metapath2vec: Scalable Representation Learning for Heterogeneous Networks

Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami.
ACM SIGKDD

DSAIL 2023 Winter Internship
JunYoung Kim

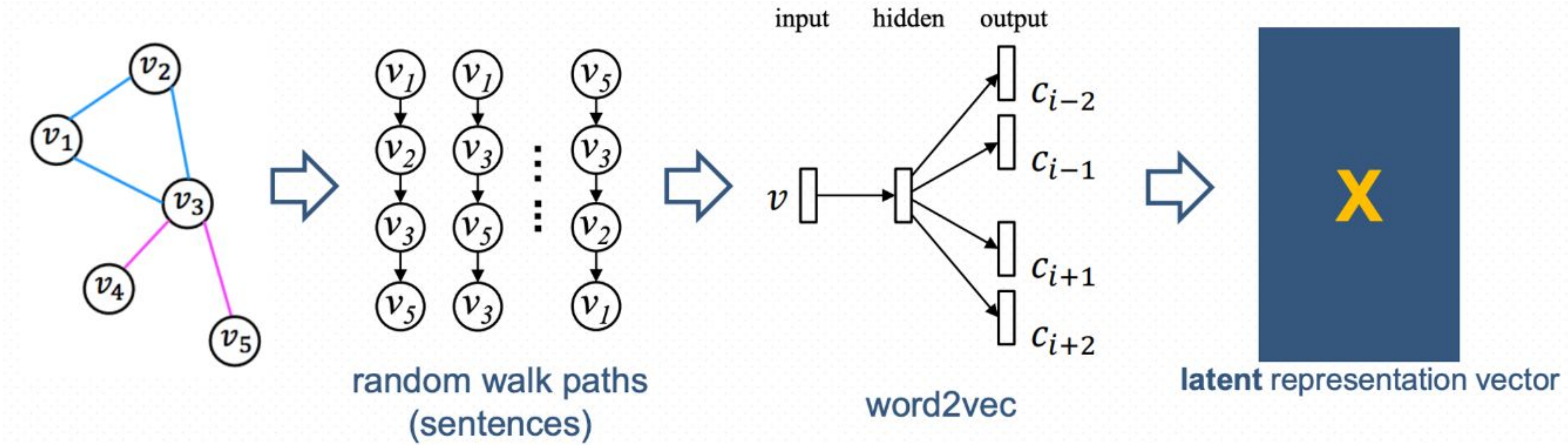
Index

1. Introduction

- Neural network-based learning models can represent latent embeddings that capture the internal relations of rich, complex data across various modalities.
- Social and information networks are similarly rich and complex data that encode the dynamics and types of human interactions, and are similarly amenable to representation learning using neural networks.
- Recent research publications have proposed word2vec-based network representation learning frameworks
    - DeepWalk, LINE, node2vec

      → These work has far focused on representation learning for homogeneous networks-representative of singular type of nodes and relationships.

1. Introduction



1. Introduction

- However, these work has thus far focused on representation learning for homogeneous networks—representative of singular type of nodes and relationships.

1. Introduction

Heterogeneous Network Embedding: Challenges

- How do we effectively preserve the concept of “node-context” among multiple types of nodes?
- 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.

1. Introduction

Contributions

(1) Formalizes the problem of heterogeneous network representation learning and identi□es its unique challenges resulting from network heterogeneity.

(2) Develops effective and efficient network embedding frameworks for preserving both structural and semantic correlations of heterogeneous networks.

- metapath2vec & metapath2vec++

(3) Through extensive experiments, demonstrates the efficacy and scalability of the presented methods in various heterogeneous network mining tasks

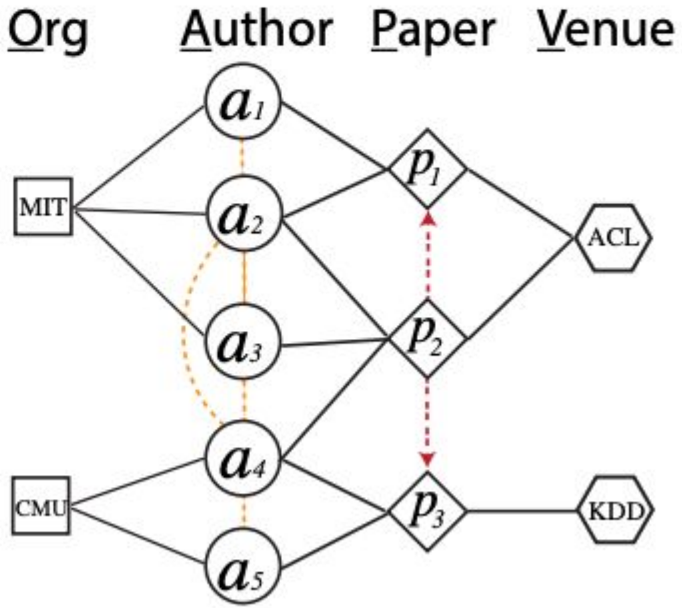
(4) Demonstrates the automatic discovery of internal semantic relationships between different types of nodes in heterogeneous networks by metapath2vec & metapath2vec++, not discoverable by existing work.

2. Problem definition

heterogeneous network

- defined as a graph G = (V, E, T)
- $\phi(v) : V \rightarrow T_v$

  $\varphi(e) : E \rightarrow T_E$ $|T_V| + |T_E| > 2$



2. Problem definition

Problem 1. Heterogeneous Network Representation Learning:

- The task is to learn the d-dimensional latent representations $X \in \mathbb{R}^{|V| \times d}, d \ll |V|$ that are able to capture the structural and semantic relations among them.
- The output : the low-dimensional matrix X, with the v th row-a d-dimensional vector Xv - corresponding to the representation of node v
- There are different types of nodes in V → their representations are mapped into the same latent space
- The learned node representations can benefit various embedding vector of each node can be used as the feature input of node classification, clustering, and similarity search tasks

2. Problem definition

Problem 1. Heterogeneous Network Representation Learning:

- The premise of network embedding models is to preserve the proximity between a node and its neighborhood.

  In a heterogeneous environment…

  ❖ how do we define and model this ‘node–neighborhood’ concept?
  ❖ how do we optimize the embedding models that effectively maintain the structures and semantics of multiple types of nodes and relations?

3. The Metapath2Vec Framework

- The objective of metapath2vec is to maximize the network probability in consideration of multiple types of nodes and edges.

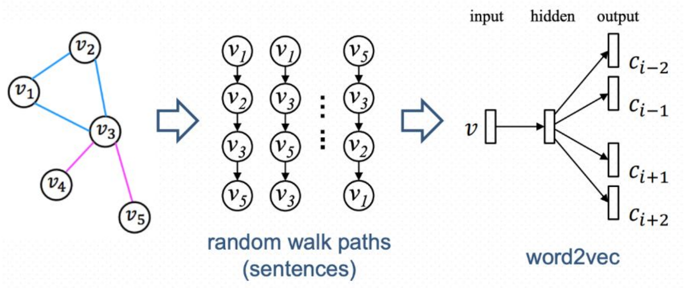
- To model the heterogeneous neighborhood of a node,

  → metapath2vec introduces the *heterogeneous skip-gram model.*

- To incorporate the heterogeneous network structures into skip-gram…

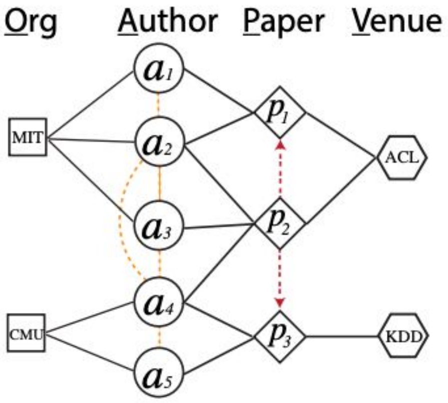
  → *meta-path-based random walks* in heterogeneous networks.

3. The Metapath2Vec Framework



Meta-Path-Based Random Walk

- 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.

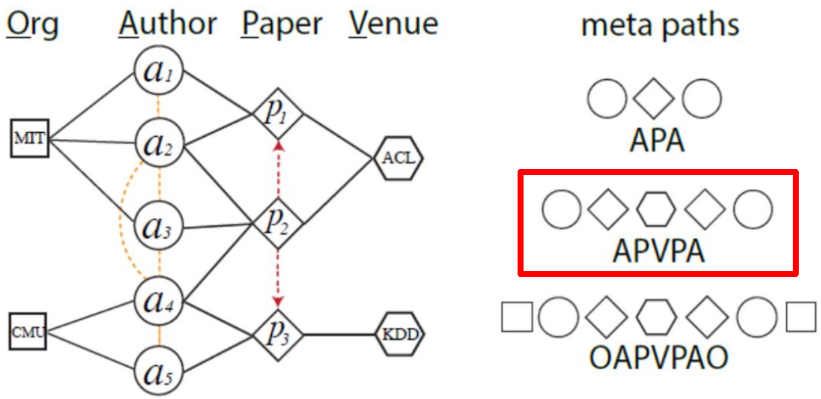


3. The Metapath2Vec Framework

Meta-Path-Based Random Walk

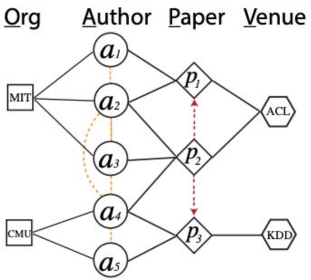
- meta-path scheme P

$$V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \dots V_t \xrightarrow{R_t} V_{t+1} \dots \xrightarrow{R_{l-1}} V_l$$



3. The Metapath2Vec Framework

Meta-Path-Based Random Walk



- 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

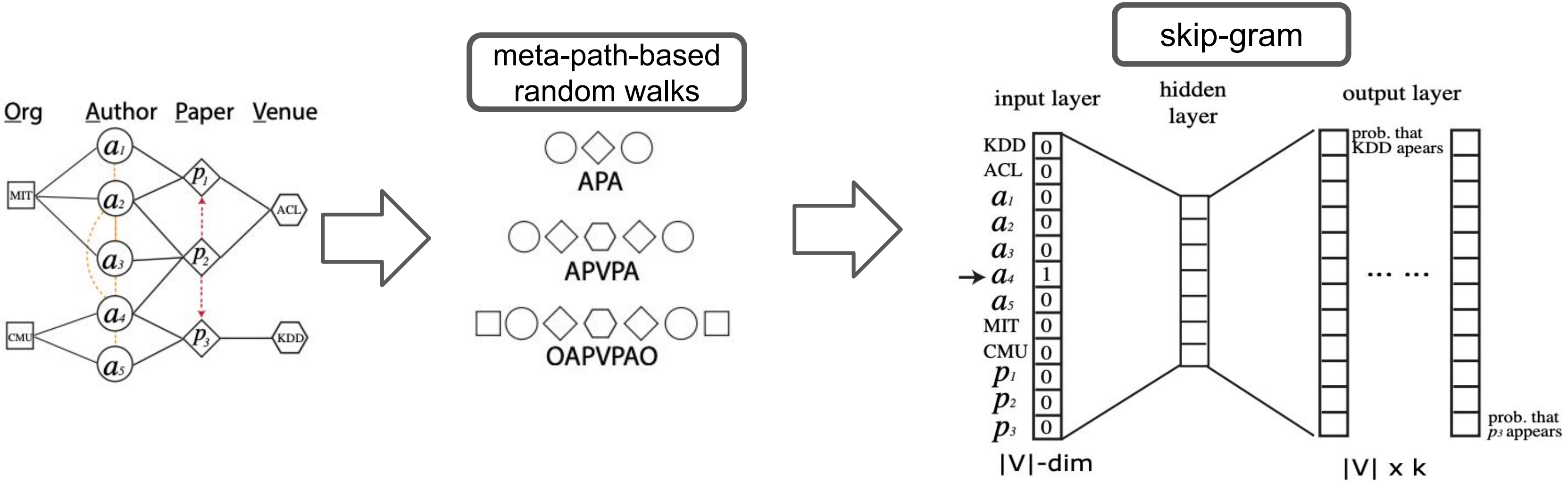
$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

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

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

3. The Metapath2Vec Framework

metapath2vec : Skip-Gram



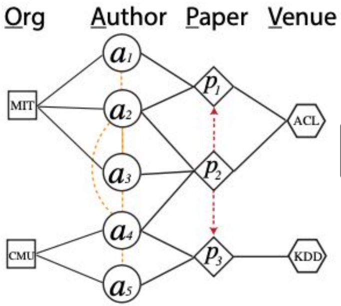
$$\underset{\theta}{argmax} \Sigma_{v \in V} \Sigma_{t \in T_v} \Sigma_{c_t \in N_t(v)} log p(c_t | v; \theta)$$

$$\log \sigma(X_{c_t} \cdot X_v) + \Sigma_{m=1}^{M} \mathbb{E}_{u^m \sim P(u)} [\log \sigma(-X_{u^m} \cdot X_v)]$$

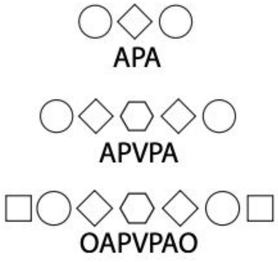
- p : softmax function
- N_t(v)
- X_v
- P(u)
- σ(x) : logistic function

3. The Metapath2Vec Framework

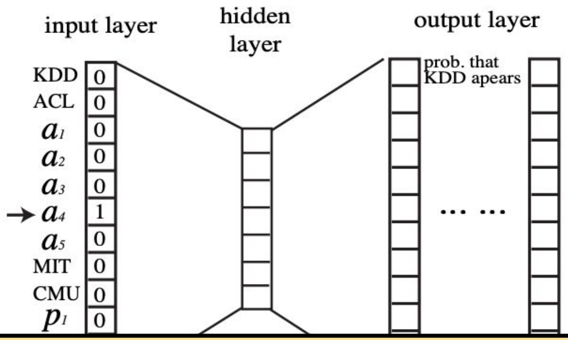
metapath2vec : Skip-Gram



meta-path-based random walks



skip-gram



$$\underset{\theta}{argmax} \Sigma_{v\in V}\Sigma_{t\in T_v}\Sigma_{c_t\in N_t(v)} logp(c_t|v;\theta)$$

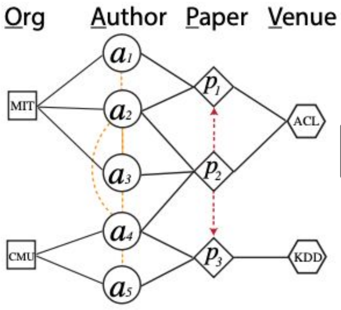
$$\log\sigma(X_{c_t}\cdot X_v)+\Sigma_{m=1}^{M}\mathbb{E}_{u^m\sim P(u)}[\log\sigma(-X_{u^m}\cdot X_v)]$$

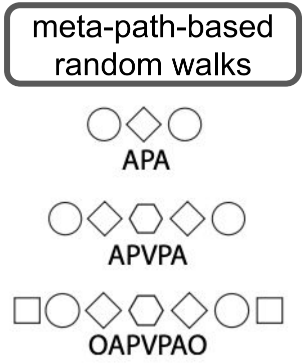
The potential issue of skip-gram for heterogeneous network embedding:

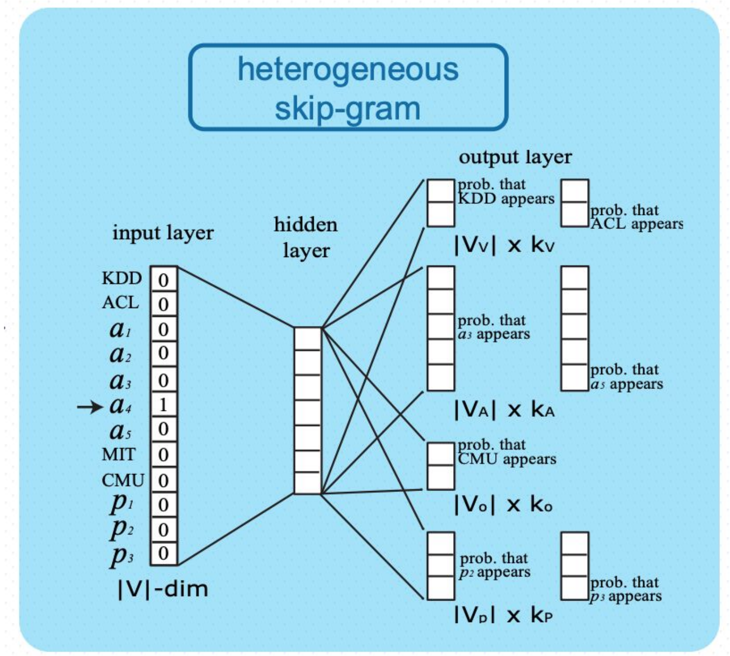
To predict the context code c_t given a node v, metapath2vec encourages all types of node to appear in this context position

3. The Metapath2Vec Framework

metapath2vec++ : heterogeneous Skip-Gram

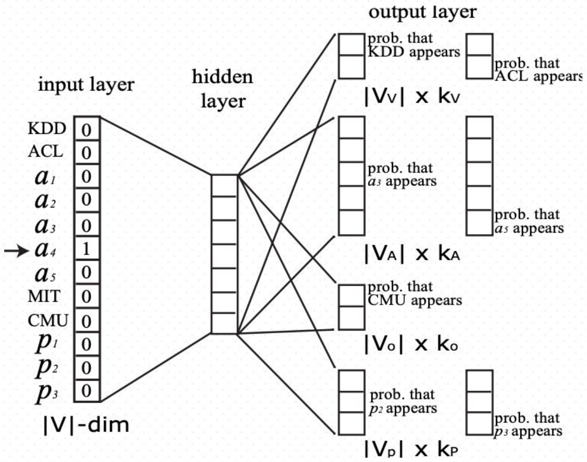






3. The Metapath2Vec Framework

metapath2vec++ : heterogeneous Skip-Gram



- softmax in *metapath2vec*

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

- softmax in *metapath2vec++*

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

- 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

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

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

3. The Metapath2Vec Framework

metapath2vec++

```
Input: The heterogeneous information network G = (V, E, T),
       a meta-path scheme 𝒫, #walks per node w, walk
       length l, embedding dimension d, neighborhood size k
Output: The latent node embeddings X ∈ ℝ^{|V|×d}

initialize X ;
for i = 1 → w do
    for v ∈ V do
        MP = MetaPathRandomWalk(G, 𝒫, v, l) ;
        X = HeterogeneousSkipGram(X, k, MP) ;
    end
end
return X ;

MetaPathRandomWalk(G, 𝒫, v, l)
MP[1] = v ;
for i = 1 → l−1 do
    draw u according to Eq. 3 ;
    MP[i+1] = u ;
end
return MP ;

HeterogeneousSkipGram(X, k, MP)
for i = 1 → l do
    v = MP[i] ;
    for j = max(0, i-k) → min(i+k, l) & j ≠ i do
        c_t = MP[j] ;
        X^new = X^old − η · ∂O(X)/∂X (Eq. 7) ;
    end
end
```

4. Experiments

Data

1. **AMiner** Computer Science (CS) dataset
   - three types of nodes: authors, papers, and venues (9,323,739 computer scientists, 3,194,405 papers, 3,883 computer science venues)
2. Database and Information Systems (**DBIS**) dataset
   - three types of nodes: authors, publications, and venues (464 venues, their top-5000 authors, and corresponding 72,902 publications)

Experimental Setup

(1) The number of walks per node w: 1000;

(2) The walk length l: 100;

(3) The vector dimension d: 128 (LINE: 128 for each order);

(4) The neighborhood size k: 7;

(5) The size of negative samples: 5

4. Experiments

Baselines

- DeepWalk
- node2vec
- LINE
- PTE

Mining Tasks

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

4. Experiments

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

4. Experiments

Application 1: Multi-Class Node Classification

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

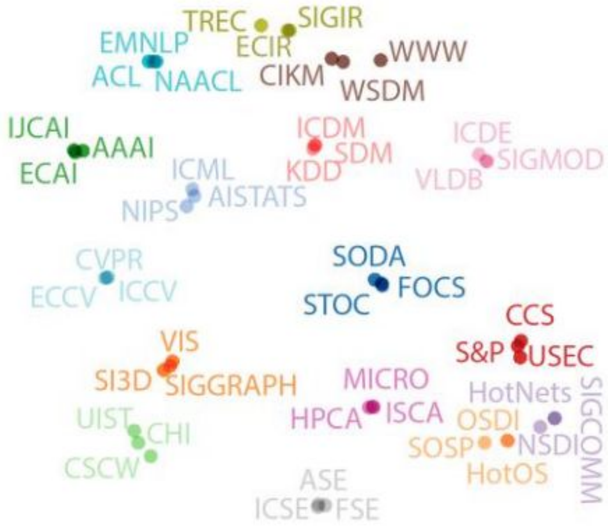
Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.7153	0.7222	0.7256	0.7270	0.7273	0.7274	0.7273	0.7271	0.7275	0.7275
	LINE (1st+2nd)	0.8849	0.8886	0.8911	0.8921	0.8926	0.8929	0.8934	0.8936	0.8938	0.8934
	PTE	0.8898	0.8940	0.897	0.8982	0.8987	0.8990	0.8997	0.8999	0.9002	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

4. Experiments

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



4. Experiments

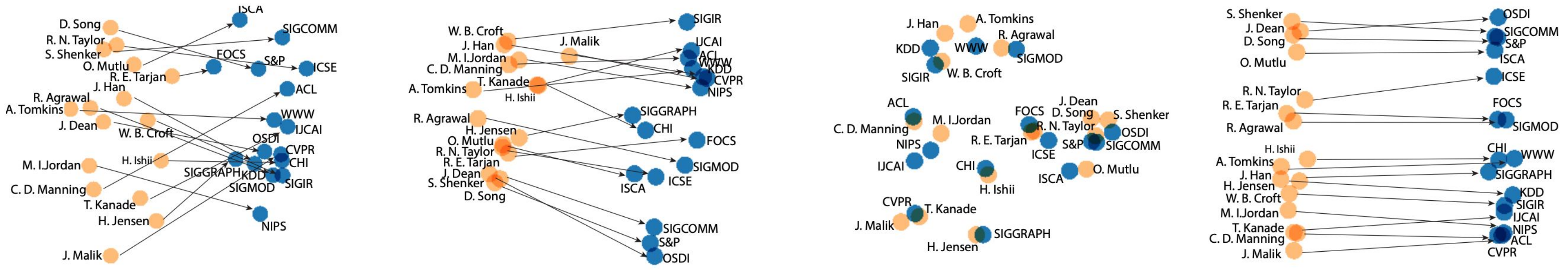
Application 3: Similarity Search

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

4. Experiments

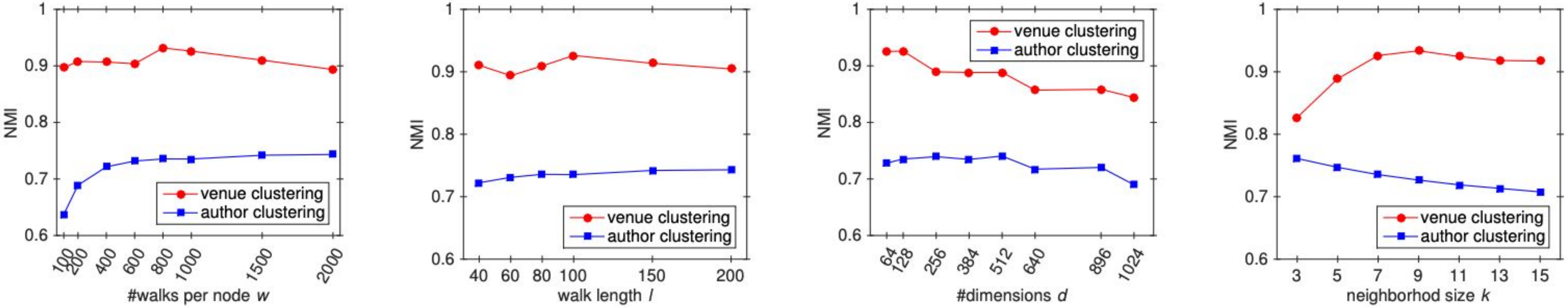
Visualization



(a) DeepWalk / node2vec (b) PTE (c) *metapath2vec* (d) *metapath2vec++*

4. Experiments

Parameter sensitivity

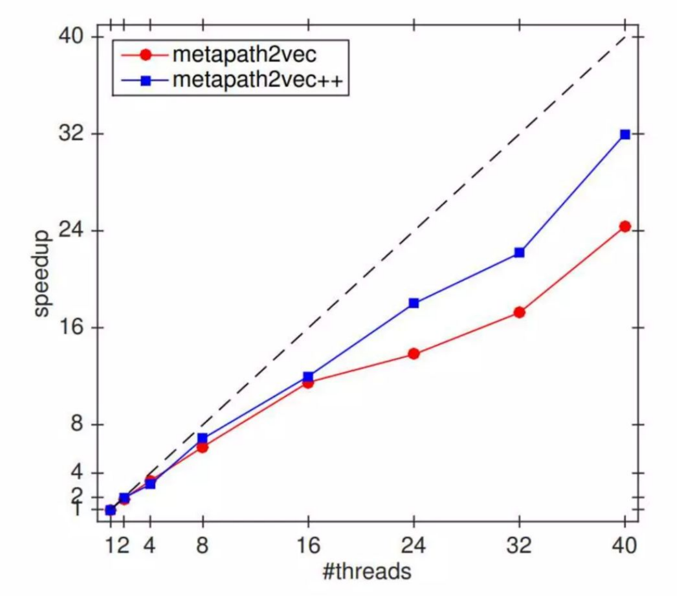


(a) #walks per node w (b) walk length l (c) #dimensions d (d) neighborhood size k

Figure 4: Parameter sensitivity in clustering.

4. Experiments

Scalability of metapath2vec & metapath2vec++



5. Implementation

dataset

- MovieLens100K

```
HeteroData(
  movie={ x=[1682, 18] },
  user={ x=[943, 24] },
  (user, rates, movie)={
    edge_index=[2, 80000],
    rating=[80000],
    time=[80000],
    edge_label_index=[2, 20000],
    edge_label=[20000],
  },
  (movie, rated_by, user)={
    edge_index=[2, 80000],
    rating=[80000],
    time=[80000],
  }
)
```

5. Implementation

```python
class metapath2vec(nn.Module):
    def __init__(self, N_user, N_movie, prob):
        super(metapath2vec, self).__init__()

        self.N_user = N_user
        self.N_movie = N_movie
        self.N_total = N_user + N_movie
        self.prob = prob

        self.l = WALK_LEN
        self.d = D
        self.k = NEIGHBORHOOD
        self.lr = LR

        self.embedding = nn.Embedding(num_embeddings=self.N_total, embedding_dim=self.d)
```

```python
    def walk(self, starting_points, URM, MRU):
        def _random_select(tf):
            ret = torch.zeros(tf.size(0), 1, dtype=int)
            for i in range(tf.size(0)):
                idx = tf[i].nonzero().squeeze(1)
                x = random.randint(0, idx.size(0))
                ret[i] = x
            return ret

        path = starting_points.clone().detach()
        for _ in range(0, self.l, 2):
            #Only considered UMU
            m = URM[1, _random_select(URM[0, :] == path[:, -1].unsqueeze(1))]
            u = MRU[1, _random_select(MRU[0, :] == m)]
            path = torch.cat([path, u], dim=-1)
        return path
```

BATCH_SIZE = 32

WALK_LEN = 100 # walk length l

D = 10 # embedding dimension d

NEIGHBORHOOD = 3 # neighborhood size k

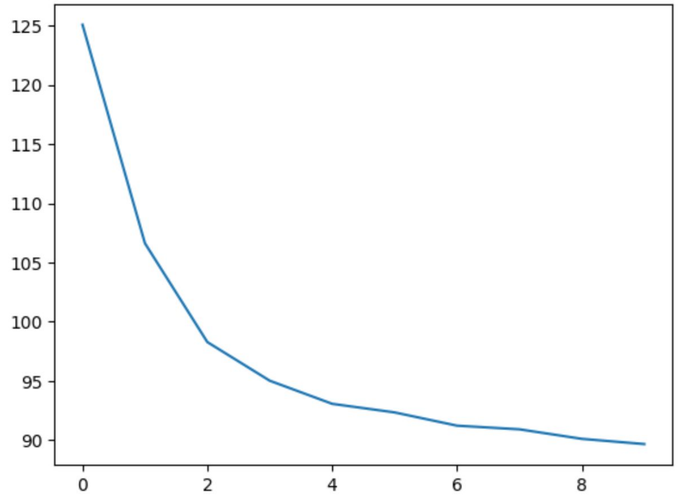
LR = 1e-3 # learning rate

N_WALK = 10 # walk per node w

5. Implementation

```
def skipgram(self, path):
    def _negative_sample(prob, bound):
        samples = torch.tensor([], dtype=int)

        for i in range(bound.size(0)):
            idx = prob.multinomial(num_samples=self.k).unsqueeze(0)

            while torch.isin(idx, bound[i]).sum().item() != 0:
                idx = prob.multinomial(num_samples=self.k).unsqueeze(0)
            samples = torch.cat([samples, idx], dim=0)

        return samples

    optimizer = optim.SGD([self.embedding.weight], lr=self.lr)
    O_x = 0
    n = 0
    for i in range(path.size(1)):
        lbd = max(0, i-self.k)
        rbd = min(path.size(1), i+self.k)
        for j in range(lbd, rbd):
            optimizer.zero_grad()

            #positive
            pos = torch.log(
                    F.sigmoid(
                        (
                        self.embedding.weight[path[:,i]] *
                        self.embedding.weight[path[:,j]]
                        ).sum(dim=1).unsqueeze(1)
                    )
                )

            #negative
            neg_samples = _negative_sample(self.prob, path[:, lbd:rbd])
            neg = torch.sum(
                    torch.log(
                        F.sigmoid(
                        ((
                        -self.embedding.weight[neg_samples].transpose(0,1)) *
                        self.embedding.weight[path[:,i]]
                        ).sum(dim=2).unsqueeze(2)
                        )
                    )
                    ,dim=0)
            O_x += -torch.sum(pos + neg)
            n += 1
    loss = O_x / n
    loss.backward()
    optimizer.step()
    return loss
```

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



6. Conclusion

Conclusion

- To address the network heterogeneity challenge, we propose the metapath2vec and metapath2vec++ methods
- To leverage this method, we formalize the heterogeneous neighborhood function of a node, enabling the skip-gram-based maximization of the network probability in the context of multiple types of nodes
- Finally, we achieve effective and efficient optimization by presenting a heterogeneous negative sampling technique

Future work

1) Face the challenge of large intermediate output data when sampling a network into a huge pile of paths, and thus identifying and optimizing the sampling space is an important direction

2) As is also the case with all metapath-based heterogeneous network mining methods, metapath2vec and metapath2vec++ can be further improved by the automatic learning of meaningful meta-paths

3) Extending the models to incorporate the dynamics of evolving heterogeneous networks

4) Generalizing the models for different genres of heterogeneous networks.

감사합니다:)