它的三个核心组件HDFS、Yarn、MapReduce构成了大数据处理的三驾马车。HDFS提供了分布式文件存储，Yarn管理整个集群的资源和任务调度，MapReduce实现分布式的并行计算。

所以选择了开源日志库Zap来对用户产生的点击事件进行记录。Zap针对语言底层，实现了一个无反射，零分配的JSON编码器，减少了在序列化数据时造成的时间开销。

HDFS从设计之初就假设存储节点并不可靠，并设计了心跳检测、数据冗余备份等机制来帮助节点在发生故障时及时发现并恢复。 HDFS的主要组件包括一个namenode和其管理的多个datanode，以及Secondarynamenode检测节点。

Spark是一个面向内存的计算框架，这使得它的计算效率远远超过MR任务。而SparkSQL更是在此基础上抽象出DataFrame数据结构，并为其做了大量的性能优化，使得Python/Java等语言有了等同于原生scala编程的执行效率。

# 相关技术与理论基础

## 推荐算法理论研究

### 基于内容的推荐算法

基于内容的推荐算法结构简单、历史悠久，如今依旧在很多场景下表现良好。整个算法的核心思想就是为用户X推荐之前喜欢的物品相似的物品列表Y。下图2-1展示了基于内容推荐算法的Codeforces题目推荐流程：

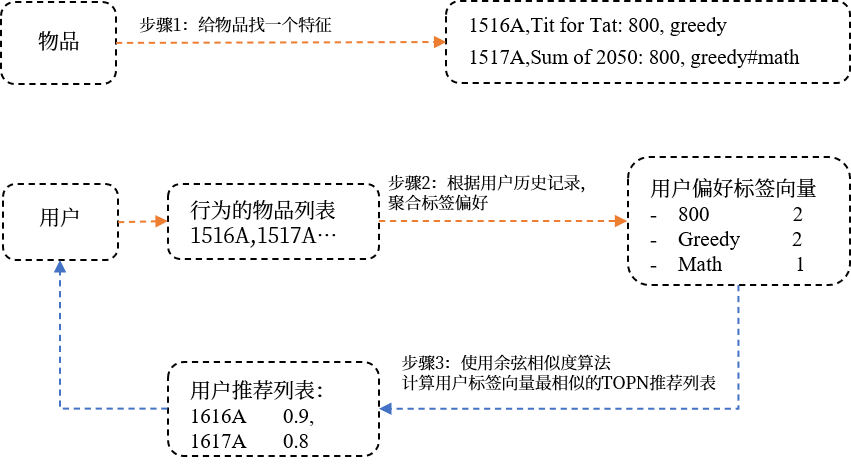


图 2‑1 基于内容推荐算法流程

**（1）步骤一**：为每个物品找一个特征，如难度系数、分类标签等。

**（2）步骤二**：收集用户的历史提交记录，比如用户A做过1516A、1517A这两道题，将题目列表标签做聚合统计，形成用户的偏好标签向量。

**（3）步骤三**：利用余弦相似度算法，计算与用户的偏好标签向量最接近的TopN列表（如上图1616A这道题和用户的偏好相似程度90%，非常可能在选择做），然后推荐给用户。

其优缺点分析如下表所示：

表 2-1基于内容推荐算法优缺点分析

|  |  |
| --- | --- |
| **优点** | **缺点** |
| 不需要其他用户的数据 | 一直处在舒适圈之中，不断推荐水题 |
| 不存在冷启动问题  推荐结果直观易于解释 | 推荐的同质化现象严重  无法与更多的特征交叉融合 |

### 常用向量相似度计算方法

在上文步骤三中提到了计算用户偏好向量和物品向量的相似程度，下面列举出几种常用的相似度计算方式，并对其含义进行说明。

（1）**余弦相似度**：如（式2-1）所示，衡量了向量**x1**和向量**x2**之间的向量夹角大小，夹角越小说明两个向量之间的相似程度越高。

（2）**皮尔逊相关系数**：如（式2-2）所示，用户对某件物品可能抱有极端的评分思想，给满分或者0分。皮尔逊相关系数在计算余弦相似度之前，先将两个向量减去自己的样本均值，来达到减少用户过激评分带来的影响。

（3）**Jaccard相关系数**：如（式2-3）所示，杰卡德相似系数从集合的角度衡量相似度，计算方法是看相同元素占总体的比例。而杰卡德距离（式2-4）与之恰恰相反，用不同元素占总体比例来衡量两个集合的区分度。

无论选取哪种相似度计算方式都需要具体问题具体分析，灵活的相似度算法使用可以帮助我们有效提升推荐的效果。

### 基于协同过滤的推荐算法

协同过滤（Collaborative Filtering）的推荐依据是使用行为数据，利用集体的智慧来进行推荐。UserCF与ItemCF的区别与联系如下图2-2所示：UserCF是根据找到和你兴趣接近的人，将他们喜欢的其他物品推荐给你，ItemCF是找到与你所喜欢物品类似的物品，然后推荐给你。左图中User1、User2、User4都喜欢Item1和Item2，那么认为这三个人的兴趣相同，当User2和User4新喜欢一个物品Item4时，便把此物品推荐给User1。右图中User2和4同时喜欢Item2和Item3，系统便认为Item2和3是相似物品。当User1喜欢Item2时，系统也会把Item3也推荐给他。

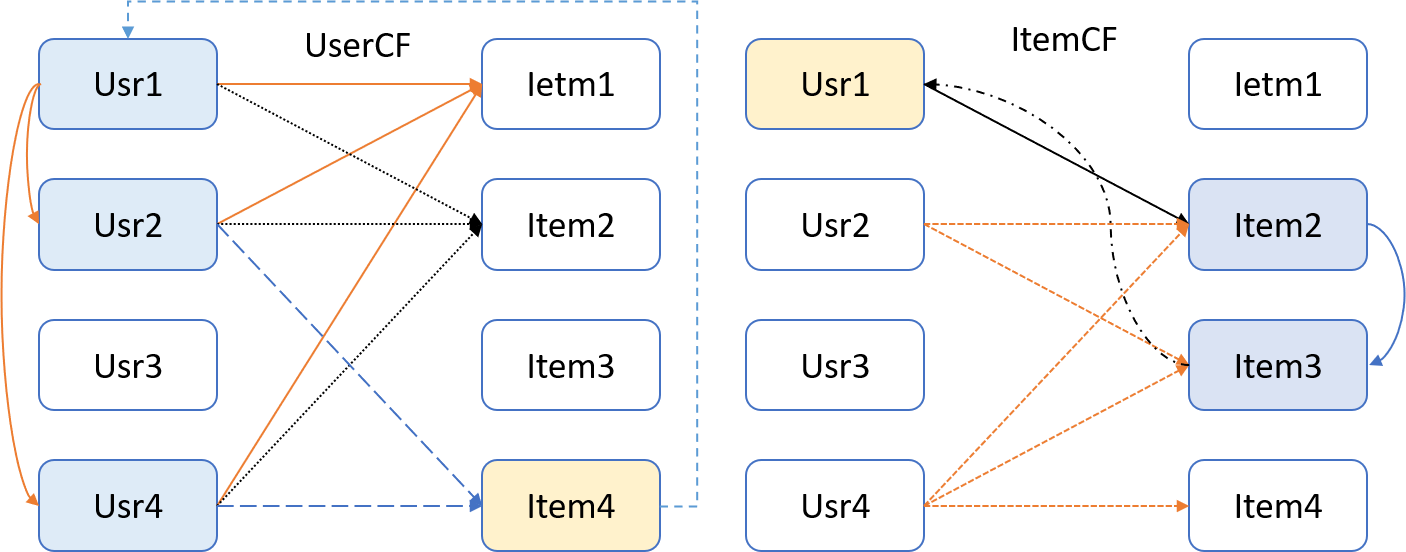


图 2‑2 UserCF与ItemCF原理

在获得了TopN个相似用户之后，我们就可以轻易的获得用户对于其还没有评价的物品的预估值。其实现方式通常为利用其他用户s对该物品p的评分的加权评分，权重w为其他用户s与推荐用户u的相似度，最终得到用户u对于物品p的预估评分。整个过程如（式2-5）所示，

协同过滤算法虽然经典，但是缺点也是显而易见的，比如上图2-2里的Uer3，当他还未对某个物品进行评分时就无法找到与他相似的用户，这就是推荐系统中常见的冷启动问题。还有在物品数量和用户数量特别多的时候，维护用户-物品评分矩阵对存储的压力极大，所以许多公司选择了ItemCF作为其推荐算法。

ItemCF的具体流程如下：

（1）获得用户的历史行为信息，组成M×N维的用户-物品矩阵

（2）以列为向量，通过皮尔逊相关系数计算物品之间的相似程度，最终构建出N×N维的物品相似度矩阵。

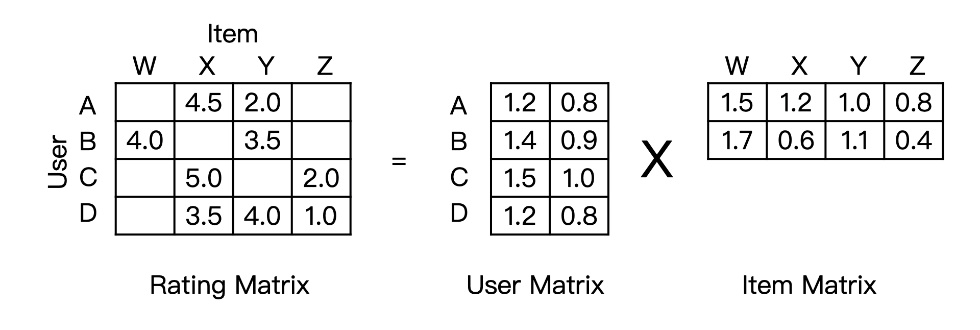
（3）获得用户的喜爱列表（正反馈可以靠与平均值相比来界定）

（4）针对目标用户u的喜爱列表，分别每个对已评价物品生成TopN个相似物品，组成相似物品候选集。

（5）对相似物品候选集里面的物品，对其按照加权后的总相似度进行排序，最终生成排序列表。加权公式如（式2-6）所示：

其中Ru,p是目标用户u对未评分物品p的预估情况，H是用户u的喜爱物品集合，权重wp,h是未评分物品p与已评分物品h的相似度，Ru,h是用户u对h的评分。

矩阵分解的主要过程，就是先分解协同过滤生成的共现矩阵，生成用户和物品的隐向量，再通过用户和物品隐向量间的相似性进行推荐。那这个过程的关键就在于如何分解这个共现矩阵了。从形式上看，矩阵分解的过程是直观的，就是把一个 mxn 的共现矩阵，分解成一个 mxk 的用户矩阵和 kxn 的物品矩阵相乘的形式（如图 3）。



是希望让物品向量和用户向量之积跟原始的评分之差的平方尽量小。简单来说就是，我们希望用户矩阵和物品矩阵的乘积尽量接近原来的共现矩阵。

这里，我先尝试着解释一下：一个物品能被向量表示，是因为这个向量跟其他物品向量之间的距离反映了这些物品的相似性。更进一步来说，两个向量间的距离向量甚至能够反映它们之间的关系。

首先，Embedding 是处理稀疏特征的利器。

大量使用 One-hot 编码会导致样本特征向量极度稀疏，而深度学习的结构特点又不利于稀疏特征向量的处理，因此几乎所有深度学习推荐模型都会由 Embedding 层负责将稀疏高维特征向量转换成稠密低维特征向量。

其次，Embedding 可以融合大量有价值信息，本身就是极其重要的特征向量 。 相比由原始信息直接处理得来的特征向量，Embedding 的表达能力更强，特别是 Graph Embedding 技术被提出后，Embedding 几乎可以引入任何信息进行编码，使其本身就包含大量有价值的信息，所以通过预训练得到的 Embedding 向量本身就是极其重要的特征向量。