LESSR

이은경

목차

1. Abstract

1. Abstract

1. Lossy session encoding problem

- Message passing 동안 purmutation-invariant aggregation과 lossy encoding 때문에 무시되는 item transitions에 대한 sequential information 문제

➢ Lossless encoding scheme

➢ GRU 기반 Edge-order preserving aggragation layer : losslessly encoded graphs

2. Ineffective long-range dependency capturing problem

- 제한된 수의 layer 때문에 long-range dependencies capture 가 잘 안됨.

➢ Shortcut connections

3. Two kinds of layer를 combine함으로써 해당 문제(2가지)를 해결

2. Introduction

2. Introduction

1) 사전연구

- 고정된 수의 이전 action을 사용하여 다음 action을 예측하는 일반적인 시스템과 달리 session based recommendation system은 사용자 action을 분리된 session으로 그룹화하고 활성 session의 이전 action만 사용하여 recommendation 제공
- Session recommendation : next-item recommendation 의 special case
- intra-session dependency가 inter-session dependency보다 next item에 더 큰 영향을 미친다는 관찰에서 비롯[3]

- 기존 general next item recommendation 문제 > combining uncorrelated session과 extracting incomplete sesion의 문제 > session은 없으므로 accurate recommendation 가능 & online service 제공

Session + GNN : 두 가지 문제 발생

1) Lossy session encoding problem
2) Information loss problem

1) Lossy session encoding

2. Introduction

- GNN 사용을 위해 각 Session을 directed graph 로 변환해야 함. (node : unique items, edge : transitions(weighted / unweighted))
- One-to-one mappin으로 인한 Lossy operation 발생. 다른 session도 같은 graph 로 표현됨. Fig1.

> Session을 directed graph로 변환하는 Lossless encoding scheme & GRU를 사용한 propagated information을 aggregate하는 EOPA layer 제안

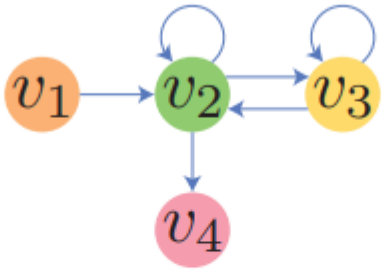


Figure 1: Two different sessions $[v_1, v_2, v_3, v_3, v_2, v_2, v_4]$ and $[v_1, v_2, v_2, v_3, v_3, v_2, v_4]$ are converted to the same graph.

2) Ineffective long-range dependency capturing

- GNN은 모든 long-range capture 불가
- GNN의 각 layer는 1개의 layer에 1 hop relation만 capture. overfitting, over-smoothing 문제로 3 layer 이상은 잘 사용하지 않음.
- real-world에서는 3 layer 보다 큰 relation 을 갖는것이 쉬움. 3보다 긴 중요한 sequential patterns이 있을 가능성이 매우 높음. 그러나 GNN 은 이러한 정보 캡쳐 불가.

➢ Attention mechanism을 사용하여 shortcut connections 을 따라 SGAT layer 적용 제안

➢ 두 가지 solutions을 결합하여 만든 GNN model LESSR

3. Preliminary

1) GNN

- Session based recommantaion formulation
- $G(V, E)$: given graph, V : sets of nodes / E : sets of edges
- 각 node $i \in V$: initial node representation으로 GNN의 첫 번째 layer에 전달되는 node feature vector x_i와 연관
- GNN의 각 layer에서 edge를 따라 message passing함으로써 node representation update

$$\boldsymbol{x}_i^{(l+1)} = f_{upd}^{(l)}\left(\boldsymbol{x}_i^{(l)}, \mathrm{agg}_i^{(l)}\right) \quad (1)$$

$$\mathrm{agg}_i^{(l)} = f_{agg}^{(l)}\left(\left\{f_{msg}^{(l)}\left(\boldsymbol{x}_i^{(l)}, \boldsymbol{x}_j^{(l)}\right) : (j, i) \in E_{in}(i)\right\}\right) \quad (2)$$

- $x_i^{(l)}$: layer l 의 node I 의 representation
- $E_{in}(i)$: node I 의 incoming edges set
- $f_{msg}^{(l)}$: Neighboring node에서 target node로 propagated message function.
- $f_{agg}^{(l)}$: target node에 전달된 정보를 aggregates / $f_{upd}^{(l)}$: original node representatio과 aggregated information을 새로운 node representation으로 update function

1) GNN

- L : GNN의 layer 개수
- L layer 에서 L step의 message passing 후에 final node representations은 L-hop communit내의 graph structure 와 the features of nodes 를 캡쳐
- graph classification task에서, readout function f_{out}은 최종 계층의 모든 노드의 표현을 집계하여 그래프 수준 표현 h_G를 생성하는 데 사용.

$$\boldsymbol{h}_G = f_{out}(\{\boldsymbol{x}_i^{(L)} : i \in V\}) \qquad (3)$$

4. Methology

1) Problem definition

- 목적 : active session 에서 이미 click된 item을 바탕으로 next click item 추천하는 것.
- Item의 universal set $I = \{v_1, v_2, \cdots, v_{|I|}\}$
- Session $s_i = [s_{i,1}, s_{i,2}, \cdots, s_{i,l_i}]$은 시간에 따라 item seqence를 정렬한 것. $s_{i,t} \in I$ 는 time step t에서의 item. l_i 는 s_i 의 길이.
- 모델의 objective 는 next item s_{i,l_i+1} 을 예측하는 것.
- Session-based recom sys는 next item 의 probability distribution 을 generate $p(s_{i,l_i+1})$
- Top-K probabilities items는 recommendations의 candidate set 안에 있음

1) Problem definition

- [11, 16, 17, 21, 23, 25]에 이어 본 논문에서 user ID 및 item attributes 와 같은 추가 context information는 고려하지 않음.
- item ID는 d차원 공간에 embedding되어 모델에서 initial item feature로 사용.
- 이는 session based recommandation paper의 일반적인 관행.

- 그러나 추가 context information 를 고려하도록 LESSR 를 조정하는 것은 쉬움. 예) user ID embedding 은 graph level attribute로 제공될 수 있으며 각 layer의 item ID embedding 에 추가할 수 있음[24].
- Item feature는 item ID embedding [9]과 결합하거나 대체 가능

2)Converting sessions to graphs

1. S2MG
2. S2SG

- Session을 graph로 변환하는게 가장 우선
- S2MG : session to EOP multigraphs
- S2SG : session to shortcut graphs

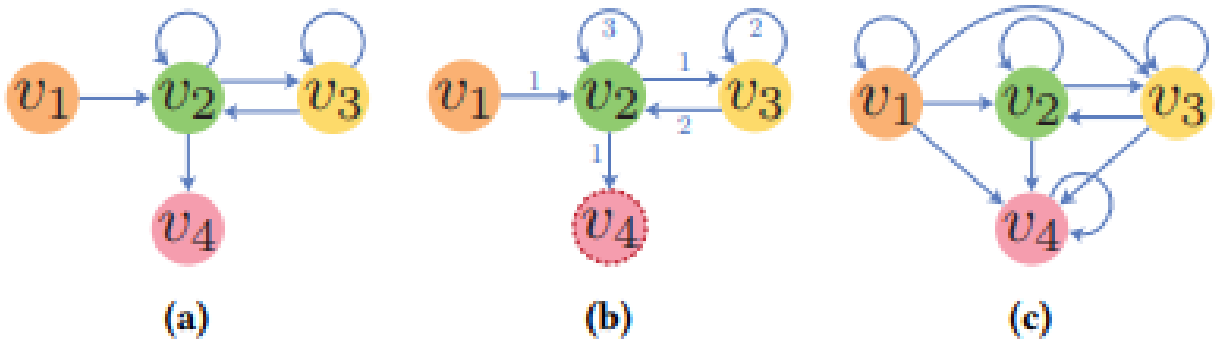


Figure 3: The weighted graph (a), EOP multigraph (b) and shortcut graph (c) of session $[v_1, v_2, v_3, v_3, v_2, v_2, v_4]$ converted by S2WG, S2MG and S2SG, respectively. Note that the weights in (a) are omitted because they are all 1.

2)Converting sessions to graphs

1. **S2MG**
2. S2SG

1) S2G : Unweighted graph $G = (V, E)$, V : session 에서의 unique items node set. E : edge set > if edge $u = s_{i,t}, v = s_{i,t+1}$ (u,v) for $1 \leq t < l_i$

2) S2WG : weighted graph edge (u,v)에 weight를 줌. $u \rightarrow v$ 의 transition 횟수에 따라

- S2G, S2WG 는 lossy conversion methods(변환된 그래프가 주어지면 원래 session을 reconstruct 할 수 없기 때문)
- Session $s_1 = [v_1, v_2, v_3, v_3, v_2, v_2, v_4]$와 $s_2 = [v_1, v_2, v_2, v_3, v_3, v_2, v_4]$ 가 같이 변환됨.

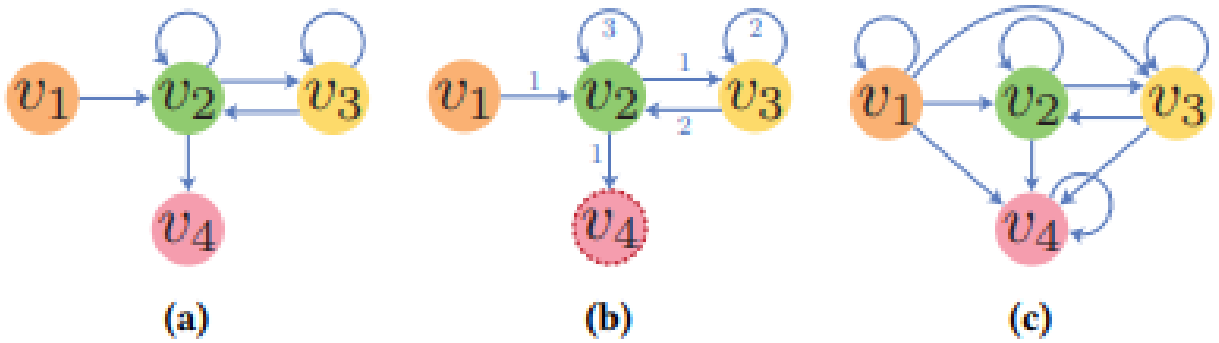


Figure 3: The weighted graph (a), EOP multigraph (b) and shortcut graph (c) of session $[v_1, v_2, v_3, v_3, v_2, v_2, v_4]$ converted by S2WG, S2MG and S2SG, respectively. Note that the weights in (a) are omitted because they are all 1.

2)Converting sessions to graphs

1. **S2MG**
2. S2SG

- lossy conversion 에서 무시된 정보가 next item 을 결정하는 데 중요할 수 있기 때문에 문제가 될 수 있음.
- 모델이 lossy conversion method를 사용하여 “blindly(맹목적으로)" decision을 내리는 대신 무시할 수 있는 정보를 결정하는 방법을 자동으로 배우도록 해야함
- 그렇지 않으면 모델의 modeling capacity가 lossy conversion method 에 의해 제한되기 때문에 복잡한 데이터 세트에 적합할 만큼 유연하지 않음

➢ S2MG(session to EOP multigraph) 제안 : session 을 edge order를 유지하는 directed multi-graph 로 변환

- Original session 에서 각 transition $u \rightarrow v$, edge (u, v) 로 만듬. U에서 v로 transition 이 여러 번 있을 경우 u에서 v로 edge를 여러 개 만들기 때문에 그래프는 multi-graph.
- 각 node v, edge $E_{in}(v)$ 는 발생 시간에 따라 정렬 가능. $E_{in}(v)$: integer attribute

2)Converting sessions to graphs

1. **S2MG**
2. S2SG

- EOP multigraph(즉, S2MG를 사용하여 변환된 그래프)가 주어진 원래 세션을 재구성하는 방법을 보여줌으로써 S2MG가 세션에서 그래프로의 무손실 변환 방법임을 증명.
- 세션에서 발생한 item의 역순으로 복구.
- 예) MG_{s_1} last item 은 라벨이 붙은 노드 v_4 , last item 을 알고, v_4 incoming edge의 순서를 알고, 마지막 edge를 (v_2, v_4)로 결정할 수 있음.
- 두 번째 last item은 마지막 edge의 source node, 즉 v_2 . 이를 반복하여 세 번째 last item 을 결정 가능.
- 이러한 방식으로 그래프 MG_{s_1} 에서 세션 s_1 을 복구할 수 있음.
- 주어진 그래프에서 동일한 절차를 적용하여 원래 세션을 재구성 가능.

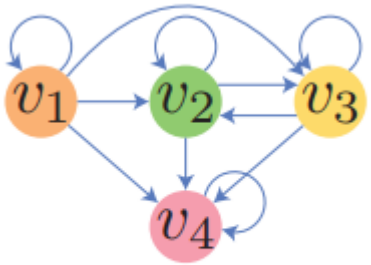
➢ S2MG는 세션에서 그래프로의 무손실(lossless conversion method) 변환 방법.

2)Converting sessions to graphs

1. S2MG
2. **S2SG**

- GNN-based models 에서 발생하는 ineffective long-range dependency problem
- shortcut graph attention (SGAT) layer > Sec. 4.3.2
- SGAT layer input : EOP multigraph와 다른 입력 그래프 필요 > S2SG (session to shortcut graph)
- Session s_i 가 주어졌을 때, node : s_i 의 unique item인 그래프를 만듬. 순서가 정해진 노드 (u,v) pair에 대해, 다음과 같은 pair (s_{i,t_1}, s_{i,t_2})가 존재하는 경우에만 u 에서 v까지 edge를 만듬.
- 그래프는 inter-mediate item을 거치지 않고 item을 연결하므로 short-cut graph라고 함.

- SGAT layer이 message passing 을 수행할 때 update function 과 aggregate function 를 결합할 수 있도록 그래프에 self-loops 를 추가[16, 22].
- $s_1 = [v_1, v_2, v_3, v_3, v_2, v_2, v_4]$의 short-cut graph 는 그림 3(c)에 표시된 그래프



(c)

3) Our GNN Model : LESSR

- losslessly 변환되는 EOP multigraphs 를 고려할 때, GNN은 different session이 different representations에 mapping 될 수 있도록 그래프를 처리해야 함.
- 기존 GNN 모델에서는 edge의 relative order를 무시하는 permutation-invariant aggregation functions를 사용하기 때문에 불가능. (edge 간 order info가 중요하지 않은 dataset에만 적합)

➢ 이 문제를 해결하기 위해, GRU를 사용하여 인접 node로부터 전달된 정보를 집계하는 edge-order preserving aggregation (EOPA) layer 제안.

- $OE_{in}(i) = [(j_1, i), (j_2, i), \cdots, (j_{d_i}, i)]$ 를 순서 $E_{in}(i)$로 나열. $\mathrm{d_i}$ 는 node i의 각도. EOP 에서 edge의 integer attributes를 사용하여 $E_{in}(i)$ 에서 $OE_{in}(i)$ 를 얻을 수 있음. 이웃으로부터 집계된 정보는 다음과 같이 정의

$$\mathrm{agg}_i^{(l)} = \boldsymbol{h}_{d_i}^{(l)} \tag{4}$$

$$\boldsymbol{h}_k^{(l)} = \mathrm{GRU}^{(l)}\left(f_{msg}^{(l)}\left(\boldsymbol{x}_i^{(l)}, \boldsymbol{x}_{j_k}^{(l)}\right), \boldsymbol{h}_{k-1}^{(l)}\right) \tag{5}$$

1. **EOPA layer**
2. SGAT layer
3. Stacking
4. Generating
5. Prediction and Training

3) Our GNN Model : LESSR

$$\mathrm{agg}_i^{(l)} = \boldsymbol{h}_{d_i}^{(l)} \tag{4}$$

$$\boldsymbol{h}_k^{(l)} = \mathrm{GRU}^{(l)}\left(f_{msg}^{(l)}\left(\boldsymbol{x}_i^{(l)}, \boldsymbol{x}_{j_k}^{(l)}\right), \boldsymbol{h}_{k-1}^{(l)}\right) \tag{5}$$

- $\{ h_k^{(l)} : 0 \leq k \leq d_i \}$ GRU의 hidden states, $f_{msg}^{(l)}$ 는 노드 j_k 에서 노드 i로 전달되는 메시지를 계산하는 valid message function 이 될 수 있음.
- Initial state $h_0^{(l)}$ 는 zero vector 로 설정
- GRU aggregator 는 RNN aggregator 의 일종. Session based recom에서 GRU가 LSTM을 능가하는 것으로 나타났기 때문에 LSTM 대신 GRU를 선택[8, 11].
- 주요 차이 : RNN aggregator는 edge의 random permutation 에서 aggregation 을 수행하여 incoming edge의 relative order를 의도적으로 무시하기 위해 사용되는 반면, GRU aggregator 는 fixed order 로 aggregation 를 수행 > 주된 기여는 아님

3) Our GNN Model : LESSR

1. **EOPA layer**
2. SGAT layer
3. Stacking
4. Generating
5. Prediction and Training

- 기존 GNN 모델의 동일한 message 와 update functions 을 GRU aggregator 와 함께 사용할 수 있지만, GRU의 강력한 표현 능력을 고려하여 message 와 update functions 모두에 linear transformation 을 사용.

$$\boldsymbol{x}_i^{(l+1)} = \boldsymbol{W}_{upd}^{(l)} \left(\boldsymbol{x}_i^{(l)} \| \boldsymbol{h}_{d_i}^{(l)} \right) \tag{6}$$

$$\boldsymbol{h}_k^{(l)} = \mathrm{GRU}^{(l)} \left(\boldsymbol{W}_{msg}^{(l)} \boldsymbol{x}_{j_k}^{(l)}, \boldsymbol{h}_{k-1}^{(l)} \right) \tag{7}$$

- $W_{upd}^{(l)} \in \mathbb{R}^{d \times 2d}, \mathrm{W}_{\mathrm{msg}}^{(l)} \in \mathbb{R}^{d \times d}$ learnable parameters, ‖ : concat

1. EOPA layer
2. **SGAT layer**
3. Stacking
4. Generating
5. Prediction and Training

3) Our GNN Model : LESSR

- 일반적으로 각 layer는 one step을 위한 정보를 propagation 하므로 한 layer는 노드 간의 1-hop relationship 만 캡처할 수 있음.
- Multi-hop relationship을 캡처하기 위해 여러 개의 GNN 레이어를 쌓을 수 있으나 over-smooth problem [12, 26] 초래. > 노드 표현이 동일한 값으로 수렴.
- 보통 layer 수가 3개보다 클 때 발생하기 때문에 여러 layer를 쌓는 것은 Multi-hop relationship 를 포착하는 좋은 방법이 아님.
- layer를 여러 개 쌓을 수 있더라도, GNN은 k가 layer 수와 동일한 k-hop relationship 까지만 capture 가능.
- 그러나 실제 애플리케이션에서는 세션의 길이가 k보다 큰 것이 일반적 > session-based recommendation 을 위한 기존 GNN 모델은 매우 긴 범위에서 dependencies을 효과적으로 포착할 수 없음.
- 문제 해결을 위해 S2SG에서 얻은 shortcut graph의 edge를 본질적으로 빠른 information propagation를 위해 shortcut graph attention (SGAT) layer를 제안.

3) Our GNN Model : LESSR

1. EOPA layer
2. **SGAT layer**
3. Stacking
4. Generating
5. Prediction and Training

$$x_i^{(l+1)} = \sum_{(j,i)\in E_{in}(i)} \alpha_{ij}^{(l)} W_{val}^{(l)} x_j^{(l)} \quad (8)$$

$$\alpha_i^{(l)} = \mathrm{softmax}(e_i^{(l)}) \quad (9)$$

$$e_{ij}^{(l)} = \left(p^{(l)}\right)^T \sigma\left(W_{key}^{(l)} x_i^{(l)} + W_{qry}^{(l)} x_j^{(l)} + b^{(l)}\right) \quad (10)$$

- $E_{in}(i)$: shortcut graph의 node I 에 대한 incoming edges의 set
- $P^{(l)}, b^{(l)} \in \mathbb{R}^{d}$, $W_{key}^{(l)}$, $W_{qry}^{(l)}$, $W_{val}^{(l)} \in \mathbb{R}^{d\times d}$: learnable parameters
- shortcut graph 의 edge는 intermediate items 을 거치지 않고 각 items을 이후의 모든 items 과 직접 연결. shortcut graph 의 edge는 items 간의 “shortcut connections”로 볼 수 있음.
- SGAT 계층은 intermediate items 을 거치지 않고 items 간의 shortcut 연결을 따라 정보를 전달하기 때문에 모든 길이의 장기 의존성을 효과적으로 포착.
- 기존 GNN 기반 방법의 원래 layer와 결합하여 long-range dependencies를 캡처하는 능력을 높일 수 있음.

3) Our GNN Model : LESSR

1. EOPA layer
2. SGAT layer
3. **Stacking**
4. Generating
5. Prediction and Training

- information loss problems 가 없는 GNN 모델을 구축하기 위해 여러 EOPA 및 SGAT layer를 쌓음.
- EOPA layer와 SGAT layer 을 interleave 하는 이유

1) shortcut graph는 original session 의 lossy conversion 이므로 여러 SGAT layer를 쌓으면 lossy session encoding problem 발생. information loss accumulates의 양이 누적되므로 layer가 많을 수록 문제가 심각. EOPA layer 과 SGAT layer 을 상호 분리함으로써 손실된 정보는 후속 EOPA layer에서 유지될 수 있고 SGAT layer 은 장거리 의존성을 포착하는 데 집중.
2) 각 layer 가 다른 종류의 layer 에 의해 포착된 features를 효과적으로 활용가능. EOPA layer 은 local context information를 캡처할 수 있고 SGAT layer 은 global dependencies를 capture할 수 있기 때문에, 두 종류의 layer를 interleaving 하면 장점을 효과적 결합 & 복잡한 의존성을 학습하는 능력 향상

- feature reuse 을 용이하게 하기 위해 [10]에서 제안된 dense connections 소개. 각 계층에 대한 입력은 모든 이전 계층의 출력 기능으로 구성. 구체적으로 말하자면, 원래, layer l에 대한 입력은 $\{x_i^{l-1} : i \in V\}$ 계층 l에 대한 입력은 이전의 모든 계층의 출력을 연결한 $\{x_i^{(0)} \| x_i^{(1)} \| \cdots \| x_i^{(l-1)} : i \in V\}$로 변경. 적은 paramer 로 동일성능 달성

3) Our GNN Model : LESSR

1. EOPA layer
2. SGAT layer
3. Stacking
4. **Generating**
5. Prediction and Training

- All layer의 message passing이 완료되면, 모든 node의 final representations을 얻음. 현재 session을 embedding vector로 표현하기 위해 [23]에서 제안한 readout function을 적용
- attention mechanism을 사용하여 node representations 을 집계하여 graph-level representation 을 계산.
- $x_{\text{last}}^{(\text{L})}$: session의 마지막 item의 최종 node representations 을 나타냄.
- graph-level representation h_G는 다음과 같이 정의.

$$\boldsymbol{h}_G = \sum_{i \in V} \beta_i \boldsymbol{x}_i^{(L)} \tag{11}$$

$$\boldsymbol{\beta} = \text{softmax}(\boldsymbol{\epsilon}) \tag{12}$$

$$\epsilon_i = \boldsymbol{q}^T \sigma(\boldsymbol{W}_1 \boldsymbol{x}_i^{(L)} + \boldsymbol{W}_2 \boldsymbol{x}_{last}^{(L)} + \boldsymbol{r}) \tag{13}$$

3) Our GNN Model : LESSR

1. EOPA layer
2. SGAT layer
3. Stacking
4. **Generating**
5. Prediction and Training

$$\boldsymbol{h}_G = \sum_{i \in V} \beta_i \boldsymbol{x}_i^{(L)} \tag{11}$$

$$\boldsymbol{\beta} = \mathrm{softmax}(\boldsymbol{\epsilon}) \tag{12}$$

$$\epsilon_i = \boldsymbol{q}^T \sigma(\boldsymbol{W}_1 \boldsymbol{x}_i^{(L)} + \boldsymbol{W}_2 \boldsymbol{x}_{last}^{(L)} + \boldsymbol{r}) \tag{13}$$

- $q, r \in \mathbb{R}^{\mathrm{d}}, W_1, W_2 \in \mathbb{R}^{\mathrm{d} \times d}$:learnable parameters.
- graph-level representation은$s_g = h_G$ 로 표시된 current session의 global preferences를 캡처.
- 이전 연구[11, 17, 23]에서 사용자의 최근 관심사를 명시적으로 고려하는 것도 중요. $s_l = x_{last}^{(L)}$ 로 local preference vector 를 정의. session embedding을 global 과 local session preferences의 선형 변환으로 계산.
- $W_h \in \mathbb{R}^{\mathrm{d} \times 2d}$: learnable parameter.

$$\boldsymbol{s}_h = \boldsymbol{W}_h(\boldsymbol{s}_g \| \boldsymbol{s}_l) \tag{14}$$

3) Our GNN Model : LESSR

- session embedding을 얻은 후, 다음 item의 probability distribution를 계산하여 recommendations 에 사용가능.
- 각 item i∈ I 에 대해, embedding v_i 와 session embedding 을 사용하여 점수를 계산.

$$z_i = \boldsymbol{s}_h^T \boldsymbol{v}_i \qquad (15)$$

- 다음 item i 의 예측 확률 $\hat{y}_i$ 는 다음과 같이 계산.

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{j \in I} \exp(z_j)} \qquad (16)$$

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\boldsymbol{y}^T \log \hat{\boldsymbol{y}}$$

- Top-K recommendation에서, top K probabilities를 가진 items를 recommend
- y : 원핫 벡터인 다음 item의 실제 확률 분포. loss function는 cross-entropy. 모든 매개 변수와 item embedding이 무작위로 초기화, end-to-end propagation training paradigm 에서 공동으로 학습.

5. Experiments

1) Datasets

Table 1: Statistics of datasets used in the experiments

Statistic	Diginetica	Gowalla	Last.fm
No. of Clicks	981,620	1,122,788	3,835,706
No. of Sessions	777,029	830,893	3,510,163
No. of Items	42,596	29,510	38,615
Average length	4.80	3.85	11.78

2) Baselines and Evaluation Metrics

- Item-KNN
- FPMC
- NARM
- NextItNet
- SR-GNN
- FGNN
- GC-SAN

> hyper-parameters ranges {16, 32, 64, 96, 128} for embedding dim d

- $\{10^{-3}, \cdots, 10^{-1}\}$ $leaning\ rate\ \eta$
- GNN layer 개수 {1,2,3,4,5}
- Adam optimizer
- Batch size 512
- HR@20, MRR@20

3) Performance Comparisons

Table 2: Experimental results (%) on three datasets

Method	Diginetica		Gowalla		Last.fm	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
Item-KNN	39.51	11.22	38.60	16.66	14.90	4.04
FPMC	28.50	7.67	29.91	11.45	12.86	3.78
NextItNet	45.41	15.19	45.15	21.26	20.12	7.08
NARM	49.80	16.57	50.07	23.92	21.83	7.59
FGNN	50.03	17.01	50.06	24.12	22.20	8.02
SR-GNN	50.81	17.31	50.32	24.25	22.33	8.23
GC-SAN	50.90	17.63	50.68	24.67	22.64	8.42
LESSR	**51.71**	**18.15**	**51.34**	**25.49**	**23.37**	**9.01**
Improv.	1.59%	2.95%	1.30%	3.32%	3.22%	7.01%

4) Ablation Studies

- Effectiveness of EOPA Layer

Table 3: The performance of different message passing layers

Layer(s)	Diginetica		Gowalla		Last.fm	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
WGAT	49.71	16.46	50.03	24.02	21.89	7.89
GGNN	49.85	16.59	50.24	24.23	22.08	8.02
EOPA (rand)	49.81	16.56	50.18	24.11	22.05	8.06
EOPA	**50.30**	**16.93**	**50.86**	**24.89**	**22.31**	**8.36**
GGNN+SAN	50.06	16.72	50.37	24.44	22.22	8.18
GGNN+EOPA	**50.28**	**16.91**	**50.76**	**24.96**	**22.41**	**8.40**

4) Ablation Studies

- Effectiveness of SGAT Layer.

Table 4: Performance differences in terms of HR@20

Model	LESSR	FGNN	SR-GNN	GC-SAN
Original	50.29	49.51	50.02	50.01
Modified	50.56	50.08	50.18	50.36
Improv.	0.54%	1.15%	0.32%	0.70%

Table 5: The performance of different orders of layers

Model	Diginetica		Gowalla		Last.fm	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
SSEE	50.17	16.75	50.77	24.70	21.98	8.19
EESS	50.34	16.83	50.71	24.83	22.11	8.25
SESE	50.36	16.81	50.74	24.75	22.07	8.20
ESES	**50.63**	**17.08**	**50.82**	**25.04**	**22.48**	**8.41**

5) Hyper-parameter Study

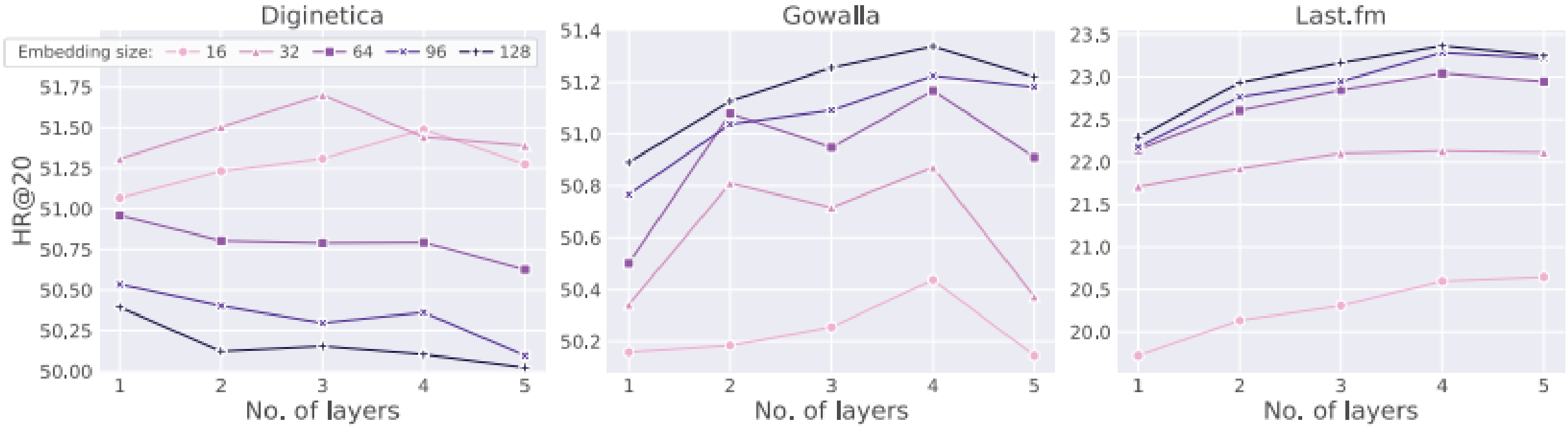


Figure 4: The performance of LESSR

6. Conclusion

6. Conclusion

- session-based recommendation에 대한 기존 GNN 모델에서 두 가지 정보 손실 문제, lossy session encoding 과 ineffective long-range dependency capturing problems 를 식별.
- 두 가지 문제를 해결하기 위해 세션을 그래프로 변환하는 두 가지 변환 방법인 S2MG와 SGAT layer을 사용하는 EOPA와 SGAT layer을 제안.
- 두 종류의 계층을 결합하여 두 가지 정보 손실 문제가 없는 LESSR이라는 모델을 구축하며 실험 결과는 LESR이 세 가지 공개 데이터 세트에서 SOTA을 능가.
- Future work : personalized and streaming session-based recommendation에 LESSR을 적용하는 데 관심.

Q & A
감사합니다