上机任务-week6

任务背景

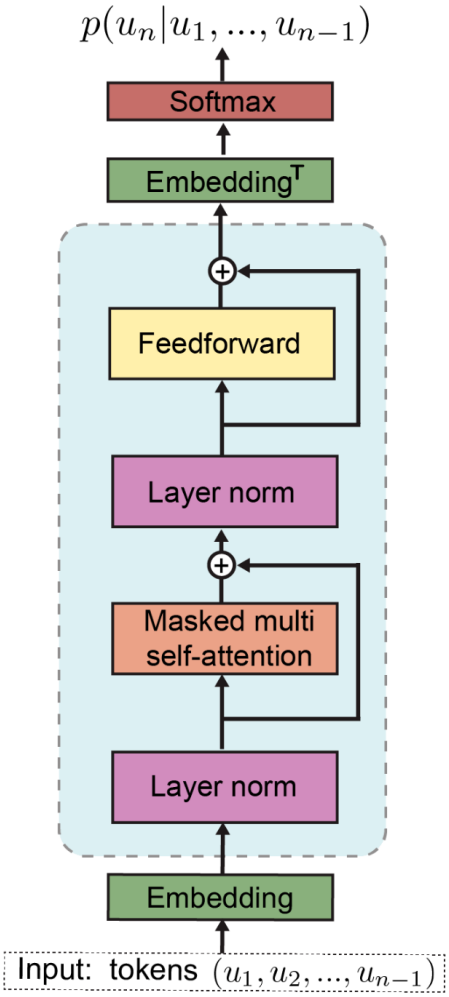
在实际生活中，用户连续行为之间具有明显的联系和因果关系。我们可以将用户以往的一系列互动行为视为行为序列 $u = \{i_1, i_2 \ldots, i_n\}$，并通过构建模型对其建模，来预测下一时刻用户最感兴趣的内容 i_{n+1}，这就是序列化推荐（Sequential Recommendation）的核心思想。目前序列化推荐的常用模型包括RNN、CNN、GNN以及Transformer。**本次实验尝试将Transformer的多头自注意力机制替换为全连接神经网络完成序列预测任务。**

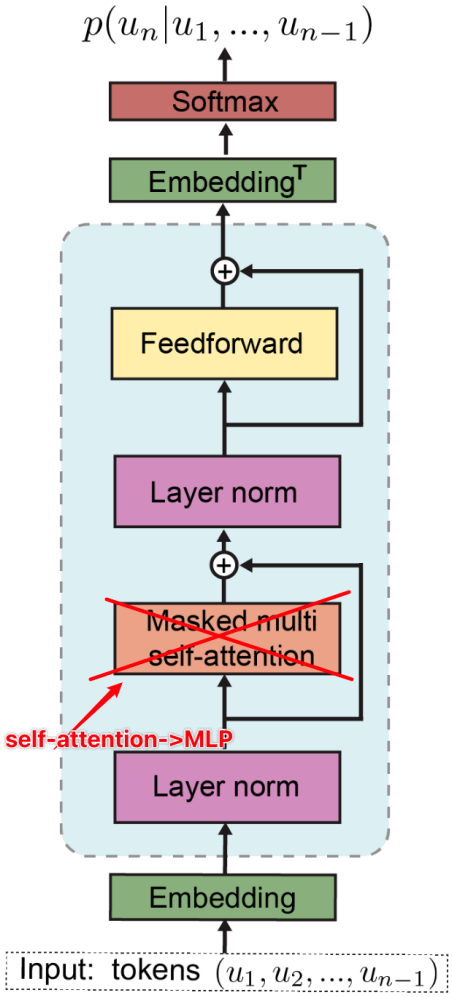
数据集介绍

Amazon review数据集[1] 涵盖了若干个领域（例如，家居用品、美容产品、汽车、电子产品等），并包括了用户对这些产品的评分、评价和其他元数据。本次实验所使用的数据集是其子类之一 `Amazon Beauty`，数据格式已经过预处理，其中每行包含一个 `user id` 和 `item id`（从 1 开始），表示交互（按时间戳排序），具体处理细节可参考[2, 3]，相关代码已在实验中给出。

模型架构

- 模型结构可以参考下图，将transformer中的自注意力层替换成全连接神经网络
- 注意力层的替换势必会导致模型预测性能下降，但同时也使得模型计算得到加速，因此本次实验的重点在于如何在保持计算高效性的同时提高模型性能。关于这一点可以**自行探索**或者**尝试复现**[4]的做法，即利用滤波器对数据进行前处理后再送入MLP。





代码说明

- data/: 存放数据集 `Amazon Beauty`
- output/: 存放模型训练日志文件
- main.py: 主函数
- models.py: 模型文件
- modules.py: 模型组件
- trainers.py: 训练文件

课堂任务

1. 完成对Transformer Encoder架构的改造，将多头注意力层替换成MLP，实现对所给序列数据的预测任务，可以自行探索提升性能的方案，也可以参考所给文献的方案
2. 训练过程中涉及的评价指标有：['HIT@1', 'NDCG@1', 'HIT@5', 'NDCG@5', 'HIT@10', 'NDCG@10', 'MRR']，相关计算函数和日志保存方式在代码中已给出，将模型训练日志文件保存下来并提交（提交一份）
3. 提交补全后的代码以及实验报告

参考文献

[1] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In SIGIR 2015. 43–52.

[2] https://github.com/kang205/SASRec

[3] W.-C. Kang and J. J. McAuley. 2018. Self-Attentive Sequential Recommendation. In ICDM 2018. 197–206.

[4] Rao, Y., Zhao, W., Zhu, Z., Lu, J., & Zhou, J. 2021. Global filter networks for image classification. In NIPS 2021, *34*, 980-993.

Tip:

数据预处理已经做好了，代码中目前还缺少模型的搭建，完善即可

Encoder结构仅供参考...

```
Encoder(
  (layer): ModuleList(
    (0): Layer(
      (filterlayer): FilterLayer(
        (out_dropout): Dropout(p=0.5, inplace=False)
        (LayerNorm): LayerNorm()
      )
      (intermediate): Intermediate(
        (dense_1): Linear(in_features=64, out_features=256, bias=True)
        (dense_2): Linear(in_features=256, out_features=64, bias=True)
        (LayerNorm): LayerNorm()
        (dropout): Dropout(p=0.5, inplace=False)
      )
    )
    (1): Layer(
      (filterlayer): FilterLayer(
        (out_dropout): Dropout(p=0.5, inplace=False)
        (LayerNorm): LayerNorm()
      )
      (intermediate): Intermediate(
        (dense_1): Linear(in_features=64, out_features=256, bias=True)
        (dense_2): Linear(in_features=256, out_features=64, bias=True)
        (LayerNorm): LayerNorm()
        (dropout): Dropout(p=0.5, inplace=False)
      )
    )
  )
)
```