不到100行代码实现一个简单的推荐系统

Nov 05 2014

作者Kai Zhou’ s Blog 发布：2014-11-05 05:56 分类：Python 抢沙发

似乎咱的产品七，八年前就想做个推荐系统的，就是类似根据用户的喜好，自动的找到用户喜欢的电影或者节目，给用户做推荐。可是这么多年过去了，不知道是领导忘记了还是怎么了，连个影子还没见到。

而市场上各种产品的都有了推荐系统了。比如常见的各种购物网站京东，亚马逊，淘宝之类的商品推荐，视频网站优酷的的类似影片推荐，豆瓣音乐的音乐推荐……

一个好的推荐系统推荐的精度必然很高，能够真的发现用户的潜在需求或喜好，提高购物网站的销量，让视频网站发现用户喜欢的收费电影… 可是要实现一个高精度的推荐系统不是那么容易的，netflix曾经悬赏高额奖金寻找能给其推荐系统的精确度提高10%的人，可见各个公司对推荐系统的重视和一个好的推荐系统确实能带来经济效益。

下面咱以电影电视的推荐系统为例，一步一步的来实现一个简单的推荐系统吧, 由于比较简单，整个推荐系统源码不到100行，大概70-80行吧，应该很容易掌握。 为了快速开发原型，咱采用Python代码来演示

1. 推荐系统的第一步，需要想办法收集信息

不同的业务，不同的推荐系统需要收集的信息不一样 针对咱要做的电影推荐，自然是每个用户对自己看过的电影的评价了，如下图所示:

Name	Friends	Bedtime Stories	Dawn of the Planet	RoboCop	Fargo	Cougar Town
Kai Zhou	4	3			1	2
Shuai Ge		3.5	3	4	2.5	4.5
Mei Nv	3	4	2	3	2	3
xiaoxianro	2.5	3.5	3	3.5	2.5	3
fengzhi	3	4		5	3.5	3
mincat	3	3.5	1.5	5	3.5	3
alex	2.5	3		3.5		4

RecommandationSystem_Estimate_Data

Kai Zhou对Friends打分是4分， 对Bedtime Stories打分是3分，没有对RoboCop打分 Shuai Ge没有对Friends打分，对Bedtime Stories打分是3.5分 …… 为简单，咱将此数据存成csv文件，形成一个二维的矩阵，假设存在D:\train.csv， 数据如下：

Source code

```
Name,Friends,Bedtime Stories,Dawn of the Planet of the Apes,RoboCop,Fargo,Cougar Town
Kai Zhou,4,3,5,,1,2
Shuai Ge,,3.5,3,4,2.5,4.5
Mei Nv,3,4,2,3,2,3
xiaoxianrou,2.5,3.5,3,3.5,2.5,3
fengzhi,3,4,,5,3.5,3
meinv,,4.5,,4,1,
mincat,3,3.5,1.5,5,3.5,3
alex,2.5,3,,3.5,,4
```

先从csv文件中加载二维矩阵，代码如下：

Source code

```
def load_matrix():
 matrix = {}
 f = open("d:\\train.csv")
 columns = f.readline().split(',')

 for line in f:
 scores = line.split(',')
 for i in range(len(scores))[1:]:
 matrix[(scores[0], columns[i])] = scores[i].strip("\n")

 return matrix

matrix = load_matrix()
print "matrix:", matrix
```

load_matrix()解析csv文件，返回一个dictionary, 该dictionary以（行名，列名）为索引

Recent Posts
[转]-石头剪刀布如何获胜
[转]-动态规划：从新手到专家
Newtonsoft.Json encounter Error reading JObject from JsonReader. Current JsonReader item is not an object: StartArray. Path ‘’, line 1, position 1
程序员修炼之道从小工到专家-读书笔记
Fix Could not load file or assembly or one of its dependencies. The system cannot find the file specified in Form designer
Tags
C# Issue Python Tool VC VisualStudio 概率 计算机相关
Recent Comments
最新日志 热评日志 随机日志
Error while trying to run project: Cannot
python pip install error:Unable to create
sqlite data provider does not exist in
Fix SQLite DateTime type only accepts
Fix The solution was offline during its
Fix Could not load file or assembly or one
程序员修炼之道从小工到专家-读书笔记
Newtonsoft.Json encounter Error reading
[转]-动态规划：从新手到专家
[转]-石头剪刀布如何获胜

数据有了，下面咱就正式开始干活了 ，推荐系统要干些什么呢？

咱以电影推荐来说，推荐系统需要解决的几个主要问题：

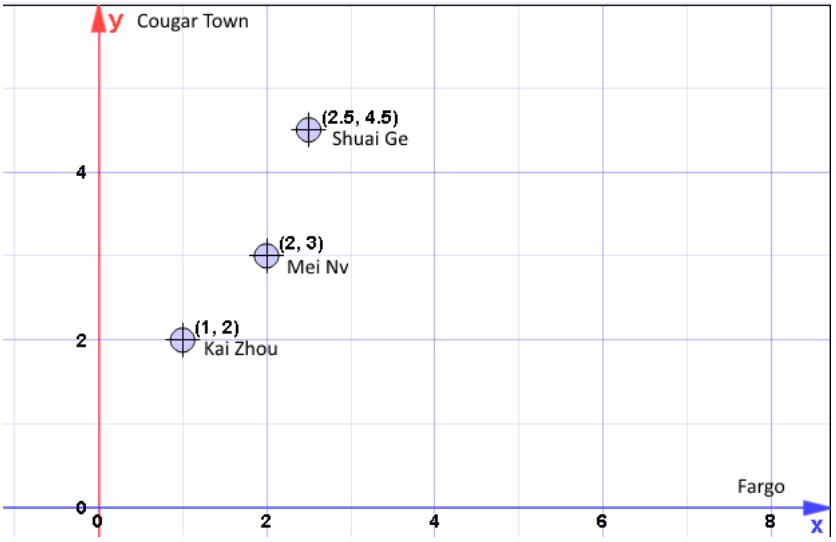
1. 判断两个电影，两个观影人之间的相似度

2. 找到和某影片最相似的影片， 或找到和某观影人有同样兴趣的人

3. 找到某观影人可能喜欢的电影，或找到对某影片感兴趣的人

2. 推荐系统的基础，判断相似度

针对咱的电影推荐来说，就是判断两个电影，两个观影人之间的相似度。 **2.1 欧几里德距离计算相似度** 最简单的，最容易理解的就是欧几里德距离. 那么，什么是欧几里德距离，怎么用呢？ 请对比评价数据，看下图：



EuclideanDistance

咱用两个电影Fargo和Cougr Town来取例 图中X轴代表电影Fargo，Y轴代表电影Cougr Town，Kai Zhou给电影Fargo 打1分，Cougr Town打2分，画到图上

同理，咱可以将Shuai Ge和Mei Nv的数据点都画到图上 很明显，咱可以看出Kai Zhou与Mei Nv 离得近，与Shuai Ge离得远，所以说Kai Zhou与Mei Nv的兴趣更相近. 用数学式子表达出来就是：

Kai Zhou与Mei Nv的距离的平方： (2 – 1)^2 + (3 – 2)^2 = 2

Kai Zhou 与Shuai Ge的距离的平方： (2.5 – 1)^2 + (4.5 – 2)^2 = 8.5

2 < 8.5， 所以Kai Zhou与Mei Nv比Shuai Ge兴趣更近. 这就是利用欧几里得距离来判断相似度 两个用户对所有电影的评价相似度的和，就是两用户的相似度

2.2 归一化处理

为了方便比较处理后的数据，一般还需要对计算出来的结果进行归一化处理。

数据标准化（归一化）处理是数据挖掘的一项基础工作，不同评价指标往往具有不同的量纲和量纲单位，这样的情况会影响到数据分析的结果，为了消除指标之间的量纲影响，需要进行数据标准化处理，以解决数据指标之间的可比性。

原始数据经过数据标准化处理后，各指标处于同一数量级，适合进行综合对比评价。

上面的介绍太学术化了吧，不容易懂，我的理解：归一化化就是要把你需要处理的数据经过处理后(通过某种算法)限制在你需要的一 定范围内。

简单的说，我们希望，处理后的数据取值范围在0-1之间. 在数学上有很多归一化处理的方法 常用的有

一、min-max标准化（Min-Max Normalization）

也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0 – 1]之间。

二、Z-score标准化方法

这种方法给予原始数据的均值（mean）和标准差（standard deviation）进行数据的标准化。经过处理的数据符合标准正态分布，即均值为0，标准差为1

咱可以根据需要选择，不过，针对咱这系统采用的是欧几里德距离，咱可以用下面的更简单的公式:

假设计算出来的欧几里德距离为：n

1 / (1 + n)

当距离为0，归一化后的值为：1

距离越大，归一化后的值越接近0

最新评论

博客统计

日志总数：60 篇
评论总数：1259 篇
标签数量：8 个
链接总数：0 个
建站日期：2007-04-22
运行天数：2907 天
最后更新：2015-4-3

有了上面的基础知识之后，下面的代码就水到渠成了

Source code

```
def sim_distance(matrix, row1, row2):
    columns = set(map(lambda l: l[1], matrix.keys()))
    si = filter(lambda l: matrix.has_key((row1, l)) and matrix[(row1, l)] != "" and mat
    if len(si) == 0: return 0
    sum_of_distance = sum([pow(float(matrix[(row1, column)]) - float(matrix[(row2, column
    return 1 / (1 + sqrt(sum_of_distance))

print sim_distance(matrix, "Kai Zhou", "Shuai Ge")
```

3. 找到和和某观影人有同样兴趣的人，某影片最相似的影片

a.有了上面的代码，找到和某用户有同样兴趣的人，就非常简单了。只要将某用户和其它所有用户的相似度计算出来，排下序就行了。

Source code

```
def top_matches(matrix, row, similarity=sim_distance):
    rows = set(map(lambda l: l[0], matrix.keys()))
    scores = [(similarity(matrix, row, r), r) for r in rows if r != row]
    scores.sort()
    scores.reverse()
    return scores

person = "Kai Zhou"
print "top match for:", person
print top_matches(matrix, person)
```

b. 找到和某影片相似的影片，这个需要稍微变化下。咱的输入数据是以用户为行数据，影片为列数据， 只要改成以影片为行数据，用户为列数据，一样的调用。 所以需要一个函数，将矩阵转置

Source code

```
def transform(matrix):
    rows = set(map(lambda l: l[0], matrix.keys()))
    columns = set(map(lambda l: l[1], matrix.keys()))

    transform_matrix = {}
    for row in rows:
        for column in columns:
            transform_matrix[(column, row)] = matrix[(row, column)]
    return transform_matrix
```

找到和Friends 相似的影片:

Source code

```
trans_matrix = transform(matrix)
print "trans:", trans_matrix

film = "Friends"
print "top match for:", film
print top_matches(trans_matrix, film)
```

4. 找到某观影人可能喜欢的电影，找到对某影片感兴趣的人

最理想的是找到两个相似度一样的人，可以认为某个人喜欢的电影，另外那个也喜欢。 但是这样有它的缺点，比较好的办法是把所有人的数据都用上，方法如下： 1. 先计算所有人和Kai Zhou的相似度 2. 对于Kai Zhou没有看过，没有评分，而其它人有评分的的影片， 将其评分与相似度相乘，得到的值再除以相似度之和 3. 排序 咱先以给Kai Zhou推荐影片为例来说明， Dawn of the Planet of the Apes 和 RoboCop 这两部影片Kai Zhou都没有看，我们该推荐他看哪部呢？ 假设我们计算出来Kai Zhou与其它人的相似度如下：

[(0.3333333333333333, ‘Mei Nv’),

(0.29429805508554946, ‘xiaoxianrou’),

(0.2857142857142857, ‘alex’),

(0.2553967929896867, ‘mincat’),

(0.252650308587072, ‘Shuai Ge’),

(0.2474401533514073, ‘fengzhi’)]

即Kai Zhou与Mei Nv 相似度为0.3333333333333333，与xiaoxiaorou相似度为0.29429805508554946，其它类似... 那么计算Dawn of the Planet of the Apes对Kai Zhou的推荐值过程如下：

1. 找到Shuai Ge对Dawn of the Planet of the Apes的评价值 乘以Shuai Ge与Kai Zhou的相似度： 3 * 0.252650308587072

2. 找到Mei Nv对Dawn of the Planet of the Apes的评价值 乘以其与Kai Zhou的相似度： 2 * 0.3333333333333333

3. 找到xiaoxianrou 对Dawn of the Planet of the Apes的评价值 乘以其与Kai Zhou的相似度： 3 * 0.29429805508554946

4. fengzhi 没有对Dawn of the Planet of the Apes评价，不用计算

5. 找到mincat对Dawn of the Planet of the Apes的评价值 乘以其与Kai Zhou的相似度： 1.5 * 0.2553967929896867

6. alex 没有对Dawn of the Planet of the Apes评价，不用计算

7. 将 1， 2， 3， 5 步的计算结果相加 得到： 3 * 0.252650308587072 + 2 * 0.3333333333333333 + 3 * 0.29429805508554946 + 1.5 * 0.2553967929896867

= 2.6906069471690612

8. 将1，2，3，5步的参与计算的人的相似度相加： 0.252650308587072 + 0.3333333333333333 + 0.29429805508554946 + 0.2553967929896867 = 1.1356784899956416

9. 将第7步结果除以第8步的结果，就是Dawn of the Planet对Kai Zhou的推荐值： 2.6906069471690612 / 1.1356784899956416 = 2.369162549851047

同样的方法，计算出来RoboCop 对Kai Zhou的推荐值为:3.9277923180363326 所以RoboCop应该对Kai Zhou的吸引力比Dawn of the Planet of the Apes更大. 代码如下：

Source code

```
def get_recommendations(matrix, row, similarity=sim_distance):
    rows = set(map(lambda l: l[0], matrix.keys()))
    columns = set(map(lambda l: l[1], matrix.keys()))

    sum_of_column_sim = {}
    sum_of_column = {}

    for r in rows:
        if r == row: continue
        sim = similarity(matrix, row, r)
        if sim <= 0:  continue

        for c in columns:
            if matrix[(r, c)] == "": continue

            sum_of_column_sim.setdefault(c, 0)
            sum_of_column_sim[c] += sim
            sum_of_column.setdefault(c, 0)
            sum_of_column[c] += float(matrix[(r, c)]) * sim

    scores = [(sum_of_column[c] / sum_of_column_sim[c], c) for c in sum_of_column]
    scores.sort()
    scores.reverse()
    return scores

print get_recommendations(matrix, person)
```

找到对某影片感兴趣的人和之前类似，需要将矩阵转置就行了,代码如下：

Source code

```
trans_matrix = transform(matrix)
print get_recommendations(trans_matrix,  "Friends")
```

这就是一个简单的推荐系统的雏型，当然，要实现一个可用的推荐系统，还有很多工作要做。比如推荐的精确度，用户喜欢打斗片，咱不可能给他推荐爱情片吧？比如数据量大了之后，性能问题，扩展性？是基于用户推荐还是物品推荐？.....

5. 参考资料

我为了了解学习推荐系统，找了一些资料，觉得下面的书值得阅读：

1.《**推荐系统实践**》项亮

入门级教材，很薄，可以很快就看完，把很多基础而简单的问题讲的很详细。总体来说，此书性价比很高，值得入手一本研读

我买书喜欢上亚马逊, 因为亚马逊上很多都可以试读，这本书亚马逊就提供了试读，推荐大家先去试读下，再决定有没有购买价值。

推荐系统实践
项亮
亚马逊价: ￥36.90

亚马逊购买

隐私声明

2.《**Recommender Systems Handbook**》Paul B. Kantor

有这本书就不用其它的了，很细很全，就是英文原版的有点小贵，真有志于做推荐系统的才去买吧，用到哪就翻书查。按人家的说法，所有敢自称handbook的书都是神书，没看过这本书出去吹牛逼时你都不好意思说自己是做推荐的。

本文固定链接: http://www.kai-zhou.com/recommandation-system-implement-in-100-line/ | Kai Zhou’ s Blog

该日志由 Kai Zhou’ s Blog于2014年11月05日发表在 Python 分类下， 你可以发表评论，并在保留原文地址及作者的情况下引用到你的网站或博客。
原创文章转载请注明: 不到100行代码实现一个简单的推荐系统 | Kai Zhou’ s Blog
关键字: Python, 概率, 计算机相关

【上一篇】Fix Visual Studio 2013 Not Responding frequently, CPU is high
【下一篇】Error while trying to run project: Cannot start debugging. The assembly to be debugged was built with a platform incompatible with the current system

您可能还会对这些文章感兴趣！

转-我是一段死代码
kindle当作电脑显示器使用
python threadpool安装
Sort dictionary by value, by keys... in python
Fix SQLite DateTime type only accepts Python
波斯公主选驸马定律
Fiddler 发送序列http请求
[转]-动态规划：从新手到专家
pip安装教程步骤
python pip install error:Unable to create process
Windows Server 2003下无官方显卡驱动，看我如
Fix Error: Can not enable

不到100行代码实现一个简单的推荐系统：等您坐沙发呢！

发表评论

昵称 *
邮箱 *（教你设置自己的个性头像）
网址

提交留言 重写
快捷键：Ctrl+Enter