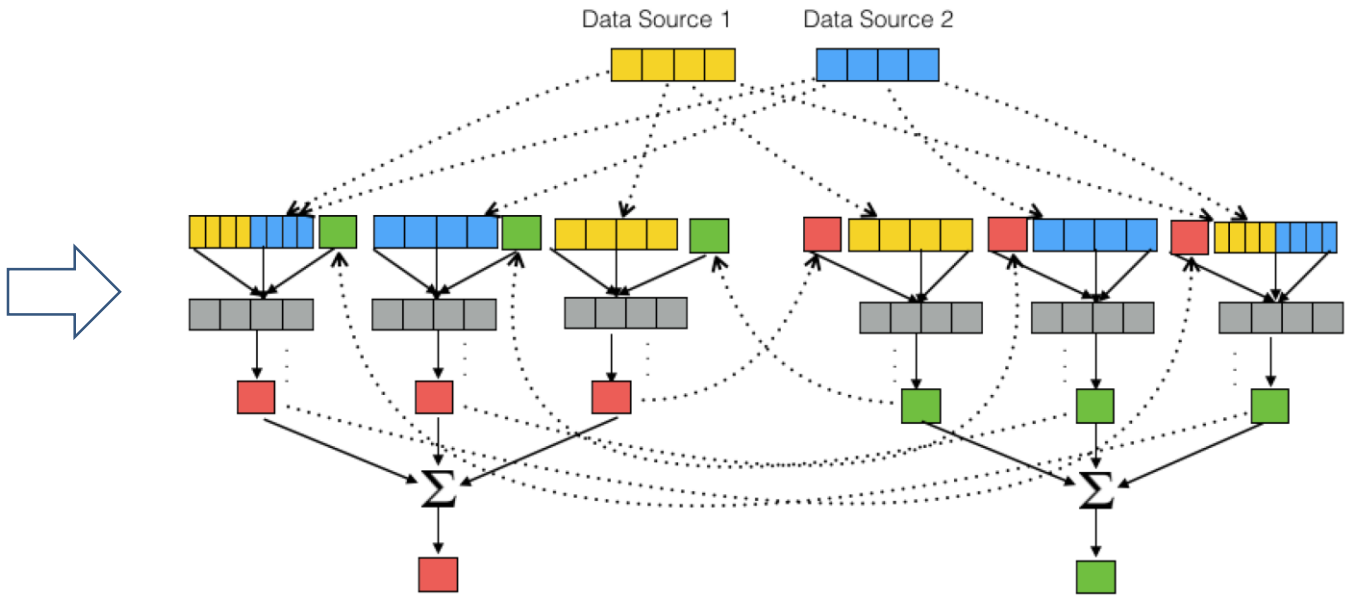


Correlations between features are missing

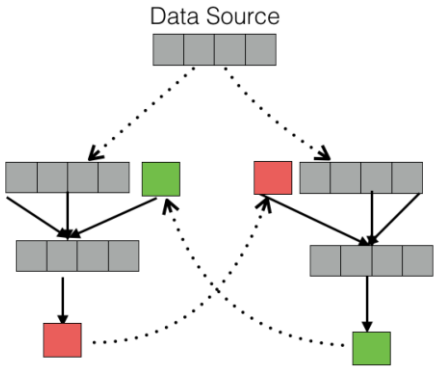
Deep user profiling model (UDMF)

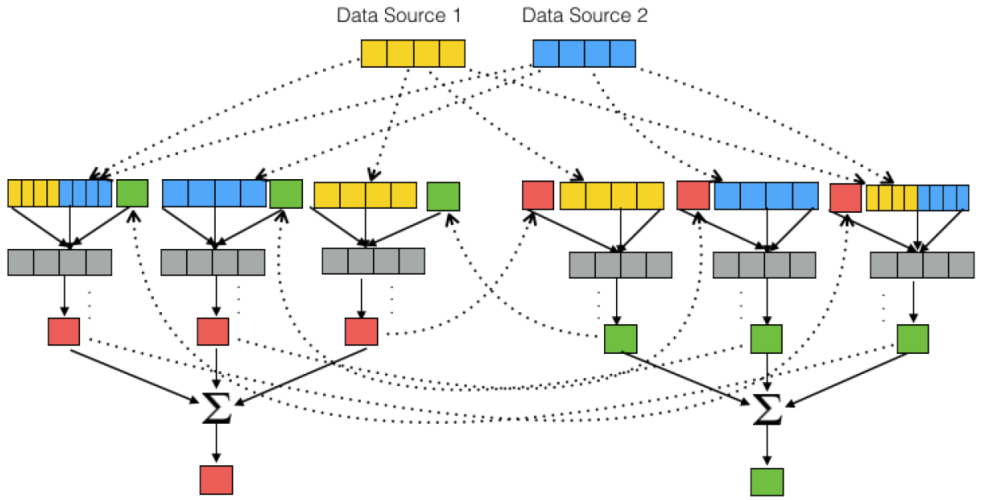






Deep user profiling model (UDMF)





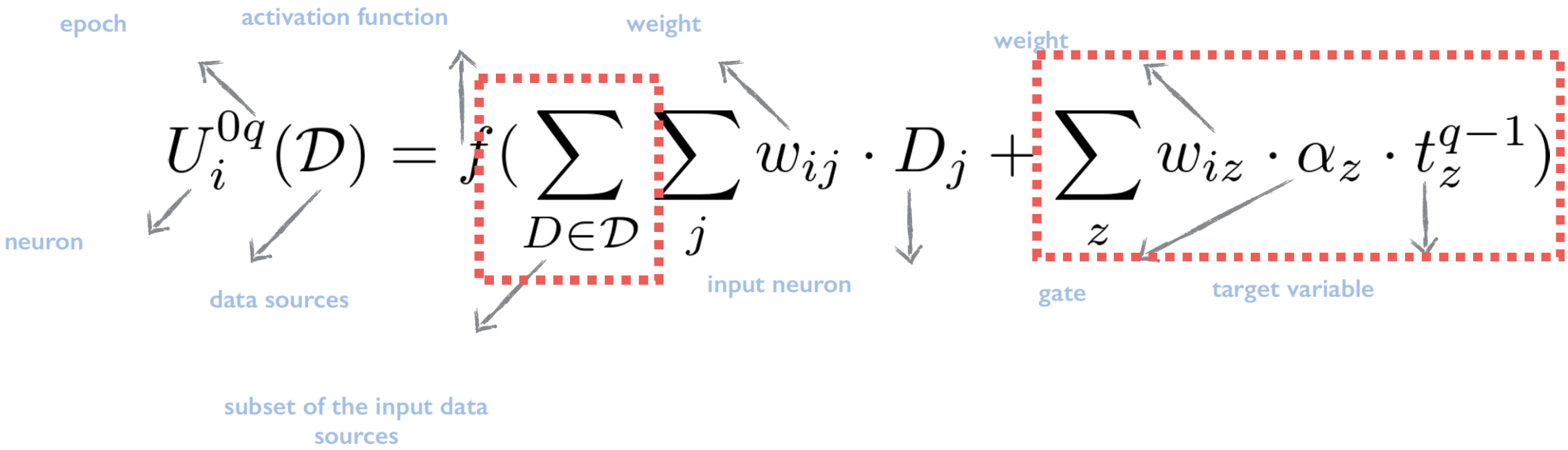
powerset combination **stacking**

$$U_i^{0q}(\mathcal{D}) = f\Big(\sum_{D\in\mathcal{D}} \sum_j w_{ij}\cdot D_j + \sum_z w_{iz}\cdot\alpha_z\cdot t_z^{q-1}\Big)$$

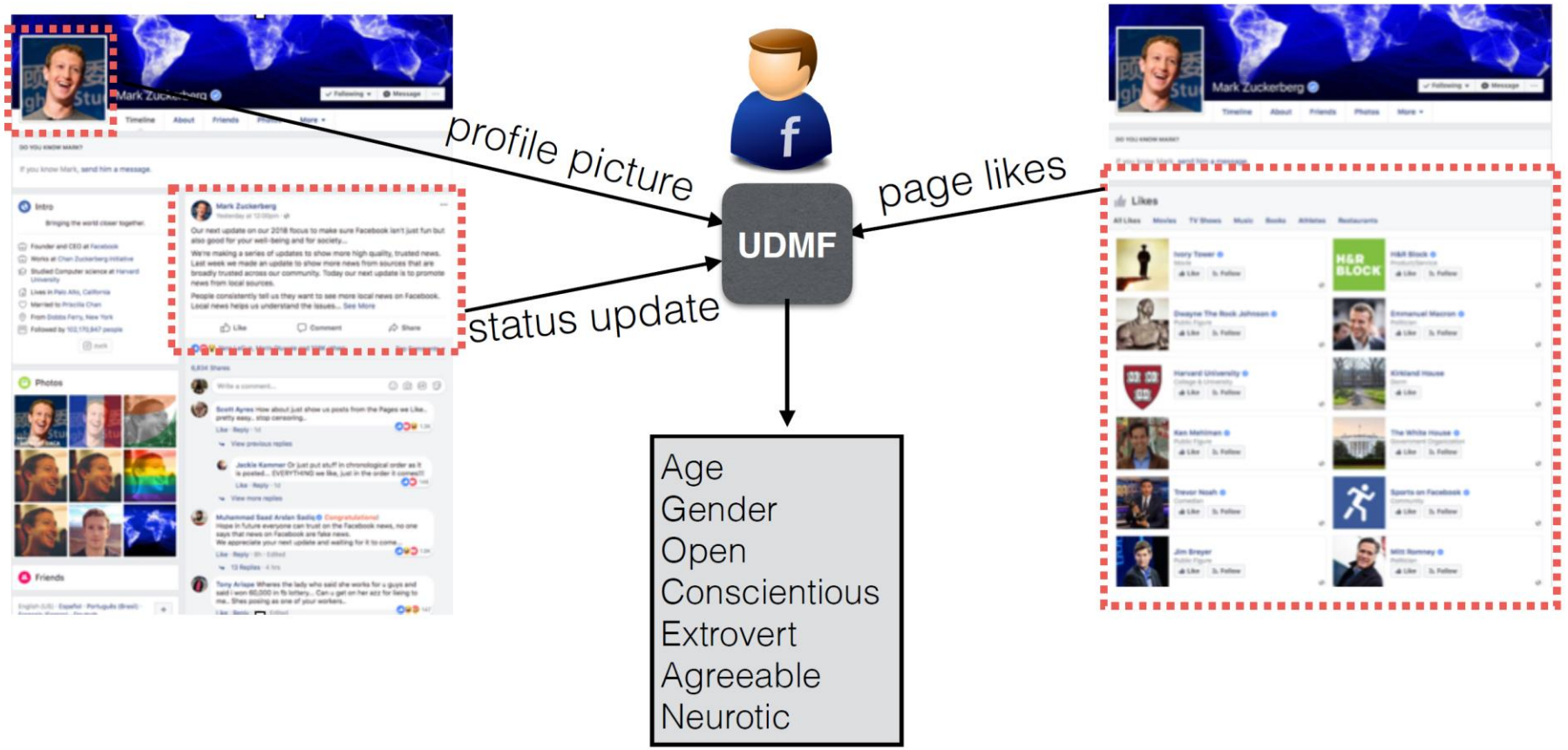
epoch
activation function
weight
weight
neuron
data sources
input neuron
gate
target variable
subset of the input data sources

Deep user profiling model (UDMF)

powerset combination **stacking**

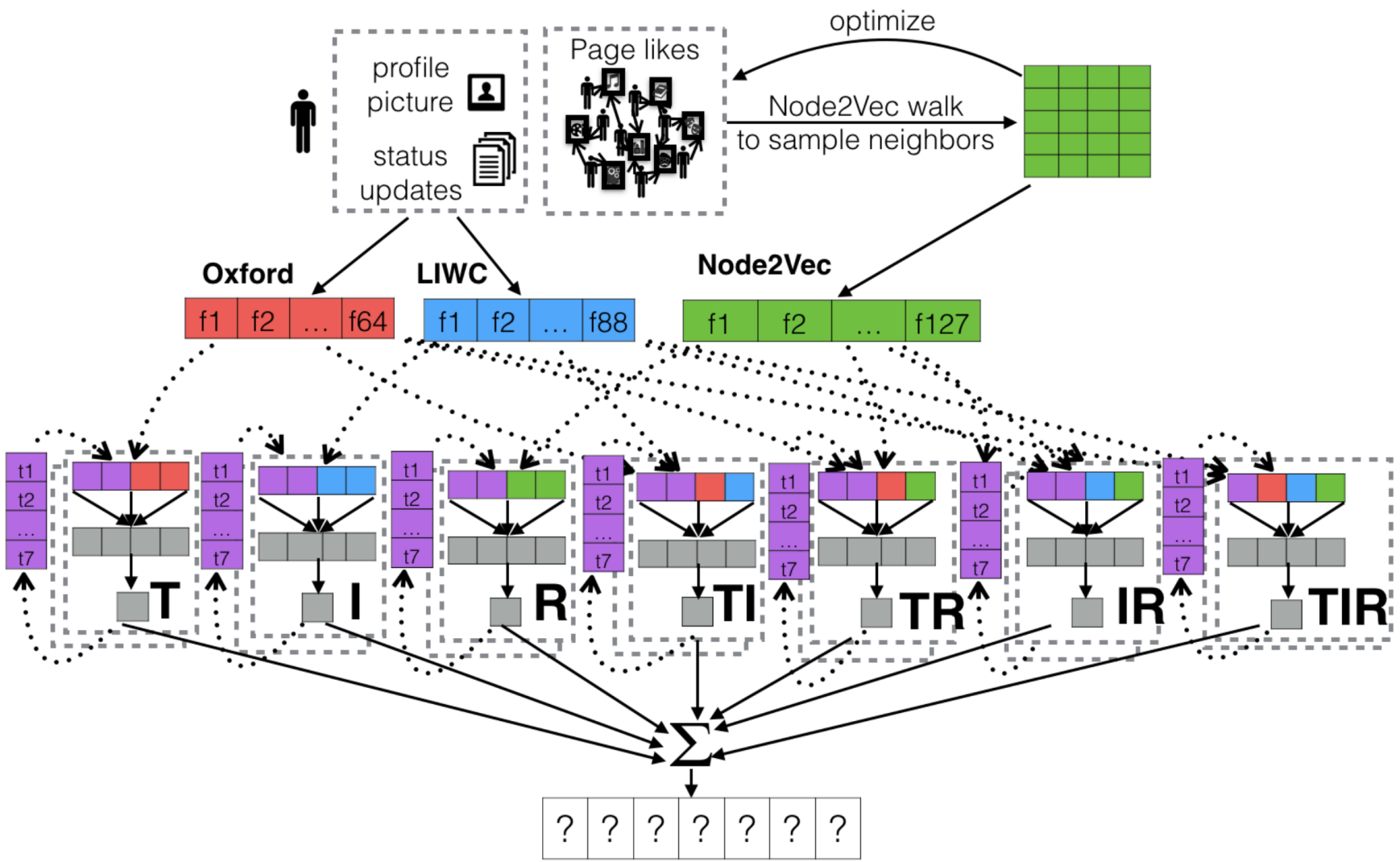


Deep User Profiling



Infer Facebook users' age, gender and personality traits

Deep User Profiling



- 3 data sources
- 7 combinations
- 7 targets

Results

Model	Stack	Age	Gender	Opn	Con	Ext	Agr	Neu
Baseline		0.488	0.492	0.502	0.502	0.506	0.506	0.486
One source								
Text	✗	0.741±0.022	0.668±0.020	0.550±0.016	0.575±0.017	0.536±0.016	0.547±0.016	0.523±0.016
	✓	0.748±0.022	0.668±0.020	0.553±0.017	0.574±0.017	0.545±0.016	0.550±0.016	0.524±0.016
Image	✗	0.552±0.016	0.915±0.027	0.502±0.015	0.500±0.015	0.504±0.015	0.512±0.015	0.520±0.016
	✓	0.550±0.016	0.897±0.027	0.516±0.015	0.511±0.015	0.518±0.015	0.519±0.015	0.541±0.016
Relation	✗	0.875±0.026	0.886±0.027	0.601±0.018	0.571±0.017	0.567±0.017	0.525±0.016	0.558±0.017
	✓	0.893±0.027	0.898±0.027	0.622±0.018	0.589±0.018	0.573±0.017	0.533±0.016	0.563±0.016
Two sources								
Early approach	✗	0.734±0.022	0.873±0.026	0.569±0.017	0.588±0.018	0.536±0.016	0.545±0.016	0.547±0.016
TI	✓	0.746±0.022	0.864±0.026	0.546±0.016	0.568±0.017	0.542±0.016	0.546±0.016	0.536±0.016
Early approach	✗	0.878±0.026	0.896±0.027	0.610±0.018	0.586±0.018	0.567±0.017	0.535±0.016	0.554±0.017
TR	✓	0.891±0.027	0.899±0.027	0.627±0.019	0.601±0.019	0.572±0.017	0.551±0.016	**0.574±0.017**
Early approach	✗	0.878±0.026	0.951±0.028	0.606±0.018	0.574±0.017	0.569±0.017	0.524±0.016	0.562±0.017
IR	✓	0.895±0.027	0.951±0.028	0.633±0.019	0.592±0.018	0.577±0.017	0.537±0.016	0.564±0.017
Three sources								
Ensemble	✗	0.876±0.026	**0.952±0.028**	0.603±0.018	0.587±0.018	0.569±0.017	0.537±0.016	0.562±0.017
(Late approach)	✓	0.893±0.027	0.949±0.028	0.626±0.019	0.606±0.018	**0.582±0.017**	0.549±0.016	0.570±0.017
Early approach	✗	0.887±0.027	0.947±0.028	0.617±0.018	0.577±0.017	0.567±0.017	0.541±0.016	0.566±0.017
TIR	✓	**0.899±0.027**	0.934±0.028	**0.635±0.019**	**0.607±0.018**	0.560±0.018	**0.551±0.016**	0.572±0.017

Results

Model	Age	Gender	Opn	Con	Ext	Agr	Neu
One/Two sources							
Page likes	0.743±0.020	0.699±0.022	0.605±0.017	0.516±0.016	0.555±0.016	0.540±0.0161	0.527±0.016
LR (T)	0.711±0.021	0.654±0.020	0.564±0.017	0.568±0.017	0.551±0.016	0.548±0.016	0.530±0.016
LR (I)	0.584±0.017	0.858 ±0.026	0.514±0.015	0.520±0.015	0.528±0.016	0.528±0.016	0.525±0.016
LR(T,I)	0.711±0.017	0.852 ±0.025	0.555±0.017	0.564±0.017	0.551±0.016	0.550±0.016	0.542±0.016
UDMF(T,I)	0.756±0.023	0.886±0.027	0.569±0.017	0.575±0.017	0.552±0.017	0.552±0.016	0.539±0.016
UDMF(T,R)	0.879±0.026	0.943±0.028	0.628±0.019	0.607±0.018	0.580±0.017	**0.564±0.017**	0.575±0.017
UDMF(I,R)	0.892±0.027	0.955±0.029	0.630±0.019	0.607±0.018	0.587±0.018	0.551±0.016	0.571±0.017
Three sources							
Weighted Soft Voting	0.656±0.019	0.861±0.026	0.523±0.016	0.0523±0.016	0.508±0.015	0.507±0.015	0.518±0.015
Random Forest (100)(T,I,R)	0.786 ±0.023	0.900±0.027	0.588±0.018	0.564 ±0.017	0.544±0.016	0.549±0.016	0.538±0.016
LR(T,I,R)	0.808 ±0.024	0.888±0.027	0.603±0.018	0.585 ±0.018	0.550±0.017	0.550±0.016	0.572±0.017
UDMF(T,I,R)	**0.903±0.027**	**0.956±0.029**	**0.647±0.019**	**0.615±0.018**	**0.592±0.018**	0.556±0.017	**0.580±0.017**

- Deep learning improves the accuracy of user profile predictions;
- Powerset combination enhances the fusion power;
- Stacking works when we have multi-task learning (e.g., user profiling);
- Your age, gender and personality traits are highly predictable from your Facebook activities.

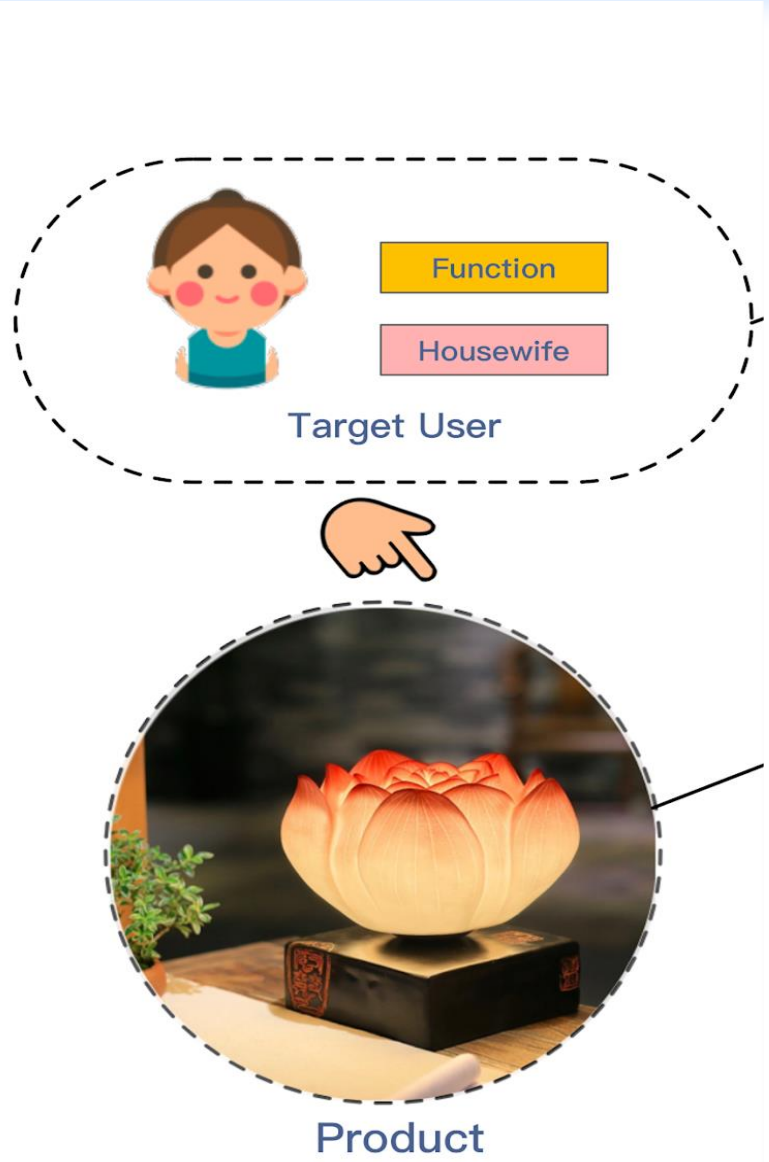
Towards Knowledge-Based Personalized Product Description Generation in E-commerce

Chen et al. Towards Knowledge-Based Personalized Product Description Generation in E-commerce. KDD'19

Motivation

- Online stores in E-commerce heavily rely on textual product descriptions to provide crucial product information and, eventually, convince customers to buy the recommended products.

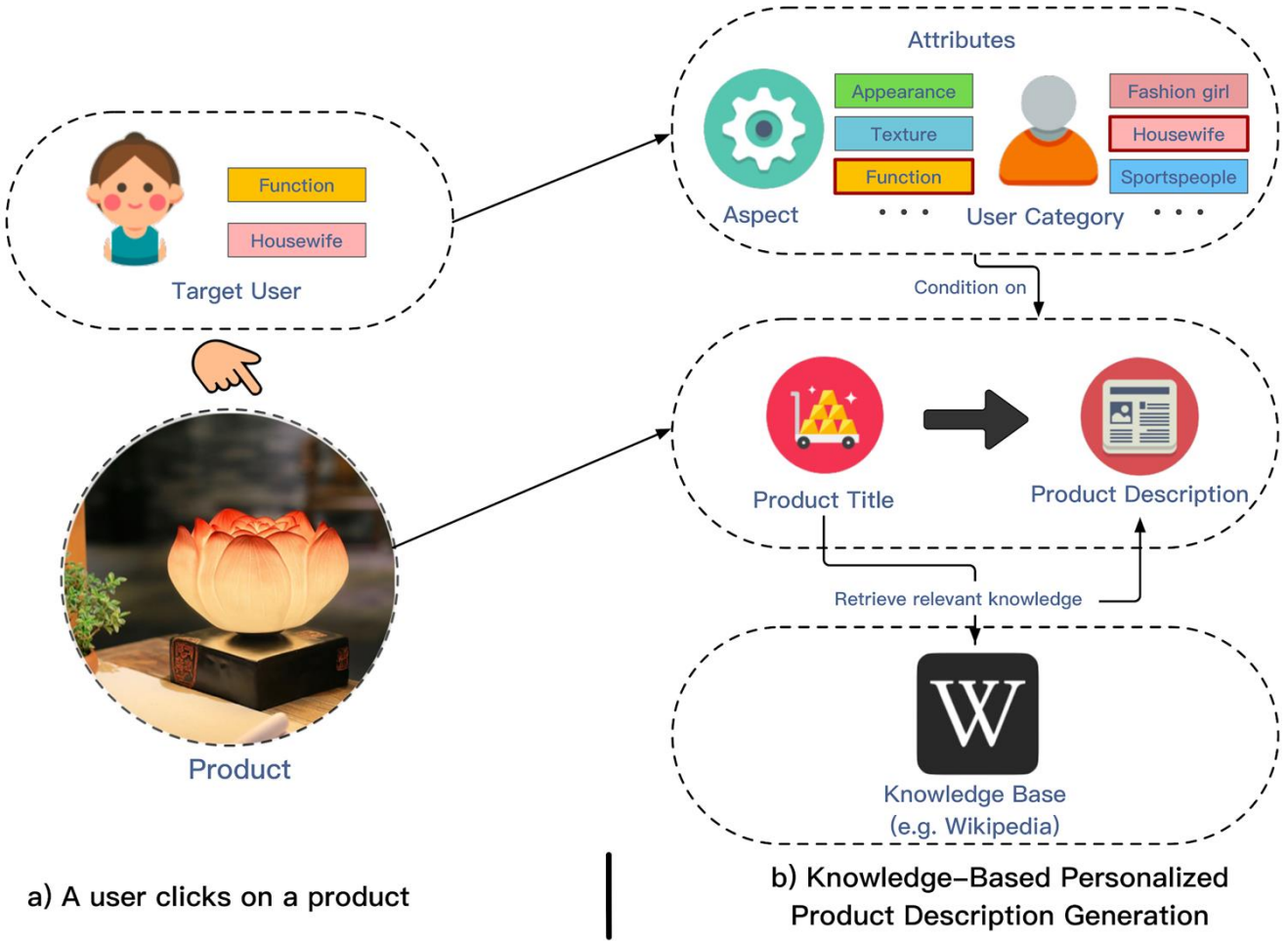
- Can we automatically generate **personalized** product descriptions?



a) A user clicks on a product

Motivation

1. Manually creating product descriptions in online shopping platforms is tedious and time-consuming.
2. Such descriptions should be personalized, based on customers' preferences, and promote their interests.
3. Descriptions should also contain relevant knowledge to help customers make an informed decision.



Problem

Input:

- Product title: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
- Attributes $\boldsymbol{a} = (a_1, \ldots, a_l)$
  - product aspect and user category
- Relevant Knowledge: $\boldsymbol{w} = (w_1, w_2, \ldots, w_u)$

Output:

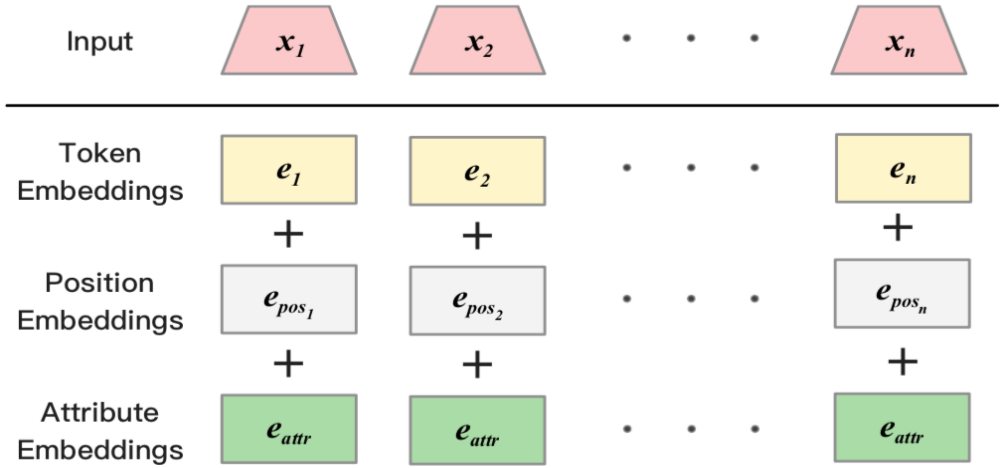
- Product description: $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$

Methodology

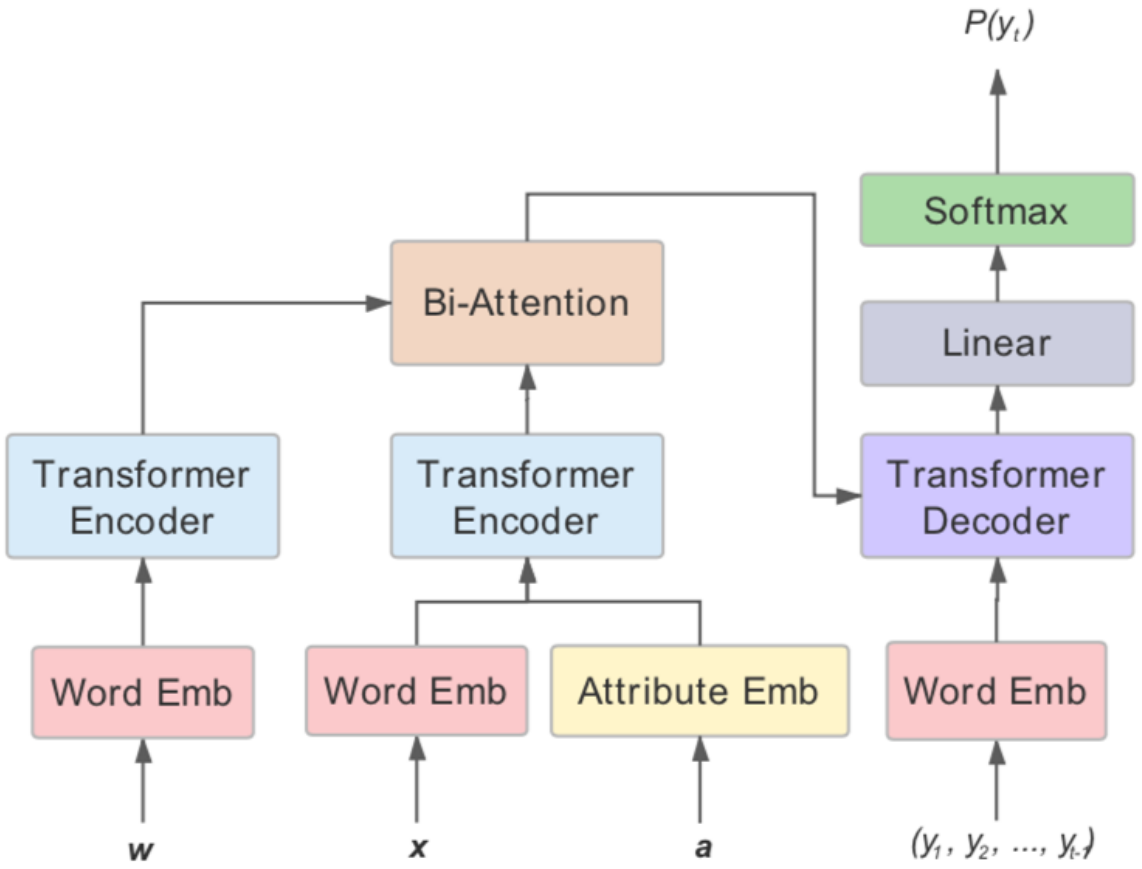
- The objective is to generate a sequence to approximate the target (product description) based on the input sequence (product title) and attributes. Formally, we have:

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{a}) = \prod_{t=1}^{m} P(y_t|y_1, y_2, \ldots y_{t-1}, \boldsymbol{x}, \boldsymbol{a})$$

- Attributes are embedded and added to token embeddings.



Methodology



1. Transformer Encoder-Decoder
   - Seq2seq framework based on multi-head self-attention
2. Attribute Fusion
   - Product aspects and user categories are unified as different types of *attributes*
   - Attribute embeddings are added to product title embeddings to represent different interpretations for a product
3. Knowledge Incorporation
   - *Knowledge* retrieved by the product title is encoded and combined into the generation framework via bi-attention

Experiments

- The **TaoDescribe** dataset contains 2,129,187 products and their descriptions, composed by the sellers and content producers on the website from November 2013 to December 2018.

- We adopt 2 evaluation metrics common for general text generation tasks and devise another metric suitable for this personalized task:
  - BLEU
  - Lexical Diversity
  - Attribute Capturing Score

The **TaoDescribe** dataset and code will be made publicly available at https://github.com/qibinc/KOBE

Experiments

Table 4: Attribute capturing score

Model	Aspect (%)	User Category (%)
Baseline	63.0	71.9
Attr-S (Both)	74.0	**86.3**
Attr-T (Both)	72.8	83.3
Attr-A (Both)	**74.6**	86.0

Table 5: Model ablations on 3 model components.

Model	BLEU	Diversity (n=4) ($\times 10^6$)
KOBE	7.6	**15.1**
- knowledge	**8.2**	11.1
- user category	7.6	9.6
- aspect	7.2	7.8

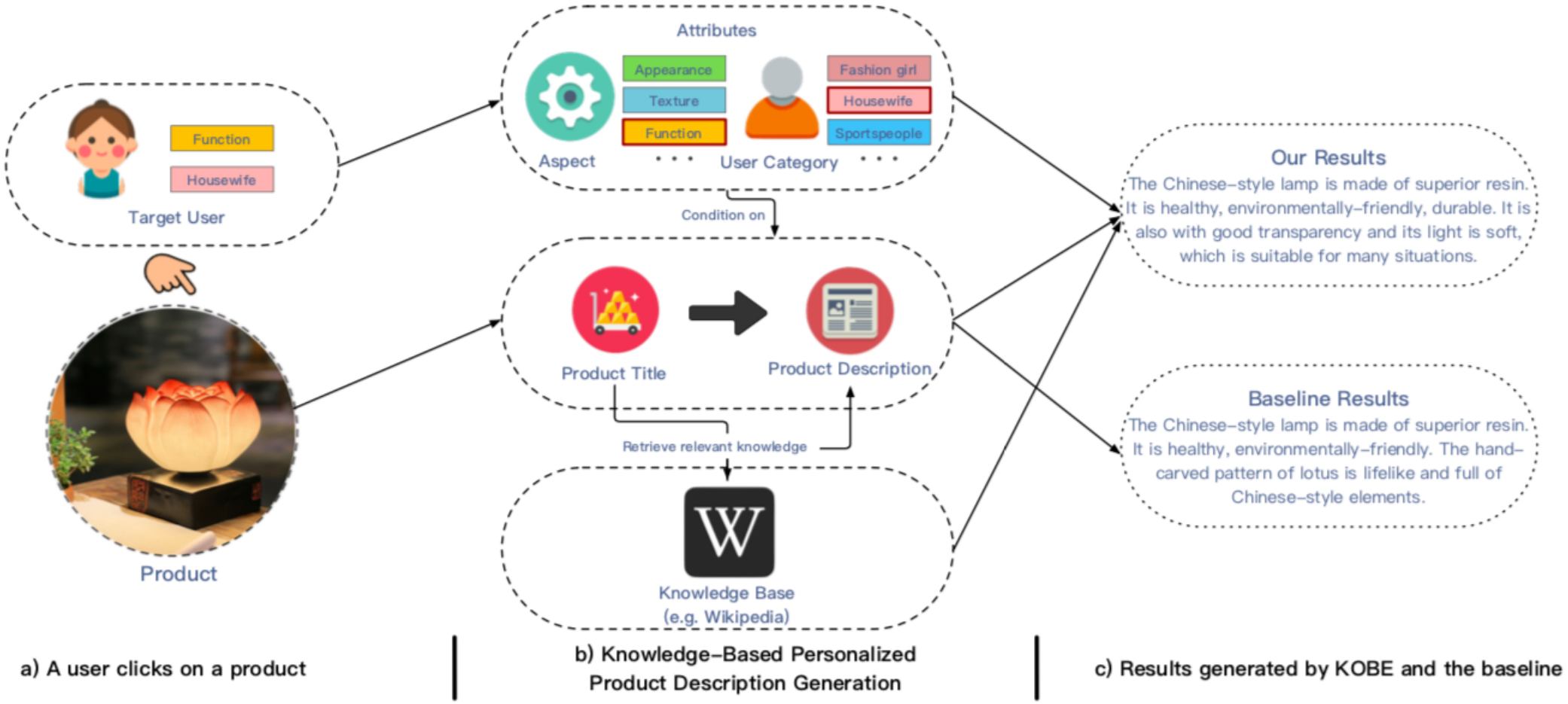
- KOBE improves both the generation quality and diversity.
- KOBE better reflects a specific user category and a product aspect, showing the effectiveness of personalization.

Experiments

Varying description aspects	
"appearance"	*"function"*
The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly. The hand-carved pattern of lotus is lifelike and full of Chinese-style elements.	The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly, durable. It is also with good transparency and its light is soft, which is suitable for many situations.

Varying user categories	
"housewife"	*"geek"*
The Euro-style children's desk is made of superior wood, which is hard, solid and durable. It is covered with environmentally friendly oil paint, which is healthy and safe.	The Euro-style wooden computer desk is made of superior rubberwood, which is hard. It has a clear texture and solid structure and it is covered with environmentally friendly oil paint, which is healthy and safe. It has a multifunctional design for storage, which can meet your requirements for gathering.

Summary


Function
Housewife
Target User
Product
Attributes
Appearance
Texture
Function
Aspect
Fashion girl
Housewife
Sportspeople
User Category
Condition on
Product Title
Product Description
Retrieve relevant knowledge
Knowledge Base
(e.g. Wikipedia)
Our Results
The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly, durable. It is also with good transparency and its light is soft, which is suitable for many situations.
Baseline Results
The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly. The hand-carved pattern of lotus is lifelike and full of Chinese-style elements.
a) A user clicks on a product
b) Knowledge-Based Personalized Product Description Generation
c) Results generated by KOBE and the baseline

Representation Learning on Networks

- **The first part:**
  - Conventional network analysis
    - Node classification
    - Social tie & link prediction
    - Influence & behavior modeling
  - Network embeddings
    - Embedding models
    - Theoretical understanding
    - Large-scale embedding
- **The second part:**
  - Graph neural networks
    - CNN → Graph convolution
    - GAN → Graph GAN
  - Large-scale applications
    - Knowledge graph linking
    - Recommendation in E-commerce
    - Online-to-offline recommendations
    - Social influence in gaming

https://www.aminer.cn/nrl_www2019

Thank you !

Collaborators: John Hopcroft, Jon Kleinberg, Chenhao Tan (**Cornell**)

Jiawei Han (**UIUC**), Philip Yu (**UIC**)

Jian Pei (**SFU**), Hanghang Tong (**ASU**)

Tiancheng Lou (**Google&Baidu**), Jimeng Sun (**GIT**)

Wei Chen, Ming Zhou, Long Jiang, Chi Wang, Kuansan Wang (**Microsoft**)

Hongxia, Jingren Zhou, Chang Zhou (**Alibaba**)

Jiezhong Qiu, Jie Zhang, Fanjin Zhang, Qibin Chen, Yukuo Cen, et al. (**THU**)

Jie Tang, KEG, Tsinghua U,

Download all data & Codes,

http://keg.cs.tsinghua.edu.cn/jietang

http://arnetminer.org/data

http://arnetminer.org/data-sna