Getting Started with a Movie Recommendation System

电影推荐系统的入门

华邵钶 | Llonvne | 2021.11.12

Dataset From kaggle.com

https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system

Start of Movie Recommender System

开始学习电影推荐系统

1. Demographic Filtering

人口统计学过滤/热度推荐

这个过滤器根据电影的受欢迎程度和类型，向每个用户提供总体性的推荐。该系统向同一个数据集覆盖的用户推荐相同的电影。由于每个用户都是不同的，这个方法可能过于简单。这个系统背后的基本想法是，受到更多好评的电影将有更高的概率被普通观众所喜欢。

2 . Content Based Filtering

基于内容的过滤/相同类型推荐

他们根据特定的项目来推荐类似的项目。这个系统使用项目的原数据，或者可以说为关键词，如电影的类型、导演、描述、演员等，来进行这些推荐。这个推荐系统背后的原理是如果一个人喜欢某一个特定的事务，那么他就有可能喜欢类似的事务。

3. Collaborative Filtering

协作过滤 你相同兴趣人看的东西协作过滤—该系统将具有类似兴趣的人进行匹配，并根据这种匹配提供推荐。协作式过滤器不需要像其基于内容的同行那样的项目元数据。

1. Demographic Filtering

人口统计学过滤/热度推荐

1. Demographic Filtering

人口统计学过滤/热度推荐

- 我们需要一个衡量标准来给电影打分或评级
- 计算每部电影的分数
- 对分数进行排序，并向用户推荐最佳评分的电影。
-

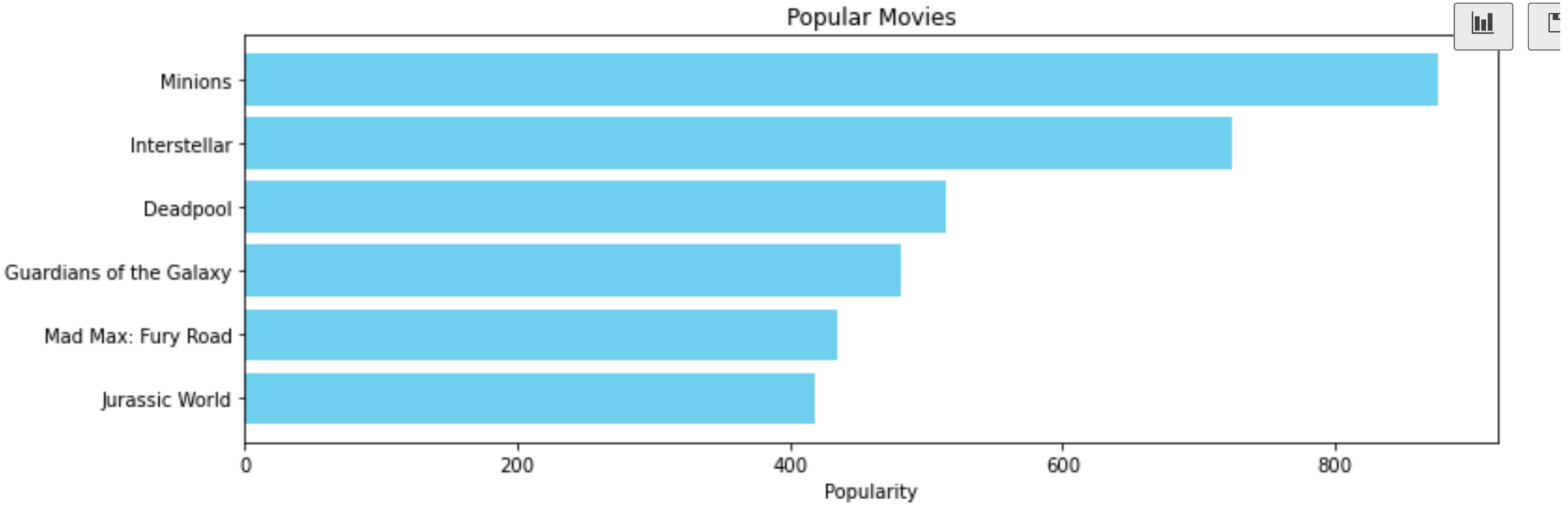
Evaluation function of Movie

电影评价函数

- $$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m}.R\right) + \left(\frac{m}{v+m}.C\right)$$
- 其中：
- v是电影的票数。
- m是在图表中列出的最低票数要求。
- R是电影的平均评分；以及
- C是整个报告的平均票数

Visualization for Result of Demographic Filtering

结果可视化



-

Summary for Demographic Filtering
人工统计学的总结

- 恭喜！我们做出了第一个（尽管是非常简单的）推荐器。在这些系统的Trending Now标签下，我们找到了非常受欢迎的电影，它们可以通过对数据集的人气列进行排序而获得。

- 现在需要记住的是，这些人口统计学推荐器向所有用户提供一个推荐电影的一般图表。它们对特定用户的兴趣和品味不敏感。这时，我们就会转向一个更精细的系统—内容过滤。

2. Content Based Filtering

基于内容的推荐

2. Content Based Filtering

基于内容的推荐

- 在这个推荐系统中，电影的内容（概述、演员、工作人员、关键词、标语等）被用来寻找它与其他电影的相似性。然后，最可能相似的电影被推荐。
- **Plot description based Recommender 基于情节描述的推荐程序 （基于一种分析）**
- Credits, Genres and Keywords Based Recommender 基于评分，类型和关键词的推荐（多种特征合并分析）
- TF-IDF 词频-逆文件频率
- Cosine Similarity Score 计算余弦相似度分数
- Recommended function 推荐函数

TF-IDF Algorithm

词频-逆文件频率

- 是一种用于资讯检索与资讯探勘的常用加权技术。TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

- 上述引用总结就是, 一个词语在一篇文章中出现次数越多, 同时在所有文档中出现次数越少, 越能够代表该文章.

- 但是, 需要注意, 一些通用的词语对于主题并没有太大的作用, 反倒是一些出现频率较少的词才能够表达文章的主题, 所以单纯使用是TF不合适的。权重的设计必须满足：一个词预测主题的能力越强，权重越大，反之，权重越小。所有统计的文章中，一些词只是在其中很少几篇文章中出现，那么这样的词对文章的主题的作用很大，这些词的权重应该设计的较大。IDF就是在完成这样的工作.

Cosine Similarity Score

余弦相似度

文本是由词组成的，我们一般通过计算词频来构造文本向量——词频向量。

所以，用余弦夹角来计算两个文本的距离的步骤就是：

首先，将两个文本数字化（TF-IDF），变成两个向量；

其次，计算两个向量的夹角余弦cos(θ)

这就是余弦相似度

Recommended function

推荐函数

1. 获取电影标题的索引。 | 使用刚刚创建的表格
2. 获取该特定电影与所有电影的余弦相似度分数列表。将其转换成一个图元列表，其中第一个元素是其位置，第二个元素是相似度分数。
3. 根据相似度分数对上述图元列表进行排序；
4. 得到这个列表中的前10个元素。忽略第一个元素，因为它指的是自己（与某部电影最相似的电影就是这部电影本身）。
5. 返回顶部元素的索引所对应的标题。

Result For Plot description based Recommender

基于电影描述推荐器的结果

```
get_recommendations('The Dark Knight Rises')
```

✓ 0.3s

```
65                              The Dark Knight
299                              Batman Forever
428                              Batman Returns
1359                                    Batman
3854    Batman: The Dark Knight Returns, Part 2
119                              Batman Begins
2507                                 Slow Burn
9            Batman v Superman: Dawn of Justice
1181                                      JFK
210                              Batman & Robin
Name: title, dtype: object
```

Summary about the Plot description based Recommender

情节推荐器的总结

- 虽然我们的系统在寻找具有类似情节描述的电影方面做得不错，但推荐的质量并不高。"黑暗骑士崛起 “返回所有的蝙蝠侠电影，而喜欢这部电影的人更有可能喜欢克里斯托弗-诺兰的其他电影。这是目前的系统所不能推断的。

Credits, Genres and Keywords Based Recommender

基于评分，类型和关键词的推荐

Credits, Genres and Keywords Based Recommender

基于评分，类型和关键词的推荐

- 如果使用更好的元数据，我们的推荐器的质量将得到提高。我们将根据以下元数据建立一个推荐器：3个主演、导演、相关类型和电影情节的关键词。
- 从演员、工作人员和关键词的特征中，我们需要提取三个最重要的演员、导演和与该电影相关的关键词。现在，我们的数据是以 "字符串化 “列表的形式出现的，我需要需要把它转换更加好用
- 从字符串提取导演主演等信息
- 数据清洗，转小写，去除空格
- 创建原数据汤
- 生成推荐函数2 （不用 TF-IDF）

Data Clean (Turn to lowercase and free of space)

数据清洗，转小写，去除空格

- 下一步将是把名字和关键词的实例转换为小写，并剥离它们之间的所有空格。这样做是为了让我们的向量器不把 "Johnny Depp "和 "Johnny Galecki "的Johnny算作是同一个人。

Generate recommendation functions

生成推荐函数

- 接下来的步骤与我们在基于情节描述的推荐器中所做的相同。一个重要的区别是，我们使用CountVectorizer而不是TF-IDF。这是因为，如果一个演员/导演在相对较多的电影中担任过演员或导演，我们不想降低其存在的权重。这并没有什么直观的意义。
  - 与ContentBased类似的使用CountVectorizer函数转化
  - 一样计算余弦相似度
  - 同样进行反向索引

Result of Credits, Genres and Keywords Based Recommender

基于评分，类型和关键词的推荐的结果

```
get_recommendations('The Dark Knight Rises', cosine_sim2)
```

✓ 0.3s

```
65                The Dark Knight
119                 Batman Begins
4638    Amidst the Devil's Wings
1196                 The Prestige
3073           Romeo Is Bleeding
3326               Black November
1503                       Takers
1986                       Faster
303                      Catwoman
747                Gangster Squad
Name: title, dtype: object
```

-

Summary for Result of Credits, Genres and Keywords Based Recommender

基于评分，类型和关键词的推荐的总结

- 我们看到，由于元数据（关键词）较多，我们的推荐器成功地获取到了更多的信息，并给了我们（可以说）更好的推荐。DC漫画的粉丝更有可能喜欢同一制作公司的电影。因此，在我们的上述特征中，我们可以增加production_company 。我们还可以增加导演的权重，通过在汤中添加更多的关键词。

Collaborative Filtering

协同过滤

Collaborative Filtering

协同过滤

- 我们基于内容的引擎受到一些严重的限制。它只能够推荐与某部电影接近的电影。也就是说，它没有能力捕捉品味并提供跨类型的推荐。
- 此外，我们建立的引擎并不是真正的个人化，因为它没有捕捉到用户的个人品味和偏见。任何查询我们的引擎以获得基于电影的推荐的人，都会收到关于该电影的相同推荐，无论她/他是谁。
- 1. 基于用户的过滤—这些系统向用户推荐类似用户喜欢的产品。为了测量两个用户之间的相似度，我们可以使用佩尔森相关或余弦相似。
- 2. 基于项目的协同过滤不是测量用户之间的相似性，而是根据它们与目标用户评价的项目的相似性来推荐项目。同样地，相似度可以用皮尔逊相关或余弦相似来计算。

Result of Code for Predict users' interest in any movie

预测用户的兴趣值的结果

- Prediction(uid=1, iid=302, r_ui=3, est=2.754950316852029, details={'was_impossible': False})
- 对于 ID 为 1 的用户 对于 ID 为 302 的电影预测值为 2.618
- 对于ID为302的电影，我们得到的估计预测值为2.618。这个推荐系统的一个惊人的特点是，它并不关心电影是什么（或它包含什么）。它纯粹是根据一个指定的电影ID来工作，并试图根据其他用户对电影的预测来预测评分。

Total Article Summary

总结

- 我们使用人口统计、基于内容和协作过滤来创建推荐器。虽然人口统计学过滤是非常重要的，不能实际使用，但混合系统可以利用基于内容和协作过滤的优势，因为这两种方法被证明是几乎互补的。这个模型是非常基线的，只提供了一个基本的框架来开始。

- 在本次学习Kaggle文章中，我学会了最基本的基于人口的推荐器，学会了简单的基于内容的推荐器和基于评分关键字的推荐器，再后来我又学习如何使用协同过滤的方法，在其中我又学会了如何导入，清洗数据，怎么处理数据，和关于大数据的各种算法