知乎

如何用Python搭建一个简单的推荐系统?

第四范式...
已认证的官方帐号

关注他

20 人赞同了该文章

推荐系统的相关知识我们已在前文中提到，在这篇文章中，我们会介绍如何用Python来搭建一个简单的推荐系统。

本文使用的数据集是**MovieLens数据集**，该数据集由明尼苏达大学的Grouplens研究小组整理。它包含1,10和2亿个评级。 Movielens还有一个网站，我们可以注册，撰写评论并获得电影推荐。接下来我们就开始实战演练。

在这篇文章中，我们会使用Movielens构建一个基于item的简易的推荐系统。在开始前，第一件事就是导入pandas和numPy。

```
import pandas as pd
import numpy as np
```

知乎

```
df = pd.read_csv('u.data', sep='\t', names=['user_id','item_id','rating','titmestamp']
```

接下来查看表头，检查一下正在处理的数据。

```
df.head()
```

如果我们能够看到电影的标题而不仅仅是ID，那再好不过了。之后加载电影标题并把它与此数据集合并。

```
movie_titles = pd.read_csv('Movie_Titles')
movie_titles.head()
```

由于item_id列相同，我们可以在此列上合并这些数据集。

```
df = pd.merge(df, movie_titles, on='item_id')
df.head()
```

数据集中的每一列分部代表：

•user_id - 评级电影的用户的ID。

•item_id - 电影的ID。

•rating - 用户为电影提供的评级，介于1和5之间。

•timestamp - 电影评级的时间。

•title - 电影标题。

使用describe或info命令，就可以获得数据集的简要描述。如果想要真正了解正在使用的数据集的话，这一点非常重要。

```
df.describe()
```

知乎

数越高，电影越为相似。

以下例子将使用Pearson相关系数 （Pearson correlation coefficient），该数字介于-1和1之间，1表示正线性相关，-1表示负相关， 0表示没有线性相关。也就是说，具有零相关性的电影完全不相似。

我们会使用pandas groupby 功能来创建dataframe。按照标题对数据集进行分组，并计算其平均值获得每部电影的平均评分。

```
ratings = pd.DataFrame（df.groupby（'title'）['rating'].mean（））
ratings.head（）
```

接下来我们创建number_of_ratings列，这样就能看到每部电影的评分数量。完成这步操作后，就可以看到电影的平均评分与电影获得的评分数量之间的关系。五星级电影很有可能只被一个人评价，而这种五星电影在统计上是不正确的。

因此，在构建推荐系统时，我们需要设置阈值。我们可以使用pandas groupby功能来创建新列，然后按标题栏分组，使用计数函数计算每部电影的评分。之后，便可以使用head（）函数查看新的dataframe。

```
rating ['number_of_ratings'] = df.groupby（'title'）['rating'].count（）
ratings.head（）
```

接下来我们使用pandas绘制功能来绘制直方图，显示评级的分布：

```
import matplotlib.pyplot as plt
%matplotlib inline
ratings['rating'].hist(bins=50)
```

可以看到，大多数电影的评分都在2.5-4之间。通过类似的方法还可以将number_of_ratings列可视化。

```
ratings['number_of_ratings'].hist(bins=60)
```

从上面的直方图中可以清楚地看出，多数电影的评分都很低，评分最高的电影是一些非常有名的电影。

```
import seaborn as sns
sns.jointplot(x='rating', y='number_of_ratings', data=ratings)
```

从图中我们可以看出，电影平均评分与评分数量之间呈正相关关系，电影获得的评分数量越多，其平均评分越高。

创建基于item的简易推荐系统

接下来我们会快速创建一个基于item的简单的推荐系统。

首先，我们需要将数据集转换为矩阵，电影标题为列，user_id为索引，评级为值。完成这一步，我们将得到一个dataframe，其中列是电影标题，行是用户ID。每列代表所有用户对电影的所有评级。评级为NAN表示用户未对这部电影评分。

我们可以用该矩阵来计算单个电影的评级与矩阵中其余电影的相关性，该矩阵可以通过pandas pivot_table实现。

```
movie_matrix = df.pivot_table（index ='user_id', columns ='title', values ='rating'）
movie_matrix.head（）
```

接下来让我们找到评分数量最多的电影，并选择其中的两部电影。然后使用pandas sort_values并将升序设置为false，以便显示评分最多的电影。然后使用head（）函数来查看评分数目最多的前十部电影。

```
ratings.sort_values('number_of_ratings', ascending=False).head(10)
```

假设一个用户曾看过*Air Force One*（1997）和*Contact*（1997），我们想根据这两条观看记录向该用户推荐其他类似的电影，那么这一点可以通过计算这两部电影的评级与数据集中其他电影的评级之间的相关性来实现。第一步是创建一个dataframe，其中包含来自movie_matrix的这些电影的评级。

```
AFO_user_rating = movie_matrix['Air Force One (1997)']
contact_user_rating = movie_matrix['Contact (1997)']
```

知乎

使用pandas corwith功能计算两个dataframe之间的相关性。有了这一步，就能够获得每部电影的评级与Air Force One电影的评级之间的相关性。

```
similar_to_air_force_one = movie_matrix.corrwith（AFO_user_rating）
```

可以看到，*Air Force One*电影和*Till There Was You*（1997）之间的相关性是0.867。这表明这两部电影之间有很强的相似性。

```
similar_to_air_force_one.head（）
```

还可以计算*Contact*（1997）的评级与其他电影评级之间的相关性，步骤同上：

```
similar_to_contact = movie_matrix.corrwith（contact_user_rating）
```

可以从中发现，*Contact*（1997）和*Till There Was You*（1997）之间存在非常强的相关性（0.904）。

```
similar_to_contact.head（）
```

前边已经提到，并非所有用户都对所有电影进行了评分，因此，该矩阵中有很多缺失值。为了让结果看起来更有吸引力，删除这些空值并将相关结果转换为dataframe。

```
corr_contact = pd.DataFrame(similar_to_contact, columns=['Correlation'])
corr_contact.dropna(inplace=True)
corr_contact.head()corr_AFO = pd.DataFrame(similar_to_air_force_one, columns=['correla
corr_AFO.dropna(inplace=True)
corr_AFO.head()
```

上面这两个dataframe分别展示了与*Contact*（1997）和*Air Force One*（1997）电影最相似的电影。然而，问题出现了，有些电影的实际质量非常低，但可能因为一两位用户给他们5星评级而被推荐。

这个问题可以通过设置评级数量的阈值来解决。从早期的直方图中看到，评级数量从100开始急剧

知乎

```
corr_AFO [corr_AFO ['number_of_ratings']> 100] .sort_values（by ='correlation', ascend:
```

现在，我们就能得到与*Air Force One*（1997）最相似的电影，并把这些电影限制在至少有100条评论的电影中，然后可以按相关列对它们进行排序并查看前10个。

我们注意到*Air Force One*（1997）与自身相关性最高，这并不奇怪。下一部与*Air Force One*（1997）最相似的电影是*Hunt for Red October*，相关系数为0.554。

显然，通过更改评论数量的阈值，我们可以按之前的方式得到不同的结果。限制评级数量可以让我们获得更好的结果。

现在重复上边的步骤，可以看到与Contact（1997）电影最相关的电影：

```
corr_contact [corr_contact ['number_of_ratings']> 100] .sort_values（by ='Correlation'
```

与*Contact*（1997）最相似的电影是Philadelphia（1993），相关系数为0.446，有137个评级。所以，如果有人喜欢*Contact*（1997），我们可以向他们推荐上述电影。

以上是构建推荐系统的一种非常简单的方法，但并不符合行业标准。后续的话我们可以通过构建基于存储器的协同过滤系统来改进该系统。在这种情况下，将数据划分为训练集和测试集，使用诸如余弦相似性来计算电影之间的相似性；或者构建基于模型的协作过滤系统，然后使用Root Mean Squared Error（RMSE）等技术评估模型。

Github: mwitiderrick/simple-recommender-

文章来源：How to build a Simple Recommender System in Python

（以上内容由第四范式先荐整理发布）

相关阅读：