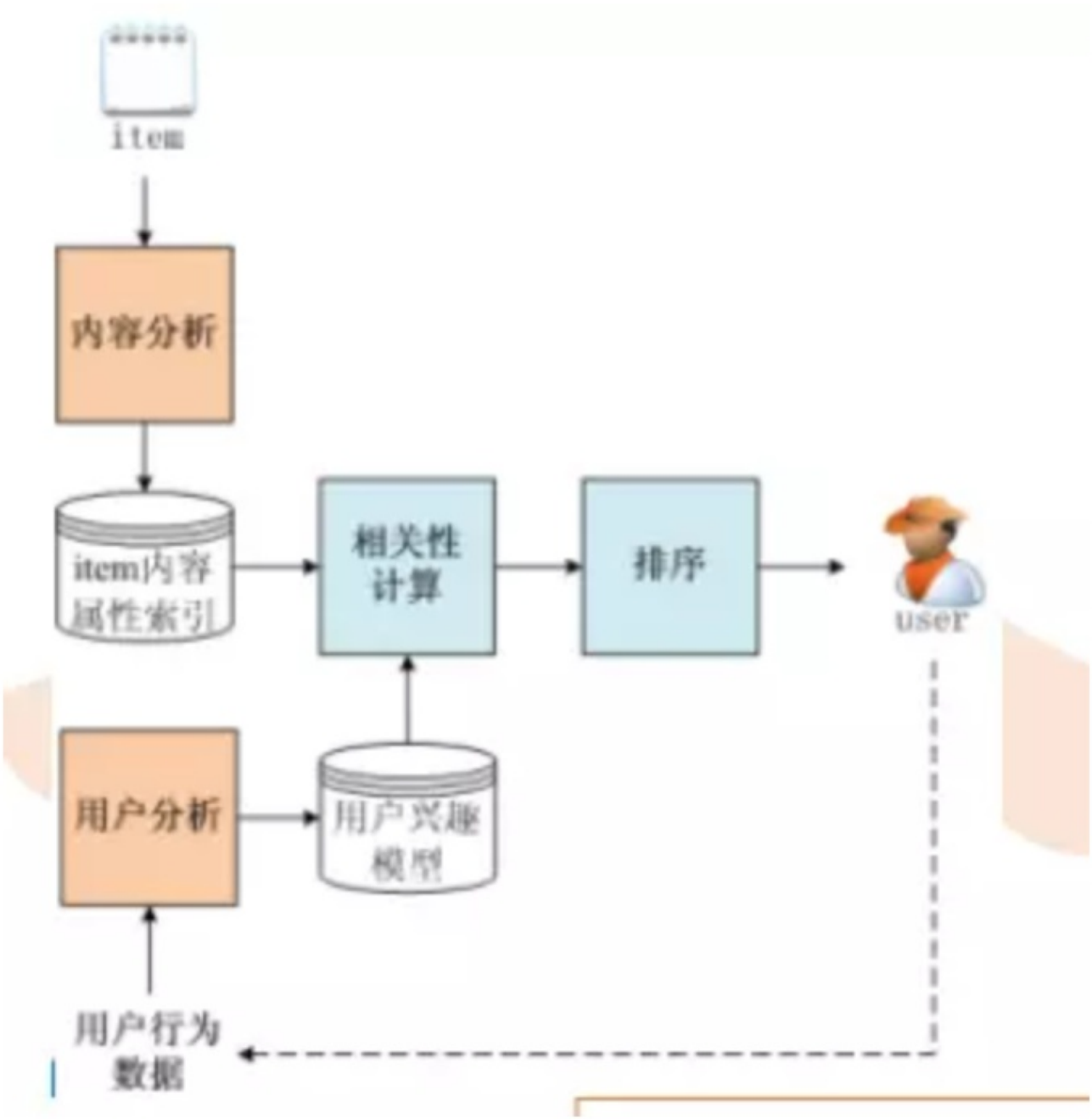


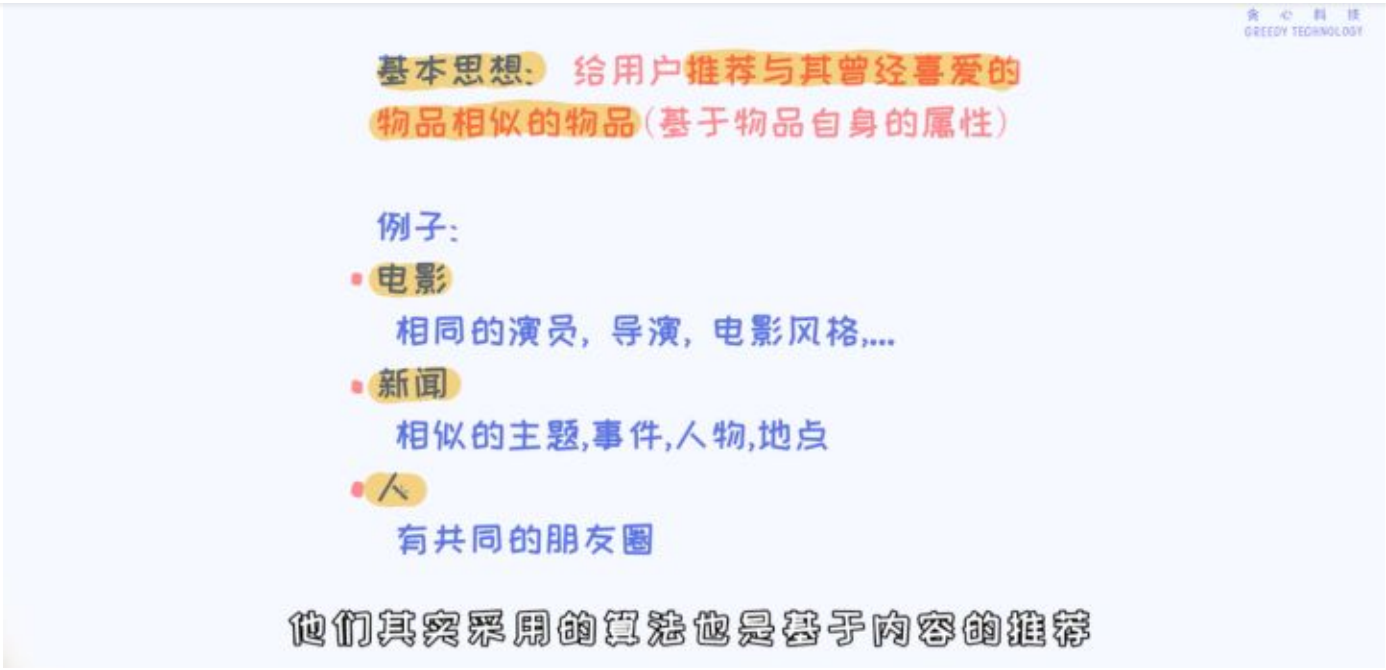
推荐系统（4）-基于内容的推荐系统

Alan
数据分析、挖掘、机器学习

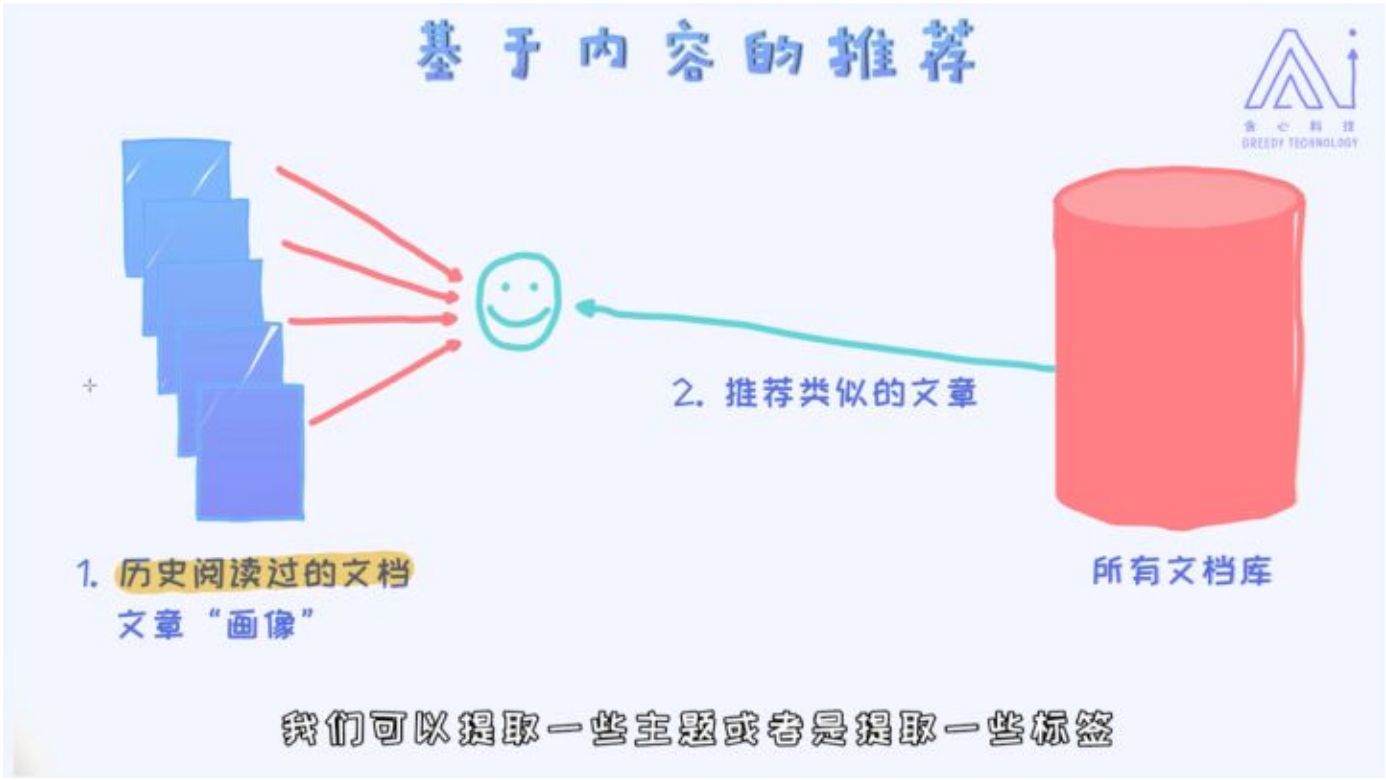
一、基于内容（CB）的推荐系统

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。



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！

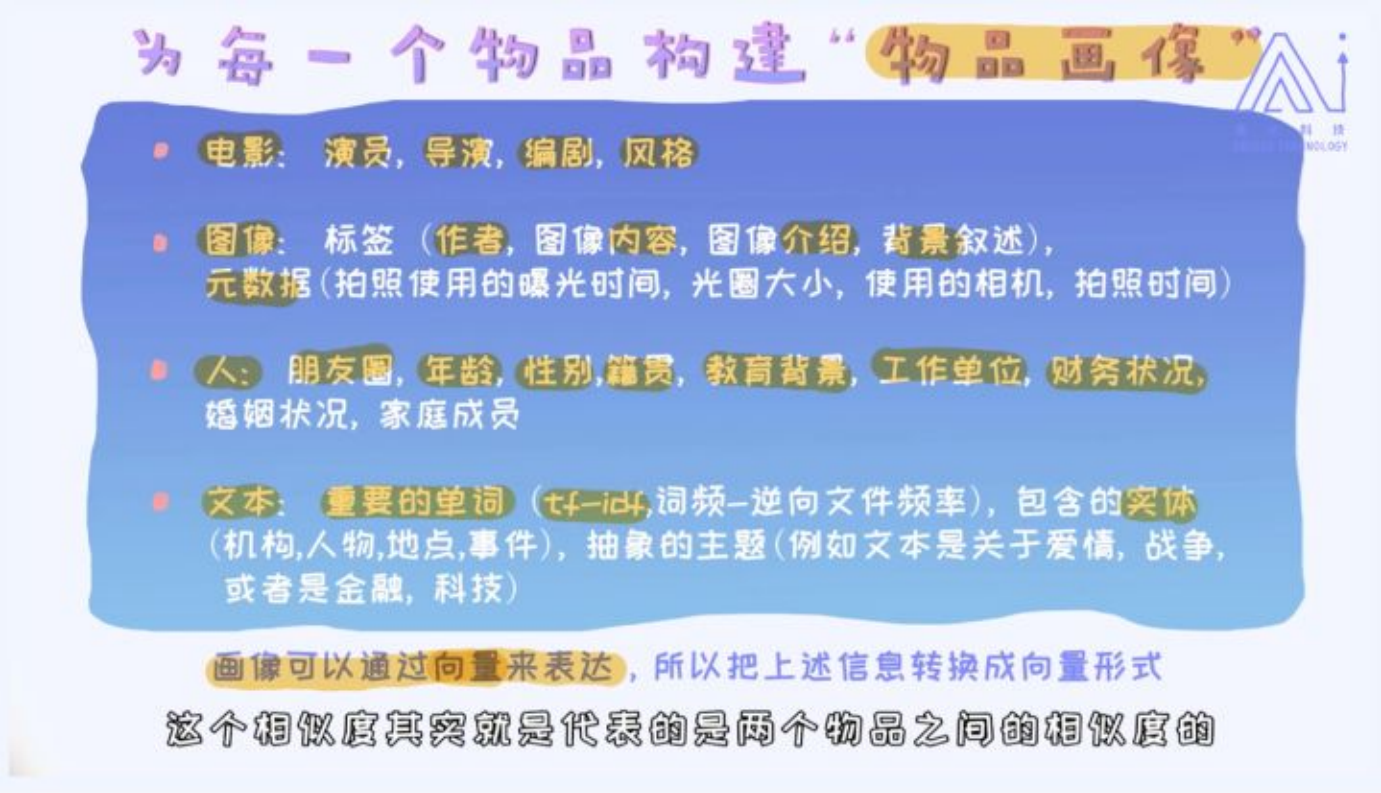


二、基于内容推荐系统的算法原理

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



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**）

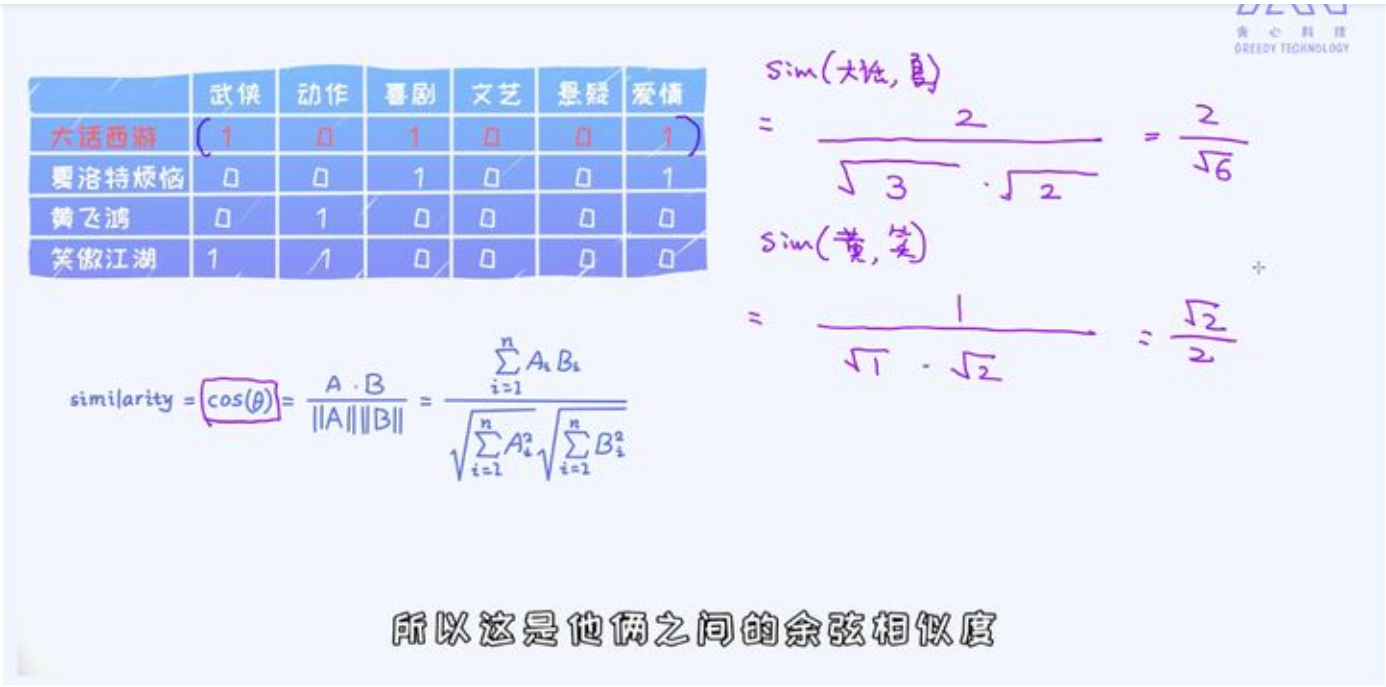
最小化

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

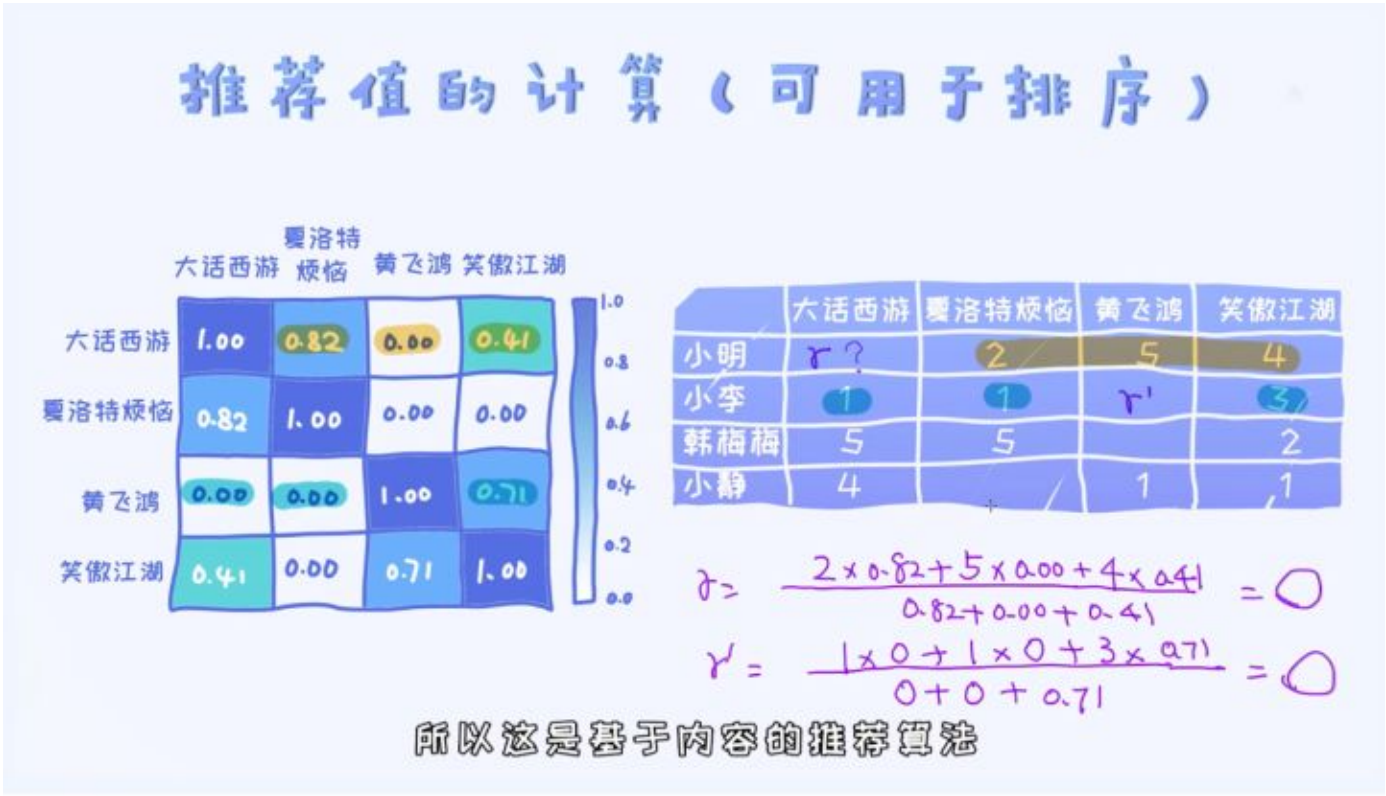
$$= \frac{a \bullet b}{||a|| \times ||b||}$$

公式(4)

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。

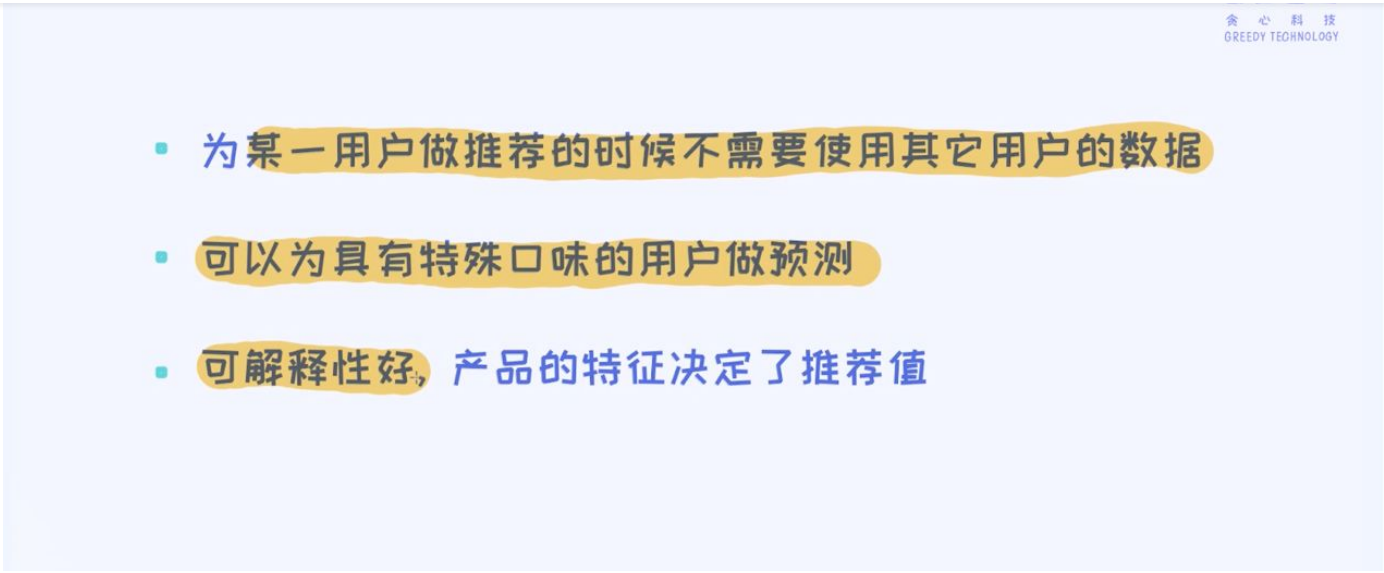


2.2相似度排序推荐

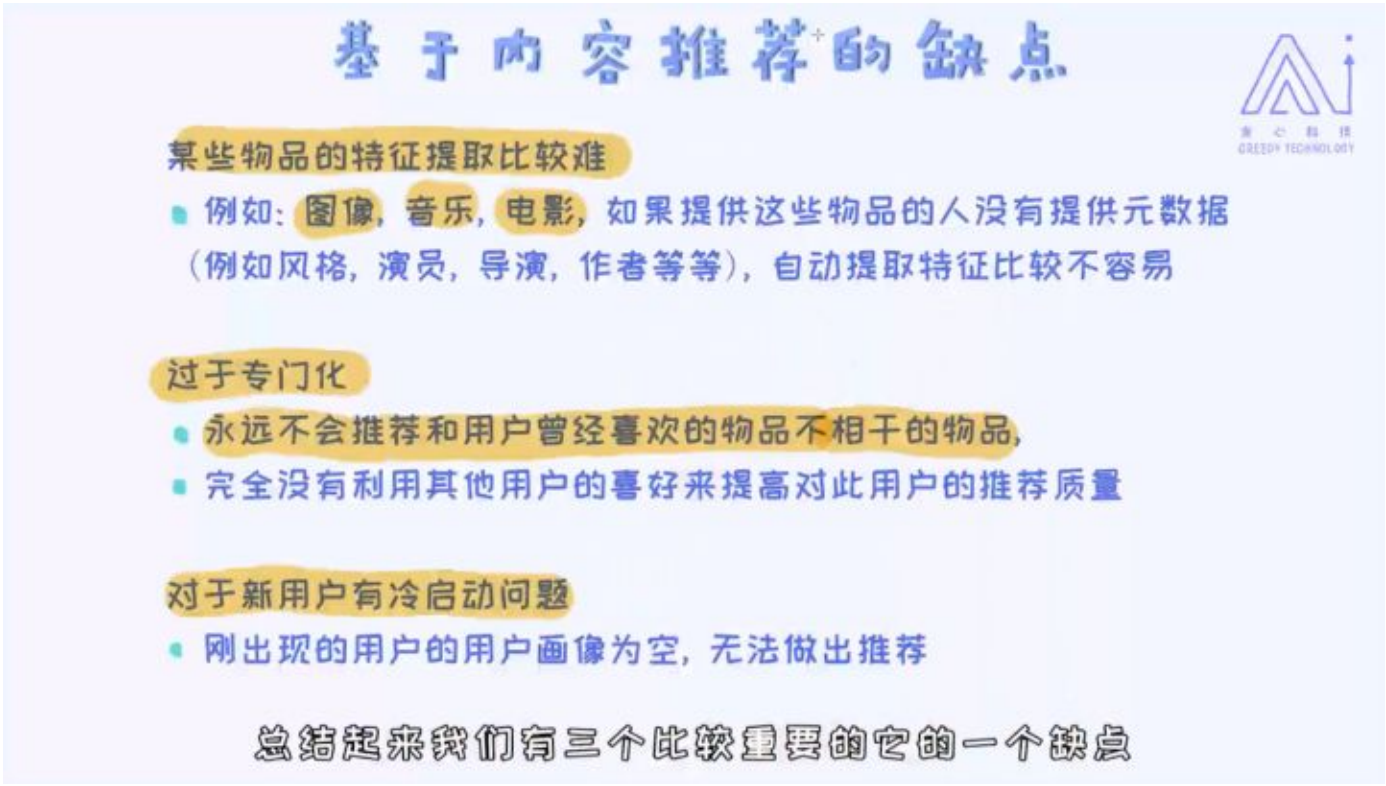


2.3基于内容推荐算法的优却点

优点：推荐较为准确

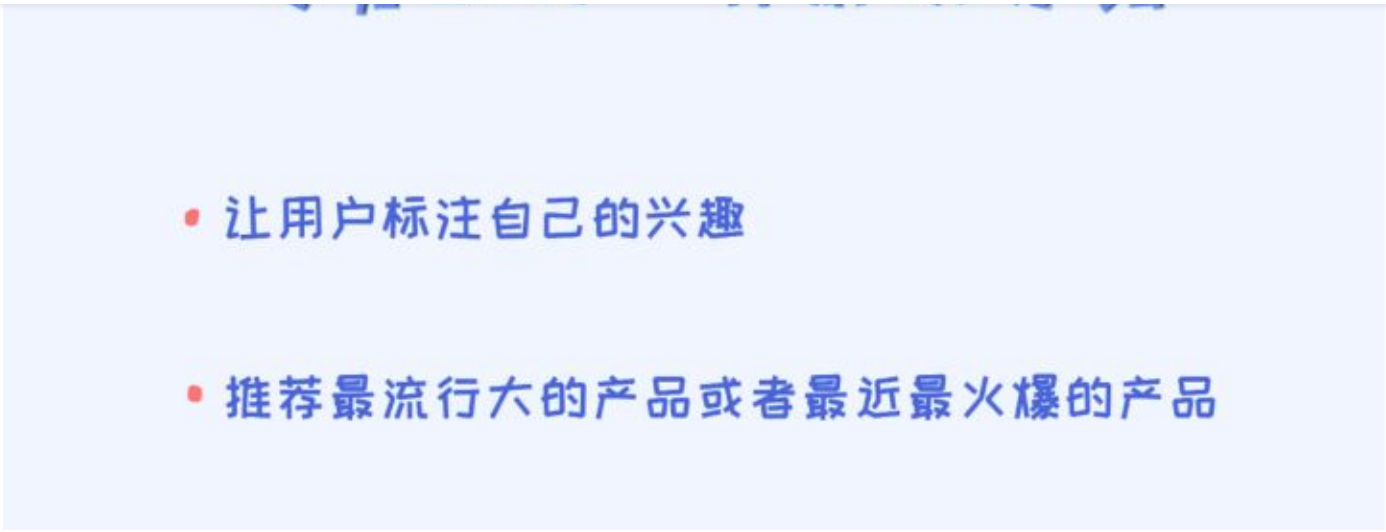


缺点：（主要冷启动问题）



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。



解决冷启动问题总结：

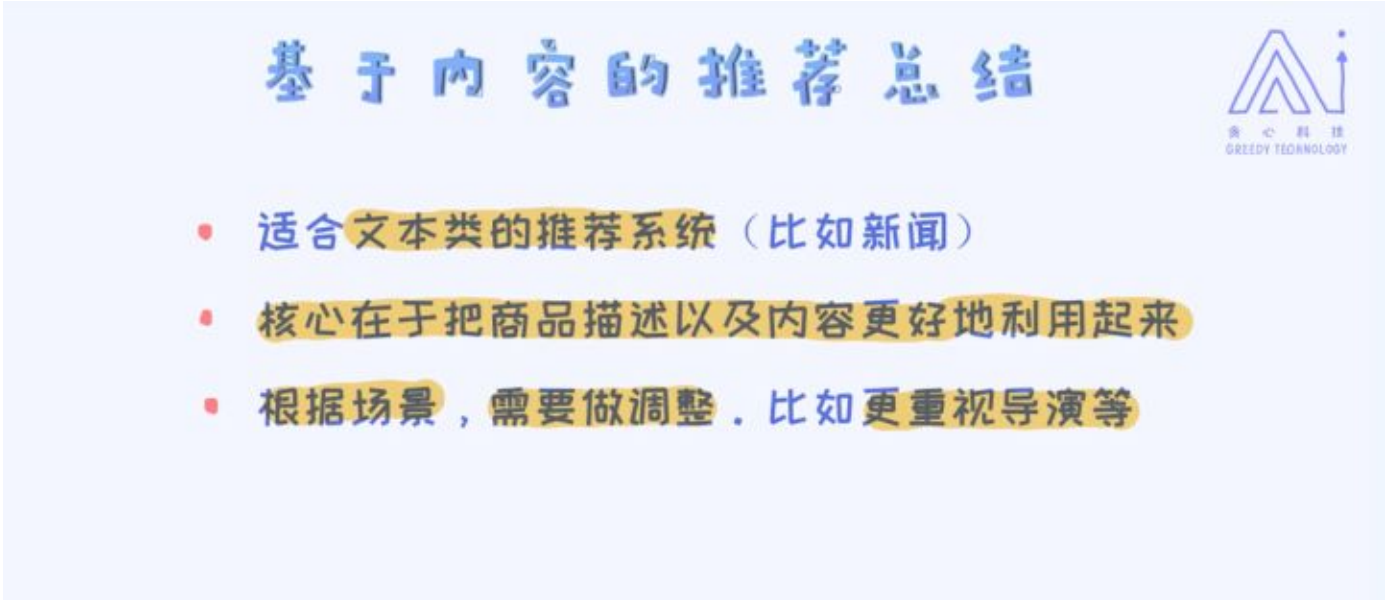
1、 推荐目前热度最高的商品；

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。



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