电子商务系统结构
实验三（重交）

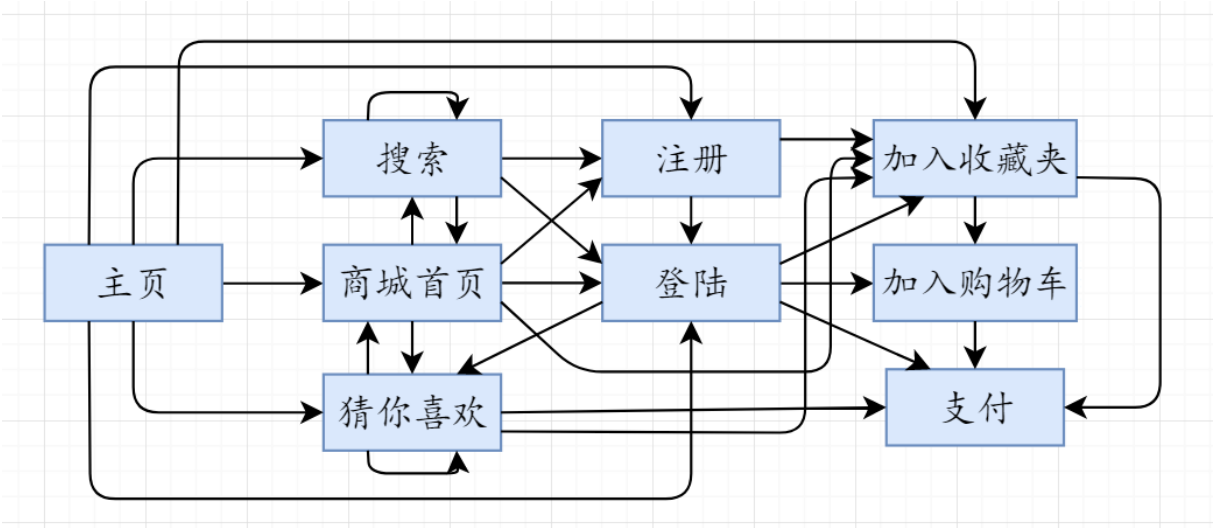
蒋仕彪
3170102587
叶梓成
3160103785

November 8, 2019

1　电子商务网站用户行为建模

1.1　以实例公司核心业务状态用CBMG 分析其用户行为模型

根据淘宝网的核心业务用CBMG 分析。

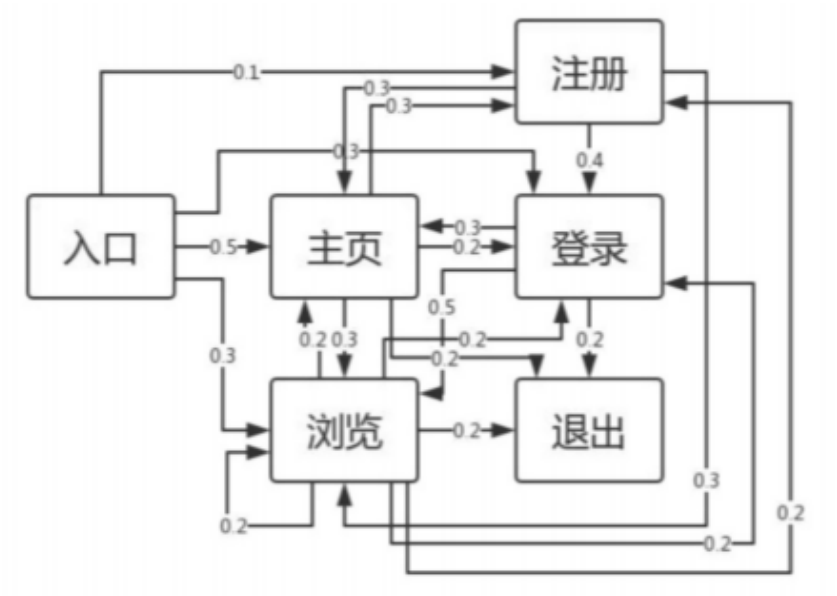


1.2　根据给定客户行为模型转换概率矩阵求解问题

题目给出的概率矩阵P 如图：

	入口	登录	注册	主页	浏览	退出
入口	0	0.3	0.1	0.5	0.1	0
登录	0	0	0	0.3	0.5	0.2
注册	0	0.4	0	0.3	0.3	0
主页	0	0.2	0.3	0	0.3	0.2
浏览	0	0.2	0.2	0.2	0.2	0.2
退出	0	0	0	0	0	0

1.2.1 客户行为模型图CBMG



1.2.2 CBMG各个状态平均访问次数

设入口平均访问次数为1，根据概率矩阵建立多元一次方程组：

$$
\begin{aligned}
V1 &= 0.3 + 0.4V2 + 0.2V3 + 0.2V4 \\
V2 &= 0.1 + 0.3V3 + 0.2V4 \\
V3 &= 0.5 + 0.3V1 + 0.3V2 + 0.2V4 \\
V4 &= 0.1 + 0.5V1 + 0.3V2 + 0.3V3 + 0.2V4 \\
V5 &= 0.2V1 + 0.2V3 + 0.2V4
\end{aligned}
$$

简记成：

$$V^T P^T = V^T$$

其中P为题目中概率矩阵，V为各个状态的平均访问次数的向量，即：

$$V = (1.0, V_1, V_2, V_3, V_4, V_5)^T$$

依次分别对应入口，登录，注册，主页，浏览，退出。
使用高斯消元的方法：

```
32      for (int col = 1, row; col < N; ++col){
33          for (row = 1; row < N; ++row){
34              if (fabs(a[row][col]) > eps && !used[row]){
35                  break;
36              }
37          }
38          if (row >= N){
39              break;
40          }
41          used[row] = 1;
42          for (int i = 0; i < N; ++i){
43              if (fabs(a[i][col]) > eps && i != row){
44                  double ra = a[i][col] / a[row][col];
45                  for (int j = 0; j < N; ++j){
46                      a[i][j] -= ra * a[row][j];
47                  }
48              }
49          }
50          print(a);
51      }
52      double tot = 0;
53      for (int i = 1; i < N; ++i){
54          v[i] = -a[i][0] / a[i][i];
55          printf("%d %lf\n", i, v[i]);
56          if (i != N-1) tot += v[i];
57      }
```

最后解得：

$$V = (1.000000, 1.409759, 0.979278, 1.612299, 1.977941, 1.000000)^T$$

1.2.3 平均会话长度

平均会话长度$= (1.409759 + 0.979278 + 1.612299 + 1.977941) = 5.979278$

2 协同过滤算法的学习

2.1 Correlation-Based Similarity算法实践与分析

修改main中有关测试的代码，我加入了所有用户（并去掉了那个无用的用户），代码修改如下：

```
test.predict("Jim");
test.predict("Alice");
test.predict("Sally");
test.predict("Kevin");
test.predict("Tommy");
test.predict("George");
test.predict("Joseph");
```

看了看CorSimilarityMethod 的java 文件后，**发现第一个错误**：特判0 的语句后面没有接else，这样相当于无效（而且i 和j 也打反了）。以下是修改前和修改后的结果：

```
if (den == 0.0){
    similarityResult[i][j]=0.0;
    similarityResult[i][j]=0.0;
}
similarityResult[i][j] = num / den;
similarityResult[j][i] = num / den;
```

```
if (den == 0.0){
    similarityResul:[i][j]=0.0;
    similarityResul:[j][i]=0.0;
}
else{
    similarityResul:[i][j] = num / den;
    similarityResul:[j][i] = num / den;
}
```

随后在预测部分**发现了第二个错误**：论文中对$P_{u,i}$函数的定义如下：

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$$

然而实现中却没有取绝对值。于是我把代码改成了：

```
public double predict(int uid,int iid){
    double simTotal =0.0, ratingSimToal=0.0;
    for(int i=0;i<uiRating[0].length;i++){
        if(i!=iid && uiRating[uid][i]>0){
            simTotal+=Math.abs(similarityResult[i][iid]);
            ratingSimToal+= similarityResult[i][iid] * uiRating[uid][i];
        }
    }
```

作业截止前看了看QQ 群，**经过助教的提示发现第三个错误**：物品i 和j 是要共同拥有的，而原来的实现并不能做到这一点。一个最简单的改进方法是，在原先代码实现里，每个有值判断的if 后面都加上else continue，如下：

```
for(int k=0;k<userNum;k++){
    double diff1 = 0.0;
    double diff2 = 0.0;
    if(uiRating[k][i]>0)
        diff1 = uiRating[k][i] - avg[k];
    else continue;
    if(uiRating[k][j]>0)
        diff2 = uiRating[k][j] - avg[k];
    else continue;
```

改正了以上三个问题后，最终运行结果如下：

```
Jim to mutton: predictedscore 3.653657232792785
Alice to sausage: predictedscore 3.3426637116322606
Alice to mutton: predictedscore 3.476262407794173
Sally to sausage: predictedscore 2.02799113489678
Sally to mutton: predictedscore 2.428787223382519
Kevin to chicken: predictedscore 1.4990272336720842
Tommy to pork: predictedscore 2.0
Tommy to chicken: predictedscore 1.9999999999999993
George to pork: predictedscore 1.0
George to chicken: predictedscore 0.9999999999999997
Joseph to pork: predictedscore 5.0
Joseph to chicken: predictedscore 4.999999999999998
```

相似度矩阵如下：

```
0.0 0.9710083124552243 0.9093531090412993 0.1709408646894569 -1.5700924586837752E-16
0.9710083124552243 0.0 0.9779599654127862 0.8626979700976577 1.0
0.9093531090412993 0.9779599654127862 0.0 1.0 0.0
0.1709408646894569 0.8626979700976577 1.0 0.0 0.9961164901835047
-1.5700924586837752E-16 1.0 0.0 0.9961164901835047 0.0
```

考虑关系矩阵，前三列物品和后两列物品分别相似：

```
double[][]uiRating_matrix ={
        {4,4,0,3,3},
        {4,0,0,3,3},
        {4,0,0,1,1},
        {1,1,1,2,0},
        {2,2,2,0,0},
        {1,1,1,0,0},
        {5,5,5,0,0}
};
```

- Jim相比较于Alice，多买了sausage，所以对于Jim的第三样物品mutton的推荐值略大一些。
- Sally与Alice比较，对后两样的喜好更弱一些。然后后两样与前三样物品存在的相似度较弱，则Sally推荐值比Alice小一些。
- Tommy，Geroge和Joseph与前三样物品的关系分别为2,1,5，与后两样物品无关系。于是对他们而言，**推荐值与它们对别的物品的推荐值相关，与彼此无关**。
- Kevin和George相比，与pork的关系多了2，然后pork与chicken相似，那么推荐值会更高。

说明这个推荐算法还是比较符合人们的直观感受的。

2.2 Adj-Cosine Similarity算法实践与分析

完善代码很简单。我们先把CorSimilarityMethod.java 里的程序复制过来，但是Adj-Cosine Similarity 算法取的平均值是针对人的，而不是物品的，如下图：

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j} - \bar{R}_j)^2}}.$$

Here $R_{u,i}$ denotes the rating of user u on item i, $\bar{R}_i$ is the average rating of the i-th item.

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j} - \bar{R}_u)^2}}.$$

Here $\bar{R}_u$ is the average of the u-th user's ratings.

我们只需把代码中的对物品i 求平均改成对人j 求平均即可。测试结果如下：

```
Alice to sausage: predictedscore -0.5594816571878153
Alice to mutton: predictedscore 0.5
Sally to sausage: predictedscore 0.743227387722989
Sally to mutton: predictedscore 1.5
Kevin to chicken: predictedscore 0.007380073800737906 6
Tommy to pork: predictedscore -2.0
Tommy to chicken: predictedscore -2.0
George to pork: predictedscore -1.0
George to chicken: predictedscore -1.0
Joseph to pork: predictedscore -4.999999999999999
Joseph to chicken: predictedscore -5.0
```

其中相似度矩阵为：

```
0.0 0.9999999999999998 1.0 -0.8792524835189095 -0.9780219780219781
0.9999999999999998 0.0 1.0 -0.8682431421244593 -1.0
1.0 1.0 0.0 -1.0 0.0
-0.8792524835189095 -0.8682431421244593 -1.0 0.0 0.9999999999999999
-0.9780219780219781 -1.0 0.0 0.9999999999999999 0.0
```

与上一次算法不同的是，这次**相似度会出现负数**。我尝试分析原因。

继续上一个算法我们发现，beef,sausage 和mutton 以及后两样物品pork,chicken 彼此相似度为正数，而这两类之间的相似度均为负数。这能说明什么问题呢？如果用户A 喜欢前三种，那么他很可能反而讨厌后两种——所以测试结果里才会有负数。