web信息处理实验三报告

实验组成员：

- 邓一川，学号PB19000050
- 赖铮岩，学号PB19000061

实验要求

本次实验要求基于所提供的数据集，对于每个用户进行音乐的推荐。

算法思路设计

这里我们采用了**基于物品的协同过滤推荐**，考虑到逐步计算的计算量过大，我们想要使用`numpy`的高效算力，所以基于原始算法进行了一些线性代数的分析，将所有计算转化为矩阵乘法，其大致思想如下：

- 首先我们读入初始数据，对其进行处理得到矩阵$\mathbf{R}\in\mathbb{R}^{m\times n}$，其中$m=23599$为用户数目，$n=21602$为物品数目，该矩阵的第$(i,j)$个元素为第$i$个用户对第$j$个物品的评分，以`scipy.sparse.coo_matrix`的格式存储，并存于本地文件，命名为`raw.npz`。
- 同时我们针对上述矩阵进行一定的规范处理，即，忽略掉所有没有评分的交互，同时对每一个物品，将其每一个评分结果减去平均得分作为对应项，令该矩阵为$\mathbf{R}'\in\mathbb{R}^{m\times n}$，即，对于任意$i,j$，有

  $$r'_{i,j}=r_{i,j}-\frac{\sum_i r_{i,j}}{c_j}$$

  其中c_j为第j个物品获得的评分数目。我们同样以`scipy.sparse.coo_matrix`的格式存储$\mathbf{R}'$，并存于本地文件，命名为`normalized.npz`。
- 这里我们在计算过程中得到了每一个物品获得的平均分，我们以`ndarray`的格式存储，并存于本地，命名为`item_avrg.npy`。
- 之后我们计算物品间的相似度矩阵，即，该矩阵的第(i,j)个元素为第i个和第j个物品的相似度，相似度的定义为

  $$sim(a,b)=\frac{\sum_{u\in usr(U)}(r_{a,u}-\overline{r_a})(r_{b,u}-\overline{r_b})}{\sqrt{\sum_{u\in usr(U)}(r_{a,u}-\overline{r_a})^2}\sqrt{\sum_{u\in usr(U)}(r_{b,u}-\overline{r_b})^2}}$$

  我们这里为了加速，使用`numpy`进行矩阵运算，令相似度矩阵为$\mathbf{S}$，那么其可以被表示为

  $$\mathbf{S}=\mathbf{R}''^T\mathbf{R}''$$

  这里$\mathbf{R}''$为对$\mathbf{R}'$的每一列进行l_2-norm的归一化得到的矩阵。
- 得到相似度矩阵后，我们可以进行预测：给出用户未评分项目的分数预测值，这里仍然使用`numpy`进行矩阵运算：

  我们预测分数的定义为：

  $$pred(a,p)=\frac{\sum_{q!=p}sim(p,q)\cdot r_{a,q}}{\sum_{q!=p}sim(p,q)}$$

  所以可以写成矩阵形式如下，其中$\mathbf{S}'$为$\mathbf{S}$对每一行进行l_1-norm的归一化得到的矩阵：

  $$\mathbf{P}=\mathbf{S}'\mathbf{R}^T$$

- 最后，给出矩阵$\mathbf{P}$中每个用户预测评分最高的100个物品作为解答

算法实现

如上述算法，我们的算法实现大致如下：

- 数据预处理部分：

  我们首先调用函数`data_loader()`导入所需的实验数据，并且转为`coo_matrix`格式，以此稀疏矩阵的形式存于内存和文件中，文件命名在上述算法中提及。

  然后使用`normalized(raw_matrix)`来对上述原始数据进行了基本的处理，即得到了算法描述中提到的矩阵$\mathbf{R}'$。

- 计算相似度矩阵：

  我们首先读入之前所计算得到的预处理矩阵`normalized_matrix`，即$\mathbf{R}'$，然后进入函数`compute_item_sim(normalized_matrix)`。首先我们对其每一列进行l_2-norm的归一化，得到了矩阵$\mathbf{R}''$，然后再进行矩阵的乘法，最终得到了相似度矩阵$\mathbf{S}$，因为其是稠密的，我们只能以`ndarray`的形式存储。

- 计算预测分数：

  进入函数`compute_pred()`，调用上述过程得到的相似度矩阵$\mathbf{S}$，以及原始评分矩阵$\mathbf{R}$，最终我们得到了预测评分矩阵$\mathbf{P}$

- 给出推荐结果：

  进入函数`predict()`，我们依据矩阵$\mathbf{P}$，给出每个用户预测评分最高的100个物品作为解答，导出为`answer.txt`

算法优化

如上所说，我们一开始并未考虑使用高效的`numpy`矩阵运算，而是人工手动进行矩阵乘法的循环，但后来我们逐渐发现这样的时间成本是我们没法承担的（接近10个小时），于是我们对于原始算法进行了线性代数的分析，并给出了上述在可接受的时间内得到结果解决方案。

最优结果截图