Metapath2vec
KGCN

JU&ST
GENI

KDD17:Metapath2vec: Scalable Representation Learning for Heterogeneous Networks

CIKM18:Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings

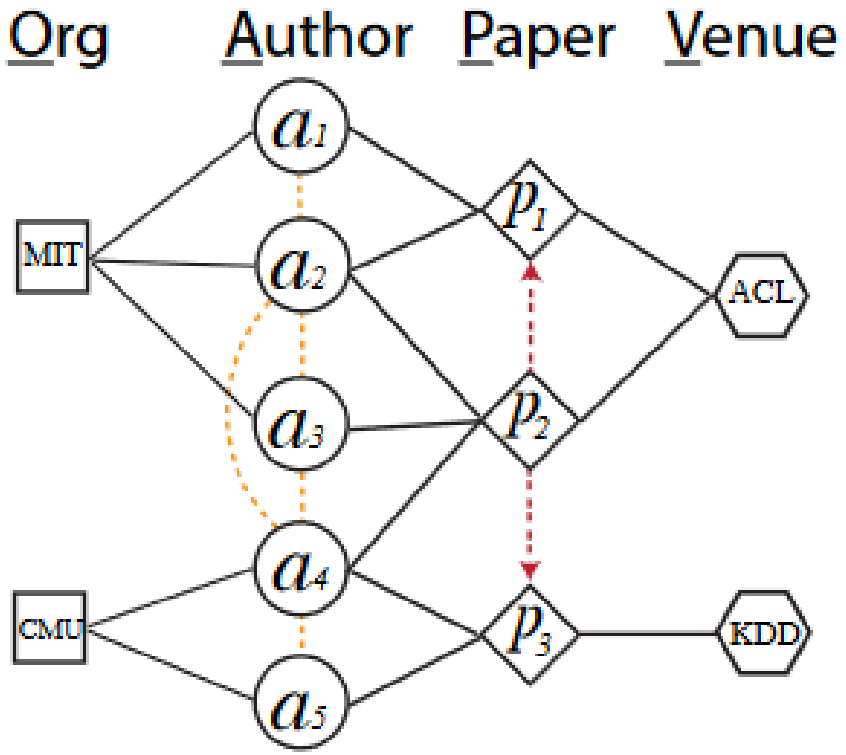
WWW19:Knowledge Graph Convolutional Networks for Recommender Systems

KDD19:Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks

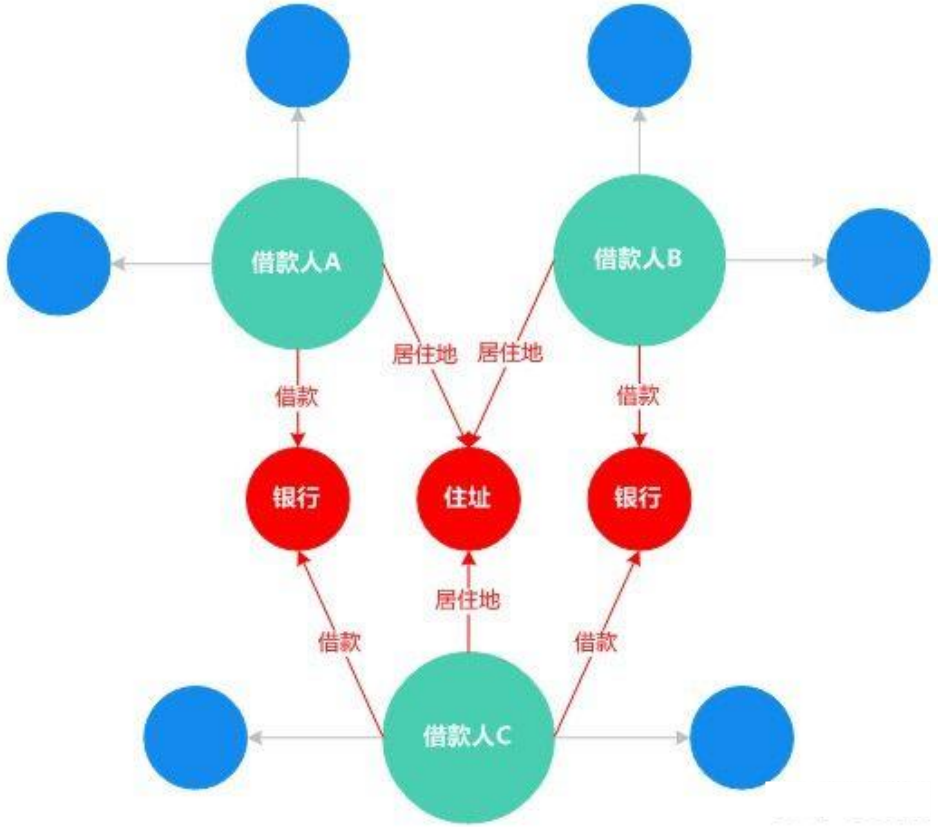
Metapath2vec

KDD17:Metapath2vec: Scalable Representation Learning for Heterogeneous Networks

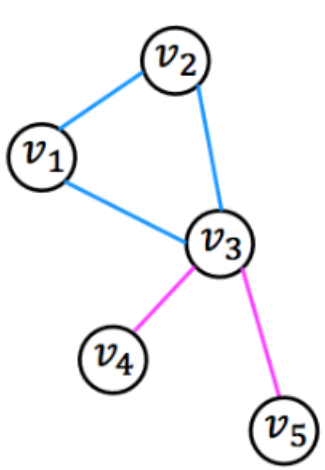
Heterogeneous Networks



Knowledge Graph



Conventional Network Mining and Learning



hand-crafted feature matrix

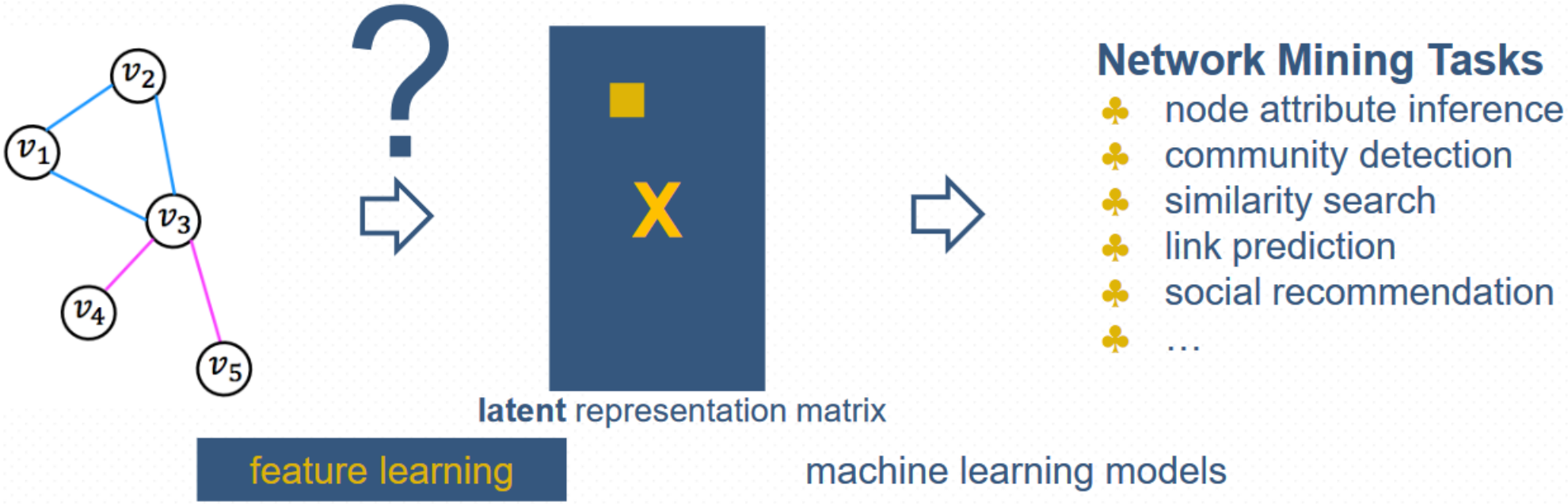
Network Mining Tasks

- node attribute inference
- community detection
- similarity search
- link prediction
- social recommendation
- …

feature engineering

machine learning models

Network Embedding for Mining and Learning

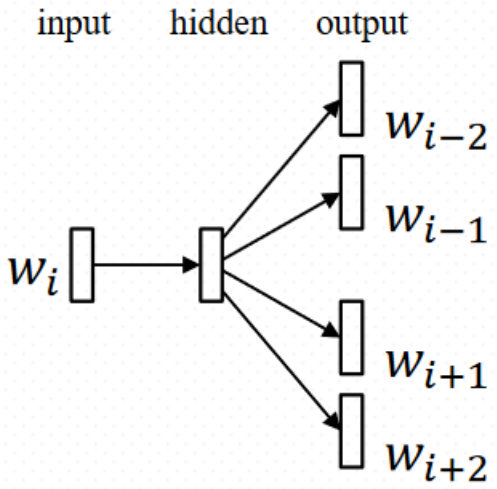


Word Embedding in NLP

- Input: a text corpus $D = \{W\}$
- Output: $\boldsymbol{X} \in R^{|W| \times d}, d \ll |W|$, *d*-dim vector $\boldsymbol{X}_w$ for each word *w*.

- Computational lens on big social and information networks.
- The connections between individuals form the structural …
- In a network sense, individuals matters in the ways in which …
- Accordingly, this thesis develops computational models to investigating the ways that …
- We study two fundamental and interconnected directions: user demographics and network diversity
- … …

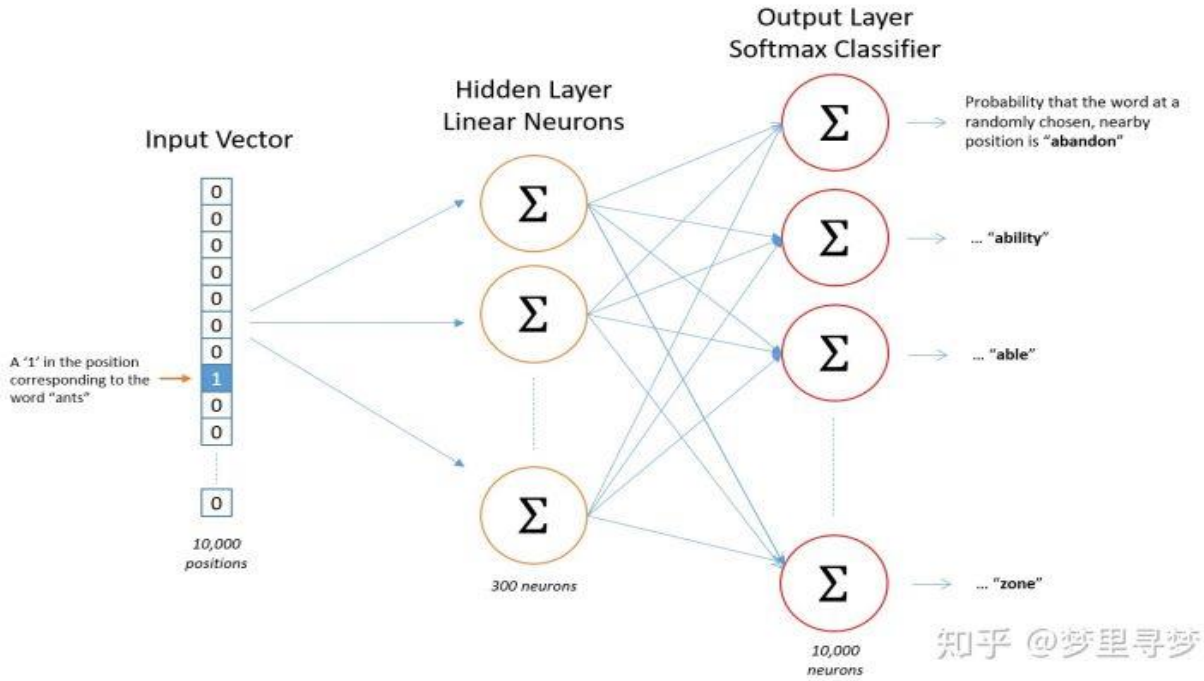
sentences



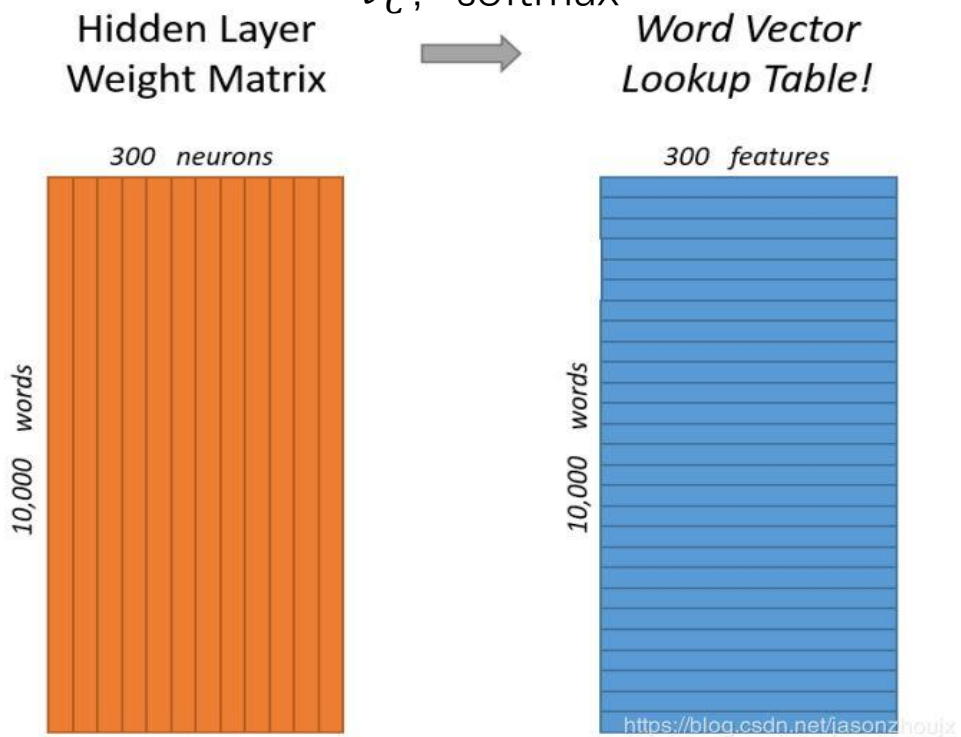
word2vec

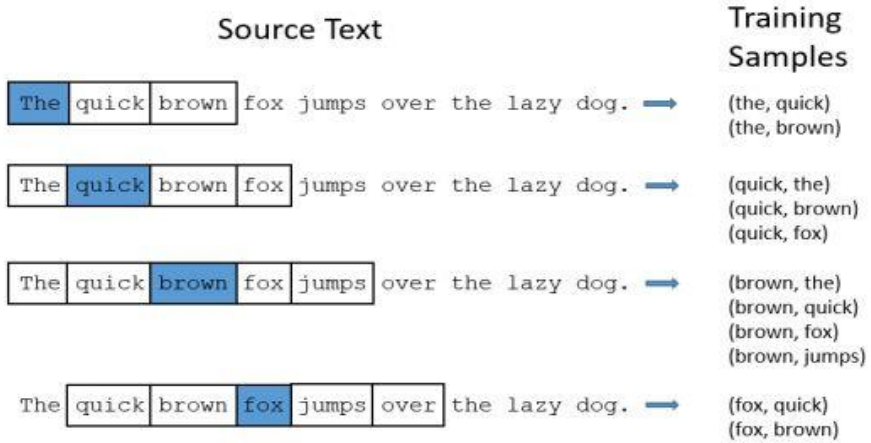
latent representation vector

Skip-gram



$* v_c$, softmax

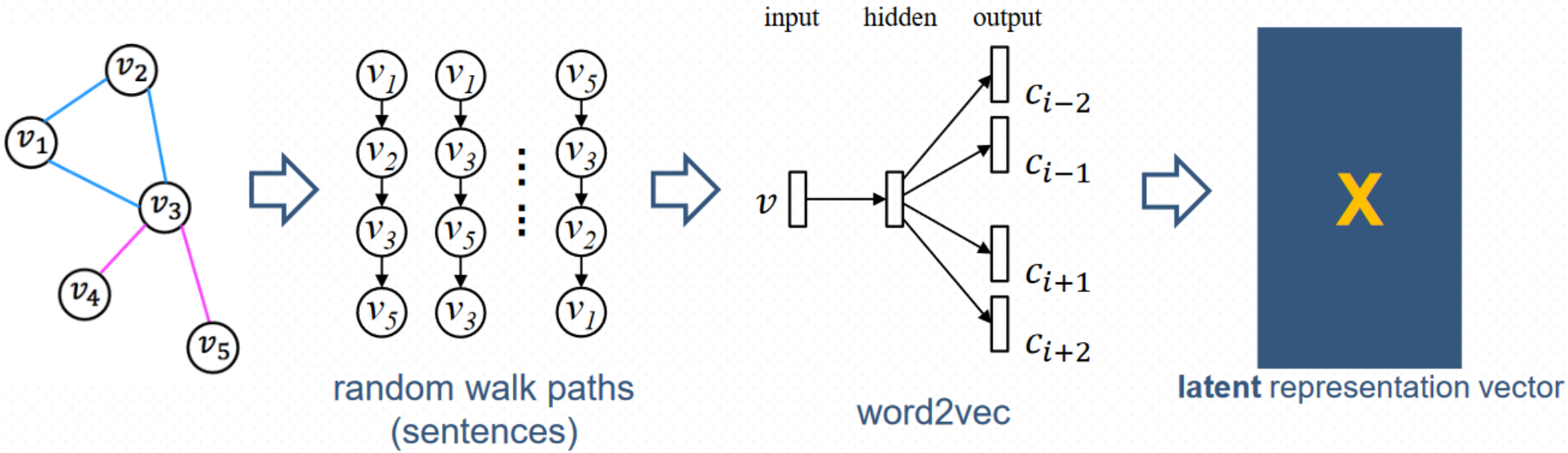




$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Network Embedding

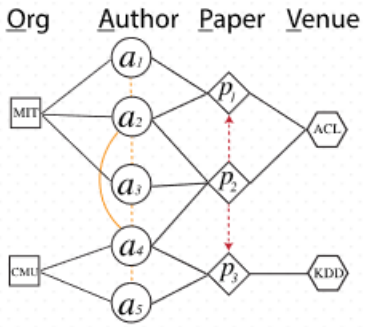
- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times d}, d \ll |V|$, d-dim vector $\boldsymbol{X}_v$ for each node v.



DeepWalk [Perozzi et al., KDD14]

Heterogeneous Network Embedding : Problem

- Input: **a heterogeneous information network** $G = (V, E, T)$
- Output: $\boldsymbol{X} \in R^{|V| \times d}, d \ll |V|$, *d*-dim vector $\boldsymbol{X}_v$ for each node *v*.



?

X

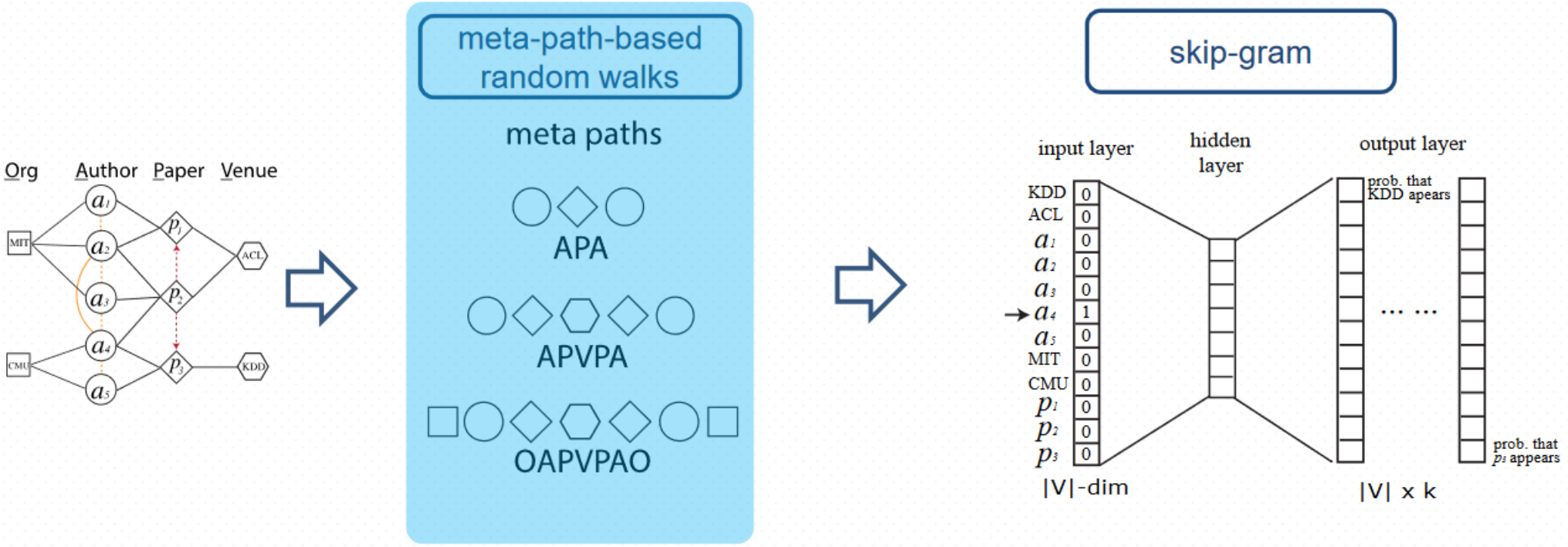
latent representation vector

Heterogeneous Network Embedding : Challenges

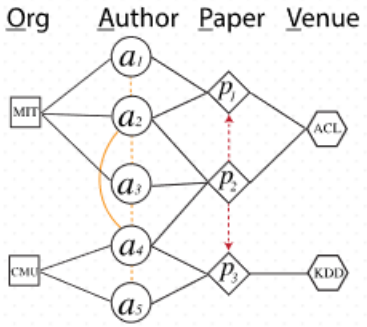
- How do we effectively preserve the concept of “node-context” among multiple types of nodes, e.g., authors, papers, & venues in academic heterogeneous networks?
- Can we directly apply homogeneous network embedding architectures to heterogeneous networks?
- It is also difficult for conventional meta-path based methods to model similarities between nodes without connected meta-paths.

Heterogeneous Network Embedding : Solutions

Metapath2vec

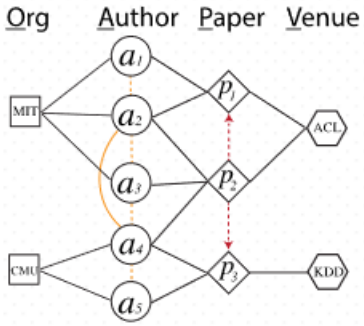


Metapath2vec : Meta-Path-Based Random Walks



Goal: to generate paths that are able to capture both the semantic and structural correlations between different types of nodes, facilitating the transformation of heterogeneous network structures into skip-gram.

Metapath2vec : Meta-Path-Based Random Walks



- Given a meta-path scheme

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

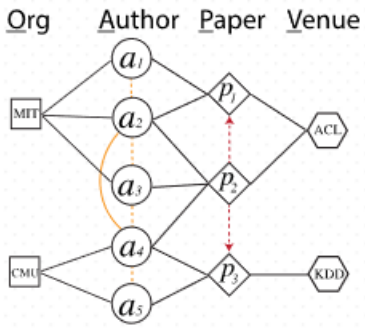
- The transition probability at step *i* is defined as

$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

- Recursive guidance for random walkers, i.e.,

$$p(v^{i+1}|v_t^i) = p(v^{i+1}|v_1^i), \text{ if } t = l$$

Metapath2vec : Meta-Path-Based Random Walks



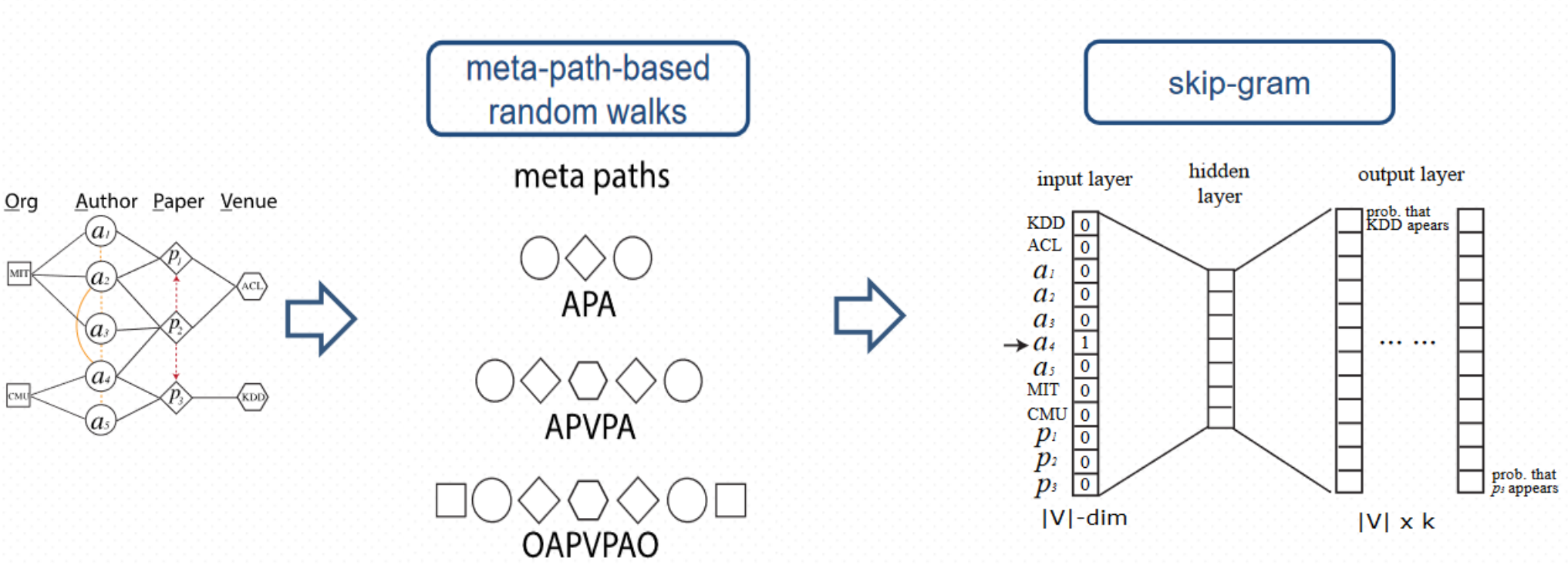
- Given a meta-path scheme (Example)

  OAPVPAO

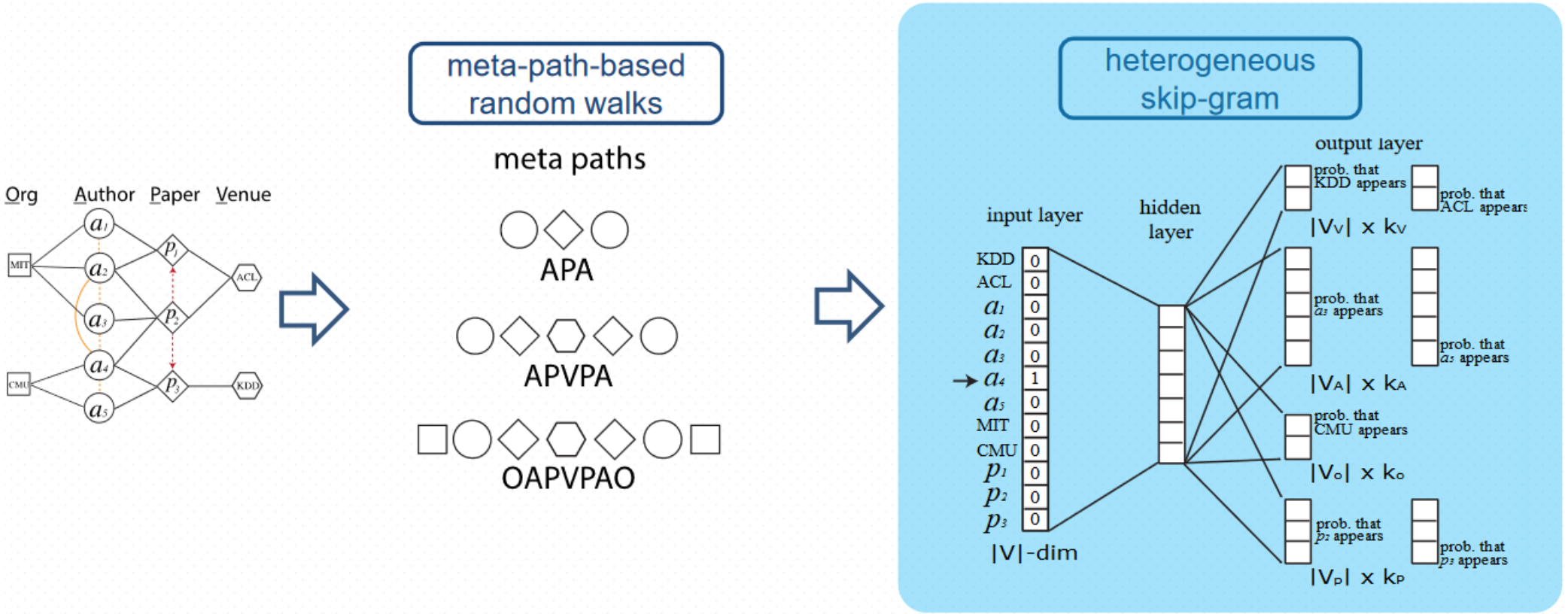
- In a traditional random walk procedure, in the toy example, the next step of a walker on node a4 transitioned from node CMU can be all types of nodes surrounding it—a2, a3, a5, p2, p3, and CMU.

- Under the meta-path scheme ‘OAPVPAO’, for example, the walker is biased towards paper nodes (P) given its previous step on an organization node CMU (O), following the semantics of this meta-path.

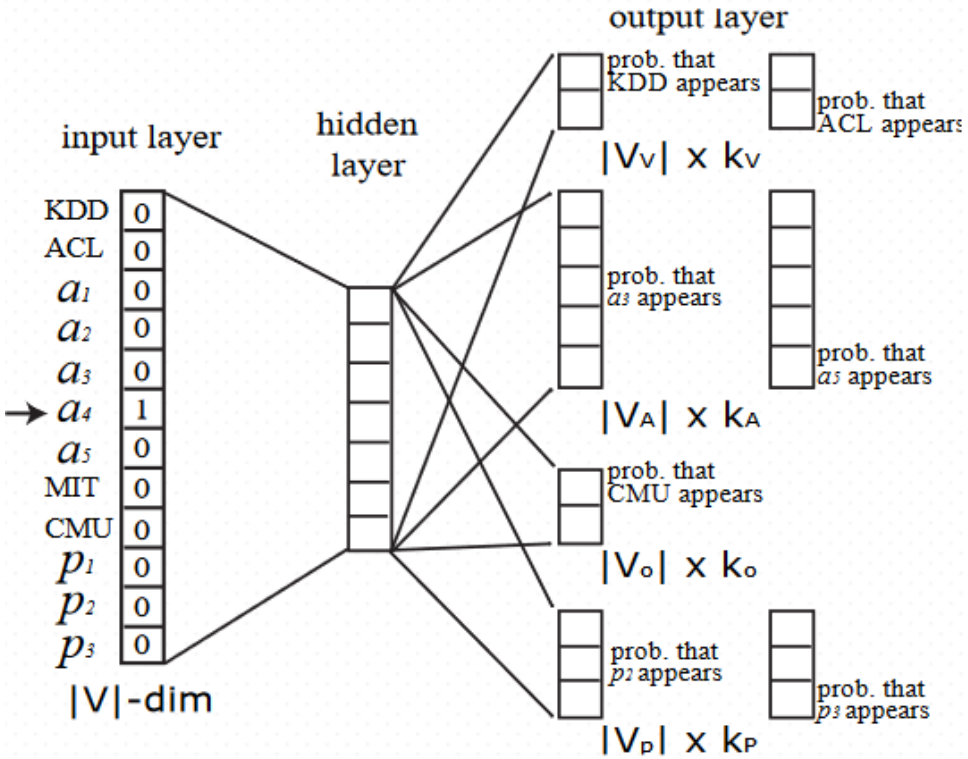
Metapath2vec



Metapath2vec++



Metapath2vec++ : Heterogeneous Skip-Gram



- softmax in *metapath2vec*

$$p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u} \cdot e^{X_v}}$$

- softmax in *metapath2vec++*

$$p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u_t \in V_t} e^{X_{u_t}} \cdot e^{X_v}}$$

- objective function (heterogeneous negative sampling)

$$\mathcal{O}(\mathbf{X}) = \log \sigma(X_{c_t} \cdot X_v) + \sum_{k=1}^{K} \mathbb{E}_{u_t^k \sim P_t(u_t)} [\log \sigma(-X_{u_t^k} \cdot X_v)]$$

- stochastic gradient descent

$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_{u_t^k}} = (\sigma(X_{u_t^k} \cdot X_v - \mathbb{I}_{c_t}[u_t^k]))X_v$$

$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_v} = \sum_{k=0}^{K} (\sigma(X_{u_t^k} \cdot X_v - \mathbb{I}_{c_t}[u_t^k]))X_{u_t^k}$$

Algorithm

Input: The heterogeneous information network $G = (V, E, T)$, a meta-path scheme $\mathcal{P}$, #walks per node w, walk length l, embedding dimension d, neighborhood size k

Output: The latent node embeddings $\mathbf{X} \in \mathbb{R}^{|V| \times d}$

initialize $\mathbf{X}$;
for $i = 1 \rightarrow w$ **do**
  for $v \in V$ **do**
    MP = MetaPathRandomWalk$(G, \mathcal{P}, v, l)$;
    $\mathbf{X}$ = HeterogeneousSkipGram$(\mathbf{X}, k, MP)$;
  end
end
return $\mathbf{X}$;

MetaPathRandomWalk$(G, \mathcal{P}, v, l)$
$MP[1] = v$;
for $i = 1 \rightarrow l-1$ **do**
  draw u according to Eq. 3 ;
  MP[i+1] = u ;
end
return MP ;

HeterogeneousSkipGram$(\mathbf{X}, k, MP)$
for $i = 1 \rightarrow l$ **do**
  $v = MP$[i] ;
  for $j = max(0, i-k) \rightarrow min(i+k, l)$ & $j \neq i$ **do**
    $c_t = MP$[j] ;
    $X^{new} = X^{old} - \eta \cdot \frac{\partial O(\mathrm{X})}{\partial X}$ (Eq. 7) ;
  end
end

Experiments

Heterogeneous Data

- AMiner Academic Network
  - ~~9~~ 1.7 million authors
  - 3 million papers
  - 3800+ venues
  - 8 research areas

246678 labeled authors
0 labeled papers
133 labeled venues
8 areas

Baselines

- DeepWalk [KDD '14]
- node2vec [KDD '16]
- LINE [WWW '15]
- PTE [KDD '15]

Parameters

- #walks: 1000
- walk-length: 100
- #dimensions: 128
- neighborhood size: 7

Mining Tasks

- node classification
  - logistic regression
- node clustering
  - k-means
- similarity search
  - cosine similarity

Application 1: Multi-Class Node Classification

Table 2: Multi-class venue node classification results in AMiner data.

Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.0723	0.1396	0.1905	0.2795	0.3427	0.3911	0.4424	0.4774	0.4955	0.4457
	LINE (1st+2nd)	0.2245	0.4629	0.7011	0.8473	0.8953	0.9203	0.9308	0.9466	0.9410	0.9466
	PTE	0.1702	0.3388	0.6535	0.8304	0.8936	0.9210	0.9352	0.9505	0.9525	0.9489
	metapath2vec	0.3033	0.5247	0.8033	0.8971	0.9406	0.9532	0.9529	0.9701	0.9683	0.9670
	metapath2vec++	0.3090	0.5444	0.8049	0.8995	0.9468	0.9580	0.9561	0.9675	0.9533	0.9503
Micro-F1	DeepWalk/node2vec	0.1701	0.2142	0.2486	0.3266	0.3788	0.4090	0.4630	0.4975	0.5259	0.5286
	LINE (1st+2nd)	0.3000	0.5167	0.7159	0.8457	0.8950	0.9209	0.9333	0.9500	0.9556	0.9571
	PTE	0.2512	0.4267	0.6879	0.8372	0.8950	0.9239	0.9352	0.9550	0.9667	0.9571
	metapath2vec	0.4173	0.5975	0.8327	0.9011	0.9400	0.9522	0.9537	0.9725	0.9815	0.9857
	metapath2vec++	0.4331	0.6192	0.8336	0.9032	0.9463	0.9582	0.9574	0.9700	0.9741	0.9786

Application 1: Multi-Class Node Classification

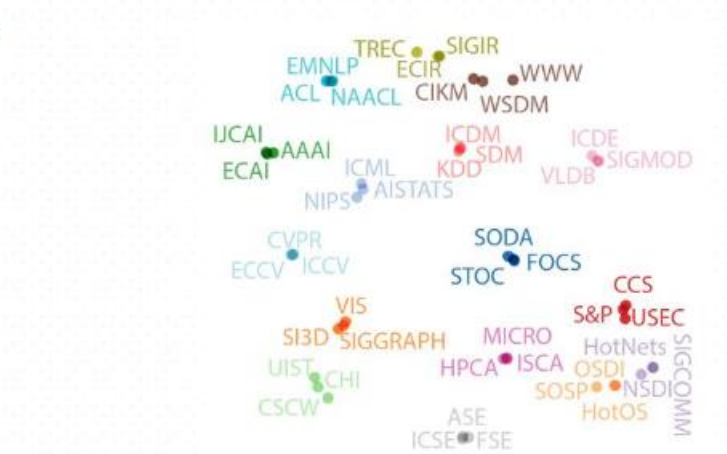
Table 3: Multi-class author node classification results in AMiner data.

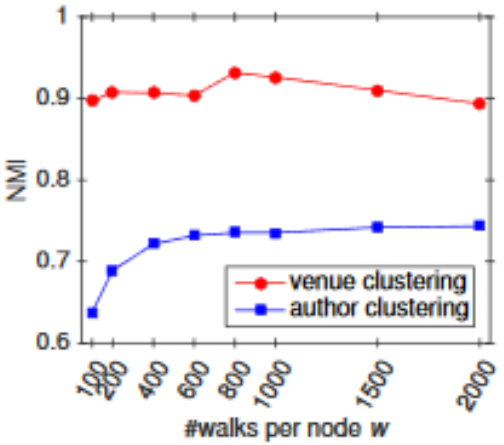
Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.7153	0.7222	0.7256	0.7270	0.7273	0.7274	0.7273	0.7271	0.7275	0.7275
	LINE (1st+2nd)	0.8849	0.8886	0.8911	0.8921	0.8926	0.8929	0.8934	0.8936	0.8938	0.8934
	PTE	0.8898	0.8940	0.897	0.8982	0.8987	0.8990	0.8997	0.8999	0.9002	0.9005
	metapath2vec	0.9216	0.9262	0.9292	0.9303	0.9309	0.9314	0.9315	0.9316	0.9319	0.9320
	metapath2vec++	0.9107	0.9156	0.9186	0.9199	0.9204	0.9207	0.9207	0.9208	0.9211	0.9212
Micro-F1	DeepWalk/node2vec	0.7312	0.7372	0.7402	0.7414	0.7418	0.7420	0.7419	0.7420	0.7425	0.7425
	LINE (1st+2nd)	0.8936	0.8969	0.8993	0.9002	0.9007	0.9010	0.9015	0.9016	0.9018	0.9017
	PTE	0.8986	0.9023	0.9051	0.9061	0.9066	0.9068	0.9075	0.9077	0.9079	0.9082
	metapath2vec	0.9279	0.9319	0.9346	0.9356	0.9361	0.9365	0.9365	0.9365	0.9367	0.9369
	metapath2vec++	0.9173	0.9217	0.9243	0.9254	0.9259	0.9261	0.9261	0.9262	0.9264	0.9266

Application 2: Node Clustering

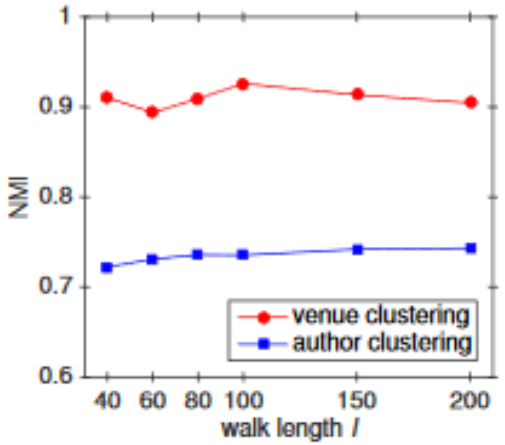
Node clustering results (NMI) in AMiner

methods	venue	author
DeepWalk/node2vec	0.1952	0.2941
LINE (1st+2nd)	0.8967	0.6423
PTE	0.9060	0.6483
metapath2vec	0.9274	0.7470
metapath2vec++	0.9261	0.7354

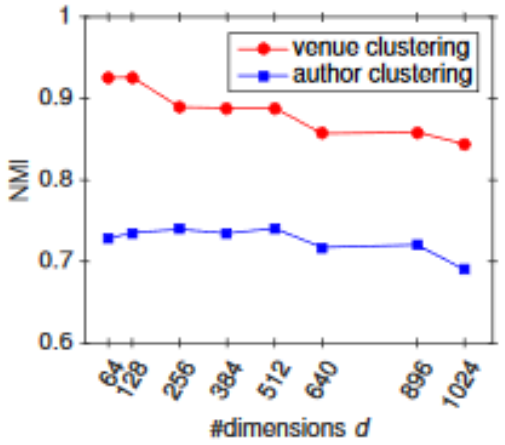




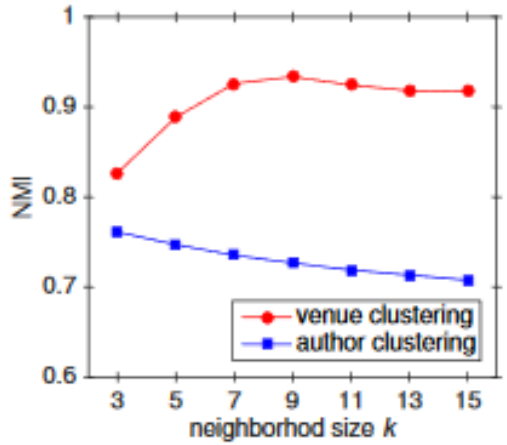
(a) #walks per node w



(b) walk length l



(c) #dimensions d



(d) neighborhood size k

Application 3: Similarity Search

Table 5: Case study of similarity search in AMiner Data

Rank	ACL	NIPS	IJCAI	CVPR	FOCS	SOSP	ISCA	S&P	ICSE	SIGGRAPH	SIGCOMM	CHI	KDD	SIGMOD	SIGIR	WWW
0	ACL	NIPS	IJCAI	CVPR	FOCS	SOSP	ISCA	S&P	ICSE	SIGGRAPH	SIGCOMM	CHI	KDD	SIGMOD	SIGIR	WWW
1	EMNLP	ICML	AAAI	ECCV	STOC	TOCS	HPCA	CCS	TOSEM	TOG	CCR	CSCW	SDM	PVLDB	ECIR	WSDM
2	NAACL	AISTATS	AI	ICCV	SICOMP	OSDI	MICRO	NDSS	FSE	SI3D	HotNets	TOCHI	TKDD	ICDE	CIKM	CIKM
3	CL	JMLR	JAIR	IJCV	SODA	HotOS	ASPLOS	USENIX S	ASE	RT	NSDI	UIST	ICDM	DE Bull	IR J	TWEB
4	CoNLL	NC	ECAI	ACCV	A-R	SIGOPS E	PACT	ACSAC	ISSTA	CGF	CoNEXT	DIS	DMKD	VLDBJ	TREC	ICWSM
5	COLING	MLJ	KR	CVIU	TALG	ATC	ICS	JCS	E SE	NPAR	IMC	HCI	KDD E	EDBT	SIGIR F	HT
6	IJCNLP	COLT	AI Mag	BMVC	ICALP	NSDI	HiPEAC	ESORICS	MSR	Vis	TON	MobileHCI	WSDM	TODS	ICTIR	SIGIR
7	NLE	UAI	ICAPS	ICPR	ECCC	OSR	PPOPP	TISS	ESEM	JGT	INFOCOM	INTERACT	CIKM	CIDR	WSDM	KDD
8	ANLP	KDD	CI	EMMCVPR	TOC	ASPLOS	ICCD	ASIACCS	A SE	VisComp	PAM	GROUP	PKDD	SIGMOD R	TOIS	TIT
9	LREC	CVPR	AIPS	T on IP	JAlG	EuroSys	CGO	RAID	ICPC	GI	MobiCom	NordiCHI	ICML	WebDB	IPM	WISE
10	EACL	ECML	UAI	WACV	ITCS	SIGCOMM	ISLPED	CSFW	WICSA	CG	IPTPS	UbiComp	PAKDD	PODS	AIRS	WebSci

JU&ST

CIKM18:Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings

Metapath2vec:Challenges

1. The number of possible meta-paths can be large (exponentially growing with their length).
2. How to select meta-paths from a given heterogeneous graph remains unclear.
3. The choice of meta-paths highly affect the quality of the learnt node embeddings.

"Are meta-paths really necessary for heterogeneous graph embed-dings?"

Why meta-path?

The initial motivation of using meta-paths in heteroge-neous graph embeddings lies in the fact that random walks on heterogeneous graphs are biased to highly visible types (domains)of nodes, where nodes are associated with a dominant number of paths.

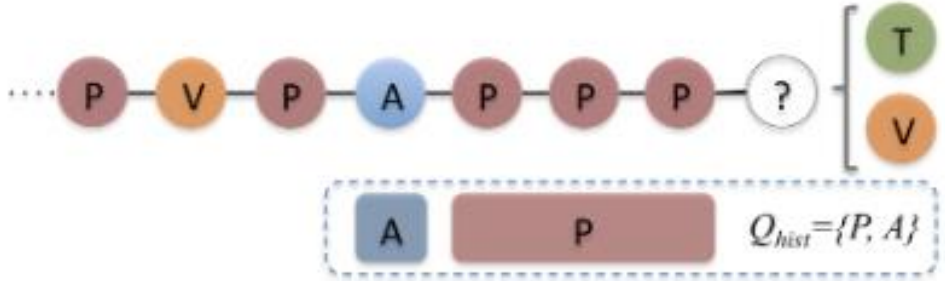
In other words, the node distribution from the randomwalk sequences are skewed towards these highly visible domains.Subsequently, the learnt node embeddings are also biased to these domains in the sense that they mostly preserve the node proximityin these domains.

JU&ST

Hyper parameters
α ↓ more E_{he}
m ↓ more 'randomness'

Random walk with JUmp or Stay strategies.
(1) JUmp or STay with given probabilities.
(2) If jump, decide where to jump. (Using exponential decay function)
(3) Choosing where to jump. (Memory the last m visited domains)

$$Pr_{stay}(v_i) = \begin{cases} 0, \text{ if } V_{stay}(v_i) = \emptyset \\ 1, \text{ if } \{V^q_{jump}(v_i) | q \in Q, q \neq \phi(v_i)\} = \emptyset \\ \alpha^l, \text{ otherwise} \end{cases} \quad (1)$$

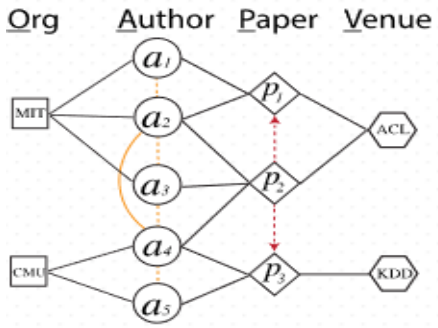


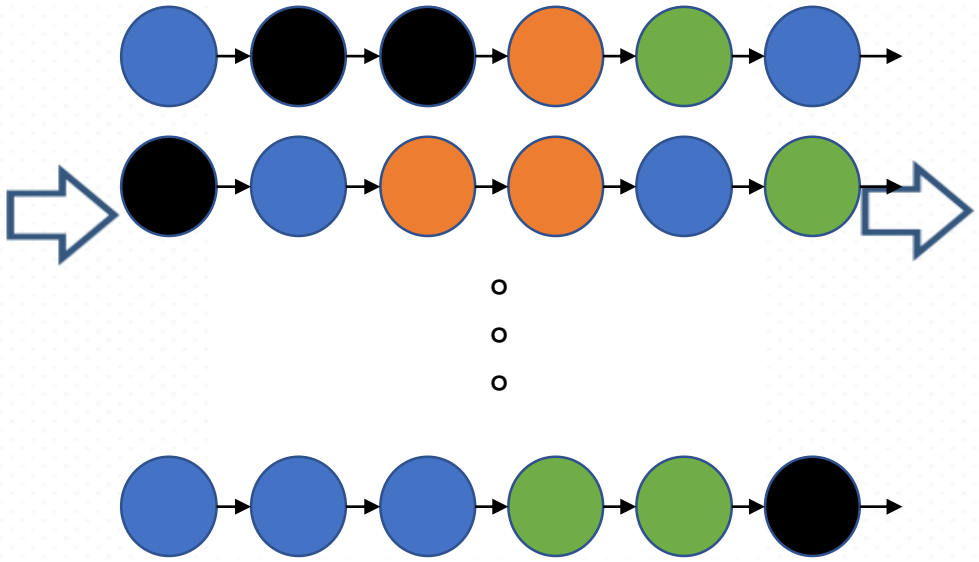
m=2

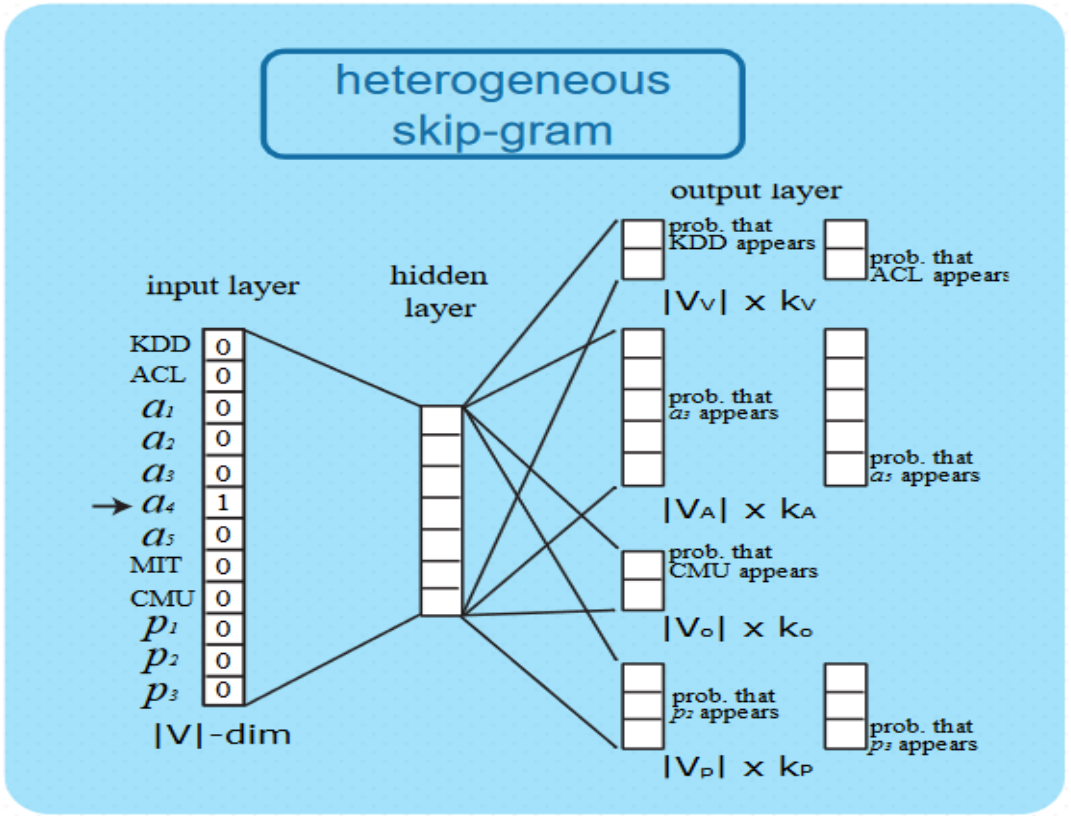
$$Q_{Jump}(v_i) = \begin{cases} \{q | q \in Q \wedge q \notin Q_{hist}, V^q_{jump}(v_i) \neq \emptyset\}, \text{ if not empty} \\ \{q | q \in Q, q \neq \phi(v_i), V^q_{jump}(v_i) \neq \emptyset\}, \text{ otherwise} \end{cases} \quad (2)$$

JU&ST

JU&ST Random Walks







Datasets

Dataset	DBLP	Movie	Foursquare
$\|V\|$	15,649	21,345	29,771
$\|E\|$	51,377	89,038	83,407
$\|E_{he}\|$	44,379	34,354	77,712
$\|E_{ho}\|$	6,998	54,684	5,695
#Labels	4	5	10

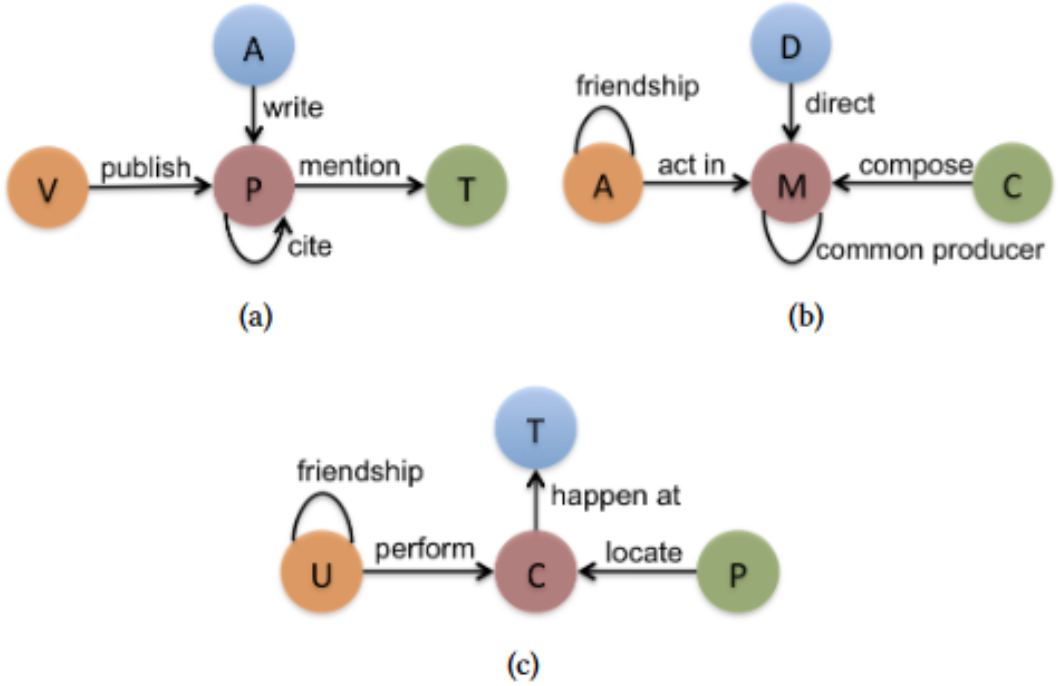


Figure 1: Structures of three heterogeneous graphs: a) a DBLP graph with four data domains: Author (A), Paper (P), Venue (V) and Topic (T); b) a movie graph with four data domains: Director (D), Movie (M), Actor (A) and Composer (C); c) a Foursquare graph with four data domains: User (U), Point of interest (P), Check-in (C) and Time slot (T).

Application 1: Node Classification

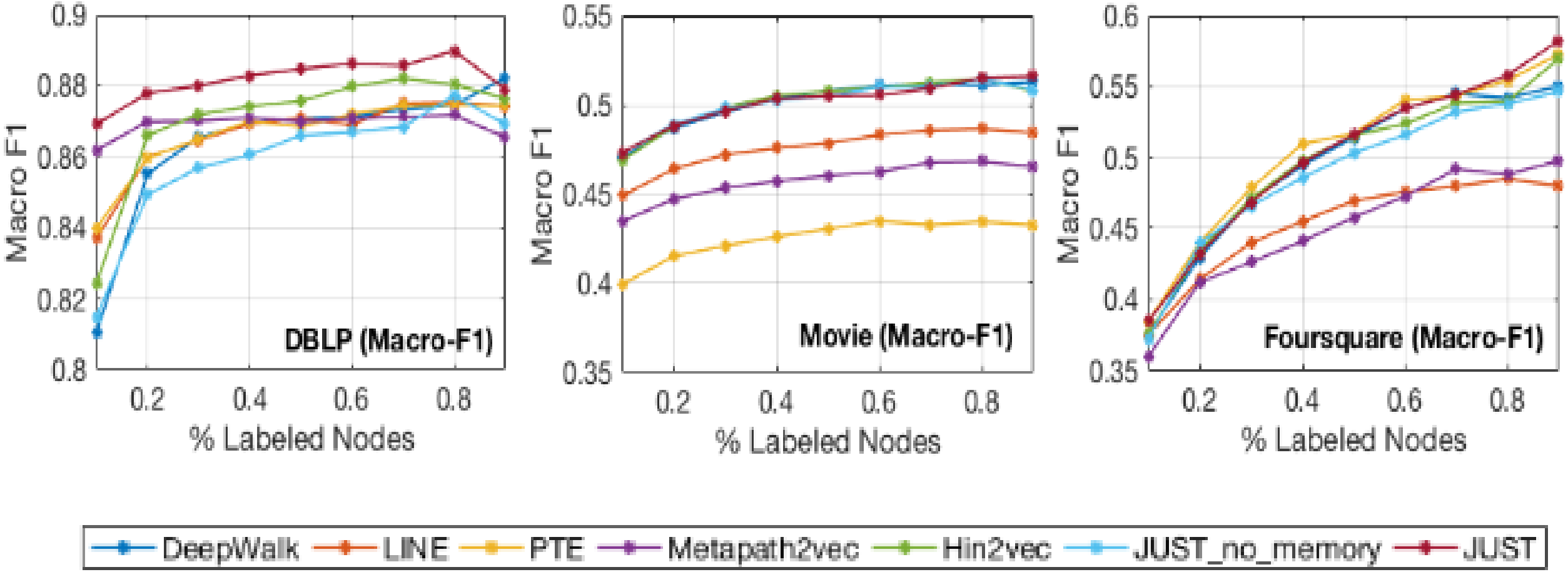


Figure 4: Performance on node classification task

Application 2: Node Clustering

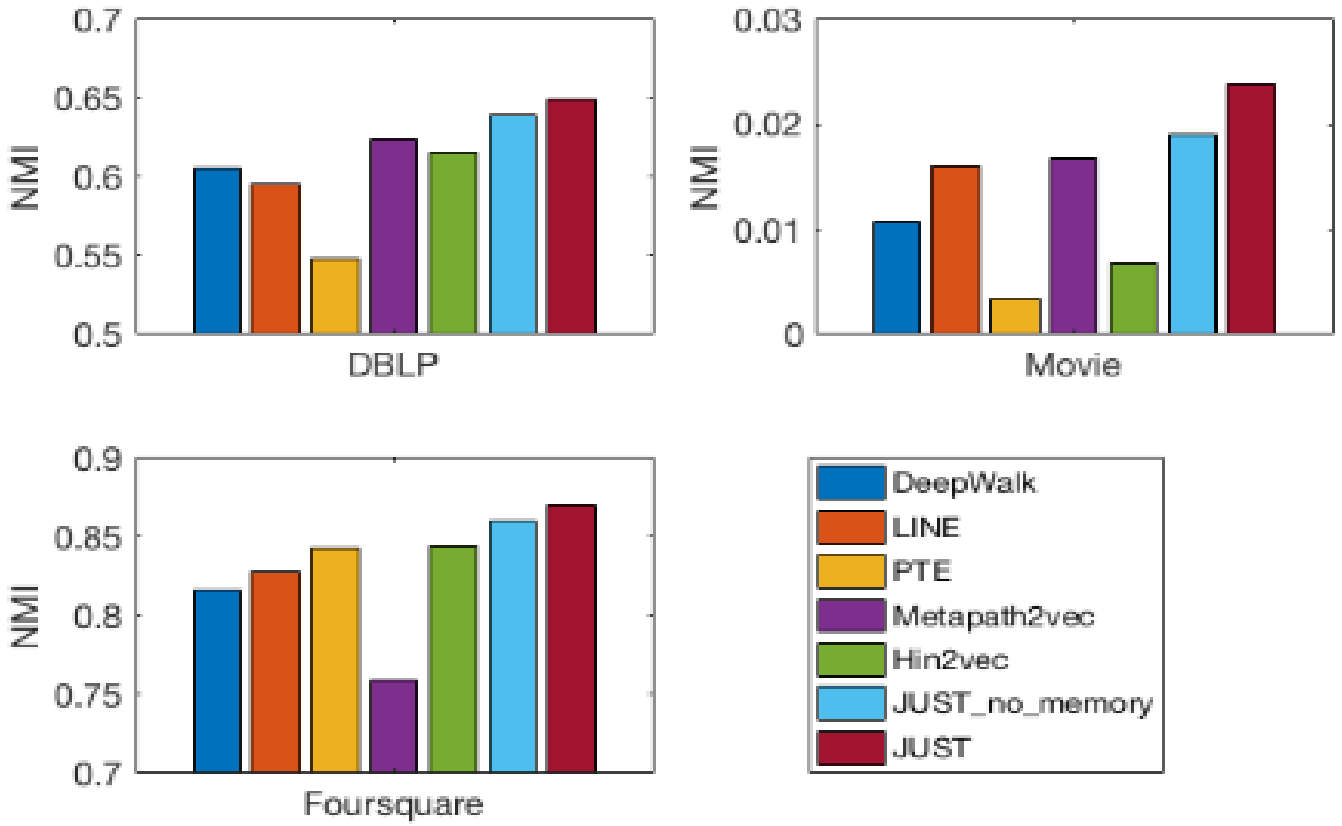
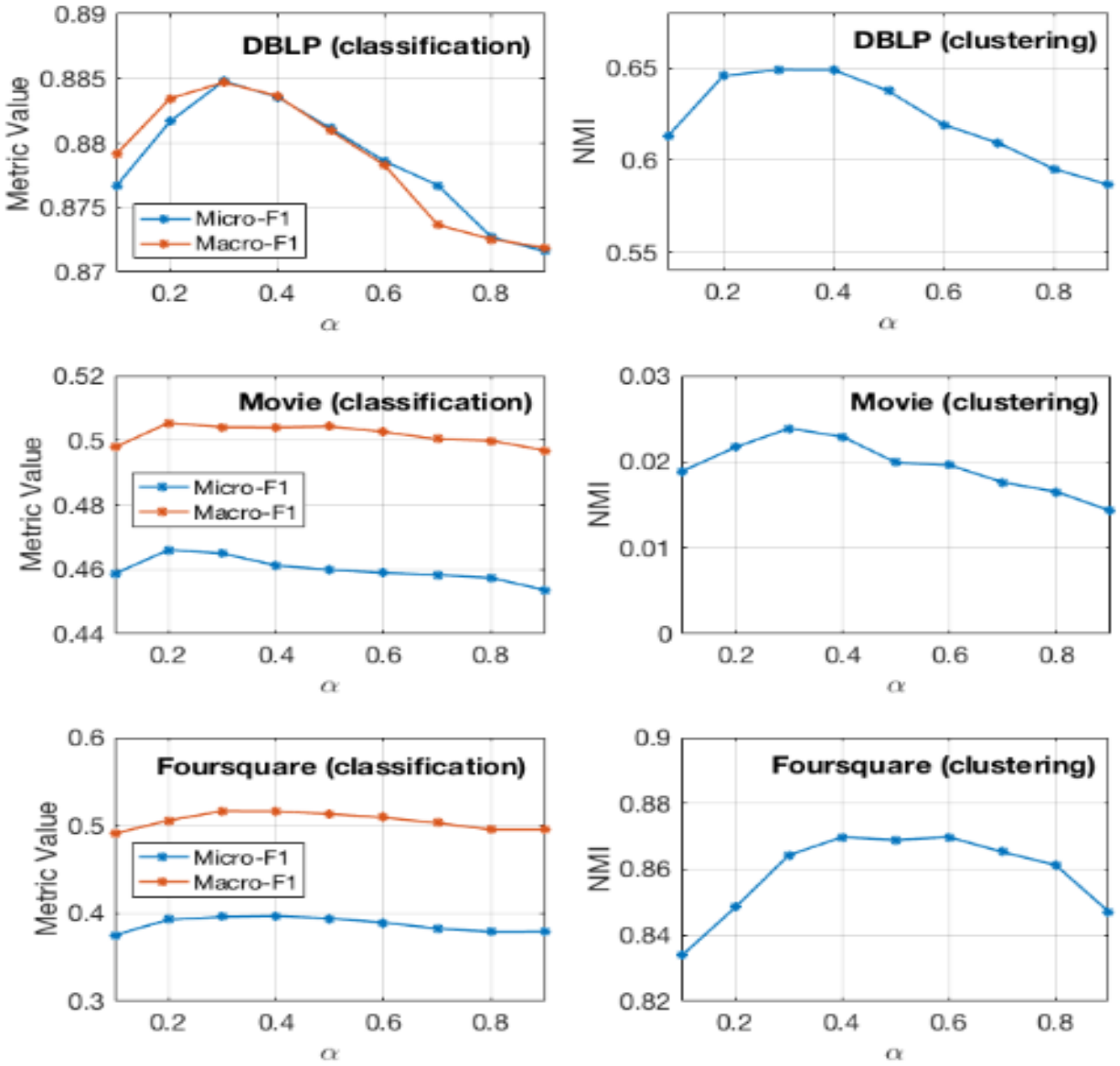
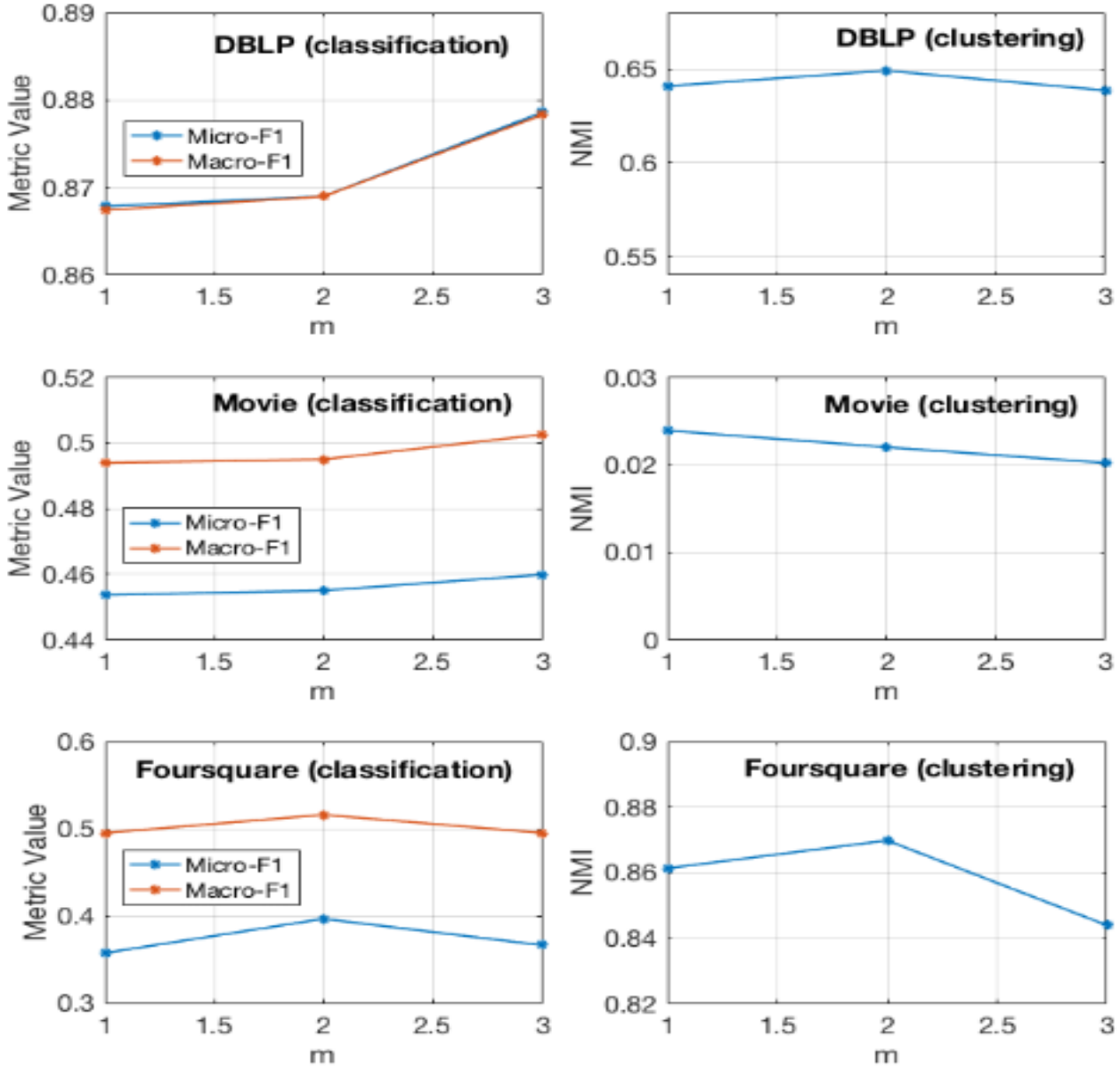


Figure 5: Performance on node clustering task

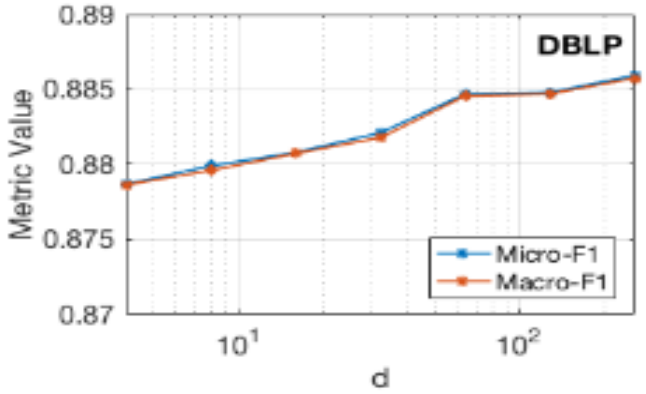
Hyper parameters: Initial Stay Probability α

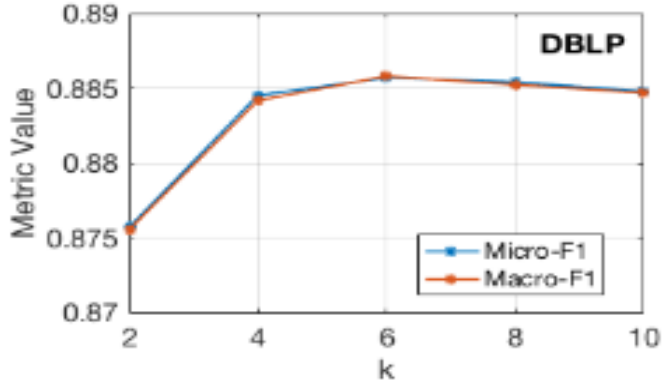


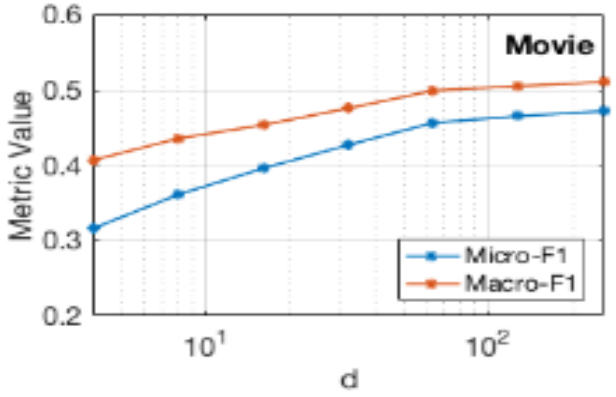
Hyper parameters : number of Memorized Domains m

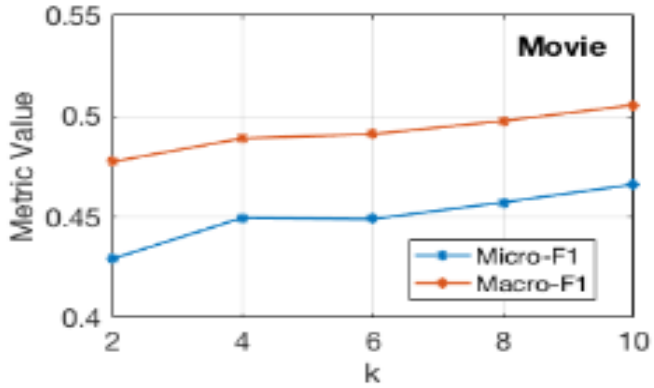


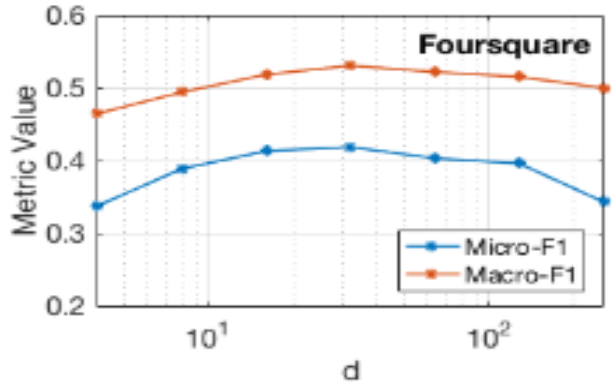
Hyper parameters: d and k

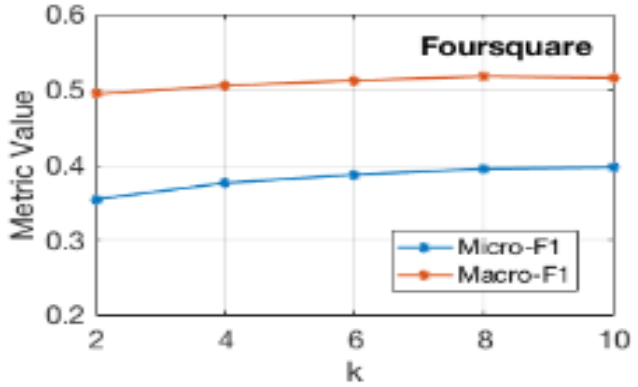












Runtime Performance

	DBLP	Movie	Foursquare
DeepWalk	236	333	484
Metapath2vec (original)	965	19,200	2,248
Metapath2vec (ours)	290	408	550
Hin2vec	904	1,301	1,801
JUST	310	442	616

JUST. In particular, we note that the original implementation of Metapath2vec released by the authors is highly customized to the DBLP graph (the only dataset used in the paper), and also takes long to run even on a small graph. Therefore, we implemented our own version of Metapath2vec (ours) using the efficient Gensim library[5] for the SkipGram training process.

Discussion

In this study , we evaluate our proposed technique using several widely used heterogeneous graphs, which do not involve complex structures. However, real-world heterogeneous graphs often have complex structures, such as a large number of domains, or multi-relational edges (hyper edges simultaneously linking multiple nodes). For example, Knowledge Graphs often contain a large number of node domains (types). Therefore, it is interesting to investigate how our proposed technique without meta-paths performs on those complex heterogeneous graphs.

The Basics: Graph Neural Networks

Based on material from:

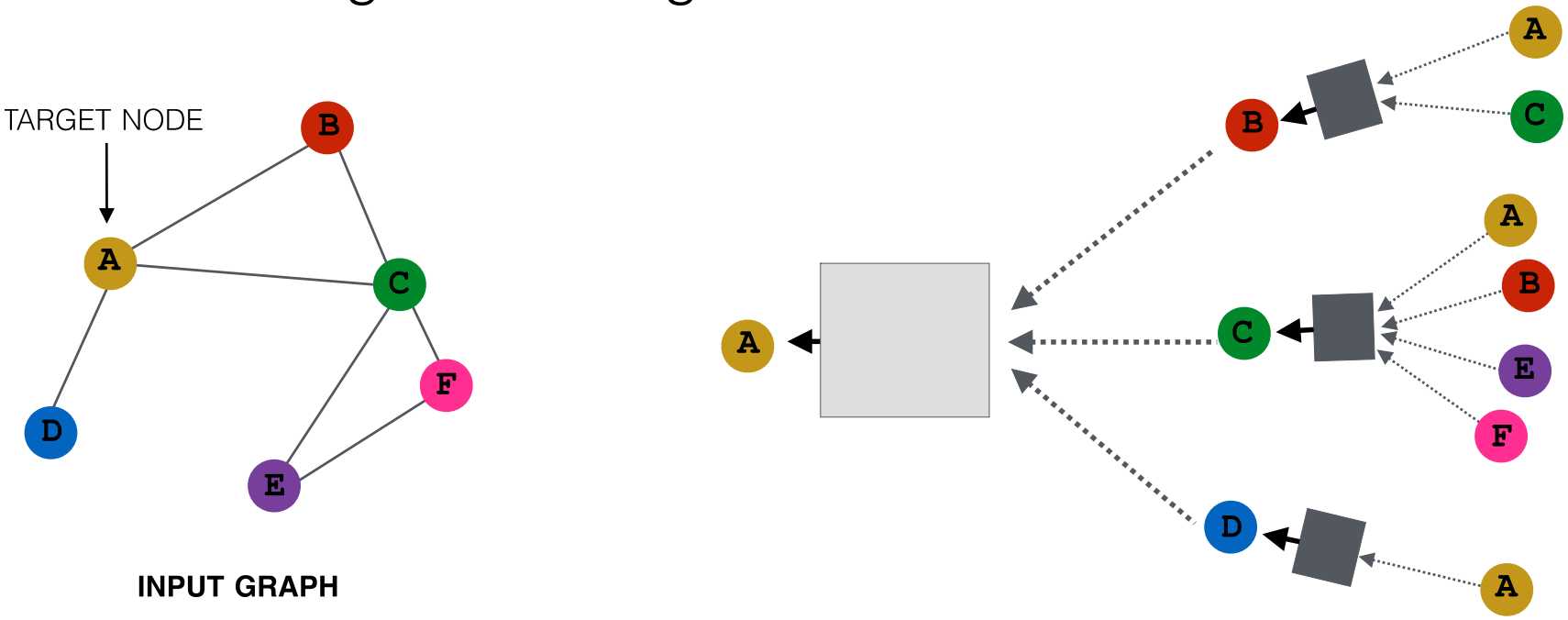
- Hamilton et al. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin on Graph Systems*.
- Scarselli et al. 2005. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*.

Setup

- Assume we have a graph $\mathbf{G}$:
  - $\mathbf{V}$ is the vertex set.
  - $\mathbf{A}$ is the adjacency matrix (assume binary).
  - **$\mathbf{X} \in \mathbf{R}^{m \times |V|}$ is a matrix of node features.**
    - Categorical attributes, text, image data
      - E.g., profile information in a social network.
    - Node degrees, clustering coefficients, etc.
    - Indicator vectors (i.e., one-hot encoding of each node)

Neighborhood Aggregation

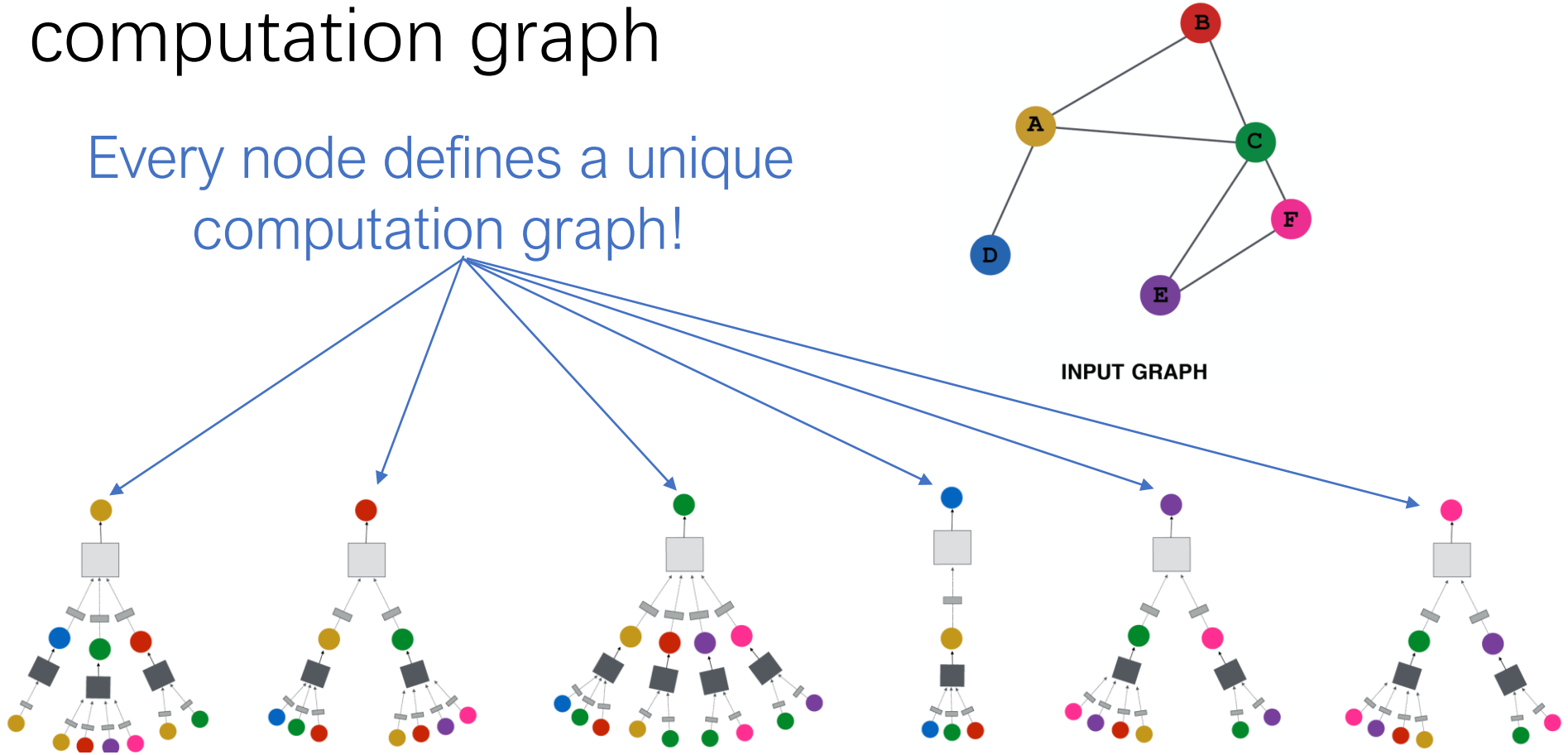
- **Key idea:** Generate node embeddings based on local neighborhoods.
- **Intuition:** Nodes aggregate information from their neighbors using neural networks



Neighborhood Aggregation

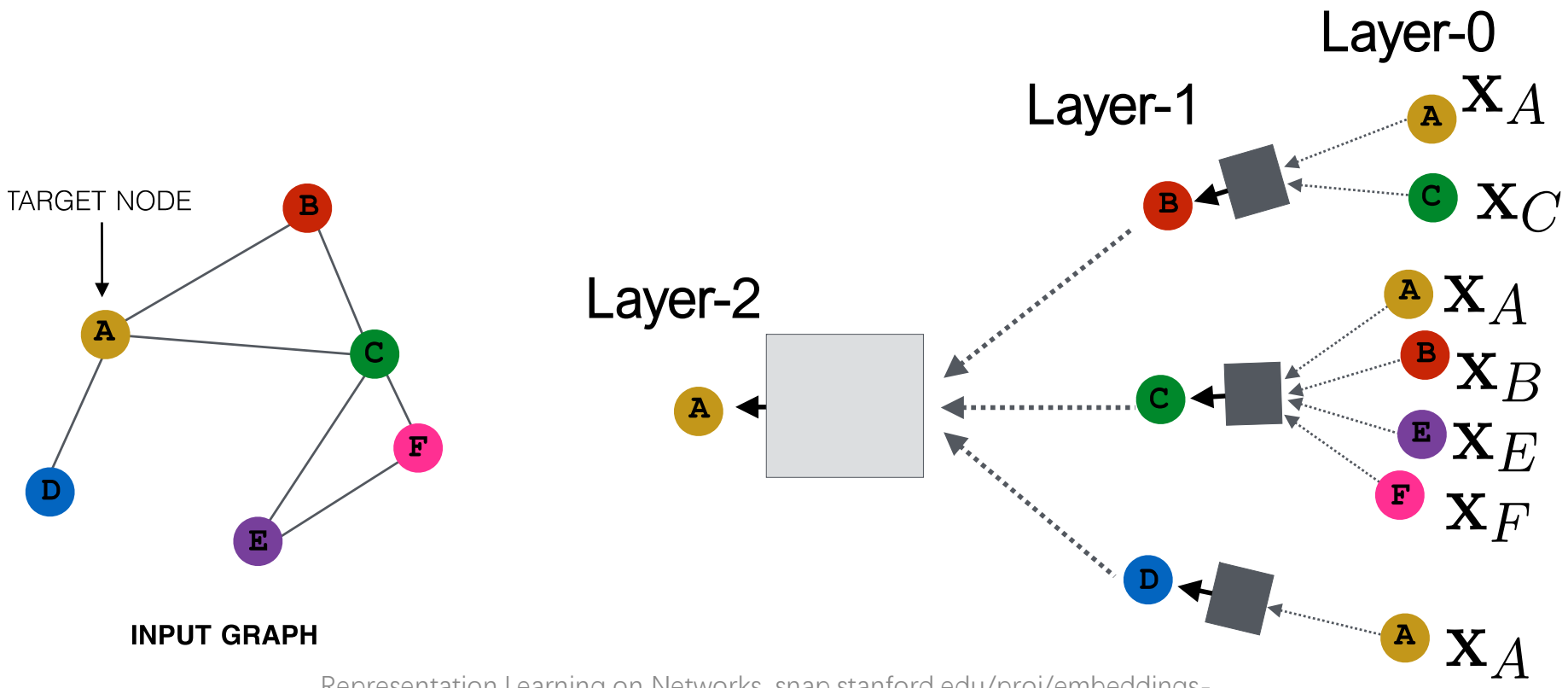
- **Intuition:** Network neighborhood defines a computation graph

Every node defines a unique computation graph!



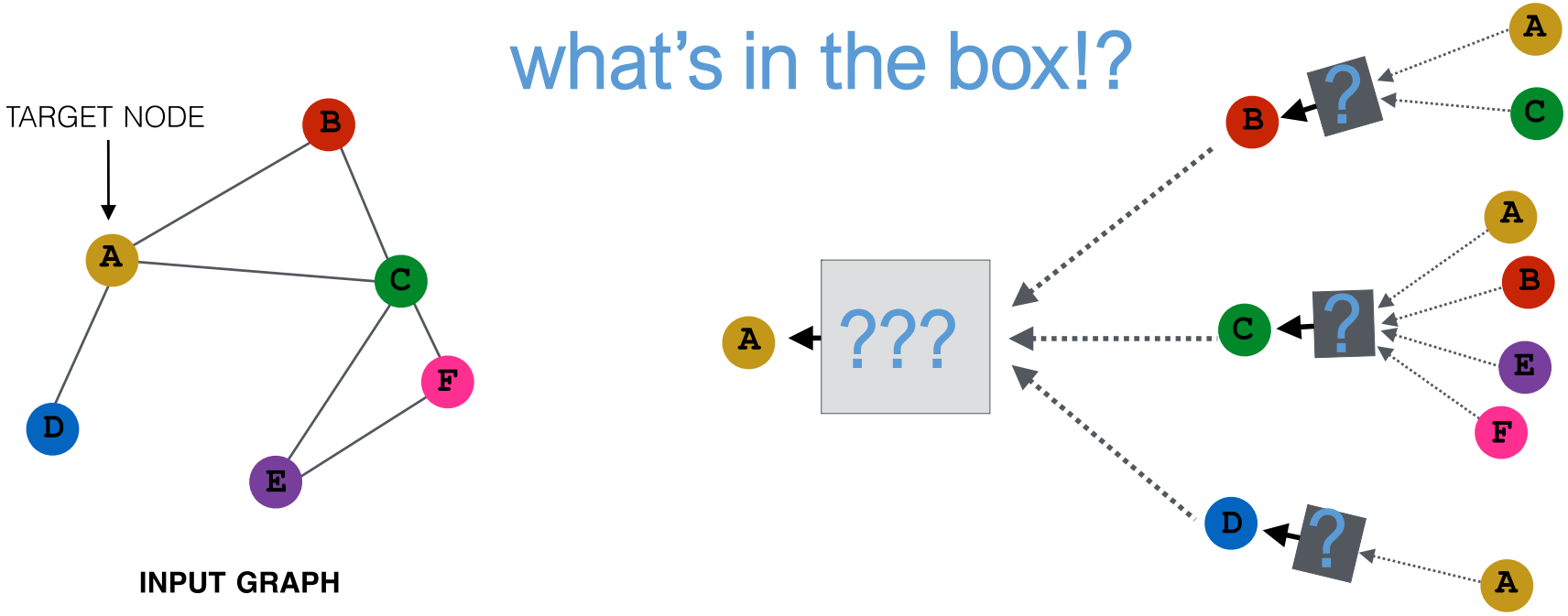
Neighborhood Aggregation

- Nodes have embeddings at each layer.
- Model can be arbitrary depth.
- "layer-0" embedding of node $\mathbf{u}$ is its input feature, i.e. $\mathbf{x}_u$.



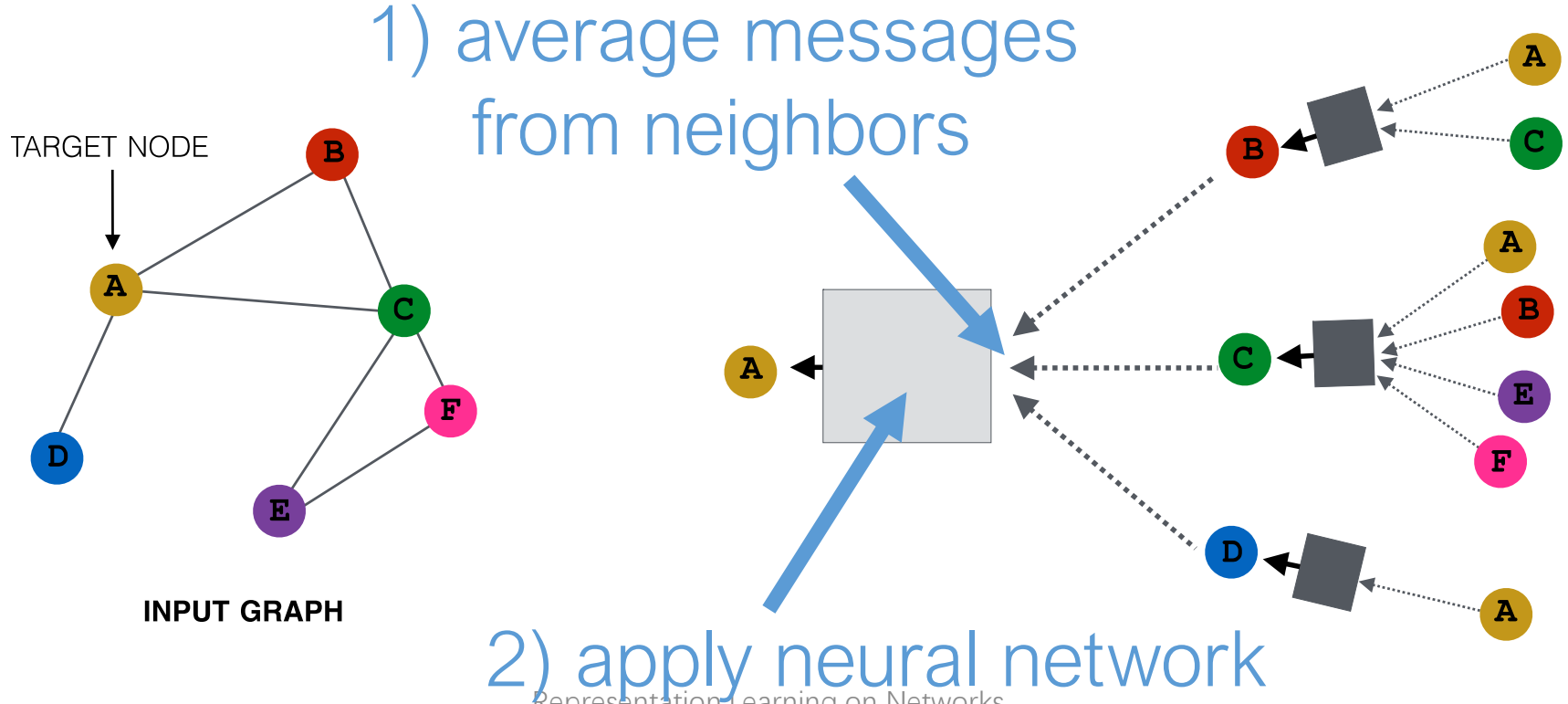
Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate information across the layers.



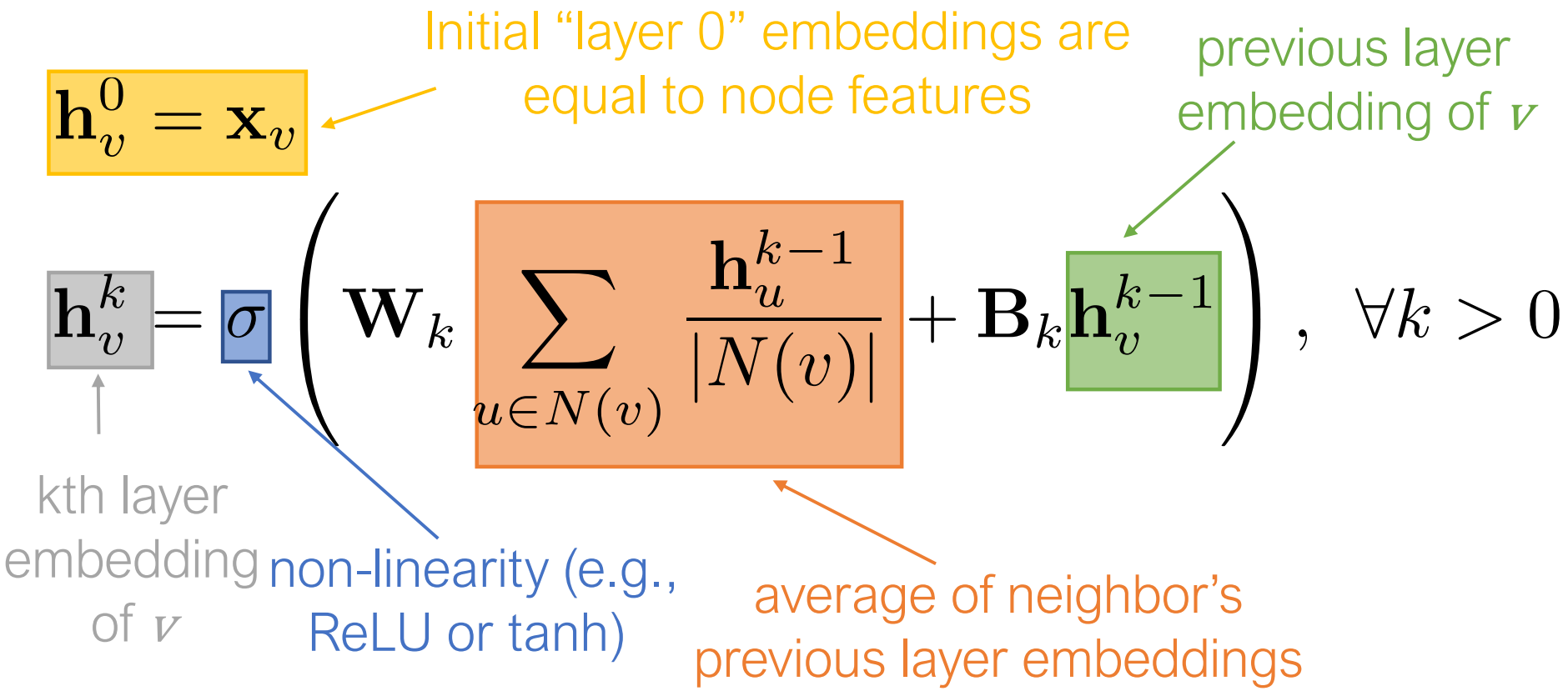
Neighborhood Aggregation

- **Basic approach:** Average neighbor information and apply a neural network.



The Math

- **Basic approach:** Average neighbor messages and apply a neural network.



Training the Model

trainable matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

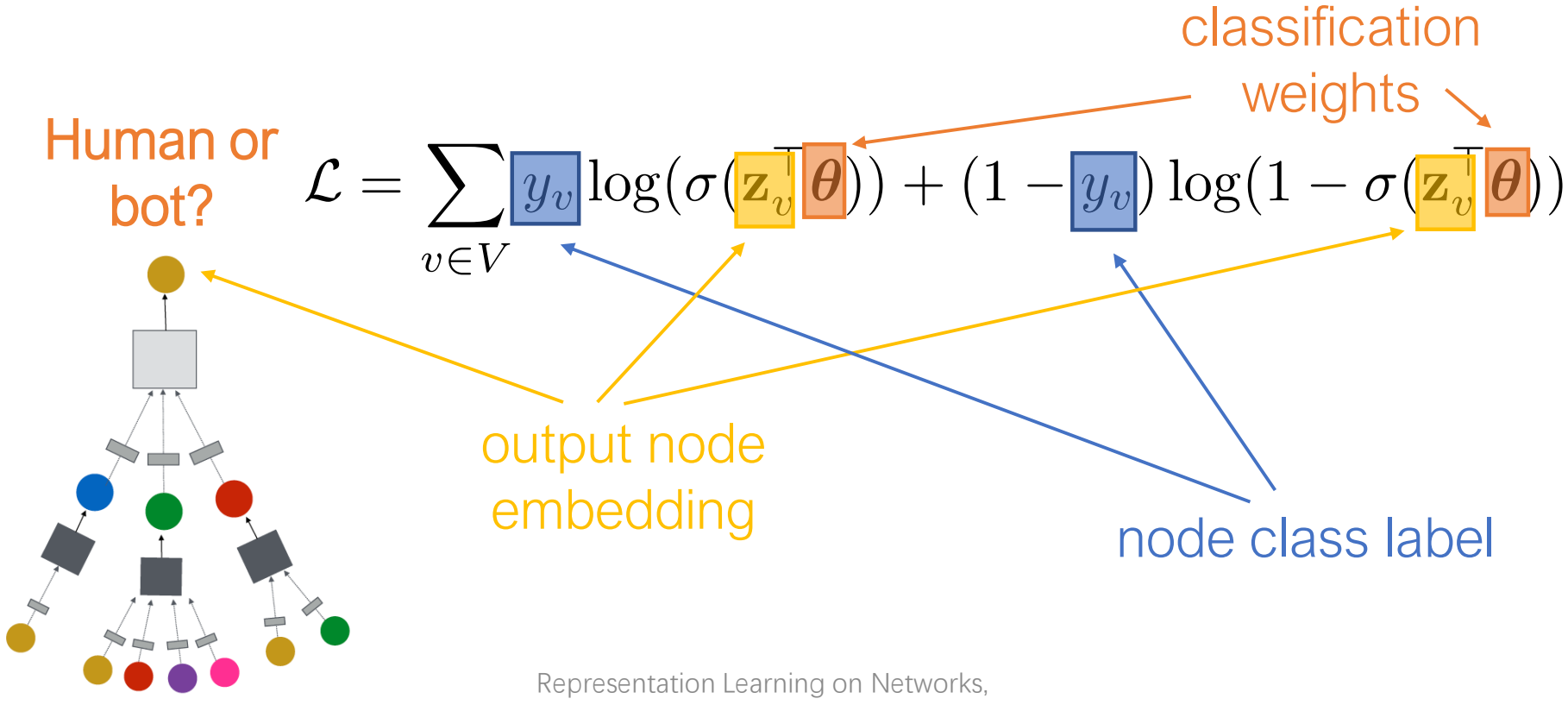
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

- After K-layers of neighborhood aggregation, we get output embeddings for each node.
- **We can feed these embeddings into any loss function** and run stochastic gradient descent to train the aggregation parameters.

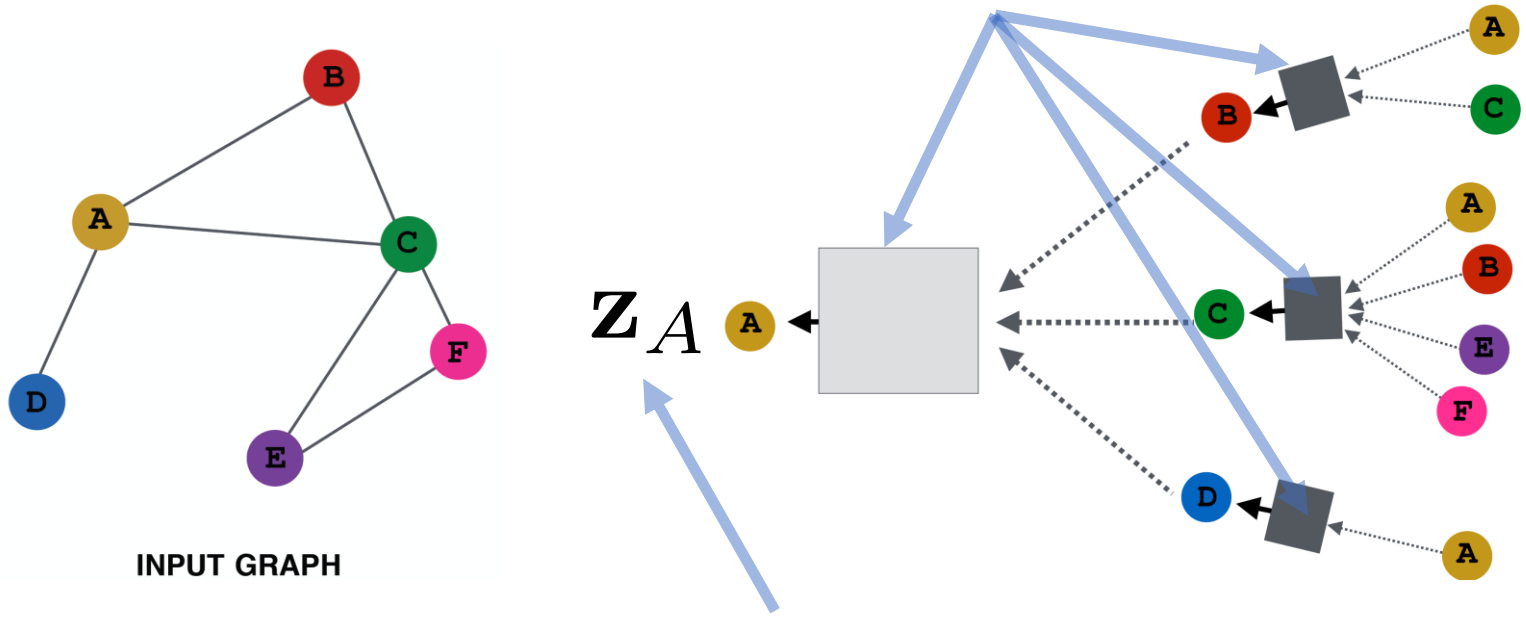
Training the Model

- **Alternative**: Directly train the model for a supervised task (e.g., node classification):



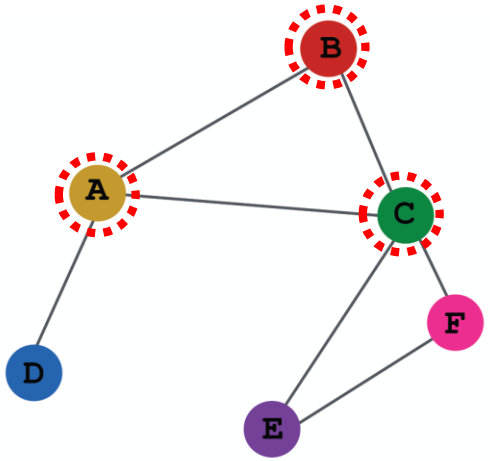
Overview of Model Design

1) Define a neighborhood aggregation function.



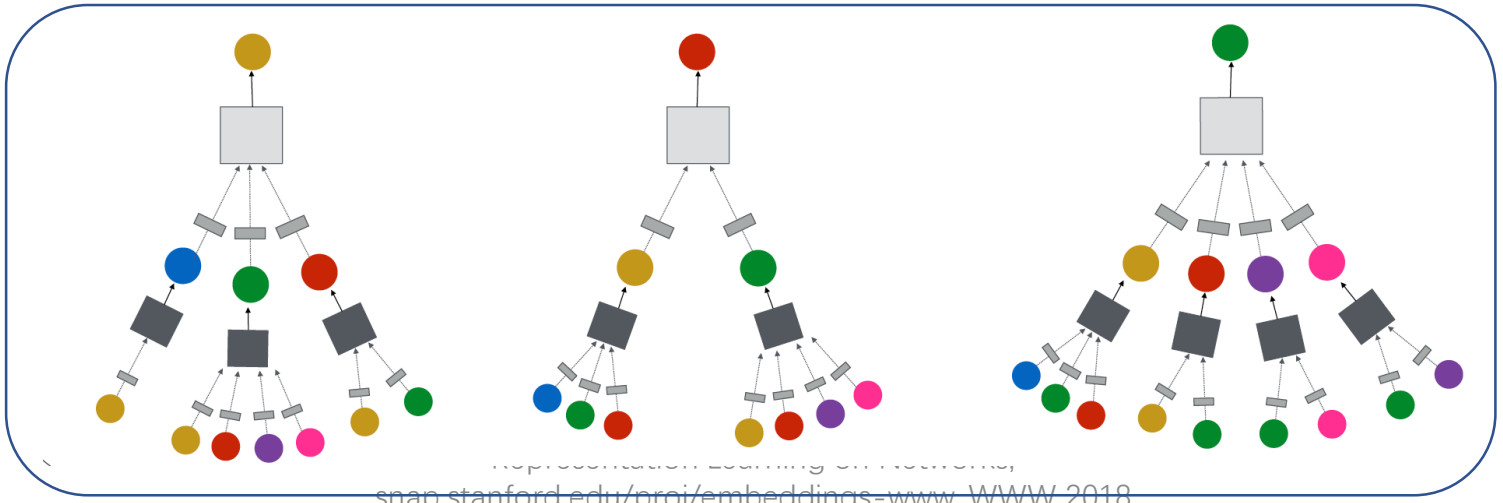
2) Define a loss function on the embeddings, $\mathbf{L}(\mathbf{z}_u)$

Overview of Model Design

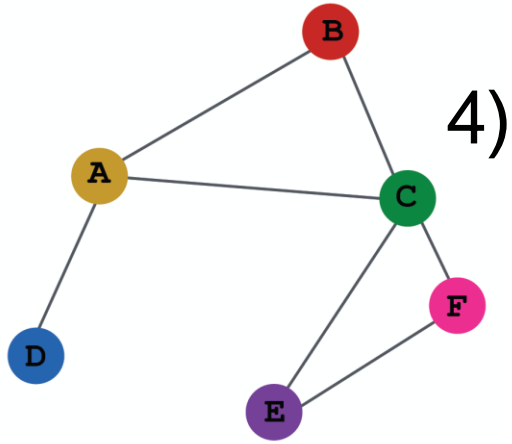


INPUT GRAPH

3) Train on a set of nodes, i.e., a batch of compute graphs



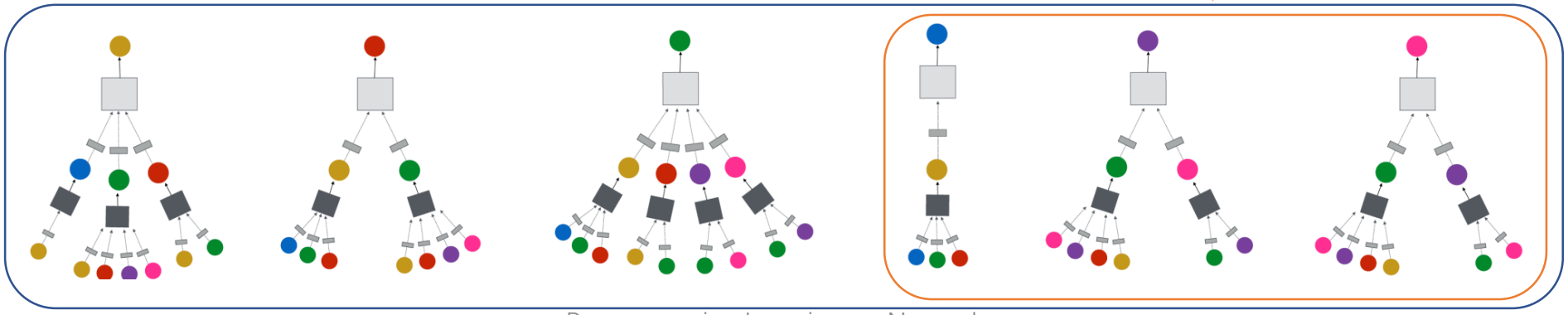
Overview of Model



INPUT GRAPH

4) Generate embeddings for nodes as needed

Even for nodes we never trained on!!!!



Graph Convolutional Networks

Basic Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

VS.

GCN Neighborhood Aggregation

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

same matrix for self and neighbor embeddings

per-neighbor normalization

Graph Convolutional Networks

- Empirically, they found this configuration to give the best results.
  - More parameter sharing.
  - Down-weights high degree neighbors.

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

use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

Representation Learning on Networks, snap.stanford.edu/proj/embeddings-www, WWW 2018

KGCN

WWW19:Knowledge Graph Convolutional Networks for Recommender Systems

Motivation

Inspired by graph convolutional networks (GCN) that try to generalize convolution to the graph domain, we propose Knowledge Graph Convolutional Networks (KGCN) for recommender systems.

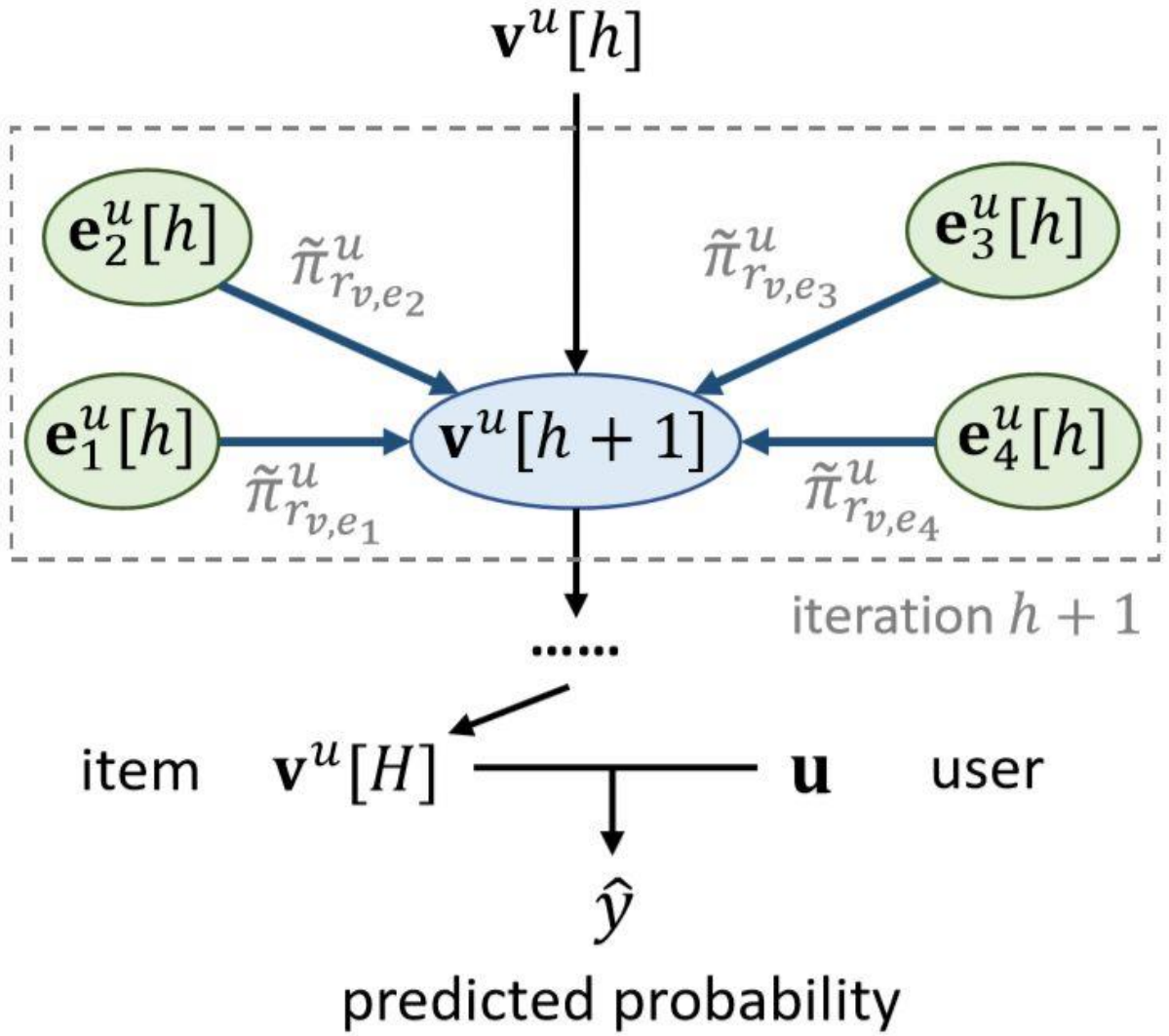
The key idea of KGCN is to aggregate and incorporate neighborhood information with bias when calculating the representation of a given entity in the KG.

Advantages

Such a design has two advantages: (1) Through the operation of neighborhood aggregation, the local proximity structure is successfully captured and stored in each entity. (2) Neighbors are weighted by scores dependent on the connecting relation and specific user, which characterizes both the semantic information of KG and users' personalized interests in relations.

Note that the size of an entity's neighbors varies and may be prohibitively large in the worst case. Therefore, we sample a fixed-size neighborhood of each node as the receptive field, which makes the cost of KGCN predictable.

KGCN model


$\mathbf{v}^u[h]$
$\mathbf{e}_2^u[h]$
$\tilde{\pi}^u_{r_{v,e_2}}$
$\tilde{\pi}^u_{r_{v,e_3}}$
$\mathbf{e}_3^u[h]$
$\mathbf{e}_1^u[h]$
$\mathbf{v}^u[h+1]$
$\mathbf{e}_4^u[h]$
$\tilde{\pi}^u_{r_{v,e_1}}$
$\tilde{\pi}^u_{r_{v,e_4}}$
iteration $h+1$
.....
item
$\mathbf{v}^u[H]$
$\mathbf{u}$
user
$\hat{y}$
predicted probability

KGCN model

Use a function g : Rd ×Rd → R (e.g., inner product) to compute the score between a user and a relation,

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

where u ∈ Rd and r ∈ Rd are the representations of user u and relation r , respectively, d is the dimension of representations. In general, πu r characterizes the importance of relation r to user u.

For example, a user may have more potential interests in the movies that share the same “star" with his historically liked ones, while another user may be more concerned about the “genre" of movies.

KGCN model

To characterize the topological proximity structure of item v, we compute the linear combination of v's neighborhood:

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

Where:

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

KGCN model

Three types of aggregators:

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

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

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

Learning algorithm

The final H-order entity representation is denoted as v^u which is fed into a function f : Rd × Rd ⟶ R （inner product） together with user representation u for predicting the probability: $\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}^u)$

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

$$\mathcal{J}(y_{uv}, \hat{y}_{uv}) = \mathcal{J}(1, f(u, v^u))$$

$$= \mathcal{J}(1, f(u, \sum_{e \in S(v)} \hat{\pi}^u_{r_{v,e}} e))$$

$$= \mathcal{J}(1, f(u, \sum_{e \in S(v)} \frac{\exp(g(u,r))}{\sum_{e \in N(v)} \exp(g(u,r))} e))$$

f, g : inner products $\mathcal{J}$: cross-entropy

P: n-samples distribution. T^u: number of n-samples

Learning algorithm

Algorithm 1: KGCN algorithm

Input: Interaction matrix Y; knowledge graph $\mathcal{G}(\mathcal{E}, \mathcal{R})$;
neighborhood sampling mapping $\mathcal{S} : e \rightarrow 2^{\mathcal{E}}$; trainable parameters: $\{\mathbf{u}\}_{u \in \mathcal{U}}, \{\mathbf{e}\}_{e \in \mathcal{E}}, \{\mathbf{r}\}_{r \in \mathcal{R}}, \{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^{H}$;
hyper-parameters: $H, d, g(\cdot), f(\cdot), \sigma(\cdot), agg(\cdot)$

Output: Prediction function $\mathcal{F}(u, v|\Theta, \mathrm{Y}, \mathcal{G})$

while *KGCN not converge* **do**
  for (u, v) *in* Y **do**
    $\{\mathcal{M}[i]\}_{i=0}^{H} \leftarrow$ Get-Receptive-Field (v);
    $\mathbf{e}^u[0] \leftarrow \mathbf{e}, \forall e \in \mathcal{M}[0]$;
    for $h = 1, ..., H$ **do**
      for $e \in \mathcal{M}[h]$ **do**
        $\mathbf{e}^u_{\mathcal{S}(e)}[h-1] \leftarrow \sum_{e' \in \mathcal{S}(e)} \tilde{\pi}^u_{r_{e,e'}} \mathbf{e}'^u[h-1]$;
        $\mathbf{e}^u[h] \leftarrow agg\left(\mathbf{e}^u_{\mathcal{S}(e)}[h-1], \mathbf{e}^u[h-1]\right)$;
    $\mathbf{v}^u \leftarrow \mathbf{e}^u[H]$;
    Calculate predicted probability $\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}^u)$;
    Update parameters by gradient descent;
return $\mathcal{F}$;

Function Get-Receptive-Field (v)
  $\mathcal{M}[H] \leftarrow v$;
  for $h = H-1, ..., 0$ **do**
    $\mathcal{M}[h] \leftarrow \mathcal{M}[h+1]$;
    for $e \in \mathcal{M}[h+1]$ **do**
      $\mathcal{M}[h] \leftarrow \mathcal{M}[h] \cup \mathcal{S}(e)$;
  return $\{\mathcal{M}[i]\}_{i=0}^{H}$;

Experiments

Datasets:

K: neighbor sampling ; d: dim; H: depth; λ: coefficient of L2

	MovieLens-20M	Book-Crossing	Last.FM
# users	138,159	19,676	1,872
# items	16,954	20,003	3,846
# interactions	13,501,622	172,576	42,346
# entities	102,569	25,787	9,366
# relations	32	18	60
# KG triples	499,474	60,787	15,518
K	4	8	8
d	32	64	16
H	2	1	1
λ	10^{-7}	2×10^{-5}	10^{-4}
learning rate	2×10^{-2}	2×10^{-5}	5×10^{-4}

Experiments

The results of AUC and F1 in CTR prediction.

Model	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	*F1*	*AUC*	*F1*	*AUC*	*F1*
SVD	0.963 (-1.5%)	0.919 (-1.4%)	0.672 (-8.9%)	0.635 (-7.7%)	0.769 (-3.4%)	0.696 (-3.5%)
LibFM	0.959 (-1.9%)	0.906 (-2.8%)	0.691 (-6.4%)	0.618 (-10.2%)	0.778 (-2.3%)	0.710 (-1.5%)
LibFM + TransE	0.966 (-1.2%)	0.917 (-1.6%)	0.698 (-5.4%)	0.622 (-9.6%)	0.777 (-2.4%)	0.709 (-1.7%)
PER	0.832 (-14.9%)	0.788 (-15.5%)	0.617 (-16.4%)	0.562 (-18.3%)	0.633 (-20.5%)	0.596 (-17.3%)
CKE	0.924 (-5.5%)	0.871 (-6.5%)	0.677 (-8.3%)	0.611 (-11.2%)	0.744 (-6.5%)	0.673 (-6.7%)
RippleNet	0.968 (-1.0%)	0.912 (-2.1%)	0.715 (-3.1%)	0.650 (-5.5%)	0.780 (-2.0%)	0.702 (-2.6%)
KGCN-sum	**0.978**	**0.932**	**0.738**	**0.688**	0.794 (-0.3%)	0.719 (-0.3%)
KGCN-concat	0.977 (-0.1%)	0.931 (-0.1%)	0.734 (-0.5%)	0.681 (-1.0%)	**0.796**	**0.721**
KGCN-neighbor	0.977 (-0.1%)	**0.932**	0.728 (-1.4%)	0.679 (-1.3%)	0.781 (-1.9%)	0.699 (-3.1%)
KGCN-avg	0.975 (-0.3%)	0.929 (-0.3%)	0.722 (-2.2%)	0.682 (-0.9%)	0.774 (-2.8%)	0.692 (-4.0%)

Experiments

AUC result of KGCN with different neighbor sampling size K.

K	2	4	8	16	32	64
movie	0.978	**0.979**	0.978	0.978	0.977	0.978
book	0.680	0.727	**0.736**	0.725	0.711	0.723
music	0.791	0.794	**0.795**	0.793	0.794	0.792

Experiments

AUC result of KGCN with different receptive field H.

H	1	2	3	4
movie	0.972	**0.976**	0.974	0.514
book	**0.738**	0.731	0.684	0.547
music	**0.794**	0.723	0.545	0.534

Experiments

AUC result of KGCN with different dimension of embedding.

d	4	8	16	32	64	128
movie	0.968	0.970	0.975	**0.977**	0.973	0.972
book	0.709	0.732	0.733	0.735	**0.739**	0.736
music	0.789	0.793	**0.797**	0.793	0.790	0.789

GENI

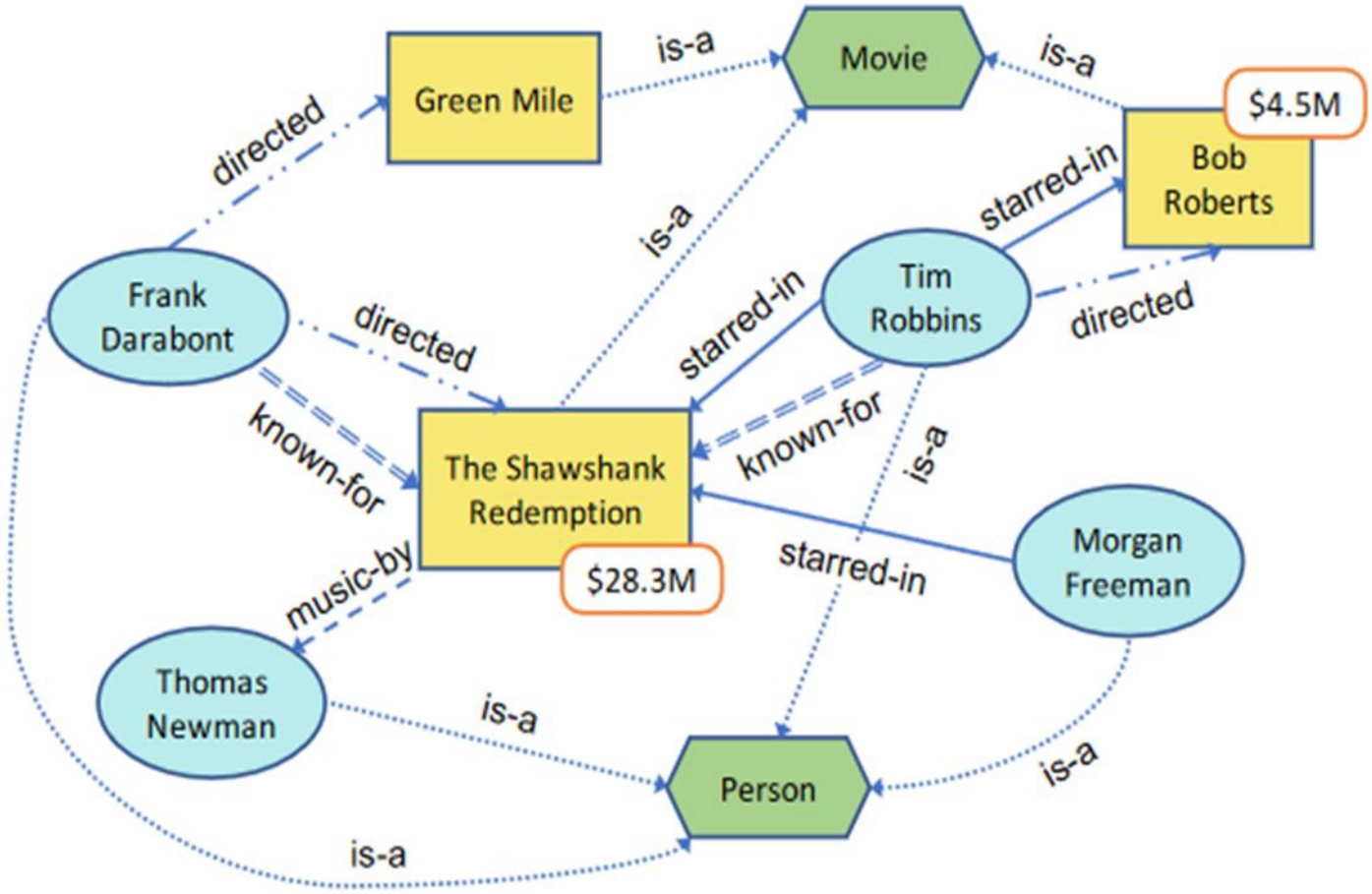
KDD19:Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks

Motivation

PR scores are based only on the graph structure, PPR does not take edge types into account. Existing GNN shave focused on graph representation learning via embedding aggregation, and have not been designed to tackle challenges that arise with supervised estimation of node importance in KGs

GENI performs an aggregation of importance scores instead of aggregating node embeddings via predicate-aware attention mechanism and flexible centrality adjustment to get node importance score.

Problem Definition



Given a KG and importance scores {s} for a subset $V_s \subseteq V$ of nodes, learn a function $S: V \rightarrow [0, \infty)$ that estimates the importance score of every node in KG.

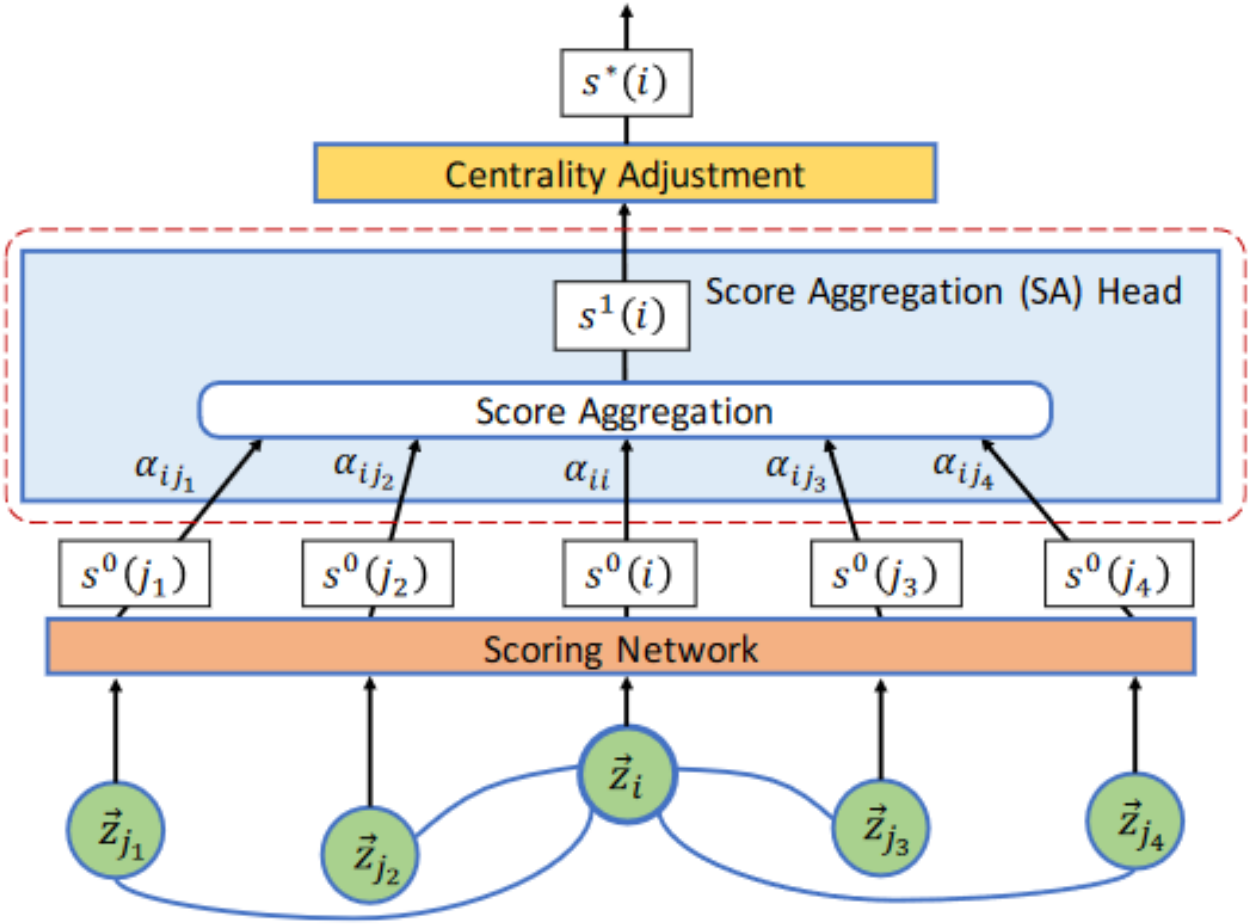
For example, the total gross of a movie can be used as an importance score for a movie KG.

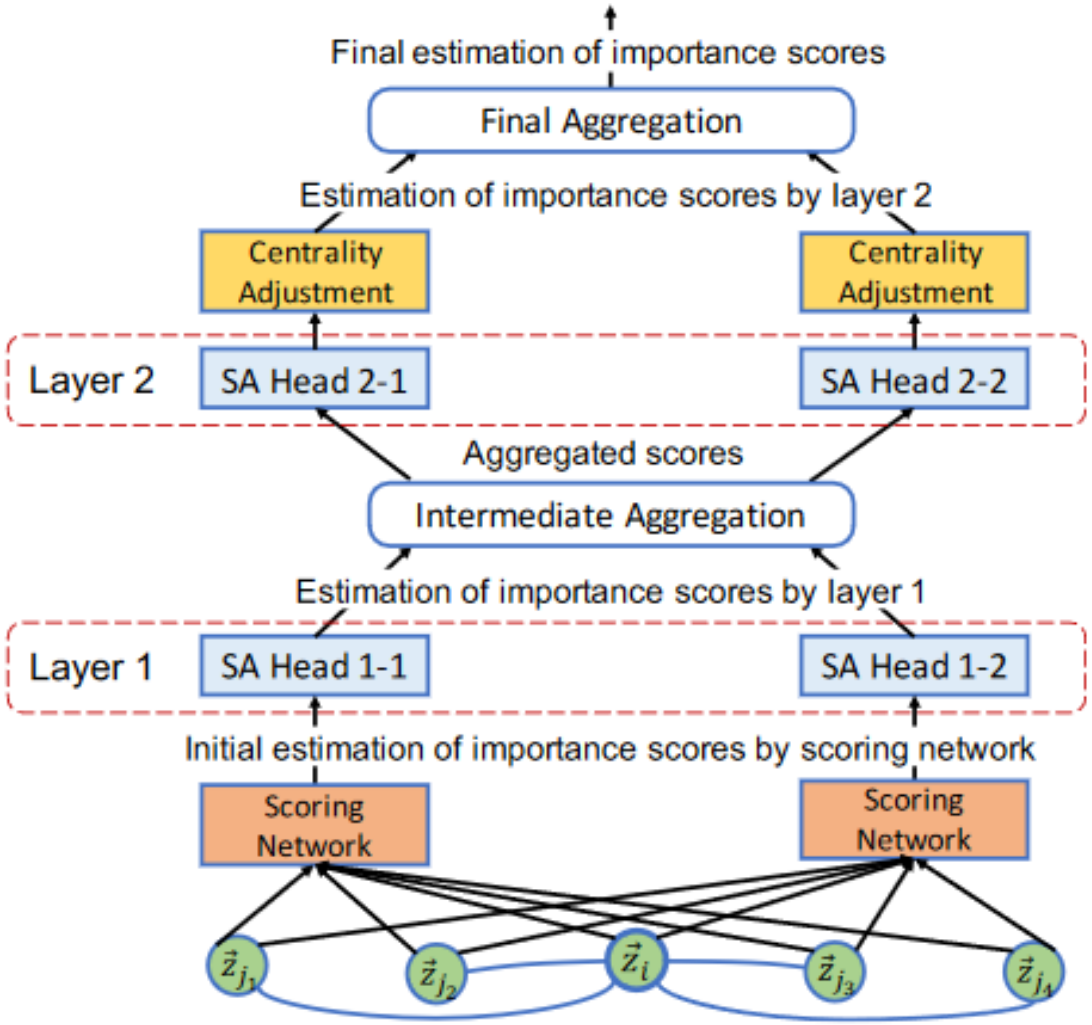
Motivation

PR scores are based only on the graph structure, PPR does not take edge types into account. Existing GNN shave focused on graph representation learning via embedding aggregation, and have not been designed to tackle challenges that arise with supervised estimation of node importance in KGs

GENI performs an aggregation of importance scores instead of aggregating node embeddings via predicate-aware attention mechanism and flexible centrality adjustment to get node importance score.

GENI model


$s^*(i)$
Centrality Adjustment
$s^1(i)$
Score Aggregation (SA) Head
Score Aggregation
α_{ij_1}
α_{ij_2}
α_{ii}
α_{ij_3}
α_{ij_4}
$s^0(j_1)$
$s^0(j_2)$
$s^0(i)$
$s^0(j_3)$
$s^0(j_4)$
Scoring Network
$\vec{z}_{j_1}$
$\vec{z}_{j_2}$
$\vec{z}_i$
$\vec{z}_{j_3}$
$\vec{z}_{j_4}$


Final estimation of importance scores
Final Aggregation
Estimation of importance scores by layer 2
Centrality Adjustment
Centrality Adjustment
Layer 2
SA Head 2-1
SA Head 2-2
Aggregated scores
Intermediate Aggregation
Estimation of importance scores by layer 1
Layer 1
SA Head 1-1
SA Head 1-2
Initial estimation of importance scores by scoring network
Scoring Network
Scoring Network
$\vec{z}_{j_1}$
$\vec{z}_{j_2}$
$\vec{z}_i$
$\vec{z}_{j_3}$
$\vec{z}_{j_4}$

GENI model

1. Score Aggregation

$$s^{\ell}(i) = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{\ell} \, s^{\ell-1}(j) \qquad (3)$$

2. Predicate-Aware Attention Mechanism

$$\alpha_{ij}^{\ell} = \frac{\exp\left(\sigma_a\left(\sum_m \vec{a}_{\ell}^{\top}[s^{\ell}(i) \| \phi(p_{ij}^{m}) \| s^{\ell}(j)]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\sigma_a\left(\sum_m \vec{a}_{\ell}^{\top}[s^{\ell}(i) \| \phi(p_{ik}^{m}) \| s^{\ell}(k)]\right)\right)} \qquad (5)$$

3. Centrality Adjustment

$$c^{*}(i) = \gamma \cdot c(i) + \beta \qquad s^{*}(i) = \sigma_s\left(c^{*}(i) \cdot s^{L}(i)\right)$$

Score Aggregation

1. Score Aggregation

$$s^{\ell}(i) = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{\ell} \, s^{\ell-1}(j) \qquad (3)$$

$$s^{0}(i) = \textsc{ScoringNetwork}(\vec{z}_i)$$

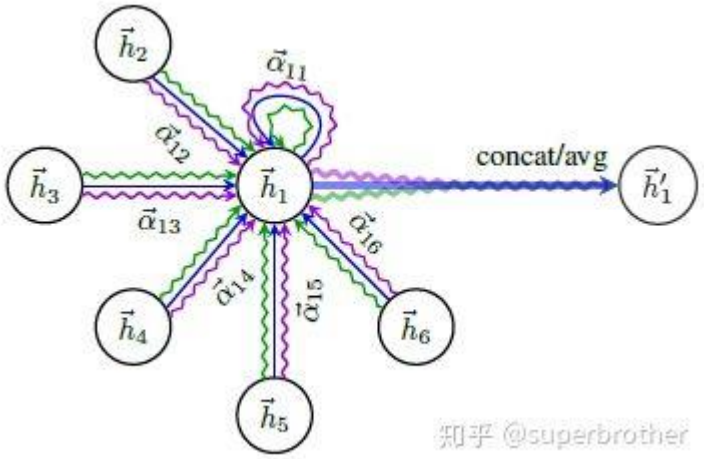
where ScoringNetwork can be any neural network that takes in a node feature vector and returns an estimation of its importance. We used a simple fully-connected neural network for our experiments.

Predicate-Aware Attention Mechanism

2. Predicate-Aware Attention Mechanism

$$\alpha_{ij}^{\ell} = \frac{\exp\left(\sigma_a\left(\sum_m \vec{a}_{\ell}^{\top}[s^{\ell}(i)||\phi(p_{ij}^{m})||s^{\ell}(j)]\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp\left(\sigma_a\left(\sum_m \vec{a}_{\ell}^{\top}[s^{\ell}(i)||\phi(p_{ik}^{m})||s^{\ell}(k)]\right)\right)} \quad (5)$$

Like GAT:



$$e_{ij} = a\left([Wh_i||Wh_j]\right), j \in \mathcal{N}_i \quad (1)$$

$$\alpha_{ij} = \frac{exp\left(LeakyReLU(e_{ij})\right)}{\sum_{k\in\mathcal{N}_i} exp\left(LeakyReLU(e_{ik})\right)} \quad (2)$$

Centrality Adjustment

3. Centrality Adjustment

$$c^*(i) = \gamma \cdot c(i) + \beta$$

$$s^*(i) = \sigma_s \left(c^*(i) \cdot s^L(i) \right)$$

Existing methods such as PR, PPR, and HAR make a common assumption that the importance of a node positively correlates with its centrality in the graph. In the context of KGs, it is also natural to assume that more central nodes would be more important than less central ones, unless the given importance scores present contradictory evidence. Making use of this prior knowledge becomes especially beneficial in cases where we are given a small number of importance scores compared to the total number of entities, and in cases where the importance scores are given for entities of a specific type out of the many types in KG.

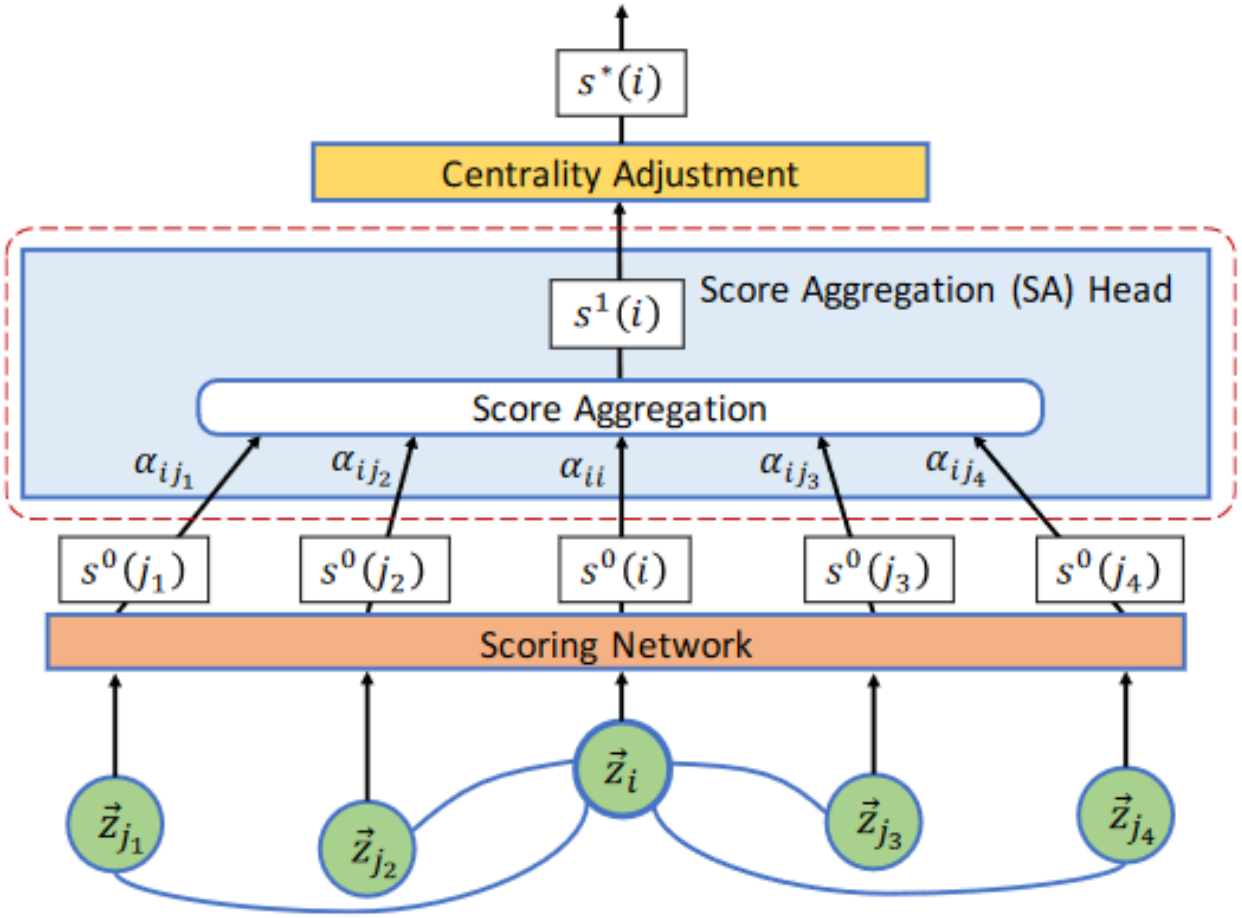
Model Training

Loss： MSE

$$\frac{1}{|V_s|}\sum_{i\in V_s}\left(s^*(i)-g(i)\right)^2$$

ScoringNetwork is trained jointly with the rest of GENI

Vs： set of nodes with known importance scores



Dataset and Evaluation

Table 3: Real-world KGs. See Section 4.1 and Appendix A for details. SCC: Strongly connected component. OOD: Out-of-domain.

Name	# Nodes	# Edges	# Predicates	# SCCs.	Input Score Type	# Nodes w/ Scores	Data for OOD Evaluation
FB15K	14,951	592,213	1,345	9	# Pageviews	14,108 (94%)	N/A
MUSIC10K	24,830	71,846	10	130	Song hotttnesss	4,214 (17%)	Artist hotttnesss
TMDB5K	123,906	532,058	22	15	Movie popularity	4,803 (4%)	Director ranking
IMDB	1,567,045	14,067,776	28	1	# Votes for movies	215,769 (14%)	Director ranking

$$DCG@k = \sum_{i=1}^{k} \frac{r_i}{\log_2(i+1)}$$

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

$$Spearman = \frac{\sum_i (g_{ri} - \bar{g}_r)(s_{ri} - \bar{s}_r)}{\sqrt{\sum_i (g_{ri} - \bar{g}_r)^2}\sqrt{\sum_i (s_{ri} - \bar{s}_r)^2}}$$

Results

Table 4: In-domain prediction results on real-world datasets. GENI consistently outperforms all baselines. Numbers after ± symbol are standard deviation from 5-fold cross validation. Best results are in bold, and second best results are underlined.

Method	FB15K		MUSIC10K		TMDB5K		IMDB	
	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN
PR	0.8354 ± 0.016	0.3515 ± 0.015	0.5510 ± 0.021	−0.0926 ± 0.034	0.8293 ± 0.026	0.5901 ± 0.011	0.7847 ± 0.048	0.0881 ± 0.004
PPR	0.8377 ± 0.015	0.3667 ± 0.015	0.7768 ± 0.009	0.3524 ± 0.046	0.8584 ± 0.013	0.7385 ± 0.010	0.7847 ± 0.048	0.0881 ± 0.004
HAR	0.8261 ± 0.005	0.2020 ± 0.012	0.5727 ± 0.017	0.0324 ± 0.044	0.8141 ± 0.021	0.4976 ± 0.014	0.7952 ± 0.036	0.1318 ± 0.005
LR	0.8750 ± 0.005	0.4626 ± 0.019	0.7301 ± 0.023	0.3069 ± 0.032	0.8743 ± 0.015	0.6881 ± 0.013	0.7365 ± 0.009	0.5013 ± 0.002
RF	0.8734 ± 0.005	0.5122 ± 0.019	0.8129 ± 0.012	0.4577 ± 0.012	0.8503 ± 0.016	0.5959 ± 0.022	0.7651 ± 0.010	0.4753 ± 0.005
NN	0.9003 ± 0.005	0.6031 ± 0.012	0.8015 ± 0.017	0.4491 ± 0.027	0.8715 ± 0.006	0.7009 ± 0.009	0.8850 ± 0.016	0.5120 ± 0.008
GAT	0.9205 ± 0.009	0.7054 ± 0.013	0.7666 ± 0.016	0.4276 ± 0.023	0.8865 ± 0.011	0.7180 ± 0.010	0.9110 ± 0.011	0.7060 ± 0.007
GENI	**0.9385 ± 0.004**	**0.7772 ± 0.006**	**0.8224 ± 0.018**	**0.4783 ± 0.009**	**0.9051 ± 0.005**	**0.7796 ± 0.009**	**0.9318 ± 0.005**	**0.7387 ± 0.002**

Table 5: Out-of-domain prediction results on real-world datasets. GENI consistently outperforms all baselines. Numbers after ± symbol are standard deviation from 5-fold cross validation. Best results are in bold, and second best results are underlined.

Method	MUSIC10K		TMDB5K		IMDB	
	NDCG@100	NDCG@2000	NDCG@100	NDCG@2000	NDCG@100	NDCG@2000
PR	0.6520 ± 0.000	0.8779 ± 0.000	0.8337 ± 0.000	0.8079 ± 0.000	0.0000 ± 0.000	0.1599 ± 0.000
PPR	0.7324 ± 0.006	0.9118 ± 0.002	0.8060 ± 0.041	0.7819 ± 0.022	0.0000 ± 0.000	0.1599 ± 0.000
HAR	0.7113 ± 0.004	0.8982 ± 0.001	0.8913 ± 0.010	0.8563 ± 0.007	0.2551 ± 0.019	0.3272 ± 0.005
LR	0.6644 ± 0.006	0.8667 ± 0.001	0.4990 ± 0.013	0.5984 ± 0.002	0.3064 ± 0.007	0.2755 ± 0.003
RF	0.6898 ± 0.022	0.8796 ± 0.003	0.5993 ± 0.040	0.6236 ± 0.005	0.4066 ± 0.145	0.3719 ± 0.040
NN	0.6981 ± 0.017	0.8836 ± 0.005	0.5675 ± 0.023	0.6172 ± 0.009	0.2158 ± 0.035	0.3105 ± 0.019
GAT	0.6909 ± 0.009	0.8834 ± 0.003	0.5349 ± 0.016	0.5999 ± 0.007	0.3858 ± 0.065	0.4209 ± 0.016
GENI	**0.7964 ± 0.007**	**0.9121 ± 0.002**	**0.9078 ± 0.004**	**0.8776 ± 0.002**	**0.4519 ± 0.051**	**0.4962 ± 0.025**

Parameter Sensitivity