实时推荐系统的3种方式

简书

推荐系统介绍

自从1992年施乐的科学家为了解决信息负载的问题，第一次提出协同过滤算法，个性化推荐已经经过了二十几年的发展。1998年，林登和他的同事申请了“item-to-item”协同过滤技术的专利，经过多年的实践，亚马逊宣称销售的推荐占比可以占到整个销售GMV（Gross Merchandise Volume，即年度成交总额）的30%以上。随后Netflix举办的推荐算法优化竞赛，吸引了数万个团队参与角逐，期间有上百种的算法进行融合尝试，加快了推荐系统的发展，其中SVD（Sigular Value Decomposition，即奇异值分解，一种正交矩阵分解法）和Gavin Potter跨界的引入心理学的方法进行建模，在诸多算法中脱颖而出。其中，矩阵分解的核心是将一个非常稀疏的用户评分矩阵R分解为两个矩阵：User特性的矩阵P和Item特性的矩阵Q，用P和Q相乘的结果R’来拟合原来的评分矩阵R，使得矩阵R’在R的非零元素那些位置上的值尽量接近R中的元素，通过定义R和R’之间的距离，把矩阵分解转化成梯度下降等求解的局部最优解问题。Netflix最新的实时推荐系统如图9-5所示。

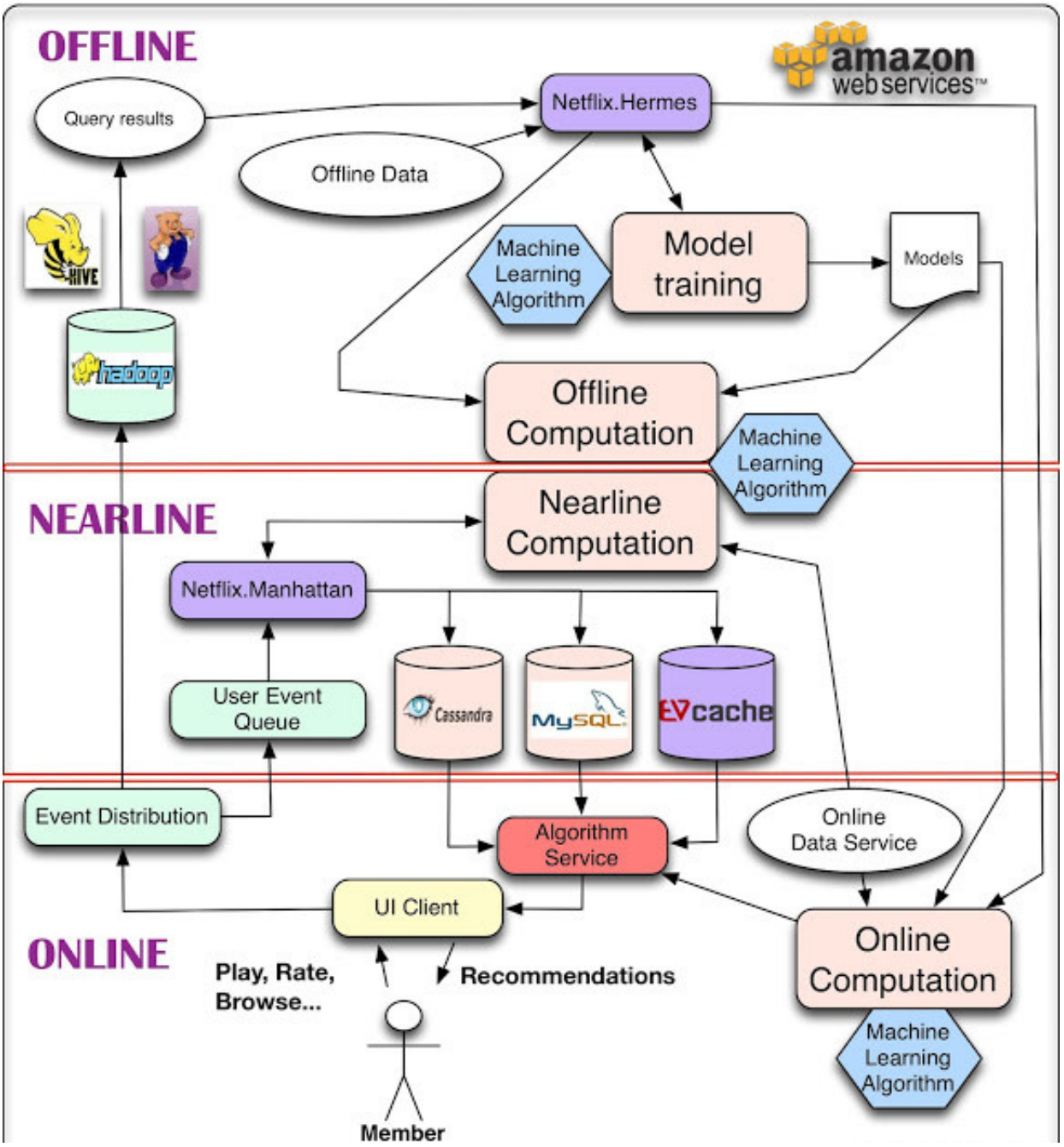


图9-5 NetFlix的实时推荐系统系统架构图（来源：http://techblog.netflix.com/2013/03/system-architectures-for.html）

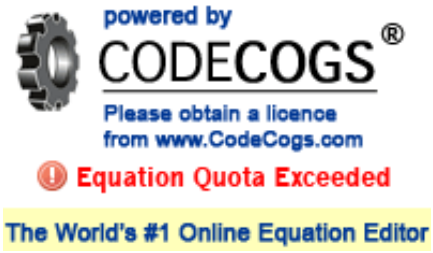
与此同时，Pandora、LinkedIn、Hulu、Last.fm等一些网站在个性化推荐领域都展开了不同程度的尝试，使得推荐系统在垂直领域有了不少突破性进展，但是在全品类的电商、综合的广告营销上，进展还是缓慢，仍然有很多的工作需要探索。特别是在全品类的电商中，单个模型在母婴品类的效果还比较好，但在其他品类就可能很差，很多时候需要根据品类、推荐栏位、场景等不同，设计不同的模型。同时由于用户、SKU不停地增加，需要定期对数据进行重新分析，对模型进行更新，但是定期对模型进行更新，无法保证推荐的实时性，一段时间后，由于模型训练也要相当时间，可能传统的批处理的Hadoop的方法，无法再缩短更新频率，最终推荐效果会因为实时性问题达到一个瓶颈。

推荐算法主要有基于人口统计学的推荐、基于内容的推荐、基于协同过滤的推荐等，而协同过滤算法又有基于邻域的方法（又称基于记忆的方法）、隐语义模型、基于图的随机游走算法等。基于内容的推荐解决了商品的冷启动问题，但是解决不了用户的冷启动问题，并且存在过拟合问题（往往在训练集上有比较好的表现，但在实际预测中效果大打折扣），对领域知识要求也比较高，通用性和移植性比较差，

换一个产品形态，往往需要重新构建一套，对于多媒体文件信息特征提取难度又比较大，往往只能通过人工标准信息。基于邻域的协同过滤算法，虽然也有冷启动问题和数据稀疏性等问题，但是没有领域知识要求，算法通用性好，增加推荐的新颖性，并且对行为丰富的商品，推荐准确度较高。基于模型的协同过滤算法在一定程度上解决了基于邻域的推荐算法面临的一些问题，在RMSE（Root Mean Squared Error，即均方根误差）等推荐评价指标上更优，但是通常算法复杂，计算开销大，所以目前基于邻域的协同过滤算法仍然是最为流行的推荐算法。

基于邻域的协同过滤主要分为User CF和Item CF，根据以下条件不同，各自又有不同的使用场景。

计算量大小不同。基于邻域的协同过滤的时间复杂度为



，其中n为用户数，m为产品数，应用SVD等降维方法可以降低算法复杂度，但是分解矩阵又会花费一定的时间。
数据稀疏性倾斜度不同。例如，User CF主要基于用户对共同项目的评分，如果用户远远多于物品，没有足够评分将导致两个用户很少有共同评分的项目，找最近邻用户非常的不准确，虽然通过基于BP神经网络、朴素贝叶斯分类、基于内容的预测等方法可以填充矩阵，但是都会不同程度地带来的计算时间。

对于用户数量远远大于产品，并且产品相对稳定的电商系统，计算产品相似度计算量小，适用Item CF，否则用户量大，并且如果用户购买频繁，计算用户相似度计算量很大，极端情况下，100个用户对应2个产品，一个要计算C1002次相似度，一个只要计算C22，即一次相似度；反之，对于更新频繁，物品数量海量的新闻、博客、微博等系统，User CF效果更好。

当然，虽然SVD在分解矩阵上花费了一定时间，同时降维也会导致用户-项目矩阵中的信息丢失，但是用户-项目矩阵降维后，运算复杂度大大降低，同时矩阵稀疏性问题得到了较好地解决，作为Netflix比赛中最终提升效果较好的两个方法之一，被众多网站采用。用户-项目矩阵中的信息丢失问题可以通过选取合适的保留维数k在一定程度上得到缓解。

在一个电商系统中，有商品、类目、品牌、团购、闪购、搜索、店铺、广告、促销活动、抵用券等诸多实体；有首页的大轮播、猜你喜欢栏位，详情页的看了还看、看了还买、推荐品牌等栏位，购物车页面的买了还买、凑单免邮等栏位。如何在不同的栏位融入不同的推荐算法给用户推荐相应的实体，构建出属于电商自己的场景引擎，实现全站精准化，让网站的GMV或者利润达到最高，是每一个电商需要思考的问题。在实际中，很多推荐算法不一定要求实时，实时推荐在哪些场景下能带给栏位更高的GMV转化率，也是需要一定时间摸索和试错的。

目前基于用户画像的推荐，主要用在基于内容的推荐，从最近的RecSys大会（ACM Recommender Systems）上来看，不少公司和研究者也在尝试基于用户画像做Context-Aware的推荐（情境感知，又称上下文感知）。利用用户的画像，结合时间、天气等上下文信息，给用户做一些更加精准化的推荐是一个不错的方向。

9.2.2 实时推荐系统的方法

目前的商用推荐系统，当用户数和商品数达到一定数目时，推荐算法都面临严重的可扩展性问题，推荐的实效性变得非常差，如何在算法和架构上提高推荐速度是很多公司不得不思考的问题。目前，在算法上主要通过引入聚类技术和改进实时协同过滤算法提高推荐速度；在架构上，目前实时推荐主要有基于Spark、Kiji框架和Storm的流式计算3种方法。

1．聚类技术和实时协同过滤算法

在算法上，一般采用EM（Expectation-Maximization）、K-means、吉布斯（Gibbs Sampling）、模糊聚类等聚类技术提高推荐速度。因为使用聚类技术可以大大缩小用户或项目的最近邻居搜索范围，从而提高推荐的实时性，如表9-1所示。

表9-1 聚类技术比较

算法	概　念	缺　点
EM	最大期望算法，估计用户或项目属于某一类的概率	每个用户或项目属于两个不同的用户分类或项目分类，EM算法就不再适用
K-means	主要思想是以空间中k个点为中心进行聚类，对最靠近它们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果	聚类数目k需要事先给定而且不同的应用中k值是不同的，难于选取。 另外初始聚类中心是随机选取的，对于同一组数据，可能因为初始聚类中心的不同而产生不同的聚类结果。 有些类型的数据，比如说全是1和0组成的一个二进制数组，如果要对这种二进制数组进行聚类，K-means不适合，因为如果采用欧式距离，很难定义和计算它们的聚类中心点，这时可以采用Jaccard相似度和ROCK等层次聚类算法
吉布斯采样	与 EM 算法类似，不同的是吉布斯采样方法基于贝叶斯模型，计算可以离线进行	算法复杂度较大，聚类过程比较耗时
模糊聚类	利用模糊等价关系将给定的对象分为一些等价类，并由此得到与关系对应的模糊相似矩阵，该模糊相似矩阵满足传递性.根据相似矩阵求出其传递关系的闭包，然后在传递关系的闭包上实现分类，计算可以离线进行	可能性划分的收敛速度慢，当数据离散程度大，即数据灰度大，预测精度越差，需要对历史数据的平滑处理

除此之外，实时协同过滤算法本身一直是人们研究的热点，早在2003年，Edward F. Harrington就第一次提出了基于感知器的实时协同过滤算法，但是这种方法需要所有用户的偏好，实用性较差；2010年，杨强等提出了实时进化的协同过滤算法，给予新得分更高的权重来增量更新User和Item的相似度；2011

年，UC Berkeley的Jacob Abernethy等人提出了OCF-SGD算法，我们知道传统的矩阵分解把用户评分矩阵R分解成多个矩阵，比如R≈P*Q，该方法提出当新来一个User到Item的得分，把更新R矩阵的问题转换成更新P和Q矩阵，从而达到实时协同过滤；近几年的RecSys大会上，实时协同过滤也是讨论的热点，OCF-SGD算法每次只考虑一个用户，忽略了用户之间的关系，Jialei Wang等人提出了基于多任务学习的实时协同过滤算法，把每一个用户当做一个任务，定义一个表示各个任务间相似性和交互程度的矩阵A，当新来一个User到Item的得分，通过矩阵A来更新其他用户的得分。

2．基于Spark的方式

在架构上，第一种是使用Spark把模型计算放在内存中，加快模型计算速度，Hadoop中作业的中间输出结果是放到硬盘的HDFS中，而Spark是直接保存在内存中，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的模型计算，如表9-2所示。

表9-2 MapReduce和Spark的Shuffle过程对比

	MapReduce	Spark
collect	在内存中构造了一块数据结构用于Map输出的缓冲	没有在内存中构造一块数据结构用于Map输出的缓冲，而是直接把输出写到磁盘文件
sort	Map输出的数据有排序	Map输出的数据没有排序
merge	对磁盘上的多个spill文件最后进行合并成一个输出文件	在Map端没有merge过程，在输出时直接是对应一个Reduce的数据写到一个文件中，这些文件同时存在并发写，最后不需要合并成一个
copy框架	Jetty	Netty或者直接socket流
对于本节点上的文件	仍然是通过网络框架拖取数据	不通过网络框架，对于在本节点上的Map输出文件采用本地读取的方式
copy过来的数据存放位置	先放在内存，内存放不下时写到磁盘	一种方式全部放在内存；另一种方式先放在内存
merge sort	最后会对磁盘文件和内存中的数据进行合并排序	对于采用另一种方式时也会有合并排序的过程

（来源：http://www.csdn.net/article/2014-05-19/2819831-TDW-Shuffle/2）

3．基于Kiji框架的方式

第二种是使用Kiji，它是一个用来构建大数据应用和实时推荐系统的开源框架，本质上是对HBase上层的一个封装，用Avro来承载对象化的数据，使得用户能更容易地用HBase管理结构化的数据，使得用户姓

名、地址等基础信息和点击、购买等动态信息都能存储到一行，在传统数据库中，往往需要建立多张表，在计算的时候要关联多张表，影响实时性。Kiji与HBase的映射关系如表9-3所示。

表9-3 Kiji到HBase的映射关系

项	Kiji	HBase
Entity相关	Entity	相同键的值都属于同一行
	EntityID	行键（row key）
Column相关	locality:family:key	Family:qualifier
	locality	Family
	Family:key	Qualifier
Schema相关	Table Layout	HBase上的KijiMetaTable，如kiji.default.meta
	Cell Schema	Avro Schema
	Cell Schema mapping	HBase上的Schema Table，如kiji.default.schema_hash、keji.default.schema_id

Kiji提供了一个KijiScoring模块，它可以定义数据的过期策略，如综合产品点击次数和上次的点击时间，设置数据的过期策略把数据刷新到KijiScoring服务器中，并且根据自己定义的规则，决定是否需要重新计算得分。如用户有上千万浏览记录，一次的行为不会影响多少总体得分，不需要重新计算，但如果用户仅有几次浏览记录，一次的行为，可能就要重新训练模型。Kiji也提供了一个Kiji模型库，使得改进的模型部署到生产环境时不用停掉应用程序，让开发者可以轻松更新其底层的模型。

4. 基于Storm的方式

最后一种基于 Storm 的实时推荐系统。在实时推荐上，算法本身不能设计的太复杂，并且很多网站的数据库是TB、PB级别，实时读写大表比较耗时。可以把算法分成离线部分和实时部分，利用Hadoop离线任务尽量把查询数据库比较多的、可以预先计算的模型先训练好，或者把计算的中间数据先计算好，比如，线性分类器的参数、聚类算法的群集位置或者协同过滤中条目的相似性矩阵，然后把少量更新的计算留给Storm实时计算，一般是具体的评分阶段。

基于Storm的实时推荐系统

基于本章前面的学习，我们可以设计图9-6所示的实时推荐系统。

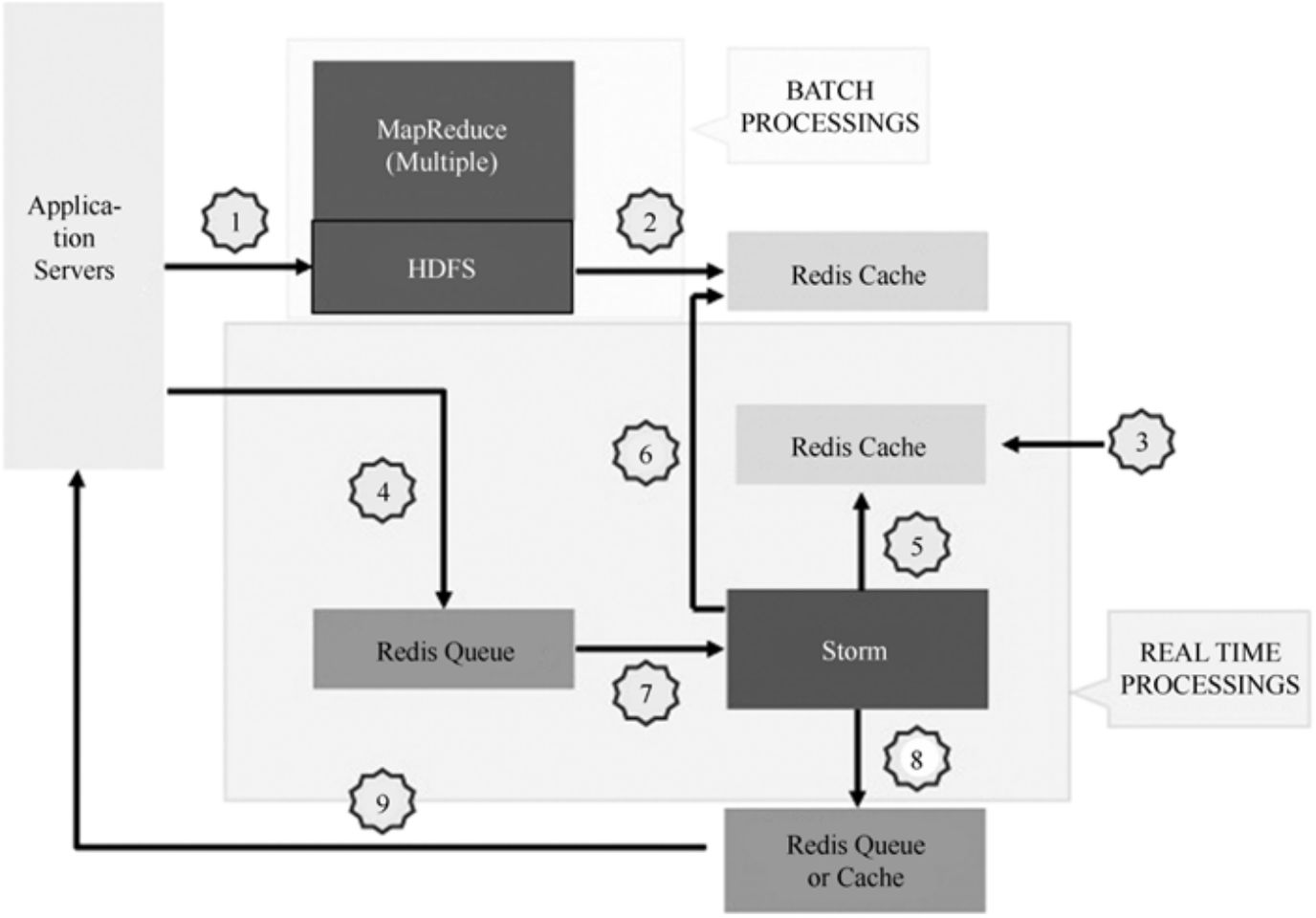


图9-6 实时推荐系统（图片来源PRANAB GHOSH，Big Data Cloud meetup。版权归原书作者所有）

用HBase或HDFS存储历史的浏览、购买行为信息，用Hadoop基于User CF的协同过滤，先把用户的相似度离线生成好，用户到商品的矩阵往往比较大，运算比较耗时，把耗时的运行先离线计算好，实时调用离线的结果进行轻量级的计算有助于提高产品的实时性。

我们来简单回顾一下协同过滤算法（如图9-7所示）：首先程序获取用户和产品的历史偏好，得到用户到产品的偏好矩阵，利用Jaccard相似系数（Jaccard coefficient）、向量空间余弦相似度（Cosine similarity）、皮尔逊相关系数（Pearson correlation coefficient）等相似度计算方法，得到相邻的用户（User CF）或相似商品（Item CF）。在User CF中，基于用户历史偏好的相似度得到邻居用户，将邻居用户偏好的产品推荐给该用户；在Item CF中，基于用户对物品的偏好向量得到相似产品，然后把这款产品推荐给喜欢相似产品的其他用户。

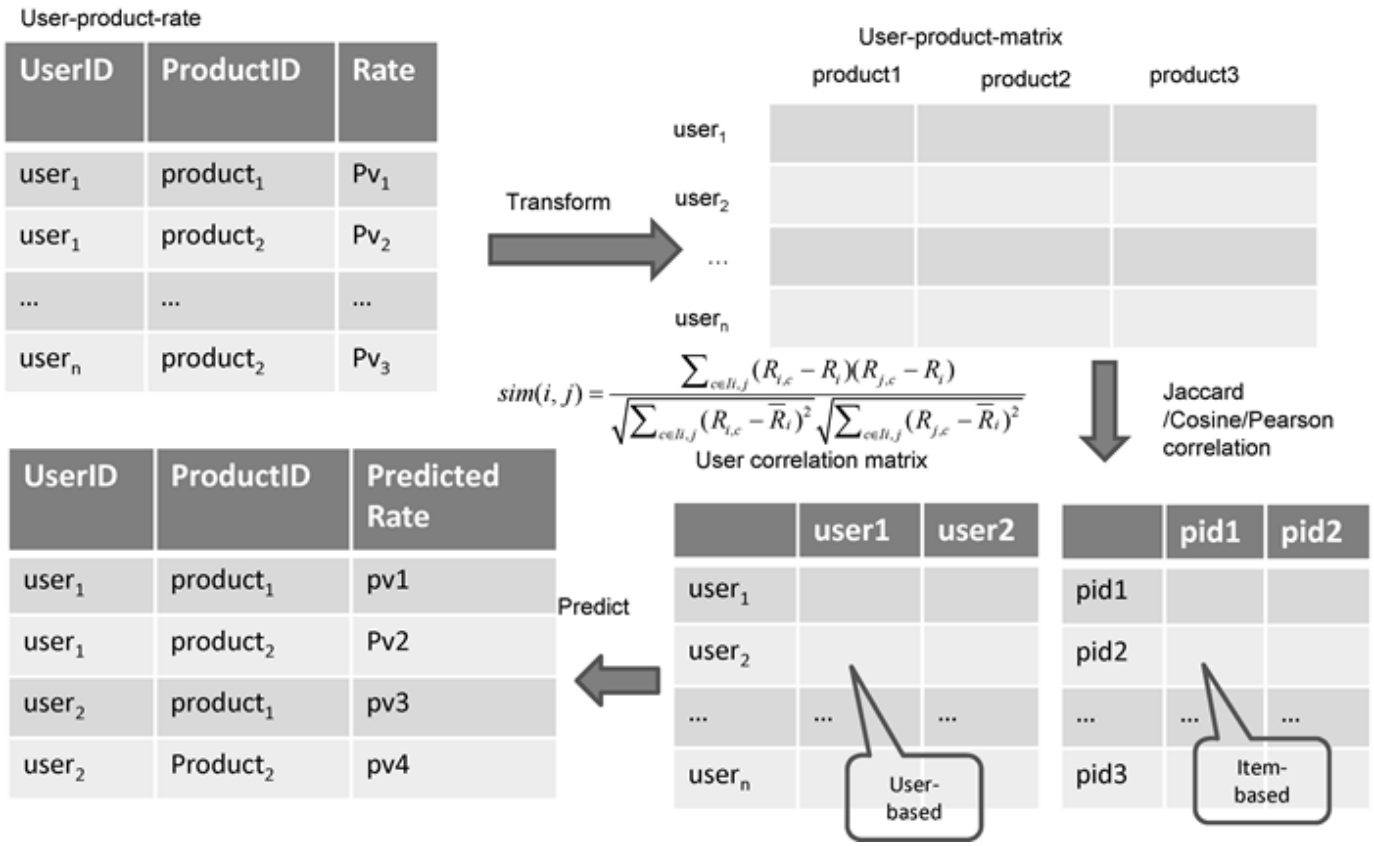


图9-7 协同过滤算法过程

然后通过Kafka或者Redis队列，保存前端的最新浏览等事件流，在Storm的Topology中实时读取里面的信息，同时获取缓存中用户topN个邻居用户，把邻居用户喜欢的商品存到缓存中，前端从缓存中取出商品，根据一定的策略，组装成推荐商品列表。

当然除了相似性矩阵，其他模型大体实现也相似，比如实际的全品类电商中不同的品类和栏位，往往要求不同的推荐算法，如母婴产品，如图9-8所示，如果结合商品之间的序列模式和母婴年龄段的序列模式，效果会比较好，可以把模型通过Hadoop预先生成好，然后通过Storm实时计算来预测用户会买哪些产品。

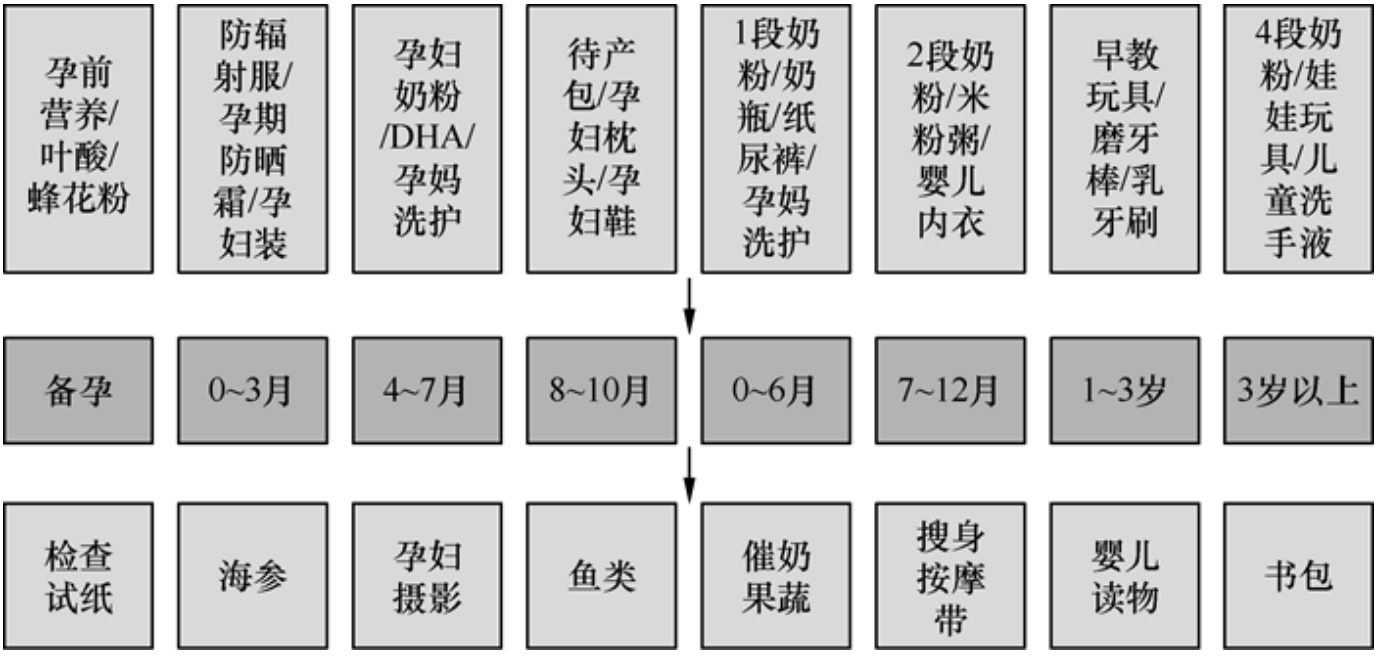


图9-8 序列模式在母婴类目推荐中的应用

本文摘自《Storm技术内幕与大数据实践》，点击阅读原文可以购买。

Storm

技术内幕与大数据实践

[redacted]

人民邮电出版社