湖南省人民政府学位委员会办公室
湖南省教育厅学位管理与研究生教育处

# 2016 湖南省研究生数学建模竞赛参赛承诺书

我们仔细阅读了湖南省研究生数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：

我们的参赛报名号为（如果组委会设置报名号的话）：

所属学校（请填写完整的全名）：国防科学技术大学

参赛队员 (打印并签名) ：1.何林

2.蒋远翔

3.赵翔

指导教师或指导教师组负责人(打印并签名)：

日期： 2016 年 04 月 19 日

---

评阅编号（由组委会评阅前进行编号）：

# 2016 湖南省研究生数学建模竞赛

# 编 号 专 用 页

评阅编号（由组委会评阅前进行编号）：

评阅记录（可供评阅时使用）：

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 评阅人 | | | | | | | | | | |
| 评分 | | | | | | | | | | |
| 备注 | | | | | | | | | | |

# 湖南省第二届研究生数学建模竞赛

题 目 基于 Web 日志挖掘的用户行为分析


**摘 要：**

本文依据法律网站用户访问的基本现状，通过建立数学模型，在挖掘现有数据的基础上，分析并得出了用户访问法律网站的一系列特征和规律，评估了网站的组织结构和运行效率。最后，论文还对提高网站运行效率、改善网站组织结构提出了具体方案，并对方案的改进效果进行了评估。

论文首先在对原始日志数据进行数据净化的基础上，运用描述型统计模型，得出用户关注的法律领域和访问路径规律。结果显示定罪量刑、故意伤害、妇女权益、劳动合同纠纷、质量问题是用户最关注的五个法律领域；领养、医疗事故、酒驾等领域用户关注程度低。

访问路径方面，结果显示 57.2%通过搜索引擎搜索关键词进入网页，40.8%的用户通过网站首页获取信息，2%的用户直接登录网站子页面，还有极少数从境外入口访问网站的用户。用户访问法律咨询网站，主要目标网页在咨询内容页，占访问总量的预约律师、参加律师讲坛的用户较少。

关于用户访问效率的问题，本文建立了基于会话识别的效率评估模型，。首先对日志文件进行会话识别，区分单记录会话和多记录会话，对单记录会话根据用户访问习惯建立模型求解访问成功率，对多记录会话求解用户每次访问的无效访问时间 $t$，进而评估用户访问效率。结果显示访问效率最低的项目为在线特约，律师访谈。综合考虑单记录会话和多记录会话的情况下，咨询类项目的访问效率最高，且从搜索引擎进入网页具有更高的访问效率。

关于用户针对特定问题找到相关页面的成功率，本文将法律领域的常用关键词与每个领域的标题建立并联关系，通过训练为每个法律领域建立一个关键词库。在将用户搜索的关键词检索关联该条记录对应的关键词库，以此判断用户的关键词是否关联到正确的网页，求解事件成功率。结果显示搜索类用户整体访问的成功率为 87.71%。成功率与

在设计提高网页服务效率的方案上，本文通过经典的 APRIOR 关联算法，挖掘标题类型、页面类型、访问时间、访问路径之间的二维和多维关联，得出一系列结论，并据此设计了网页的自动推送方案。

在改善网页组织结构上，本文跟踪网页路径信息，结合网页链接的点击量与点击链接所用的时间，定义网页与网页之间链接的效费值，量化评估所有从目录页到目标页面的链接花费的效能，分析链接之间的权重，为改善网页组织结构提供了具体建议。



**关键词：**统计模型 会话识别 关联 APRIOR 置信度

# 一、问题重述

全国法律资源稀缺和地区分配不平衡的现状催生了法律咨询网站的诞生，用户在法律咨询网站寻求法律咨询、获取法律知识、寻求法律援助等。用户浏览网页的过程可以简化为根据既定目标，在网页上寻找对应信息的过程。在网上寻求法律资源存在以下几点缺陷：

用户的搜索习惯主要分为直接搜索关键词、浏览网页通过路径寻找信息、网上咨询三类。其中后两种搜索成功率高，但是效率低，直接搜索关键词的方法效率相对较高，但是法律事务专业性强，复杂度高，大部分用户缺乏基本的法律知识，难以通过合理的关键字搜索查找目标信息。

网站的页面组织结构限制用户高效地找到所需求的内容，导致用户获得有效信息的效率和成功率低。用户获取有效信息成功率和效率低的主要表现为寻找目标信息时间过长或者点击网页次数过多。

用户浏览网站生成的日志文件数据类型复杂，数据量大，关联性强。综合分析浏览日志与其他文件可以对用户的个人特征、关注的领域、浏览习惯、等方面做出合理预测，实现消息推送。网站缺乏对用户日志的数据分析，对用户特征和规律掌握不足，难以根据用户行为预判用户的需求。

根据以上法律咨询网站面临的以上几点问题，要求为提高网站的服务效率做以下两点工作：

1、根据提供的数据，分析用户在浏览网站过程中关注领域、用户体验、访问路径等方面的特征和规律。评估用户在法律咨询网站时，既定需求获得满足的成功率和效率。

2、根据对网站现有数据分析得出的结论，设计网站的改进方案，采用改进页面组织结构、增加新功能、建立新型服务模式等方法，改进服务效率。对比改进前后网站的运行效率，定量评估改进的方案在提高用户找到针对性的法律知识、相关案例或专业律师等方面产生的效果。

# 二、模型假设

1、日志文件中的数据准确反应了用户浏览网站的全部记录。

2、网站的搜索引擎能给出包含搜索关键字的全部信息。

3 所有用户在浏览网站时均为单人浏览

4、从数据中显示的时间至今，该网站的搜索引擎、页面组织结构没有发生大的变化。

5、所有用户在浏览网站时均没有刻意浏览与目的无关的内容。

6、不考虑点击链接后的响应时间对浏览效率的影响。

7、不考虑用户在单次浏览中存在的多目的现象

# 三、基本符号说明

$t$　　用户一次多记录会话中的无效访问时间

$\eta$ 会话识别中设置的时间阈值

$\boldsymbol{R}$ 标题类型数据集

$\boldsymbol{P}$ 常用关键词组成的关键词集合

$\boldsymbol{Q}$ 页面类型数据集

$\boldsymbol{R_i}$ 标题类型$i$包含的关键词库

$\boldsymbol{R_i}'$ 标题类型$i$包含的所有题目组成的集合

$(Pid_i, t_i)$ 用户在一次会话中第$i$项纪录和纪录的时间戳

$G_{ij}$ 目录页$G_i$到目标页$G_j$的链接

$T_{ij}$ $G_{ij}$的平均耗时

# 四、问题分析

## 4.1 用户行为分析

### 4.1.1 用户访问过程分析

用户访问网站的目的和方式具有多样性，不可预见性，为简化模型，在不失一般性前提下，将用户获取法律信息、寻求法律援助的过程可以简化为以下几类操作：

1、用户在各类搜索引擎（数据中主要包括百度、搜狗等搜索引擎）中直接输入关键字，搜索引擎弹出关联的网页列表，用户从列表中点击进入法律快车网站的子目录。

2、用户进入网站首页，在进一步浏览过程中，通过点击链接获取信息。

3、用户进入网站首页，在首页提供的搜索对话框内输入关键字，通过网站内部的搜索引擎寻找信息。

4、用户直接根据具体网址（输入网址或添加收藏等方式）进入网站子目录，寻找相关信息。

5、境外入口登陆

### 4.1.2 用户访问特征分析

根据用户访问网站的统计特征，用户访问网站大致遵循以下经验规律：

在判断用户关注领域方面特征的问题上，可以根据用户的访问时间。如果用户在某个页面停留时间长，那么可以认为此用户对该页面的兴趣度高于其他页面，如果某页面访问频率高，则认为该页面反映了多数人的兴趣。可见时间和频率是页面子系统两个重要特征。

在判断用户是否成功找到相关信息的问题上，可以结合用户访问页面数量和时间，用户在单页面长时间浏览，可以认为找到了相关信息，访问效率高；用户在一段时间内浏览多个网页，可以认为用户长时间寻找相关信息页面，访问效率低。

对于用户针对特定问题找到相关网页或对口律师的事件，我们定义事件 A：用户成功找到目标内容或专业律师[1]。判断 A 事件是否成立，可以根据用户浏览的频次会话数量，每个会话停留的长度判断。对搜索类用户而言，搜索的关键词与显示网页内容相匹配，则认为 A 成立。

### 4.2 web 日志文件初步分析

每当用户访问站点时，在 Web 日志文件都记录了所访问的页面、时间、用户 ID 等信息。与其他 Web 数据相比，Web 日志具有更加完整严格的结构，因此对 Web 日志中的数据进行挖掘是分析用户浏览站点的个性与共性的重要方式。

下面依次给出以上 5 种登陆方式生成的 Web 日志示例：

由日志文件的数据可知，只有在从其他搜索引擎搜索关键字进入网站子目录的情况下，数据会显示入口网页、搜索关键字和来源。经过网站首页获取信息的用户，无论哪种操作，在 Web 日志中都不会生成以上三样数据。

根据以上分析，将用户的日志数据分成四类，经统计，各类用户数据所占比重如下：

表 3-2 不同登陆方式生成的日志文件比例

| 通过搜索引擎登陆网站子页面 | 直接登陆网站子页面 | 通过登陆网站首页检索信息 | 从境外入口登陆网站 |
|---|---|---|---|
| 57.21% | 1.93% | 40.81% | 0.05% |

由上表可知，通过搜索引擎搜索关键字登陆的用户占大多数，分析这类用户的特征和规律对提高网站的效率有重要意义。

## 五、问题一模型的建立与求解

### 5.1 用户特征建模与求解

Web 日志文件中的数据详细记录了用户浏览网页的时间、入口、路径、页面等信息，对日志文件进行统计数据建模分析，得到用户在关注领域、访问效率两方面的规律。

#### 5.1.1 用户关注领域特征

首先，对日志数据进行数据预处理。用户感兴趣的信息主要集中在以“HTML”关键词结束的网页中，其他的网页可以理解为目录页面或者路径页面（非信息类

页面）。因此，预处理的主要目的是筛选出访问网页以.html 结尾的数据日志[2]，筛选结果如下表：

表 5-1 信息类页面数量及所占比重

| 信息类页面 | 信息类页面占比 | 非信息类页面 | 非信息类页面占比 |
|---|---|---|---|
| 670326 | 80.04% | 167122 | 19.96% |

用户在浏览网站时，生成的日志文件中，页面标题反映了网页相关的法律领域，在 C 环境下对筛选后的日志文件按页面标题进行统计分类，得出不同内容被访问的频次和所占的百分比。

表 5-2 标题类型访问情况

| 页面标题类型 | 频率 | 页面标题类型 | 频率 | 页面标题类型 | 页面标题类型 | 页面标题类型 | 频率 |
|---|---|---|---|---|---|---|---|
| 0 | 0.0377% | 21 | 4.2775% | 41 | 2.9521% | 63 | 0.0080% |
| 1 | 0.2832% | 22 | 0.1169% | 42 | 0.0891% | 64 | 0.0054% |
| 2 | 0.0064% | 23 | 6.9530% | 43 | 4.6706% | 65 | 0.0220% |
| 3 | 0.0562% | 25 | 0.6848% | 44 | 0.0001% | 66 | 0.1679% |
| 4 | 0.2190% | 26 | 14.8180% | 45 | 1.4921% | 67 | 0.1107% |
| 5 | 0.0001% | 27 | 0.0890% | 46 | 0.1955% | 68 | 3.9010% |
| 6 | 0.0019% | 28 | 0.1051% | 47 | 0.4862% | 69 | 3.4482% |
| 7 | 2.5467% | 29 | 0.0518% | 48 | 0.5891% | 70 | 0.1073% |
| 8 | 1.8018% | 30 | 0.0195% | 49 | 0.7620% | 71 | 0.0013% |
| 9 | 0.4304% | 31 | 12.1377% | 51 | 1.5011% | 72 | 0.0980% |
| 10 | 3.1881% | 32 | 0.0101% | 52 | 0.2102% | 73 | 0.9180% |
| 11 | 0.0731% | 33 | 0.1009% | 53 | 0.0900% | 74 | 4.8276% |
| 12 | 0.4124% | 34 | 2.3338% | 55 | 0.0117% | 75 | 0.1854% |
| 13 | 2.8988% | 35 | 0.6605% | 56 | 0.2690% | 76 | 0.0048% |
| 15 | 0.0422% | 36 | 0.1725% | 57 | 0.1735% | 77 | 0.0018% |
| 17 | 0.3655% | 37 | 2.4458% | 59 | 0.0050% | 78 | 0.4838% |
| 18 | 0.2170% | 38 | 0.8926% | 60 | 0.1612% | | |
| 19 | 1.1234% | 39 | 1.7645% | 61 | 1.8756% | | |
| 20 | 2.9024% | 40 | 0.0042% | 62 | 5.9301% | | |

注：编号为 14、16、24、50、54、58 的页面标题，在数据显示的时间段无用户访问，无从得知页面标题名称。

由表可知，用户访问排名前五的页面类型分别为

26-定罪量刑 14.81%　　31-故意伤害 12.14%

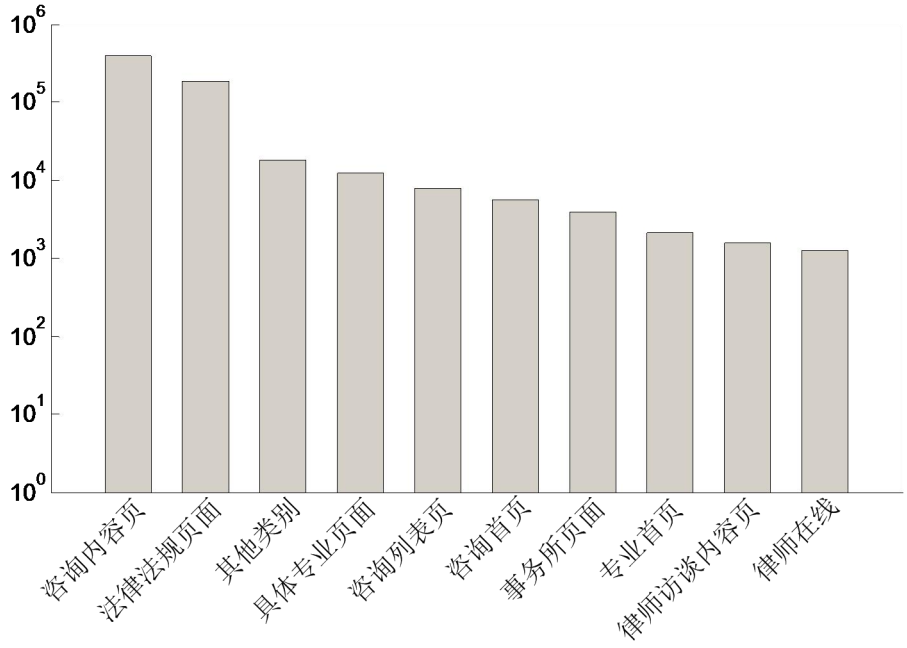
23-妇女权益 6.95%　　62-劳动合同纠纷 5.93%

74-质量问题纠纷 4.83%。

### 5.1.2 用户访问路径特征

对用户访问的页面类型进行分析，得到访问量排名前十的页面类型如下图：

图 5-1 用户主要访问页面类型柱状图




访问路径方面，结果显示 57.2%通过搜索引擎搜索关键词进入网页，40.8%的用户通过网站首页获取信息，2%的用户直接登录网站子页面，还有极少数从境外入口访问网站的用户。用户访问法律咨询网站，主要目标网页在咨询内容页，占访问总量的预约律师、参加律师讲坛的用户较少。

## 5.2 网站运行效率分析

### 5.2.1 模型建立

用户在访问网站时，用户所处的实际情况复杂，目的性和操作习惯不可预测性高，难以将其完全考虑。在不是一般性情况下，为了方便分析，假设用户访问网站时满足以下条件：

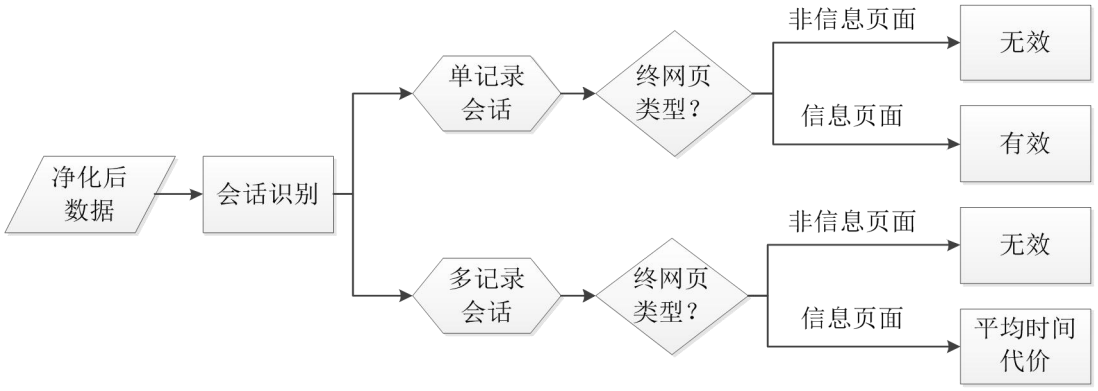
用户在每次访问时目的明确，且单次访问目的唯一，在寻找到目标信息后，用户不再进行其他操作；

用户访问网站以寻找目标信息为基础，进行一定次数的会话；

用户所需的信息全部在以.html 结尾的内容页，其他页面（非信息类页面）不含用户所需的信息。

在满足以上三点条件下，我们建立了会话效率评估模型。

图 5-2 会话效率评估模型




### 5.2.2 会话识别

首先，采用基于时间阈值的会话识别[3]，区分单记录会话与多记录会话。具体过程如下：

（1）设定会话的持续时间阈值$\theta$。在单次访问中一个会话总的时间不超过$\theta$。根据统计规律（引用），本次模型将$\theta=25.5$分钟作为缺省值。

（2）设定页面的访问时间阈值[3]$\eta$。假设$(Pid_i,t_i)$，$(Pid_{i+1},t_{i+1})$为一个用户访问序列中的两条相邻访问记录。设定当$t_{i+1}-t_i\le\eta$，认为这两条记录属于同一个会话，当$t_{i+1}-t_i>\eta$时，$(Pid_i,t_i)$是上一会话的最后一条访问记录，$(Pid_{i+1},t_{i+1})$是新会话的第一条记录。本文$\eta$取 10 分钟（引用：Web 日志挖掘中的会话识别方法）。

（3）依据以上时间阈值条件，在 C 语言环境下对日志数据进行会话识别，得到结果如下表：

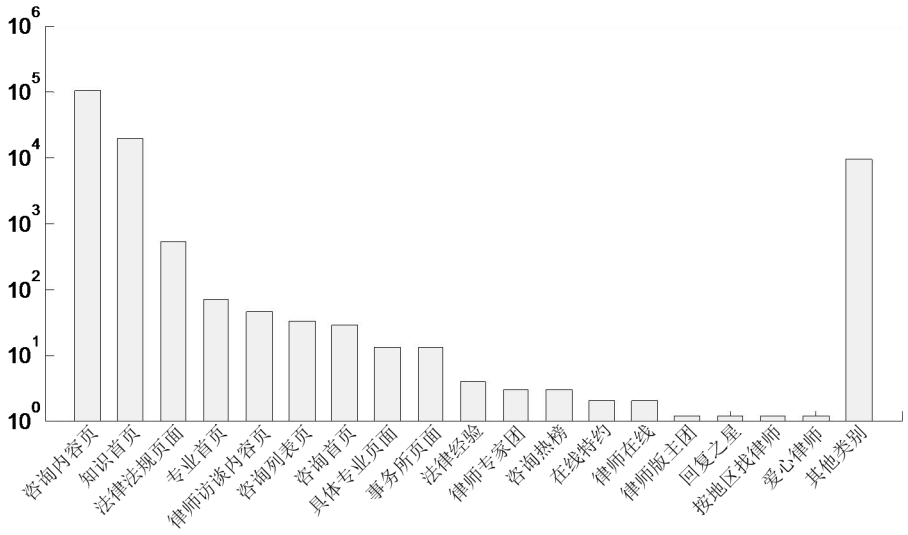
表 5-3 日志文件中会话相关数据统计表（单位：条）

| 日志记录总数量 | 用户数量（基于真实 IP） | 单记录会话数量 | 多记录会话数量 |
|---|---|---|---|
| 837450 | 230149 | 132119 | 291958 |

### 5.2.3 单记录会话效率评估

用户在进行单记录会话访问时，会话页面可分为以.html 结尾的内容页面和其他非信息类页面，若为内容页面，则会话有效，若为非信息类页面，则会话无效。附件 3 提供的页面分类信息，分为以.html 结尾的内容网页，不以.html 结尾的非信息类网页，后缀信息无法判断的网页。因此，综合附件 3 中的页面分类信息，可以判断用户访问的网页是否为内容网页，进而判断此次会话是否有效。得出的按页面类型得出的结果如下图所示：

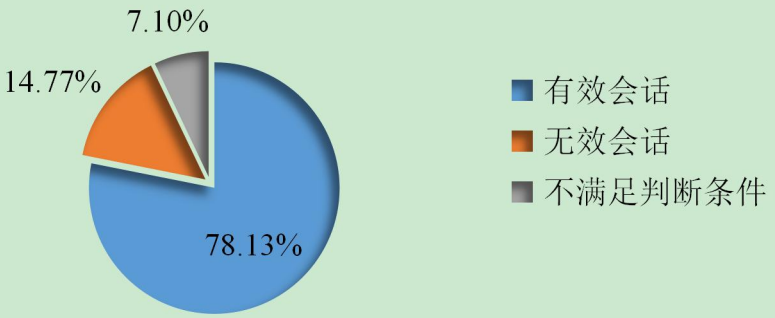
图 5-3 用户访问量排名前 10 的页面类型统计




由图 5-3 结果所示，用户访问法律咨询网站，主要集中在咨询内容页、法律法规页面。用户的访问途径主要是关键词搜索、访问的目的主要是咨询法律内容或者了解法律知识，关注的群体主要是非法律相关从业人员。

为了判定有效性用户单次的有效性，将筛选的单次会话记录按是否以.html 结尾，判断访问有效性。访问的有效性结果如下表：

图4-4 单记录用户访问有效性




结论：网上咨询是单记录会话最主要的形式，单记录会话成功率较高，且单记录有效会话集中在内容咨询页面。

### 5.2.4 多记录会话效果评估

用户在一次多记录会话过程中，经过若干次路径访问，最终结果 P 存在两种情况：

为此，定义无效访问时间$t$评估用户访问的效率。在一次多记录会话中，第一条记录$(Pid_0,t_0)$与最后一条记录$(Pid_n,t_n)$的时间为无效时间$t$。

$$t = t_n - t_0$$

若$P=1$，$t$代表用户在寻找目标信息中所用的时间；若$P=0$，$t$代表此次会话所持续的时间。综上，$t$基本能反映用户的访问效率（s）。

在C语言环境下对多会话数据进行分类统计，得到各页面类型之间的无效访问时间数据如下表：

表5-4：各页面类型消耗的无效访问时间统计表（单位：s）

| 页面类别 | 会话时间 | 会话数量 | 平均耗时 | 页面类别 | 会话时间 | 会话数量 | 平均耗时 |
|---|---|---|---|---|---|---|---|
| 咨询首页 | 159334 | 578 | 276 | 按地区找律师 | 2198 | 13 | 169 |
| 咨询列表 | 187105 | 562 | 333 | 按专业找律师 | 1770 | 21 | 84 |
| 咨询内容 | 12208861 | 171581 | 71 | 爱心律师 | 5939 | 33 | 180 |
| 律师专家 | 2311 | 21 | 110 | 律师文集 | 3391 | 41 | 83 |
| 律师版主 | 5780 | 16 | 361 | 成功案例 | 20677 | 72 | 287 |
| 回复之星 | 4375 | 18 | 243 | 律师团队 | 2144 | 34 | 63 |
| 咨询热榜 | 9349 | 27 | 346 | 律师访谈列表 | 2059 | 5 | 412 |
| 在线特约 | 9686 | 18 | 538 | 律师访谈内容 | 90712 | 342 | 265 |
| 法律经验 | 23076 | 148 | 156 | 事务所页面 | 87276 | 890 | 98 |
| 专业首页 | 66831 | 399 | 167 | 知识首页 | 6989299 | 66617 | 105 |
| 具体专业 | 400996 | 1216 | 330 | 法律法规页面 | 770957 | 4484 | 172 |
| 律师在线 | 29344 | 151 | 194 | 其他类别 | 5452339 | 44671 | 122 |

### 5.2.5 模型结论与分析

由表5-4中信息可以得出以下结论：

大多数用户访问法律咨询网站目的是咨询法律相关内容、获取法律知识；

找律师、案件委托、访谈等会话项目访问用户少；

用户访问网站时，访问效率最低的几类网页分别为在线特约、律师访谈、咨询项目中除去具体内容的其他页面。

## 5.3 用户访问成功率建模与求解

### 5.3.1 模型问题分析

由已知结论，超过半数的用户通过其他搜索引擎访问的方式，查找对特定问题的相关页面，用户在输入关键词搜索进入网站页面时，是否成功找到相关页面，与以下因素有关：

（1）用户使用的搜索关键词是否准确表达了自己的目的；

（2）搜索引擎的搜索算法、效率；

（3）网站上带用户检索关键词的相关页面是否满足用户的需求。

用户在检索过程中，关键词信息难以量化；在关键词匹配的情况下，页面信息也存在不符合用户需求的情况，这些过程难以通过模型和数据分析量化判断。因此用户的浏览规律和关键词是分析的关键。在利用关键词判定用户是否访问成功的问题上，考虑一下两点：

（1）用户搜索的关键词与标题类型并无重合，许多相关领域不能用字符本身来判定访问是否成功。

（2）用户搜索的关键词中包含许多无实义且出现频率很高的字符，如“的”、“怎么办”等，要排除这类字符干扰，避免匹配集扩大，减少误判。

### 5.3.2 基于训练的关键词关联模型

根据以上分析，我们以关键词（query）分析为基础，，利用关键词与目标网页的并发关系，提出构造样本的关键词库，建立基于训练的关键词关联模型[5]，评估检索类用户针对特定问题找到相关页面的成功率。

模型实现主要分两个大部分，基本实现步骤如下：

（1） 通过训练为每个标题类型建立关键词词库

1）在日志文件中筛选出从其他搜索引擎用关键词进入页面的所有日志数据（占总数的 57.21%），生成总数据集 $R$；

2）将关键词列表中的关键词记做集合 $\boldsymbol{P}$，第 $j$ 个关键词记做 $\boldsymbol{P}_j$；

3）对 $R$ 按标题类型分类，集合 $R$ 对应的所有标题内容称为训练集 $R'$，其中标题类型 $i$ 对应的所有标题内容记为 $\boldsymbol{R}_i'$；

4）对 $i$ 从 0 到 78（标题类型总数），将常见关键词列表集合 $\boldsymbol{P}$ 在 $\boldsymbol{R}_i'$ 中遍历，若 $\boldsymbol{P}_j$ 与 $\boldsymbol{R}_i'$ 中某个词匹配，则将 $\boldsymbol{P}_j$ 计入 $R$ 对应的标题类型中，第 $i$ 个标题类型生成的关键词集合记为 $\boldsymbol{R}_i$；

5）$\boldsymbol{R}_i$ 表示经过训练，标题类型 $i$ 所关联的所有关键词，即关键词词库建立完毕。

（2） 第二步：用关键词词库判断用户访问是否成功

1）提取用户搜索的关键词。

在日志文件中，给出了部分搜索的关键词，一些关键词没有给出，在入口网址中隐藏了输入的关键字信息，用解码工具将其解码，提取用户搜索的所有关键词；

2）判断搜索的关键词是否与对应的标题类型匹配。

将关键词以字符为单位，在标题类型所对应的关键词词库中遍历，若标题中的关键词能与词库中的关键词重合，则说明关键词与对应的标题类型匹配，即访问成功；若标题中的字符与该类别的关键词词库无一重合，则判定访问不成功。

### 5.3.3 模型求解

在 C 环境中实现上述算法，整体用户成功率结果如下所示：

表 5-5 搜索类用户访问成功率

| | 样本数 | 比例 |
|---|---|---|
| 成功搜索 | 253237 | 87.71% |
| 失败 | 35478 | 12.29% |
| 样本总数 | 288715 | 100.00% |

求解用户访问成功率与页面类型之间的联系，结果如下表所示：

表 5-6 用户访问不同页面的成功率

| | 成功 | 失败 | 总量 | 成功率 |
|---|---|---|---|---|
| 咨询首页 | 27 | 37 | 64 | 42.19% |
| 咨询列表页 | 50 | 32 | 82 | 60.98% |
| 咨询内容页 | 164330 | 24593 | 188923 | 86.98% |
| 在线特约 | 0 | 6 | 6 | 0.00% |
| 法律经验 | 0 | 4 | 4 | 0.00% |
| 专业首页 | 33 | 5 | 38 | 86.84% |
| 具体专业页面 | 807 | 44 | 851 | 94.83% |
| 按专业找律师 | 2 | 0 | 2 | 100.00% |
| 爱心律师 | 4 | 0 | 4 | 100.00% |
| 律师文集 | 13 | 12 | 25 | 52.00% |
| 成功案例 | 10 | 0 | 10 | 100.00% |
| 律师团队 | 45 | 10 | 55 | 81.82% |
| 律师访谈列表页 | 1 | 0 | 1 | 100.00% |
| 律师访谈内容页 | 187 | 24 | 211 | 88.63% |
| 事务所页面 | 1196 | 110 | 1306 | 91.58% |
| 知识内容页 | 47991 | 4343 | 52334 | 91.70% |
| 其他类别 | 38553 | 6243 | 44796 | 86.06% |

结果分析：由表 5-5，表 5-6 可知，咨询法律内容是搜索类用户访问网页的主要目的。各种主要类型的访问成功率普遍较高，通过检索进行在线特约，查找

律师文集等项目，成功率低。同时，考虑大样本的页面类型训练集丰富，关键词库含关键词数量多，匹配成功率相对较高。

# 六、问题二模型建立与求解

## 6.1 网站服务模式与效率评估

通过对日志数据做统计分析，已经得出一系列和用户访问、网页运行相关的特征和规律。为进一步提高网站个性化服务水平，预测用户访问行为，需要对日志数据进行关联度分析，挖掘用户访问行为的深层次联系，实现自动消息推荐、改善网页组织结构的目的。

Apriori 算法是最经典的关联规则挖掘算法，其核心方法是基于频集理论的递推方法[8]。本文采用基于数据分割的 Apriori 算法，建立关联数据挖掘模型，分析各项访问信息之间的联系。

## 6.2 基于 APRIOR 算法的关联数据挖掘模型

### 6.2.1 关联规则

设$T=\{T_1\ ,\ T_2\ ,\ldots,T_i,...T_m\}$是$m$个不同项的项集，$X\in T$，$Y\in T$，并且$X$和$Y$是不相交的项集，即$X\cap Y=\varnothing$。

定义：$X\to Y$为$X$与$Y$的关联规则。其中，$X$和$Y$分别称为关联规则的先导和后继。关联规则的属性可以用以下三个参数描述:

（1）支持度：表示关联规则的频繁程度。定义为全体事务集$T$中，有 s%的事务同时支持事务集$X$和$Y$，则称 s%为关联规则$X\to Y$的支持度。

$$S(X\to Y)=\frac{N(A\cap B)}{N}\times 100\%$$

其中，最小支持度用$MinS$表示。

（2）置信度：表示关联规则的强度。定义为全体事务集$T$中，在属于支持事务集$X$的条件下，有$c\%$的事务同时支持事务集$Y$，则称$c$为关联规则$X\to Y$的置信度。

$$C(X\to Y)=\frac{N(A\cap B)}{N(A)}\times 100\%$$

其中, 最小置信度用$MinC$表示。

（3）频繁项集：如果事物集 $M$ 的相对支持度不小于定义的最小支持度阈值，则称 $M$ 为频繁项集。

### 6.2.2 APRIOR 算法

APRIOR 算法是第一个关联规则挖掘算法[6]，也是最经典的算法。它利用逐层搜索的迭代方法找出数据库中项集的关系，以形成规则。

算法步骤如下:

(1)设定最小支持度和最小置信度。

(2)确定候选 1 项集。

(3)计算支持度。首先从数据库读入候选项集，得出各项的支持度，找出所有的频繁项集 1。再使用项集的频繁项集 1 来产生候选 2 项集。因为先验原理保证所有非频繁的 1 项集的超集都是非频繁的。

(4)再扫描数据库，得出候选 2 项集，再找出频繁 2 项集，并利用这些频繁 2 项集集合来产生候选 3 项集。

(5)重复扫描数据库，与最小支持度比较，产生更高层次的频繁项集, 再从该集合里产生下一级候选项集，直到不再产生新的候选项集为止。

在此算法中要不断地重复两个步骤:连接和剪枝。

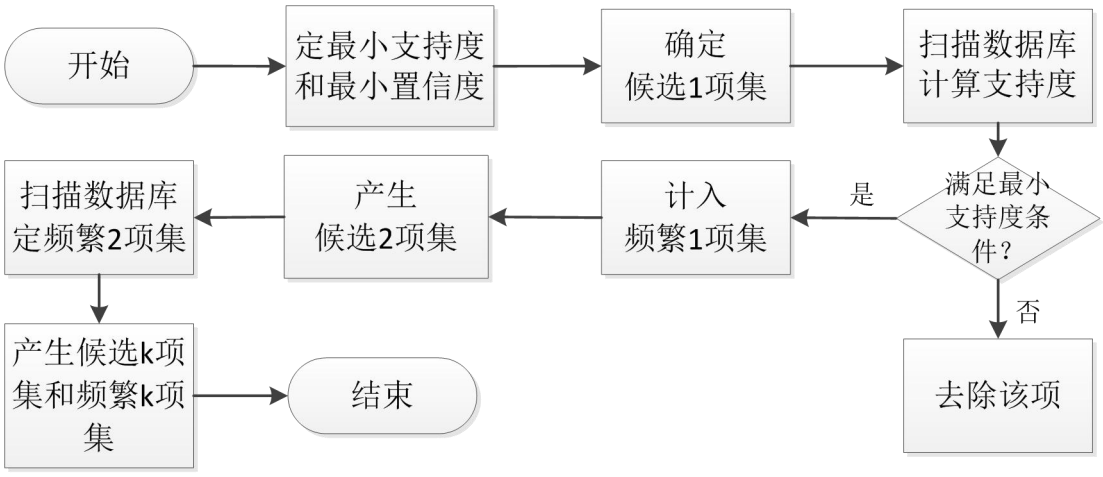
具体内容如下:

(1)连接。为找 $F_k$, 通过 $F_{k-1}$ 与自己连接产生候选 $k$ 项集。该候选项集的集合记做 $L_k$。设 $F_1$ 和 $F_2$ 是 $F_{k-1}$ 中的项集。执行连接 $F_{k-1} \infty F_{k-1}$ ，其中 $F_{k-1}$ 的元素 $F_1$ 和 $F_2$ 是可以连接的。

(2)剪枝。 $L_k$ 的成员不一定都是频繁的, 所有的频繁 $k$ 项集都包含在 $L_k$ 中。扫描数据库, 确定 $L_k$ 中每个候选集计数, 并利用 $F_{k-1}$ 剪掉 $L_k$ 中的非频繁项, 从而确定 $F_k$。

算法流程图如下：

图 6-1 Aprior 算法流程图




### 6.2.3 模型求解

将数据集用标题类型集合 $\boldsymbol{R}$，页面类型数据集 $\boldsymbol{Q}$，作为项集代入上述模型，在 C 语言环境下运行，以访问量为指标，得到标题类型之间的关联度、页面类型之间的关联度。具体结果如下：

图 6-2 标题与页面类型的置信度散点图

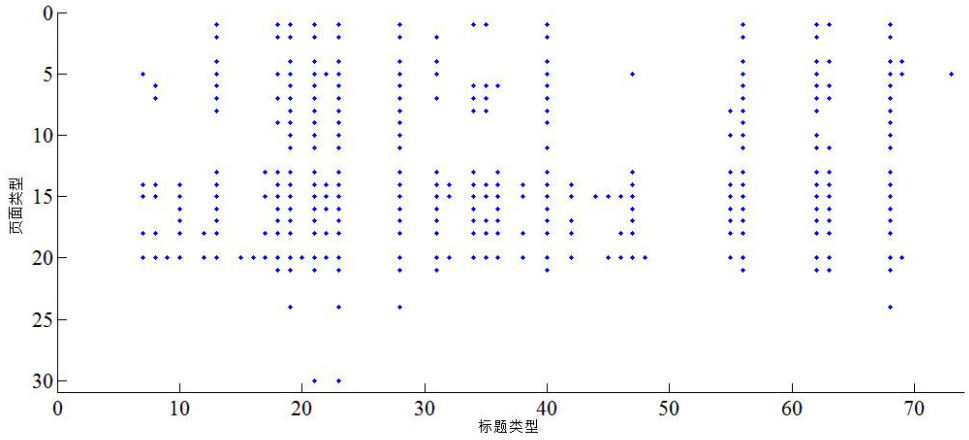



图 6-2 显示了标题类型与页面类型高置信度（大于 0.3）的部分。高亮的点表示该页面类型与对应的标题类型关联度高，同时发生和条件发生的概率高。结合散点图对应的页面信息，可以得出以下结论：

（1）质量问题纠纷、房产买卖、劳动合同、工伤赔偿、医患纠纷、定罪量刑、民间借贷这几个法律领域，在各咨询类、专业律师类页面访问度都很高，适合在线上做自动推送；

（2）企业注册、劳动仲裁、责任认定、股权纠纷、违约赔偿这几个法律领域，在咨询类页面访问度不高，在专业律师相关类页面访问度较高，适合在专业首页和找律师相关页面做自动推送；

（3）其他个别高相关的法律领域与页面类型，如民事赔偿与律师版主团等，适合个别考虑，综合其他信息处理。

### 6.3 网页组织结构分析

网页组织结构的一个重要表现即网站首页与各分页面之间的树状链接。根据第一问中的假设，网站首页不含用户所需信息，一般认为用户登录首页后，寻找目标字网页对应的链接后进入。

定义目录页$G_i$到目标页$G_j$（可能是内容页或者目录页）的链接定义为$G_{ij}$，$G_{ij}$的平均耗时记为$T_{ij}$，链接点击量占总点击数量的比值为$\alpha_{ij}$。则目录页$G_i$到目标页$G_j$的总耗时可表示为$\Phi_{ij}$，则

$$\Phi_{ij} = T_{ij} \times \alpha_{ij}$$

$\Phi_{ij}$的值反映了链接$G_{ij}$消耗网站效能产生的影响。$\Phi_{ij}$越大，用户浪费在点击次链接的总时间越多，对网站效率损失越大。以咨询首页为例，该页面到其他目标页面具体数据如下表：

表 6-1 咨询首页链接及各链接平均耗时

| 目标页面类型 | $T_{ij}$ | $\alpha_{ij}$ | $\Phi_{ij}$ |
|---|---|---|---|
| 咨询首页 | 44.76 | 38.59% | 17.27 |
| 咨询热榜 | 32.59 | 0.78% | 0.25 |
| 咨询内容页 | 57.19 | 16.50% | 9.44 |
| 咨询列表页 | 21.58 | 19.82% | 4.28 |
| 专业首页 | 42.47 | 1.59% | 0.68 |
| 知识内容页 | 40.35 | 12.08% | 4.88 |
| 在线特约 | 11.52 | 0.81% | 0.09 |
| 事务所页面 | 63.18 | 0.39% | 0.25 |
| 律师专家团 | 20.12 | 0.88% | 0.18 |
| 律师文集 | 7.00 | 0.18% | 0.01 |
| 律师团队 | 73.00 | 0.04% | 0.03 |
| 律师访谈内容页 | 16.90 | 0.35% | 0.06 |
| 律师访谈列表页 | 7.67 | 0.11% | 0.01 |
| 律师版主团 | 4.80 | 0.18% | 0.01 |

| | | | |
|---|---|---|---|
| 具体专业页面 | 39.34 | 3.89% | 1.53 |
| 回复之星 | 26.89 | 0.64% | 0.17 |
| 法律经验 | 14.48 | 3.00% | 0.43 |
| 成功案例 | 17.67 | 0.11% | 0.02 |
| 按地区找律师 | 21.00 | 0.07% | 0.01 |

由表 6-1 可知，咨询首页到各目标页面的链接中，到咨询内容页、咨询列表页、知识内容页三个网页的时间期望消耗最高，对网页运行效率的影响最大，在改善网页组织结构时应着重考虑。

### 6.4 网站改进方案概述

#### 6.4.1 网页间组织结构优化

由 6-3 的模型可知，在网页组织结构中，超链接的位置由用户点击链接的数量和此链接的寻找时间确定。设计时应考虑两方面因素：

（1） 链接的点击量，链接的点击量越大，设计的位置应越显眼

（2） 寻找次链接的耗时，点击量不大的链接，若把他们放在特别不起眼的地方，使用户寻找的时间过长，也会产生不良影响。

根据 6-3 的结论，在咨询首页页，推荐的优先链接为咨询内容页、咨询列表页和知识内容页。

#### 6.4.2 自动推送模式介绍

根据 6-2 的结论，网页的自动推送方案做如下设计：

1）质量问题纠纷、房产买卖、劳动合同、工伤赔偿、医患纠纷、定罪量刑、民间借贷这几个法律领域，在各咨询类、专业律师类页面访问度都很高，适合在线上做自动推送；

（2）企业注册、劳动仲裁、责任认定、股权纠纷、违约赔偿这几个法律领域，在咨询类页面访问度不高，在专业律师相关类页面访问度较高，适合在专业首页和找律师相关页面做自动推送；

（3）其他个别高相关的法律领域与页面类型，如民事赔偿与律师版主团等，适合个别考虑，综合其他信息处理。

## 七、模型评价与进一步拓展

### 7.1 模型评价

1. 通用性强：在挖掘日志过程的所用的统计模型、关联模型、基于会话识别的效用评价模型等，在本文适用，同样的数据处理模式可以扩展到其他网站用户特征分析的案例。

2. 直观性：模型建立的假设符合绝大多数用户的习惯，在此基础上的模型直接针对网站和用户的特征和规律，输出的结果直接反应目标问题，不确定性因素少。

3. 精确性：在日志挖掘过程前，将少数非信息类网页过滤，每个模型前都有针对性的做数据预处理，提高模型精确性。

### 7.2 模型的改进与拓展

1.本文通过建立模，讨论了 58%的用户访问网页的规律。用户情况各异，在更大范围内分析用户访问规律有待进一步讨论。

2.在统计搜索类用户搜索关键词过程中，一部分关键词由于网站转码的问题，没有转换为关键词信息，关键词信息包含在入口网页网址中。对这一部分用户的信息，可以通过转码生成关键词再做进一步求解。

3.直接通过网站首页访问的用户，本文在基于会话识别的效率模型中，对用户的访问的成功率和效率进行了评估，要准确求解这类用户访问的成功率，还需进一步工作。

## 八、参考文献

[1] 王大玲. 支持个性化推荐的用户分类规则挖掘的研究[J]. 小型微型计算机系统，2004.11:1948-1951.
[2] 向坚持，陈晓红，刘相滨. 基于 Web Log 的数据预处理研究[J]. 湖南师范大学自然科学学报, 2004,27:33-54.
[3] 周爱武. Web 日志挖掘中的会话识别方法[J]. 计算机工程与设计，2010,31:936-964.
[4] DOUG BEEFERMAN, ADAM BERGER, JOHN LAFFERTY. Statistical Models for Text Segmentation. Machine Learning 34, 177–210 (1999).
[5] 陈健, 印鉴, Web 使用挖掘技术研究综述. 计算机工程[J]，2005,31:3-6.
[6] 牛丽敏. Apriori 算法分析与改进综述. 桂林电子科技大学学报, 2007,27:27-30.
[7] Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, Liyun Ru. How Do Users Describe Their Information Need: Query Recommendation based on Snippet Click Model. Expert Systems With Applications. 38(11): 13847-13856, 2011.
[8] 井福荣. 关联规则在网站结构优化中的改进算法[J], 2007,1:44-47

## 八、附录

附录一：数据归类汇总

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    FILE *fp=NULL,*fp1=NULL,*fhao=NULL;
    char fname[100],string[10000],nuc[10];
    int i=0;
    fp=fopen("E:\\mathmatic_module\\data\\待测库.txt","r");
    fhao=fopen("E:\\mathmatic_module\\data\\hao.txt","r");
    if(fp==NULL||fhao==NULL)
    {
        printf("打开文件错误！\n");
        return -1;
    }
    while(fgets(nuc,10,fhao)!=NULL)
    {
        printf("\r%s",nuc);
        i=atoi(nuc);
        sprintf(fname,"E:\\%d 待测库.txt",i);
        fp1=fopen(fname,"a+");
        if(fp1==NULL)
        {
            printf("%s 打开错误！",fname);
            break;
        }
        fgets(string,10000,fp);
        fputs(string,fp1);
        fflush(fp1);
        fclose(fp1);
    }
    fclose(fp);
    fclose(fhao);
    fclose(fp1);
    return 0;
}
```

附录二：统计用户满意度

```
#include <stdio.h>
#include <string.h>
#define USER struct user
#define USMSG struct usmsg
```

```
USMSG
{
    int aeraid;
    unsigned long long int time;
    int way;
    int aim;
    USMSG *next;
};
USER
{
    unsigned int usrid;
    USER *next;
    USMSG *msghead;
};

void disp(USER *head);
USER *dividesingle(USER **hand,FILE *frule);
USER *readdat(FILE *fp,FILE *frule);
USER *readperson(FILE *fp,int numb);
void sigusraly(USER *hand);
void dubusraly(USER *head);

main()
{
    FILE *fp=NULL,*frule=NULL;
    int person,numb;
    USER *simp_hand=NULL,*sighand=NULL,*dubhand=NULL;
    fp=fopen("E:\\mathmatic_module\\data\\user_new.dat","r");
    frule=fopen("E:\\mathmatic_module\\data\\rule_new.pi","r");
    if(fp==NULL||frule==NULL)
    {
        printf("文件打开错误！");
        exit(0);
    }
    simp_hand=readdat(fp,frule);
    fclose(fp);

    dubhand=simp_hand;
    sighand=dividesingle(&dubhand,frule);
    sigusraly(sighand);
    dubusraly(dubhand);
    return(0);
}
void dubusraly(USER *head)
```

```
{
    FILE *fp;
    USER *p=NULL;
    USMSG *q=NULL,*ql=NULL;
    char *s[30];
    unsigned int dt=0,time=0,htm[30],hnum[30];
    int frq=0,huihua=0,i;
    for(i=0;i<30;i++)
    {
        htm[i]=0;
        hnum[i]=0;
    }
    fp=fopen("E:\\mathmatic_module\\data\\多次用户使用数据统计.txt","w");
    p=head;
//  printf("用户\t 会话编号\t 会话用时/min\t 路径数目\n");
    while(p!=NULL)
    {
        huihua=0;
        q=p->msghead;
        while(q!=NULL)
        {
            ql=q;
            q=q->next;
            if(q!=NULL)
                dt=(q->time-ql->time)/1000;
            if(dt>600||q==NULL)
            {
                huihua=huihua+1;
    //          printf("%u\t%d\t%u\t%d\n",p->usrid,huihua,time,frq);
                fprintf(fp,"%u\t%d\t%u\t%d\n",p->usrid,huihua,time,frq);
                fflush(fp);
                switch(ql->way)//对路径还是对主题的判断依据
                {
                    case 101001 : htm[0]+=time; hnum[0]++; break;
                    case 101002 : htm[1]+=time; hnum[1]++; break;
                    case 101003 : htm[2]+=time; hnum[2]++; break;
                    case 101004 : htm[3]+=time; hnum[3]++; break;
                    case 101005 : htm[4]+=time; hnum[4]++; break;
                    case 101006 : htm[5]+=time; hnum[5]++; break;
                    case 101007 : htm[6]+=time; hnum[6]++; break;
                    case 101008 : htm[7]+=time; hnum[7]++; break;
                    case 101009 : htm[8]+=time; hnum[8]++; break;
                    case 102001 : htm[9]+=time; hnum[9]++; break;
                    case 102002 : htm[10]+=time; hnum[10]++; break;
```

```
                    case 102003 : htm[11]+=time; hnum[11]++; break;
                    case 102004 : htm[12]+=time; hnum[12]++; break;
                    case 102005 : htm[13]+=time; hnum[13]++; break;
                    case 102006 : htm[14]+=time; hnum[14]++; break;
                    case 102007 : htm[15]+=time; hnum[15]++; break;
                    case 102008 : htm[16]+=time; hnum[16]++; break;
                    case 102009 : htm[17]+=time; hnum[17]++; break;
                    case 103001 : htm[18]+=time; hnum[18]++; break;
                    case 103002 : htm[19]+=time; hnum[19]++; break;
                    case 103003 : htm[20]+=time; hnum[20]++; break;
                    case 104001 : htm[21]+=time; hnum[21]++; break;
                    case 105001 : htm[22]+=time; hnum[22]++; break;
                    case 106001 : htm[23]+=time; hnum[23]++; break;
                    case 107001 : htm[24]+=time; hnum[24]++; break;
                    case 107002 : htm[25]+=time; hnum[25]++; break;
                    case 107003 : htm[26]+=time; hnum[26]++; break;
                    case 201001 : htm[27]+=time; hnum[27]++; break;
                    case 301001 : htm[28]+=time; hnum[28]++; break;
                    default : htm[29]+=time; hnum[29]++; break;
                }
                time=0;
                frq=0;
                dt=0;
            }
            else
            {
                dt=(q->time-ql->time)/1000;
                time=time+dt;
                frq=frq+1;
            }
        }
        p=p->next;
    }
    fclose(fp);
    fp=NULL;
    s[0]="咨询首页"; s[1]="咨询列表页"; s[2]="咨询内容页"; s[3]="律师专家团";
s[4]="律师版主团"; s[5]="回复之星"; s[6]="咨询热榜"; s[7]="在线特约";s[8]="法律经验";
s[9]="专业首页"; s[10]="具体专业页面"; s[11]="律师在线"; s[12]="按地区找律师";
s[13]="按专业找律师"; s[14]="爱心律师";s[15]="律师文集";s[16]="成功案例";
s[17]="律师团队"; s[18]="律师访谈"; s[19]="律师访谈列表页"; s[20]="律师访谈内容页";
s[21]="电话咨询页面"; s[22]="案件委托页面"; s[23]="事务所页面";s[24]="知识首页";
s[25]="知识内容页"; s[26]="知识列表页";s[27]="精英律师页面";s[28]="法律法规页面";
s[29]="其他类别";
    fp=fopen("E:\\mathmatic_module\\data\\页面类型效率数据.txt","w");
```

```
    if(fp==NULL)
    {
        printf("打开文件出错！ \n");
        exit(0);
    }
    printf("页面类别\t 访问总用时/min\t 访问量\n");
    for(i=0;i<30;i++)
    {
        printf("%s\t%u\t%d\n",s[i],htm[i],hnum[i]);
        fprintf(fp,"%s\t%u\t%d\n",s[i],htm[i],hnum[i]);
    }
    fclose(fp);
}
void sigusraly(USER *hand)
{
    char *s[30];
    USER *p;
    int i,a[30];
    FILE *fp;
    for(i=0;i<30;i++)
    {
        a[i]=0;
    }
    p=hand;
    while(p!=NULL)
    {
        switch((p->msghead)->way)
        {
            case 101001 : a[0]++; break;
            case 101002 : a[1]++; break;
            case 101003 : a[2]++; break;
            case 101004 : a[3]++; break;
            case 101005 : a[4]++; break;
            case 101006 : a[5]++; break;
            case 101007 : a[6]++; break;
            case 101008 : a[7]++; break;
            case 101009 : a[8]++; break;
            case 102001 : a[9]++; break;
            case 102002 : a[10]++; break;
            case 102003 : a[11]++; break;
            case 102004 : a[12]++; break;
            case 102005 : a[13]++; break;
            case 102006 : a[14]++; break;
            case 102007 : a[15]++; break;
```

```
                case 102008 : a[16]++; break;
                case 102009 : a[17]++; break;
                case 103001 : a[18]++; break;
                case 103002 : a[19]++; break;
                case 103003 : a[20]++; break;
                case 104001 : a[21]++; break;
                case 105001 : a[22]++; break;
                case 106001 : a[23]++; break;
                case 107001 : a[24]++; break;
                case 107002 : a[25]++; break;
                case 107003 : a[26]++; break;
                case 201001 : a[27]++; break;
                case 301001 : a[28]++; break;
                default : a[29]++; break;
            }
            p=p->next;
        };

        fp=fopen("E:\\mathmatic_module\\data\\单次用户使用数据统计.txt","w");
        for(i=0;i<30;i++)
        {
            printf("%s\t%d\n",s[i],a[i]);
            fprintf(fp,"%s\t%d\n",s[i],a[i]);
            fflush(fp);
        }
        fclose(fp);
}
USER *dividesingle(USER **hand,FILE *frule)
{
        USER *singlehand=NULL,*sigusr=NULL,*dubusr=NULL,*p;
        int person,numb,i;
        int singlenumb=0,dubnumb=0;
        rewind(frule);
        fscanf(frule,"%d",&person);
        p=*hand;
        for(i=0;i<person;i++)
        {
            fscanf(frule,"%d",&numb);
            if(numb==1)
            {
                if(singlenumb==0)
                {
                    singlehand=p;
                    sigusr=p;
```

```
            }
            else
            {
                sigusr->next=p;
                sigusr=p;
            }
            singlenumb++;
        }
        else
        {
            if(dubnumb==0)
            {
                *hand=p;
                dubusr=p;
            }
            else
            {
                dubusr->next=p;
                dubusr=p;
            }
            dubnumb++;
        }
        p=p->next;
    }
    printf("单次用户%d 人，多次用户%d 人",singlenumb,dubnumb);
    return(singlehand);
}
USER *readdat(FILE *fp,FILE *frule)
{
    int person,i;
    int numb,j;
    USER *simp=NULL,*q=NULL,*simp_hand=NULL;
    USMSG *p=NULL,*lie=NULL;

    rewind(fp);
    rewind(frule);

    fscanf(frule,"%d",&person);
    for(i=0;i<person;i++)
    {
        fscanf(frule,"%d",&numb);
        q=(USER*)malloc(sizeof(USER));
        if(q==NULL)
        {
```

```
            printf("内存不足!\n");
            exit(0);
        }
        fscanf(fp,"%u",&q->usrid);
        q->next=NULL;
        q->msghead=NULL;
        for(j=0;j<numb;j++)
        {
            p=(USMSG*)malloc(sizeof(USMSG));
            if(p==NULL)
            {
                printf("内存不足！ \n");
                exit(0);
            }
            fscanf(fp,"%d %llu %d %d",&p->aeraid,&p->time,&p->way,&p->aim);
            p->next=NULL;
            if(q->msghead==NULL)
            {
                q->msghead=p;
                lie=p;
            }
            else
            {
                lie->next=p;
                lie=p;
            }
        }
        if(simp_hand==NULL)
        {
            simp_hand=q;
            simp=q;
        }
        else
        {
            simp->next=q;
            simp=q;
        }
    }
    return (simp_hand);
}

void disp(USER *head)
{
    USER *p;
```

```
    USMSG *q;
    for(p=head;p!=NULL;p=p->next)
    {
        for(q=(*p).msghead;q!=NULL;q=(*q).next)
        {
            printf("%u %d %llu %d %d\n",p->usrid,q->aeraid,q->time,q->way,q->aim);
        }
    }
}
```

附录三：统计用户搜索成功率

```
#include <stdio.h>
#include <string.h>
int main()
{
    FILE *fp=NULL,*fpt=NULL;
    char fname[100],keyword[25];
    char order[1000];
    int i,chenggong=0,shibai=0,numb=0;
    int flag=0;
    fpt=fopen("E:\\mathmatic_module\\data\\成功率样本.txt","r");
    if(fpt==NULL)
    {
        printf("文件打开错误！\n");
        exit(0);
    }
    while(fgets(order,1000,fpt)!=NULL)
    {
        numb++;
        flag=0;
        printf("\r 正在测试第%d 条数据。。。",numb);
        sscanf(order,"%d",&i);
        sprintf(fname,"%d 资料库.txt",i);
        fp=fopen(fname,"r");
        if(fp==NULL)
        {
            printf("打开文件失败！\n");
            exit(0);
        }
        while(fgets(keyword,25,fp)!=NULL)
        {
            if(strstr(order,keyword)!=NULL)
            {
                chenggong++;
```

```
                    flag=1;
                    break;
                }
            }
            if(flag==0)
            {
                shibai++;
            }
        }
        fclose(fp);
        fclose(fpt);
        printf("\r 测试完成，采样样本共%d 条。\n
            成功%f\t 失败%f",numb,(double)chenggong/numb,(double)shibai/numb);
}
```

附录四、

```
#include <stdio.h>
#include <string.h>
#define USER struct user
#define USMSG struct usmsg
USMSG
{
    int aeraid;
    unsigned long long int time;
    int way;
    int aim;
    USMSG *next;
};
USER
{
    unsigned int usrid;
    USER *next;
    USMSG *msghead;
};

USER *readdat(FILE *fp,FILE *frule);
void layer1(USER *head,double l1[]);
void layer2(USER *head,double l2[72][72]);
int tihuantime(USMSG *q);
int tihuan(int a);
void wayswitch(USMSG *q,double hum[],int flag[]);
void aimswitch(USMSG *q,double hum[],int flag[]);
void timeswitch(USMSG *q,double hum[],int flag[]);
```

```
int main()
{
    FILE *fp,*frule;
    USER *simp_hand;
    double l1[72],l2[72][72];
    fp=fopen("E:\\mathmatic_module\\data\\user_new.dat","r");
    frule=fopen("E:\\mathmatic_module\\data\\rule_new.pi","r");
    if(fp==NULL||frule==NULL)
    {
        printf("文件打开错误！\n");
        exit(0);
    }
    simp_hand=readdat(fp,frule);
    fclose(fp);
    fclose(frule);
//  layer1(simp_hand,l1);
    layer2(simp_hand,l2);
    return(0);
}

USER *readdat(FILE *fp,FILE *frule)
{
    int person,i;
    int numb,j;
    USER *simp=NULL,*q=NULL,*simp_hand=NULL;
    USMSG *p=NULL,*lie=NULL;

    rewind(fp);
    rewind(frule);

    fscanf(frule,"%d",&person);
    for(i=0;i<person;i++)
    {
        fscanf(frule,"%d",&numb);
        q=(USER*)malloc(sizeof(USER));
        if(q==NULL)
        {
            printf("内存不足!\n");
            exit(0);
        }
        fscanf(fp,"%u",&q->usrid);
        q->next=NULL;
        q->msghead=NULL;
```

```
            for(j=0;j<numb;j++)
            {
                p=(USMSG*)malloc(sizeof(USMSG));
                if(p==NULL)
                {
                    printf("内存不足！\n");
                    exit(0);
                }
                fscanf(fp,"%d %llu %d %d",&p->aeraid,&p->time,&p->way,&p->aim);
                p->next=NULL;
                if(q->msghead==NULL)
                {
                    q->msghead=p;
                    lie=p;
                }
                else
                {
                    lie->next=p;
                    lie=p;
                }
            }
            if(simp_hand==NULL)
            {
                simp_hand=q;
                simp=q;
            }
            else
            {
                simp->next=q;
                simp=q;
            }
        }
        return (simp_hand);
}
void layer1(USER *head,double l1[])
{
        FILE *fp;
        USER *p;
        USMSG *q;
        double hum[8];
        int total=0,i,flag[8];
        p=head;
        for(i=0;i<8;i++)
        {
```

```
            hum[i]=0;
            flag[i]=0;
        }
        while(p!=NULL)
        {
            total++;
            for(i=0;i<8;i++)
            {
                flag[i]=0;
            }
            q=p->msghead;
            while(q!=NULL)
            {
            //  wayswitch(q,hum,flag);
            //  aimswitch(q,hum,flag);
                timeswitch(q,hum,flag);
                q=q->next;
            }
            p=p->next;
        }
        fp=fopen("E:\\mathmatic_module\\data\\L1 时间.txt","w");
        for(i=0;i<8;i++)
        {
            l1[i]=(hum[i]/total)*100;
            printf("%d\t%f\n",i,l1[i]);
            fprintf(fp,"%d\t%f\n",i,l1[i]);
        }
        fclose(fp);
}
void layer2(USER *head,double l2[72][72])
{
        FILE *fp;
        USER *p;
        USMSG *q,*temp;
        double hum[8][72];
        int total=0,i,j,flag[8][72];
        p=head;
        for(i=0;i<8;i++)
        {
            for(j=0;j<72;j++)
            {
                hum[i][j]=0;
                flag[i][j]=0;
            }
```

```
    }
    while(p!=NULL)
    {
        total++;
        for(i=0;i<8;i++)
        {
            for(j=0;j<72;j++)
            {
                flag[i][j]=0;
            }
        }
        q=p->msghead;
        while(q!=NULL)
        {
            temp=p->msghead;
            while(temp!=NULL)
            {

    if(flag[tihuantime(q)][tihuan(temp->aim)]==0)/*&&(flag[tihuan(temp->way)][tihuan(q-
>way)]==0)*/
                {
                    hum[tihuantime(q)][tihuan(temp->aim)]++;
                    flag[tihuantime(q)][tihuan(temp->aim)]=1;
                }
                temp=temp->next;
            }
            q=q->next;
        }
        p=p->next;
    }
/*  total=0;
    for(i=0;i<30;i++)
    {
        for(j=0;j<30;j++)
        {
            total=hum[i][j]+total;
        }
    }*/
    fp=fopen("L2 时间与页面类型.txt","w");
    for(i=0;i<8;i++)
    {
        fprintf(fp,"%d\t",i);
        for(j=0;j<72;j++)
        {
```

```
                l2[i][j]=hum[i][j]/total*100;
                printf("%e\t",l2[i][j]);
                fprintf(fp,"%f\t",l2[i][j]);
            }
            printf("\b\n");
            fprintf(fp,"\b\n");
        }
        fclose(fp);
}
int tihuantime(USMSG *q)
{
        int i,hour;
        hour=(q->time%86400000)/3600000;
        if(hour>=0&&hour<3)
        {
            i=2;
        }
        else if(hour>=3&&hour<5)
        {
            i=3;
        }
        else if(hour>=5&&hour<9)
        {
            i=4;
        }
        else if(hour>=9&&hour<11)
        {
            i=5;
        }
        else if(hour>=11&&hour<12)
        {
            i=6;
        }
        else if(hour>=12&&hour<16)
        {
            i=7;
        }
        else if(hour>=16&&hour<21)
        {
            i=0;
        }
        else
        {
            i=1;
```

```
    }
    return(i);
}
int tihuan(int a)
{
    int i;
    if(a>1000)
    {
        switch(a)
        {
        case 101001 : i=0; break;
        case 101002 : i=1;break;
        case 101003 : i=2;break;
        case 101004 : i=3;break;
        case 101005 : i=4;break;
        case 101006 : i=5;break;
        case 101007 : i=6; break;
        case 101008 : i=7;break;
        case 101009 : i=8;break;
        case 102001 : i=9;break;
        case 102002 : i=10;break;
        case 102004 : i=12;break;
        case 102005 : i=13;break;
        case 102006 : i=14;break;
        case 102007 : i=15;break;
        case 102008 : i=16;break;
        case 102009 : i=17;break;
        case 103001 : i=18;break;
        case 103002 : i=19;break;
        case 103003 : i=20;break;
        case 104001 : i=21;break;
        case 105001 : i=22;break;
        case 106001 : i=23;break;
        case 107001 : i=25;break;
        case 107002 : i=26;break;
        case 107003 : i=27;break;
        case 201001 : i=28;break;
        default : i=29;break;
        }
    }
    else
    {
        switch(a)
        {
```

```
case 1 : i=0; break;
case 2 : i=1; break;
case 3 : i=2; break;
case 4 : i=3; break;
case 5 : i=4; break;
case 6 : i=5; break;
case 7 : i=6; break;
case 8 : i=7; break;
case 9 : i=8; break;
case 10 : i=9; break;
case 11 : i=10; break;
case 12 : i=11; break;
case 13 : i=12; break;
case 15 : i=13; break;
case 17 : i=14; break;
case 18 : i=15; break;
case 19 : i=16; break;
case 20 : i=17; break;
case 21 : i=18; break;
case 22 : i=19; break;
case 23 : i=20; break;
case 25 : i=21; break;
case 26 : i=22; break;
case 27 : i=23; break;
case 28 : i=24; break;
case 29 : i=25; break;
case 30 : i=26; break;
case 31 : i=27; break;
case 32 : i=28; break;
case 33 : i=29; break;
case 34 : i=30; break;
case 35 : i=31; break;
case 36 : i=32; break;
case 37 : i=33; break;
case 38 : i=34; break;
case 39 : i=35; break;
case 40 : i=36; break;
case 41 : i=37; break;
case 42 : i=38; break;
case 43 : i=39; break;
case 44 : i=40; break;
case 45 : i=41; break;
case 46 : i=42; break;
case 47 : i=43; break;
```

```
            case 48 : i=44; break;
            case 49 : i=45; break;
            case 51 : i=46; break;
            case 52 : i=47; break;
            case 53 : i=48; break;
            case 55 : i=49; break;
            case 56 : i=50; break;
            case 57 : i=51; break;
            case 59 : i=52; break;
            case 60 : i=53; break;
            case 61 : i=54; break;
            case 62 : i=55; break;
            case 63 : i=56; break;
            case 64 : i=57; break;
            case 65 : i=58; break;
            case 66 : i=59; break;
            case 67 : i=60; break;
            case 68 : i=61; break;
            case 69 : i=62; break;
            case 70 : i=63; break;
            case 71 : i=64; break;
            case 72 : i=65; break;
            case 73 : i=66; break;
            case 74 : i=67; break;
            case 75 : i=68; break;
            case 76 : i=69; break;
            case 77 : i=70; break;
            default : i=71; break;
            }
    }
    return(i);
}
void wayswitch(USMSG *q,double hum[],int flag[])
{
    switch(q->way)//对路径还是对主题的判断依据
    {
        case 101001 : if(flag[0]==0){hum[0]++; flag[0]=1;}break;
        case 101002 : if(flag[1]==0){hum[1]++; flag[1]=1;}break;
        case 101003 : if(flag[2]==0){hum[2]++; flag[2]=1;}break;
        case 101004 : if(flag[3]==0){hum[3]++; flag[3]=1;}break;
        case 101005 : if(flag[4]==0){hum[4]++; flag[4]=1;}break;
        case 101006 : if(flag[5]==0){hum[5]++; flag[5]=1;}break;
        case 101007 : if(flag[6]==0){hum[6]++; flag[6]=1;}break;
        case 101008 : if(flag[7]==0){hum[7]++; flag[7]=1;}break;
```

```
            case 101009 : if(flag[8]==0){hum[8]++; flag[8]=1;}break;
            case 102001 : if(flag[9]==0){hum[9]++; flag[9]=1;}break;
            case 102002 : if(flag[10]==0){hum[10]++; flag[10]=1;}break;
            case 102003 : if(flag[11]==0){hum[11]++; flag[11]=1;}break;
            case 102004 : if(flag[12]==0){hum[12]++; flag[12]=1;}break;
            case 102005 : if(flag[13]==0){hum[13]++; flag[13]=1;}break;
            case 102006 : if(flag[14]==0){hum[14]++; flag[14]=1;}break;
            case 102007 : if(flag[15]==0){hum[15]++; flag[15]=1;}break;
            case 102008 : if(flag[16]==0){hum[16]++; flag[16]=1;}break;
            case 102009 : if(flag[17]==0){hum[17]++; flag[17]=1;}break;
            case 103001 : if(flag[18]==0){hum[18]++; flag[18]=1;}break;
            case 103002 : if(flag[19]==0){hum[19]++; flag[19]=1;}break;
            case 103003 : if(flag[20]==0){hum[20]++; flag[20]=1;}break;
            case 104001 : if(flag[21]==0){hum[21]++; flag[21]=1;}break;
            case 105001 : if(flag[22]==0){hum[22]++; flag[22]=1;}break;
            case 106001 : if(flag[23]==0){hum[23]++; flag[23]=1;}break;
            case 107001 : if(flag[24]==0){hum[24]++; flag[24]=1;}break;
            case 107002 : if(flag[25]==0){hum[25]++; flag[25]=1;}break;
            case 107003 : if(flag[26]==0){hum[26]++; flag[26]=1;}break;
            case 201001 : if(flag[27]==0){hum[27]++; flag[27]=1;}break;
            case 301001 : if(flag[28]==0){hum[28]++; flag[28]=1;}break;
            default : if(flag[29]== 0){hum[29]++; flag[29]=1;}break;
        }
}
void aimswitch(USMSG *q,double hum[],int flag[])
{
    switch(q->aim)
    {
            case 1 : if(flag[0]==0){hum[0]++; flag[0]=1;} break;
            case 2 : if(flag[1]==0){hum[1]++; flag[1]=1;} break;
            case 3 : if(flag[2]==0){hum[2]++; flag[2]=1;} break;
            case 4 : if(flag[3]==0){hum[3]++; flag[3]=1;} break;
            case 5 : if(flag[4]==0){hum[4]++; flag[4]=1;} break;
            case 6 : if(flag[5]==0){hum[5]++; flag[5]=1;} break;
            case 7 : if(flag[6]==0){hum[6]++; flag[6]=1;} break;
            case 8 : if(flag[7]==0){hum[7]++; flag[7]=1;} break;
            case 9 : if(flag[8]==0){hum[8]++; flag[8]=1;} break;
            case 10 : if(flag[9]==0){hum[9]++; flag[9]=1;} break;
            case 11 : if(flag[10]==0){hum[10]++; flag[10]=1;} break;
            case 12 : if(flag[11]==0){hum[11]++; flag[11]=1;} break;
            case 13 : if(flag[12]==0){hum[12]++; flag[12]=1;} break;
            case 15 : if(flag[13]==0){hum[13]++; flag[13]=1;} break;
            case 17 : if(flag[14]==0){hum[14]++; flag[14]=1;} break;
            case 18 : if(flag[15]==0){hum[15]++; flag[15]=1;} break;
```

```
case 19 : if(flag[16]==0){hum[16]++; flag[16]=1;} break;
case 20 : if(flag[17]==0){hum[17]++; flag[17]=1;} break;
case 21 : if(flag[18]==0){hum[18]++; flag[18]=1;} break;
case 22 : if(flag[19]==0){hum[19]++; flag[19]=1;} break;
case 23 : if(flag[20]==0){hum[20]++; flag[20]=1;} break;
case 25 : if(flag[21]==0){hum[21]++; flag[21]=1;} break;
case 26 : if(flag[22]==0){hum[22]++; flag[22]=1;} break;
case 27 : if(flag[23]==0){hum[23]++; flag[23]=1;} break;
case 28 : if(flag[24]==0){hum[24]++; flag[24]=1;} break;
case 29 : if(flag[25]==0){hum[25]++; flag[25]=1;} break;
case 30 : if(flag[26]==0){hum[26]++; flag[26]=1;} break;
case 31 : if(flag[27]==0){hum[27]++; flag[27]=1;} break;
case 32 : if(flag[28]==0){hum[28]++; flag[28]=1;} break;
case 33 : if(flag[29]==0){hum[29]++; flag[29]=1;} break;
case 34 : if(flag[30]==0){hum[30]++; flag[30]=1;} break;
case 35 : if(flag[31]==0){hum[31]++; flag[31]=1;} break;
case 36 : if(flag[32]==0){hum[32]++; flag[32]=1;} break;
case 37 : if(flag[33]==0){hum[33]++; flag[33]=1;} break;
case 38 : if(flag[34]==0){hum[34]++; flag[34]=1;} break;
case 39 : if(flag[35]==0){hum[35]++; flag[35]=1;} break;
case 40 : if(flag[36]==0){hum[36]++; flag[36]=1;} break;
case 41 : if(flag[37]==0){hum[37]++; flag[37]=1;} break;
case 42 : if(flag[38]==0){hum[38]++; flag[38]=1;} break;
case 43 : if(flag[39]==0){hum[39]++; flag[39]=1;} break;
case 44 : if(flag[40]==0){hum[40]++; flag[40]=1;} break;
case 45 : if(flag[41]==0){hum[41]++; flag[41]=1;} break;
case 46 : if(flag[42]==0){hum[42]++; flag[42]=1;} break;
case 47 : if(flag[43]==0){hum[43]++; flag[43]=1;} break;
case 48 : if(flag[44]==0){hum[44]++; flag[44]=1;} break;
case 49 : if(flag[45]==0){hum[45]++; flag[45]=1;} break;
case 51 : if(flag[46]==0){hum[46]++; flag[46]=1;} break;
case 52 : if(flag[47]==0){hum[47]++; flag[47]=1;} break;
case 53 : if(flag[48]==0){hum[48]++; flag[48]=1;} break;
case 55 : if(flag[49]==0){hum[49]++; flag[49]=1;} break;
case 56 : if(flag[50]==0){hum[50]++; flag[50]=1;} break;
case 57 : if(flag[51]==0){hum[51]++; flag[51]=1;} break;
case 59 : if(flag[52]==0){hum[52]++; flag[52]=1;} break;
case 60 : if(flag[53]==0){hum[53]++; flag[53]=1;} break;
case 61 : if(flag[54]==0){hum[54]++; flag[54]=1;} break;
case 62 : if(flag[55]==0){hum[55]++; flag[55]=1;} break;
case 63 : if(flag[56]==0){hum[56]++; flag[56]=1;} break;
case 64 : if(flag[57]==0){hum[57]++; flag[57]=1;} break;
case 65 : if(flag[58]==0){hum[58]++; flag[58]=1;} break;
case 66 : if(flag[59]==0){hum[59]++; flag[59]=1;} break;
```

```
            case 67 : if(flag[60]==0){hum[60]++; flag[60]=1;} break;
            case 68 : if(flag[61]==0){hum[61]++; flag[61]=1;} break;
            case 69 : if(flag[62]==0){hum[62]++; flag[62]=1;} break;
            case 70 : if(flag[63]==0){hum[63]++; flag[63]=1;} break;
            case 71 : if(flag[64]==0){hum[64]++; flag[64]=1;} break;
            case 72 : if(flag[65]==0){hum[65]++; flag[65]=1;} break;
            case 73 : if(flag[66]==0){hum[66]++; flag[66]=1;} break;
            case 74 : if(flag[67]==0){hum[67]++; flag[67]=1;} break;
            case 75 : if(flag[68]==0){hum[68]++; flag[68]=1;} break;
            case 76 : if(flag[69]==0){hum[69]++; flag[69]=1;} break;
            case 77 : if(flag[70]==0){hum[70]++; flag[70]=1;} break;
            default : if(flag[71]==0){hum[71]++; flag[71]=1;} break;
        }
}
void timeswitch(USMSG *q,double hum[],int flag[])
{
    int hour,i;
    hour=(q->time%86400000)/3600000;
    if(hour>=0&&hour<3)
    {
        i=2;
    }
    else if(hour>=3&&hour<5)
    {
        i=3;
    }
    else if(hour>=5&&hour<9)
    {
        i=4;
    }
    else if(hour>=9&&hour<11)
    {
        i=5;
    }
    else if(hour>=11&&hour<12)
    {
        i=6;
    }
    else if(hour>=12&&hour<16)
    {
        i=7;
    }
    else if(hour>=16&&hour<21)
    {
```

```
            i=0;
        }
        else
        {
            i=1;
        }
        if(flag[i]==0)
        {
            hum[i]++;
            flag[i]=1;
        }
}
```

附录四：Apriori 算法统计相关度

```
#include <stdio.h>
#include <string.h>
#define USER struct user
#define USMSG struct usmsg
USMSG
{
    int aeraid;
    unsigned long long int time;
    int way;
    int aim;
    USMSG *next;
};
USER
{
    unsigned int usrid;
    USER *next;
    USMSG *msghead;
};

USER *readdat(FILE *fp,FILE *frule);
void layer1(USER *head,double l1[]);
void layer2(USER *head,double l2[72][72]);
int tihuantime(USMSG *q);
int tihuan(int a);
void wayswitch(USMSG *q,double hum[],int flag[]);
void aimswitch(USMSG *q,double hum[],int flag[]);
void timeswitch(USMSG *q,double hum[],int flag[]);


int main()
```

```
{
    FILE *fp,*frule;
    USER *simp_hand;
    double l1[72],l2[72][72];
    fp=fopen("E:\\mathmatic_module\\data\\user_new.dat","r");
    frule=fopen("E:\\mathmatic_module\\data\\rule_new.pi","r");
    if(fp==NULL||frule==NULL)
    {
        printf("文件打开错误！\n");
        exit(0);
    }
    simp_hand=readdat(fp,frule);
    fclose(fp);
    fclose(frule);
//  layer1(simp_hand,l1);
    layer2(simp_hand,l2);
    return(0);
}

USER *readdat(FILE *fp,FILE *frule)
{
    int person,i;
    int numb,j;
    USER *simp=NULL,*q=NULL,*simp_hand=NULL;
    USMSG *p=NULL,*lie=NULL;

    rewind(fp);
    rewind(frule);

    fscanf(frule,"%d",&person);
    for(i=0;i<person;i++)
    {
        fscanf(frule,"%d",&numb);
        q=(USER*)malloc(sizeof(USER));
        if(q==NULL)
        {
            printf("内存不足!\n");
            exit(0);
        }
        fscanf(fp,"%u",&q->usrid);
        q->next=NULL;
        q->msghead=NULL;
        for(j=0;j<numb;j++)
        {
```

```
                p=(USMSG*)malloc(sizeof(USMSG));
                if(p==NULL)
                {
                    printf("内存不足！\n");
                    exit(0);
                }
                fscanf(fp,"%d %llu %d %d",&p->aeraid,&p->time,&p->way,&p->aim);
                p->next=NULL;
                if(q->msghead==NULL)
                {
                    q->msghead=p;
                    lie=p;
                }
                else
                {
                    lie->next=p;
                    lie=p;
                }
            }
            if(simp_hand==NULL)
            {
                simp_hand=q;
                simp=q;
            }
            else
            {
                simp->next=q;
                simp=q;
            }
        }
        return (simp_hand);
}
void layer1(USER *head,double l1[])
{
        FILE *fp;
        USER *p;
        USMSG *q;
        double hum[8];
        int total=0,i,flag[8];
        p=head;
        for(i=0;i<8;i++)
        {
            hum[i]=0;
            flag[i]=0;
```

```
    }
    while(p!=NULL)
    {
        total++;
        for(i=0;i<8;i++)
        {
            flag[i]=0;
        }
        q=p->msghead;
        while(q!=NULL)
        {
        //  wayswitch(q,hum,flag);
        //  aimswitch(q,hum,flag);
            timeswitch(q,hum,flag);
            q=q->next;
        }
        p=p->next;
    }
    fp=fopen("E:\\mathmatic_module\\data\\L1 时间.txt","w");
    for(i=0;i<8;i++)
    {
        l1[i]=(hum[i]/total)*100;
        printf("%d\t%f\n",i,l1[i]);
        fprintf(fp,"%d\t%f\n",i,l1[i]);
    }
    fclose(fp);
}
void layer2(USER *head,double l2[72][72])
{
    FILE *fp;
    USER *p;
    USMSG *q,*temp;
    double hum[8][72];
    int total=0,i,j,flag[8][72];
    p=head;
    for(i=0;i<8;i++)
    {
        for(j=0;j<72;j++)
        {
            hum[i][j]=0;
            flag[i][j]=0;
        }
    }
    while(p!=NULL)
```

```
    {
        total++;
        for(i=0;i<8;i++)
        {
            for(j=0;j<72;j++)
            {
                flag[i][j]=0;
            }
        }
        q=p->msghead;
        while(q!=NULL)
        {
            temp=p->msghead;
            while(temp!=NULL)
            {

    if(flag[tihuantime(q)][tihuan(temp->aim)]==0)/*&&(flag[tihuan(temp->way)][tihuan(q-
>way)]==0)*/
                {
                    hum[tihuantime(q)][tihuan(temp->aim)]++;
                    flag[tihuantime(q)][tihuan(temp->aim)]=1;
                }
                temp=temp->next;
            }
            q=q->next;
        }
        p=p->next;
    }

    fp=fopen("L2 时间与页面类型.txt","w");
    for(i=0;i<8;i++)
    {
        fprintf(fp,"%d\t",i);
        for(j=0;j<72;j++)
        {
            l2[i][j]=hum[i][j]/total*100;
            printf("%e\t",l2[i][j]);
            fprintf(fp,"%f\t",l2[i][j]);
        }
        printf("\b\n");
        fprintf(fp,"\b\n");
    }
    fclose(fp);
}
```

```
int tihuantime(USMSG *q)
{
	int i,hour;
	hour=(q->time%86400000)/3600000;
	if(hour>=0&&hour<3)
	{
		i=2;
	}
	else if(hour>=3&&hour<5)
	{
		i=3;
	}
	else if(hour>=5&&hour<9)
	{
		i=4;
	}
	else if(hour>=9&&hour<11)
	{
		i=5;
	}
	else if(hour>=11&&hour<12)
	{
		i=6;
	}
	else if(hour>=12&&hour<16)
	{
		i=7;
	}
	else if(hour>=16&&hour<21)
	{
		i=0;
	}
	else
	{
		i=1;
	}
	return(i);
}
int tihuan(int a)
{
	int i;
	if(a>1000)
	{
		switch(a)
```

```
        {
        case 101001 : i=0; break;
        case 101002 : i=1;break;
        case 101003 : i=2;break;
        case 101004 : i=3;break;
        case 101005 : i=4;break;
        case 101006 : i=5;break;
        case 101007 : i=6; break;
        case 101008 : i=7;break;
        case 101009 : i=8;break;
        case 102001 : i=9;break;
        case 102002 : i=10;break;
        case 102004 : i=12;break;
        case 102005 : i=13;break;
        case 102006 : i=14;break;
        case 102007 : i=15;break;
        case 102008 : i=16;break;
        case 102009 : i=17;break;
        case 103001 : i=18;break;
        case 103002 : i=19;break;
        case 103003 : i=20;break;
        case 104001 : i=21;break;
        case 105001 : i=22;break;
        case 106001 : i=23;break;
        case 107001 : i=25;break;
        case 107002 : i=26;break;
        case 107003 : i=27;break;
        case 201001 : i=28;break;
        default : i=29;break;
        }
}
else
{
        switch(a)
        {
        case 1 : i=0; break;
        case 2 : i=1; break;
        case 3 : i=2; break;
        case 4 : i=3; break;
        case 5 : i=4; break;
        case 6 : i=5; break;
        case 7 : i=6; break;
        case 8 : i=7; break;
        case 9 : i=8; break;
```

```
case 10 : i=9; break;
case 11 : i=10; break;
case 12 : i=11; break;
case 13 : i=12; break;
case 15 : i=13; break;
case 17 : i=14; break;
case 18 : i=15; break;
case 19 : i=16; break;
case 20 : i=17; break;
case 21 : i=18; break;
case 22 : i=19; break;
case 23 : i=20; break;
case 25 : i=21; break;
case 26 : i=22; break;
case 27 : i=23; break;
case 28 : i=24; break;
case 29 : i=25; break;
case 30 : i=26; break;
case 31 : i=27; break;
case 32 : i=28; break;
case 33 : i=29; break;
case 34 : i=30; break;
case 35 : i=31; break;
case 36 : i=32; break;
case 37 : i=33; break;
case 38 : i=34; break;
case 39 : i=35; break;
case 40 : i=36; break;
case 41 : i=37; break;
case 42 : i=38; break;
case 43 : i=39; break;
case 44 : i=40; break;
case 45 : i=41; break;
case 46 : i=42; break;
case 47 : i=43; break;
case 48 : i=44; break;
case 49 : i=45; break;
case 51 : i=46; break;
case 52 : i=47; break;
case 53 : i=48; break;
case 55 : i=49; break;
case 56 : i=50; break;
case 57 : i=51; break;
case 59 : i=52; break;
```

```
            case 60 : i=53; break;
            case 61 : i=54; break;
            case 62 : i=55; break;
            case 63 : i=56; break;
            case 64 : i=57; break;
            case 65 : i=58; break;
            case 66 : i=59; break;
            case 67 : i=60; break;
            case 68 : i=61; break;
            case 69 : i=62; break;
            case 70 : i=63; break;
            case 71 : i=64; break;
            case 72 : i=65; break;
            case 73 : i=66; break;
            case 74 : i=67; break;
            case 75 : i=68; break;
            case 76 : i=69; break;
            case 77 : i=70; break;
            default : i=71; break;
            }
    }
    return(i);
}
void wayswitch(USMSG *q,double hum[],int flag[])
{
    switch(q->way)//对路径还是对主题的判断依据
    {
        case 101001 : if(flag[0]==0){hum[0]++; flag[0]=1;}break;
        case 101002 : if(flag[1]==0){hum[1]++; flag[1]=1;}break;
        case 101003 : if(flag[2]==0){hum[2]++; flag[2]=1;}break;
        case 101004 : if(flag[3]==0){hum[3]++; flag[3]=1;}break;
        case 101005 : if(flag[4]==0){hum[4]++; flag[4]=1;}break;
        case 101006 : if(flag[5]==0){hum[5]++; flag[5]=1;}break;
        case 101007 : if(flag[6]==0){hum[6]++; flag[6]=1;}break;
        case 101008 : if(flag[7]==0){hum[7]++; flag[7]=1;}break;
        case 101009 : if(flag[8]==0){hum[8]++; flag[8]=1;}break;
        case 102001 : if(flag[9]==0){hum[9]++; flag[9]=1;}break;
        case 102002 : if(flag[10]==0){hum[10]++; flag[10]=1;}break;
        case 102003 : if(flag[11]==0){hum[11]++; flag[11]=1;}break;
        case 102004 : if(flag[12]==0){hum[12]++; flag[12]=1;}break;
        case 102005 : if(flag[13]==0){hum[13]++; flag[13]=1;}break;
        case 102006 : if(flag[14]==0){hum[14]++; flag[14]=1;}break;
        case 102007 : if(flag[15]==0){hum[15]++; flag[15]=1;}break;
        case 102008 : if(flag[16]==0){hum[16]++; flag[16]=1;}break;
```

```
            case 102009 : if(flag[17]==0){hum[17]++; flag[17]=1;}break;
            case 103001 : if(flag[18]==0){hum[18]++; flag[18]=1;}break;
            case 103002 : if(flag[19]==0){hum[19]++; flag[19]=1;}break;
            case 103003 : if(flag[20]==0){hum[20]++; flag[20]=1;}break;
            case 104001 : if(flag[21]==0){hum[21]++; flag[21]=1;}break;
            case 105001 : if(flag[22]==0){hum[22]++; flag[22]=1;}break;
            case 106001 : if(flag[23]==0){hum[23]++; flag[23]=1;}break;
            case 107001 : if(flag[24]==0){hum[24]++; flag[24]=1;}break;
            case 107002 : if(flag[25]==0){hum[25]++; flag[25]=1;}break;
            case 107003 : if(flag[26]==0){hum[26]++; flag[26]=1;}break;
            case 201001 : if(flag[27]==0){hum[27]++; flag[27]=1;}break;
            case 301001 : if(flag[28]==0){hum[28]++; flag[28]=1;}break;
            default : if(flag[29]== 0){hum[29]++; flag[29]=1;}break;
        }
}
void aimswitch(USMSG *q,double hum[],int flag[])
{
    switch(q->aim)
    {
            case 1 : if(flag[0]==0){hum[0]++; flag[0]=1;} break;
            case 2 : if(flag[1]==0){hum[1]++; flag[1]=1;} break;
            case 3 : if(flag[2]==0){hum[2]++; flag[2]=1;} break;
            case 4 : if(flag[3]==0){hum[3]++; flag[3]=1;} break;
            case 5 : if(flag[4]==0){hum[4]++; flag[4]=1;} break;
            case 6 : if(flag[5]==0){hum[5]++; flag[5]=1;} break;
            case 7 : if(flag[6]==0){hum[6]++; flag[6]=1;} break;
            case 8 : if(flag[7]==0){hum[7]++; flag[7]=1;} break;
            case 9 : if(flag[8]==0){hum[8]++; flag[8]=1;} break;
            case 10 : if(flag[9]==0){hum[9]++; flag[9]=1;} break;
            case 11 : if(flag[10]==0){hum[10]++; flag[10]=1;} break;
            case 12 : if(flag[11]==0){hum[11]++; flag[11]=1;} break;
            case 13 : if(flag[12]==0){hum[12]++; flag[12]=1;} break;
            case 15 : if(flag[13]==0){hum[13]++; flag[13]=1;} break;
            case 17 : if(flag[14]==0){hum[14]++; flag[14]=1;} break;
            case 18 : if(flag[15]==0){hum[15]++; flag[15]=1;} break;
            case 19 : if(flag[16]==0){hum[16]++; flag[16]=1;} break;
            case 20 : if(flag[17]==0){hum[17]++; flag[17]=1;} break;
            case 21 : if(flag[18]==0){hum[18]++; flag[18]=1;} break;
            case 22 : if(flag[19]==0){hum[19]++; flag[19]=1;} break;
            case 23 : if(flag[20]==0){hum[20]++; flag[20]=1;} break;
            case 25 : if(flag[21]==0){hum[21]++; flag[21]=1;} break;
            case 26 : if(flag[22]==0){hum[22]++; flag[22]=1;} break;
            case 27 : if(flag[23]==0){hum[23]++; flag[23]=1;} break;
            case 28 : if(flag[24]==0){hum[24]++; flag[24]=1;} break;
```

```
case 29 : if(flag[25]==0){hum[25]++; flag[25]=1;} break;
case 30 : if(flag[26]==0){hum[26]++; flag[26]=1;} break;
case 31 : if(flag[27]==0){hum[27]++; flag[27]=1;} break;
case 32 : if(flag[28]==0){hum[28]++; flag[28]=1;} break;
case 33 : if(flag[29]==0){hum[29]++; flag[29]=1;} break;
case 34 : if(flag[30]==0){hum[30]++; flag[30]=1;} break;
case 35 : if(flag[31]==0){hum[31]++; flag[31]=1;} break;
case 36 : if(flag[32]==0){hum[32]++; flag[32]=1;} break;
case 37 : if(flag[33]==0){hum[33]++; flag[33]=1;} break;
case 38 : if(flag[34]==0){hum[34]++; flag[34]=1;} break;
case 39 : if(flag[35]==0){hum[35]++; flag[35]=1;} break;
case 40 : if(flag[36]==0){hum[36]++; flag[36]=1;} break;
case 41 : if(flag[37]==0){hum[37]++; flag[37]=1;} break;
case 42 : if(flag[38]==0){hum[38]++; flag[38]=1;} break;
case 43 : if(flag[39]==0){hum[39]++; flag[39]=1;} break;
case 44 : if(flag[40]==0){hum[40]++; flag[40]=1;} break;
case 45 : if(flag[41]==0){hum[41]++; flag[41]=1;} break;
case 46 : if(flag[42]==0){hum[42]++; flag[42]=1;} break;
case 47 : if(flag[43]==0){hum[43]++; flag[43]=1;} break;
case 48 : if(flag[44]==0){hum[44]++; flag[44]=1;} break;
case 49 : if(flag[45]==0){hum[45]++; flag[45]=1;} break;
case 51 : if(flag[46]==0){hum[46]++; flag[46]=1;} break;
case 52 : if(flag[47]==0){hum[47]++; flag[47]=1;} break;
case 53 : if(flag[48]==0){hum[48]++; flag[48]=1;} break;
case 55 : if(flag[49]==0){hum[49]++; flag[49]=1;} break;
case 56 : if(flag[50]==0){hum[50]++; flag[50]=1;} break;
case 57 : if(flag[51]==0){hum[51]++; flag[51]=1;} break;
case 59 : if(flag[52]==0){hum[52]++; flag[52]=1;} break;
case 60 : if(flag[53]==0){hum[53]++; flag[53]=1;} break;
case 61 : if(flag[54]==0){hum[54]++; flag[54]=1;} break;
case 62 : if(flag[55]==0){hum[55]++; flag[55]=1;} break;
case 63 : if(flag[56]==0){hum[56]++; flag[56]=1;} break;
case 64 : if(flag[57]==0){hum[57]++; flag[57]=1;} break;
case 65 : if(flag[58]==0){hum[58]++; flag[58]=1;} break;
case 66 : if(flag[59]==0){hum[59]++; flag[59]=1;} break;
case 67 : if(flag[60]==0){hum[60]++; flag[60]=1;} break;
case 68 : if(flag[61]==0){hum[61]++; flag[61]=1;} break;
case 69 : if(flag[62]==0){hum[62]++; flag[62]=1;} break;
case 70 : if(flag[63]==0){hum[63]++; flag[63]=1;} break;
case 71 : if(flag[64]==0){hum[64]++; flag[64]=1;} break;
case 72 : if(flag[65]==0){hum[65]++; flag[65]=1;} break;
case 73 : if(flag[66]==0){hum[66]++; flag[66]=1;} break;
case 74 : if(flag[67]==0){hum[67]++; flag[67]=1;} break;
case 75 : if(flag[68]==0){hum[68]++; flag[68]=1;} break;
```

```
            case 76 : if(flag[69]==0){hum[69]++; flag[69]=1;} break;
            case 77 : if(flag[70]==0){hum[70]++; flag[70]=1;} break;
            default : if(flag[71]==0){hum[71]++; flag[71]=1;} break;
        }
}
void timeswitch(USMSG *q,double hum[],int flag[])
{
    int hour,i;
    hour=(q->time%86400000)/3600000;
    if(hour>=0&&hour<3)
    {
        i=2;
    }
    else if(hour>=3&&hour<5)
    {
        i=3;
    }
    else if(hour>=5&&hour<9)
    {
        i=4;
    }
    else if(hour>=9&&hour<11)
    {
        i=5;
    }
    else if(hour>=11&&hour<12)
    {
        i=6;
    }
    else if(hour>=12&&hour<16)
    {
        i=7;
    }
    else if(hour>=16&&hour<21)
    {
        i=0;
    }
    else
    {
        i=1;
    }
    if(flag[i]==0)
    {
        hum[i]++;
```

```
            flag[i]=1;
        }
}
```