文献笔记

8.26-9.2

1.STAN: Spatio-Temporal Attention Network for Next Location Recommendation

(1) 研究内容：考虑非相邻位置和非连续访问来理解用户行为，聚合来自用户轨迹的所有相关访问并从加权表示中召回最合理的 next-POI

(2) 文章整体要点：

- 提出了一个用于位置推荐的时空注意网络 (STAN)
  - STAN 明确利用轨迹上所有带有 self-attention 层的签到的相对时空信息。这种改进允许在非相邻位置和非连续签到之间进行点对点交互，并具有明确的时空效应
  - STAN 使用双层注意力架构，首先聚合用户轨迹内的时空相关性，然后考虑个性化项目频率 (PIF) 召回目标
- 目前的模型仍未解决的问题：
  - 非相邻位置和非连续访问之间的相关性没有得到有效的学习
    - 用户的移动性可能更多地取决于几天前访问过的相关位置，而不是刚刚访问过的不相关位置
  - 用户访问功能相关/相似的远距离位置并不少见。而大多数当前模型关注的是当前和未来步骤之间的空间和/或时间差异，而忽略了时空轨迹内的相关性
  - 先前实践的用于空间离散化的分层网格化对空间距离不敏感。基于网格的注意力网络聚合了相邻位置，但无法感知空间距离。彼此靠近的网格与那些不靠近的网格没有区别，会抛出大量空间信息。
  - 以前的模型广泛忽视了个性化项目频率（PIF）。对同一地点的重复访问反映了频率，这强调了重复位置的重要性以及用户重访的可能性。由于记忆机制和归一化操作，以前的基于 RNN 的模型和自注意力模型很难反映 PIF
- 提出了 STAN，一种用于下一个位置推荐的时空自我注意网络。
  - 设计了一个自注意力层用于聚合历史轨迹中的重要位置，另一个自注意力层用于召回最合理的候选者，两者都考虑了点对点的显式时空效应
  - 自注意力层可以为轨迹内的每次访问分配不同的权重，克服了常用循环层的长期依赖问题。双层系统允许考虑 PIF 的有效聚合

  - 使用线性插值来嵌入时空转换矩阵来解决稀疏问题，这与 GPS 网格不同，它对空间距离很敏感。由于输入到模型中的所有签到的时空效应，STAN 可以学习非相邻位置和非连续访问之间的相关性

- 本文的贡献：
  - 提出了 STAN,一种时空双向注意模型,以充分考虑聚合相关位置的时空效应。STAN 是 POI 推荐中的第一个明确地结合了时空相关性来学习非相邻位置和非连续访问之间的规律的模型。
  - 用一种简单的空间离散化线性插值技术代替 GPS 网格化，可以恢复空间距离并反映用户的空间偏好，而不仅仅是聚合邻居。将此方法集成到 STAN 中以获得更准确的表示。
  - 提出了 PIF 的双向注意架构。第一层聚合轨迹内的相关位置以更新表示，以便第二层可以将目标与所有签到匹配，包括重复。

- 结论：
  - 提出了一个时空注意力网络，缩写为 STAN
  - 使用一个真实的轨迹示例来说明非相邻位置和非连续访问之间的功能相关性，并建议使用双向注意系统来学习轨迹内的显式时空相关性
  - 该架构首先聚合轨迹内的时空区间 ，然后调用目标。由于轨迹的所有表示都是加权的，因此目标的召回充分考虑了个性化项目频率（PIF）的影响。
  - 提出了一种用于匹配计算交叉熵损失的平衡采样器，优于常用的二元和/或普通交叉熵损失。
  - 在实验部分进行全面的消融研究、稳定性研究和可解释性研究
  - 建议用简单的线性插值技术代替用于空间离散化的层次网格方法，该技术可以在提供密集表示的同时反映连续的空间距离

(3) 问题描述：

- 给定大小为 U 的用户集合 $U = \{u_1, u_2, ..., u_U\}$, 大小为 L 的位置集合 $L = \{l_1, l_2, ..., l_L\}$ 以及大小为 T 的时间集合 $T = \{t_1, t_2, ..., t_T\}$
- Historical Trajectory: 用户 u_i 的轨迹是按时间排序的签到。每个签到 r_k 都是一个元组 (u_i, l_k, t_k)，其中 l_k 是位置，t_k 是时间戳。每个用户有一个可变长度的轨迹 $tra(u_i) = r_1, r_2, ..., r_{m_i}$。将每个轨迹转换成一个固定长度的序列 $seq(u_i) = r_1, r_2, ..., r_n$
- Trajectory Spatio-Temporal Relation Matrix: 将时间间隔和地理距离建模为两个访问位置之间明确的时空关系, 具体而言，轨迹空间关系矩阵 $\Delta^s \in R^{n \times n}$ 和轨迹时间关系矩阵 $\Delta^t \in R^{n \times n}$ 分别表示为:
  -

$$\Delta^{t,s} = \begin{bmatrix} \Delta^{t,s}_{11} & \cdots & \Delta^{t,s}_{1n} \\ & \ddots & \vdots \\ \Delta^{t,s}_{n1} & & \Delta^{t,s}_{nn} \end{bmatrix}$$

- Candidate Spatio-Temporal Relation Matrix: 考虑了本文中的下一个时空矩阵。计算每个候选位置 i 和每个签到位置 j 之间的距离为 $N^s_{i,j}$, 并表示 t_{m+1} 和 $t_1, t_2, .., t_m$ 重复 L 次以扩展为 2D 之间的时间间隔 $N^t_{i,j}$。候选空间关系矩阵 N^s 和候选时间关系矩阵 N^t 分别表示为:
  -
$$N^{t,s} = \begin{bmatrix} N^{t,s}_{11} & \dots & N^{t,s}_{1n} \\ & \ddots & \vdots \\ N^{t,s}_{n1} & & N^{t,s}_{nn} \end{bmatrix}$$
- Mobility Prediction: 目标是找到所需的输出 l

4 STAN:

- 整体架构:
  - 一个多模态嵌入模块，用于学习用户、位置、时间和时空效应的密集表示
  - 一个自注意力聚合层，聚合用户轨迹内的重要相关位置，以更新每次签到的表示
  - 一个注意力匹配层，用于从加权签到表示中计算 softmax 概率，以计算每个候选位置对下一个位置的概率
  - 一个平衡的采样器，用于使用一个正样本和几个负样本来计算交叉熵损失
  - STAN: 图 1
- Multimodal Embedding Module: 多模态嵌入模块由两部分组成，即轨迹嵌入层和时空嵌入层
  - User Trajectory Embedding Layer:
    * 多模态嵌入层用于将用户 (e^u)、位置 (e^l) 和时间 (e^t) 编码为潜在表示
    * 嵌入模块被合并到其他模块中，以将标量转换为密集向量，以减少计算并改善表示
    * 每个签到 r 的用户轨迹嵌入层的输出是 $e^r = e^u + e^l + e^t$ 。对于每个用户序列 $seq(u_i) = r_1, r_2, ..., r_n$ 的嵌入，表示为 $E(u_i) = e^{r_1}, e^{r_2}, ..., e^{r_n}$
  - Spatio-Temporal Embedding Layer: 单位嵌入层用于密集表示空间差和时间差
    * 该层将空间和时间间隔分别与单位嵌入向量 e_{Δ_s} 和 e_{Δ_t} 相乘。单位嵌入向量反映了具有基本单元的连续时空上下文，避免了密集维度的稀疏编码。特别是，可以用这种对空间距离敏感的技术来代替层次网格化方法，这种方法只聚合相邻的位置，不能表示空间距离
      . $\begin{cases} e^{\Delta t}_{i,j} = \Delta^t_{ij} \times e_{\Delta t} \\ e^{\Delta s}_{i,j} = \Delta^s_{ij} \times e_{\Delta s} \end{cases}$
    * 还可以考虑另一种插值嵌入层，它设置一个上限单位嵌入向量和一个下限单位嵌入向量，并将显式区间表示为线性插值，这是一种近似值到单元嵌入层。插值嵌入计算如下:

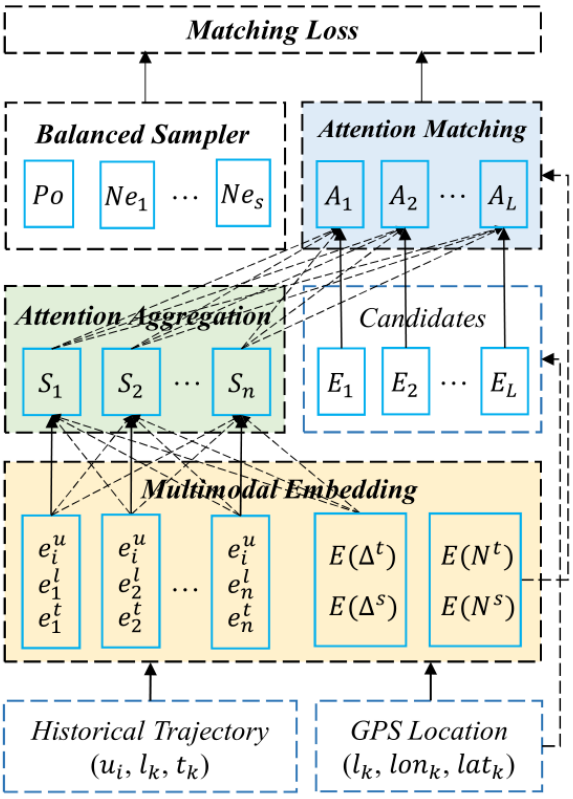


Figure 2: The architecture of the proposed STAN model.

图 1:

$$\cdot \begin{cases} e_{i,j}^{\Delta t} = \dfrac{e_{\Delta t}^{sup}(Upper(\Delta t) - \Delta t) + e_{\Delta t}^{inf}(\Delta t - Lower(\Delta t))}{Upper(\Delta t) - Lower(\Delta t)} \\ e_{i,j}^{\Delta s} = \dfrac{e_{\Delta s}^{sup}(Upper(\Delta s) - \Delta s) + e_{\Delta s}^{inf}(\Delta s - Lower(\Delta s))}{Upper(\Delta s) - Lower(\Delta s)} \end{cases}$$

∗ 该层处理两个矩阵：轨迹时空关系矩阵和候选时空关系矩阵，可以使用最后一个维度的加权和，并将空间和时间嵌入相加来创建：

$$\cdot \begin{cases} E(\Delta) = Sum(E(\Delta^t)) + Sum(E(\Delta^s)) \in R^{n \times n} \\ E(N) = Sum(E(N^t)) + Sum(E(N^s)) \in R^{L \times n} \end{cases}$$

- Self-Attention Aggregation Layer:
  - 提出了一个扩展模块来考虑轨迹中两次访问之间的不同空间距离和时间间隔。该模块旨在聚合相关的访问位置并更新每次访问的表示。自注意力层可以捕获长期依赖关系，并为轨迹内的每次访问分配不同的权重。
  - 可以得到 S(u) 作为用户轨迹的更新表示
  - $S(u) = Attention(E(u)W_Q, E(u)W_k, E(u)W_V, E(\Delta), M)$
  - $Attention(Q, K, V, \Delta, M) = (M * softmax(\frac{QK^T + \Delta}{\sqrt{d}}))V$

- Attention Matching Layer:
  - 该模块旨在通过与用户轨迹的更新表示相匹配，从所有 L 个位置中召回最合理的候选者。该层计算每个候选位置成为下一个位置的概率为:
    * A(u)=Matching(E(l),S(u),E(N))
    * Matching(Q,K,N)=Sum(softmax($\frac{QK^T+N}{\sqrt{d}}$))
- Balanced Sampler：
  - 普通 cross-entropy loss 写成：
    * $-\sum_i \sum_{m_i}(log\sigma(a_k) + \sum_{j=1,j!=k}^{L} log(1-\sigma(a_j)))$
  - 可以简单地将交叉熵损失中使用的负样本数设置为超参数 s。在这里，提出了一个平衡采样器，用于在训练的每个步骤中随机采样负样本。因此，在每个训练步骤之后更新负采样器的随机种子。损失计算为:
    * $-\sum_i \sum_{m_i}(log\sigma(a_k) + \sum_{j_1,j_2,...,j_s \in [1,L]} \sum_{(j_1,j_2,...j_s)!=k} log(1-\sigma(a_j)))$

(5) 实验部分：

- 数据集：Gowalla;SIN;TKY;NYC
- 评价指标：
  - Recall@5
  - Recall@10
- Result:

Table 2: Recommendation performance comparison with baselines.

	Gowalla		TKY		SIN		NYC	
	Recall@5	Recall@10	Recall@5	Recall@10	Recall@5	Recall@10	Recall@5	Recall@10
STRNN	0.1664	0.2567	0.1836	0.2791	0.1791	0.2016	0.2365	0.2802
DeepMove	0.1959	0.2699	0.2684	0.3509	0.2389	0.3155	0.3268	0.4014
STGN	0.1528	0.2422	0.1940	0.2710	0.2292	0.2727	0.2439	0.3015
ARNN	0.1810	0.2745	0.1852	0.2696	0.1817	0.2538	0.1970	0.3483
LSTPM	0.2015	0.2701	0.2568	0.3310	0.2579	0.3327	0.2791	0.3564
TiSASRec	0.2411	0.3546	0.3031	0.3693	0.2963	0.3753	0.3664	0.5020
GeoSAN	0.2764	0.3645	0.2957	0.3740	0.3397	0.3943	0.4006	0.5267
STAN	**0.3016**	**0.3998**	**0.3461**	**0.4264**	**0.3751**	**0.4301**	**0.4669**	**0.5962**
Improvement	9.12%	9.68%	17.04%	14.01%	10.42%	9.08%	16.55%	13.20%

图 2:

–

- 消融实验：图 3

Table 3: Ablation Analysis, in which we compare different modules in STAN.

	Gowalla		TKY		SIN		NYC	
	Recall@5	Recall@10	Recall@5	Recall@10	Recall@5	Recall@10	Recall@5	Recall@10
STAN	**0.3016**	**0.3998**	**0.3461**	**0.4264**	**0.3751**	**0.4301**	**0.4669**	**0.5962**
-TIM-BS	0.2835	0.3718	0.3006	0.3819	0.3416	0.3873	0.4126	0.5245
-EWTI-BS	0.2794	0.3717	0.3052	0.3781	0.3404	0.3890	0.4083	0.5272
-TIM	0.2946	0.3925	0.3315	0.4099	0.3643	0.4176	0.4495	0.5814
-SIM-BS	0.2823	0.3729	0.3123	0.3865	0.3337	0.3901	0.4126	0.5299
-EWSI-BS	0.2812	0.3724	0.3132	0.3794	0.3313	0.3916	0.4124	0.5277
-SIM	0.2977	0.3908	0.3405	0.4141	0.3636	0.4165	0.4502	0.5860
-ALL	0.2645	0.3531	0.2867	0.3660	0.3239	0.3776	0.3896	0.5094

图 3:

–

  - 平衡采样器对于提高推荐性能非常重要，它可以提高近 5-12% 的召回率。空间和时间间隔可以明确表达非连续访问和非相邻位置之间的相关性，添加空间距离和时间间隔都可以使召回率提高近 4-8%。本文引入时空相关性的方法等效于 TiSASRec 中使用的方法，且本文的方法更容易实现

- Stability Study: 图 4
  - Embedding dimension：

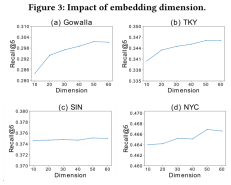


图 4:

    *
  - Number of negative samples: 图 5

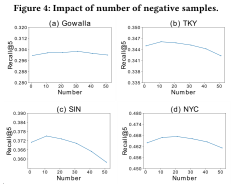


图 5:

    *

- Interpretability Study:
  - STAN 的机制，自注意力聚合层执行的非连续访问和非相邻位置的聚合是核心
  - 图 6

2.Learning Graph-based Disentangled Representations for Next POI Recommendation

(1) 研究内容：考虑 POI 的内在特征，即来自 POI 不同方面的特性对 next-POI 推荐的影响

(2) 文章整体要点：

- 提出了一种新型的 Disentangled Representation-enhanced Attention Network (DRAN)：
  - 利用 Disentangled Representation 来明确地模拟不同方面和相应的影响，以更精确地表示一个 POI
  - 设计了一个传播规则，通过完善两种类型的 POI 关系图来学习基于图的分解表征，充分利用基于距离和过渡的影响来学习表征

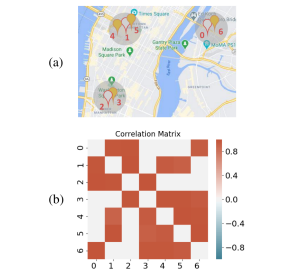


Figure 5: The mechanism of STAN. (a) An example map showing the aggregation of visited locations. The locations with the same colored marks and locations within the range of the same dark circles are aggregated. This gives solid evidence that non-adjacent locations may be correlated and aggregated in our model. (b) The correlation matrix. Here, we take the softmax of the multiplication of query and key in the self-attention aggregation layer as a correlation matrix, which is used to update the representation of check-ins.

图 6:

  - 扩展了注意力架构，以聚合个性化的时空信息，为下一个时间戳上的动态用户偏好建模，同时保持分解表征的不同组成部分的独立性

- 作者认为先前纳入地理信息的方法未能对 POI 的复杂影响进行建模：
  - POI 对用户过渡的各种潜在影响没有被有效地分解开来
  - 基于距离的 POI 影响也被过度简化了 (没有考虑地形地貌而仅仅考虑距离)
- 把重点放在用 POI 表示法来增强序列建模方案，对潜在的方面和相应的影响有很强的表达能力
  - 提出了一种新型的纠缠表征增强注意网络（DRAN），用于 next-POI 推荐
  - 提出将每个 POI 的表示明确地分解为多个独立的组件，而不是直接在整体的 POI 嵌入中对纠缠的影响进行建模, 每个组件都被用来表示单个方面和相应的影响
  - 提出了一种新的分解图卷积网络（DGCN）,DGCN 学习了两种类型的分解表征，即基于过渡的和基于距离的分解表征
  - DGCN 通过细化全局 POI 关系图，充分利用了 POI 的图结构，避免了只考虑签到序列中的空间背景
  - 通过考虑个性化的时空信息来扩展 self-attention 机制，为用户的动态偏好建模，同时保持分解表示的每个组成部分的独立性
- 本文的贡献：
  - 明确地分解 POI 的多个潜在方面，以捕捉相应的影响。提出了一个新的 DGCN, 在 next-POI 中首次尝试用分解表征学习来处理 POI 之间的复杂关联
  - 提出了 DRAN，充分利用 POI 的全球影响和个人层面的动态偏好来完成下一个 POI 的推荐任务
- 不仅考虑 user-item 关系上，同时考虑底层的 item-item 关系，以在连续的数据中探索不同的用户意图
- DRAN 的整体架构:
  - 基于图的解耦表示建模模块, 利用新提出的 DGCN 层学习 POI 的解耦表示

  - 用户时空偏好建模模块，整合相关时空信息对用户历史签到序列进行建模
  - 预测和优化模块，估计用户偏好并优化所有可训练参数
- 未来的研究：通过引入 POI 边信息来将潜在组件与 POI 属性耦合

(3) 问题描述：一个典型的 next-POI 推荐场景

- 给定大小为 N 的 POI 集合 $L = \{l_1, l_2, ..., l_N\}$(其中 l=(longitude,latitude) 分别表示经度和纬度) 和大小为 M 的用户集合 $U = u_1, u_2, u_M$, 其中用户 u 的历史签到序列记为 $H(u) = \{(l_i^u, t_{l_i^u}) | i = 1, 2, ..., m\}$(其中每个元组 $(l_i^u, t_{l_i^u})$ 表示用户 u 在时间戳 $t_{l_i^u}$ 访问 POIl_i^u
-
  - 输入：用户集合 U、POI 集合 L 和用户签到序列 H
  - 输出：用户 u 在下一个时间戳会感兴趣的 POI 排名列表

(4) DRAN:

- 整体架构：

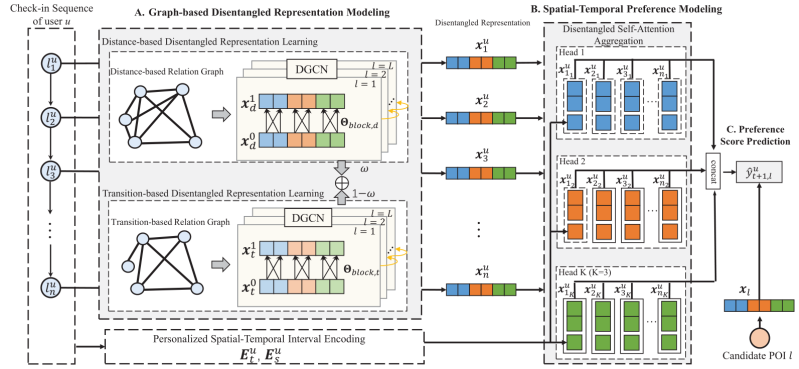


Figure 2: The overall architecture of DRAN. Here, we assume that there are three latent aspects for POIs. DRAN utilizes DGCN to learn the graph-based disentangled representation for each POI, and then aggregates multiple useful information based on check-in sequences for user preference estimation.

图 7:

- Graph-based Disentangled Representation Modeling:
  - POI Relation Graph Construction:
    - 采用两个全局 POI 关系图来指导表示学习。POI 之间的关系一般有两种，即距离关系和过渡关系。分别构造相应的图
      - 基于距离的关系图 $G_d = (V_d, E_d, A_d)$ 是一个无向图
      - 基于转移的关系图 $G_t = (V_t, E_t, A_t)$ 是一个加权图
  - Disentangled Representation Propagation:
    - 利用图卷积操作从 POI 关系图中提取有用信息以学习解耦的表示，同时保持表示的不同组件独立
    - 图卷积操作的参数化为：

- $X^{update} = (D+I)^{-1/2}\hat{A}(D+I)^{-1/2}X\Theta = \widetilde{A}X\Theta$

* $X^{update}(m,n)$ 的第 n 维 x_m^{update} 在卷积运算后计算为:
  - $X^{updata}(m,n) = \sum_{i=1}^{N} A(m,i)(\sum_{j=1}^{N} X(i,j)\Theta(j,n))$
* 只关联同一组件的不同维度, 即第 k 个分量 x_{m_k} 中的每个维度计算为:
  - $X^{updata}(m,n) = \sum_{i=1}^{N} A(m,i)(\sum_{j=index(x_{m_k}(1))}^{index(x_{m_k}(f_{in}))} X(i,j)\Theta(j,n))$
* 可推出 DGCN 的传播规则是:
  - $X^{update} = \widetilde{A}X\Theta_{block}$

    with
  - $\Theta_{block} = diag(\Theta_1, \Theta_2, ..., \Theta_K)$
* 使用 DGCN 来获取两种类型的解缠结表示:
  - 基于距离的 POI 关系图的解缠结表示 x_d 和基于转换的 POI 关系图的解耦表示 x_t
  - 堆叠多个 DGCN 层并引入激活函数以获得不同阶邻居的贡献:
  - $\begin{cases} X_d^{(l+1)} = tanh(\widetilde{A}X_d^l\Theta_{block,d}^l) \\ X_t^(l+1) = tanh(\widetilde{A}X_t^l\Theta_{block,t}^l) \end{cases}$

– Representation Aggregation:

* 利用层聚合策略来组合来自不同层的表示
* 采用 sumpooling 作为聚合函数，公式为:
  - $\begin{cases} X_d = Sum(X_d^0, X_d^1, ..., X_d^L) \\ X_t = Sum(X_t^0, X_t^1, ..., X_t^L) \end{cases}$
* 通过上述方式, 不仅保持了来自不同阶邻居的有影响的信号, 而且还缓解了由于卷积层的增加引起的过度平滑问题。需要注意的是，由于求和操作是逐元素的，因此在聚合后保持了不同组件的独立性
* 分别为 x_t 和 x_d 生成 K 分量，并通过 $x_{d,i}$ 和 $x_{t,i}$ 的加权和得到 POIl_i 的最终解纠缠表示 $x_i \in X_{f_{in}}$
  - $\begin{cases} x_i = (x_{i_1}, x_{i_2}, ..., x_{i_K}) \\ x_{i_j} = \omega_j x_{d,i_j} + (1-\omega_j)x_{t,i_j} \end{cases}$

- User Spatial-Temporal Preference Modeling:

– Personalized Spatial-Temporal Interval Encoding:

* 对 POI 之间的空间和时间关系进行建模, 考虑每个用户都有自己的签到偏好, 采用区间的相对长度来模拟 S(u) 内的空间和时间关系。即 u 的时间区间矩阵 M_t^u 和空间区间矩阵 M_s^u
  .

$$M_t^u = \begin{bmatrix} t_{1,1}^u & \cdots & t_{1,n}^u \\ & \ddots & \vdots \\ t_{n,1}^u & & t_{n,n}^u \end{bmatrix}$$

.

$$M_s^u = \begin{bmatrix} s_{1,1}^u & \cdots & s_{1,n}^u \\ & \ddots & \vdots \\ s_{n,1}^u & & s_{n,n}^u \end{bmatrix}$$

* 为了在序列中生成空间和时间关系的密集表示，嵌入操作用于将缩放间隔投影到 d 维向量。区间矩阵 M_t^u 和 M_s^u 转化为嵌入矩阵 E_t^u 和 E_s^u

.

$$E_t^u = \begin{bmatrix} e_{1,1}^{u,t} & \cdots & e_{1,n}^{u,t} \\ & \ddots & \vdots \\ e_{n,1}^{u,t} & & e_{n,n}^{u,t} \end{bmatrix}$$

.

$$E_s^u = \begin{bmatrix} e_{1,1}^{u,s} & \cdots & e_{1,n}^{u,s} \\ & \ddots & \vdots \\ e_{n,1}^{u,s} & & e_{n,n}^{u,s} \end{bmatrix}$$

– Disentangled Self-Attention Aggregation:

* 为了捕捉签到序列的多级规律性，提出了对相对位置的 self-attention 的扩展，以将 POI 之间的不同关系合并到一个序列中。解耦的表示 $x \in X_{f_{in}}$ 是由几个独立的组件组成的，直接应用 self-attention 聚合会破坏它们的独立性
* 本文在聚合过程中将每个组件分成一个单独的 attention-head，然后将它们收集起来。即给定输入序列 S(u)，首先生成 K 序列，其中仅包含 x_i^u 的单个组件:
  · $S(u)_k = \{x_{1_k}^u, x_{2_k}^u, ..., x_{n_k}^u\}$
* 一个新的序列 $R(u)_k = \{r_{1_k}^u, r_{2_k}^u, ..., r_{n_k}^u\}$ 在第 k 个 attention-head 内产生且元素 $r_{i_k}^u$ 由相关访问 POI 的加权和计算为:
  · $r_{i_k}^u = \sum_{j=1}^{i} \alpha_{ij}(x_{i_k}^u W^V + e_{i,j,k}^{u,t} + e_{i,j,k}^{u,s} + p_j)$
* α_{ij} 表示通过 softmax 函数计算的权重系数为:

$$\alpha_{ij} = \frac{exp \quad a_{ij}}{\sum_{m=1}^{i} exp \quad a_{im}}$$

  · $\{ \quad a_{ij} = \dfrac{x_{i_k}^u W^Q (x_{j_k}^u W^K + e_{i,j,k}^{u,t} + e_{i,j,k}^{u,s} + p_j)^T}{\sqrt{\frac{d}{k}}}$
* 注意: 拆分 $e_{i,j}^u$ 的主要目的是降维
* 在每个 self-attention 层之后应用前馈网络，其赋予模型非线性并将维度之间的交互编码为:
  · $z_{i_k}^u = FFN(r_{i_k}^u) = ReLU(r_{i_k}^u W_{1_k} + b_{1_k})W_{2_k} + b_{2_k}$
* 利用 dropout 正则化、层归一化和残差连接来缓解过度拟合问题并加快训练过程。注意不同的 attention-head 中使用了不同的参数矩阵和偏差，其可以防止不同组件之间潜在的信息纠缠
* 堆叠几个 attention-blocks 后，得到输出序列 $Z(u)_k = z_{1_k}^u, z_{2_k}^u, ..., z_{n_k}^u$。由于 $Z(u)_K$ 的每个元素 $z_{i_k}^u$ 是 POI 的第 k 方面、位置、空间和时间关系的组合表示，因此可以将其视为用户对 POI 的第 k 方面的偏好的表示

    - 充分考虑各个组件的影响，将所有的 K 输出序列整合成一个整体的序列 Z(u) 来表示用户 u 的整个签到行为为:
        - $Z(u) = Concat(Z(u)_1, Z(u)_2, ..., Z(u)_K) = (z_1^u, z_2^u, ..., z_n^u)$
        - 对于每个元素 z_i^u，不仅表示在步骤 i 的用户偏好的整体表示，而且还包括几个独立的组件来描述用户对 POI 不同方面的兴趣
  - User Preference Estimation:
    - 通过 SoftMax 功能计算每个候选人的首选项得分。即在时间戳 t+1 处，POI l 的首选项得分 $\hat{y}_{y+1,l}^u$ 计算为:
        - $\hat{y}_{y+1,l}^u = \frac{exp(z_t^u x_l^T)}{\sum_{i=1}^{L} exp(z_t^u x_i^T)}$
    - 上式的分子可改写成:
        - $exp(z_t^u x_l^T) = exp(\sum_{k=1}^{K} \sum_{i=1}^{\frac{d}{k}} z_{t_k}^u(i) \cdot x_k(i))$
    - 偏好得分可以看作是不同组件通过求和运算的综合偏好
- Model Optimization:
  - 应用交叉熵损失函数来优化所有参数:
    - $J = -\sum_{u \in U} \sum_{i \in H(u)} \sum_{j=1}^{N} y_{i,j}^u log(\hat{y}_{i,j}^u) + \lambda ||\Theta||_2$

(5) 实验部分:

- 数据集:
  - Foursquare：Singapore;August 2010 to July 2011
  - Gowalla：California,Nevada;February 2009 to October 2010
- 评价指标:
  - Recall@k;$k \in 2, 5, 10$
  - NDCG@k;$k \in 2, 5, 10$
- Results: 图 8

Table 2: Performance comparison with baselines. Bold scores are the best results for each metric, while the second best scores are underlined. ∗ represents significance level p-value< 0.01 of comparing DRAN with the best baseline.

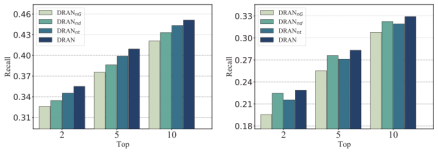
	Foursquare						Gowalla					
	Recall@2	Recall@5	Recall@10	NDCG@2	NDCG@5	NDCG@10	Recall@2	Recall@5	Recall@10	NDCG@2	NDCG@5	NDCG@10
MF	0.1457	0.1909	0.2356	0.1223	0.1484	0.1632	0.0975	0.1404	0.1778	0.0821	0.1013	0.1170
FPMC	0.1789	0.2492	0.2966	0.1553	0.1865	0.2174	0.1021	0.1489	0.1863	0.0904	0.1072	0.1216
TMCA	0.2042	0.2761	0.3325	0.1857	0.2192	0.2453	0.1284	0.1893	0.2401	0.1074	0.1332	0.1563
STGN	0.2002	0.2684	0.3261	0.1865	0.2132	0.2562	0.1178	0.1814	0.2365	0.1057	0.1378	0.1529
LSTPM	0.2364	0.2966	0.3609	0.2289	0.2404	0.2611	0.1452	0.2043	0.2576	0.1255	0.1532	0.1811
SGRec	0.2964	0.3511	0.3975	0.2809	0.3053	0.3205	0.2122	0.2666	0.3127	0.198	0.2224	0.2375
STAN	0.3257	0.3730	0.4204	0.3121	0.3367	0.3526	0.1873	0.2461	0.2973	0.1711	0.1977	0.2154
DRAN	**0.3551**∗	**0.4092**∗	**0.4512**∗	**0.3389**∗	**0.3631**∗	**0.3775**∗	**0.2288**∗	**0.2832**∗	**0.3291**∗	**0.2145**∗	**0.2384**∗	**0.2535**∗
Improvement	9.02%	9.70%	7.32%	8.58%	7.84%	7.06%	7.82%	6.22%	5.24%	8.33%	7.19%	6.73%

图 8:

  –

- 消融实验:
  - $DRAN_{nG}$：移除整个 DGCN 层并通过随机初始化生成 POI 表示
  - $DRAN_{nd}$：去除了基于距离的关系图，只考虑了基于转移的影响。

Figure 3: Performance comparison of different variants

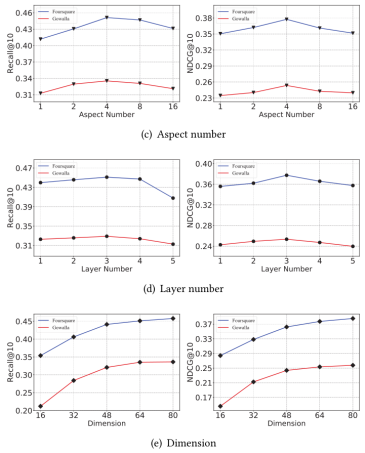


(a) Foursquare (b) Gowalla

图 9:

- $DRAN_{nt}$：这个变体去除了基于转移的关系图，只考虑了基于距离的影响
- 图 9

- Hyperparameter Study:
  - Effect of Component Number: 验证显式建模 POI 的不同潜在方面是否可以提高性能
  - Effect of Representation Dimension: 扩大纬度 d 不仅使解开的表示更具信息性，而且还提高了组件的表现力
  - Effect of DGCN Layer Number: 堆叠更多的 DGCN 层能够聚合来自目标节点的高阶邻居的有用信息。特别是，增加 DGCN 的深度有助于捕捉相关 POI 之间的复杂相关性。而连续堆叠层提供的改进并不明显，表明冗余堆叠可能会在学习过程中引入噪声并导致过度平滑问题
  - 图 10



(c) Aspect number

(d) Layer number

(e) Dimension

Figure 4: Effect of several key hyperparameter

图 10:

- Effect of Propagation Rule：
  - 用三种基于图卷积的方法替换 DGCN，分别包括 GCN、LightGCN 和 CIGCN, 所有这些变体都出现了不同程度的性能下降

Table 3: Performance comparison of different GCN variants

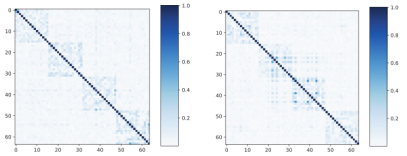
	Foursquare		Gowalla	
	Recall@10	NDCG@10	Recall@10	NDCG@10
GCN+	0.4158	0.3525	0.3128	0.2346
LightGCN+	0.4248	0.3457	0.3177	0.2447
CIGCN+ (4 heads)	0.4518	0.3783	0.3077	0.2366
CIGCN+ (64 heads)	0.2886	0.2128	0.1531	0.1089
DRAN	0.4512	0.3775	0.3291	0.2535

图 11:

- 图 11
- DGCN 可以看作是上述方法的泛化，它可以有效地学习解开的表示以提高性能，并且更适合顺序推荐

- Visualization:
  - DRAN 有效地学习了带有 DGCN 的 POI 的解纠缠表示 X_{fin}，表明模型在区分解耦表示的潜在分量方面具有很强的能力
  - 图 12

Figure 5: Correlation values of representation dimensions



(a) Foursquare (b) Gowalla

图 12:

3.Empowering Next POI Recommendation with Multi-Relational Modeling

(1) 研究内容：考虑关系异质性 (relational heterogeneities) 和用户与 POI 之间的随着时间的推移相互影响从而进行 next-POI 推荐

(2) 文章整体要点：

- 提出了一个新的框架 MEMO，有效地利用了多网络表示学习模块的异构关系，并明确地将跨时的 uesr-POI 的相互影响与耦合的递归神经网络结合起来。
- 目前关于 LBSN 的 POI 推荐存在的问题:
  - 社会关系的异质性 (由于用户的偏好经常与不同类型的社会关系有着密切的联系，故而不同的社会关系对用户的影响是不同的)，再加上用户对 POI 的访问形成的 uesr-POI 关系，给有效利用嵌入的丰富信息带来了巨大挑战
  - 捕捉用户和兴趣点之间关键的、跨时空的相互影响仍然具有挑战性 (用户的偏好和 POI 的潜在状态（如声誉）会随着时间的推移相互影响)
- 关系模型（MEMO）：

  - 开发了一个基于多图卷积网络（GCNs）和 self-attention 的多关系建模模块，从而利用不同的 user-user 社会关系和 user-POI 关系 (异构关系)
    * 将每一种关系映射到相应的网络,以学习特定的关系表征;然后采用 self-attention 机制，从不同的关系类型中聚合用户表征
  - 设计了一个基于耦合的递归神经网络（RNN）的 user-POI 相互影响建模组件相互更新对方的表征, 从而捕捉用户和 POI 之间随时间推移的相互影响
- 本文的贡献:
  - 模型考虑了关键的、但被忽视的异质关系，特别是用户之间的异质社会关系，以及用户与 POI 之间的跨时空相互影响
  - 提出了一个新的框架 MEMO，通过多网络表征学习模块纳入异质关系，并通过耦合的 RNNs 捕捉用户与 POI 之间的相互影响

(3) 问题定义:

- 给定大小为 U 的用户集合$\mathcal{U} = \{u_1, u_2, ..., u_U\}$,和大小为 L 的 POI 集合 $\mathcal{L} = \{l_1, l_2, ..., l_L\}$。每个 POI　l 都与表示其地理位置的地理坐标 (longitude, latitude) 相关联
- Trajectory: 每个用户 u_i 的轨迹是一个时序序列: $C(u_i) = c_1(u_i), c_2(u_i, ..., c_K(u_i))$ 每个元素 $c_K(u_i)$ 都可以用一个元组 l_k, t_k 表示，其中 l_k 是用户 u 在时间戳 t_k 处访问的 POI 的地理坐标
- User-User Social Relations:$R = r_1, r_2, ..., r_p$ 表示用户之间的一组社会关系 P。我们将用户-用户社交关系定义为三元组 (u_i, u_j, r_p)，表示用户 u_i 和 u_j 具有 r_p 类型的社交关系
- Next POI Recommendation: 在每个时间戳 t，next-POI 推荐任务将每个用户从时间戳 1 到 t 的轨迹和不同类型的社会关系作为输入，并为每个用户推荐一个在 t+1 时间戳预测分数最高的合适 POI 列表

(4) MEMO:

- 整体架构:
  - 由两个主要组件组成:
    * 一个关系建模模块，用于有效利用异构关系
    * 一个 user-POI 相互影响建模模块，用于捕获用户和 POI 之间随着时间的推移的相互影响
  - 图 13
- Relation Modeling:
  - MEMO 利用异构关系捕捉每个用户对 next-POI 的偏好，包括从 LBSN 收集的多个 uesr-user social 关系和 user-POI 关系
  - 为了对用户集 U 之间的 P 种不同类型的 uesr-user social 关系进行建模，使用了 P 的网络 $G_1, ..., G_P$，每个网络都有其对应的对称邻接矩阵 $A_1, ..., A_P$
  - user-POI 网络 G_C 对应于一个邻接矩阵 $A_C \in R^{(U+L)\times(U+L)}$，该矩阵捕获用户是否访问过特定 POI

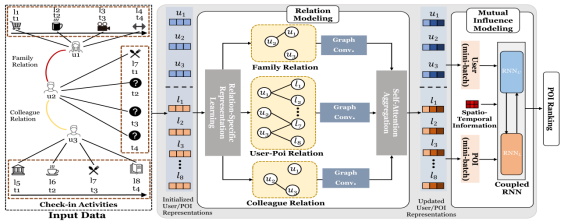


Figure 2: Overview of *MEMO* for next POI recommendation.

图 13:

- 总共有 P + 1 种关系，包括 P 种类型的 user-user social 关系和 1 种 user-POI 关系
- Relation-Specific Representation Learning: 利用特定于关系的表示学习模块将节点映射到分别对应于每个关系的潜在表示空间, 从而适应不同类型的关系
  - 对于每个节点 v_i，随机初始化一个通用嵌入 x_i
  - 使用关系特定的转换函数 $\Phi_p(\cdot)$，将每个 x_i 映射到一个新的嵌入 x_i^p 对应于任何 $p \in [P+1] : x_i^p = \Phi_p(x_i)$ 的第 p 个关系类型
  - 基于 x_i^p，通过聚合每个网络 G_p 上的邻层的嵌入来学习每个节点 v_i 的关系特定表示 h_i^p，这是由具有注意机制的 GCN 层实现的。
- Aggregation over Different Relation Types:
  - 不应简单地假设同质关系或节点表示跨相同潜在空间内的特定关系网络。本文利用 self-attention 机制将每个节点的所有特定于关系的表示聚合到一个公共潜在空间中, 从而能够有效地捕捉嵌入在不同类型关系中的每个用户的偏好
  - 给定每个节点 v_i,以及它在不同网络中的所有表示,将对应于第 p 个 ($p \in [P+1]$) 关系类型 h_i^p 的每个表示映射到密钥向量 $k_i^p = W_p^K h_i^p$，查询向量 $q_i^p = W_p^Q h_i^p$ 和消息向量 $m_i^p = W_p^M h_i^p$
  - 计算每对关系类型 (p_1, p_2) 之间的注意力权重，其中第 1 个查询向量和第 2 个关键向量之间的相似性：$\alpha(p_1, p_2) = Softmax_{\forall p_2 \in [P+1]}(k_i^{p_2} q_i^{p_1} / \sqrt{d})$
  - 注意，与假设输入表示具有相同潜在空间的普通 attention 机制不同，本文框架中的上述线性投影是特定于关系的，因此允许聚合来自不同潜在空间的特定于关系的表示
  - 将来自不同关系类型的表示聚合为 $h_i = MLP([\tilde{h}_i^1 || \tilde{h}_i^2 || ... || \tilde{h}_i^{P+1}])$, 其中 $\tilde{h}_i^p = \bigoplus_{p' \in [P+1]} \{\alpha(p, p') \cdot m_i^{p'}\}$

- User-POI Mutual Influence Modeling:
  - 在 user-POI 关系中，用户和兴趣点的潜在状态可能会随着时间的推移相互影响。因此，需要更新用户和 POI 的表示来捕捉这种相互影响
  - 用户 RNN (RNN_U) 和位置 RNN (RNN_L) 组成的耦合 RNN(RNN_S)，以分别学习每个时间戳的用户和 POI 表示
  - RNN_U 结合 POI 的表示来更新用户的表示, 假设用户 u 在 C(u) 中的时间戳 t 访问了 POI l，用户 u 在时间戳 t + 1 的表示受到位置 l 在时间戳 t 的表示的影响，形式化为：$h_u^{t+1} = \sigma(W_1^U h_u^t + W_2^U h_l^t + W_3^U z^{\Delta t} + W_4^U z^{\Delta d})$

  - RNN_L 还利用用户的表示来更新每个 POI　l 在下一个时间戳 h_l^{t+1} 的表示。具体来说，将在时间戳 t 访问 l 的用户列表表示为 U(l,t)，并将他们的表示列表表示为 $h^t_{U(l,t)}$。POI　l 的表示更新为：$h_l^{t+1} = RNN_L(h_l^t, h^t_{U(l,t)})$。在每个时间戳 t，POI l 的表示被递归更新 |U (l,t)| 按用户访问时间顺序排列。为了在保持时间依赖性的同时并行更新用户兴趣点表示，通过从用户轨迹中选择独立的用户兴趣点访问来创建每个小批量，以确保在同一批次中处理的每两个访问不共享任何普通用户或兴趣点
  - Spatio-Temporal Representation:
    * 结合了时空信息来促进 next-POI 推荐
    * 设计了一种机制，通过利用用户连续访问的每对 POI 之间的空间转换（距离）和时间转换（访问时间间隔）来有效地整合这种时空信息。
    * 具体来说，引入了两个可学习的向量 t_s 和 t_l 来分别表示短时和长时转换。然后正式获得时间间隔的表示为：$z^{\Delta t} = t * t + tanh(t)$，其中如果时间转换短于预设阈值 θ_t ，则 t = t_s ，否则 t = t_l 。
    * 类似地，获得了具有空间阈值 θ_d 的空间间隔 $z^{\Delta d}$ 的表示。两个 POI 之间的距离由 Haversine 距离形成。在更新每个用户 u 的表示 h_u^t 和每个 POI　l 的表示 h_l^t 之后，通过选择具有最高预测分数的 POI 来推荐用户 u 在下一个时间戳 t + 1 访问的 POI 列表连接层
    * 将 next-POI 的预测建模为一个多分类问题，其中基本事实是每个用户在下一个时间戳实际访问的 POI，并采用交叉熵损失进行模型优化。

(5) 实验部分：

- 数据集：
  - Baltimore (B’ more) July
  - DC July
  - DC August
  - 注意，原有数据集没有对用户关系的区分，本文的数据集将用户的家和工作地点分别检测为工作日夜间和白天最频繁的停止点；并由此推断出他们的主要社会关系：家庭关系和同事关系
- 评价指标：
  - Recall@K;K=10
  - MRR@K;K=10
- Results:
  - 图 14
  - MEMO 优点发析：
    * MEMO 通过整合从不同关系中学习到的用户表示，更全面地捕捉用户的偏好
    * MEMO 对 user-POI 随时间的相互影响进行建模，学习的表示可以更好地捕捉用户/兴趣点的潜在状态

Table 3: Performance of next POI recommendation for different methods.

	Baltimore July		DC July		DC August	
	Recall@10	MRR@10	Recall@10	MRR@10	Recall@10	MRR@10
Markov	0.257 ± 0.001	0.088 ± 0.004	0.471 ± 0.012	0.224 ± 0.012	0.431 ± 0.013	0.224 ± 0.012
LightGCN	0.650 ± 0.044	0.392 ± 0.011	0.609 ± 0.021	0.291 ± 0.034	0.622 ± 0.019	0.311 ± 0.014
GRU4Rec	0.657 ± 0.009	0.384 ± 0.006	0.617 ± 0.019	0.300 ± 0.013	0.624 ± 0.005	0.306 ± 0.005
DeepMove	0.715 ± 0.031	0.402 ± 0.003	0.660 ± 0.007	0.333 ± 0.006	0.648 ± 0.014	0.331 ± 0.009
SASRec	0.738 ± 0.029	0.415 ± 0.003	0.694 ± 0.019	0.335 ± 0.006	0.687 ± 0.006	0.353 ± 0.007
TiSASRec	0.836 ± 0.002	0.420 ± 0.007	0.760 ± 0.011	0.353 ± 0.014	0.763 ± 0.006	0.379 ± 0.003
STAN	0.868 ± 0.003	0.426 ± 0.004	0.807 ± 0.005	0.360 ± 0.007	0.816 ± 0.007	0.426 ± 0.004
MEMO	**0.891±0.004**	**0.446±0.009**	**0.831±0.002**	**0.380±0.006**	**0.840±0.003**	**0.435±0.008**

图 14:

- Ablation Studies:
  - MEMO-NG，其中删除了关系建模组件
  - MEMO-NA，其中自注意力机制被基于双线性函数的普通注意力机制所取代
  - MEMO-NR，其中去除了相互影响组件，即仅从关系建模组件中学习表示
  - MEMO-NTS，其中不使用时空表示
  - 图 15

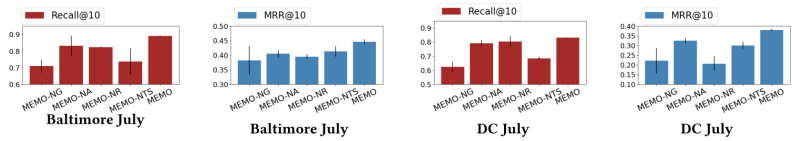


Figure 3: Ablation studies.

图 15:

- Parameter Studies: 图 16

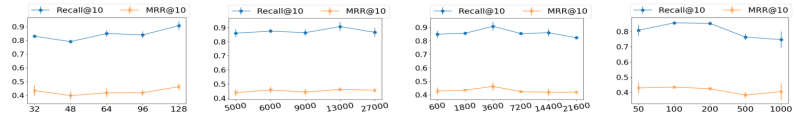


Figure 4: Parameter studies.

图 16: