# 推荐系统业务实践

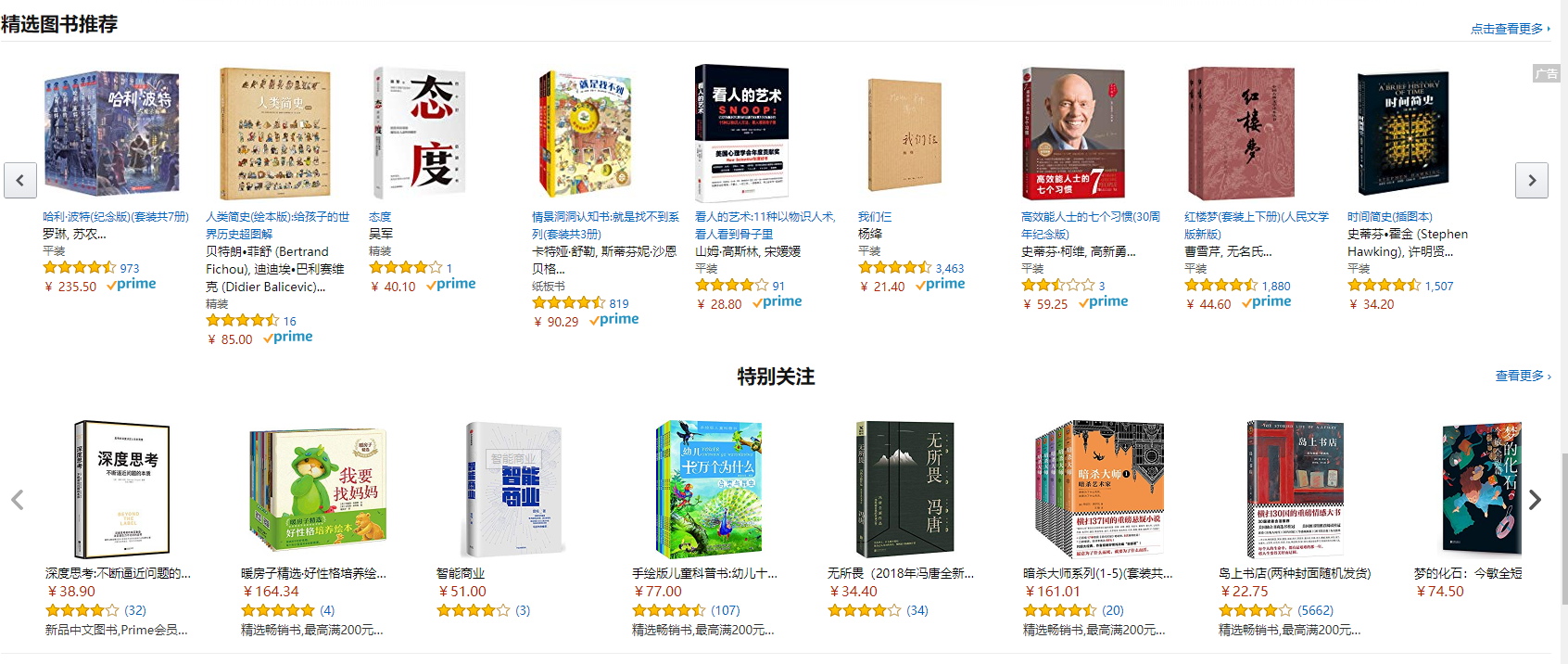
## 每日目标

通过学习推荐系统业务方面知识，熟悉推荐系统业务场景、推荐架构设计，通过对互联网公司推荐架构分析，设计一套推荐系统架构。为成为一名推荐算法工程师提供业务支持。

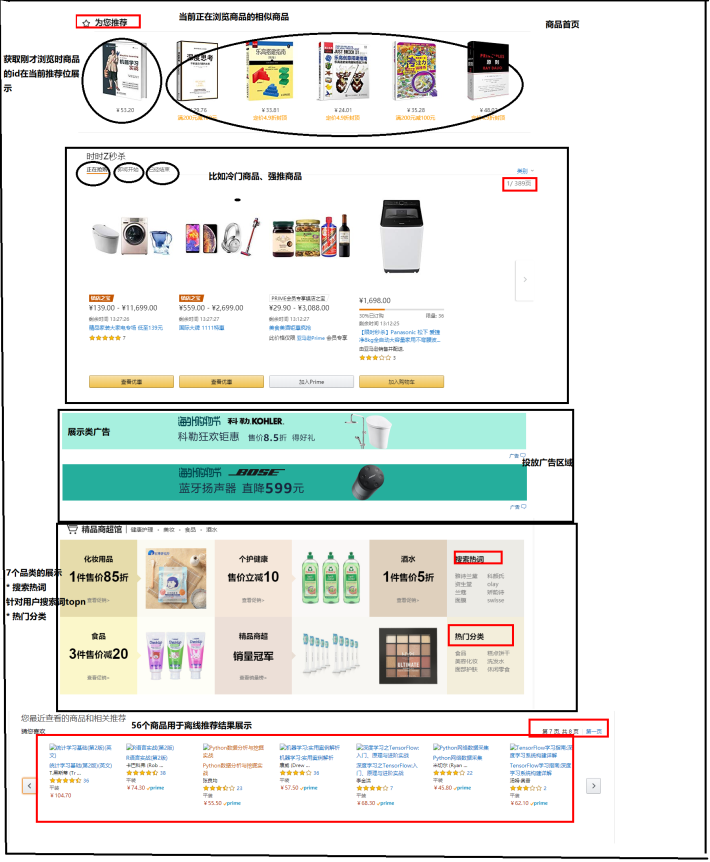
# 亚马逊推荐系统业务理解

当进入首页没有搜索行为，看到下面推荐

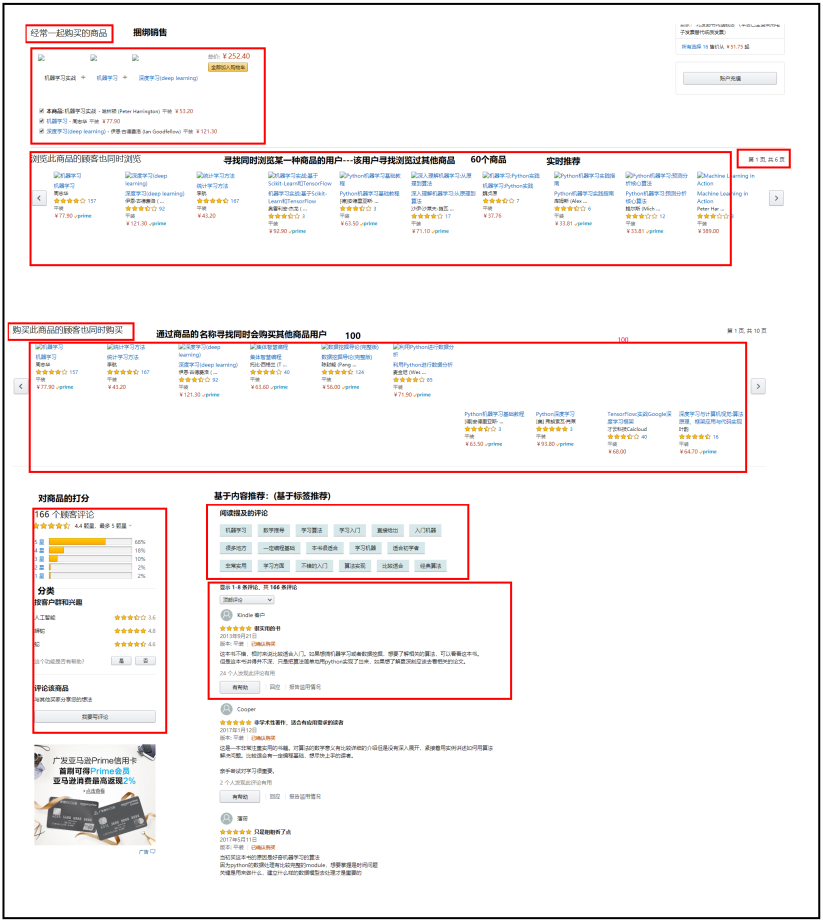




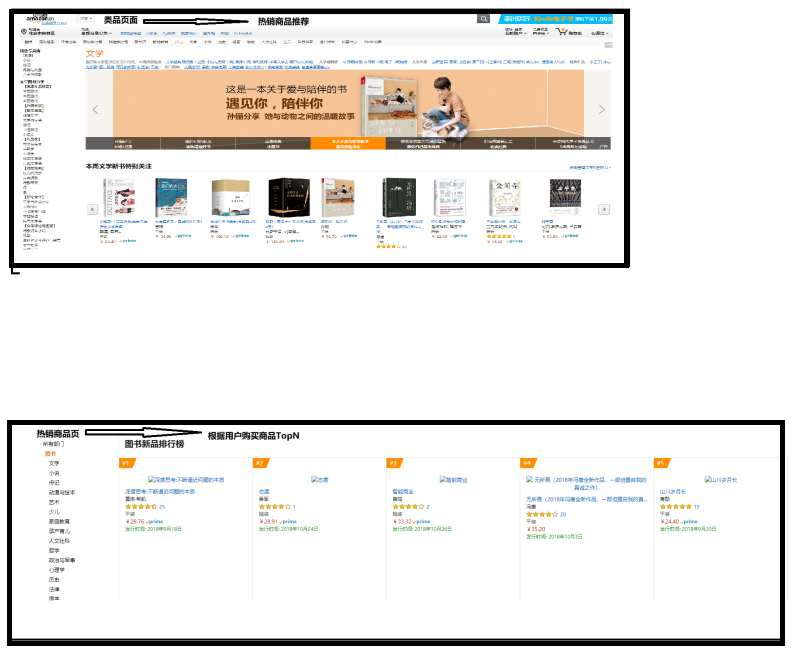
## 0.1首页推荐



## 0.2商品推荐页推荐



## 0.3品牌页推荐



# 1.推荐系统引入

## 1.1什么是推荐系统？

想买啤酒，通过附近的便利店，通过比较几个牌子的口碑或者价格找到自己喜欢的，扫码付款。如果你很宅，你可以打开京东或天猫超市输入啤酒关键字，然后会看到一大堆关于啤酒的商品，找到喜欢的品牌下单、付款然后等待送货上门。

上述都是用户在有明确需求的情况下，**面对信息过载**的时候。

所采用的措施。如果**用户没有明确的需求**？比如你今天很无聊，想下载一部电影，但是你面对如此之多的电影，你会手足无措，不知道该看哪一部。此时你遇到了信息过载的问题，需要一个人或工具帮助你完成筛选，如果你有一个好朋友你可以询问他，

不过，总不能时时刻刻都去麻烦“专家”给你推荐，你需要的是一个自动化工具，它

可以分析你的兴趣，从庞大的电影库中找到几部符合你兴趣的电影供你选择。

这个工具就是**个性化推荐系统。**

**推荐系统：是在面对用户信息过载的情况下，或者在用户没有明确购买信息的情况下，通过程序或工具自动化实现物品或商品的自动推荐过程。(而不依赖于外部专家)**

## 1.2案例引入

我们一天会看到多达20M左右的文字信息，会听到600mb左右的声音信息，每秒会看到2MB左右的图像信息。

每天有10W左右的新闻报道。

每秒钟优酷土豆爱奇艺都会多出长达几百小时的视频

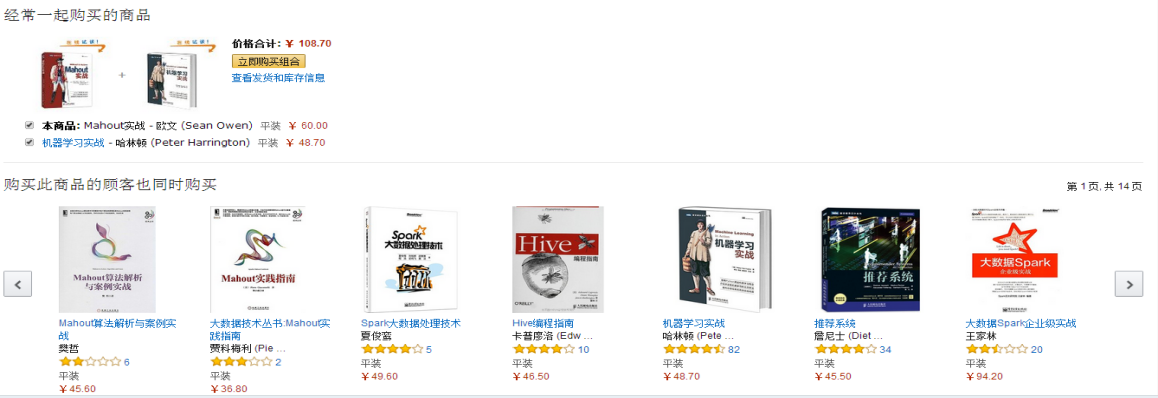
淘宝天猫京东亚马逊当当一天上架上百万商品。

【微博】大家在使用微博过程中，每天都有大量的微博被创作和转发，信息量的爆炸式增长在给用户不断带来新的信息的同时，也增加了用户筛选信息的难度，为了能够为用户推荐其感兴趣的信息，我们**首先分析该用户的兴趣，从海量信息中选择出与用户兴趣相似的信息，并将这些信息推荐给用户**。

推荐系统正是在这样的背景下被提出的，推荐算法根据用户的历史行为，挖掘出用户的喜好，并为用户推荐与其喜好相符的商品或者信息。

**推荐系统的任务就是能够连接信息与用户，帮助用户找到其感兴趣的信息，同时让一些有价值的信息能够到达潜在的用户中。**





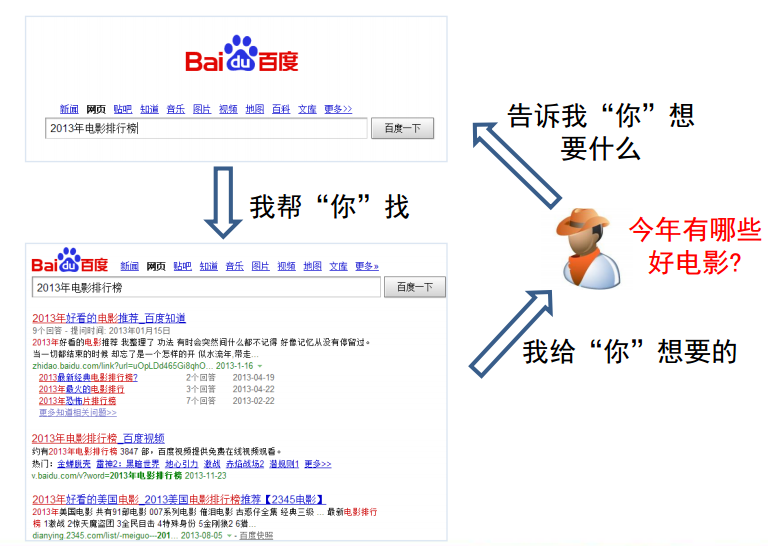




## 1.3为什么使用推荐系统

随着互联网行业的发展，信息量也在以几何倍数式爆发增长。垃圾信息越来越多，导致用户获取有价值信息的成本大大增加。由于信息的爆炸式增长，对信息获取的有效性，针对性的需求也就自然出现了。面对信息过载，推荐系统应运而生。

其实我们每天都在无意识的接触推荐系统。包括手机端的APP和各大类购物网站等等。比如我们可以通过搜索引擎搜索到“今年有那些好电影？”的需求。



我们针对信息过载的问题，寻求了很多的思路，最初是的就是上述提到的搜索引擎和分类导航页，但是搜索引擎能做的是**共性的推荐**，有时候不愿意去想一些搜索词，希望系统自动挖掘自己的兴趣点，更希望各大网站和系统给我们surprise。

这就不是我们今天讲的个性化推荐，现在我们需要根据每个用户自己的行为去做个性化的推荐。如今的今日头条，各大电商平台，豆瓣都已经做了很多推荐的事情。

## 1.4推荐系统用途和使用场景

我们会根据每个用户自己的特征，如网站给用户打了那些标签，你在网站上做了那些评论，你浏览过什么网页，看过那些视频等，系统收集到用户的这些行为数据，我们通过推荐算法帮我们推荐很可能感兴趣的文章、视频、音频等内容。



**推荐系统利用用户数据来发现规律：**

促进厂商商品销售，帮助用户找到想要的商品

推荐系统无处不在，体现在生活的各个方面

图书推荐；QQ好友推荐；优酷，爱奇艺的视频推荐；豆瓣的音乐推荐；大从点评的餐饮推荐；世纪佳缘的相亲推荐；智联招聘的职业推荐。

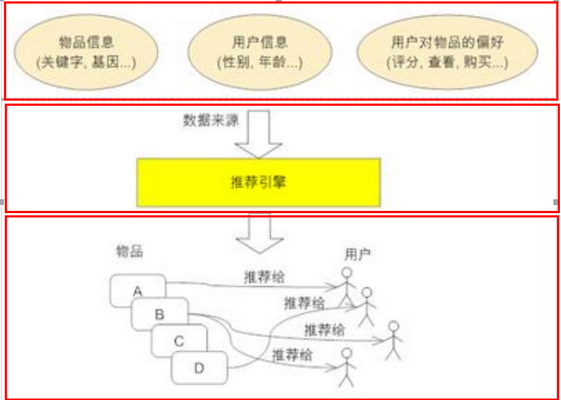
亚马逊的推荐系统深入到网站的各类商品，为亚马逊带来了至少30%的销售额。

## 1.5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。

**下面一起看推荐系统的基本原理(课堂演示)：**



## 1.6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。

# 推荐方法简述

\* 协同过滤推荐算法

\* 基于内容的推荐算法

\* 基于知识的推荐算法

\* 混合推荐算法

\* 流行度推荐算法

每一种推荐算法都有其优点和缺点，当然也有其限制条件，在做出决定选择算法之前必须一一考量，**在实践中，可能需要测试几种算法，以发现哪一种最适合你的用户**，测试中你也会直观地发现他们是什么以及他们的工作原理。

## 2.1 基于内容的推荐系统



基于内容推荐是推荐系统中比较常见的一种做法，这种方法对于每个item基于其自身属性，抽取一些特征用来表示这个item的内容，从而推荐那些和当前item含有相同或相近特征的一些item。

 这种推荐系统多用于一些资讯类的应用上，针对文章本身抽取一些tag作为该文章的关键词，继而可以通过这些tag来评价两篇文章的相似度。抽取tag经常采用的方案是基于TF-IDF得到的一系列权值较高的term，即认为：在一篇文章中，那些出现频率高的（除停用词）词，并且在其他文章中出现频率较低的词更能代表这篇文章的内容。经过TF-IDF公式计算，权重大于某一阈值的term作为该文章的tag。通常，也考虑tag的词性，一般来说名词和形容词更加适合作为tag。

 这种推荐系统的优点在于易于实现，不需要用户数据因此不存在稀疏性和冷启动问题。基于item本身特征推荐，因此不存在过度推荐热门的问题。所涉及的技术都是搜索引擎中应用比较成熟的技术。

缺点在于抽取的特征既要保证准确性又要具有一定的实际意义，否则很难保证推荐结果的相关性。豆瓣网采用人工维护tag的策略，依靠用户去维护item的tag的准确性。

**基于内容其他推荐方式**

（1）打标签：运动商品类，快速消费品类。粒度划分越细，推荐结果就越精确。

（2）商品名称与关键字：通过从商品的描述信息中提取关键字，利用关键字的相似来作推荐。

## 2.2 基于关联规则的推荐系统

基于关联规则的推荐更常见于电子商务系统中，并且也被证明行之有效。其实际的意义为购买了一些物品的用户更倾向于购买另一些物品。基于关联规则的推荐系统的首要目标是挖掘出关联规则，也就是那些同时被很多用户购买的物品集合，这些集合内的物品可以相互进行推荐。

目前关联规则挖掘算法主要从Apriori和FP-Growth两个算法发展演变而来。Apriori算法思路实现简单，通过迭代不断通过K-1元频繁项目集生成K元频繁项目集，直到不能生成为止，最终可以得到最大频繁项目集。Apriori算法存在的问题是每次迭代都要判断生成K元集合的K-1元子集是否都是频繁项目集，计算量巨大；并且Apriori算法是一个挖掘最大频繁项目集的算法，无法得到全部频繁模式集合。FP-Growth算法通过构建FP-Tree（频繁模式树），去发现频繁模式集。相比于Apriori算法，计算量较少，并且可以得出几乎所有的频繁项目集；但该算法实现起来要比Apriori复杂，并且FP-Growth需要将全部数据库事务加载到内存中，而Apriori虽然需要反复读取数据库事务，但是不需要全部载入内存。一般而言，FP-Growth要比Apriori快至少一个数量级。

关联规则挖掘中有两个主要的概念，支持度和置信度。支持度是指包含某频繁项目集的事务数和总数据库事务数的百分比。置信度指包含某频繁项目集的事务数和包含被推荐项目集的事务数的百分比。而最小支持度阈值和最小置信度阈值也是决定一个事务集合是否是频繁事务集和一个关联规则是否成立的决定因素。因此这两个阈值也决定了推荐系统的准确率和召回率。

基于关联规则的推荐系统一般转化率较高，因为当用户已经购买了频繁集合中的若干项目后，购买该频繁集合中其他项目的可能性更高。缺点在于计算量较大，但是可以离线计算，因此影响不大。同时基于关联规则的推荐系统由于采用用户数据，不可避免的存在冷启动和稀疏性问题。并且存在热门项目容易被过度推荐的问题。此外，基于item的推荐体系多采用1toN的推荐模式，因此实际的关联规则相当于从二元频繁项目集产生（即1to1的模式），1->1这种关联规则的相关性要远低于M->N这种最初关联规则的推荐形式。因此基于关联规则的推荐系统很少被应用于item的推荐体系。

## 2.3 基于协同过滤的推荐系统

协同过滤是一种在推荐系统中广泛采用的推荐方法。这种算法基于一个假设，喜欢相同item的用户更有可能具有相同的兴趣。基于协同过滤的推荐系统一般应用于有用户评分的系统之中，通过分数去刻画用户对于item的好恶。协同过滤被视为利用集体智慧的典范，不需要对项目进行特殊处理，而是通过用户建立项目与项目之间的联系。

经过发展演变，协同过滤逐渐分化为两种类型：基于用户(User-based)的协同过滤系统和基于项目(Item-based)的协同过滤系统。

### 2.3.1 User-based协同过滤系统



采用这种方法的推荐系统通过比较当前用户和其他用户阅读的项目进行比较（其评分为权值），从而选择当前用户的TopN邻近用户，然后根据当前用户的临近用户，对当前用户的未评分项进行模拟评分，然后再将评分高的项目推荐给用户。

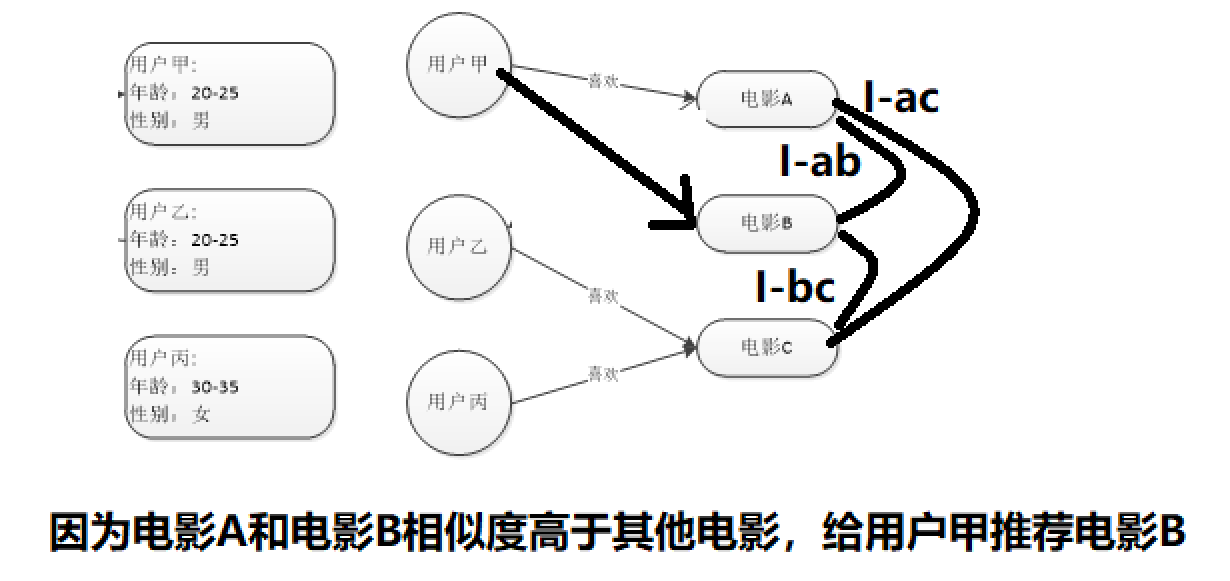
这种推荐系统的优点在于推荐项目之间在内容上可能完全不相关，因此可以发现用户的潜在兴趣，并且针对每个用户生成其个性化的推荐结果。缺点在于一般的Web系统中，用户的增长速度都远远大于项目的增长速度，因此其计算量的增长巨大，系统性能容易成为瓶颈。因此在业界中单纯的使用User-based协同过滤系统较少。

豆瓣网的“豆瓣猜”是一种个性化的推荐，因此推荐其背后可能采用了User-based协同过滤技术。但是可以看出，豆瓣猜页面上仍主要采用了基于tag的推荐技术，并且利用“友邻”代替了协同过滤技术中的临近用户计算，并且“豆瓣猜”也并非豆瓣网的主打推荐系统。

### 2.3.2 Item-based协同过滤系统

这种协同过滤和User-based协同过滤相似，只不过是通过比较对当前item的用户评分和其他item的用户评分来选择当前item的相似item进行推荐。Item-based协同过滤可以看作是关联规则推荐的一种退化（尤其是进行1to1形式的推荐时），但由于协同过滤更多考虑了用户的实际评分，并且只是计算相似度而非寻找频繁集，因此可以认为Item-based协同过滤准确率较高并且覆盖率更高。

同User-based相比Item-based应用更为广泛，扩展性和算法性能更好。由于项目的增长速度一般较为平缓，因此性能变化不大。缺点就是无法提供个性化的推荐结果。



## 2.4 基于用户模型学习的推荐系统

**这种推荐系统根据*用户已阅读（或评分）项目*进行监督学习，从而得到该用户的行为模型，继而根据该用户的模型去对用户未评分的项目进行分类预测，从而得到用户可能喜欢的项目。**根据用户模型的学习算法，一般也需要抽取项目的一些特征。目前这种推荐系统的效果还没有生产环境应用的资料证实，因此尚无法判断其优缺点。

## 2.5 其他

 推荐系统和搜索引擎都是海量信息的过滤器，因此有人提出利用搜索引擎的一些成熟技术去解决推荐系统的问题。例如，将item看作是一个doc，将对该item有过评价的用户看作是该doc的一个term，然后利用BM-25、TF-IDF等算法对item的相似度进行计算，这种方案可以避免热门问题，同时搜索引擎技术的发展相对要比推荐系统成熟。存在的问题就是，同基于用户模型学习的推荐系统并未有生产系统使用效果的资料。

 除此之外还有基于效用和基于知识的推荐系统。

## 2.6 综述

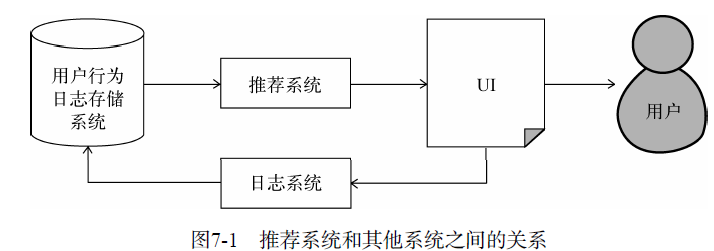
推荐系统的**未来发展方向不会是用单一方案解决问题**，很有可能是通过组合以上几种推荐方法，来确定最终的推荐结果。可以采用加权、变换、混合、特征组合等方法综合考虑这些推荐方法。

# 3.推荐系统项目实例

如何设计一个真实的推荐系统处理不同的数据，根据不同的数据设计算法，并将这些算法融合到一个系统当中是我们讨论的主要问题。我们将首先介绍推荐系统的外围架构，然后介绍推荐系统的架构，并对架构中每个模块的设计进行深入讨论。

每个网站都会有一个UI系统，UI系统负责给用户展示网页并和用户交互。网站会通过日志系统将用户在UI上的各种各样的行为记录到用户行为日志中。日志可能存储在内存缓存里，也可能存储在数据库中，也可能存储在文件系统中。而推荐系统通过分析用户的行为日志，给用户生成推荐列表，最终展示到网站的界面上。

各系统之间的关系：



从上面的结构可以看到，推荐系统要发挥强大的作用，除了推荐系统本身，主要还依赖于两个条件——界面展示和用户行为数据。

通过一定方式展示物品，主要包括物品的标题、缩略图和介绍等。

 很多推荐界面都提供了推荐理由，理由可以增加用户对推荐结果的信任度。

 推荐界面还需要提供一些按钮让用户对推荐结果进行反馈，这样才能让推荐算法不断改

善用户的个性化推荐体验。

在设计推荐界面时，我们可以综合考虑其他网站的设计并结合自己网站的特点。

## 3.1数据收集和存储

个性化推荐算法依赖于**用户行为数据**，而在任何一个网站中都存在着各种各样的用户行为数据。那么如何存取这些数据就是推荐系统需要解决的首要问题。

表7-1展示了一个假想的电子商务网站上的典型用户行为数据。如表所示，从产生行为的用户角度看，有些行为是只有注册用户才能产生的，而有些行为是所有用户都可以产生的。从规模上看，**浏览网页、搜索记录的规模都很大，因为这种行为所有用户都能产生**，而且平均每个用户都会产生很多这些行为。**购买、收藏行为规模中等，因为只有注册用户才能产生这种行为**，但购买行为又是电商网站的主要行为，所以它们相对于评论来说规模更大，但相对于网页浏览行为来说规模要小得多，**最后剩下的行为是注册用户里的一小部分人才有的，所以规模不会很大。**从实时存取的角度上看，**购买、收藏、评论、评分、分享等行为都是需要实时存取的，因为只要用户有了这些行为，界面上就需要体现出来，**比如用户购买了商品后，用户的个人购买列表中就应立即显示用户购买的商品。而有些行为，比如浏览网页的行为和搜索行为并不需要实时存取。



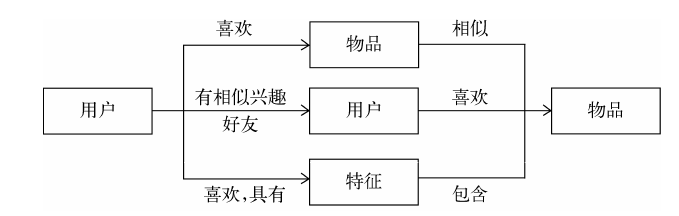
按照前面数据的规模和是否需要实时存取，不同的行为数据将被存储在不同的媒介中。一般来说，需要实时存取的数据存储在数据库和缓存中，而大规模的非实时地存取数据存储在分布式文件系统（如HDFS）中。

数据能否实时存取在推荐系统中非常重要，因为推荐系统的实时性主要依赖于能否实时拿到**用户的新行为**。只有快速拿到大量用户的新行为，推荐系统才能够实时地适应用户当前的需求，

给用户进行实时推荐。

## 3.2推荐系统架构

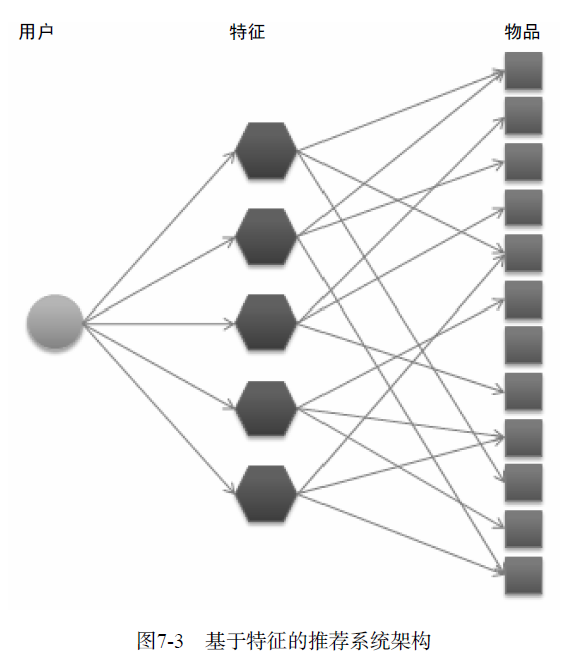
前面提到推荐系统是联系用户和物品的媒介，而推荐系统联系用户和物品的方式主要有3种，如果将这3种方式都抽象一下就可以发现，如果认为用户喜欢的物品也是一种用户特征，或者和用户兴趣相似的其他用户也是一种用户特征，那么用户就和物品通过特征相联系。



根据上面的抽象，可以设计**一种基于特征的推荐系统架构**。如图7-3所示，当用户到来之后，推荐系统需要为用户生成特征，然后对每个特征找到和特征相关的物品，从而最终生成用户的推荐列表。因而，推荐系统的核心任务就被拆解成两部分：

**1.一个是如何为给定用户生成特征**

**2.另一个是如何根据特征找到物品**。



用户的特征种类非常多，主要包括如下几类。

**人口统计学特征**：包括用户的年龄、性别、国籍和民族等用户在注册时提供的信息。

**用户的行为特征**：包括用户浏览过什么物品、收藏过什么物品、给什么物品打过什么样的分数等用户行为相关的特征。同时，用户行为从时间上也可以分为用户近期的行为和长期的行为。

**用户的主题特征**：可以根据用户的历史行为利用**主**题模型（topic model）将电视剧和电

影聚合成不同的**主**题，并且计算出每个用户对什么**主**题感兴趣。比如用户如果看了《叶问》、《新龙门客栈》和《醉拳》，那么可以认为用户对“香港武侠电影”这个话题感兴趣。

推荐系统的推荐任务也有很多种，如下所示。

 **将最新加入的物品推荐给用户**。

 **将商业上需要宣传的物品推荐给用户。**

 **给用户推荐不同种类的物品**，比如亚马逊会推荐图书、音像、电子产品和服装等

 **给用户混合推荐**，有时需要将图书和音像制品放到一个推荐列表中展示给用户（如图1

所示）。

 **对于不同的产品推荐不同新颖度的物品**。比如在首页给用户展示比较热门的推荐结果，

在推荐系统页面给用户展示长尾中的物品（如图2所示）。

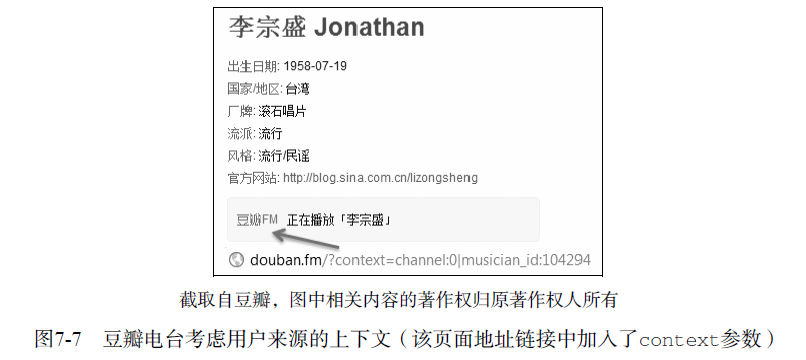
 **考虑到用户访问推荐系统的上下文，**比如当你在豆瓣音乐找到“李宗盛”时，右侧会有

一个链接告诉你可以在豆瓣电台收听“李宗盛”。单击了这个链接后，豆瓣电台给你推荐

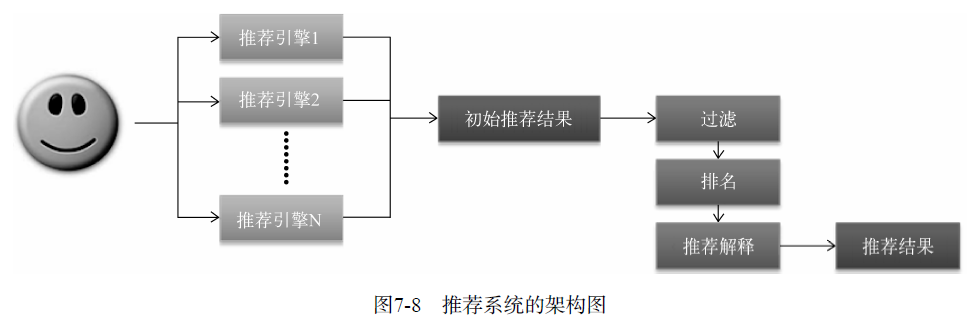
的音乐就考虑了“李宗盛”这个**上下文**。







如果要在一个系统中把上面提到的各种特征和任务都统筹考虑，那么系统将会非常复杂，而且很难通过配置文件方便地配置不同特征和任务的权重。因此，推荐系统需要由多个推荐引擎组成，**每个推荐引擎负责一类特征和一种任务，而推荐系统的任务只是将推荐引擎的结果按照一定权重或者优先级合并、排序然后返回**（如图7-8所示）。



这样做还有两个好处。

 可以**方便地增加/删除引擎，控制不同引擎对推荐结果的影响**。对于绝大多数需求，只需要通过不同的引擎组合实现。

 可以**实现推荐引擎级别的用户反馈**。每一个推荐引擎其实代表了一种推荐策略，而不同

的用户可能喜欢不同的推荐策略。有些用户可能喜欢利用他的年龄性别作出的推荐，有

些用户可能比较喜欢看到新加入的和他兴趣相关的视频，有些用户喜欢比较新颖的推荐，

有些用户喜欢专注于一个邻域的推荐，有些用户喜欢多样的推荐。**我们可以将每一种策**

**略都设计成一个推荐引擎，然后通过分析用户对推荐结果的反馈了解用户比较喜欢哪些**

**引擎推荐出来的结果，从而对不同的用户给出不同的引擎组合权重。**

将推荐系统拆分成不同推荐引擎后，如何设计一个推荐引擎变成了推荐系统设计的核心部

分。接下来将讨论推荐引擎的设计方法。

## 3.3推荐引擎构建

推荐引擎使用一种或几种用户特征，按照一种推荐策略生成一种类型物品的推荐列表。图7-9展示了每个具体推荐引擎的架构。

如图7-9所示，**推荐引擎架构**主要包括3部分。

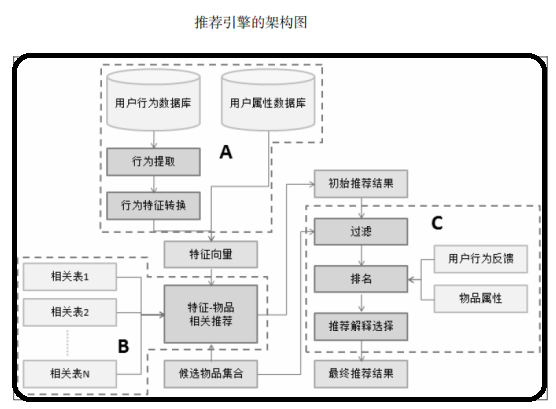
 该部分**负责从数据库或者缓存中拿到用户行为数据**，通过分析不同行为，生成当前用户

的特征向量。不过如果是使用非行为特征，就不需要使用行为提取和分析模块了。该模

块的输出是用户特征向量。

 该部分**负责将用户的特征向量通过特征-物品相关矩阵转化为初始推荐物品列表**。

 该部分**负责对初始的推荐列表进行过滤、排名等处理，从而生成最终的推荐结果。**



### 3.3.1生成用户特征向量

一般来说，用户的特征包括两种，**一种是用户的注册信息中可以提取出来的**，主要包括用户的人口统计学特征。对于使用这种特征的推荐引擎，如果内存够，可以将存储这些特征的信息直接缓存在内存中，在推荐时直接拿到用户的特征数据并生成特征向量。除了这种特征，**另一种特征主要是从用户的行为中计算出来的**，本节着重讨论如何生成特征。

一个特征向量由特征以及特征的权重组成，在利用用户行为计算特征向量时需要考虑以下

因素。

* **用户行为的种类：**在一个网站中，用户可以对物品产生很多不同种类的行为。
* 用户可以**浏览物品、单击物品的链接、收藏物品、给物品打分、购买物品、评论物品**、给物品打上不同的标签、和好友分享物品、搜索不同的关键词等。
* 这些行为都会**对物品特征的权重产生影响**，但不同行为的影响不同，大多时候很难确定什么行为更加重要，一般的标准就是用户付出代价越大的行为权重越高。比如，购买物品需要用户掏钱，所以用户一定会三思而后行，因此购买行为最为重要。相反，浏览物品的网页代价很小，所以这种行为对反映用户的真实兴趣的影响很小。
* **用户行为产生的时间：**一般来说，用户近期的行为比较重要，而用户很久之前的行为相

对比较次要。因此，如果用户最近购买过某一个物品，那么这个物品对应的特征将会具

有比较高的权重。

* **用户行为的次数：**有时用户对一个物品会产生很多次行为。比如用户会听一首歌很多次，

看一部电视剧的很多集等。因此用户对同一个物品的同一种行为发生的次数也反映了用

户对物品的兴趣，行为次数多的物品对应的特征权重越高。

* **物品的热门程度：**如果用户对一个很热门的物品产生了行为，往往不能代表用户的个性，

因为用户可能是在跟风，可能对该物品并没有太大兴趣，特别是在用户对一个热门物品

产生了偶尔几次不重要的行为（比如浏览行为）时，就更说明用户对这个物品可能没有

什么兴趣，可能只是因为这个物品的链接到处都是，很容易点到而已。

* 反之，如果用户对一个不热门的物品产生了行为，就说明了用户的个性需求。因此，推荐引擎在生成用户特征时会加重不热门物品对应的特征的权重。

### 3.3.2物品和物品相关推荐

在得到用户的特征向量后，我们可以根据离线的相关表得到初始的物品推荐列表。离线相关表可以存储在MySQL中，其存储格式如表7-2所示。



对于每个特征，我们可以在相关表中存储和它最相关的*N*个物品的ID。

从上面的架构图可以看到，**特征—物品相关推荐模块**还可以接受一个**候选物品集合**。

**候选物品集合的目的是保证推荐结果只包含候选物品集合中的物品。**它的应用场合一般是产品需求希望将某些类型的商品推荐给用户。比如有些产品要求给用户推荐最近一周加入的新物品，那么候选物品集合就包括最近一周新加的物品。

一般来说，如**果需要在一个小的候选物品集合中给用户推荐物品**，那么可以考虑上述方法。

但如果是要在一个很大的候选物品集合中给用户推荐物品，**那么可以考虑直接在初始推荐列表中过滤掉不在候选物品集合中物品的方法。**特征—物品相关推荐模块除了给用户返回物品推荐列表，还需要给推荐列表中的每个推荐结果产生一个解释列表，表明这个物品是因为哪些特征推荐出来的。

### 3.3.3过滤模块

在得到初步的推荐列表后，还不能把这个列表展现给用户，首先需要按照产品需求对结果进行过滤，过滤掉那些不符合要求的物品。一般来说，过滤模块会过滤掉以下物品。

 用户已经产生过行为物品 因为推荐系统的目的是帮助用户发现物品，因此没必要给用

户推荐他已经知道的物品，这样可以保证推荐结果的新颖性。

 候选物品以外的物品 候选物品集合一般有两个来源，**一个是产品需求**。比如在首页可

能要求将新加入的物品推荐给用户，因此需要在过滤模块中过滤掉不满足这一条件的物品。另一**个来源是用户自己的选择**，比如用户选择了某一个价格区间，只希望看到这个价格区间内的物品，那么过滤模块需要过滤掉不满足用户需求的物品。

 某些质量很差的物品 为了提高用户的体验，推荐系统需要给用户推荐质量好的物品，

那么对于一些绝大多数用户评论都很差的物品，推荐系统需要过滤掉。这种过滤一般以

用户的历史评分为依据，比如过滤掉平均分在2分以下的物品。

### 3.3.4排名模块

经过过滤后的推荐结果直接展示给用户一般也没有问题，但**如果对它们进行一些排名，则可以更好地提升用户满意度**，一般排名模块需要包括很多不同的子模块，下面将对不同的模块分别加以介绍。

1. **新颖性排名**

新颖性排名模块的**目的是给用户尽量推荐他们不知道的、长尾中的物品**。虽然前面的过滤模块**已经过滤掉了用户曾经有过行为的物品，保证了一定程度的新颖性**，但是用户在当前网站对某个物品没有行为并不代表用户不知道这个物品，比如用户可能已经在别的途径知道这个物品了。

1. **多样性排名**

多样性也是推荐系统的重要指标之一。增加多样性可以让推荐结果覆盖尽可能多的用户兴

趣。当然，这里需要指出的是提高多样性并不是时时刻刻都很好。比如在个性化网络电台中，因为用户某一固定时刻的兴趣是固定的，所以不希望听到不同曲风的歌曲，尽管这些曲风可能都是

用户之前表示喜欢的。不过，我们需要讨论如果要提高多样性，应该怎么提高。

**第一种提高多样性的方法是将推荐结果按照某种物品的内容属性分成几类，然后在每个类中都选择该类中排名最高的物品组合成最终的推荐列表。**比如，如果是电影，可以按照电影的类别（爱情片、动作片、科幻片等）对推荐结果中的电影分类，然后每种类别都选出几部电影组成最终的推荐结果。

这种方法的好处是比较简单直观，但这种方法也有严重的缺点。首先，选择什么样的内容属性进行分类对结果的影响很大。其次，就算选择了某种类别，但物品是否属于某个类别是编辑确定的，并不一定能够得到用户的公认。比如成龙的电影，有人认为是功夫片，有人认为是喜剧片，不同人看法不一。

因此，**第二种提高推荐结果多样性的方法是控制不同推荐结果的推荐理由出现的次数。**如果推荐系统对于每个推荐出来的物品都有一个推荐理由，这个推荐理由一般是产生推荐结果的重要特征。那么，要提高推荐结果的多样性，就需要让推荐结果尽量来自不同的特征，具有不同的推荐理由，而不是所有的推荐结果都对应一个理由。

1. **时间多样性**

时间多样性主要是为了**保证用户不要每天来推荐系统都看到同样的推荐结果。**大家已经熟知，提高推荐系统的时间多样性要从两个地方着手。

首先要**保证推荐系统的实时性**，在用户有新行为时实时调整推荐结果以满足用户最近的需求。这一点，这里的推荐系统设计中已经考虑到了。如果用户有实时行为发生，那么行为提取和分析模块就能实时拿到行为数据并转化为新的特征，然后经过**特征-物品相关模块转换成和新特征最相关的物品**，因而推荐列表中就立即反应了用户最新行为的影响。

提高推荐结果多样性的第二个方面是要在用户没有新的行为时，也要保证推荐结果每天都有变化。要实现这一点，只能通过如下方式。

 记录用户每次登陆推荐系统看到的推荐结果。

 将这些结果发回日志系统。这种数据不需要实时存储，只要能保证小于一天的延时就足

够了。

 在用户登录时拿到用户昨天及之前看过的推荐结果列表，从当前推荐结果中将用户已经

看到的推荐结果降权。

1. **用户反馈---CTR预估**

**排名模块**最重要的部分就是**用户反馈模块**。用户反馈模块主要通过分析用户之前和推荐结果的交互日志，预测用户会对什么样的推荐结果比较感兴趣。

如果推荐系统的目标是**提高用户对推荐结果的点击率，那么可以利用点击模型（click model）预测用户是否会点击推荐结果。**

点击模型在很多领域得到了广泛应用，比如搜索结果的点击预测

1. 搜索广告的点击预测
2. 上下文广告的点击预测

点击预测的主要问题是**预测用户看到某个推荐结果时是否会点击**。那么要进行点击率预测，首先需要提取特征。在推荐系统的点击率预测中可以用如下特征预测用户u会不会点击物品i：

* 用户u相关的特征，比如年龄、性别、活跃程度、之前有没有点击行为；
* 物品i相关的特征，比如流行度，平均分，内容属性；
* 物品i在推荐列表中的位置。用户的点击和用户界面的设计有很高的相关性，因此物品i在推荐列表中的位置对预测用户是否点击很重要；
* 用户之前是否点击过和推荐物品i具有同样推荐解释的其他推荐结果；
* 用户之前是否点击过和推荐物品i来自同样推荐引擎的其他推荐结果。

**点击模型需要离线计算好，在线将模型加载到内存中。**为了提高在线预测的效率，一般只可以使用线性模型。

# 4.推荐经验之谈

提出了10条在设计推荐系统中学习到的经验和教训。

(1) **确定你真的需要推荐系统。**推荐系统只有在用户遇到信息过载时才必要。如果你的网站物品不太多，或者用户兴趣都比较单一，那么也许并不需要推荐系统。所以不要纠结于推荐系统这个词，不要为了做推荐系统而做推荐系统，而是应该从用户的角度出发，设计出能够真正帮助用户发现内容的系统，无论这个系统算法是否复杂，只要能够真正帮助用户，就是一个好的系统。

(2) **确定商业目标和用户满意度之间的关系。**对用户好的推荐系统不代表商业上有用的推荐系统，因此要首先确定用户满意的推荐系统和商业上需求的差距。一般来说，有些时候用户满意和商业需求并不吻合。但是一般情况下，用户满意度总是符合企业的长期利益，因此这一条的主要观点是要平衡企业的长期利益和短期利益之间的关系。

(3) **选择合适的开发人员。**一般来说，如果是一家大公司，应该雇用自己的开发人员来专门进行推荐系统的开发。

(4) **忘记冷启动的问题。**不断地创新，互联网上有任何你想要的数据。只要用户喜欢你的产品，他们就会不断贡献新的数据。

(5) **平衡数据和算法之间的关系。**使用正确的用户数据对推荐系统至关重要。对用户行为数据的深刻理解是设计好推荐系统的必要条件，因此分析数据是设计系统中最重要的部分。**数据分析决定了如何设计模型，而算法只是决定了最终如何优化模型。**

(6) **找到相关的物品很容易，**但是何时以何种方式将它们展现给用户是很困难的。不要为了推荐而推荐。

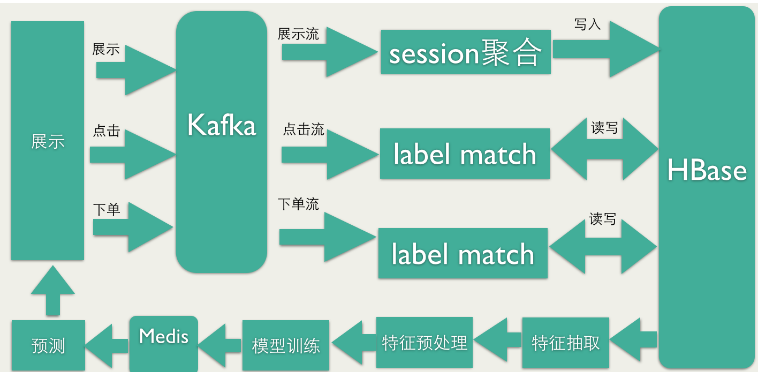
(7) 不要浪费时间计算相似兴趣的用户，可以**直接利用社会网络数据**。

(8) 需要不断地提升算法的扩展性。

(9) 选择合适的**用户反馈方式ctr。**

(10) 设计合理的评测系统，时刻关注推荐系统各方面的性能。

# 5.美团技术架构



线上的展示日志，点击日志和下单日志会写入不同的Kafka流。读取Kafka流，以HBase为中间缓存，完成label match（下单和点击对映到相应的展示日志），在做label match的过成中，会对把同一个session的日志放在一起，方便后面做skip above：

一些特征信息：



# 6.数据集集锦

<https://grouplens.org/datasets/movielens/>

MovieLens数据集是一个关于电影评分的数据集，里面包含了从IMDB, The Movie DataBase上面得到的用户对电影的评分信息，详细介绍。

## 6.1links.csv:

文件里面的内容是帮助你如何通过网站id在对应网站上找到对应的电影链接的。数据格式如下：

movieId, imdbId, tmdbId

* + movieId:表示这部电影在movielens上的id，可以通过链接https://movielens.org/movies/(movieId)来得到。
  + imdbId:表示这部电影在imdb上的id，可以通过链接http://www.imdb.com/title/(imdbId)/
  + 来得到。
  + tmdbId:表示这部电影在themoviedb上的id，可以通过链接http://www.imdb.com/title/(tmdbId)/ 来得到。

## 6.2movies.csv:

movieId, title, genres

文件里包含了一部电影的id和标题，以及该电影的类别。数据格式如下：

movieId, title, genres

* movieId:每部电影的id
* title:电影的标题
* genres:电影的类别（详细分类见readme.txt）

## 6.3ratings.csv:

文件里面的内容包含了每一个用户对于每一部电影的评分。数据格式如下：

userId, movieId, rating, timestamp

* + userId: 每个用户的id
  + movieId: 每部电影的id
  + rating: 用户评分，是5星制，按半颗星的规模递增(0.5 stars - 5 stars)
  + timestamp: 自1970年1月1日零点后到用户提交评价的时间的秒数
  + 数据排序的顺序按照userId，movieId排列的。

## 6.4tags.csv

文件里面的内容包含了每一个用户对于每一个电影的分类。数据格式如下：

userId, movieId, tag, timestamp

* userId: 每个用户的id
* movieId: 每部电影的id
* tag: 用户对电影的标签化评价
* timestamp: 自1970年1月1日零点后到用户提交评价的时间的秒数
* 数据排序的顺序按照userId，movieId排列的。

下载链接：

官网地址: https://grouplens.org/datasets/movielens/

ml-latest-small(1MB): http://files.grouplens.org/datasets/movielens/ml-latest-small.zip

ml-latest(234.2MB): http://files.grouplens.org/datasets/movielens/ml-latest.zip

读取方法

利用Python的csv模块进行读取操作。

import csv

with open(file\_url, 'r') as f:

data = csv.reader(f)

for i in data:

print(i)