Session-Based Recommendation with Graph Neural Networks

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, Tieniu Tan

2019

임동주

목차

Introduction

- 과거 추천 알고리즘의 한계 (e.g. MC, RNN, NARM, STAMP)
  - 한 세션에서 유저의 충분한 액션이 없는 경우, 다음 액션의 추정이 어려움
    ➔ 대부분의 세션들은 익명으로 발생하기에, 특정 세션에 대한 유저의 표현을 추정하기는 어려움
  - 이전의 세션 기반 모델들은 연속된 아이템들의 단방향 트랜지션 만을 고려했으며, 세션 내의 다른 아이템과 같은 다양하고 복잡한 트랜지션을 고려하지 못함
    ➔ 트랜지션의 패턴을 로컬 팩터로서 사용하기에 중요

- Proposal
  - Session-Based Recommendation Graph Neural Networks (SR-GNN)
    - 아이템 간의 다양한 트랜지션을 탐색
    - 아이템들의 정확한 잠재 벡터(Latent Vector)를 만드는 알고리즘

  → 과거 세션 시퀀스를 기반으로 그래프 구성

  → 세션 그래프 기반으로 아이템들의 트랜지션을 포착해 임베딩 벡터 생성

  → 세션 표현을 더 신뢰할 수 있음

Related Work

- Conventional Recommendation Methods

- Matrix Factorization
  → 유저의 선호도가 positive clicks에 의해서만 제공되기에 세션 기반에는 적합하지 않음
- Item-Based Neighborhood methods
  → 아이템의 순차적인 순서를 고려하기 어렵고, 세션 내 마지막 클릭에 의해서만 예측이 가능
- Markov-chain & FPMC
  → 이전 구성요소들을 독립적으로 결합하기 때문에 예측 정확도가 제한됨

Related Work

- DL-based Methods

- RNN-Based
  - NARM : Attention mechanism on RNN
    → 세션 내 순차적 동작과 유저의 주요 목적, 유저 특성 모델링
  - STAMP : MLP + Attention
    → 유저의 일반적인 관심사와 현재 관심사를 효과적으로 포착
- Neural network on graphs
  - DeepWalk : 랜덤 순회 과정을 기반으로 그래프 노드의 표현을 학습하는 알고리즘
  - LINE/node2vec : unsupervised network embedding algorithms
  - CNN/RNN
  - GNN

Method

- Workflow of SR-GNN

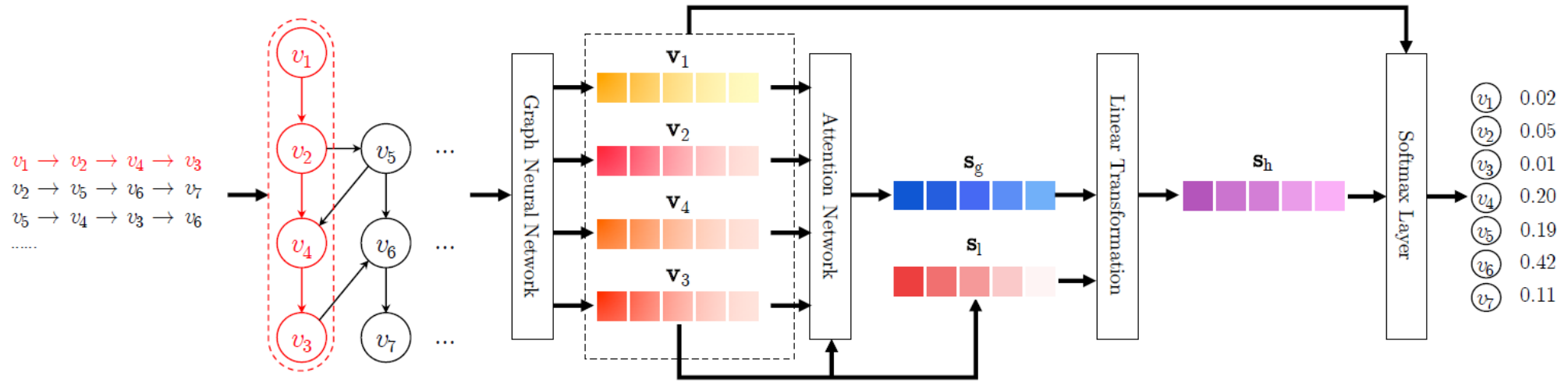
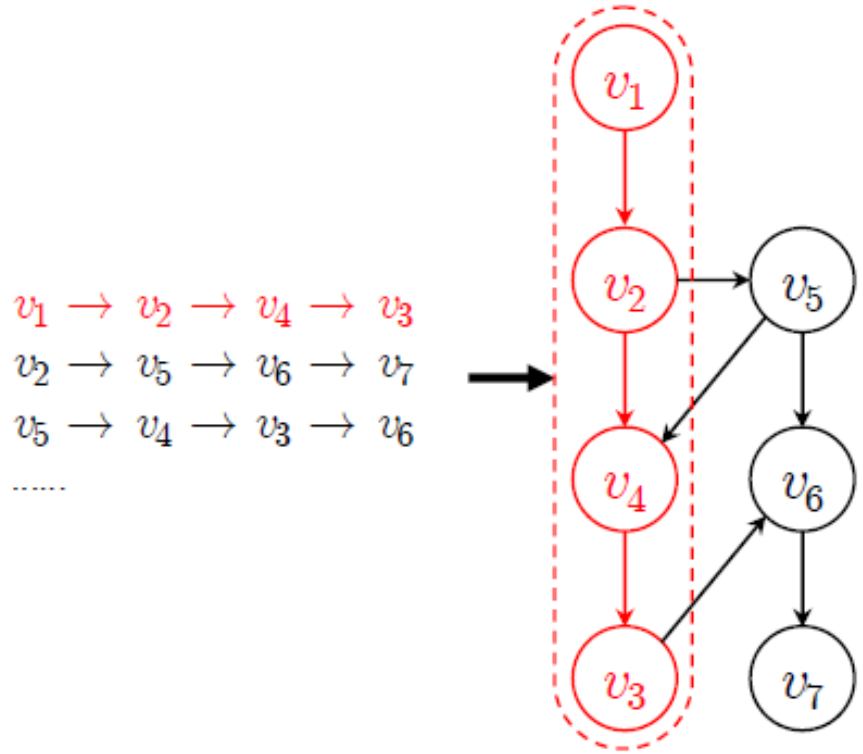


Figure 1: The workflow of the proposed SR-GNN method. We model all session sequences as session graphs. Then, each session graph is proceeded one by one and the resulting node vectors can be obtained through a gated graph neural network. After that, each session is represented as the combination of the global preference and current interests of this session using an attention net. Finally, we predict the probability of each item that will appear to be the next-click one for each session.

Method

- Workflow of SR-GNN

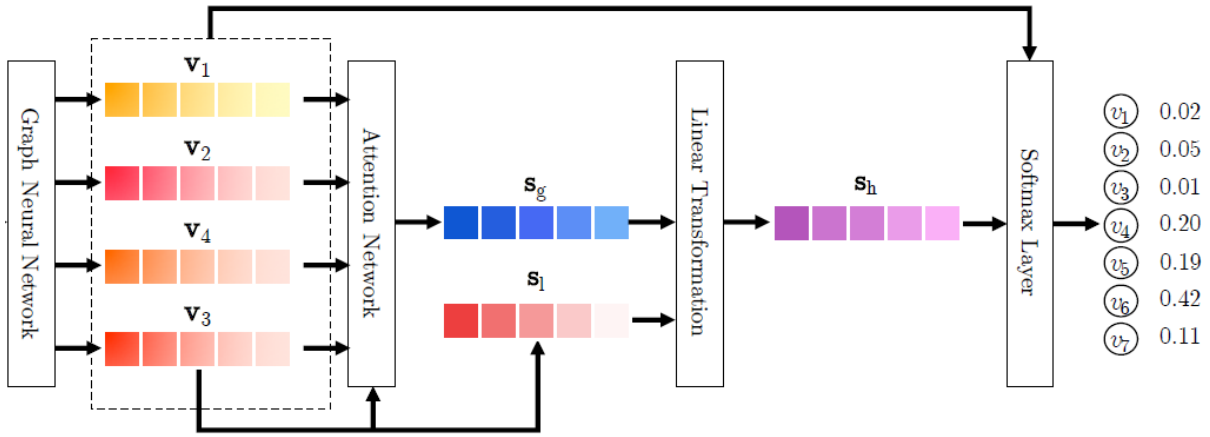
- $v_1 \sim v_7$: 아이템 리스트
- $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3$: session
- $v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7$: session



Method

- Workflow of SR-GNN

- V_1 : 노드 잠재 벡터 (node latent vector)
- S_g : Global Session Vector
  → 모든 벡터를 aggregate
  → attention network를 통해 가중치 부여
- S_l: Local Session Vector
  → 마지막 노드 벡터 (last click item)
- S_h: 최종 임베딩
  → Sg, Sl을 concat하여 선형 변환



Method

- Notation
  - $V = \{v_1, v_2, \dots, v_m\}$: 전체 세션 내에 포함된 unique한 아이템들의 집합으로 표현
  - $s = [v_{s,1}, v_{s,2}, \dots, v_{s,n}]$: 익명의 세션 sequence s의 시간 순서에 따른 item을 나타냄
    → $v_{s,i} \in V$: 세션 내 클릭된 아이템
  - 모델링의 목표는 세션 s에 대해 그 다음에 클릭될 아이템 $v_{s,n+1}$을 예측하는 것
  - 해당 모델 내에서, 세션 s 의 모든 아이템에 대해 확률($\hat{y}$, score)을 구함
- Constructing Session Graphs
  - 각 session sequence s는 **directed graph $\boldsymbol{G_s} = (\boldsymbol{V_s}, \boldsymbol{E_s})$**로 표현 가능
  - 해당 세션 내 각 노드는 item $v_{s,i}$로 표현
  - 해당 세션 내 각 엣지는 $(v_{s,i-1}, v_{s,i}) \in E_s$
  - 여러 아이템이 중복되어 나오기에 **각 엣지의 weight는 normalize 진행**
    → 해당 엣지가 시작된 노드의 degree로 나눠주는 방식
  - 모든 item을 GNN을 통하여 **d차원 벡터공간에 임베드** ($\boldsymbol{v} \in \mathbb{R}^{\boldsymbol{d}}$)
  - 노드 벡터에 기반하여 **각 세션 s는 임베딩 벡터 s로 매핑**되고, graph 내의 노드 벡터가 됨

Method

- Learning Item Embeddings on Session Graphs

- $\mathrm{H} \in \mathbb{R}^{d \times 2d}$: weight control
- $z_{s,i}$ $r_{s,i}$: reset, update gates
- $[\mathrm{v}_1^{t-1}, \ldots, \mathrm{v}_n^{t-1}]$: session s 내의 노드 벡터리스트
- $\sigma(\cdot)$: sigmoid 함수
- $\odot$: element-wise multiplication
- v_i: 노드 $v_{s,i}$ 의 latent vector
- $A_s \in \mathbb{R}^{n \times 2n}$: 노드들 간의 관계 표현을 정의하는 connection matrix
  - $A_s^{(in)}$과 $A_s^{(out)}$으로 이루어짐

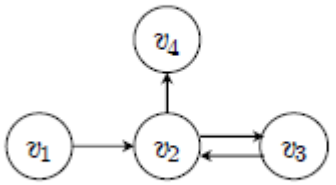
$$\mathbf{a}_{s,i}^t = \mathbf{A}_{s,i:} \left[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}\right]^\top \mathbf{H} + \mathbf{b}, \quad (1)$$

$$\mathbf{z}_{s,i}^t = \sigma\left(\mathbf{W}_z \mathbf{a}_{s,i}^t + \mathbf{U}_z \mathbf{v}_i^{t-1}\right), \quad (2)$$

$$\mathbf{r}_{s,i}^t = \sigma\left(\mathbf{W}_r \mathbf{a}_{s,i}^t + \mathbf{U}_r \mathbf{v}_i^{t-1}\right), \quad (3)$$

$$\widetilde{\mathbf{v}}_i^t = \tanh\left(\mathbf{W}_o \mathbf{a}_{s,i}^t + \mathbf{U}_o \left(\mathbf{r}_{s,i}^t \odot \mathbf{v}_i^{t-1}\right)\right), \quad (4)$$

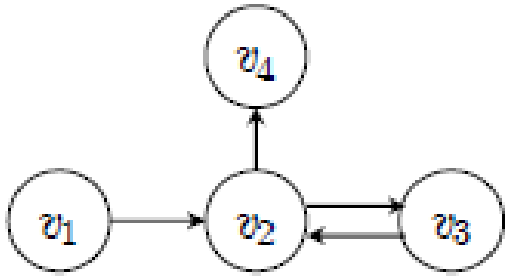
$$\mathbf{v}_i^t = \left(1 - \mathbf{z}_{s,i}^t\right) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_{s,i}^t \odot \widetilde{\mathbf{v}}_i^t, \quad (5)$$



	Outgoing edges				Incoming edges			
	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Method

- Learning Item Embeddings on Session Graphs



	Outgoing edges				Incoming edges			
	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Method

- Generating Session Embeddings

- Attention mechanism을 통해 Sg를 생성
- Sg와 Sl의 선형변환을 통해 Sh 생성

$$\alpha_i = \mathbf{q}^\top \sigma(\mathbf{W}_1 \mathbf{v}_n + \mathbf{W}_2 \mathbf{v}_i + \mathbf{c}),$$
$$\mathbf{s}_\mathrm{g} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{6}$$

$$\mathbf{s}_\mathrm{h} = \mathbf{W}_3 \left[\mathbf{s}_\mathrm{l}; \mathbf{s}_\mathrm{g}\right], \tag{7}$$

Method

- Making Recommendation and Model Training

- 각 후보 item의 score를 각 아이템별 임베딩과 세션 임베딩의 행렬곱으로 얻음
- 이후 softmax 취함

$$\hat{\mathbf{z}}_i = \mathbf{s}_{\mathrm{h}}^{\top} \mathbf{v}_i. \tag{8}$$

$$\hat{\mathbf{y}} = \mathrm{softmax}\left(\hat{\mathbf{z}}\right), \tag{9}$$

- Loss 함수는 관측과 예측값에 대한 cross entropy

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{m} \mathbf{y}_i \log\left(\hat{\mathbf{y}}_i\right) + \left(1 - \mathbf{y}_i\right) \log\left(1 - \hat{\mathbf{y}}_i\right), \tag{10}$$

- BPTT 알고리즘 사용 (Back-Propagation Through Time)

Experiments

- Datasets

  - Yoochoose
    - RecSys Challenge 2015
    - E-commerce 6개월치 유저 클릭 데이터
    - 7,981,580개 세션
    - 37,483개 아이템
  - Diginetica
    - CIKM Cup 2016
    - Transaction 데이터
    - 204,771개 세션
    - 43,097개 아이템

- 전처리
  - 길이 1짜리 세션 삭제
  - 5번 아래로 노출된 아이템 삭제
  - Input Sequence에 대한 Label 생성

Experiments

- Baseline Algorithms
  - POP & S-POP
  - Item-KNN
  - BPR-MF
  - FPMC
  - GRU4REC
  - NARM
  - STAMP

- Metrics
  - P@20(precision)
    - top-20
  - MRR@20(Mean Reciprocal Rank)

Experiments

- Hyperparameters

- Latent vector dimension : 100
- Validation set : 10% random subset
- 전체 파라미터 : Gaussian distribution (mean 0, std 1)
- Optimizer: Adam
- Learning Rate: 0.001
- Weight decay : 0.1 / 3 epochs
- Batch size : 100
- L2 penalty : 0.00001

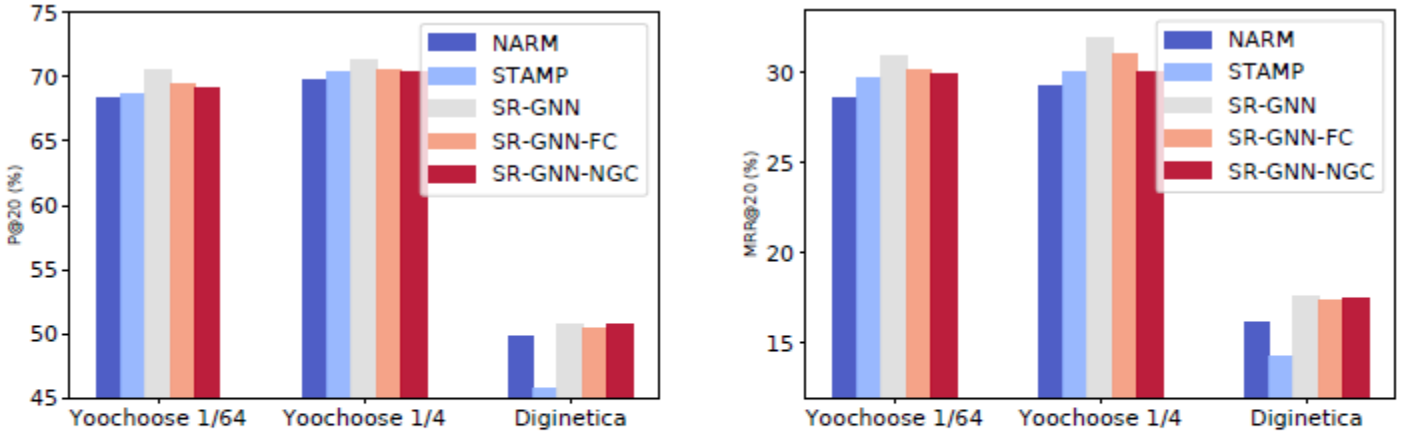
Experiments

- Baselines

Method	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20
POP	6.71	1.65	1.33	0.30	0.89	0.20
S-POP	30.44	18.35	27.08	17.75	21.06	13.68
Item-KNN	51.60	21.81	52.31	21.70	35.75	11.57
BPR-MF	31.31	12.08	3.40	1.57	5.24	1.98
FPMC	45.62	15.01	–	–	26.53	6.95
GRU4REC	60.64	22.89	59.53	22.60	29.45	8.33
NARM	68.32	28.63	69.73	29.23	49.70	16.17
STAMP	68.74	29.67	70.44	30.00	45.64	14.32
SR-GNN	**70.57**	**30.94**	**71.36**	**31.89**	**50.73**	**17.59**

Experiments

- Session Embeddings



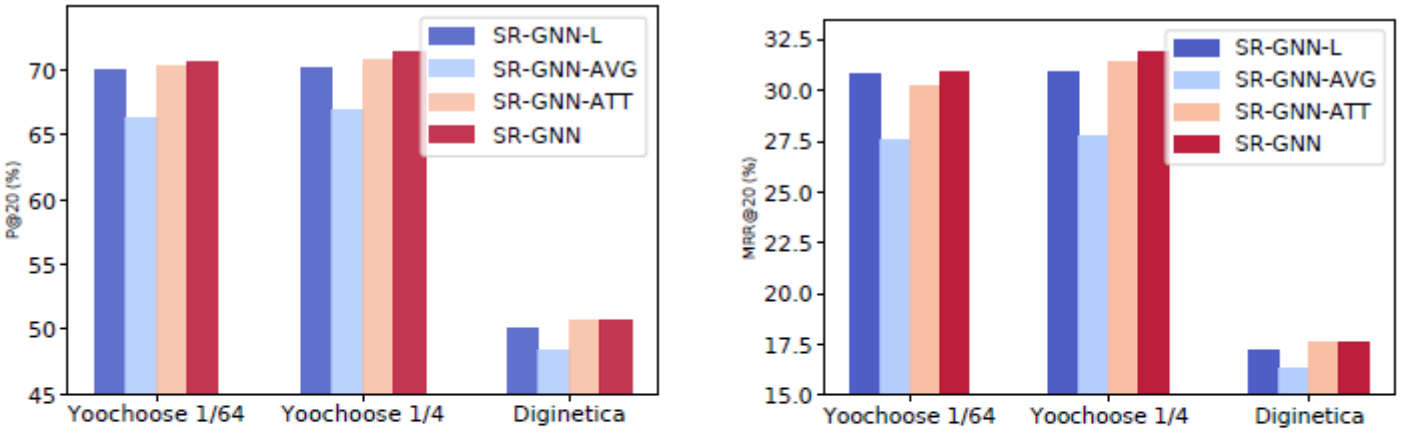
(a) P@20 (b) MRR@20

Figure 3: The performance of different connection schemes

- SR-GNN-FC : full connections
- SR-GNN-NGC : normalized global connections

Experiments

- Session Embeddings



(a) P@20 (b) MRR@20

Figure 4: The performance of different session representations

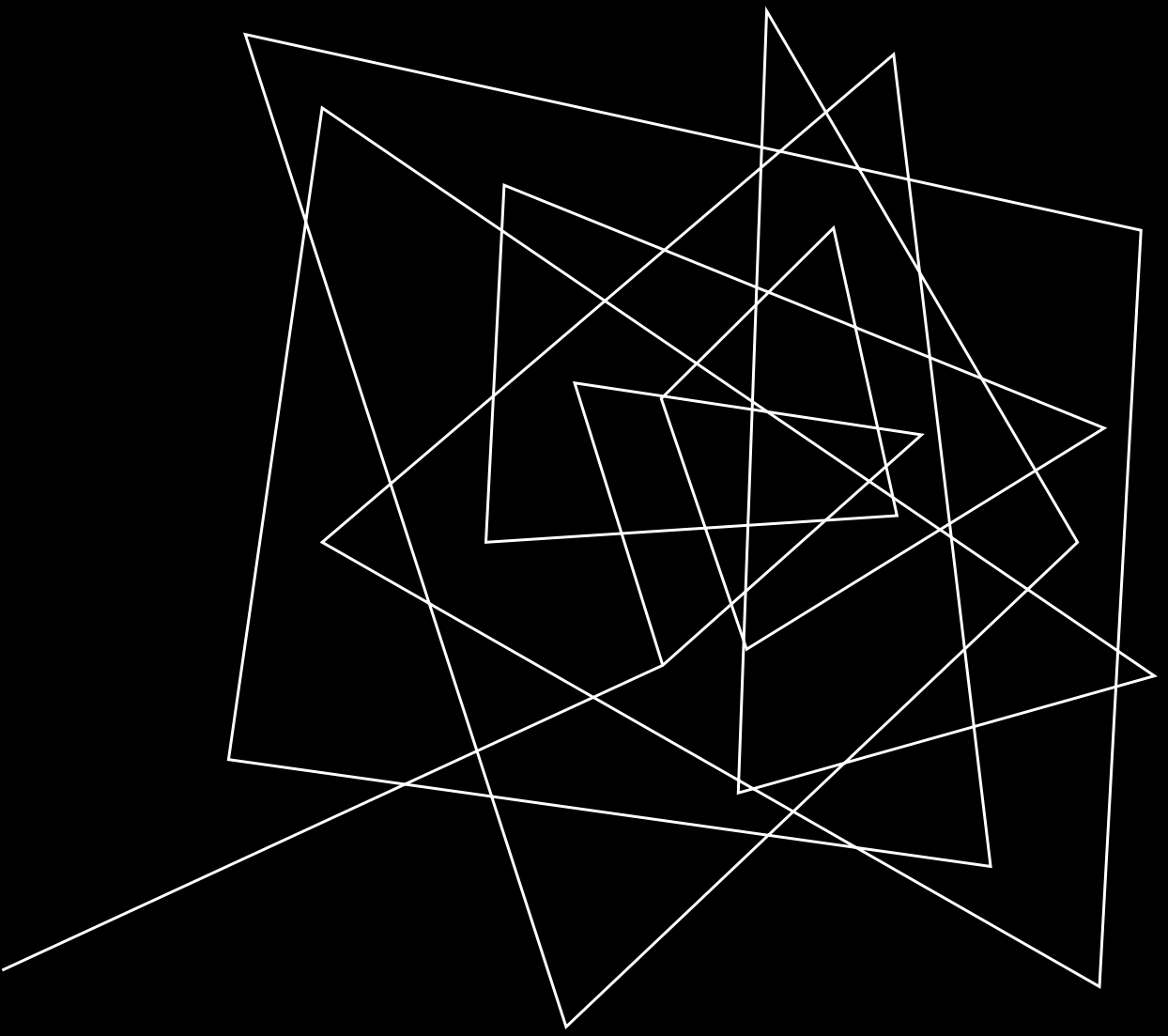
- SR-GNN-L : local embedding only
- SR-GNN-AVG : average pooling
- SR-GNN-ATT : attention mechanism

Experiments

- Session Sequence Lengths

Table 3: The performance of different methods with different session lengths evaluated in terms of P@20

Method	Yoochoose 1/64		Diginetica	
	Short	Long	Short	Long
NARM	**71.44**	60.79	**51.22**	45.75
STAMP	70.69	64.73	47.26	40.39
SR-GNN-L	70.11	69.73	49.04	50.97
SR-GNN-ATT	70.31	70.64	50.35	51.05
SR-GNN	70.47	**70.70**	50.49	**51.27**



Q&A