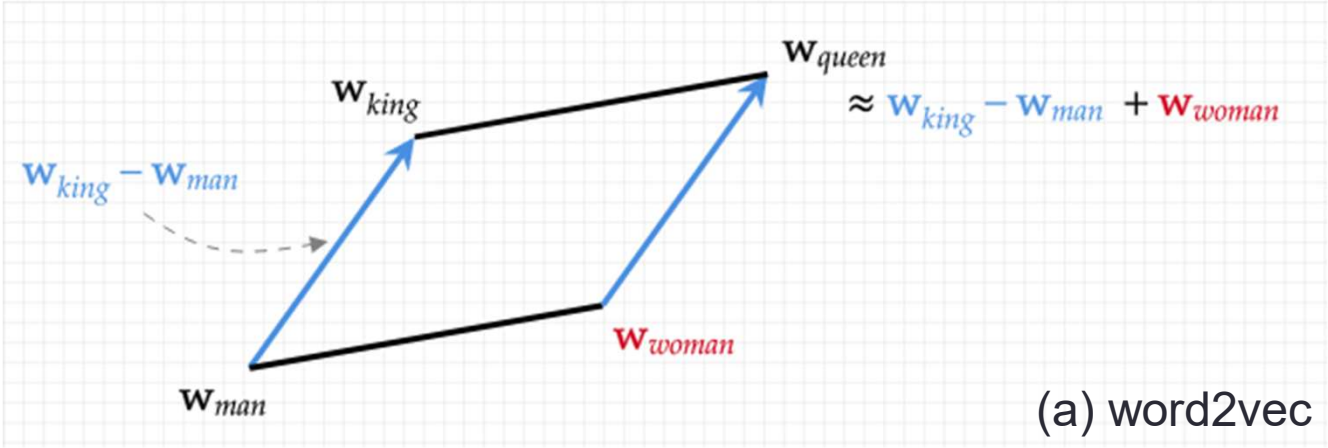
metapath2vec

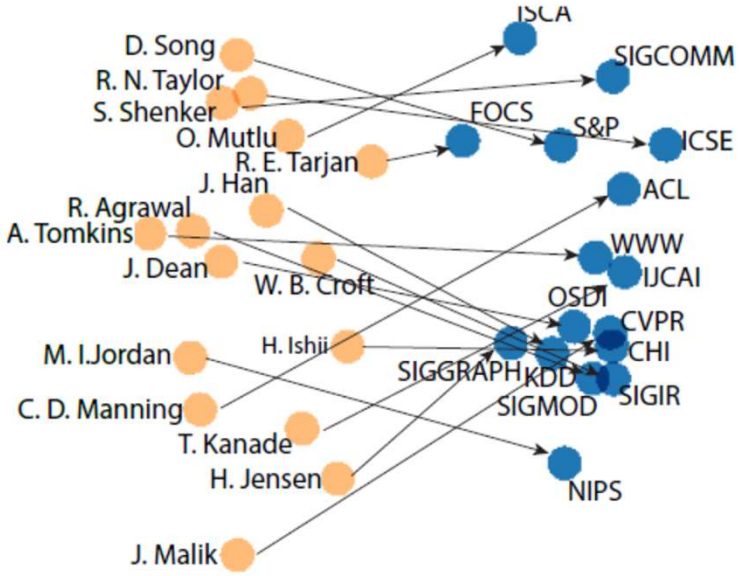
Scalable Representation Learning for Heterogeneous Networks

ALDE Lab.
이 다 영
schematique@pusan.ac.kr

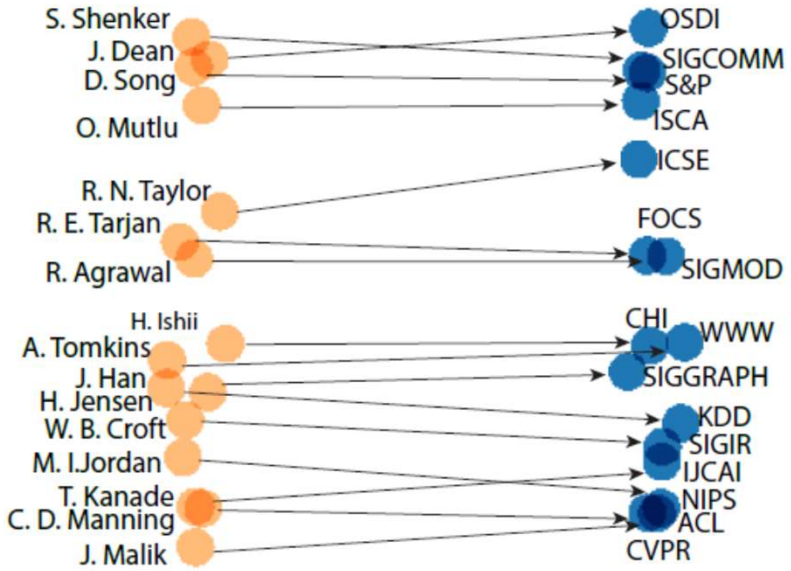
Heterogeneous Graph 임베딩 결과 예시



(a) word2vec



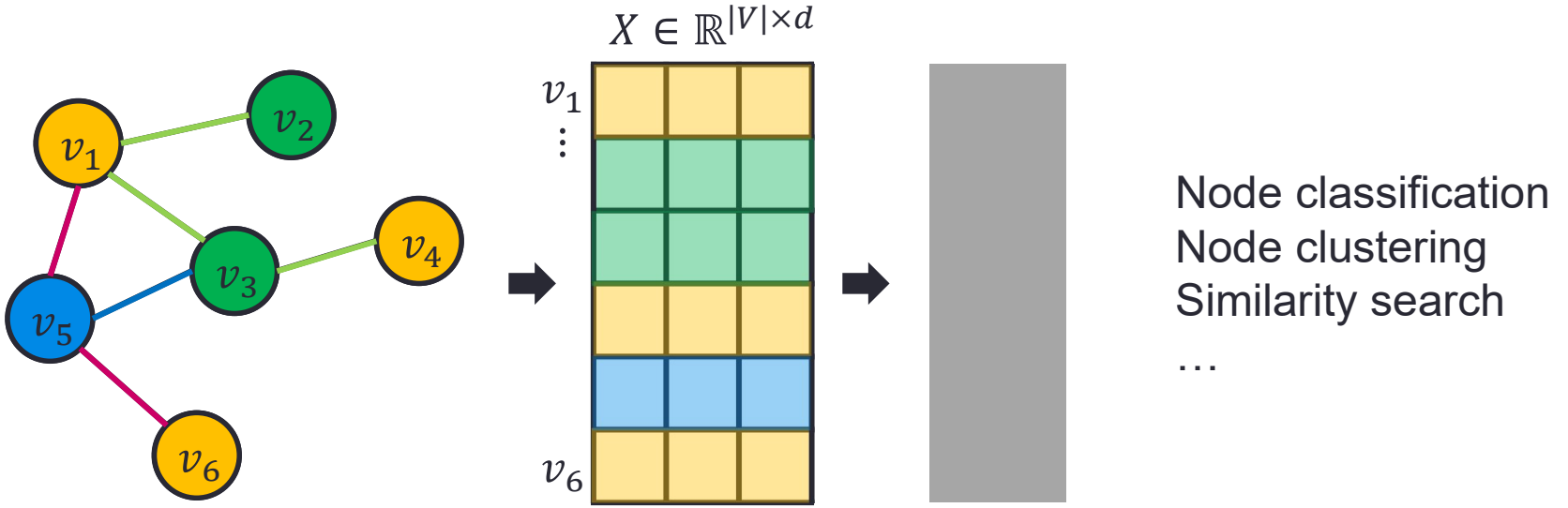
(b) DeepWalk/node2vec



(c) metapath2vec++

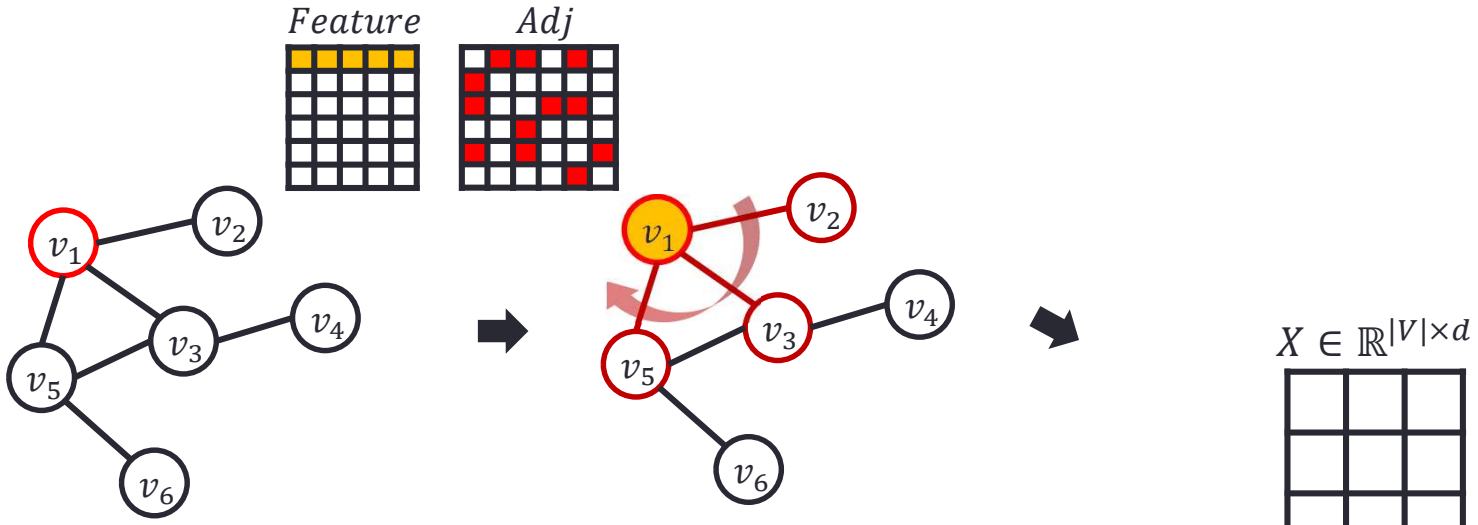
1.1 모델 소개 – 목표

- Meta-path를 이용해 Heterogeneous graph의 구조적, 의미적 관계를 효과적으로 임베딩

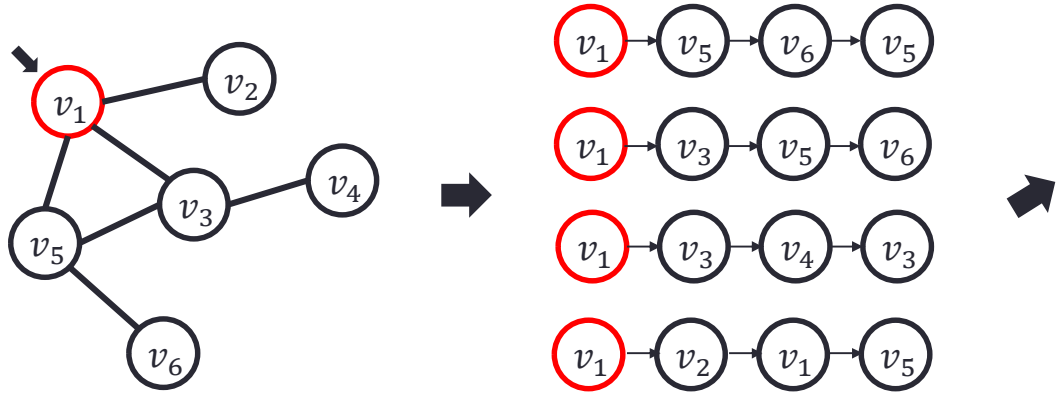


1.2 모델 소개 – 기반 방법론

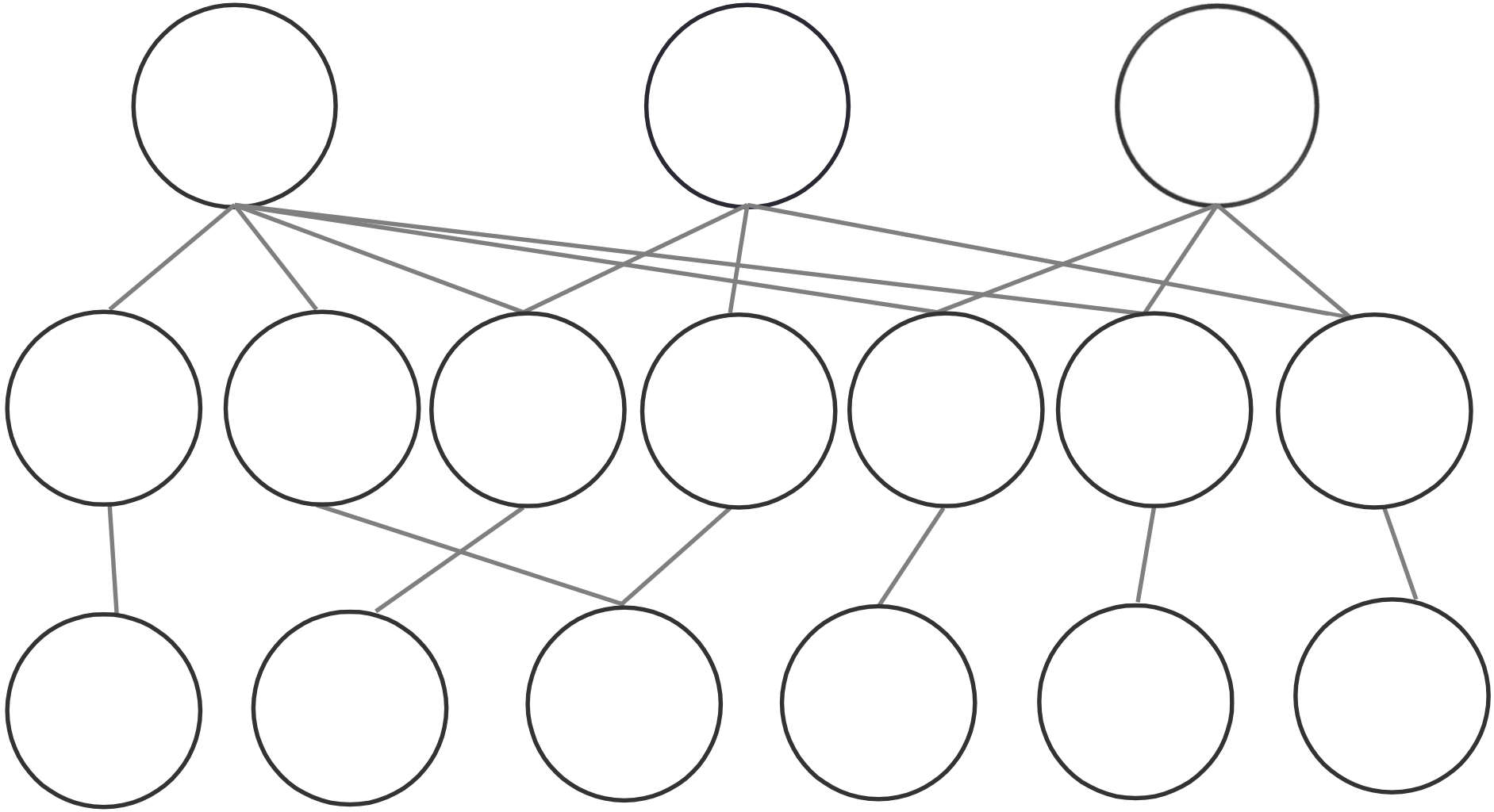
- GCN



- Random walk 기반

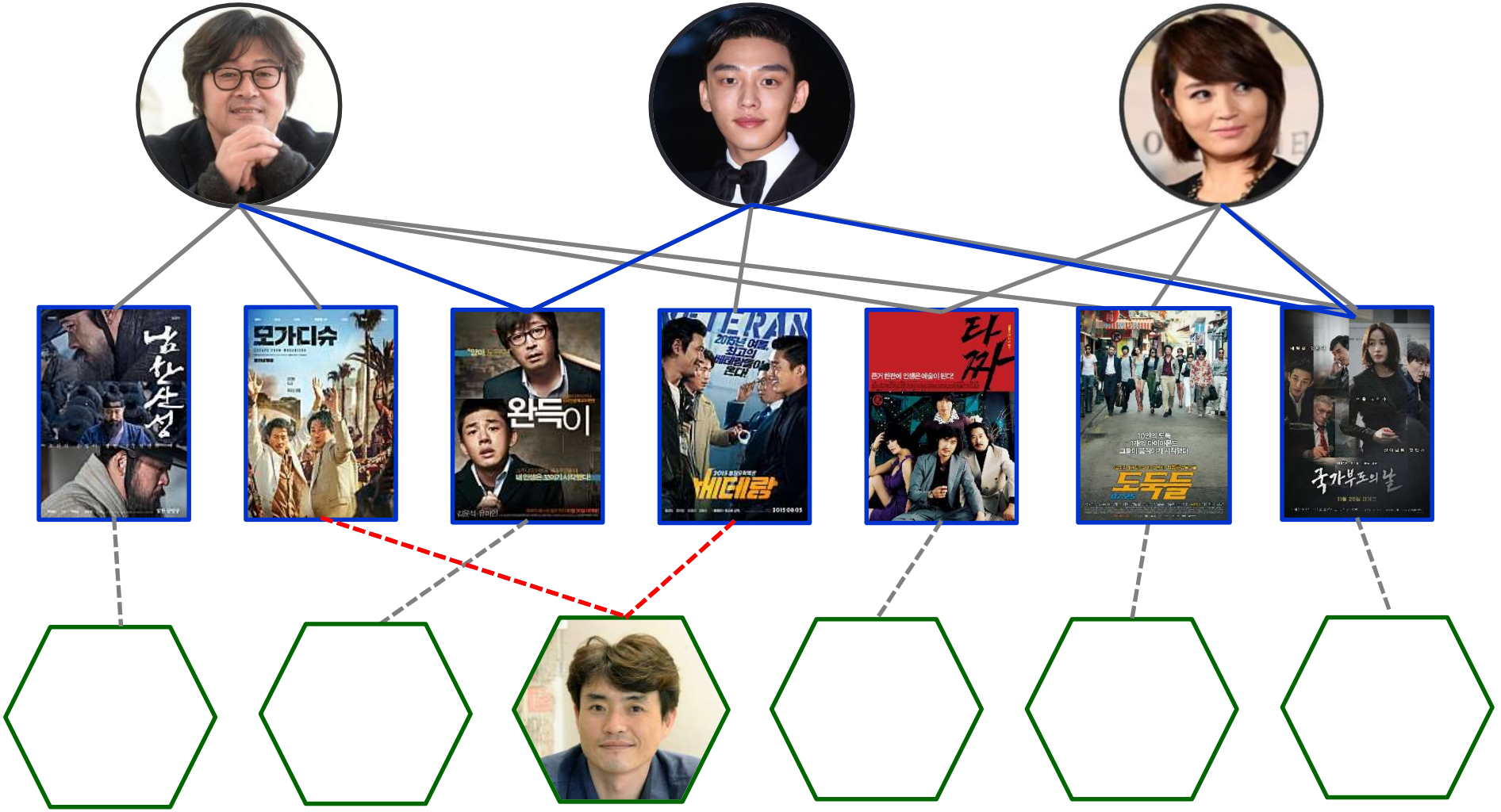


1.3 모델 소개 - 예시



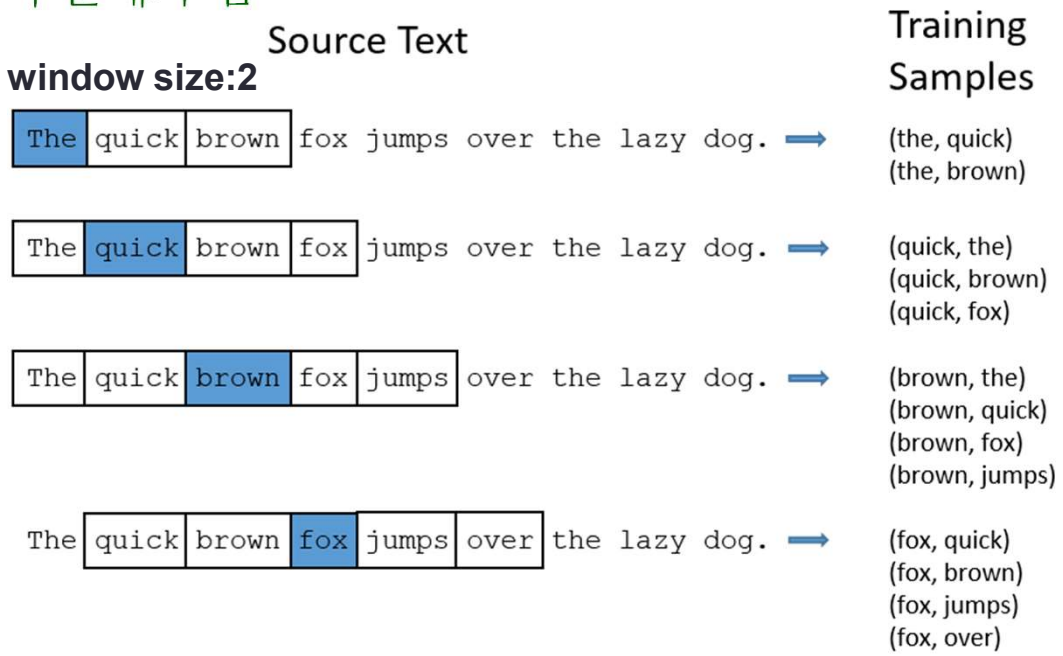
1.4 모델 소개 - 예시

meta-path
→배우-영화-배우
→영화-감독-영화



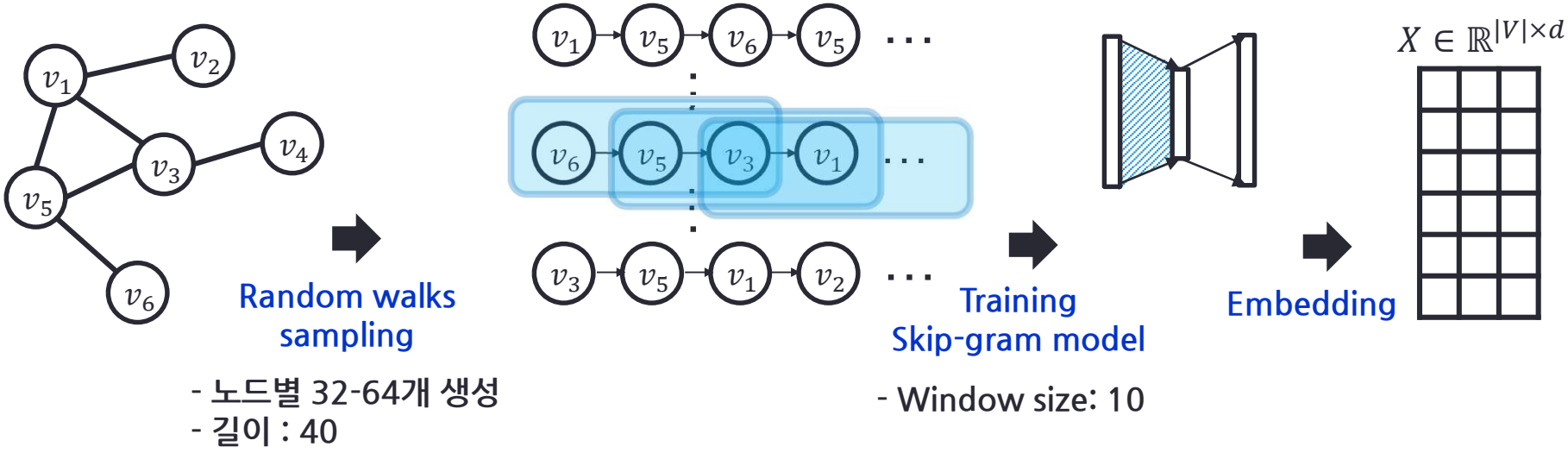
2.1 배경지식 - Word2vec(skip-gram)

- 중심 단어(c)로 주변 단어(o)를 예측하는 모델
  - Input : a text corpus = $\{W\}$
  - Output : $X \in \mathbb{R}^{|V| \times d}$, $d \ll |W|$
  - Network maximization
    - $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)}$
    - 데이터 규모가 커질수록 계산량이 문제가 됨
  - Negative sampling
    - $P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}}$



2.2 이전 연구 - Deep Walk / Node2vec

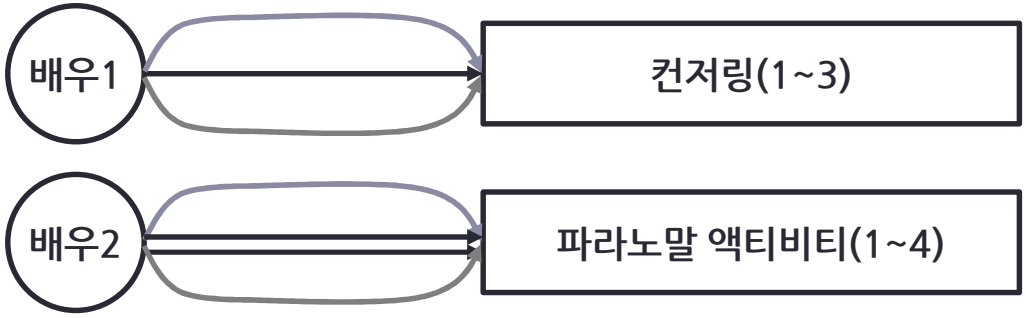
- DeepWalk



- Node2vec
  - DeepWalk는 랜덤워크를 임의로 생성해서 주변 이웃을 잘 파악하지 못함
  - 파라미터를 추가해 랜덤워크 생성 방식을 발전시킨 모델
    - p:직전 node로 돌아올 가능성 조절
    - q:직전 node에서 얼마나 멀어질지를 조절

2.3 이전 연구 - 문제점

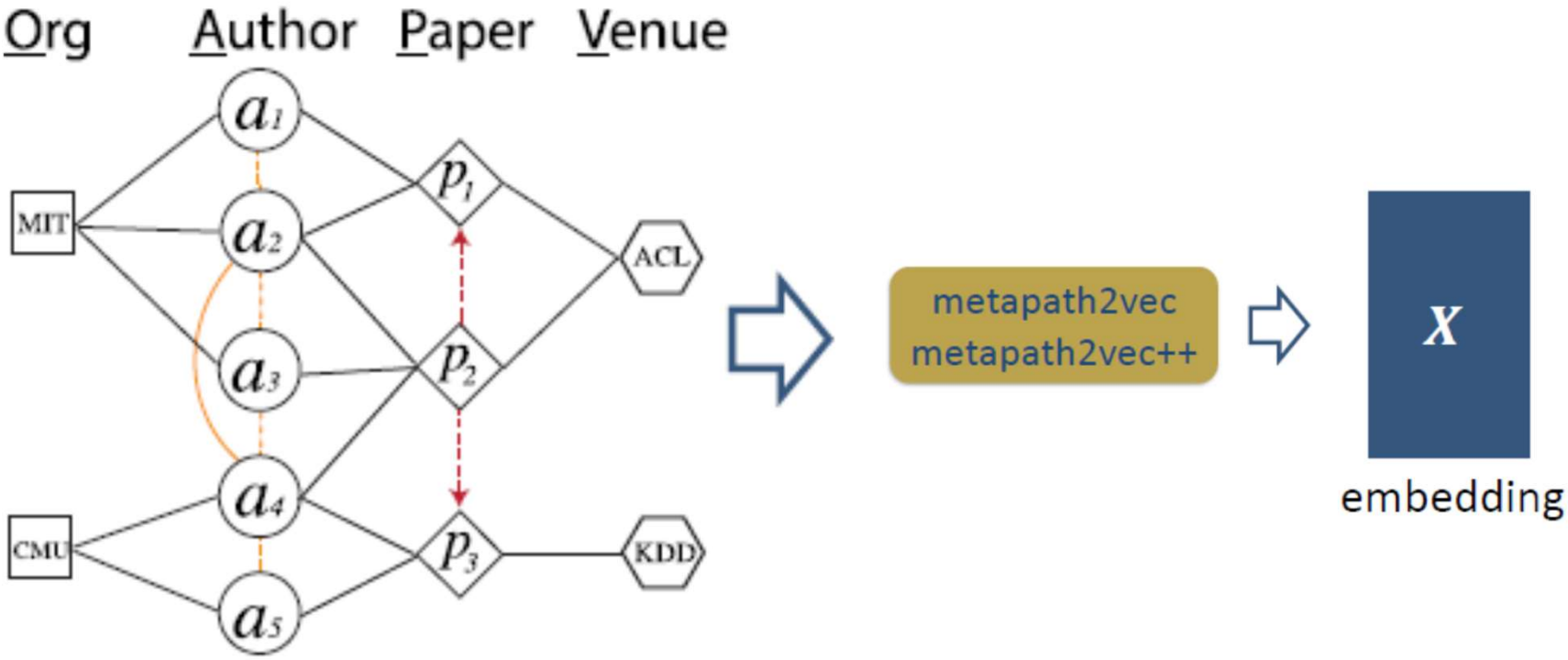
- 직접적인 연결이 없는 노드 관계
  - 두 배우가 각각 시리즈물에 계속 출연한 경우
  - “서로 같은 영화에 출연한 적이 있는가?” 가 기준일 경우, 유사도 0
  - 영화 장르가 ‘공포’라는 잠재적인 의미 분석이 이뤄지면 유사도를 개선할 수 있음



- Homogeneous embedding
  - 서로 다른 여러 타입의 노드, 에지 정보를 하나로 취급, 임베딩 결과에서 차이를 반영하지 못함

3.1 모델 개요

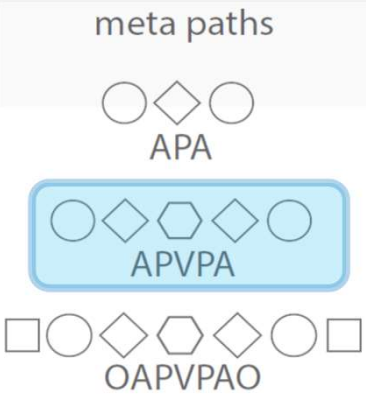
- Input : Heterogeneous network $\mathbf{G}=(V,E,T)$, $T=\{T_V,T_E\}$
- Output : $X \in \mathbb{R}^{|V|\times d}, d \ll |V|$
- Goal : X에 서로 다른 타입의 노드들의 구조/의미를 임베딩



3.2 meta-path를 Random walk에 적용

- Random walk를 생성시 meta-path에 부합하도록 강제
  - $\mathrm{G}=(V,E,T)$, $T=\{T_V,T_E\}$, $\phi(v):V\rightarrow T_V$
  - $N_t(v)$: v의 t 타입의 이웃 노드들
- Meta-path scheme
  - $\mathcal{P}:V_1\rightarrow V_2\rightarrow\cdots\rightarrow V_{t+1}\rightarrow\cdots\rightarrow V_l$
- Transition probability
  - $$p\left(v^{i+1}\middle|v_t^i,\mathcal{P}\right)=\begin{cases}\frac{1}{|N_{t+1}(v_t^i)|} & \left(v^{i+1},v_t^i\right)\in E,\phi(v^{i+1})=t+1\\ 0 & \left(v^{i+1},v_t^i\right)\in E,\phi(v^{i+1})\neq t+1\\ 0 & \left(v^{i+1},v_t^i\right)\notin E\end{cases}$$
- Recursive guidance
  - $p\left(v^{i+1}\middle|v_t^i\right)=p\left(v^{i+1}\middle|v_1^i\right),\ if\ t=l$

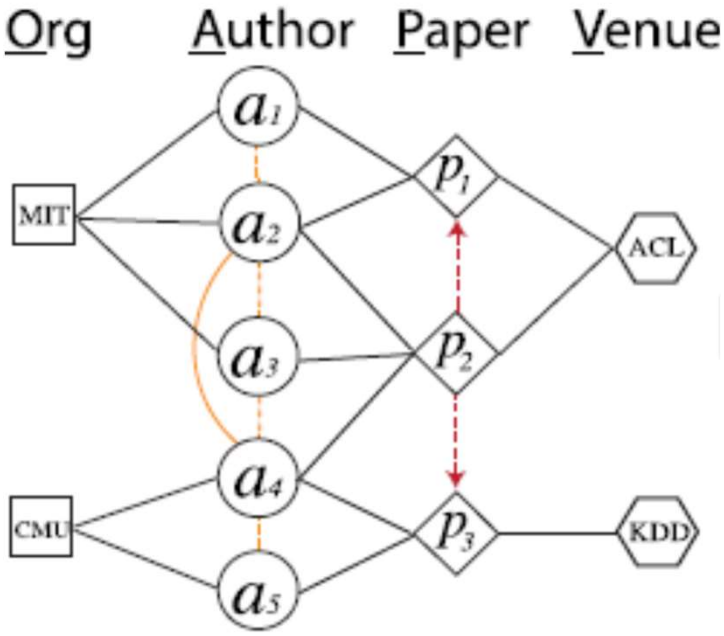
3.3 metapath2vec 모델

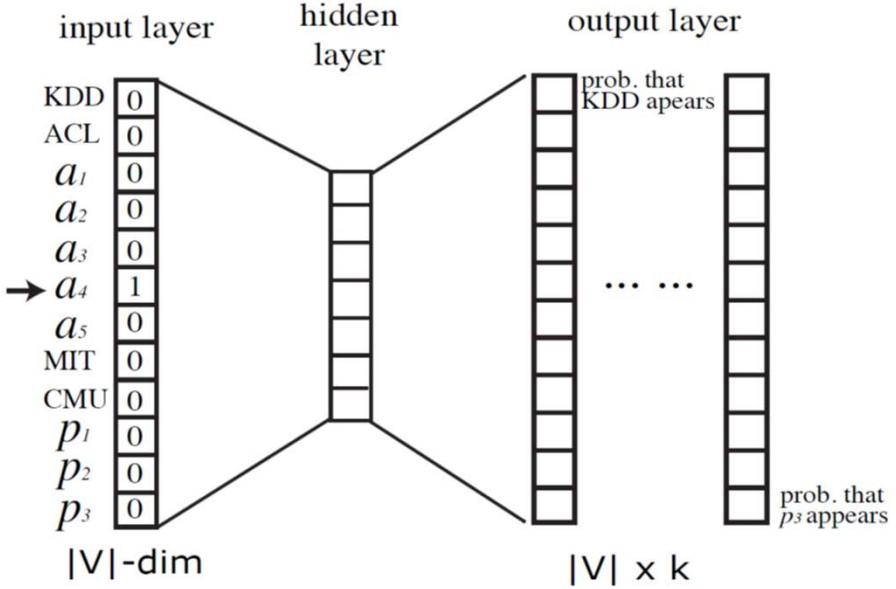


- Network maximization
  - $p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot e^{X_v}}}{\sum_{u \in V} e^{X_u} \cdot e^{X_v}}$

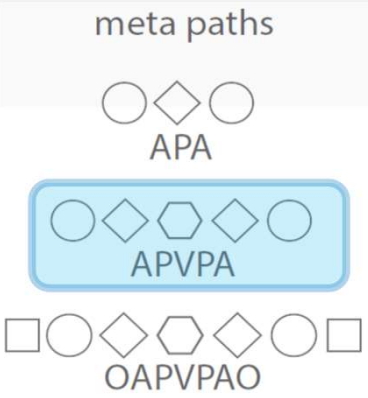
A P V P A P V P A

$a_3 \rightarrow p_2 \rightarrow ACL \rightarrow p_1 \rightarrow a_1 \rightarrow p_1 \rightarrow ACL \rightarrow p_2 \rightarrow a_4$ ...





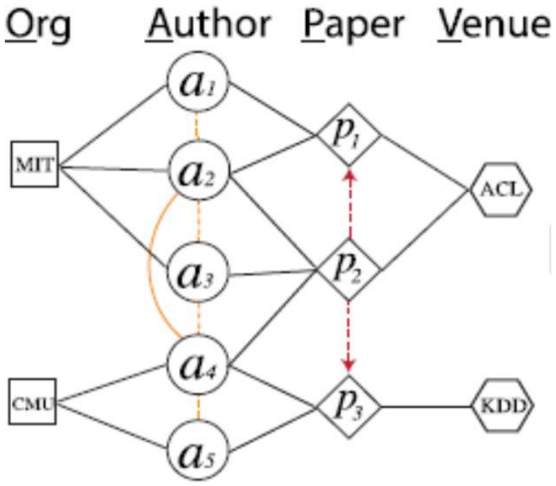
3.4 metapath2vec++ 모델

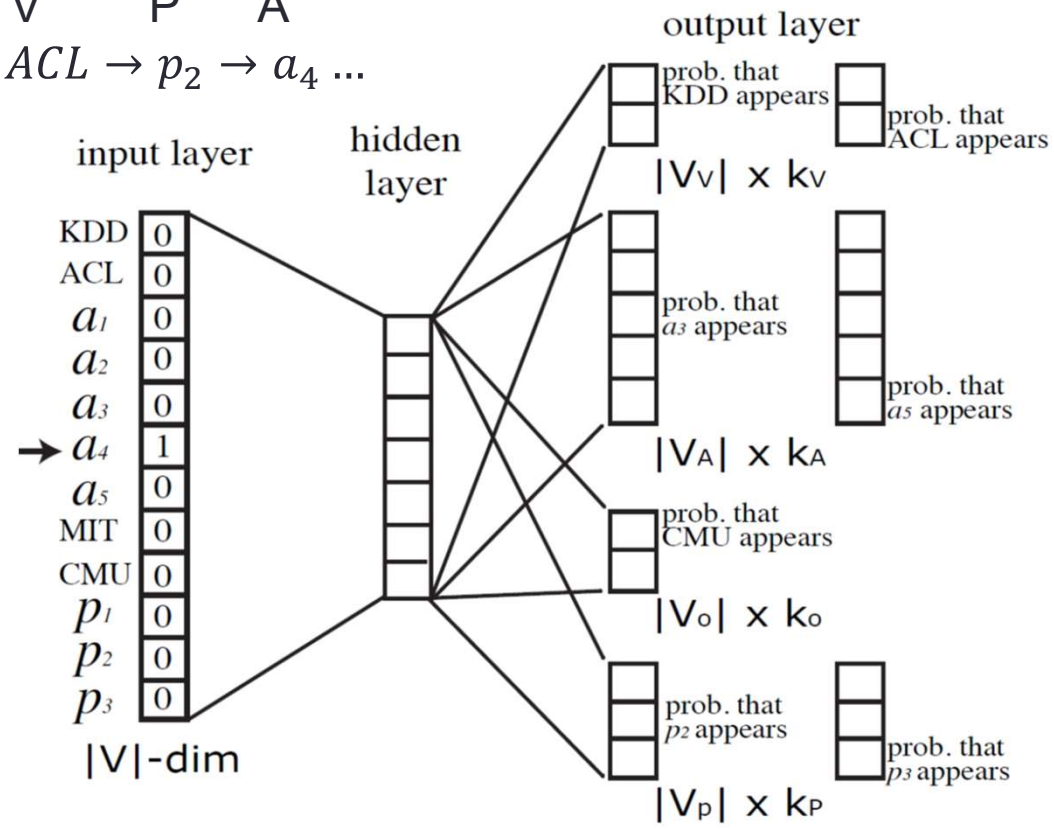


- Network maximization
  - $p(c_t|v;\theta) = \frac{e^{X_{c_t}.e^{X_v}}}{\sum_{u_t \in V_t} e^{X_{u_t}.e^{X_v}}}$

A P V P A P V P A

$a_3 \rightarrow p_2 \rightarrow ACL \rightarrow p_1 \rightarrow a_1 \rightarrow p_1 \rightarrow ACL \rightarrow p_2 \rightarrow a_4$ ...





4.1 실험 데이터셋 및 설정값

- Dataset
  - AMiner computer science dataset
    - 1.7M authors, 3M papers, 3800 venues(conference+journal)
    - Category
      - Linguistics, graphics, networks, vision, system, DB, human computer interaction, theoretical computer science
  - DBIS(Information Systems dataset)
    - 5000 authors, 70,000 papers, 400 venues
- Settings
  - Meta-path: "APVPA"
  - 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

4.2 실험1 - Node classification(venue)

- AMiner 데이터에서 각 author, venue 를 8개 category로 분류
  - Venue : 133개
  - Author : published 논문 과반수로 결정, 246,678개
- 전체 그래프로 임베딩 학습 후 training data 비율을 변경하면서 logistic regression 분류 학습
- Venue 분류 결과

Training data 비율→

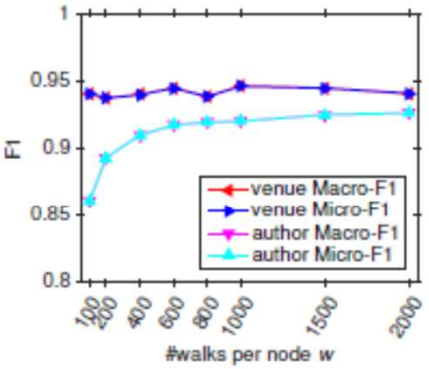
Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.0723	0.1396	0.1905	0.2795	0.3427	0.3911	0.4424	0.4774	0.4955	0.4457
	LINE (1st+2nd)	0.2245	0.4629	0.7011	0.8473	0.8953	0.9203	0.9308	0.9466	0.9410	0.9466
	PTE	0.1702	0.3388	0.6535	0.8304	0.8936	0.9210	0.9352	0.9505	0.9525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

4.3 실험1 – Node classification(author)

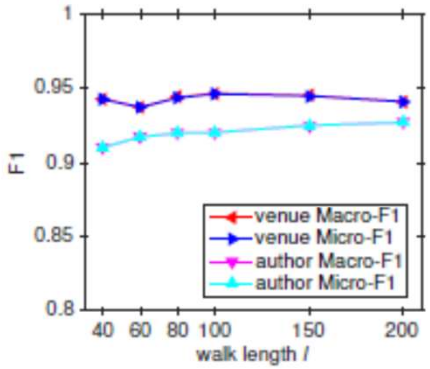
- Author 분류 결과

Training data 비율→

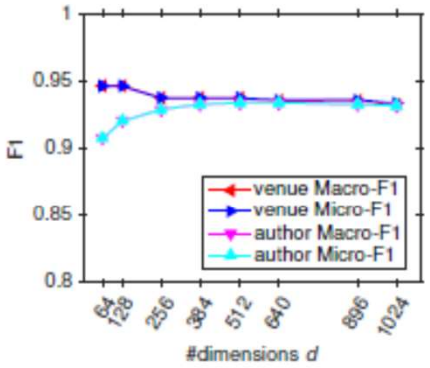
Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.7153	0.7222	0.7256	0.7270	0.7273	0.7274	0.7273	0.7271	0.7275	0.7275
	LINE (1st+2nd)	0.8849	0.8886	0.8911	0.8921	0.8926	0.8929	0.8934	0.8936	0.8938	0.8934
	PTE	0.8898	0.8940	0.897	0.8982	0.8987	0.8990	0.8997	0.8999	0.9002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



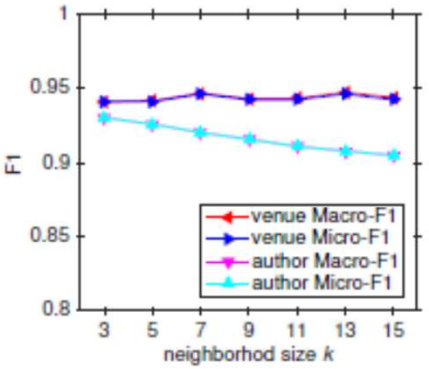
(a) #walks per node w



(b) walk length l



(c) #dimensions d

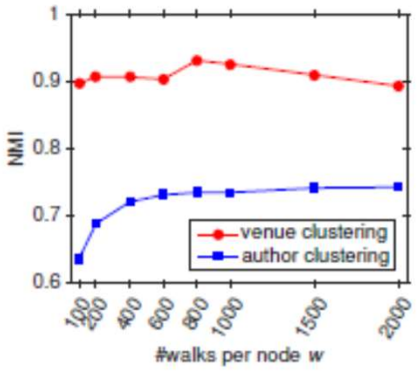


(d) neighborhood size k

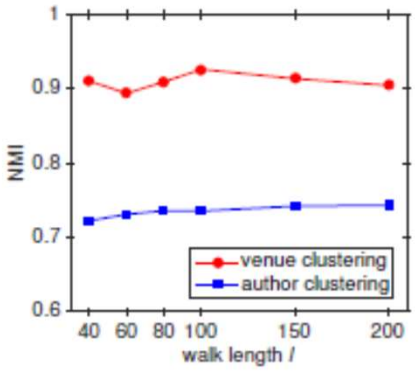
4.4 실험2 - Node clustering 성능

- 8개 category로 author, venue 구분된 데이터 사용
- 임베딩 후 k-means 알고리즘 으로 클러스터링
- NMI(normalized mutual information)으로 평가

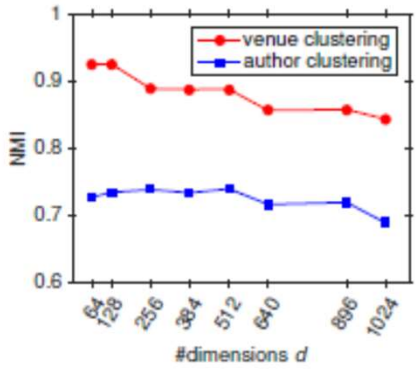
methods	venue	author
DeepWalk/node2vec	0.1952	0.2941
LINE (1st+2nd)	0.8967	0.6423
PTE	0.9060	0.6483
metapath2vec	0.9274	0.7470
metapath2vec++	0.9261	0.7354



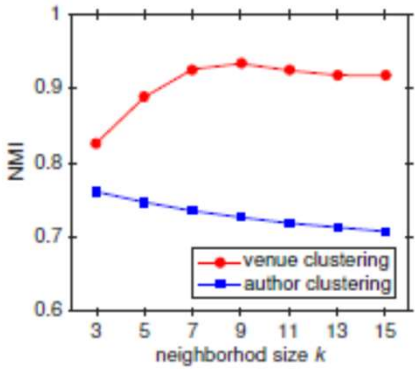
(a) #walks per node w



(b) walk length l



(c) #dimensions d



(d) neighborhood size k

4.5 실험3 – Similarity search

- 임베딩 결과에서 cosine similarity 계산
  - Input : 학회
  - Output : 쿼리와 가장 유사도 높은 10개 학회

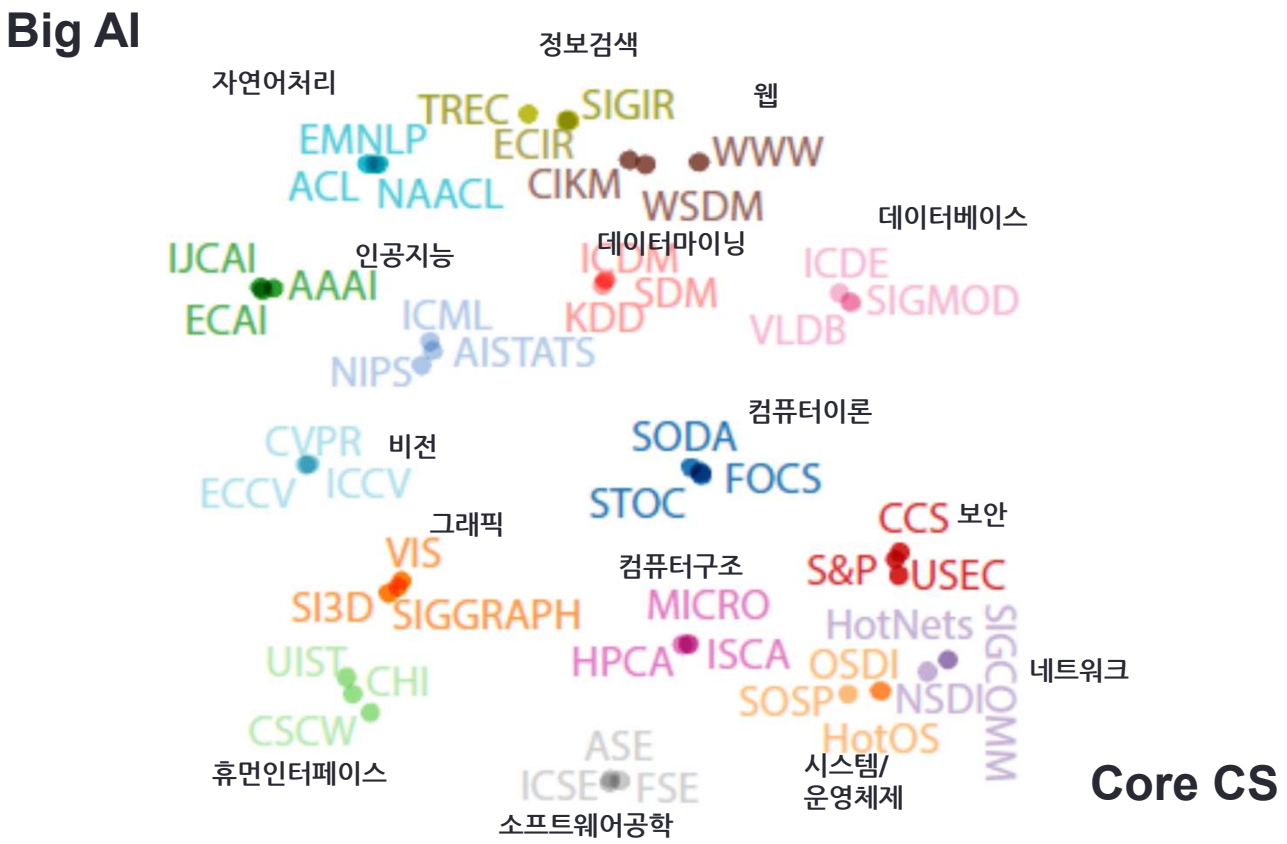
- 상위 3개 – 쿼리 학회와 비슷한 명망이 있는 학회 포함

Query venue→

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

4.6 실험4 - metapath2vec++ 시각화

- (16개 cs sub-filed) * (3개 학회) 임베딩(128d) t-SNE 시각화



5.1 예제코드 수행결과 - 저자의 분야 예측

- Parameter
  - embedding_dim=128
  - metapath="APVPA"
  - walk_length=50
  - context_size=7
  - walks_per_node=5
  - num_negative_samples=5

- 데이터 - Aminer
  - |paper|=3.1M
  - |author|=1.7M
  - |venue|=3,883

- 실험결과
  - "APVPA" : 91.83%
  - "APA": 66.18%

- 수정사항

library, win10, gpu 호환문제
loader = model.loader(batch_size=128, shuffle=True, num_workers=12)
(0으로 변경해서 실험)

```python
def test(train_ratio=0.1):
    model.eval()

    z = model('author', batch=data.y_index_dict['author'])
    y = data.y_dict['author']

    perm = torch.randperm(z.size(0))
    train_perm = perm[:int(z.size(0) * train_ratio)]
    test_perm = perm[int(z.size(0) * train_ratio):]

    return model.test(z[train_perm], y[train_perm], z[test_perm], y[test_perm],
                      max_iter=150)


for epoch in range(1, 6):
    train(epoch)
    acc = test()
    print(f'Epoch: {epoch}, Accuracy: {acc:.4f}')
```

```
Epoch: 5, Step: 11400/13231, Loss: 0.8353
Epoch: 5, Step: 11500/13231, Loss: 0.8354
Epoch: 5, Step: 11600/13231, Loss: 0.8359
Epoch: 5, Step: 11700/13231, Loss: 0.8357
Epoch: 5, Step: 11800/13231, Loss: 0.8357
Epoch: 5, Step: 11900/13231, Loss: 0.8353
Epoch: 5, Step: 12000/13231, Loss: 0.8353
Epoch: 5, Step: 12000/13231, Acc: 0.9180
Epoch: 5, Step: 12100/13231, Loss: 0.8350
Epoch: 5, Step: 12200/13231, Loss: 0.8351
Epoch: 5, Step: 12300/13231, Loss: 0.8349
Epoch: 5, Step: 12400/13231, Loss: 0.8352
Epoch: 5, Step: 12500/13231, Loss: 0.8352
Epoch: 5, Step: 12600/13231, Loss: 0.8352
Epoch: 5, Step: 12700/13231, Loss: 0.8349
Epoch: 5, Step: 12800/13231, Loss: 0.8349
Epoch: 5, Step: 12900/13231, Loss: 0.8348
Epoch: 5, Step: 13000/13231, Loss: 0.8349
Epoch: 5, Step: 13100/13231, Loss: 0.8347
Epoch: 5, Step: 13200/13231, Loss: 0.8349
Epoch: 5, Accuracy: 0.9183
```

5.2 Pilot 실험 - 영화 장르 예측

- 네이버 영화 정보 크롤링
  - 총 영화(M) 400개, 배우(A) 990명, 감독(D) 354명
    - 10424건 크롤링 한 것 중 장르, 감독 정보가 누락되지 않고 배우가 3명 이상
    - 총 8가지 장르에 대해 각 50개
      - 드라마, 멜로/로맨스, 코미디, 다큐멘터리, 액션, 애니메이션, 스릴러, 범죄

- 실험설정
  - embedding_dim=32
  - metapath="AMA", "MDM", "MAM", "AMDMA", "DMAMD"
  - walk_length=30
  - context_size=5
  - walks_per_node=5
  - num_negative_samples=3

5.3 Pilot 실험 - 장르 예측결과

- 사용한 metapath별 예측 정확도
  - 영화(M), 배우(A), 감독(D)

metapath="AMA"

```
Epoch: 1, Accuracy: 0.1500
Epoch: 2, Accuracy: 0.1250
Epoch: 3, Accuracy: 0.1250
Epoch: 4, Accuracy: 0.1750
Epoch: 5, Accuracy: 0.1500
Epoch: 6, Accuracy: 0.1250
Epoch: 7, Accuracy: 0.2250
Epoch: 8, Accuracy: 0.2000
Epoch: 9, Accuracy: 0.1750
Epoch: 10, Accuracy: 0.1500
```

metapath="MAM"

```
Epoch: 1, Accuracy: 0.2000
Epoch: 2, Accuracy: 0.1500
Epoch: 3, Accuracy: 0.2000
Epoch: 4, Accuracy: 0.1500
Epoch: 5, Accuracy: 0.1500
Epoch: 6, Accuracy: 0.2000
Epoch: 7, Accuracy: 0.2000
Epoch: 8, Accuracy: 0.1750
Epoch: 9, Accuracy: 0.2750
Epoch: 10, Accuracy: 0.2000
```

metapath="MDM"

```
Epoch: 1, Accuracy: 0.1250
Epoch: 2, Accuracy: 0.0500
Epoch: 3, Accuracy: 0.1000
Epoch: 4, Accuracy: 0.1000
Epoch: 5, Accuracy: 0.1250
Epoch: 6, Accuracy: 0.1250
Epoch: 7, Accuracy: 0.1000
Epoch: 8, Accuracy: 0.1500
Epoch: 9, Accuracy: 0.1750
Epoch: 10, Accuracy: 0.0750
```

metapath="DMAMD"

```
Epoch: 1, Accuracy: 0.2000
Epoch: 2, Accuracy: 0.1250
Epoch: 3, Accuracy: 0.1750
Epoch: 4, Accuracy: 0.2250
Epoch: 5, Accuracy: 0.1250
Epoch: 6, Accuracy: 0.1250
Epoch: 7, Accuracy: 0.2250
Epoch: 8, Accuracy: 0.1500
Epoch: 9, Accuracy: 0.2000
Epoch: 10, Accuracy: 0.1250
```

metapath="AMDMA"

```
Epoch: 1, Accuracy: 0.2000
Epoch: 2, Accuracy: 0.1750
Epoch: 3, Accuracy: 0.1250
Epoch: 4, Accuracy: 0.0750
Epoch: 5, Accuracy: 0.1750
Epoch: 6, Accuracy: 0.0750
Epoch: 7, Accuracy: 0.1500
Epoch: 8, Accuracy: 0.1500
Epoch: 9, Accuracy: 0.1000
Epoch: 10, Accuracy: 0.2500
```

6. 결론 + 추후연구

- Heterogeneous network에서
  meta-path로 가이드 되는 랜덤워크 전략을 제안,
  구조적/의미적 상호관계 모두를 표현할 수 있는 임베딩

- 다양한 heterogeneous network mining task에서
  다른 모델과 비교했을 때 더 나은 성능을 보임

- 추후연구
  - 주어진 문제에 적합한 의미 있는 meta-path를 자동으로 학습

감사합니다
QnA