# 学习周报

9.3-9.9

**1. 已完成的工作**

(1) 本周的任务依然是文章精读和代码运行:

- 目前为止 2022 年所推荐的 POI 论文已有的代码都尝试运行了一遍，并将其中有代码且提供对应数据的代码进行了运行 (不得不说是真的少)
- 本周依然是阅读了三篇文章，目前来说感觉在上课阶段的阅读速度大概在一周三篇左右 (不过感觉考试月时的阅读速度要大大减慢了)
- 目前论文阅读大概是三步:
  - 先认真精读一遍全文，包括模型的每一步和具体的实验操作
  - 结合实验研究模型
  - 结合论文和笔记最后阅读一遍
- 一些想法:
  - 目前来看大部分的 POI 推荐问题中，已访问过的 POI 对 next-POI 访问都具有相同的影响力，是否可以对已访问的 POI 添加一个时间和空间的衰减函数，从而降低古老 POI 对现有 POI 点的影响力 (类似游戏中的排名，如果长期不继续游戏会随时间增长所衰减，同时每次新的游戏记录都会为排名做出贡献)

(2) Time-sensitive POI Recommendation by Tensor Completion with Side Information

- Embedding Initialization
  - 这里的重点主要是初始化，在张量分解中使用初始化从而实现快速收敛
  - 这里使用 spectral method，用于获得初始用户、POI 和时间单位嵌入的初始值的粗略估计
  - 这里的初始化以加速收敛的思想可以学习
- Tensor Factorization Formulation
  - 通过使用相应的嵌入向量计算每个条目 $\chi_{i,j,k}$ 的值，对用户、POI 和时间间隔之间的三元交互进行建模
  - 首先计算一个对于给定用户 i、POI j 和时间间隔 k 的嵌入向量 $U_i^1$ 、$U_j^2$ 和 $U_k^3$ 的元素乘积的向量 $\Phi$

  - 将向量 $\Phi$ 通过参数 h 的密集层以预测 $\chi_{i,j,k}$ 的值
  - 为了优化嵌入 $U^1$、$U^2$、$U^3$ 和参数 h, 设计了一个具有两个损失头的混合损失函数:
    * 根据观察到的条目计算的预测误差头
    * 一种新颖的社交 Hausdorff 距离函数，用于测量朋友的 POI 子集彼此之间的距离

- Social Hausdorff Distance
  - 根据社会同质性理论，社交网络倾向于形成具有相似属性或兴趣的节点集群
  - 托布勒的第一地理定律指出，近处的事物比远处的事物更相关
  - 基于上述理论本文认为如果一个 POI 被该用户的朋友访问过，或者靠近该用户的朋友访问过的 POI，那么向该用户推荐该 POI 是很自然的
  - Hausdorff 距离是衡量两个点集彼此相距多远的度量。这里计算用户 $v_i$ 将访问的潜在 POI, 和 $v_i$ 的朋友访问过的 POI 集 Hausdoff 距离，通过最小化两个点集之间的 AHD 函数从而推荐更符合的 POI

- Diversifying Recommendation by Location Entropy
  - 这个模块是为了保证所推荐的 POI 的新颖性–即访问尚未被很多人访问过的推荐 POI
  - 使用位置熵的概念来衡量 POI 的受欢迎程度

- Learning with Whole Data
  - 由于负采样的性能对采样策略和负样本数量高度敏感。在本文中，选择在整个数据上而不是通过负采样来训练模型
  - 传统的负采样是一种有偏差的近似，通常不会收敛到与从所有条目计算 的相同损失，由于对采样策略的敏感性，因此难以收敛到最佳排名
  - 这里如果采用原有的负采样方案的方法，会由于大量负样本而导致的计算成本过高，这里采用简化的方法进行计算 (主要是对计算结果影响较小的项数进行消除)

- Loss Function
  - 将社交空间损失头 $L_1$ 和最小二乘损失头 $L_2$ 组合成最终的损失函数 L 来训练模型

(3) Modeling Spatio-temporal Neighbourhood for Personalized Point-of-interest Recommendation

- 本文聚合用户与 POI 模型，从而动态捕捉用户对 POI 的个人和动态偏好而非仅考虑 POI 之间的关系
- 本文在对 POI 进行推荐时主要从三个方面进行考虑，用户对 POI 的短期和长期偏好 (即不仅仅考虑用户更偏好的 POI, 同时考虑不同的时间段对 POI 偏好的影响)，以及不同用户对时间/距离的接受程度 (是否能接受相对更远的 POI)
- 本文是基于图结构从而提取相应的 POI 特征
- Neighbour Aggregation Layer

  - 本层包含了 User Neighbour Aggregation Layer 和 Location Neighbour Aggregation Layer，基于 TKG 的时间信息，将每个时间戳的用户和位置实体各自的邻域与构建的用户/位置接收域进行聚合
  - 个人认为该层的目的是为了寻找并构建每个 user/POI 的 user/POI 相似集
- User Neighbour Aggregation Layer
  - 该层用于聚合 TKG 上的 user neighbour 实体以进行用户表示
  - $e_u$ 的用户邻居实体 $e_{u'}$ 为在时间 t 之前与 $e_u$ 共享相同位置实体的实体, 这里的感受野是为了更好的提取 user 特征的相似性
  - 分别计算 K 个相邻实体与目标用户的时间相关性与空间相关性，从而计算时空分数 $\lambda_{st}^{u,u'}$
  - 从而获得所需的 K 个用户邻居实体
  - 个人理解这一部分是将 user 使用图卷积操作提取特征从而寻找每个用户的相似 user neighbour
- Location Neighbour Aggregation Layer
  - 该层的整体操作与 User Neighbour Aggregation Layer 类似，区别在于 location neighbour 将由用户和相应 user neighbour 的喜好所决定
  - 个人认为这里的 location neighbour 的定义是否可以进行扩展，及不仅仅将用户所喜好的 POI 定义为 location neighbour，同时可以根据用户的访问路线将 user neighbour 所顺序访问的一系列 POI 同样定义为 location neighbour
- Spatial-Temporal Interval Aware Attention Layer
  - 通过注意力机制将用户/位置的表示与时空间隔相结合
  - 将空间距离和时间间隔测量为时空间隔的注意层，并分配权重组合内部两者的相对比例用户和用户之间的表示
  - 该层主要是用 self-attention 将获得的时空间隔注入到用户/位置表示中，将时间嵌入作为注意力输入结合到表示中，最后由用户和位置表示的叉积计算获得访问概率
  - 个人理解来说该层是将 Neighbour Aggregation Layer 与 Location Neighbour Aggregation Layer 这两层的信息聚合求出所需的访问概率
- Neural Network Training
  - 这部分使用交叉熵作为损失函数，通过计算 $N_t^u, L_t^u, v_t^u, e_t^u$ 得到访问概率 $\hat{y}_t^u$ 从而使神经网络最终收敛

(4) Next Point-of-Interest Recommendation with Auto-Correlation Enhanced Multi-Modal Transformer Network

- 本文希望通过深度挖掘 POI 序列中的关系 (POI 中的依赖关系痕迹;POI 序列中的分层性质和子序列的匹配; POI 的两种模态与密度类别之间的相互作用) 从而更好的推荐 POI，即没有考虑 user-POI，user-user 之间的关系
- RNN 与 Transformer：

  - RNN:
    * 顺序处理：句子必须逐字处理
    * RNN 指的是一个序列当前的输出与之前的输出也有关，具体的表现形式为网络会对前面的信息进行记忆，保存在网络的内部状态中，并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包含输入层的输出还包含上一时刻隐藏层的输出
    * 它采取线性序列结构不断从前往后收集输入信息
  - Transformer:
    * 非顺序处理：句子是整体处理，而不是逐字处理
    * 单个的 Transformer Block 主要由两部分组成：多头注意力机制 (Multi-Head Attention) 和前馈神经网络 (Feed Forward)，Transformer Block 代替了 LSTM 和 CNN 结构作为了我们的特征提取器，使得 Transformer 不依赖于过去的隐藏状态来捕获对先前单词的依赖性，而是整体上处理一个句子，以便允许并行计算，减少训练时间，并减少由于长期依赖性而导致的性能下降

- Embedding Layer
  - 嵌入层用于将用户、类别、位置和时间信息编码为潜在表示
  - 个人认为嵌入层的作用是把离散的数据转换到连续的可以被连续表达的空间上，把稀疏高维的东西变到了稠密低维的上面
  - 就本文而言，这里的嵌入层与其他论文的嵌入层没有什么太大的差异
- Auto-Correlation Layer
  - 以前的工作主要基于循环结构，总是忽略不连续访问的信息
  - self-attention 可以捕获序列内的所有点对点交互，但无法在子序列级别提取相关性
  - auto-Correlation 通过计算子序列的自相关来发现子序列的依赖关系，并通过时间延迟聚合来聚合相似的子序列
  - 在消融实验中，MTN 用普通的 self-attention 取代了模型中的自相关，模型的性能下降，证明了子序列的匹配对于下一个 POI 任务至关重要。
  - 这里的 self-attention 的一些衍生：
    * Self-Attention 允许对依赖关系建模，而不需要考虑它们在输入或输出序列中的距离，并且可以将一个序列的不同位置串联起来
    * 向量的内积的几何意义：
      · 表征两个向量的夹角，即一个向量在另一个向量上的投影
      · 投影值大说明两个向量的相关度高
    * 注意力核心是算权重系数，权重系数的计算套路主要为：
      · 计算相似度
      · 用 softmax 函数归一化出权重系数
    * Self-Attention 的作用就是全局关联权重，然后做输入的加权和:
      · $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$

    - Q,K,V 的计算
        - Q,K,V 是 q,k,v 的矩阵形式, 而在 self-attention 中，q、k、v 都是输入参数矩阵变换而来的 (增加可学习性)
        - 其中 q 和 k 是算相似度的得权重，v 是用来跟权重做加权求和
    - Multi-head Attention
        - 例如 $a_1, a_2, a_3, a_4$ 四个输入向量，长度是 512；上一节是一个头；
        - 逻辑：多个头就是将四个输入向量的长度一分为 N 段，各自对应的部分做 self-attention，每个向量再聚合分段的结果
        - 权重计算：分完段，各自算，再汇总，即 $head_1$ 的 qkv 算 $head_1$ 的，头与头之间在计算相似度时互不打扰，每个段独立算完，再聚合
        - 多头的目的: 多样性, 多个头就有多个学习空间，学习的东西会更多，而不会产生偏移
        - 能让每个注意力机制通过 QKV 映射到不同的空间去学习特征，去优化每个词汇的不同特征部分，从而均衡同一种注意力机制可能产生的偏差，让词义拥有来自更多元的表达，实验表明可以从而提升模型效果
- Cross-Modal Auto-Correlation Layer
    - 设计了跨模态自相关，使一种模态能够从另一种模态接收信息，即 POI 和类别序列相互获取辅助信息
    - 个人认为这个模块将 POI 和类别序列有效的结合其来，后续可以看看这个跨模态模块是否适应于其他类别的信息的相互融合
- Attention Predictor
    - 分别计算候选 POI 和类别的概率，注意力预测器分别预测 $\hat{Y}^P$ 和 $\hat{Y}^C$
    - 最后应用交叉熵损失函数来量化预测值和 ground truth 的差异
- 个人感觉本文的创新点主要在
    - auto-Correlation 所达到的子序列级别的关系提取和对类别序列的信息挖掘

**2. 存在的问题**

1. 目前感觉对相应的论文阅读的积累还是不够，较难理解住论文中的一些公式和推论
2. 感觉在目前的文章阅读中文献笔记和学习周报的内容有一些重复，打算在下周的学习中予以改善
3. 感觉目前的文章阅读似乎有点过于关注次重点 (不是文章的重点但又有一定的作用)

**3. 下一步的计划**

1. 打算按 POI 推荐的列表继续阅读论文 (打算在这周把 2022 年 POI 推荐论文精读完)
2. 有点想研究一下 Github 的个人主页 (如果有时间的话)