Python 数据分析与数据挖掘（Python for Data Analysis&Data Mining）

Chap 14 推荐系统（Recommender）

内容：

-
- 问题背景
- 数据探索分析和预处理
- 推荐系统概述
- 基于协同过滤的推荐系统
- 推荐系统中的几个重要问题
- 应用：个性化推荐、相关推荐、热门推荐等，可以应用到很多领域，例如电商、社交网络、智能交通、城市规划与发展、旅游等；推荐算法有基于内容推荐、协同过滤推荐、基于关联规则推荐、基于效用推荐、基于知识推荐、组合推荐等。

实践：

- Python访问数据库的准备
  - XAMPP的安装
  - PyMySQL：conda install pymysql
  - SQLAlchemy: conda install sqlalchemy
- 了解Pandas读取数据库的方法和操作
- 推荐系统实现

实例：

- 实例1：MySQL数据库安装
- 实例2：Python访问MySQL数据库
- 实例3：基于协同过滤的推荐系统实现

安装访问MySQL数据库的Python库

- PyMySQL：pip install pymysql
- SQLAlchemy: pip install sqlalchemy

这节课是在前面数据分析和数据挖掘的基础上，针对数据库存放的数据进行面向应用的探索分析和预处理处理，并结合实际数据和应用来构建基于协同过滤的推荐系统。本节课通过对用户访问网站的行为（本实例中浏览网页与否）数据进行分析和探索，估计物品（实例中的网页）之间的相似度，实现基于协同过滤的推荐算法系统对用户进行个性化的网页推荐，从而提升用户的使用体验和满意度。本节课虽然从用户浏览网页的行为数据分析，也可以扩展到其他用户行为（如是否购买、是否评论、评分、是否点赞、浏览时间长短等）；针对物品的相似度估计，本实例是根据网页的类别，也可以要结合具体的业务扩展到其他物品（产品、电影、餐饮等），并采用自然语言处理和文本挖掘等技术进行相似度语义分析；此外，在实际应用中，也有一些重要的问题需要有针对性地聪明解决。

准备工作：导入库，配置环境等

In [ ]:

```
# -*- coding: utf-8 -*-
from __future__ import division
import os, sys

# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1. 问题背景

随着互联网和信息技术的快速发展，电子商务、网上服务与交易等网络业务越来越普及，并产生了海量的信息。用户想要从海量信息中快速准确寻找到自己感兴趣的信息变得越来越困难（信息过载）。搜索引擎在一定程度上缓解了信息过载，对于用户输入的关键词，搜索引擎返回与输入的关键词相关的信息，但对于用户想找到准确描述自己需求的关键词，则搜索引擎就无法解决。

与搜索引擎不同，推荐系统并不需要用户提供明确的需求，而是通过分析用户的历史行为，从而主动向用户推荐能够满足他们兴趣和需求的信息。因此，对于用户而言，推荐系统和搜索引擎是两个互补的工具。搜索引擎满足有明确目标用户的需求，而推荐系统能够帮助用户发现其感兴趣的内容。

因此，在电商领域中推荐技术可以起到以下作用：

1. 帮助用户发现其感兴趣的物品，节省用户时间、提升用户体验；
2. 提高用户对电商平台的忠诚度，如果推荐系统能够准确地发现用户的兴趣点，并将合适的资源推荐给用户，用户就会对该电商平台产生依赖，从而建立稳定的企业忠实顾客群。

电商推荐应用示例

用户在当当网站购买了下面的一本书：

图 1. 用户购买一本书的记录

相应的，当当网站给出了“买过购物车中商品的用户还买了”如图的其他书：

图 2. 相关推荐的记录

本实例的研究对象是一家电商类的大型法律资讯网站，致力于为用户提供丰富的法律信息和专业咨询服务，并为律师与律师事务所提供卓有成效的互联网整合营销解决方案。

随着网站访问量增大，信息量也在大幅增长。用户在面对大量信息时无法及时从中获得自己需要的信息，对信息的使用效率越来越低。这种浏览大量无关信息的过程，使用户需要花费大量的时间才能找到自己需要的信息，从而使得用户不断流失，给企业造成巨大的损失。

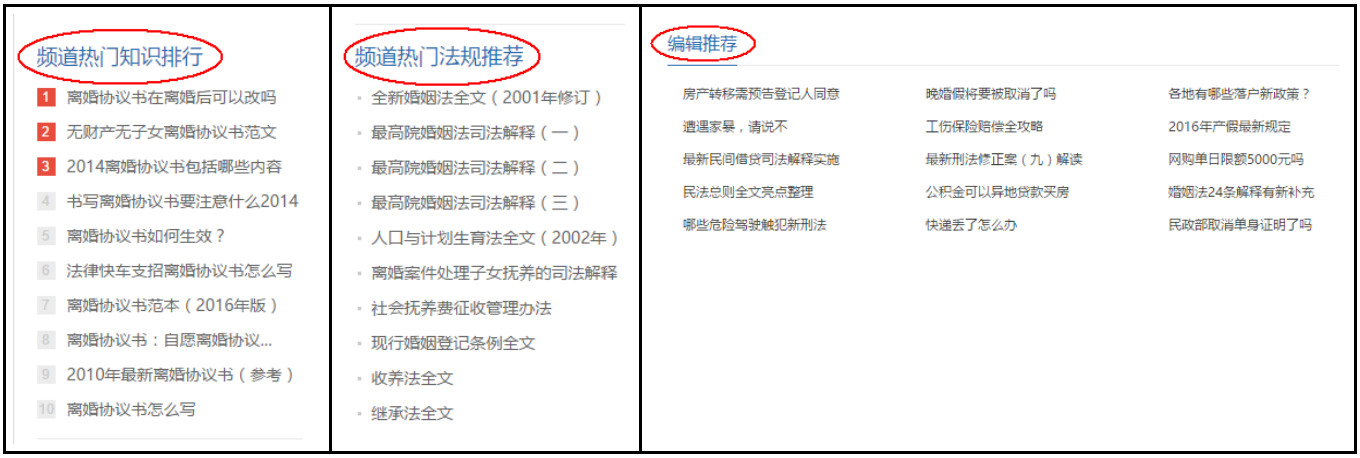
为了能够更好地满足用户需求，依据其网站海量的数据，研究用户的兴趣偏好，分析用户的需求和行为，发现用户的兴趣点，从而引导用户发现自己的信息需求，将长尾网页准确地推荐给所需用户，帮助用户发现他们感兴趣但很难发现的网页信息。为用户提供个性化的服务，并且建立网站与用户之间的密切关系，让用户对推荐系统产生依赖，从而建立稳定的企业忠实顾客群，实现客户链式反应增值，提高消费者满意度。通过提高服务效率帮助消费者节约交易成本等，制定有针对性的营销战略方针，促进企业长期稳定高速发展。

咨询推荐应用示例

首先，用户浏览了一条与某法律相关的一个知识页面，如下：

然后，网站提供了与这个浏览内容相关的文章、咨询和专业律师类别的个性化推荐如下：

此外，网站还提供了相关知识、法规的热门推荐以及编辑推荐如下：



2. 挖掘目标

1. 按地域研究用户访问时间、访问内容和访问次数等分析主题，深入了解用户对访问网站的行为和目的及关心的内容。
2. 借助大量的用户的访问记录，发现用户的访问行为习惯，对不同需求的用户进行相关的服务页面的推荐。

3. 分析方法

1. 原始数据的获得

当用户访问网站页面时，系统会记录用户访问网站的日志，包含用户IP（做数据脱敏处理）、用户访问的时间、访问内容等多项属性的记录。一条用户访问网站的记录的原始数据（已存入MySQL数据库）如下：

realIP 真实ip	realAreacode 地区编号	userAgent 浏览器代理	userOS 用户浏览器类型	userID userid	clientID clientid	timestamp 时间戳	timestamp_format 格式化的时间
308351962	140106	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1;...	Windows XP	1597050740.1422973305	1597050740.1422973305	1422973282739	2015-02-03 22:21:22

pagePath 路径	ymd	fullURL	fullURLId	hostname	pageTitle
/ask/question_5281741.html	20150203	http://www.lawtime.cn/ask/question_5281741.html	101003	www.lawtime.cn	交通事故销案后不满意赔偿可以重新立案吗 - 法律快车法律咨询

pageTitleCategoryId	pageTitleCategoryNam	pageTitleKv	fullReferrerURL	organicKeyword	fullReferrer	source
12	伤害赔偿	法律咨询	http://www.baidu.com/s?word=%E4%BA%A4%E9%	交通事故赔偿后交警要销案吗	baidu	baidu

其中，原始数据中各个属性说明情况如下：

属性名称	属性说明	属性名称	属性说明
realIP	真实 ip	fullURLId	网址类型
realAreacode	地区编号	hostname	源地址名
userAgent	浏览器代理	pageTitle	网页标题
userOS	用户浏览器类型	pageTitleCategoryId	标题类型 ID
userID	用户 ID	pageTitleCategoryName	标题类型名称
clientID	客户端 ID	pageTitleKw	标题类型关键字
timestamp	时间戳	fullReferrer	入口源
timestamp_format	标准化时间	fullReferrerURL	入口网址
pagePath	路径	organicKeyword	搜索关键字
ymd	年月日	source	搜索源
fullURL	网址		

2. 分析思路

本实例的目标是对用户进行推荐，即以一定的方式将用户与物品（这里是网页）之间建立联系，更好地帮助用户从海量的数据中快速发现感兴趣的网页。

由于用户访问网站的数据记录很多，如果不对数据进行分类处理，对所有记录直接采用推荐系统进行推荐，会存在以下问题：

1. 数据量太大意味着物品数量和用户数量很多，在模型构建用户与物品的兴趣矩阵时，矩阵很稀疏，模型计算需要消耗大量的内存空间和计算时间
2. 用户区别很大，不同用户关注的信息不一样，即使能够得到推荐结果，其推荐效果也不会好

怎么办？-- 数据分类

1. 页面类型的分类：

> 结合业务了解页面的类型，如咨询相关、知识相关、法律法规、律师相关、其他方面等

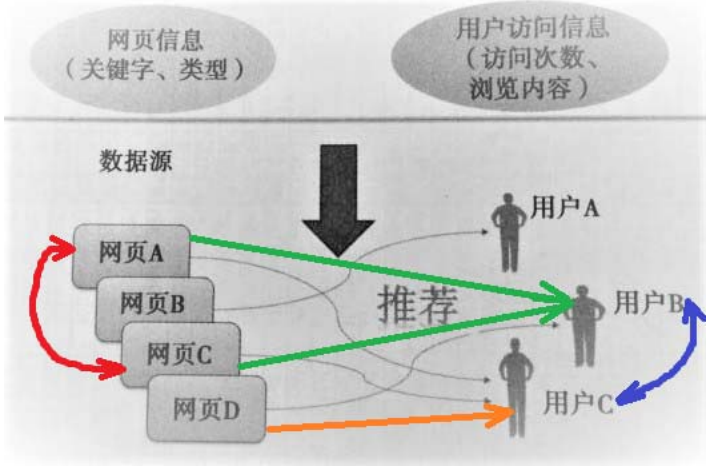
2. 用户类型的分类

> 正常情况下，需要对用户的兴趣爱好以及需求进行分类。在用户访问记录中，没有用户的人口属性，也没有记录用户访问网页时间的长短，因此，不容易判断用户的兴趣爱好。

本实例根据用户浏览的网页信息进行页面分类处理，根据用户浏览页面的次数进行点击次数分析，然后对每个类型中的内容进行推荐。怎么推荐？

3. 基于协同过滤的推荐算法原理

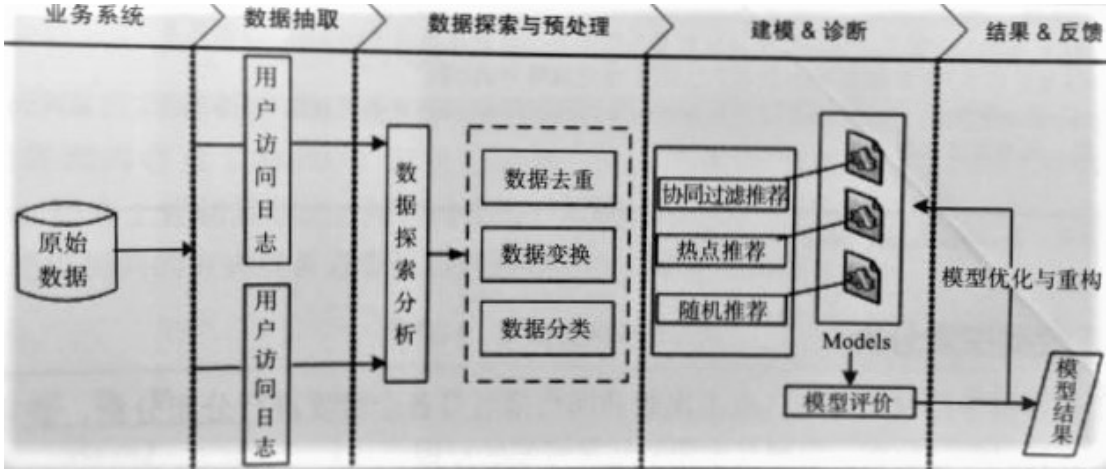
基于协同过滤的推荐算法的原理如图：



协同过滤算法的特性就是通过历史数据找出相似的用户或物品（页面），因此，需要选择尽可能大量的数据，这样能降低推荐结果的随机性，提高推荐结果的准确性，更好地发掘长尾网页中用户感兴趣的网页。

4. 系统流程

采用上述的分析方法与思路，结合本实例的原始数据以及分析目标，推荐系统流程图如下：



这个流程图的分析过程主要包含以下内容：

1. 从系统中获取用户访问网站的原始记录；
2. 对数据进行多维度分析，包括用户访问内容，流失用户分析以及用户分类等分析；
3. 对数据进行预处理，包含数据去重、数据变换和数据分类等处理过程；
4. 以用户访问html后缀的网页为关键条件，对数据进行处理；
5. 对比多种推荐算法进行推荐，通过模型评价，得到好的智能推荐模型；运行模型对测试样本数据进行推荐预测，获得推荐结果。

4. 实践过程

实践过程是：建立数据库 -> 导入数据 -> 搭建Python的数据库操作环境 -> 数据分析 -> 建立模型

1. 数据库准备

1. Pandas库本身可以利用 read_sql() 函数来读取数据库，但是它依赖于 SQLAlchemy库，SQLAlchemy 又依赖于 PyMySQL，所以需要先安装这两个库
2. 下载安装使用MySQL数据库，下载安装导入数据见讲义文件
3. 导入数据 data/lawdata.sql

安装完成后，可以通过Python连接到数据库。为了方便处理，使用Pandas的 read_sql() 函数来读取数据库。注意：Pandas在读取数据（不管是之前的csv，Excel或者现在的sql），都是讲全部数据读入内存中，因此，在数据量较大时是难以实现的。幸运地是，Pandas也提供了chunksize参数，可以分块读取大数据文件。具体见下面的代码。

2. 数据探索分析

1）网页类型的统计分析

针对原始数据中用户点击的网页类型进行统计，网页类型是指“网址类型”fullURLId （包含6-7位数字）中的前3位数字。

这个统计处理的意义是要“分块进行”，这样使得使用多线程甚至分布式计算有了可能。

In [ ]:

```
# 连接数据库
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('all_gzdata', engine, chunksize=10000)
#'''
#用create_engine建立连接，连接地址的意思依次为“数据库格式（mysql）+程序名（pymysql）+账号密码@地址端口/数据库名（test）”，最后指定编码为utf8；
#all_gzdata是表名，engine是连接数据的引擎，chunksize指定每次读取1万条记录。这时候sql是一个容器，未真正读取数据。这里账户是root，无密码
#'''
```

In [ ]:

```
counts = [ i['fullURLId'].value_counts() for i in sql] #逐块统计
counts = pd.concat(counts).groupby(level=0).sum() #合并统计结果，把相同的统计项合并（即按index分组并求和）
print counts
counts = counts.reset_index() #重新设置index，将原来的index作为counts的一列。
counts.columns = ['index', 'num'] #重新设置列名，主要是第二列，默认为0
print counts
counts['type'] = counts['index'].str.extract('(\d{3})',expand=True) #提取前三个数字作为网址类别id
print counts.head()
counts_ = counts[['type', 'num']].groupby('type').sum() #按类别合并
print counts_
counts_.sort_values(by='num', ascending = False) #降序排列
print counts_['num'].sum()
```

In [ ]:

```
print counts.head()
print counts_.head()
```

2）观察与分析

全部的记录条数为 386476 条，各种网页类型的分布情况统计如下表：

网页类型	记录数	百分比	内容
101	190151	49.20	与咨询相关（浏览咨询页面或者进行咨询等），http://www.XXX.com/ask/ (http://www.XXX.com/ask/)
199	92946	24.05	其他方面的网页
107	84202	21.79	知识相关，http://www.XXX.com/info/ (http://www.XXX.com/info/)
102	8254	2.14	律师相关的
301	8007	2.07	法规

1. 统计分析 -- 咨询类型页面101

1. 点击 与咨询相关的记录约占 49.20%；其他类型网页占24.05%， 知识相关页面占22%， ==> 用户点击的排行榜为：咨询相关、知识相关、其他方面的网页、法规、律师相关； ==> 初步得出：相对于长篇的知识，用户更加偏向于查看咨询或者进行咨询。
2. 进一步对咨询类别内部进行统计分析，结果见下表

网页类型	记录数	百分比	内容
101003	183197	96.34	咨询内容页
101002	3641	1.91	咨询列表页
101001	2523	1.33	咨询首页
101004 - 101009	790	0.42	其他

2. 统计分析 -- 知识类型页面107001

接下来，我们进一步统计分析知识类型页面107001（84202个页面）的点击情况， 知识类型只有一种，107001， ==> 利用网址对其进行分类（不能抽取最前面的3位数字），观察发现：

- 知识内容页（http://www.XXX.com/info/*/数字.html (http://www.XXX.com/info/*/数字.html)），其中数字部分可能带有下划线_
- 知识首页（http://www.XXX.com/info/*/ (http://www.XXX.com/info/*/)）
- 知识列表页（http://www.XXX.com/info/*.html (http://www.XXX.com/info/*.html)），即除了知识内容页外的html页面

In [ ]:

```
#统计107类别的情况，使用正则表达式
def count107(i): #自定义统计函数
    j = i[['fullURL']][i['fullURLId'].str.contains('107')].copy() #找出类别包含107的网址
    j['type'] = None #添加空列
    j['type'][j['fullURL'].str.contains('info/.+?/')] = u'知识首页'
    j['type'][j['fullURL'].str.contains('info/.+?/.+?')] = u'知识列表页'
    j['type'][j['fullURL'].str.contains('/\d+?_*\d+?\.html')] = u'知识内容页'
    return j['type'].value_counts()
```

In [ ]:

```
# 连接数据库
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('all_gzdata', engine, chunksize=10000)

counts2 = [count107(i) for i in sql] #逐块统计
counts2 = pd.concat(counts2).groupby(level=0).sum() #合并统计结果
print counts2
```

3. 统计分析结果 -- 知识类型页面107001

前面统计统计分析知识类型页面107001共84202个页面，按照上面定义的count107函数统计结果共有81904个页面， ==> 84202-81904 =2298个页面不在上面这三种正则表达式的包含情况内。

这三种知识类型页面的分布情况如下表：

107网页类型	记录数	百分比
知识内容页	73779	90.08
知识列表页	4157	5.08
知识首页	3968	4.84

4. 统计分析 -- 其他页面199

观察：

分析其他页面（199类型）的情况，发现：

- 网址中带有‘？’的占约 32%
- 其他咨询相关与法规专题占比约 43%
- 地区和律师的占比约 26%

在前面网页的分类中，有律师（102）、地区、咨询相关（101）的网页分类，为什么这些还会存在其他类别中？进行数据查看后，发现大部分是以下面网址的形式存在：

- http://www.XXX.com/guangzhou/p2lawfirm (http://www.XXX.com/guangzhou/p2lawfirm) 地区律师事务所
- http://www.XXX.com/guangzhou/ (http://www.XXX.com/guangzhou/) 地区网址
- http://www.XXX.com/ask/ask.php (http://www.XXX.com/ask/ask.php)
- http://www.XXX.com/ask/midques_10549897.html (http://www.XXX.com/ask/midques_10549897.html) 中间类型网页
- http://www.XXX.com/ask/exp/4317.html (http://www.XXX.com/ask/exp/4317.html) 咨询经验
- http://www.XXX.com/ask/online/138.html (http://www.XXX.com/ask/online/138.html) 在线咨询页

带有标记的3类网址本应该有相应的分类，但是由于人工设定分类规则的匹配问题，没有相应的匹配。

- 带有lawfirm 关键字对应的应该是律师事务所
- 带有ask/exp、 ask/online 关键字对应的应该是咨询经验和在线咨询页 所以，在处理数据的过程中将其进行清楚分类，便于后续数据分析。

综上分析，发现： 大部分用户浏览的网页的情况为：咨询内容页、知识内容页、法规专题页、咨询经验（在线咨询页）。因此，在后续的分析中，选择其中占比最多的两类（咨询内容页和知识内容页）进行模型分析。

In [ ]:

```
import pandas as pd
from sqlalchemy import create_engine
engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('all_gzdata', engine, chunksize=10000)

# 统计点击次数
c = [i['realIP'].value_counts() for i in sql] #逐块统计各个IP的出现次数
counts3 = pd.concat(c).groupby(level=0).sum()#合并统计结果，level=0表示按index分组，然后求和
counts3 = pd.DataFrame(counts3) #将Series转为DataFrame
print len(counts3)  # 实际的用户数量
print counts3['realIP'].sum() # 实际的点击次数总和，即数据库的记录条数
counts3[1] = 1 #添加一列，值全为1
counts3.groupby('realIP').sum() # 统计各个‘不同点击次数’分别出现的频次

c3_ = counts3.groupby('realIP').sum() / 123104 * 100  # 统计不同点击次数的用户在全部用户中所占的比例
c3_.head()
```

5. 点击次数的统计分析

点击次数	用户数	用户百分比	记录百分比
1	75096	61.00	19.43
2	22952	18.64	11.88
3	8559	6.95	6.63
4	4905	3.98	5.08
5	2684	2.18	3.45
5次以上	8908	7.24	53.53

观察发现：

- 大约80%的用户点击不超过3次，占总的浏览量的31.3%（几乎满足二八定律）

也可以对数据进行分析，观察点击浏览次数最多的页面的统计分析，以及对应的用户浏览情况等。

也可以针对浏览次数为一次的用户进行分析了解，其中问题咨询页面占比情况（77%），知识页面占比情况（15%），这些记录是否是通过搜索引擎进入的等等。由此，==>可以猜想两种可能性：

- 用户为流失用户，在问题咨询与知识页面上没有找到相关的需要；
- 用户找到其需要的信息，因此直接退出。 综合这些情况，可以将这些点击一次的用户行为定义为网页的跳出率，为了降低网页的跳出率，需要对这些网页进行针对用户的个性化推荐，帮助用户发现其感兴趣或者需要的网页。

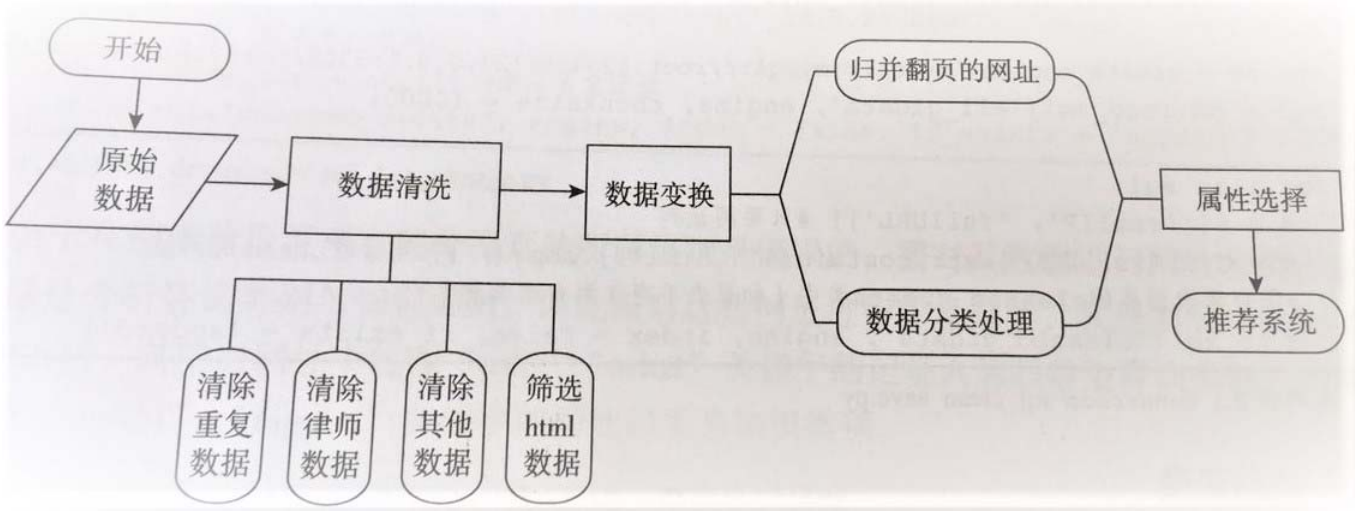
进一步分析，这些点击一次的用户浏览的网页统计情况，发现被点击次数较多（即排名靠前）的都是知识与咨询页面，因此可以猜想大量用户的关注都在知识或咨询方面。

3. 数据预处理

在对原始数据的探索分析的基础上，发现与分析目标无关或模型需要处理的数据，针对此类数据进行处理。

涉及到的数据预处理方式有： 数据清洗、数据集成和数据变换。 通过这几类的处理方式，将原始数据处理成模型需要的输入数据。

数据预处理的流程如图：



1 ） 数据清洗

从探索分析的过程中发现与分析目标无关的数据，归纳总结其数据满足这些规则：中间页面的网址、咨询发布成功页面、律师登录助手页面等。将这些整理为删除数据的规则，清洗的结果如下，可以发现，原始数据记录386,476条，其中律师用户信息占了所有记录中的22%左右，其他类型的数据，占比很小（自己统计），大概5%左右。

删除数据规则	删除数据记录	百分比
中间类型网页（带midques_ 关键字）		
（快车-律师助手）律师的浏览信息		
咨询发布成功		
主网址不包含关键字		
快搜与免费发布咨询的记录		
其他类别带有？的记录		
无 .html点击行为的用户记录		
重复记录		

经过上述数据清洗后的记录中仍然存在大量的目录网页，在进入推荐系统时，这些信息的作用不大，反而会影响推荐的结果，因此需要进一步筛选以html为后缀的网页。

根据分析目标以及探索结果可知，咨询和知识是网站主要业务来源，故需筛选咨询与知识相关的记录，将此部分数据作为模型分析需要的数据。

针对数据进行清洗的操作，清理后的数据为 310,323条。 具体Python实现的代码如下：

In [ ]:

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('all_gzdata', engine, chunksize = 10000)

for i in sql:
    d = i[['realIP', 'fullURL']] #只要网址列
    d = d[d['fullURL'].str.contains('\.html')].copy() #只要含有.html的网址
    #保存到数据库的cleaned_gzdata表中（如果表不存在则自动创建）
    d.to_sql('cleaned_gzdata', engine, index = False, if_exists = 'append')
```

2）数据变换

（1）删去翻页网址

在用户访问知识的过程中，存在翻页的情况，不同的网址属于同一类型的网页。在数据处理的过程中需要对这类网址进行处理，最简单的处理方法是直接删掉翻页的网址。但是，用户的访问页面是通过搜索引擎导入网站的，所以其入口网页不一定是其原始类别的首页，采用删除的方法会损失大量的有用数据，在进入推荐系统时，会影响推荐结果。因此，针对这些网页需要还原其原始类别，处理方式为首先识别翻页的网址，然后对翻页的网址进行还原，最后针对每个用户访问的页面进行去重操作。

用户翻页的数据处理代码如下：

In [ ]:

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('cleaned_gzdata', engine, chunksize = 10000) # 清洗后的数据集

for i in sql: #逐块变换并去重
    d = i.copy()
    d['fullURL'] = d['fullURL'].str.replace('_\d{0,2}.html', '.html') #将下划线后面部分去掉，规范为标准网址
    d = d.drop_duplicates() #删除重复记录
    d.to_sql('changed_gzdata', engine, index = False, if_exists = 'append') #保存到数据库的changed_gzdata表中（如果表不存在则自动创建）
```

（2）人工规则分类

由于在探索阶段发现有部分网页的所属类别是错误的，需对其数据进行网址分类，且分析目标是分析咨询类别与知识类别，因此需对这些网址进行手动分类，其分类的规则和结果：

- 对网址中包含‘ask’、‘askzt’关键字的记录人为归类至咨询类别
- 对网址中包含 ‘zhishi’、‘faguizt’关键字的网址归类为知识类型

因为目标是需要为用户提供个性化的推荐，在处理数据的过程中需要进一步对数据进行分类，知识部分是由很多小的类别组成。由于所提供的原始数据中知识类型无法进行内部分类，从业务上进行分析，可以采用其网址的构成对其进行分类。分类的结果见下表：

用户	类别1	类别2	类别3
863142519	zhishi	minshi	fagui
863142519	zhishi	shuifa	yys
863142519	zhishi	jiaotong	jtnews

网址分类的数据处理代码如下：

In [ ]:

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/law?charset=utf8')
sql = pd.read_sql('cleaned_gzdata', engine, chunksize = 10000) # 清洗后的数据集

for i in sql: #逐块变换并去重
    d = i.copy()
    d['type_1'] = d['fullURL'] #复制一列
    d['type_1'][d['fullURL'].str.contains('(ask)|(askzt)')] = 'zixun' #将含有ask、askzt关键字的网址的类别一归为咨询（后面的规则就不详细列出来了，实际问题自己添加即可）
    d.to_sql('splited_gzdata', engine, index = False, if_exists = 'append') #保存
```

统计分析每一类中的记录，以知识类别中的婚姻法为例进行统计分析。可见其网页的点击率基本满足二八定律，即80%的网页占了浏览量的20%左右，通过这个规则，按点击行为进行分类，20%的网页是热点网页，其他80%的页面属于点击次数少的，因此，在进行推荐过程中，需要将其分开进行推荐，才能达到最优的推荐效果。

3）属性规约

由于推荐系统模型的输入数据需要，需对处理后的数据进行属性规约，提取模型需要的属性。本实例中模型需要的数据属性为用户和用户访问的网页。因此删除其他的属性，只选择用户与用户访问的网页。

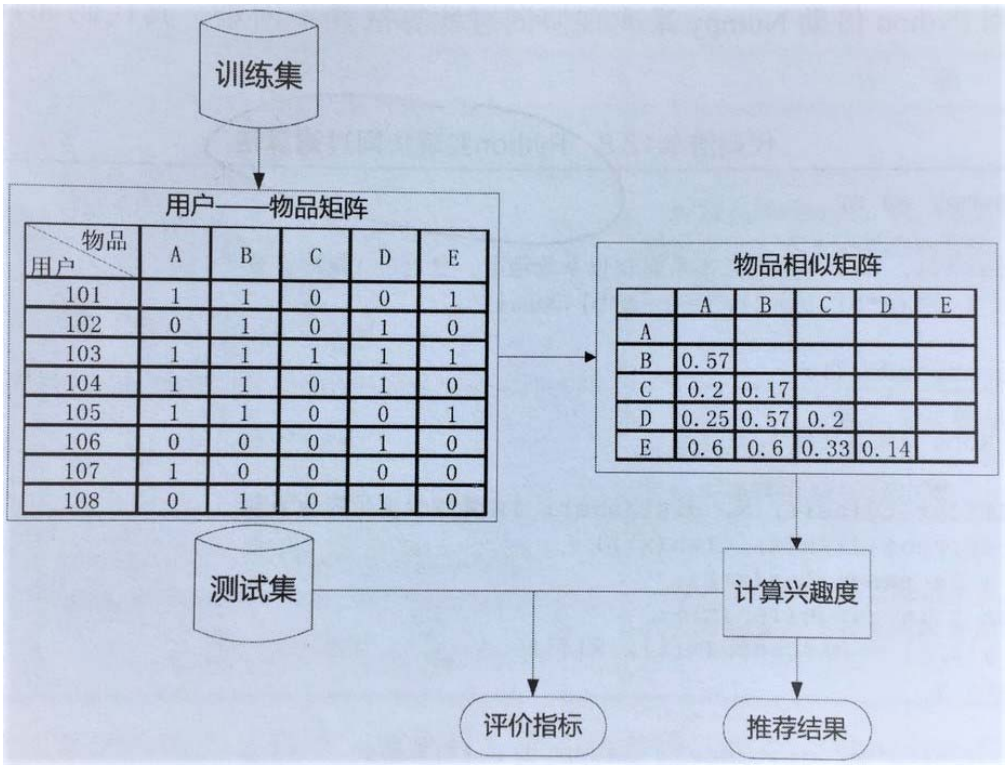
4. 推荐系统

4.1 基于物品的协同过滤算法

基于物品的协同过滤算法主要分为两步：

1. 计算物品之间的相似度
2. 根据物品的相似度和用户的历史行为给用户生成推荐列表

基于物品的协同过滤建模流程图：



用户＼物品	A	B	C	D	E
101	1	1	0	0	1
102	0	1	0	1	0
103	1	1	1	1	1
104	1	1	0	1	0
105	1	1	0	0	1
106	0	0	0	1	0
107	1	0	0	0	0
108	0	1	0	1	0

	A	B	C	D	E
A					
B	0.57				
C	0.2	0.17			
D	0.25	0.57	0.2		
E	0.6	0.6	0.33	0.14	

在实际数据中，物品数目过多，建立的用户-物品（user-item）矩阵 与 物品相似度（item-item）矩阵 都是非常庞大的矩阵，因此，在用户-物品矩阵的基础上采用Jaccard 相似系数的方法，计算出物品相似度矩阵。然后，通过物品相似矩阵 与 测试集 的用户行为，计算用户的兴趣度，获得推荐结果，进而计算出各种评价指标。

4.2 如果计算物品的相似度？

将用户对某一个物品的喜好或者评分作为一个向量，例如 用户 n (N个用户) 对物品 A1 (M个物品) 的评分或者喜好程度表示为： $A_1 = (x_{11}, x_{21}, x_{31}, \ldots, x_{n1})$, 用户的行为是二元选择，然后采用下面的这些相似度计算办法得到各个物品之间的相似度：

1. 夹角余弦 cosine
2. Jaccard 相似系数
3. 相关系数(Correlation coefficient)

计算后，可以构成一个物品之间的相似度矩阵。

4.3 如何分析用户的历史行为？

用户行为存在很多种，例如 浏览网页与否、是否购买、评论、评分、点赞等 行为，针对具体的分析目标进行具体的表示。本实例中，原始数据只记录了用户访问网站的浏览行为，因此用户的行为是浏览网页是否、并没有进行类似电商网站上的购买、评分和评论等用户行为。

4.4 如何生成推荐列表？

通过采用相似度矩阵，推荐算法会给用户推荐与其物品最相似的 K个物品。采用公式： $P = R * SIM$， 度量了推荐算法中用户对所有物品的感兴趣程度，其中，R代表用户对物品的兴趣，SIM 代表所有物品之间的相似度，P 为用户对物品感兴趣的程度。因为用户的行为是二元选择（是/否），所以，在用户对物品的兴趣矩阵R中，只存在0和1的值。

Python实现协同过滤算法的代码如下;

```
# 推荐算法
class Recommender():
    sim = None #相似度矩阵

    def similarity(self, x, distance): #计算相似度矩阵的函数
        y = np.ones((len(x), len(x)))
        for i in range(len(x)):
            for j in range(len(x)):
                y[i,j] = distance(x[i], x[j])
        return y

    def fit(self, x, distance = Jaccard): #训练函数
        self.sim = self.similarity(x, distance)

    def recommend(self, a): #推荐函数
        return np.dot(self.sim, a)*(1-a)
```

In [ ]:

```
# 定义相似度计算函数
import numpy as np
from numpy import linalg as la

def ecludSim(a, b): # a,b都是列向量
    return 1.0/(1.0 + la.norm(a - b))

def pearsSim(a, b):# a,b都是列向量
    if len(a) < 3: return 1.0
    return 0.5+0.5*corrcoef(a, b, rowvar = 0)[0][1]

def consSim(a,b):# a,b都是列向量
    num = float(a.T * b)
    denom = la.norm(a) * la.norm(b)
    return 0.5+0.5*(num/denom)

def Jaccard(a, b): #自定义杰卡德相似系数函数，仅对0-1矩阵有效
    return 1.0*(a*b).sum()/(a+b-a*b).sum()
```

推荐算法 class Recommender(): sim = None #相似度矩阵 def similarity(self, x, distance): #计算相似度矩阵的函数 y = np.ones((len(x), len(x))) for i in range(len(x)): for j in range(len(x)): y[i,j] = distance(x[i], x[j]) return y def fit(self, x, distance = Jaccard): #训练函数 self.sim = self.similarity(x, distance) def recommend(self, a): #推荐函数 return np.dot(self.sim, a)*(1-a)

4.5 比较个性化推荐算法的性能好坏

选择两个非个性化的基准系统（baseline）：

1. Random算法： 每次随机挑选用户没有产生过行为的物品，并推荐给用户
2. Popular算法： 按照物品的流行度，为用户推荐他没有产生过行为的物品中最热门的物品。

4.6 评估推荐结果并应用推荐系统的预测结果

1. 将数据分隔为训练数据集和测试数据集，在训练数据集上构建推荐系统，并在测试数据集上进行推荐，并对推荐结果进行准确性评估
2. 将数据分隔为多份，采用交叉验证的方法进行实验，并对实验结果进行比较
3. 在实际进行推荐的时候，可以综合多个不同推荐系统的结果（即Ensembel），来综合多个不同推荐系统的优点进行推荐

5. 推荐系统中的几个重要问题

1. 推荐结果出现为NULL

产生的原因：由于在目前的数据集中，出现访问此网址的只有单独一个用户，因此在协同过滤算法中计算它与其他物品的相似度为0，出现无法推荐的情况。

解决方法：在实际应用中，可以考虑其他的非个性化的方法进行推荐，例如，基于关键字、基于相似行为的用户等。

2. 推荐结果类似

通常，最热门物品往往具有较高的“相似性”。例如，热门的网址，访问各类网页的大部分人都会进行访问。

在计算物品相似度的过程中，可以知道各类网页都和某些热门的网址有关，因此处理热门网址的方法有： 1）在计算相似度的过程中，加强对热门网址的惩罚，降低其权重，比如对相似度平均化或者对数化等； 2）将推荐结果中的热门网址过滤掉，推荐其他的网址，而将热门网址以热门排行榜的形式进行推荐。

如何确定热门网址或热门物品？在最近某段时间内的行为次数（例如，访问、浏览、查询、点击、购买等）

因此，实际应用过程中要结合业务进行分析，对模型进一步改造。

3. 物品相似度的计算

在协同过滤推荐中，物品的相似是因为它们共同出现在很多用户的兴趣列表中，也就是说每个用户的兴趣列表都对物品的相似度产生贡献。但是，并不是每个用户的贡献度都相同。思考：不活跃的用户可以是新用户，也可以是只来过网站一两次的老用户。一般，可以认为新用户倾向于浏览热门物品（对网站不熟悉，只点击首页的热门物品，老用户会逐渐浏览冷门的物品）。

活跃用户对物品相似度的贡献应该小于不活跃的用户。所以，可以改进相似度的计算，取用户的活跃度对数的倒数作为分子。

Jaccard计算物品相似度的公式为： $J(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$ ，其中分母表示喜欢物品1和喜欢物品2的用户总数，分子表示同时喜欢物品1和物品2的用户总数

改进的物品相似度的计算公式为： $J(A_1, A_2) = \frac{\sum N \in |A_1 \cap A_2| \frac{1}{log(1+A(N))}}{|A_1 \cup A_2|}$

此外，原始数据中，每个物品（本实例是网页）都有很多属性描述的文本内容（以及图片等），可以采用文本挖掘的分析方法（或图像处理的技术），找出每个网页文本或者物品描述文本中的隐含语义，将用户与物品联系在一起，相关的技术有LSI，pLSA，LDA或Topic Model。当然，也可以通过自然语言处理和文本挖掘的技术，提取出物品文本表达的语义，求出那些无法得到推荐结果的物品，这样获得物品的相似度估计，可以再与基于用户兴趣的相似度结合，得到更准确的相似度计算。

在实际应用中，为了提高推荐的准确率，还会将基于物品的相似度矩阵按最大值归一化，其好处不仅仅在于增加推荐的准确度，还可以提高推荐的覆盖率和多样性（不同类别的推荐）。

4. 相关推荐

除了个性化推荐列表，另一个重要的推荐应用就是相关推荐列表。在网购中经常出现的情况是，当用户在电商平台上购买了一个商品时，平台会在商品信息下面展示相关的商品。第一种是包含了购买这个商品的用户也经常购买的其他商品，第二种是包含浏览过这个商品的用户经常购买的其他商品。

这两种相关推荐列表的区别是：使用了不同用户行为来计算物品的相似性。

5. 冷启动问题

在没有大量用户数据的情况下，如何设计推荐系统？冷启动问题有下面三类：

- 1）用户冷启动：当新的用户产生，如何对其做个性化推荐？
- 2）物品冷启动：如何将新物品推荐给可能对其感兴趣的用户。在新闻网站等时效性很强的网站中非常重要。
- 3）系统冷启动：如何在一个新开发的网站上设计个性化推荐（没有用户，只有一些物品信息），从而在网站刚发布时就让用户体验到个性化推荐服务。

解决途径：

1）利用用户注册信息。通常，用户注册信息含3种：

- 1）人口统计学信息，如年龄、性别、职业、学历等，这些特征对预测用户的兴趣有很重要的作用。如对于女性，则推荐女性都喜欢的商品推荐热门商品（该方法粒度较粗）
- 2）用户兴趣描述
- 3）从其它网站导入的用户站外行为数据，如链接豆瓣、新浪等

基本流程如下：获取用户注册信息 -->根据用户的注册信息对用户分类 -->给用户推荐他所属分类中用户喜欢的物品

实际应用中也可考虑组合特征，如将 年龄性别 作为一个特征。不过在使用组合时需注意用户不一定具有所有特征（这是因为用户不一定填写所有信息）。

核心问题是计算具有人口特征f的用户喜欢的物品i，即对于每种人口特征f，计算具有这种特征的用户对各个物品的喜好程度p(f,i)，p(f,i)可简单定义为物品i在具有f特征的用户中的热门程度。利用的用户人口统计学特征越多，越能准确预测用户兴趣。但这种方法可能会导致热门物品会在各种特征的用户中均有较高的权重，不太符合个性化推荐的要求。改进方法：$p(f,i)=\frac{U(f)\cap N(i)}{N(i)+a}$，将p(f,i)定义为喜欢物品i的用户中具有特征f的比例，其中，参数a值通常较大，用来解决数据稀疏问题。

2）选择合适的物品启动对用户兴趣的采集

通过让用户对物品进行评分来收集用户兴趣。即， 新用户第一次访问时，并不立即给用户展示推荐结果，而是给用户提供一些物品，让用户对其反馈，根据用户反馈提供个性化推荐。

需解决的首要问题是：如何选择物品让用户进行反馈。选择需评分的物品的原则是：较热门、有代表性和区分性、多样性。

如何设计一个选择启动物品集合的系统？-- 用决策树解决。根据用户对某一物品的喜好程度分类，分为3类：喜欢、不喜欢、不知道；再在每类用户中再找到最具区分度的物品。

3）挖掘物品的内容信息解决物品冷启动问题

对于新的物品，需解决第一个用户从哪儿发现新的物品的问题。解决的方法：利用物品的内容信息，将新物品先投放给曾喜欢过和它内容相似的其他物品的用户（这与基于物品的协同过滤推荐算法相同）。

基于物品的协同过滤推荐算法通常，在在线服务时会将之前计算好的物品相关度矩阵放在内存中，因此当新物品加入时，内存中的物品相关表中不会存在这个物品。解决的方法是频繁更新物品相似度表（耗时，如原来是一天一次，现在增加到半小时一次）。

4）发挥人工作用（专家标注）

没有用户行为数据，也没有充足的物品内容信息准确的计算物品相似度，故利用专家进行标注。

代表系统是Pandora和Jinni。在Pandora中，采用全人工的方式，每首歌可表示为一个400维的向量，然后用常见的向量相似度算法计算歌曲的相似度。Jinni采用半人工、半自动的方式，即专家（50个）和机器学习（影评）相结合的方式。