Session-Based Recommendation with Graph Neural Network

- Abstract:
  - session 是匿名的，意味着没有用户本身的内容信息可以用
  - 原来的模型把一个 session 当做一个 sequence 来进行建模，但是这样没有考虑物体之间复杂的过渡关系（sequential model）
  - GNN 利用了图的 structured 特性从而可以体现出物体之间的过渡关系
  - 使用 attention 的机制，可以在 representation 中既包含 global preference 也包含 current interest
- Introduction
  - 信息过载是推荐系统出现的原因
  - 大部分现存的推荐系统都是假设用户信息以及过往行为都被记录下来，但是很可能这些都没有，只有当前正在进行的 session 被记录下来，所以也就不能依赖 adequate user-item interaction
  - session-based 相关工作
    - Markov chains（假设下一时刻只与上一时刻相关）
    - RNN：通过 data augmentation 和考虑用户行为的 temporal shift 可以让模型效果更好。
    - NARM 设计了 global and local RNN recommender 从而同时得到用户的 sequential behavior 和 main

purpose

- ◆ STAMP :也是捕获 general interest 和 current interest，不过使用的是 MLP networks（多层感知器）和 attentive net
- ◆ 上面模型的一些问题：
  - 需要一个 session 里面大量的用户行为（通常 RNN 的 hidden vector 可以认为是用户的 representation？）用户行为仅仅是 session clicks 因此难以反映其 representation
  - 仅考虑连续物品之间的关联，没有考虑一个 session 中与其他 item 之间的 transition
  - 相隔很远的物品可能也有复杂的 transition，这点被忽视了

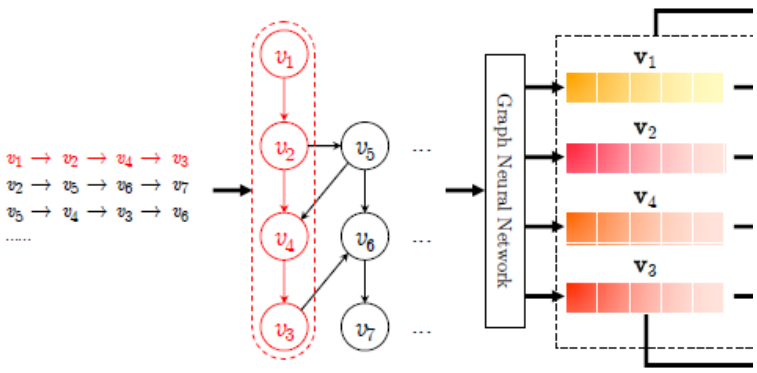
- ■ 关于 GNN 的一些工作
  - ◆ 最早是为了给图一个 representation 而出现的
  - ◆ 近些年在 NLP，CV 都有所应用
    - script event prediction
    - situation recognition
    - image classification
- ■ 接下来讲到这个 model 的 workflow
  - ◆ 把每个 session sequence 变成一个有向图，每个 session 都是其中的子图

- 然后通过 GGNN(gated graph neural network)，每个图中的每个点都可以用向量表示出来
- 接下来使用上一步所有 node 的 latent vector 表示出 session representation，这里获取了 global 和 local session embedding vector
- 最后为每个 session 预测下一个每个物品下一次 click 的概率分布

- Related Work
  - 传统的推荐方法
    - Matrix Factorization 把 user-item rating matrix 进行矩阵分解，分解出一个用户矩阵一个物品矩阵，这个不适合 session-based 因为 session-based 只是一些 clicks 而已
    - item-based neighborhood method 无法考虑 order
    - Markov decision processes
    - FPMC（对个性化概率转移矩阵的分解？）
  - Deep-learning-based methods
    - RNN
    - CNN(将 session click 与 content feature 结合其起来)
    - list-wise deep neural network(?)
    - encoder-decoder architecture NARM
    - STAMP(short-term attention priority model)

- NN on graphs
  - DeepWalk: 用随机游走来学习节点的表示
  - LINE and node2vec
  - CNN （只能作用在无向图上）
  - GNN 能作用在有向图上
  - GGNN（gated GNN）使用了 gated recurrent unit*，用 back-propagation through time（BPTT*）来计算梯度

- The Proposed Method
  - Notations
    - $V = \{v_{1,}v_{2,} \dots, v_{m,}\}$：在所有 session 中出现的全体物品
    - 一个匿名的 session 表示为：$s = [v_{s,1}, v_{s,2}, \dots, v_{s,n}]$
    - goal 是为一个 session s 来预测 $v_{s,n+1}$
    - 输出的是概率分布，top-K 概率的物品将作为候选推荐
  - Constructing Session Graphs
    - 每条边的权重等于这条边的出现次数除以起点的出度
    - 把这个图投入到 GNN 中可以得到每个点的 latent vector



- Learning Item Embeddings on Session Graphs

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

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

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

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

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

(1)式是提取出两个邻居的节点 latent vector 并作为 GNN 的输入，然后会有 update 和 reset gate 来确定哪些信息保留下来哪些丢弃，然后用上一个 state，当前 state，reset gate 从而得到 candidate state，最后把上一个 hidden state 和当前 candidate state 在 update gate 的控制下进行组合，不断更新直到收敛就可以获得 node vector 了

- Generating Session Embeddings
  - 在表达 session embedding 时：分成 local 与 global
  - local 直接用 last-clicked item 的向量来表示
  - 一个 session 里面不同的 vector 有不同的权重，这里使用 soft-attention 机制，参数 q, W1, W2 都可以控制权重

$$\alpha_i = \mathbf{q}^{\top}\sigma(\mathbf{W}_1\mathbf{v}_n + \mathbf{W}_2\mathbf{v}_i + \mathbf{c}),$$

$$\mathbf{s}_g = \sum_{i=1}^{n}\alpha_i\mathbf{v}_i,$$

  - 最后算总的 embedding，就是把 Sg 和 Sl concatenate 起来，再做线性变换

$$\mathbf{s}_h = \mathbf{W}_3\left[\mathbf{s}_l; \mathbf{s}_g\right]$$

- Making Recommendation and Model Training
  - 直接用 item 的 vector 和 session 的 vector 做点积，可以得到这个 item 作为 candidate 的 score

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

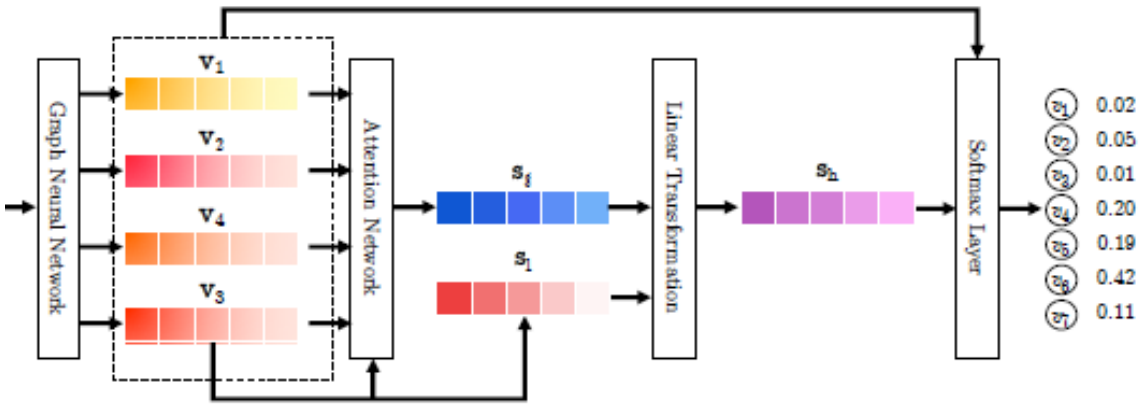
  - 最后一个 softmax 输出每个 item 的成为当前 session 的 next click 的概率

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

  - 损失函数就是直接 cross entropy

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

  - train 的方法是用 Back-Propagation Through Time(BPTT)，为了防止过拟合，建议用相对小的 training steps



- Experiments and Analysis
  - Datasets

- 删除了长度为 1 的 session 以及出现次数少于 5 的 items
- Yoochoose: 7981580 sessions and 37483 items
- Diginetica: 204771 sessions and 43097 items
- Yoochoose 中设定接下来的天为 test set，Diginetica 设置星期为 test set

Table 1: Statistics of datasets used in the experiments

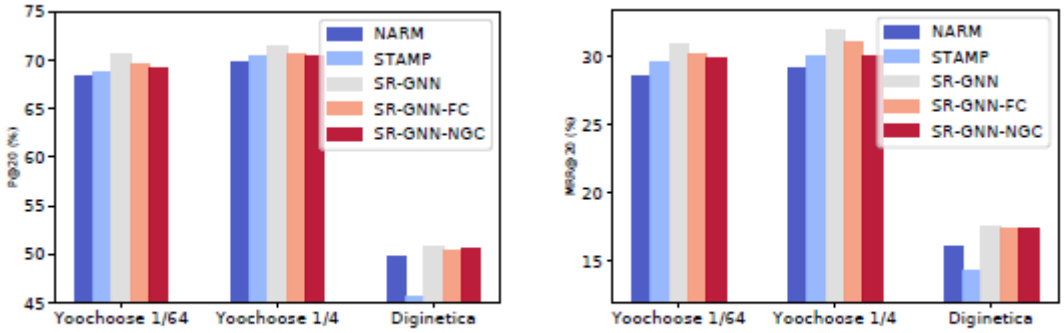
Statistics	*Yoochoose 1/64*	*Yoochoose 1/4*	*Diginetica*
# of clicks	557,248	8,326,407	982,961
# of training sessions	369,859	5,917,745	719,470
# of test sessions	55,898	55,898	60,858
# of items	16,766	29,618	43,097
Average length	6.16	5.71	5.12

- Evaluation Metrics
  - P@20 表示在前 20 个 item 正确推荐的物品占的比例
  - MRR@20 (mean reciprocal rank) 正确推荐的物体在前 20 推荐列表里面排第几位，值越大说明排的越前
- Parameter Setup
  - latent vector 的维度 d = 100
  - validation set 是 training set 中随机抽取 10%
  - 用(0, 0.1)的高斯分布初始化所有参数
  - Adam optimizer：learning rate 初始为 0.001 然后每过 3 个 epoch 就会衰减 0.1 倍
  - batch size = 100　L2 penalty = 10^(-5)
- Comparison with Baseline Methods

Table 2: The performance of SR-GNN with other baseline methods over three datasets

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**

- Comparison with Variants of Connection Schemes
  - 把所有 session 都合起来变成一张有向图(SR-GNN-NGC)
  - 同一个 session 内的点全连接，边用 boolean 变量来表示（SR-GNN-FC）

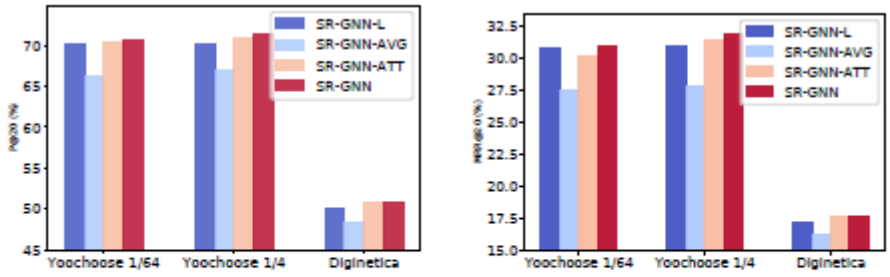


(a) P@20　　(b) MRR@20

由于边的权重是会随着领边数目增加而降低的，那么在 SR-GNN-NGC 中在当前 session 中的边就是因为受到别的 session 的影响权重变得更低，因此效果不如 SR-GNN。而在 SR-GNN-FC 中，把 high-order 的关系直接连接起来是不太好的，因为有些 intermediate stage 的确是有必要的，比如说 A->B->C 可能就不能把 C 直接推荐为 A 的

下一个，有可能 A 没有直接到 C 的链接。

- Comparison with Different Session Embeddings
  - local embedding only (SR-GNN-L)
  - global embedding with the average pooling (SR-GNN-AVG)
  - global embedding with the attention mechanism (SR-GNN-ATT)



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

Figure 4: The performance of different session representations

- Analysis on Session Sequence Lengths
  - Short vs Long = 0.701 : 0.299 (Yoochoose)
  - Short vs Long = 0.764 : 0.236 (Diginetica)

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**

- 可以看到 NARM 和 STAMP 在 short 和 long 的时候差距很大，而他们又都是基于 RNN 的，所以 RNN 难以捕获长序列的信息。

- Conclusions
  - graph models into representing session sequences
  - not only consider the complex structure and transitions, but also combine long-term preferences and current interests