**用户行为对酒店特征值的兴趣模型**

制作人：张泽鑫

日期：2013/08/15

**一、引言**

通过分析用户行为的不同，来设定其对于酒店不同特征值的喜好程度计算公式，从而获得用户对于酒店的喜好度（评分），用于修正协同过滤算法中评分项的精确性。

二**、酒店特征值**

酒店级别；酒店价格；酒店区域；酒店服务；酒店类型（暂时不易分类，以后做）。

对于酒店的特征值划分不同的取值区间，根据用户search/view的酒店分布在不同的区间，以及其search/view的行为不同来给酒店预测评分。

通过这种基于内容过滤的评分方式，来提高协同过滤的评分精确度。

**三、酒店特征值取值区间划分**

将酒店各个特征值划分为不同的取值区间，每个酒店可以对应一个特征值的一个或者多个取值区间。

**1)酒店级别**

酒店级别划分为4个区间，分别为2星及2星以下，3星，4星，5星。其数据可直接从酒店基础数据中获取。

每个酒店在酒店级别的分布上只能坐落于一个区间，不可以跨区间。

**2)酒店价格**

酒店价格目前划分为10个区间，(0,100), (101,250), (251,400), (401,550), (551,700), (701,900), (901,1100), (1101,1300), (1301,1600), (1600,+ ∞)，各区间依次为Price\_0, Price\_1, Price\_2, Price\_3, Price\_4, …, Price\_9。每个区间对应的价格范围如下表所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Price\_0 | Price\_1 | Price\_2 | Price\_3 | Price\_4 | Price\_5 | Price\_6 | Price\_7 | Price\_8 | Price\_9 |
| (0,100) | (101,250) | (251,400) | (401,550) | (551,700) | (701,900) | (901,1100) | (1101,1300) | (1301,1600) | (1600,+ ∞) |

酒店所在的价格区间存放于**HotelPriceSection**表中：

|  |  |  |  |
| --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 |
| 1 | Hotel | 酒店ID | int |
| 2 | Price\_0\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 3 | Price\_1\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 4 | Price\_2\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 5 | Price\_3\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 6 | Price\_4\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 7 | Price\_5\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 8 | Price\_6\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 9 | Price\_7\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 10 | Price\_8\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |
| 11 | Price\_9\_locate | 房型价格是否位于其中，是取1，否取0 | tinyint |

判断一个酒店所在的价格区间方法如下：

对于一个酒店Hotel，若其各房型（不关心具体房型）的价格分别为x、y、z，且x位于Price\_x表示的区间内，y位于Price\_y表示的区间内，z位于Price\_z表示的区间内，则认为Hotel所在的字段为Price\_x\_locate, Price\_y\_locate, Price\_z\_locate，这三个区间表示为1，其余字段为0

**例：**某酒店有三种房型，价格分别为220, 450和800，因为220∈(101,250)，450∈(401,550)，800∈(701,900)，所以220, 450和800分别坐落于Price\_1, Price\_3, Price\_5三个区间内，则该酒店所在的字段为Price\_1\_locate, Price\_3\_locate, Price\_5\_locate，在表中表示如下：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hotel | Price\_0\_locate | Price\_1\_locate | Price\_2\_locate | Price\_3\_locate | Price\_4\_locate | Price\_5\_locate | Price\_6\_locate | Price\_7\_locate | Price\_8\_locate | Price\_9\_locate |
| \*\* | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

表示为向量则是Upricescore = <0,1,0,1,0,1,0,0,0,0>

**3)酒店区域**

酒店区域主要为酒店所属商圈，不进行取值区间划分，单独每个商圈ID进行累加得到score分值，标准化为rating评分时需要将商圈按城市分类。

**4)酒店服务**

酒店服务以酒店设施来表现。酒店设施目前只考虑wifi，breakfast，parking。

**四、用户群体筛选规则**

根据孙文龙同学所作的统计，点击次数在10~50次之内的用户虽然少，却是最有可能下订单的用户（概率在15~30%之内），超过100的则数据紊乱，很可能是网络爬虫之类。故这里将点击次数在10~50次的用户筛选出来作为进行评分的用户群体。

**五、用户行为**

用户行为定义为用户在web页面上的操作，每次操作会对用户给酒店的评分进行一次累加（累加值根据操作不同而不同）。

用户行为包括：

**1.搜索。**

搜索行为定义为**Usearch()**。

搜索中添加价格区间，表示关注价格，其传入的参数为价格区间price，即Usearch(price)，price=(minPrice,maxPrice)。

搜索中添加星级，表示关注星级，其传入参数为star，star可选值为：”二星级及以下/经济”、”三星级/舒适”、”四星级/高档”、”五星级/豪华”。即Usearch(star)，多个参数则表示为Usearch(star\_2) + Usearch(star\_3)的形式。

搜索中添加商圈，表示关注酒店所在商圈，传入参数为zone，即Usearch(zone)，该行为只有一个参数，无多参数输入。

搜索中添加设施，表示关注酒店设施，传入的参数为facility（目前只考虑wifi，breakfast和parking），即Usearch(facility)，多个参数则表示为Usearch(facility\_1) + Usearch(facility\_2) + Usearch(facility\_3)的形式。

**2.浏览。**

浏览酒店详情页面行为定义为**Uview()**，参数为Hotel，即Uview(Hotel)代表用户对该ID酒店所具有的特征感兴趣。

**3.订单。**订单表示的是用户历史行为，故在用户基础标签中使用，这里不再做分析。

4.表示用户某种行为次数的函数为**num()**，参数为Usearch()和Uview()。

**六、用户行为对酒店特征值评分**

用户行为对酒店特征值的兴趣评分分为两个部分，一是用户不同行为对于酒店特征值的累加分值，表示为**score**，其取值区间为[0,+ ∞)；二是将score标准化后的评分，表示为**rating**，其取值区间为[0,1]。

**七、累加分值score计算规则**

对于不同的酒店特征值有不同的累加分值计算规则。

**1.用户行为对酒店级别累加分值**

酒店级别划分为4个区间，分别为2星及2星以下，3星，4星，5星。对于这4个区间的兴趣度累加分值存放在**UStarScore**表中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | Star\_2\_Score | 用户行为对2星及2星以下酒店的累加分值 | int | 生成规则见\* |
| 3 | Star\_3\_Score | 用户行为对3星酒店的累加分值 | int |
| 4 | Star\_4\_Score | 用户行为对4星酒店的累加分值 | int |
| 5 | Star\_5\_Score | 用户行为对5星酒店的累加分值 | int |

\* 对于i星级酒店的兴趣度累加分值Star\_i\_Score计算规则如下：

1) 每天计算当天兴趣度累加分值前均为0；

2) 用户每次Usearch(star\_i)行为，对于Star\_i\_Score进行2次累加；

3) 用户每次多星级参数搜索行为，即Usearch(star\_i)+Usearch(star\_j)行为，对于Star\_i\_Score和Star\_j\_Score都进行2次累加，更多参数相同处理；

4) 用户每次Uview(Hotel.star=i)行为，都对Star\_i\_Score进行2次累加（Hotel.star=i表示该酒店星级为i）；

5) 则**Star\_i\_Score = 2\*num(Usearch(star\_i)) + num(Uview(Hotel.star=i))**，Star\_i\_Score范围为[0,+ ∞)。

**2.用户行为对酒店价格累加分值**

用户行为对酒店价格的兴趣度累加分值存放在**UStarScore**表中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | Price\_0\_Score | 用户行为对Price\_0的累加分值 | int | 生成规则见\* |
| 3 | Price\_1\_Score | 用户行为对Price\_1的累加分值 | int |
| 4 | Price\_2\_Score | 用户行为对Price\_2的累加分值 | int |
| 5 | Price\_3\_Score | 用户行为对Price\_3的累加分值 | int |
| 6 | Price\_4\_Score | 用户行为对Price\_4的累加分值 | int |
| 7 | Price\_5\_Score | 用户行为对Price\_5的累加分值 | int |
| 8 | Price\_6\_Score | 用户行为对Price\_6的累加分值 | int |
| 9 | Price\_7\_Score | 用户行为对Price\_7的累加分值 | int |
| 10 | Price\_8\_Score | 用户行为对Price\_8的累加分值 | int |
| 11 | Price\_9\_Score | 用户行为对Price\_9的累加分值 | int |

\*对于Price\_i价格区间的兴趣度累加分值Price\_i\_Score的计算规则如下：

1) 每天计算当天兴趣度累加分值前均为0；

2) 用户每次Usearch(price)行为，可以获得该price(minPrice,maxPrice)的价格区间向量α；

3) 用户每次Uview(Hotel)行为，可以根据该酒店的价格区间向量获得期望的价格区间向量β；

**例：**某用户有某日3次Usearch(price)行为，搜索价格区间为(50,120), (100,150), (100,500)；有Uview(Hotel)行为2次，Hotel\_1坐落的价格区间向量为β1=<1,1,1,0,0,1,0,0,0,0>；Hotel\_2的价格区间向量为β2=<0,1,1,1,0,0,0,0,0,0,>。(50,120)所跨区间为Price\_0, Price\_1，向量表示为α1=<1,1,0,0,0,0,0,0,0,0>；(100,150)所跨区间为Price\_1,向量表示为α2=<0,1,0,0,0,0,0,0,0,0>；(100,500)所跨区间为Price\_1, Price\_2, Price\_3,向量表示为α3=<0,1,1,1,0,0,0,0,0,0>。

则该用户当日行为对酒店价格的兴趣度累加分值向量为

Upricescore = α1 + α2 +α3 +β1 +β2 = <2,5,3,2,0,1,0,0,0,0>

存放在UStarScore表中即为：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Uid | Price\_0\_Score | Price\_1\_Score | Price\_2\_Score | Price\_3\_Score | Price\_4\_Score | Price\_5\_Score | Price\_6\_Score | Price\_7\_Score | Price\_8\_Score | Price\_9\_Score |
| \* | 2 | 5 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |

**3.用户行为对酒店区域累加分值**

用户行为对酒店区域（所在商圈）的兴趣度累加分值存放在**URegionScore**中：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | City | 商圈所在城市ID | int |  |
| 3 | Zone | 商圈ID | int |  |
| 4 | ZoneScore | 对商圈兴趣度累加分值 | int | 生成规则见\* |

因为商圈编号过多且无法按数值划分区间，所以与酒店星级和酒店价格的分值表结构不同。

\*ZoneScore生成规则如下：

1) 每天计算当天兴趣度累加分值前均为0；

2) 用户每次Usearch(Zone=i)行为可以为编号为i的商圈累加2次分值；

3) 用户每次Uview(Hotel.Zone=i)行为可以为编号为i的商圈累加1次分值（Hotel.Zone=i表示该酒店属于标号为i的商圈）；

4) **ZoneScore(zone=i) = 2\*num(Usearch(zone=i)) + num(Uview(Hotel.Zone=i))**；

**4.用户行为对酒店服务累加分值**

酒店服务以酒店设施来表现。现阶段考虑的酒店设施只有wifi，breakfast和parking。

存放用户行为对酒店设施兴趣度的累加分值存放在**UFacilityScore**中：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | WifiScore | 对wifi选项的兴趣累加分值 | Int | 生成规则见\* |
| 3 | BreakfastScore | 对breakfast选项的兴趣累加分值 | Int |
| 4 | ParkingScore | 对parking选项的兴趣累加分值 | Int |

\*WifiScore，BreakfastScore，ParkingScore生成规则如下（以wifi为例）：

1) 每天计算当天兴趣度累加分值前均为0

2) 用户每次Usearch(wifi)行为为WifiScore累加2次；

3) 用户每次Uview(Hotel.wifi=true)行为为WifiScore累加1次（Hotel.wifi=true表示该酒店有wifi设施）；

4) **WifiScore = 2\*num(Usearch(wifi)) + num(Uview(Hotel.wifi=true))；**

**八、标准化评分rating**

获取累加分值score后，因为score未标准化不便于后续计算，故需要将score标准化为评分rating后才能进行进一步计算。

**1.UStarScore标准化为UStarRating**

标准化后的酒店级别评分Star\_i\_Rating存放于**UStarRating**中

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | Star\_2\_Rating | 用户行为对2星及2星以下酒店的评分 | decimal | 生成规则见\* |
| 3 | Star\_3\_Rating | 用户行为对3星酒店的评分 | decimal |
| 4 | Star\_4\_Rating | 用户行为对4星酒店的评分 | decimal |
| 5 | Star\_5\_Rating | 用户行为对5星酒店的评分 | decimal |

\*Star\_i\_Rating的计算规则如下：

1) 每天计算当天兴趣评分前均为0；

2) 用户行为对酒店级别的总评分为ΣStar\_i\_Score；

3) **Star\_i\_Rating = Star\_i\_Score/(ΣStar\_i\_Score)**；

注：ΣStar\_i\_Rating = 1

**2.UPriceScore标准化为UPriceRating**

标准后的酒店价格评分Price\_i\_Rating存放在**UPriceRating**中

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | Price\_0\_Rating | 用户行为对Price\_0的评分 | decimal | 生成规则见\* |
| 3 | Price\_1\_Rating | 用户行为对Price\_1的评分 | decimal |
| 4 | Price\_2\_Rating | 用户行为对Price\_2的评分 | decimal |
| 5 | Price\_3\_Rating | 用户行为对Price\_3的评分 | decimal |
| 6 | Price\_4\_Rating | 用户行为对Price\_4的评分 | decimal |
| 7 | Price\_5\_Rating | 用户行为对Price\_5的评分 | decimal |
| 8 | Price\_6\_Rating | 用户行为对Price\_6的评分 | decimal |
| 9 | Price\_7\_Rating | 用户行为对Price\_7的评分 | decimal |
| 10 | Price\_8\_Rating | 用户行为对Price\_8的评分 | decimal |
| 11 | Price\_9\_Rating | 用户行为对Price\_9的评分 | decimal |

\*Price\_i\_Rating的计算规则如下：

1) 每天计算当天兴趣度评分前均为0；

2) 用户行为对酒店价格的总评分为ΣPrice\_i\_Score；

3) **Price\_i\_Rating = Price\_i\_Score/(ΣPrice\_i\_Score)**；

注：ΣPrice\_i\_Rating = 1

**3.URegionScore标准化为URegionRating**

标准化后的酒店区域评分ZoneRating存放在**URegionRating**中

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | City | 商圈所在城市ID | decimal |  |
| 3 | Zone | 商圈ID | decimal |  |
| 4 | ZoneRating | 对该ID商圈兴趣度评分 | decimal | 生成规则见\* |

\*ZoneRating的计算规则如下：

1) 每天计算当天兴趣度评分前均为0；

2) 将ZoneScore按city分类；

3) 则对于制定的Uid和city，该用户行为对酒店区域的总评分为ΣZoneScore(Zone=i)，Zone=i表示不同的商圈；

4) 对于商圈ID为i的商圈，即Zone=i，其ZoneRating计算公式为**ZoneRating(Zone=i) = ZoneScore(Zone=i)/(ΣZoneScore(Zone=i))**；

注：同一city中，ΣZoneRating(Zone=i) = 1

**4.UFacilityScore标准化为UFacilityRating**

标准化后的酒店设施评分WifiRating，BreakfastRating和ParkingRating存放在**UFacilityRating**中

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 顺序 | 字段名 | 描述 | 类型 | 生成规则 |
| 1 | Uid | 用户ID | varchar | 用户标识符 |
| 2 | WifiRating | 用户行为对wifi的兴趣评分 | decimal | 生成规则见\* |
| 3 | BreakfastRating | 用户行为对breakfast的兴趣评分 | decimal |
| 4 | ParkingRating | 用户行为对parking的兴趣评分 | decimal |

\*WifiRating，BreakfastRating和ParkingRating生成规则如下（以wifi为例）：

1) 每天计算当天兴趣度评分前均为0；

2) 用户行为对设备的总评分为WifiScore+BreakfastScore+ParkingScore；

3) **WifiRating = WifiScore/(WifiScore+BreakfastScore+ParkingScore)**；

注：WifiRating+BreakfastRating+ParkingRating = 1