

推荐系统（4）-基于内容的推荐系统



Alan
数据分析、挖掘、机器学习

关注他

22 人赞同了该文章

一、基于内容（CB）的推荐系统

基于内容推荐的方法特别适用于文本领域，比如新闻的推荐等等。

核心：首先构造商品画像，之后根据此画像来寻找最相似的其他商品。



基本思想：给用户推荐与其曾经喜爱的物品相似的物品(基于物品自身的属性)

例子：

- **电影**

相同的演员, 导演, 电影风格,...

- **新闻**

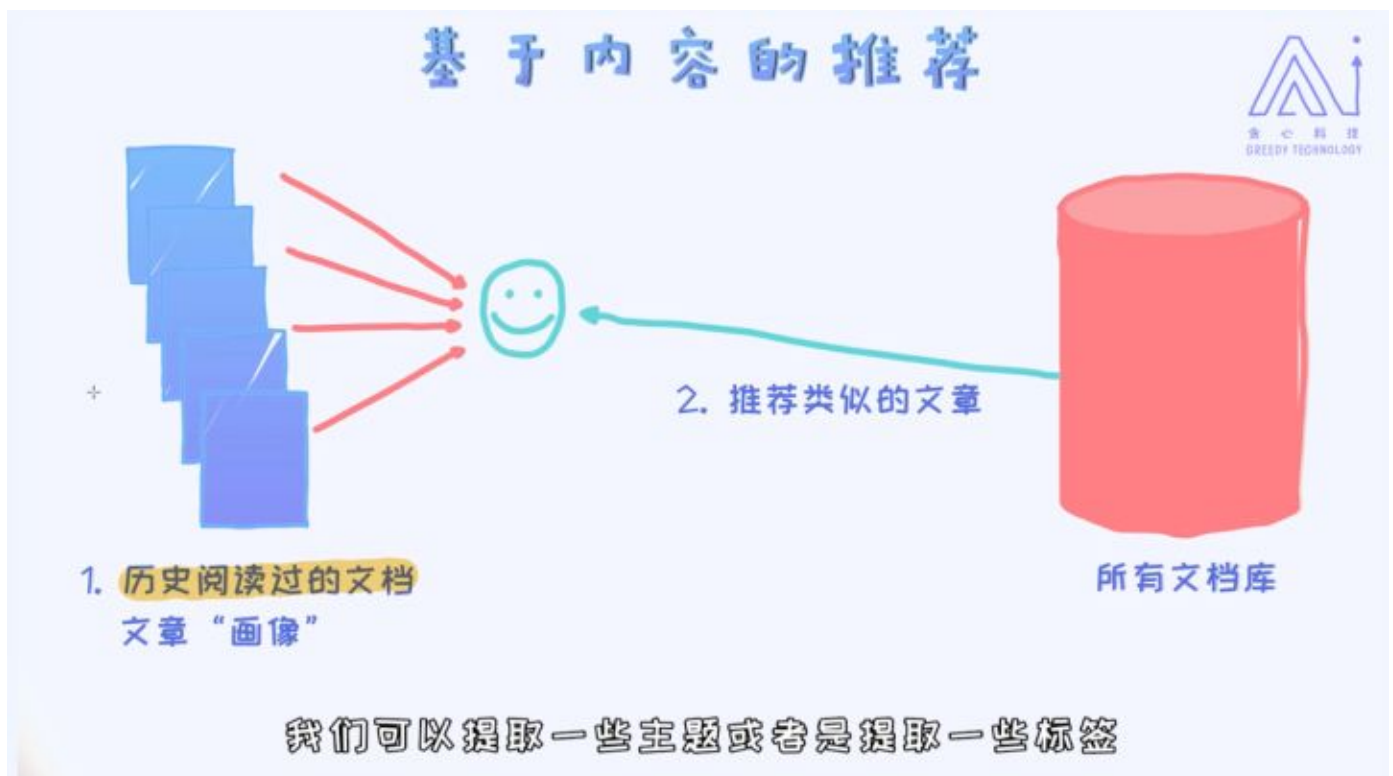
相似的主题,事件,人物,地点

- **人**

有共同的朋友圈

他们其实采用的算法也是基于内容的推荐

那具体如何来判断哪些是最相似的商品呢？ 答案是：计算相似度！



二、基于内容推荐系统的算法原理

2.1 相似度计算

那又如何计算相似度呢？



例子：电影构造物品画像

为每一个物品构建“物品画像”

- 电影：演员，导演，编剧，风格
- 图像：标签（作者，图像内容，图像介绍，背景叙述），元数据（拍照使用的曝光时间，光圈大小，使用的相机，拍照时间）
- 人：朋友圈，年龄，性别，籍贯，教育背景，工作单位，财务状况，婚姻状况，家庭成员
- 文本：重要的单词（tf-idf, 词频-逆向文件频率），包含的实体（机构, 人物, 地点, 事件），抽象的主题（例如文本是关于爱情，战争，或者是金融，科技）

画像可以通过向量来表达，所以把上述信息转换成向量形式

这个相似度其实就代表的是两个物品之间的相似度的

问题：如何把这些特征表示成向量？

离散型变量——通过独热编码的形式来转换成向量

数值型变量——直接使用等等

问题：那文本类的特征如何处理呢？比如电影的描述。

答：设计NLP领域。我们可以直接使用TF-IDF的方式即可以转换成向量的形式。当然我们也可以使用Word2Vec等技术来表示成向量的。

向量表示特征

例子：特征也叫作画像





计算相似度公式（常用余弦相似度）

$$\cos(\theta) = \frac{\sum_{i=1}^n (x_i \times y_i)}{\sqrt{\sum_{i=1}^n (x_i)^2} \times \sqrt{\sum_{i=1}^n (y_i)^2}}$$

$$= \frac{a \bullet b}{||a|| \times ||b||}$$

公式(4)

知乎 @Alan

问题：如何使用余弦相似度来计算每两个物品之间的相似度。



	武侠	动作	喜剧	文艺	悬疑	爱情
大话西游	1	0	1	0	0	1
夏洛特烦恼	0	0	1	0	0	1
黄飞鸿	0	1	0	0	0	0
笑傲江湖	1	1	0	0	0	0

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

$$\text{sim}(\text{大话}, \text{夏}) = \frac{2}{\sqrt{3} \cdot \sqrt{2}} = \frac{2}{\sqrt{6}}$$

$$\text{sim}(\text{黄}, \text{笑}) = \frac{1}{\sqrt{1} \cdot \sqrt{2}} = \frac{\sqrt{2}}{2}$$

所以这是他俩之间的余弦相似度

2.2 相似度排序推荐

推荐值的计算 (可用于排序)

	夏洛特 大话西游 烦恼	黄飞鸿	笑傲江湖	
大话西游	1.00	0.82	0.00	0.41
夏洛特烦恼	0.82	1.00	0.00	0.00
黄飞鸿	0.00	0.00	1.00	0.71
笑傲江湖	0.41	0.00	0.71	1.00

	大话西游	夏洛特烦恼	黄飞鸿	笑傲江湖
小明	?	2	5	4
小李	1	1	?	3
韩梅梅	5	5		2
小静	4		1	1

$$\bar{r} = \frac{2 \times 0.82 + 5 \times 0.00 + 4 \times 0.41}{0.82 + 0.00 + 0.41} = 0$$

$$\bar{r}' = \frac{1 \times 0 + 1 \times 0 + 3 \times 0.71}{0 + 0 + 0.71} = 0$$

所以这是基于内容的推荐算法

2.3 基于内容推荐算法的优缺点

优点：推荐较为准确



- 为某一用户做推荐的时候不需要使用其它用户的数据
- 可以为具有特殊口味的用户做预测
- 可解释性好, 产品的特征决定了推荐值

缺点: (主要冷启动问题)

基于内容推荐的缺点



某些物品的特征提取比较难

- 例如: 图像, 音乐, 电影, 如果提供这些物品的人没有提供元数据 (例如风格, 演员, 导演, 作者等等), 自动提取特征比较不容易

过于专门化

- 永远不会推荐和用户曾经喜欢的物品不相干的物品,
- 完全没有利用其他用户的喜好来提高对此用户的推荐质量

对于新用户有冷启动问题

- 刚出现的用户的用户画像为空, 无法做出推荐

总结起来我们三个比较重要的它的一个缺点

2.4如何去处理新用户的冷启动问题?

冷启动在推荐系统中非常常见。在基于内容的推荐算法中, 一旦一个新用户来了, 由于他还没有购买任何的物品, 所以无法给他推荐任何物品的。



- 让用户标注自己的兴趣
- 推荐最流行大的产品或者最近最火爆的产品

解决冷启动问题总结：

- 1、推荐目前热度最高的商品；
- 2、让用户自己标记一下自己喜欢的商品类型（APP新用户）

问题：基于内容的推荐还有一个大的问题，就是如何去维护物品之间的相似度？

答：计算单个物品与其他物品的相似度，排序存放相似度矩阵，使用时直接调度。

总结：基于内容的是目前常用，火热的推荐算法。

基于内容的推荐总结



- 适合文本类的推荐系统（比如新闻）
- 核心在于把商品描述以及内容更好地利用起来
- 根据场景，需要做调整，比如更重视导演等

三、代码实例



```
"""
    Author: Alan
    Desc:
        编写一个基于内容推荐算法的电影推荐系统（训练模型）
"""

import json
import pandas as pd
import numpy as np
import math
import random

class CBRecommend:
    # 加载dataProcessing.py中预处理的数据
    def __init__(self,K):
        # 给用户推荐的item个数
        self.K = K
        self.item_profile=json.load(open("data/item_profile.json","r"))
        self.user_profile=json.load(open("data/user_profile.json","r"))

    # 获取用户未进行评分的item列表
    def get_none_score_item(self,user):
        items=pd.read_csv("data/movies.csv")["MovieID"].values
        data = pd.read_csv("data/ratings.csv")
        have_score_items=data[data["UserID"]==user]["MovieID"].values
        none_score_items=set(items)-set(have_score_items)
        return none_score_items

    # 获取用户对item的喜好程度(余弦相似度)
    def cosUI(self,user,item):
        Uia=sum(
            np.array(self.user_profile[str(user)])
            *
            np.array(self.item_profile[str(item)])
        )
        Ua=math.sqrt( sum( [ math.pow(one,2) for one in self.user_profile[str(user)]])
        Ia=math.sqrt( sum( [ math.pow(one,2) for one in self.item_profile[str(item)]])
        return Uia / (Ua * Ia)

    # 为用户进行电影推荐
    def recommend(self,user):
        user_result={}
        item_list=self.get_none_score_item(user)
        for item in item_list:
```




```
        result = sorted(
            user_result.items(), key= lambda k:k[1], reverse=True
        )
    else:
        result = sorted(
            user_result.items(), key= lambda k:k[1], reverse=True
       )[:self.K]
    print(result)
```

推荐系统效果评估

```
def evaluate(self):
    evas=[]
    data = pd.read_csv("data/ratings.csv")
    # 随机选取20个用户进行效果评估
    for user in random.sample([one for one in range(1,6040)], 20):
        have_score_items=data[data["UserID"] == user]["MovieID"].values
        items=pd.read_csv("data/movies.csv")["MovieID"].values

        user_result={}
        for item in items:
            user_result[item]=self.cosUI(user,item)
        results = sorted(
            user_result.items(), key=lambda k: k[1], reverse=True
       )[:len(have_score_items)]
        rec_items=[]
        for one in results:
            rec_items.append(one[0])
        eva = len(set(rec_items) & set(have_score_items)) / len(have_score_items)
        evas.append( eva )
    return sum(evas) / len(evas)

if __name__=="__main__":
    cb=CBRecommend(K=10)
    cb.recommend(1)
    print(cb.evaluate())
```

发布于 2019-12-25

