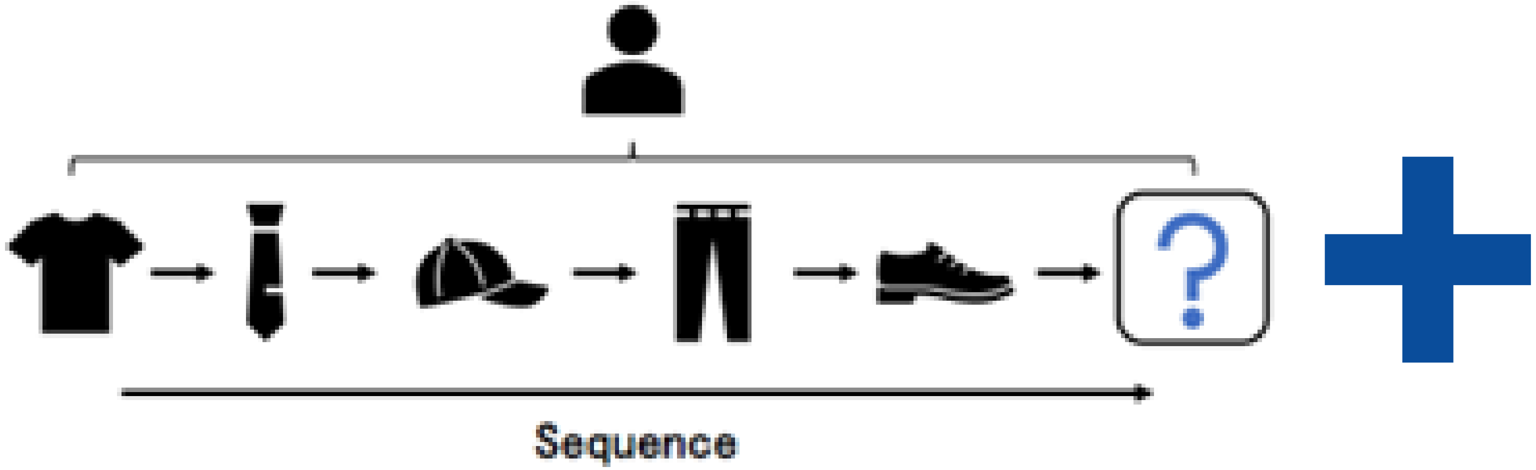
GES-SASRec

이은경

목차

1. introduction

1. Introduction



- sequential dependenc는 다양한 recommendation senarios에 따라 크게 달라짐(user에게는 여러 seq존재)
- sequential item 관계만 고려하고 recommandation similarity 측정에 중요한 semantic item 관계는 무시

> User interaction에서 얻은 item similarity를 보완-정규화하여 recommandation 정확도를 향상

2. preliminary

1) Problem Formulation

2. Preliminary

- $\mathrm{U} = \{\mathrm{u}_1, \mathrm{u}_2, \cdots, \mathrm{u}_{|\mathrm{U}|}\}$ user set, $I = \{i_1, i_2, \cdots, i_{|\mathrm{I}|}\}$ items set, $S^{(u)} = (i_1^{(u)}, i_2^{(u)}, \cdots, i_{\mathrm{n_u}}^{(u)})$ user u의 interaction sequence, n_u : length of the sequence
- 목표 : user u의 과거 상호작용 $S^{(u)}$ 를 고려하여 next itme 예측

$$p\left(i_{n_u+1}^{(u)} | S^{(u)}\right). \qquad (1)$$

- 상위 N개 item을 내림차순으로 확률에 따라 추천

2) SASRec (Self-Attention based Sequential Recommendation)

- Transformer[1]의 self-attention architecture를 통해 sequential dependency 모델링

1. look-up을 embedding 하여 user interaction sequences를 vector sequences로 변환
2. Trainable positional embedding이 input embedding에 주입
3. 두 개의 feed-forward layers 가 포함된 self-attention layer로 sequences를 encode
4. residual connection로 self-attention blocks을 쌓은 후 time step t에서 final representation vector는 sequence representation로 처리됨
5. sequence representation 과 candidate item embedding 사이의 similarity score는 dot-product에 의해 계산. cross-entropy loss이 모델 훈련에 사용

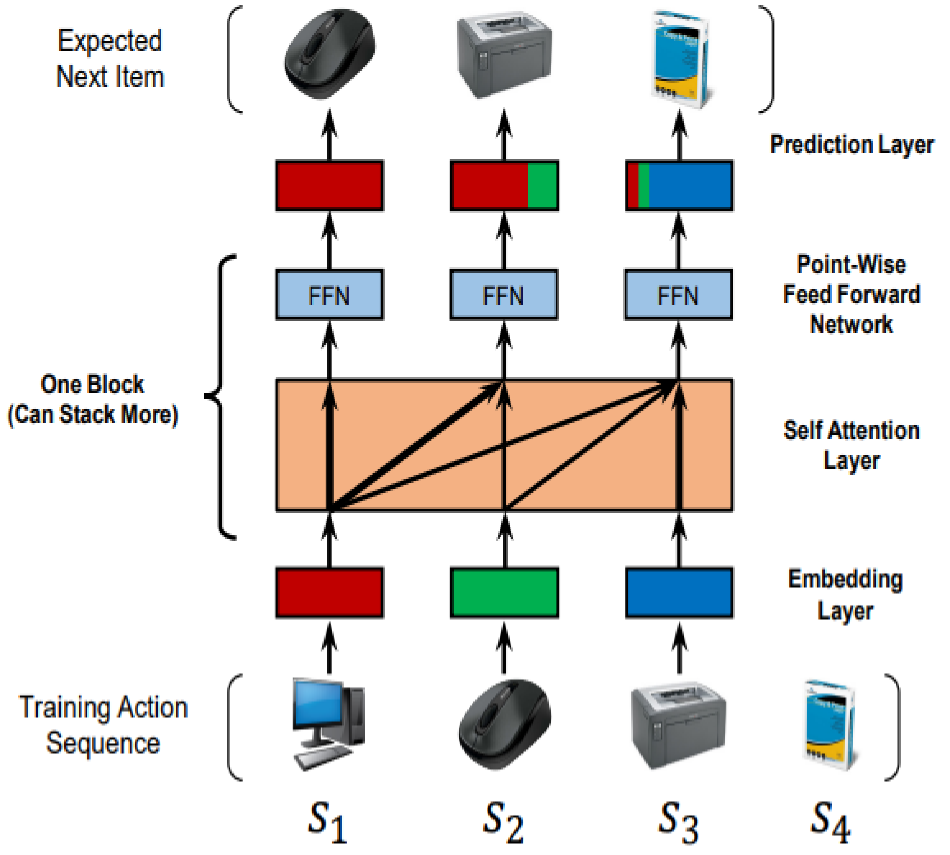


Figure 1: A simplified diagram showing the training process of SASRec. At each time step, the model considers all previous items, and uses attention to 'focus on' items relevant to the next action.

3) Graph Convolution

〈 Early efforts 〉

- normalized graph Laplacian의 eigen decomposition를 계산하여 Fourier domain 에서 graph convolution를 정의
- graph convolution은 parametric filte로 multiplication of a signal으로 정의 가능
- Eigen decomposition을 계산하지 않기 위해 filte를 Chebyshev polynomials에 의한 ChebNet [8] approximates
- GCN(GraphConvolutional Network)[6]은 first-orde approximation를 도입함으로써 ChebNet을 단순화

$$\mathbf{H}^{(k)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{\Theta}^{(k)}), \qquad (2)$$

- $\mathrm{H}^{(\mathrm{k})} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$: hidden embedding matrix in the k- th layer / $\Theta^{(k)} \in \mathbb{R}^{\mathrm{d}\times\mathrm{d}}$: filter parameter matrix in the k- th layer / $\mathrm{H}^{(0)}$= X : node feature matrix / $\sigma(\cdot)$: activation function / $\hat{\mathrm{A}} = \tilde{\mathrm{D}}^{-1/2}\tilde{\mathrm{A}}\ \tilde{\mathrm{D}}^{-1/2}$: renormalized
- adjacency matrix of the graph

3. Framework

3. Framework

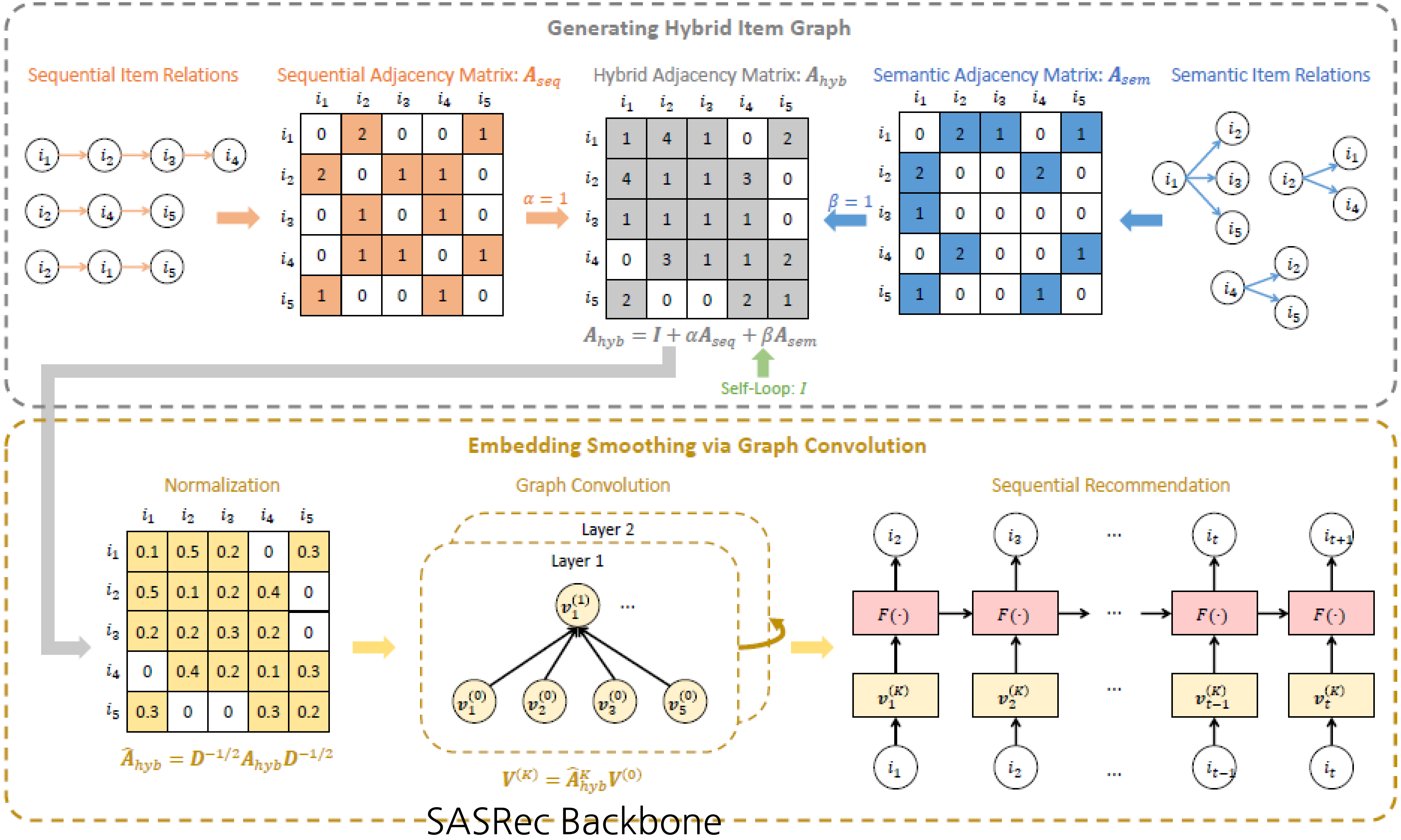
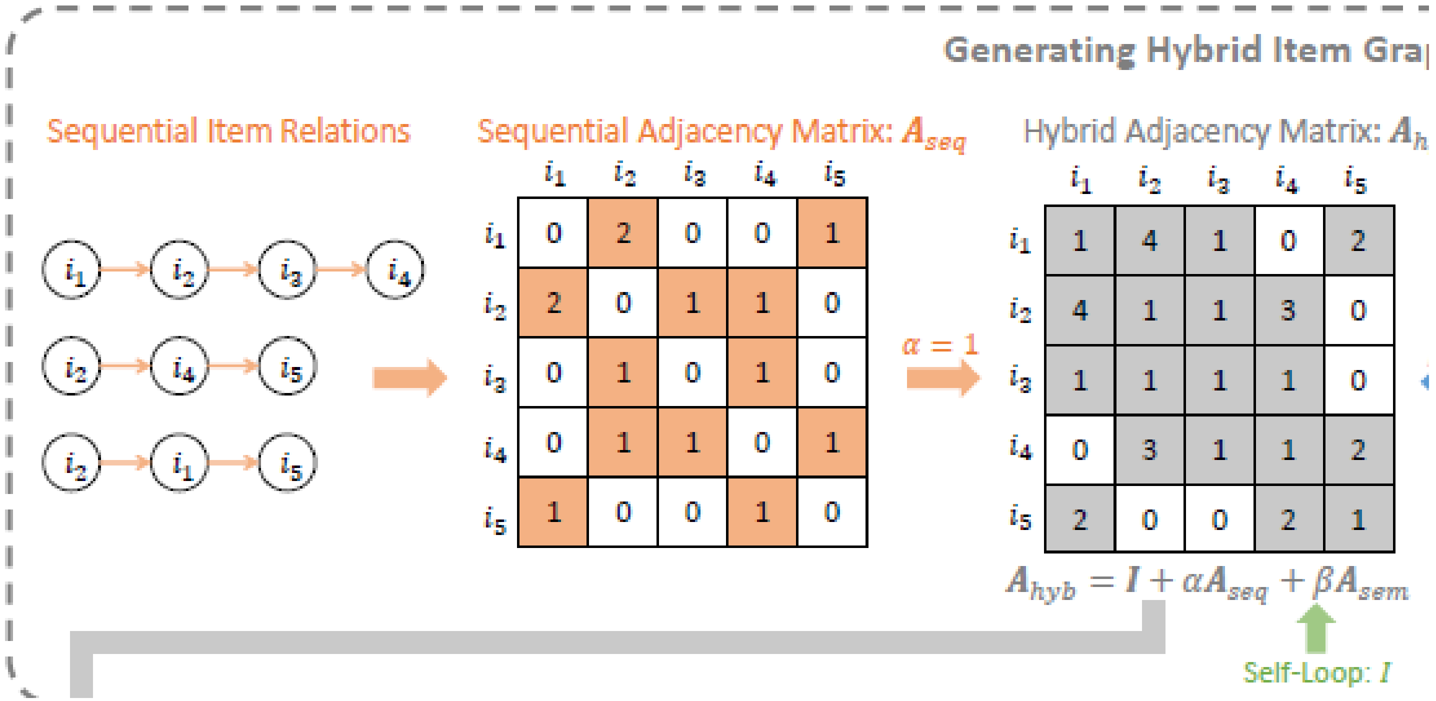


Fig. 2: The framework of Graph-based Embedding Smoothing (GES).

1) Sequential Item Graph

- 모든 user interaction sequence를 aggregate하여 user behavior 측면에서 item 간 관계를 재현하는 global sequential item graph 구축
- Sequential item graph의 adjacency matrix $A_{seq} \in \mathbb{R}^{|I| \times |I|}$
- 방향이 지정되지 않은 weighted graph로 구성
- 목표 : item 그래프에 따라 item embedding을 smoothing. item graph 구성시 adjacency 에 주로 초점
- user behavior 의 sequentiality 은 SASRec의 sequential architecture 에 의해 모델링



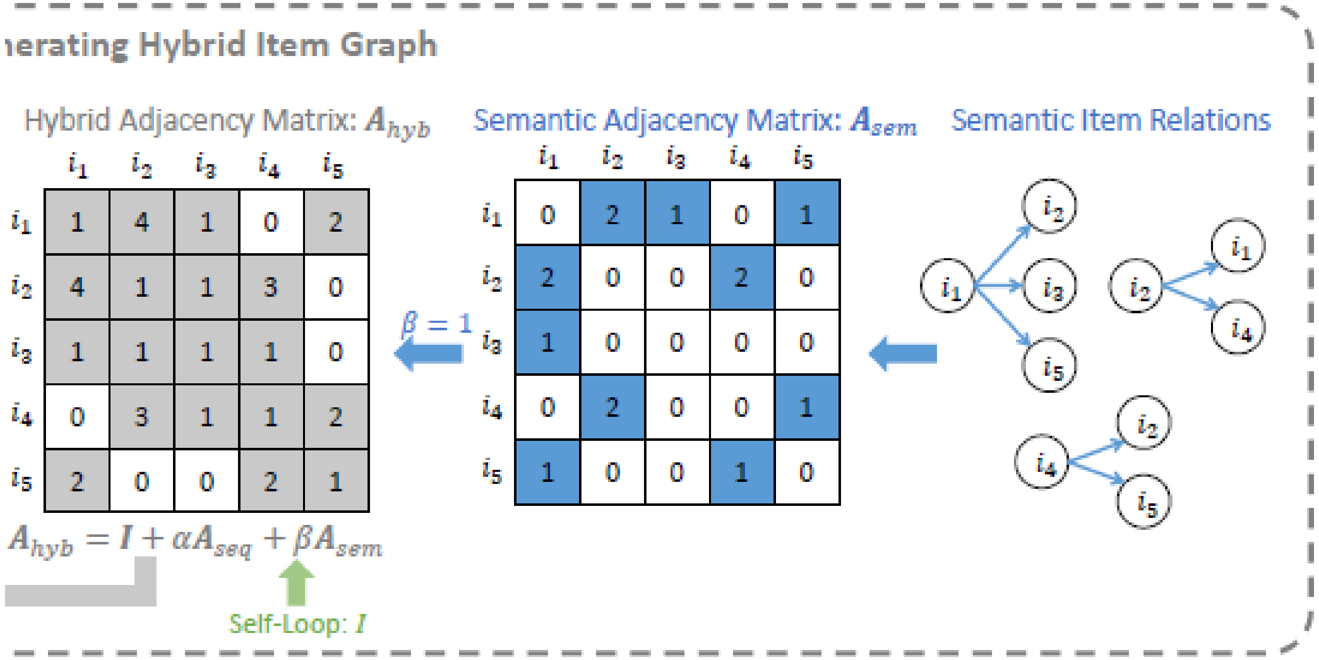
	i_1	i_2	i_3	i_4	i_5
i_1	0	2	0	0	1
i_2	2	0	1	1	0
i_3	0	1	0	1	0
i_4	0	1	1	0	1
i_5	1	0	0	1	0

	i_1	i_2	i_3	i_4	i_5
i_1	1	4	1	0	2
i_2	4	1	1	3	0
i_3	1	1	1	1	0
i_4	0	3	1	1	2
i_5	2	0	0	2	1

2) Semantic Item Graph

- semantic item relations : item에서 관찰가능한 attributes나 features의 관점에서 item 간의 연결
- Semantic item graph의 adjacency matrix $A_{sem} \in \mathbb{R}^{|I| \times |I|}$
- Semantic item 그래프를 무방향 weighted graph로 구성하고 weight는 relation strength 반영

$$a_{ij} = \begin{cases} 2 & \text{if } i \text{ has a bidirectional relation with } j, \\ 1 & \text{if } i \text{ has a unidirectional relation with } j, \\ 0 & \text{otherwise.} \end{cases}$$



	i_1	i_2	i_3	i_4	i_5
i_1	1	4	1	0	2
i_2	4	1	1	3	0
i_3	1	1	1	1	0
i_4	0	3	1	1	2
i_5	2	0	0	2	1

	i_1	i_2	i_3	i_4	i_5
i_1	0	2	1	0	1
i_2	2	0	0	2	0
i_3	1	0	0	0	0
i_4	0	2	0	0	1
i_5	1	0	0	1	0

3) Graph Fusion

- embedding smoothing를 위한 sequential 와 semantic item graphs의 결합을 위해 각 그래프에서 item embedding을 전파한 다음 출력을 융합하여 최종 item embedding을 얻음
- 단점 : 두 종류의 그래프를 느슨히 연결하고 두 관계 사이의 item transitions을 무시

➢ item embedding 전에 fusion 먼저 함

- weighted sum 사용.

$$\mathbf{A}_{hyb} = \mathbf{I} + \alpha \mathbf{A}_{seq} + \beta \mathbf{A}_{sem}, \qquad (3)$$

- α와 β는 각각 sequential item graph와 semantic item graph 에 대한 weight coefficients
- 가중치는 item에서 sequential /semantic neighbors으로 transition되는 unnormalized probabilities로 간주

1) GCN

- hybrid item graph에서 graph convolutions 을 수행하여 smoothed item embedding을 얻음.

$$\mathbf{V}^{(k)} = \sigma(\hat{\mathbf{A}}\mathbf{V}^{(k-1)}\mathbf{W}^{(k)}), \quad (4)$$

- $V^{(k)} \in \mathbb{R}^{|I|}$: k번째 layer에서 item의 hidden representations
- $W^{(k)} \in \mathbb{R}^{d \times d}$: trainable parameter matrix
- $\hat{A}$는 symmetric normalization 이후 item graph의 adjacency matrix

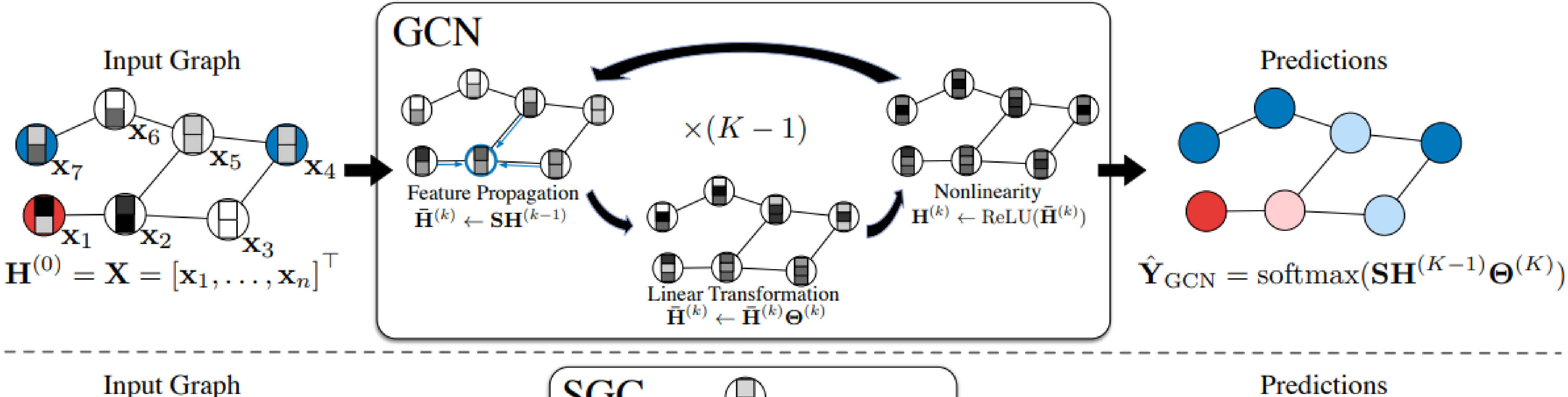
- GCN은 graph-based semi-supervised classification problems에 대해 우수한 성능을 얻지만 recommendation에서 embedding smoothing하기에는 적합하지 않을 수 있음 > multi-layer nonlinear transformation을 반복 수행시 ID embedding 학습에 영향을 미쳐 recommendation 성능이 저하.

2) SGCN

- SGCN의 영감으로 GCN 레이어에서 비선형 변환 제거, simple graph convolution 을 수행하여 sequential recommendation model에서 embedding smoothing.

$$\mathbf{V}^{(k)} = \hat{\mathbf{A}}\mathbf{V}^{(k-1)}, \quad (5)$$

- $V^{(0)}$는 item에 대한 훈련 가능한 ID embedding.
- item embedding에는 transformation을 적용하지 않음.
- graph convolution은 graph 구조에 따라 item 표현을 누그러뜨리는 weighted mean filter 역할.
- 마지막 레이어 $V^{(K)}$의 hidden representation matrix은 sequential모델의 입력으로 사용



3. Framework

- one-layer simple graph convolution
- adjacency matrix의 weights를 1 로 가정

$$\mathbf{v}_i^{(1)} = \sum_{m \in \mathcal{N}_i^+} \frac{1}{\sqrt{|\mathcal{N}_i^+|}\sqrt{|\mathcal{N}_m^+|}} \mathbf{v}_m^{(0)}, \qquad (6)$$

$$\mathcal{N}_i^+ = \mathcal{N}_i \cup \{I_i\}.$$

- 여기서 item i 와 item j의 similarity score는 dot-product로 계산

- 두 item간 similarity
- graph convolution 이후 두 item 간의 similarity는 주로 neighbor-to-neighbor similarity
- ➢ neighbor 구조가 유사할수록 representation vectors의 similarity가 더 높아질 수 있음

$$
\begin{aligned}
\mathbf{v}_i^{(1)^T}\mathbf{v}_j^{(1)} &= w_{ij}\sum_{m\in\mathcal{N}_i^+}\sum_{n\in\mathcal{N}_j^+}\frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_n^+|}}\mathbf{v}_m^{(0)^T}\mathbf{v}_n^{(0)} \\
&= w_{ij}\Big(\sum_{m\in\mathcal{N}_i}\sum_{n\in\mathcal{N}_j}\frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_n^+|}}\mathbf{v}_m^{(0)^T}\mathbf{v}_n^{(0)} \\
&\quad+\frac{1}{\sqrt{|\mathcal{N}_i^+|}\sqrt{|\mathcal{N}_j^+|}}\mathbf{v}_i^{(0)^T}\mathbf{v}_j^{(0)} \\
&\quad+\sum_{m\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_j^+|}}\mathbf{v}_m^{(0)^T}\mathbf{v}_j^{(0)} \\
&\quad+\sum_{n\in\mathcal{N}_j}\frac{1}{\sqrt{|\mathcal{N}_n^+|}\sqrt{|\mathcal{N}_i^+|}}\mathbf{v}_n^{(0)^T}\mathbf{v}_i^{(0)}\Big).
\end{aligned}
\quad (7)
$$

neighbor-to-neighbor similarity

node-to-node similarity

node-to-neighbor similarity

3. Framework

- graph convolutions 의 embedding Smoothing은 second-order proximity 기반 second-order proximity
- graph 컨볼루션 기반 embedding smoothing을 sequential recommendation에 적용하는 것은 과거 item $\{i_t, i_{t-1}, \cdots\}$ 과 sequential 또는 semantic으로 관련된 item $\{i_c: c \in \mathcal{N}_{i_t} \cup \mathcal{N}_{i_{t-1}}\}$로 다음 $item\ i_{\{t+1\}}$을 예측하는 것으로 간주 가능.

4. Discussion

1) Relationship with Graph Laplacian Regularization

<Graph Laplacian Regularization (GLR)>

- graph-based semi-supervised node classification

$$L = \underbrace{\boxed{\sum_i l(y_i, f(\mathbf{x}_i))}}_{\text{supervised loss}} + \underbrace{\boxed{\lambda \sum_{(i,j)\in\mathcal{E}} a_{ij}\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|^2,}}_{\text{graph Laplacian regularization}} \quad (8)$$

연결된 노드가 유사한 레이블 공유 가정

- λ : weight coefficient
- Semi-supervised embedding 은 regularization term을 prediction layer에서 embedding layer로 확장
- Laplacian eigenmaps의 영감으로 Graph Regularized Matrix Factorization (GRMF) 제안

$$\begin{aligned} L &= \sum_{(i,j)} l(y_{ij}, \mathbf{u}_i^T \mathbf{v}_j) + \lambda \sum_{(m,n)\in\mathcal{E}} a_{mn}\|\mathbf{v}_m - \mathbf{v}_n\|^2 \\ &= \sum_{(i,j)} l(y_{ij}, \mathbf{u}_i^T \mathbf{v}_j) + \lambda \mathrm{tr}(\mathbf{V}^T\mathbf{L}\mathbf{V}), \end{aligned} \quad (9)$$

- 이웃 item이 similar representations을 공유하는 경향이 있다고 가정 〉 그래프 embedding propagation 가능

1) Relationship with Graph Laplacian Regularization

1. graph Laplacian regularization는 first-order proximity, 즉 2개 item의 관계 유무를 기반으로 구축.
- 그러나 실제 데이터에서 관찰된 item 관계는 sparse 하며 many similar items간 명시적으로 연관되지 않을 수 있음[13].
- neighborhood structure의 similarity (즉, second-order proximity)을 고려하는 것은 item의 관계를 매우 풍부하게 할 수 있으며 embedding smoothing에 효율적.

2. graph Laplacian regularization는 semi-supervised learning framework를 기반으로 하며, 그 성능은 가중치 λ에 매우 민감할 수 있음.
- 너무 작거나 큰 정규화는 recommendation 성능을 저하시키거나 under-smoothing/over-smoothing.
- 그래프 컨볼루션은 normalized propagation matrix를 사용하며 성능은 node itself와 neighbors node 정보의 균형을 맞추는 가중치에 덜 민감.

2) Relationship with SR-GNN

< SR-GNN >

- Gated Graph Neural Network(GGNN)을 사용해 session-based recommendation에서 local dependency를 모델링
- session sequences는 global graph를 구성하기 위해 수집, 각 session sequence 는 global graph 에서 추출된 subgraph로 모델링
- A^{in}, A^{out} : session 의 subgraph에서 incoming and outgoing edges를 표현하는 adjacency matrices

$$\mathbf{a}_t^{in} = \mathbf{A}_t^{in}([\mathbf{v}_1, \cdots, \mathbf{v}_n]\mathbf{W}^{in} + \mathbf{b}^{in}), \tag{10}$$

$$\mathbf{a}_t^{out} = \mathbf{A}_t^{out}([\mathbf{v}_1, \cdots, \mathbf{v}_n]\mathbf{W}^{out} + \mathbf{b}^{out}), \tag{11}$$

$$\mathbf{a}_t = [\mathbf{a}_t^{in}; \mathbf{a}_t^{out}], \tag{12}$$

$$\mathbf{h}_t = \mathrm{GRU}(\mathbf{a}_t, \mathbf{v}_{t-1}), \tag{13}$$

2) Relationship with SR-GNN

<GES-SASRec와 SR-GNN의 차이점>

1. SR-GNN은 session 내의 item이 global graph에 따라 서로 communicate하는 방법만 고려.

- 세션에서 상호 작용하지 않는 관련 item이 제외.
- 반대로, GES-SASRec는 global graph에서 embedding smoothing.
- user's interaction sequence의 경우, 사용자가 상호 작용하지 않았지만 다른 사용자가 가진 관련 item을 고려.

2. GES-SASRec는 general framework.

- 사용자 item 상호 작용의 sequential item graph, item 속성의 semantic item graph 또는 hybrid item graph 와 같은 다양한 item graph 를 활용하여 recommendation 성능을 향상.
- 이 framework 는 Markov Chain, RNN, CNN, self-attention을 기반으로 한 모델과 같은 성능을 개선하기 위한 다양한 sequential모델에 적용될 수 있음.

3) Time Complexity Analysis

- Graph convolution time complexity. sparse matrices를 사용함으로써 original GCN은 $\mathcal{O}(|\mathcal{E}|dK + |I|d^2K)$
- $|\mathcal{E}|$: graph edges 수 / $|I|$: items 수 / d : embedding size / K : depth of GCN
- unnecessary nonlinear transformations 제거한, simple graph convolution은 $\mathcal{O}(|\mathcal{E}|dK)$
- layer aggregation strategies가 사용되지 않으면, propagation matrix $\hat{A}^K$ 사전 계산 가능 $\mathcal{O}(|\mathcal{E}|d)$

5. Experiments

1) Dataset and Experiment Settings

- Dataset 통계

TABLE 1: Statistics of the evaluation datasets.

Statistics	Amazon Books	Yelp	Google Local
#Users	27,738	32,206	16,381
#Items	43,790	27,671	34,928
#Interactions	624,573	726,154	427,910
Sparsity	0.051%	0.081%	0.075%
Avg. Length	20.52	20.55	24.12
#Relations	847,258	268,402	687,397

2) Experimental Settings

< Evaluation Protocols >

- Hit Ratio (HR)
- Normalized Discounted Cumulative Gain (NDCG)
- Mean Reciprocal Rank (MRR)
- g_u : 사용자 u에 대한 실제 item / r_u, g_u는 사용자 u 및 item g_u에 대한 추천 모델의 생성 순위 / 1(·) : indicator function

$$\mathrm{HR@}N = \frac{1}{|U|} \sum_{u \in U} \mathbf{1}(r_{u,g_u} \leq N), \tag{14}$$

$$\mathrm{NDCG@}N = \frac{1}{|U|} \sum_{u \in U} \frac{\mathbf{1}(r_{u,g_u} \leq N)}{\log_2(r_{u,g_u} + 1)}, \tag{15}$$

$$\mathrm{MRR@}N = \frac{1}{|U|} \sum_{u \in U} \frac{\mathbf{1}(r_{u,g_u} \leq N)}{r_{u,g_u}}, \tag{16}$$

3) Performance Comparison with SASRec (RQ1)

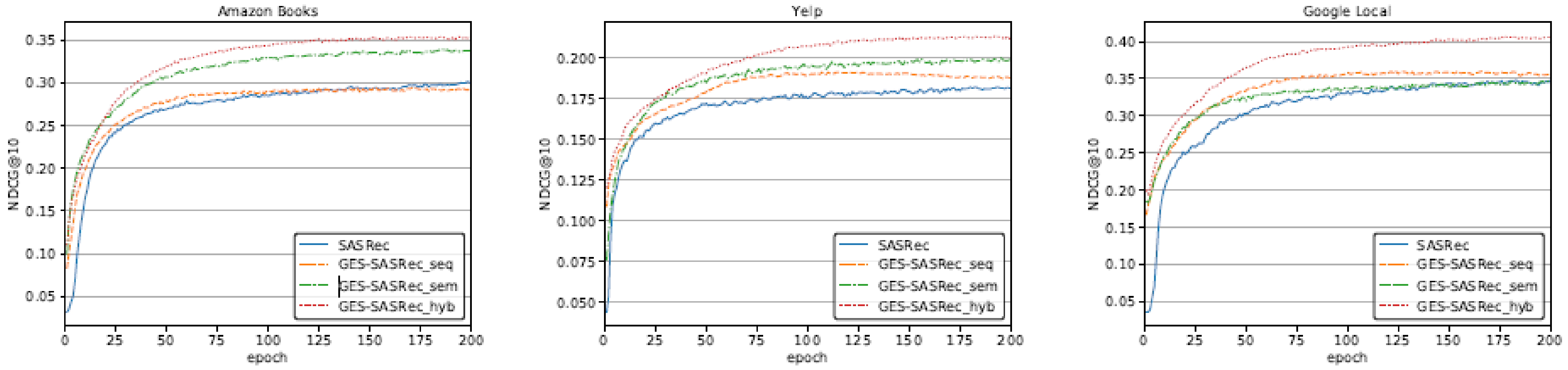
- 그래프 sequential/semantic/hybrid item graph item 그래프로 SASRec 및 GES-SASRec layer의 성능을 순차적으로 보고.

TABLE 2: Performance of GES-SASRec with different item graphs and different numbers of graph convolutional layers. Best results are in boldface.

#Layers	Item Graph	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
0 Layer	-	0.4838	0.3075	0.2535	0.3583	0.1946	0.1451	0.5680	0.3464	0.2788
1 Layer	sequential	0.4596	0.2948	0.2445	0.3422	0.1897	0.1436	0.5679	0.3626	0.2994
	semantic	0.5295	0.3364	0.2766	0.3620	0.1978	0.1478	0.5714	0.3433	0.2731
	hybrid	0.5314	0.3446	0.2877	0.3773	0.2105	0.1600	**0.6257**	**0.4031**	**0.3358**
2 Layers	sequential	0.4597	0.2923	0.2428	0.3531	0.1948	0.1468	0.5653	0.3578	0.2951
	semantic	0.5321	0.3345	0.2744	0.3208	0.1715	0.1266	0.5460	0.3220	0.2532
	hybrid	**0.5405**	**0.3553**	**0.2991**	**0.3825**	**0.2128**	**0.1620**	0.6194	0.3910	0.3206
3 Layers	sequential	0.4529	0.2832	0.2314	0.3441	0.1883	0.1411	0.5572	0.3438	0.2780
	semantic	0.5105	0.3189	0.2604	0.3041	0.1611	0.1178	0.5409	0.3176	0.2495
	hybrid	0.5359	0.3469	0.2886	0.3765	0.2061	0.1548	0.6016	0.3713	0.3001

3) Performance Comparison with SASRec (RQ1)

- proposed embedding smoothing method의 장점



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 3: Test performances of SASRec and GES-SASRec w.r.t. the training epochs.

4) Performance Comparison with State-of-the-Arts (RQ2)

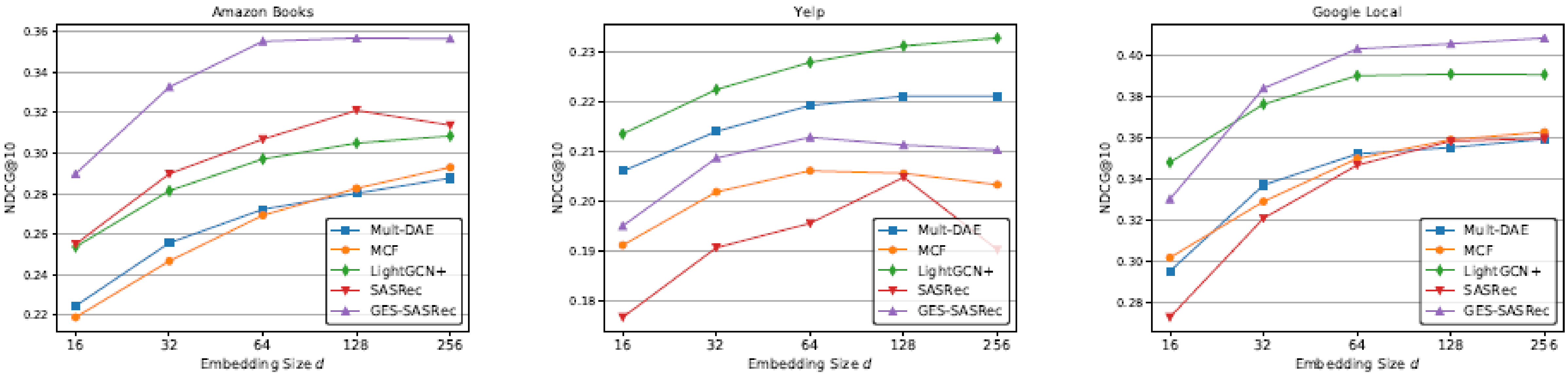
- performance comparison with baseline methods

TABLE 3: Performance comparison with embedding size 64. Best results are in boldface.

Method	Information	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
MP	-	0.0578	0.0270	0.0179	0.0868	0.0433	0.0303	0.0673	0.0326	0.0223
BPR	-	0.3877	0.2200	0.1690	0.3484	0.1905	0.1429	0.5224	0.3093	0.2438
Mult-DAE	-	0.4450	0.2722	0.2193	0.3907	0.2192	0.1670	0.5676	0.3521	0.2859
LightGCN	-	0.4461	0.2612	0.2049	0.3889	0.2167	0.1644	0.5614	0.3499	0.2860
FPMC	sequential	0.4412	0.2732	0.2218	0.3338	0.1819	0.1362	0.5654	0.3567	0.2932
TransRec	sequential	0.4332	0.2550	0.2006	0.3666	0.2027	0.1528	0.5534	0.3313	0.2647
GRU4Rec	sequential	0.4262	0.2508	0.1973	0.3225	0.1704	0.1250	0.5288	0.3131	0.2475
NARM	sequential	0.3945	0.2354	0.1873	0.3114	0.1626	0.1176	0.5175	0.3164	0.2554
Caser	sequential	0.4054	0.2499	0.2022	0.3065	0.1639	0.1225	0.5313	0.3347	0.2740
SASRec	sequential	0.4824	0.3069	0.2528	0.3588	0.1956	0.1461	0.5664	0.3466	0.2792
MCF	semantic	0.4510	0.2693	0.2145	0.3760	0.2061	0.1552	0.5722	0.3498	0.2811
CKE	semantic	0.4415	0.2530	0.1954	0.3726	0.2055	0.1548	0.5735	0.3464	0.2765
LightGCN+	semantic	0.4885	0.2971	0.2397	**0.4074**	**0.2279**	**0.1732**	0.6172	0.3901	0.3196
MoHR	hybrid	0.4946	0.2847	0.2212	0.3974	0.2215	0.1681	0.6032	0.3591	0.2845
GES-SASRec	hybrid	**0.5405**	**0.3553**	**0.2991**	0.3825	0.2128	0.1620	**0.6257**	**0.4031**	**0.3358**

5) Hyperparameter and Ablation Analyses (RQ3)

- performances of the compared methods w.r.t. the embedding size d.

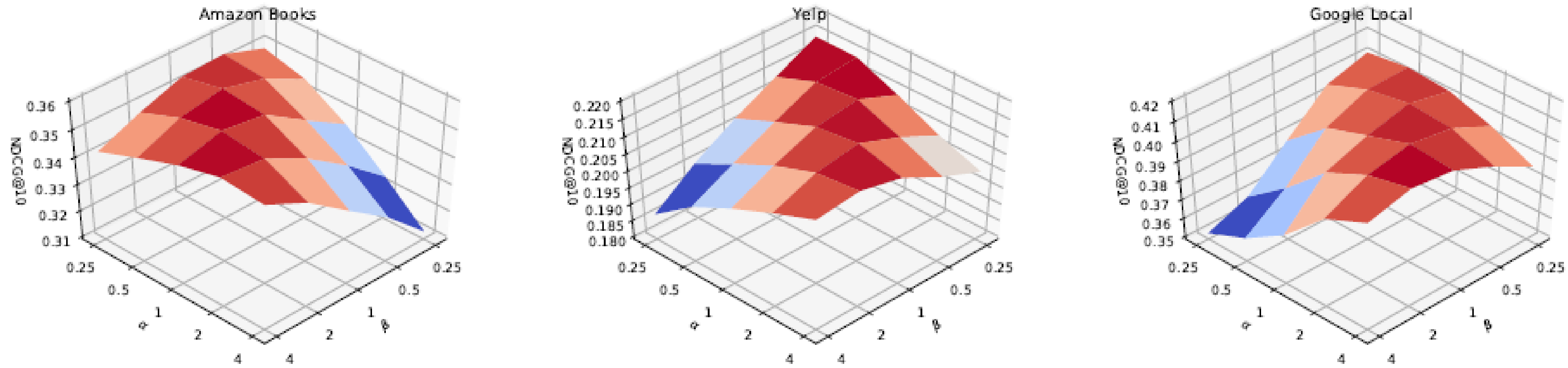


(a) Amazon Books (b) Yelp (c) Google Local

Fig. 4: Performances of the models w.r.t. the embedding size d.

5) Hyperparameter and Ablation Analyses (RQ3)

- results of GES-SASRec w.r.t. the sequential/ semantic relation coefficient α / β

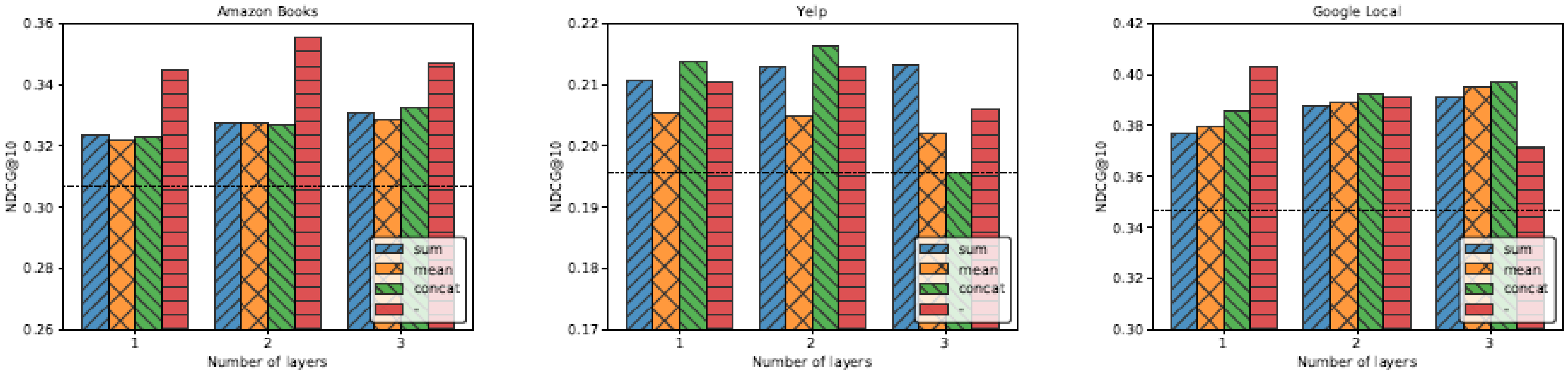


(a) Amazon Books (b) Yelp (c) Google Local

Fig. 5: Performance of GES-SASRec w.r.t. the item relation coefficients α and β.

5) Hyperparameter and Ablation Analyses (RQ3)

- the performances of GES-SASRec with different layer-aggregation functions



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 6: Performance of GES-SASRec w.r.t. the layer aggregation functions. Dotted line indicates the performance of SASRec.

5) Hyperparameter and Ablation Analyses (RQ3)

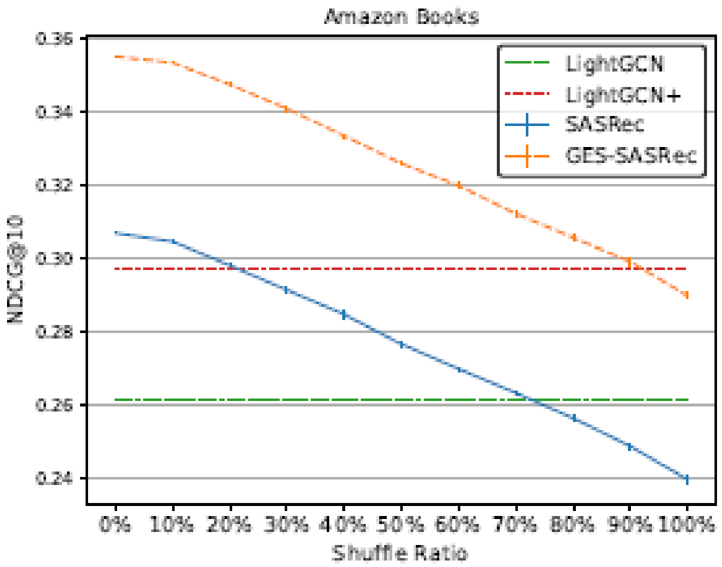
- the performances of SASRec with different embedding smoothing methods on the hybrid item graphs
  - Graph Laplacian Regularization [14] (GLRSASRec)
  - Graph Convolutional Network [6] (GCN-SASRec)
  - Simple Graph Convolution [11] (GES-SASRec)

TABLE 4: Performance of SASRec with different embedding smoothing methods. Best results are in boldface.

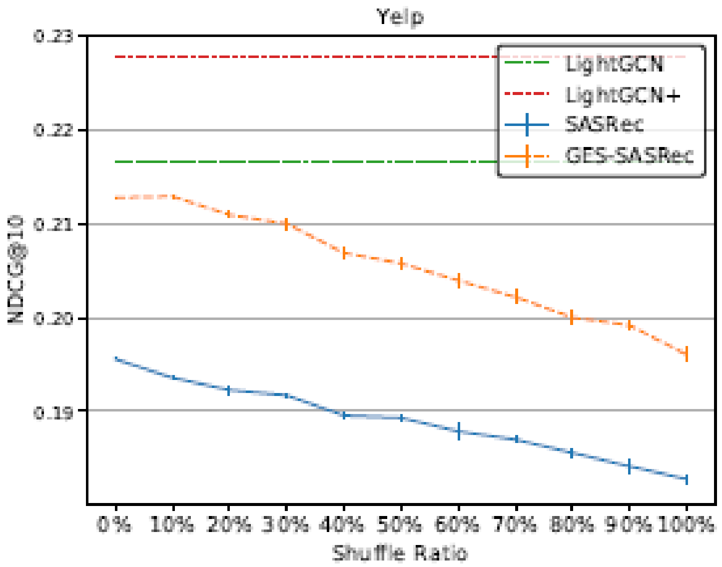
Method	Amazon Books			Yelp			Google Local		
	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
SASRec	0.4824	0.3069	0.2528	0.3588	0.1956	0.1461	0.5664	0.3466	0.2792
GLR-SASRec	0.4901	0.2982	0.2391	0.3708	0.2026	0.1519	0.6080	0.3760	0.3041
GCN-SASRec	0.5138	0.3296	0.2725	**0.3890**	**0.2169**	**0.1645**	0.6085	0.3901	0.3224
GES-SASRec	**0.5405**	**0.3553**	**0.2991**	0.3825	0.2128	0.1620	**0.6257**	**0.4031**	**0.3358**

6) Performance under shuffled User Interaction Or-ders (RQ4)

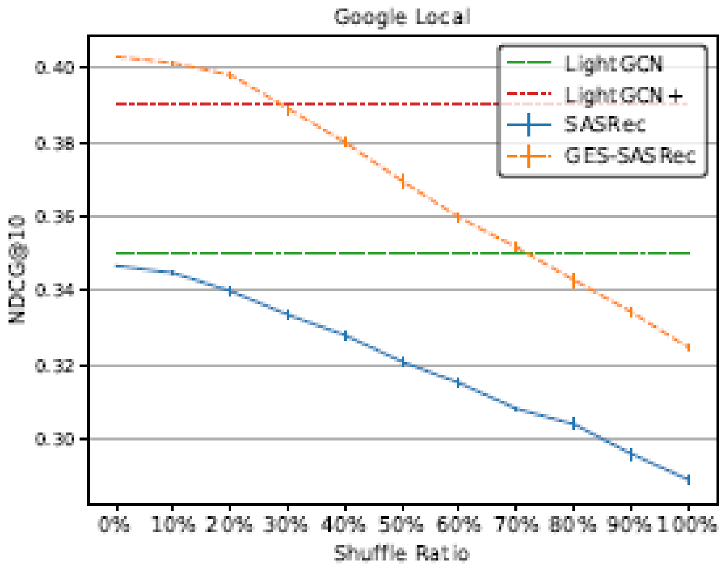
- the performances of SASRec, GES-SASRec, LightGCN, and LightGCN+ under different shuffle ratios in the test process.



(a) Amazon Books



(b) Yelp



(c) Google Local

Fig. 7: Performances of SASRec, GES-SASRec, LightGCN and LightGCN+ w.r.t. the shuffle ratio.

7) Embedding Smoothing for Other Models (RQ5)

- smooth the item embedding in these models with the proposed hybrid item graph and show the results

TABLE 5: Performances of sequential recommendation models with and without the proposed embedding smoothing method.

Category	Method	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
Markov Chain	TransRec	0.4332	0.2550	0.2006	0.3666	0.2027	0.1528	0.5534	0.3313	0.2647
	GES-TransRec	0.4670	0.2697	0.2093	0.3879	0.2139	0.1614	0.6012	0.3624	0.2892
	Improvement	7.81%	5.77%	4.31%	5.81%	5.53%	5.66%	8.64%	9.37%	9.24%
CNN	Caser	0.4054	0.2499	0.2022	0.3065	0.1639	0.1225	0.5313	0.3347	0.2740
	GES-Caser	0.5016	0.3138	0.2558	0.3706	0.2065	0.1565	0.6216	0.3967	0.3268
	Improvement	23.74%	25.56%	26.53%	20.92%	25.98%	27.72%	16.99%	18.54%	19.27%
RNN	GRU4Rec	0.4262	0.2508	0.1973	0.3225	0.1704	0.1250	0.5288	0.3131	0.2475
	GES-GRU4Rec	0.4893	0.2920	0.2315	0.3576	0.1939	0.1445	0.5933	0.3589	0.2873
	Improvement	14.78%	16.46%	17.31%	10.87%	13.80%	15.56%	12.20%	14.62%	16.09%
RNN+Attention	NARM	0.3945	0.2354	0.1873	0.3114	0.1626	0.1176	0.5175	0.3164	0.2554
	GES-NARM	0.5155	0.3182	0.2578	0.3835	0.2136	0.1622	0.6081	0.3800	0.3096
	Improvement	30.67%	35.21%	37.68%	23.18%	31.41%	37.86%	17.52%	20.08%	21.22%

6. Conclusion

Q & A

감사합니다