metapath2vec

본 논문은 heterogeneous network의 representation을 나타내기 위해 metapath2vec 이라는 방법을 제안한다. 이 방법은 multiple types of nodes and links가 존재하는 graph에 적용될 수 있어 node classification, clustering, similarity search와 같은 다양한 task에 적용될 수 있음을 보여준다.

Introduction

network representation을 하기 위해 word2vec 방식을 차용한 Deepwalk, LINE, node2vec과 같은 여러 모델들이 제안되었지만 homogeneous networks에만 사용될 수 있기 때문에 한계가 명확했다.

이러한 문제를 해결하기 위해 heterogeneous한 network를 표현할 수 있는 metapath2vec, metapath2vec++를 제안한다.

기존 random walks로 표현할 수 없기 때문에 meta-path based random walks를 제안함과 동시에 효율적으로 근처 노드들을 표현할 수 있게 negative sampling-based method를 제안한다.

Problem Definition

$G = (V, E, T)$

$V : a\ set\ of\ node$, $E : a\ set\ of\ edge$, $T : a\ set\ of\ type$

$\phi(v) : V \rightarrow T_V$, $\varphi(e) : E \rightarrow T_E$

$T_V : the\ sets\ of\ object\ types$, $T_E : the\ sets\ of\ relation\ types$

$|T_V| + |T_E| > 2$: multiple types of nodes and links

Heterogeneous Network Representation Learning

heterogeneous network G가 주어졌을 때, task는 learn the d-dimensional latent representations $X \in \mathbb{R}^{|V| \times D}$ 이다.

Network Representation은 주변 Node들의 관계에서 Embedding을 찾는 것인데 heterogeneous한 node들이 있는 상황에서 어떻게 주변 Node들을 표현할지가 문제인 상황이다.

Metapath2vec

Homogeneous Network Embedding

$$\arg\max_{\theta} \prod_{v \in V} \prod_{c \in N(v)} p(c|v;\theta)$$

- Skip-Gram: 어떠한 $node\ v$에 대해 주변 Context (local structure)를 predict 하는 문제

Heterogeneous Network Embedding

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log p(c_t|v;\theta)$$

- $N_t(v)$: v's neighborhood with the t^{th} type of nodes
- $p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u \cdot X_v}}$ (softmax function)
    - Homogeneous한 방법과 달리 Heterogeneous한 Information을 반영하기 위해 Type이 같은 Neighbor만 예측에 반영
- Efficient optimization을 위해 **negative sampling** 이용

Negative Sampling

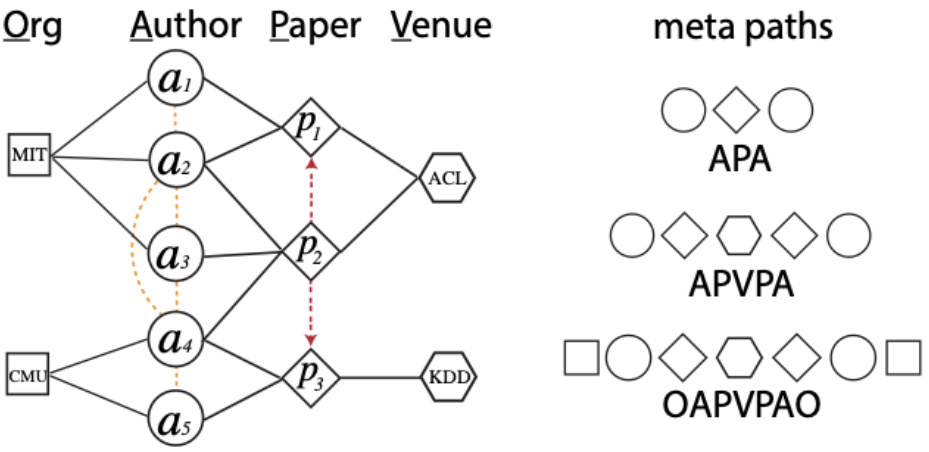
- sampled from the network for the construction of softmax
- 모든 Node들에 대해 계산하는 것이 비효율적이기 때문에 뽑히지 않았으면 하는 Negative한 Sample을 반영해서 Probability를 근사하는 것
- $O(X) = log\ \sigma(X_{c_t} \cdot X_v) + \sum_{m=1}^{M} \mathbb{E}_{u_t^m - P_t(u_t)}[log\ \sigma(-X_{u_t^m} \cdot X_v)]$
- M의 값이 커지면 Robust한 결과
- k: [5, 20]의 값을 선택하는 것이 일반적

- Negative Sample을 뽑는 함수는 Uniform한 것이 아니라 특정 Bias를 가지는 형태
- 아래와 같이 "빈도"를 기준으로 샘플링

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{n}^{j=0}\left(f(w_j)^{3/4}\right)}.$$

Meta-Path-Based Random Walks

Random Walk에서 보였던 것처럼 Neighborhood를 Random하게 방문하는 것이 아니라 미리 정해진 Meta-path scheme 대로 방문하자는 개념

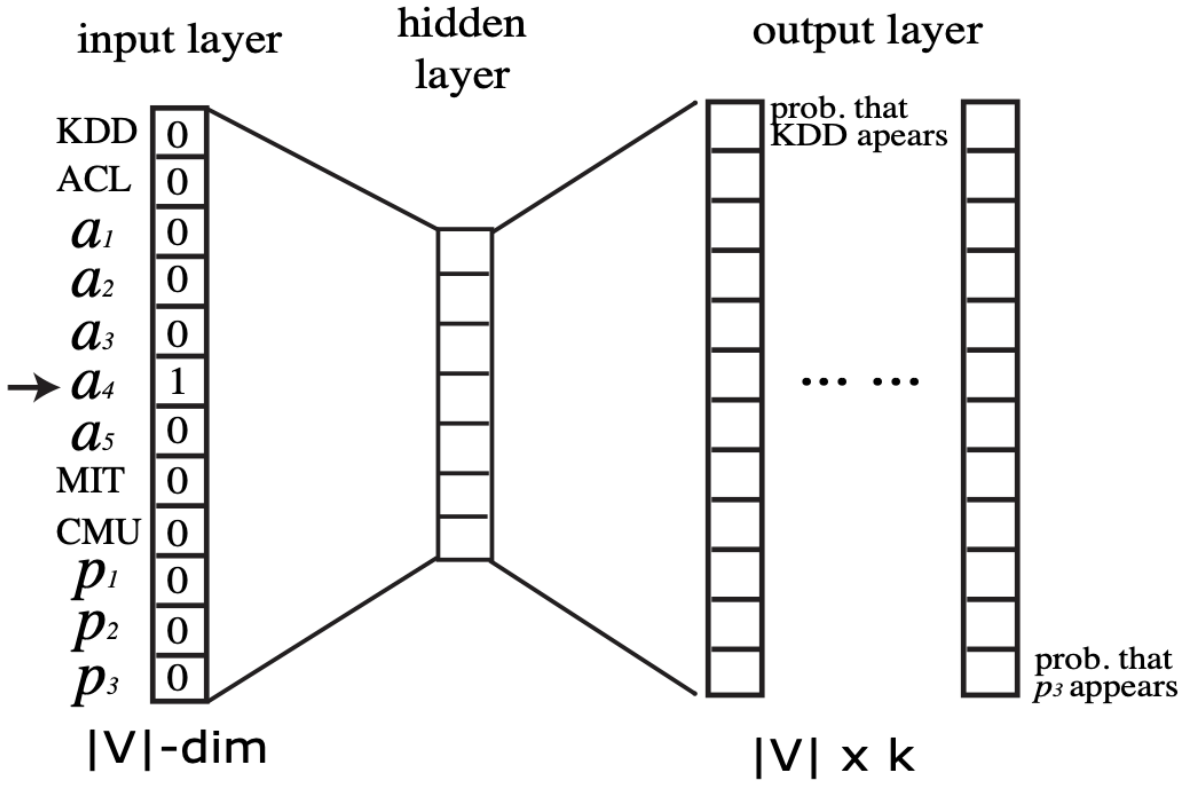


(a) An academic network

- 오른쪽의 meta paths 같이 scheme을 이미 정해놨으면 APA, APVPA, OAPVPAO 대로 type을 방문함
- 즉, i-th step의 transition probability는 아래와 같이 정의된다.

$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases} \tag{3}$$

- $\phi(v) : V \rightarrow T_V$
- 즉 정해진 Type이 아닐 경우는 이동할 확률이 0이고, Neighborhood 중에 정해진 Type이 있을 경우 Uniform 한 확률을 가진다.
- 이때, 이 pre-defined meta path는 symmetric한 형태를 갖는 게 일반적이다.



(b) Skip-gram in *metapath2vec*, node2vec, & DeepWalk

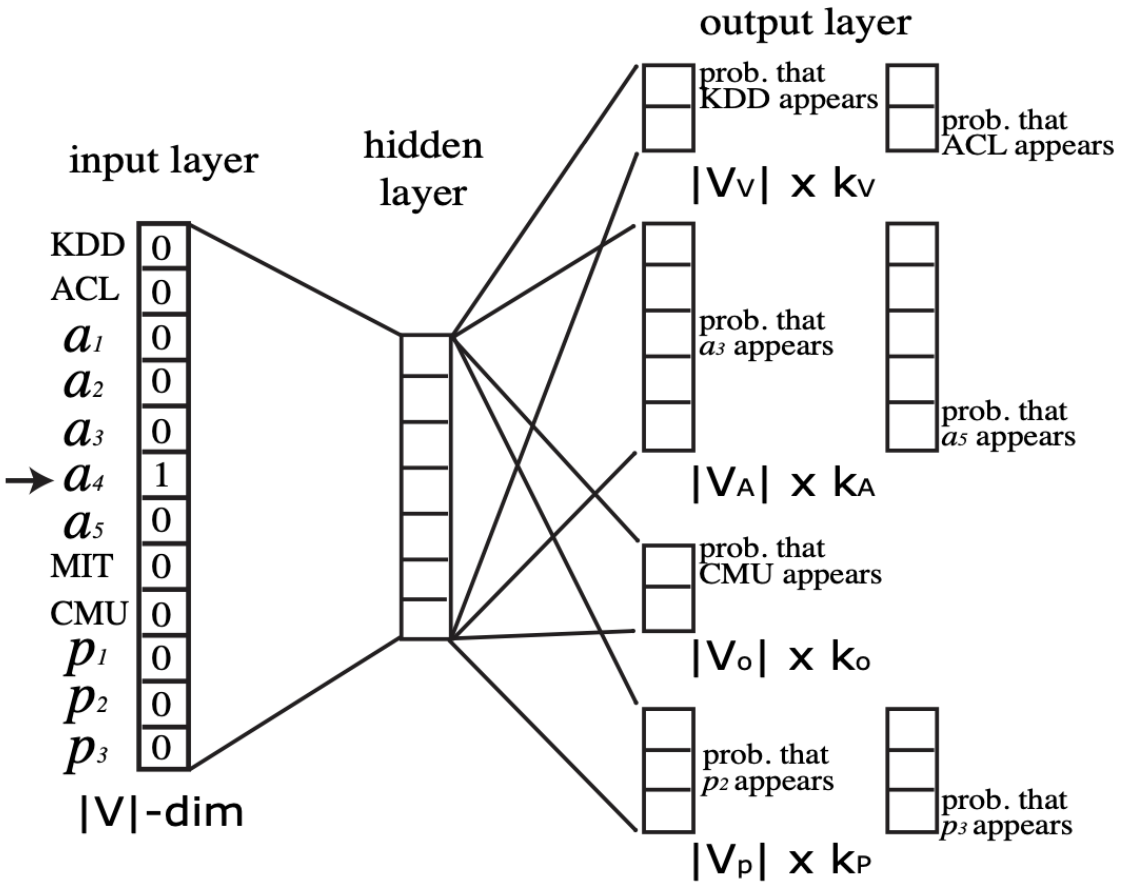
Metapath2vec++

- metapath2vec는 softmax를 계산할 때 node type을 반영하지 않는다.

- 즉, metapath2vec은 negative sampling시 모든 type의 negative samples을 반영한다.

이러한 문제를 해결하기 위해 heterogeneous negative sampling을 제안한다.

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



(c) Skip-gram in *metapath2vec++*

- 위의 그림대로 Type에 따라 각 Node의 probability를 구할 수 있게 된다.

Algorithm of metapath2vec++

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] ;
    $X^{new} = X^{old} - \eta \cdot \frac{\partial O(\mathbf{X})}{\partial X}$ (Eq. 7) ;
  end
end

ALGORITHM 1: The *metapath2vec++* Algorithm.

Gradient

$$O(X) = log\ \sigma(X_{c_t} \cdot X_v) + \sum_{m=1}^{M} \mathbb{E}_{u_t^m - P_t(u_t)}[log\ \sigma(-X_{u_t^m} \cdot X_v)]$$

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

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

- $\mathbb{I}_{ct}[u_t^m]$: an indicator function to indicate whether u_t^m is the neighborhood context node c_t and when $m = 0, u_t^0 = c_t$

Experiments

Data

- AMiner CS: 9,323,739 computer scientists and 3,194,405 papers, 3,883 computer science venues
- DBIS: 464 venues, top-500 authors, 72,902 publications
- venue: 장소 ??? type의 일종으로 생각

Node Classification

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

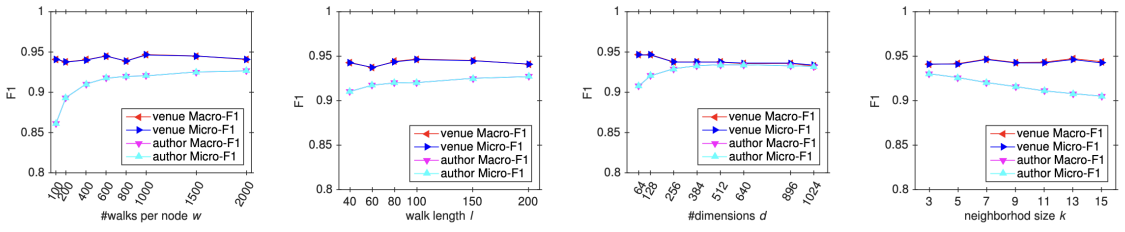
Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.0723	0.1396	0.1905	0.2795	0.3427	0.3911	0.4424	0.4774	0.4955	0.4457
	LINE (1st+2nd)	0.2245	0.4629	0.7011	0.8473	0.8953	0.9203	0.9308	0.9466	0.9410	0.9466
	PTE	0.1702	0.3388	0.6535	0.8304	0.8936	0.9210	0.9352	0.9505	0.9525	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

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

- consistently and significantly outperform
- small size of training data에서 strong

Parameter Sensitivity



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

Figure 3: Parameter sensitivity in multi-class node classification. 50% as training data and the remaining as test data.

- Base Parameter
  - the number of walks per node w: 1000
  - the walk length l: 100

    - the vector dimension d: 128
    - the neighborhood size k: 7
    - the size of negative samples: 5
- Base Paremeter를 기준으로 하나의 Parameter를 변화해가면서 실험
- #walks per node w and walk length l이 positive
- metapath2vec++가 insensitive해서 cost-efficient

Node Clustering

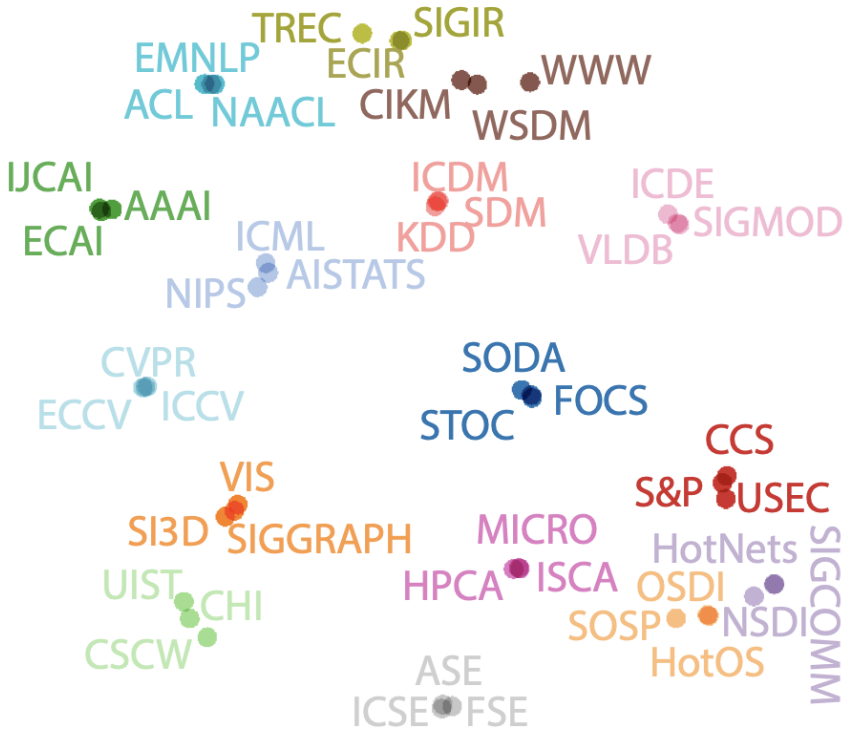


Figure 5: 2D t-SNE projections of the 128D embeddings of 48 CS venues, three each from 16 sub-fields.

Table 4: Node clustering results (NMI) in AMiner data.

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

- k-means 이용, normalized mutual information으로 evaluate
- LINE, PTE에 비해 gain이 큼

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	JAIG	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

Table 6: Case study of similarity search in DBIS Data

Rank	KDD	SIGMOD	SIGIR	WWW	WSDM
0	KDD	SIGMOD	SIGIR	WWW	WSDM
1	SDM	PVLDB	TREC	CIKM	WWW
2	ICDM	ICDE	CIKM	SIGIR	SIGIR
3	DMKD	TODS	IPM	KDD	KDD
4	KDD E	VLDBJ	IRJ	ICDE	AIRWeb
5	PKDD	PODS	ECIR	TKDE	CIKM
6	PAKDD	EDBT	TOIS	VLDB	WebDB
7	TKDE	CIDR	WWW	TOTT	ICDM
8	CIKM	TKDE	JASIST	SIGMOD	VLDB
9	ICDE	ICDT	JASIS	WebDB	VLDBJ
10	TKDD	DE Bull	SIGIR F	WISE	SDM

Scalability

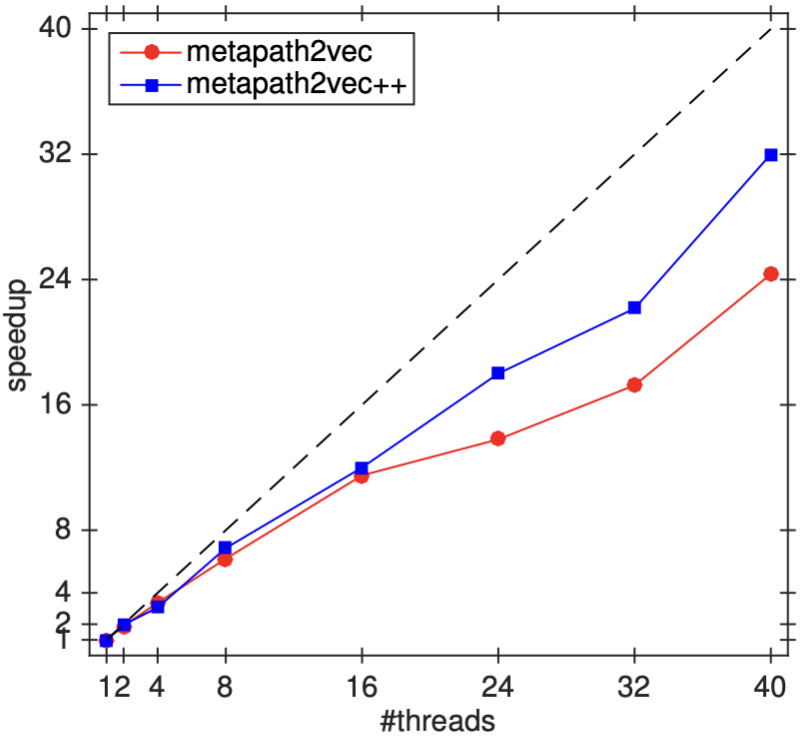


Figure 6: Scalability of *metapath2vec* and *metapath2vec++*.