# 课程介绍

# 课程内容

推荐系统在电子商务领域得到普遍的运用

**推荐系统本质上是销售系统的一部分**

在便利店，推荐系统是导购牌，类目货架，是老板娘

在超市，推荐系统是导购牌，类目货架，是销售员

在电商，推荐系统是什么？

不管是在便利店，还是超市，或者电商网站，本质上需要解决两个问题：

1、帮助用户获得想要的商品

A、用户知道自己想要什么商品，在什么位置 **用户主动的行为**

直接到货架区域去挑选商品，直接询问销售员商品在哪里

B、用户不知道自己想要什么，比如：逛街兴致。 **用户被动的行为**

2、帮助商家卖出更多的商品----**商品的质量**

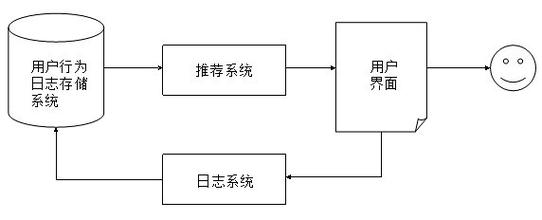
所有商家中同质同量的商品比较多，如何让不同商家的利益得到保证。

在线下，哪个厂家的商品好卖，我就多进点货。在同等质量保证的而前提下，关系户的货物。

在线上，由于电商网站的利润一般是抽取销售额提成，只有更多的商家参与，才能做大市场和品牌。

# 1、推荐系统是什么

**为了解决信息过载和用户无明确需求的问题，找到用户感兴趣的物品，才有了个性化推荐系统。**其实，解决信息过载的问题，代表性的解决方案是分类目录和搜索引擎，如hao123，电商首页的分类目录以及百度，360搜索等。不过分类目录和搜索引擎只能解决用户主动查找信息的需求，即用户知道自己想要什么，并不能解决用户没用明确需求很随便的问题。经典语录是：你想吃什么，随便！面对这种很随便又得罪不起的用户（女友和上帝），只能通过分析用户的历史行为给用户的兴趣建模，从而主动给用户推荐能够满足他们兴趣和需求的信息。比如问问女友的闺蜜，她一般什么时候喜欢吃什么。



        推荐系统广泛存在于各类网站中，作为一个应用为用户提供个性化的推荐。它需要一些用户的历史数据，一般由三个部分组成：基础数据、推荐算法系统、前台展示**。基础数据包括很多维度，包括用户的访问、浏览、下单、收藏，用户的历史订单信息，评价信息等很多信息；推荐算法系统主要是根据不同的推荐诉求由多个算法组成的推荐模型；前台展示主要是对客户端系统进行响应，返回相关的推荐信息以供展示。**

**基础数据**主要包括：

* 要推荐物品或内容的元数据，例如关键字，基因描述等；
* 系统用户的基本信息，例如性别，年龄等
* 用户对物品或者信息的偏好，根据应用本身的不同，可能包括用户对物品的评分，用户查看物品的记录，用户的购买记录等。

其实这些用户的偏好信息可以分为两类：

* 显式的用户反馈：这类是用户在网站上自然浏览或者使用网站以外，显式的提供反馈信息，例如用户对物品的评分，或者对物品的评论。
* 隐式的用户反馈：这类是用户在使用网站是产生的数据，隐式的反应了用户对物品的喜好，例如用户查看了某物品的信息等等。

显式的用户反馈能准确的反应用户对物品的真实喜好，但需要用户付出额外的代价，而隐式的用户行为，通过一些分析和处理，也能反映用户的喜好，只是数 据不是很精确，有些行为的分析存在较大的噪音。但只要选择正确的行为特征，隐式的用户反馈也能得到很好的效果，只是行为特征的选择可能在不同的应用中有很 大的不同，例如在电子商务的网站上，购买行为其实就是一个能很好表现用户喜好的隐式反馈。

# 2、推荐引擎的分类

推荐引擎的分类可以根据很多指标进行区分：

* 根据目标用户进行区分：根据这个指标可以分为基于大众行为的推荐引擎和个性化推荐引擎。
* **根据大众行为的推荐引擎**，对每个用户都给出同样的推荐，这些推荐可以是静态的由系统管理员人工设定的，或者基于系统所有用户的反馈统计计算出的当下比较流行的物品。
* **个性化推荐引擎**，对不同的用户，根据他们的口味和喜好给出更加精确的推荐，这时，系统需要了解需推荐内容和用户的特质，或者基于社会化网络，通过找到与当前用户相同喜好的用户，实现推荐。

这是一个最基本的推荐引擎分类，其实大部分人们讨论的推荐引擎都是将个性化的推荐引擎，因为从根本上说，只有个性化的推荐引擎才是更加智能的信息发现过程。

* 根据数据源进行区分：主要是根据数据之间的相关性进行推荐，因为大部分推荐引擎的工作原理还是基于物品或者用户的相似集进行推荐。
* 根据系统**用户**的基本信息发现用户的相关程度，这种被称为基于人口统计学的推荐（Demographic-based Recommendation）
* 根据推荐**物品**或内容的元数据，发现物品或者内容的相关性，这种被称为基于内容的推荐（Content-based Recommendation）
* 根据用户对物品或者信息的偏好，发现物品或者内容本身的相关性，或者是发现用户的相关性，这种被称为基于协同过滤的推荐（Collaborative Filtering-based Recommendation）。
* 根根据推荐模型进行区分：可以想象在海量物品和用户的系统中，推荐引擎的计算量是相当大的，要实现实时的推荐务必需要建立一个推荐模型，关于推荐模型的建立方式可以分为以下几种：
* 基于物品和用户本身的，这种推荐引擎将每个用户和每个物品都当作独立的实体，预测每个用户对于每个物品的喜好程度，这些信息往往 是用一个二维矩阵描述的。由于用户感兴趣的物品远远小于总物品的数目，这样的模型导致大量的数据空置，即我们得到的二维矩阵往往是一个很大的稀疏矩阵。同 时为了减小计算量，我们可以对物品和用户进行聚类， 然后记录和计算一类用户对一类物品的喜好程度，但这样的模型又会在推荐的准确性上有损失。
* 基于关联规则的推荐（Rule-based Recommendation）：关联规则的挖掘已经是数据挖掘中的一个经典的问题，主要是挖掘一些数据的依赖关系，典型的场景就是“购物篮问题”，通过关联规则的挖掘，我们可以找到哪些物品经常被同时购买，或者用户购买了一些物品后通常会购买哪些其他的物品，当我们挖掘出这些关联规则之后，我们可以基于这些规则给用户进行推荐。
* 基于模型的推荐（Model-based Recommendation）：这是一个典型的机器学习的问题，可以将已有的用户喜好信息作为训练样本，训练出一个预测用户喜好的模型，这样以后用户在 进入系统，可以基于此模型计算推荐。这种方法的问题在于如何将用户实时或者近期的喜好信息反馈给训练好的模型，从而提高推荐的准确度。

其实在现在的推荐系统中，很少有只使用了一个推荐策略的推荐引擎，一般都是在不同的场景下使用不同的推荐策略从而达到最好的推荐效果，例如 Amazon 的推荐，它将基于用户本身历史购买数据的推荐，和基于用户当前浏览的物品的推荐，以及基于大众喜好的当下比较流行的物品都在不同的区域推荐给用户，让用户 可以从全方位的推荐中找到自己真正感兴趣的物品。

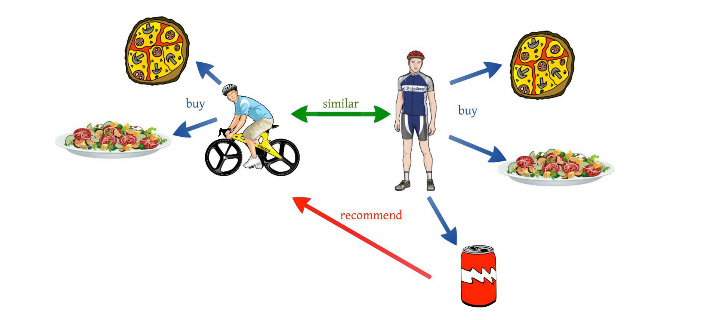
# 3、常见的推荐算法

        迄今为止，在个性化推荐系统中，协同过滤技术是应用最成功的技术。目前国内外有许多大型网站应用这项技术为用户更加智能的推荐内容。

## 3.1、基于用户的协同过滤算法

第一代协同过滤技术是基于用户的协同过滤算法，基于用户的协同过滤算法在推荐系统中获得了极大的成功，但它有自身的局限性。因为基于用户的协同过滤算法先计算的是用户与用户的相似度（兴趣相投，人以群分物以类聚），然后将相似度比较接近的用户A购买的物品推荐给用户B，专业的说法是该算法用**最近邻居**（nearest-neighbor）算法找出一个用户的邻居集合，该集合的用户和该用户有相似的喜好，算法根据邻居的偏好对该用户进行预测。

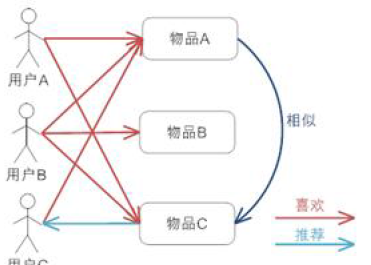
基于用户的推荐逻辑有**两个问题**：冷启动与2计算量巨大。基于用户的算法只有已经被用户选择（购买）的物品才有机会推荐给其他用户。在大型电商网站上来讲，商品的数量实在是太多了，没有被相当数量的用户购买的物品实在是太多了，直接导致没有机会推荐给用户了，这个问题被称之为协同过滤的“冷启动”。另外，因为计算用户的相似度是通过目标用户的历史行为记录与其他每一个用户的记录相比较的出来的，对于一个拥有千万级活跃用户的电商网站来说，每计算一个用户都涉及到了上亿级别的计算，虽然我们可以先通过聚类算法经用户先分群，但是计算量也是足够的大。



## 3.2、基于物品的协同过滤算法

第二代协同过滤技术是基于物品的协同过滤算法，基于物品的协同过滤算法与基于用户的协同过滤算法基本类似。他使用所有用户对物品或者信息的偏好，发现物品和物品之间的相似度，然后根据用户的历史偏好信息，将类似的物品推荐给用户。**这听起来比较拗口，简单的说就是几件商品同时被人购买了，就可以认为这几件商品是相似的**，可能这几件商品的商品名称风马牛不相及，产品属性有天壤之别，但通过模型算出来之后就是认为他们是相似的。**什么？你觉得不可思议，无法理解。是的，就是这么神奇**！

举个例子：假设用户 A 喜欢物品 A 和物品 C，用户 B 喜欢物品 A，物品 B 和物品 C，用户 C 喜欢物品 A，从这些用户的历史喜好可以分析出物品 A 和物品 C 时比较类似的，喜欢物品 A 的人都喜欢物品 C，基于这个数据可以推断用户 C 很有可能也喜欢物品 C，所以系统会将物品 C 推荐给用户 C。



其实基于项目的协同过滤推荐机制是 Amazon 在基于用户的机制上改良的一种策略，因为在大部分的 Web 站点中，物品的个数是远远小于用户的数量的，而且物品的个数和相似度相对比较稳定，同时基于项目的机制比基于用户的实时性更好一些。但也不是所有的场景都 是这样的情况，可以设想一下在一些新闻推荐系统中，也许物品，也就是新闻的个数可能大于用户的个数，而且新闻的更新程度也有很快，所以它的形似度依然不稳 定。所以，其实可以看出，推荐策略的选择其实和具体的应用场景有很大的关系。

## 3.3、其他推荐算法

### 3.3.1、基于人口统计学的推荐

基于人口统计学的推荐机制（Demographic-based Recommendation）是一种最易于实现的推荐方法，它只是简单的**根据系统用户的基本信息发现用户的相关程度(如相同地区、相同年龄段)**，然后将相似用户喜爱的其他物品推荐给当前用户。基于人口统计学的推荐机制的好处在于：因为**不使用当前用户对物品的喜好历史数据**，所以对于**新用户**来讲没有“**冷启动（Cold Start）**”的问题。这个方法不依赖于物品本身的数据，所以这个方法在不同物品的领域都可以使用，它是领域独立的（domain-independent）

### 3.3.2、基于内容的推荐

基于内容的推荐是在推荐引擎出现之初应用最为广泛的推荐机制，它的核心思想是根据推荐物品或内容的元数据，发现物品或者内容的相关性，然后基于用户以往的喜好记录，推荐给用户相似的物品。基于内容的推荐机制的好处在于它能很好的建模用户的口味，能提供更加精确的推荐。

但它也存在以下几个问题：

* 需要对物品进行分析和建模，推荐的质量依赖于对物品模型的完整和全面程度。在现在的应用中我们可以观察到**关键词**和**标签**（Tag）被认为是描述物品元数据的一种简单有效的方法；
* 物品相似度的分析仅仅依赖于物品本身的特征，这里没有考虑人对物品的态度；
* 因为需要基于用户以往的喜好历史做出推荐，所以对于新用户有“冷启动”的问题。

虽然这个方法有很多不足和问题，但他还是成功的应用在一些电影，音乐，图书的社交站点，有些站点还请专业的人员对物品进行基因编码，比如潘多拉，在一份报告中说道，在潘多拉的推荐引擎中，每首歌有超过 100 个元数据特征，包括歌曲的风格，年份，演唱者等等。

# 4、混合的推荐机制(重要)

    在现行的 Web 站点上的推荐往往都不是单纯只采用了某一种推荐的机制和策略，他们往往是将多个方法混合在一起，从而达到更好的推荐效果。关于如何组合各个推荐机制，这里讲几种比较流行的组合方法。

* 加权的混合（Weighted Hybridization）: 用线性公式（linear formula）将几种不同的推荐按照一定权重组合起来，具体权重的值需要在测试数据集上反复实验，从而达到最好的推荐效果。
* 切换的混合（Switching Hybridization）：前面也讲到，其实对于不同的情况（数据量，系统运行状况，用户和物品的数目等），推荐策略可能有很大的不同，那么切换的混合方式，就是允许在不同的情况下，选择最为合适的推荐机制计算推荐。
* 分区的混合（Mixed Hybridization）：采用多种推荐机制，并将不同的推荐结果分不同的区显示给用户。其实，Amazon，当当网等很多电子商务网站都是采用这样的方式，用户可以得到很全面的推荐，也更容易找到他们想要的东西。
* 分层的混合（Meta-Level Hybridization）: 采用多种推荐机制，并将一个推荐机制的结果作为另一个的输入，从而综合各个推荐机制的优缺点，得到更加准确的推荐。

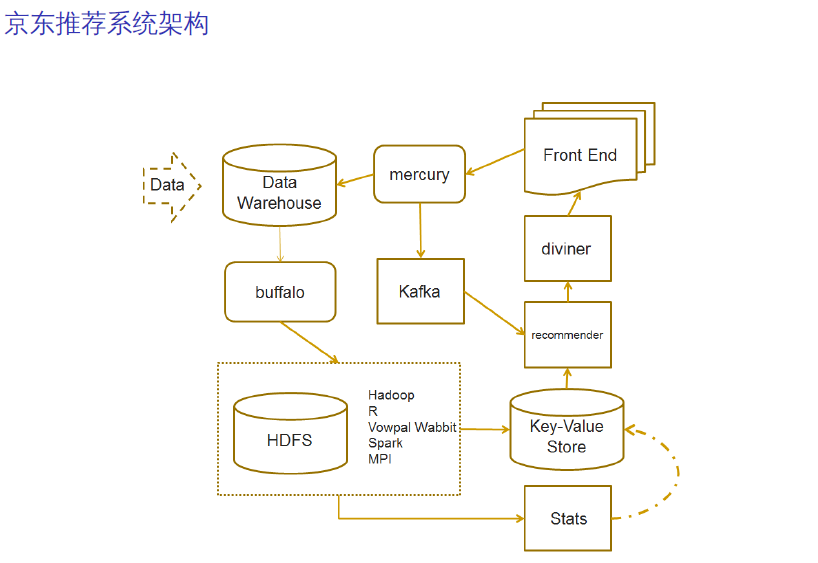
# 5、推荐系统的应用场景

Amazon 利用可以记录的所有用户在站点上的行为，根据不同数据的特点对它们进行处理，并分成不同区为用户推送推荐：

* 今日推荐 (Today's Recommendation For You): 通常是根据用户的近期的历史购买或者查看记录，并结合时下流行的物品给出一个折中的推荐。
* 新产品的推荐 (New For You): 采用了基于内容的推荐机制 (Content-based Recommendation)，将一些新到物品推荐给用户。在方法选择上由于新物品没有大量的用户喜好信息，所以基于内容的推荐能很好的解决这个“冷启动”的问题。
* 捆绑销售 (Frequently Bought Together): 采用数据挖掘技术对用户的购买行为进行分析，找到经常被一起或同一个人购买的物品集，进行捆绑销售，这是一种典型的基于项目的协同过滤推荐机制。
* 别人购买 / 浏览的商品 (Customers Who Bought/See This Item Also Bought/See): 这也是一个典型的基于项目的协同过滤推荐的应用，通过社会化机制用户能更快更方便的找到自己感兴趣的物品。

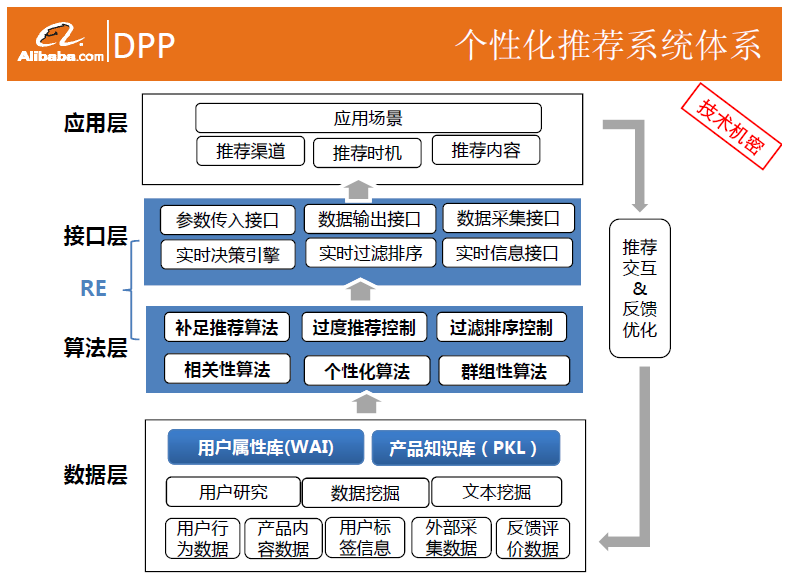
# 推荐系统架构

## 6.1、京东推荐系统架构



（以上图片来自 京东刘思哲的《京东推荐系统实战》）

## 6.2、淘宝推荐系统架构

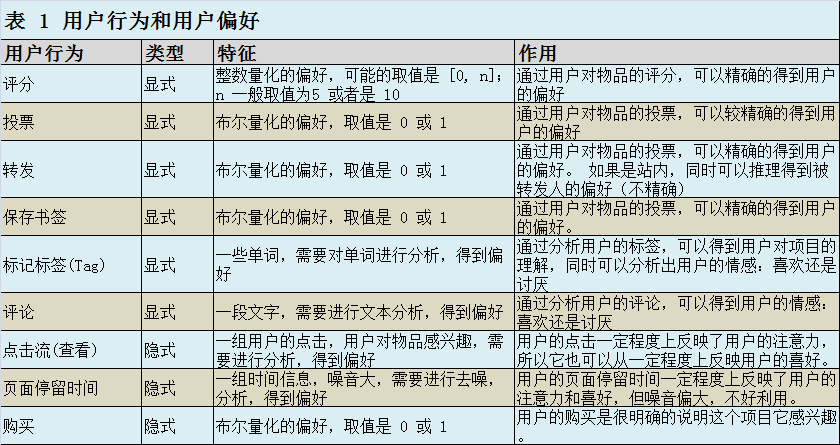


（以上信息来自阿里巴巴数据产品平台部公开的资料）

# 协同过滤的实现

## 7.1、收集用户偏好及标准化处理

要从用户的行为和偏好中发现规律，并基于此给予推荐，如何收集用户的偏好信息成为**系统推荐效果最基础的决定因素**。用户有很多方式向系统提供自己的偏好信息，而且不同的应用也可能大不相同，下面举例进行介绍：

A 1\*0.5 +1\*0.3+ 1\*0.3+ 0.2\*0.1+ 0.3\*0.2 +1\*1.0 = ?           

以上列举的用户行为都是比较通用的，推荐引擎设计人员可以根据自己应用的特点添加特殊的用户行为，并用他们表示用户对物品的喜好。在一般应用中，我们提取的用户行为一般都多于一种，关于如何组合这些不同的用户行为，基本上有以下两种方式：

* 将不同的行为分组：一般可以分为“查看”和“购买”等等，然后基于不同的行为，计算不同的用户 / 物品相似度。比如：“购买了该图书的人还购买了 ...”，“查看了图书的人还查看了 ...”
* 根据不同行为反映用户喜好的程度将它们进行加权，得到用户对于物品的总体喜好。一般来说，显式的用户反馈比隐式的权值大，但比较稀疏，毕竟进行显示反馈的用户是少数；同时相对于“查看”，“购买”行为反映用户喜好的程度更大，但这也因应用而异。

## 7.2、数据减噪和归一化

  收集了用户行为数据，我们还需要对数据进行一定的预处理，其中最核心的工作就是：减噪和归一化。

* 减噪：用户行为数据是用户在使用应用过程中产生的，它可能存在大量的噪音和用户的误操作，我们可以通过经典的数据挖掘算法过滤掉行为数据中的噪音，这样可以是我们的分析更加精确。
* 归一化：如前面讲到的，在计算用户对物品的喜好程度时，可能需要对不同的行为数据进行加权。但可以想象，不同行为的数据取值可能相差很 大，比如，用户的查看数据必然比购买数据大的多，如何将各个行为的数据统一在一个相同的取值范围中，从而使得加权求和得到的总体喜好更加精确，就需要我们 进行归一化处理。最简单的归一化处理，就是将各类数据除以此类中的最大值，以保证归一化后的数据取值在 [0,1] 范围中。

进行的预处理后，根据不同应用的行为分析方法，可以选择分组或者加权处理，之后我们可以得到一个用户偏好的二维矩阵，一维是用户列表，另一维是物品列表，值是用户对物品的偏好，一般是 [0,1] 或者 [-1, 1] 的浮点数值。

## 7.3、找到相似的用户或物品

当已经对用户行为进行分析得到用户喜好后，我们可以根据用户喜好计算相似用户和物品，然后基于相似用户或者物品进行推荐，这就是最典型的 CF 的两个分支：基于用户的 CF 和基于物品的 CF。这两种方法都需要计算**相似度**。关于相似度的计算，现有的几种基本方法都是基于向量（Vector）的 ，其实也就是计算两个向量的距离，距离越近相似度越大。在推荐的场景中，在用 户 - 物品偏好的二维矩阵中，我们可以将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，或者将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度。

1,1193,5

1,661,3

1,914,3

1,3408,4

1,2355,5

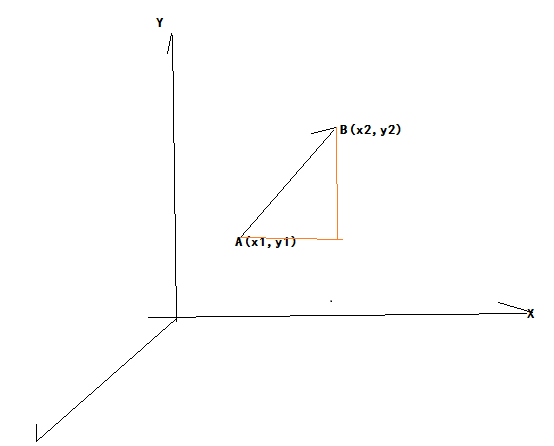
1193 661 914 3408 2355

1 5 3 3 4 5

2 5 3 3 4 5

3 5 3 3 4 5

1193 661:0.5 914:0.45 3408:0.3

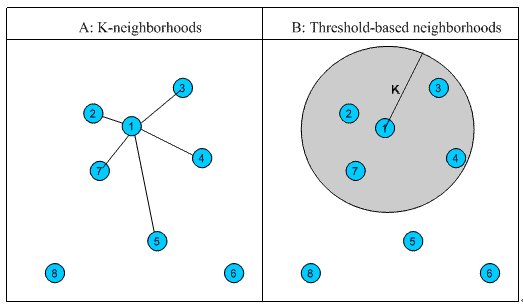


### 7.3.1、常用的相似度计算方法

* 欧几里德距离（Euclidean Distance）
* 皮尔逊相关系数（Pearson Correlation Coefficient）
* Tanimoto 系数（Tanimoto Coefficient）

### 7.3.2、相似邻居的计算

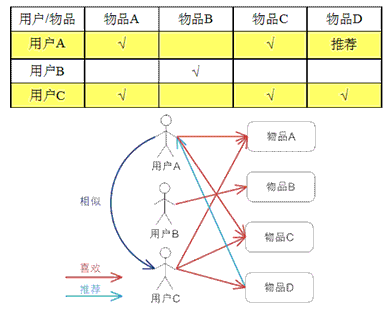
* 固定数量的邻居：K-neighborhoods 或者 Fix-size neighborhoods
* 基于相似度门槛的邻居：Threshold-based neighborhoods



## 7.4、计算基于用户的 CF（User CF）

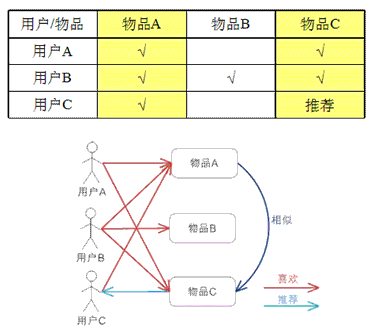
基于用户的 CF 的基本思想相当简单，基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。计算上，就是将一个用户对所有物品的偏好作为一个向量 来计算用户之间的相似度，找到 K 邻居后，根据邻居的

相似度权重以及他们对物品的偏好，预测当前用户没有偏好的未涉及物品，计算得到一个排序的物品列表作为推荐。图 2 给出了一个例子，对于用户 A，根据用户的历史偏好，这里只计算得到一个邻居 - 用户 C，然后将用户 C 喜欢的物品 D 推荐给用户 A。



## 7.5、基于物品的 CF（Item CF）

基于物品的 CF 的原理和基于用户的 CF 类似，只是在计算邻居时采用物品本身，而不是从用户的角度，即基于用户s对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。从计算 的角度看，就是将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度，得到物品的相似物品后，根据用户历史的偏好预测当前用户还没有表示偏好的 物品，计算得到一个排序的物品列表作为推荐。图 3 给出了一个例子，对于物品 A，根据所有用户的历史偏好，喜欢物品 A 的用户都喜欢物品 C，得出物品 A 和物品 C 比较相似，而用户 C 喜欢物品 A，那么可以推断出用户 C 可能也喜欢物品 C。



## 7.6、User CF vs. Item CF

### 7.6.1、计算复杂度

Item CF 和 User CF 是基于协同过滤推荐的两个最基本的算法，User CF 是很早以前就提出来了，Item CF 是从 Amazon 的论文和专利发表之后（2001 年左右）开始流行，**大家都觉得 Item CF 从性能和复杂度上比 User CF 更优，其中的一个主要原因就是对于一个在线网站，用户的数量往往大大超过物品的数量，同时物品的数据相对稳定，因此计算物品的相似度不但计算量较小，同时 也不必频繁更新。**但我们往往忽略了这种情况只适应于提供商品的电子商务网站，对于新闻，博客或者微内容的推荐系统，情况往往是相反的，物品的数量是海量 的，同时也是更新频繁的，所以单从复杂度的角度，这两个算法在不同的系统中各有优势，推荐引擎的设计者需要根据自己应用的特点选择更加合适的算法。

### 7.6.2、适用场景

在非社交网络的网站中，内容内在的联系是很重要的推荐原则，它比基于相似用户的推荐原则更加有效。比如在购书网站上，当你看一本书的时候，推荐引擎 会给你推荐相关的书籍，这个推荐的重要性远远超过了网站首页对该用户的综合推荐。可以看到，在这种情况下，Item CF 的推荐成为了引导用户浏览的重要手段。同时 Item CF 便于为推荐做出解释，在一个非社交网络的网站中，给某个用户推荐一本书，同时给出的解释是某某和你有相似兴趣的人也看了这本书，这很难让用户信服，因为用 户可能根本不认识那个人；但如果解释说是因为这本书和你以前看的某本书相似，用户可能就觉得合理而采纳了此推荐。相反的，在现今很流行的社交网络站点中，User CF 是一个更不错的选择，User CF 加上社会网络信息，可以增加用户对推荐解释的信服程度。

### 7.5.3、推荐多样性和精度

研究推荐引擎的学者们在相同的数据集合上分别用 User CF 和 Item CF 计算推荐结果，发现推荐列表中，只有 50% 是一样的，还有 50% 完全不同。但是这两个算法确有相似的精度，所以可以说，这两个算法是很互补的。

关于推荐的多样性，有两种度量方法：

第一种度量方法是从单个用户的角度度量，就是说给定一个用户，查看系统给出的推荐列表是否多样，也就是要比较推荐列表中的物品之间两两的相似度，不 难想到，对这种度量方法，Item CF 的多样性显然不如 User CF 的好，因为 Item CF 的推荐就是和以前看的东西最相似的。

第二种度量方法是考虑系统的多样性，也被称为覆盖率 (Coverage)，它是指一个推荐系统是否能够提供给所有用户丰富的选择。在这种指标下，Item CF 的多样性要远远好于 User CF, 因为 User CF 总是倾向于推荐热门的，从另一个侧面看，也就是说，Item CF 的推荐有很好的新颖性，很擅长推荐长尾里的物品。所以，尽管大多数情况，Item CF 的精度略小于 User CF， 但如果考虑多样性，Item CF 却比 User CF 好很多。

如果你对推荐的多样性还心存疑惑，那么下面我们再举个实例看看 User CF 和 Item CF 的多样性到底有什么差别。

首先，假设每个用户兴趣爱好都是广泛的，喜欢好几个领域的东西，不过每个用户肯定也有一个主要的领域，对这个领域会比其他领域更加关心。

给定一个用户，假设他喜欢 3 个领域 A,B,C，A 是他喜欢的主要领域，这个时候我们来看 User CF 和 Item CF 倾向于做出什么推荐：如果用 User CF, 它会将 A,B,C 三个领域中比较热门的东西推荐给用户；而如果用 ItemCF，它会基本上只推荐 A 领域的东西给用户。所以我们看到因为 User CF 只推荐热门的，所以它在推荐长尾里项目方面的能力不足；而 Item CF 只推荐 A 领域给用户，这样他有限的推荐列表中就可能包含了一定数量的不热门的长尾物品，同时 Item CF 的推荐对这个用户而言，显然多样性不足。但是对整个系统而言，因为不同的用户的主要兴趣点不同，所以系统的覆盖率会比较好。

从上面的分析，可以很清晰的看到，这两种推荐都有其合理性，但都不是最好的选择，因此他们的精度也会有损失。其实对这类系统的最好选择是，如果系统 给这个用户推荐 30 个物品，既不是每个领域挑选 10 个最热门的给他，也不是推荐 30 个 A 领域的给他，而是比如推荐 15 个 A 领域的给他，剩下的 15 个从 B,C 中选择。所以结合 User CF 和 Item CF 是最优的选择，结合的基本原则就是当采用 Item CF 导致系统对个人推荐的多样性不足时，我们通过加入 User CF 增加个人推荐的多样性，从而提高精度，而当因为采用 User CF 而使系统的整体多样性不足时，我们可以通过加入 Item CF 增加整体的多样性，同样同样可以提高推荐的精度。

### 7.5.4用户对推荐算法的适应度

前面我们大部分都是从推荐引擎的角度考虑哪个算法更优，但其实我们更多的应该考虑作为推荐引擎的最终使用者 -- 应用用户对推荐算法的适应度。

对于 User CF，推荐的原则是假设用户会喜欢那些和他有相同喜好的用户喜欢的东西，但如果一个用户没有相同喜好的朋友，那 User CF 的算法的效果就会很差，所以一个用户对的 CF 算法的适应度是和他有多少共同喜好用户成正比的。

Item CF 算法也有一个基本假设，就是用户会喜欢和他以前喜欢的东西相似的东西，那么我们可以计算一个用户喜欢的物品的自相似度。一个用户喜欢物品的自相似度大，就 说明他喜欢的东西都是比较相似的，也就是说他比较符合 Item CF 方法的基本假设，那么他对 Item CF 的适应度自然比较好；反之，如果自相似度小，就说明这个用户的喜好习惯并不满足 Item CF 方法的基本假设，那么对于这种用户，用 Item CF 方法做出好的推荐的可能性非常低。