中图分类号: 学校代码: 10055
UDC: 密级: 公开

南開大學

硕 士 学 位 论 文

基于互联网日志的用户行为分析算法和用户模型的研究

Behavior analysis and user modeling based on Internet access logs

论文作者 孙卓豪 指导教师 章辉

申请学位 学士 培养单位

学科专业 电子信息科学与技术 研究方向

答辩委员会主席 评 阅 人

南开大学研究生院

二〇一六年五月

南开大学学位论文使用授权书

根据《南开大学关于研究生学位论文收藏和利用管理办法》，我校的博士、硕士学位获得者均须向南开大学提交本人的学位论文纸质本及相应电子版。

本人完全了解南开大学有关研究生学位论文收藏和利用的管理规定。南开大学拥有在《著作权法》规定范围内的学位论文使用权，即：(1) 学位获得者必须按规定提交学位论文(包括纸质印刷本及电子版)，学校可以采用影印、缩印或其他复制手段保存研究生学位论文，并编入《南开大学博硕士学位论文全文数据库》；(2) 为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆等场所提供校内师生阅读，在校园网上提供论文目录检索、文摘以及论文全文浏览、下载等免费信息服务；(3) 根据教育部有关规定，南开大学向教育部指定单位提交公开的学位论文；(4) 学位论文作者授权学校向中国科技信息研究所和中国学术期刊 (光盘) 电子出版社提交规定范围的学位论文及其电子版并收入相应学位论文数据库，通过其相关网站对外进行信息服务。同时本人保留在其他媒体发表论文的权利。

非公开学位论文，保密期限内不向外提交和提供服务，解密后提交和服务同公开论文。

论文电子版提交至校图书馆网站：http://202.113.20.161:8001/index.htm。

本人承诺：本人的学位论文是在南开大学学习期间创作完成的作品，并已通过论文答辩；提交的学位论文电子版与纸质本论文的内容一致，如因不同造成不良后果由本人自负。

本人同意遵守上述规定。本授权书签署一式两份，由研究生院和图书馆留存。

作者暨授权人签字：_____

20　　年　　月　　日

南开大学研究生学位论文作者信息

论 文 题 目	基于互联网日志的用户行为分析算法和用户模型的研究				
姓　　名	孙卓豪	学号	1210403	答辩日期	
论 文 类 别	博士 □　学历硕士 □　硕士专业学位 □　高校教师 □　同等学力硕士 □				
院 / 系 / 所			专　　业		
联 系 电 话			**Email**		
通讯地址 (邮编)：					
备注：					

注:本授权书适用我校授予的所有博士、硕士的学位论文。由作者填写 (一式两份) 签字后交校图书馆，非公开学位论文须附《南开大学研究生申请非公开学位论文审批表》。

南开大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下进行研究工作所取得的研究成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：_____ 年 月 日

非公开学位论文标注说明

根据南开大学有关规定，非公开学位论文须经指导教师同意、作者本人申请和相关部门批准方能标注。未经批准的均为公开学位论文，公开学位论文本说明为空白。

论文题目			
申请密级	□ 限制 (≤2 年) □ 秘密 (≤10 年) □ 机密 (≤20 年)		
保密期限	20 年 月 日至 20 年 月 日		
审批表编号		**批准日期**	20 年 月 日

南开大学学位办公室盖章 (有效)

限制 □ 2 年（最长 2 年，可少于 2 年）

秘密 □10 年（最长 5 年，可少于 5 年）

机密 □20 年（最长10年，可少于10年）

摘　　要

随着互联网的高速发展，用户留下越来越多的网络行为信息。基于这些用户信息，分析用户行为和研究用户模型，研发适合海量数据的自然语义分析、用户兴趣爱好挖掘等算法对预测用户行为及商品推荐至关重要。本文基于 Hadoop 和 MapReduce 大数据处理平台，针对大数据精准营销的难点，实现分布式协同过滤推荐系统并构建用户兴趣矩阵，开展数据挖掘技术、场景化实践的实证研究，构建面向新媒体业务的场景化智能营销平台，并针对新媒体公司自有海量数据的特征与需求进行系统优化，提高资源利用率及推荐效果，从而全面提升公司的用户运营和内容运营能力。

关键词： Hadoop 和 MapReduce；海量数据处理；自然语义分析；数据挖掘；协同过滤推荐

Abstract

Today Internet users, typically 4G mobile users, leave billions tons of data on Servers. Base on such big data, it's urgent to develop algorithm about natural semantic analysis, text mining to predict users' behavior and recommand related content. This paper focus on big data analysis and natural semantic analysis, develop some data mining algorithm and build a system of distributed collaborative filtering recommendation based on Hadoop and MapReduce, to pop up Internet company's profit.

Key Words: Big data; Natural semantic analysis; Data mining; Hadoop and MapReduce; Collaborative filtering

目　录

摘要 …… I
Abstract …… II
第一章　背景及研究现状 …… 1
第一节　背景 …… 1
第二节　本文结构 …… 3
第二章　相关算法及 Hadoop 分布式数据处理 …… 4
第一节　语义分析 …… 4
第二节　Hadoop 和 MapReduce …… 5
第三节　基于用户历史的协同过滤算法 …… 7
第四节　基于模型的协同过滤算法 …… 9
第三章　用户上网数据采集与处理 …… 12
第一节　网络信息通信协议及其结构 …… 12
第二节　网络内容采集…… 13
第三节　网络信息处理及分类 …… 14
第四节　在 Hadoop 上实现基于用户历史的协同过滤算法…… 15
第五节　在 Hadoop 上实现基于模型的协同过滤算法 …… 17
第四章　系统工作流程与实证分析 …… 20
第一节　系统整合 …… 20
第二节　真实数据测试…… 21
第五章　总结 …… 25
参考文献 …… 26

第一章　背景及研究现状

第一节　背景

随着互联网的普及，人们每天花费越来越多时间通过网络工作，娱乐。目前，得益于芯片技术极速发展，比尔盖茨的目标，“地球上每个人都有个人电脑”，已基本实现，其中有 40% 的电脑连接着互联网；全球大概 30 亿人持有计算能力相当于 1980 年房间般大小的超级计算机的智能手机，这些移动手机用户使用各地运营商大力推广的 4G 上网服务，实现随时随地上网的梦想。除了电脑和手机，人们还将可穿戴设备及各类物联网接入互联网中。因此，互联网各网络节点每天都能采集到大量用户的上网信息。基于这些用户信息，分析用户行为和研究用户模型，我们能挖掘出用户的兴趣爱好，由此推荐相关的内容，从而提高互联网的交互能力以及网络运营商内容运营能力。

在多篇论文中，研究人员已经对用户行为进行了细致的研究。比如使用 K-means 距离算法 [1] 把行为类似的用户分为几组；又如研究上下文推断用户连续性爱好 [2]; 也有使用支持向量机给用户类型分类的 [3]；还有研究用户偏好随群体交流改变的 [3, 4]。根据协同过滤算法 [5]，用户行为的建模可以描述为一个协同过滤中的 User-Item 矩阵。为了得到这个矩阵，本文要实现一个协同过滤的推荐系统。

目前，大量互联网公司采用推荐系统给用户推荐相关的内容。全球最大网上购物平台 Amazon 最早将推荐系统引入网络销售领域 [6]。之后，推荐系统被应用于各种类型的互联网服务中。比如影视服务方面，视频网站 Netflix 将推荐系统与 A/B testing 结合起来 [7]。Google 旗下的 YouTube 给用户推荐视频 [8]。

目前推荐系统的主要使用协同过滤算法。有两种等效的协同过滤算法，一种是基于用户的协同过滤，另一种是基于物品的协同过滤。基于用户的协同过滤算法通过对相似用户的评分求和来预测用户对目标物品的评分 [5]。基于用户协同过滤算法是该类算法的最简形式 [9]，它能推广到其他复杂模型，比如引入细颗粒度的邻近权重因子 [10]，迭代寻找邻近方法 [11]，或者基于用户的个人资料计算用户相似度 [12]。

本文基于 Hadoop 和 MapReduce 大数据框架实现推荐算法。目前开源的 Hadoop 和 MapReduce 框架是 Google 的 GFS、BigTable、MapReduce 三种技术的衍生版本。Google 于为了解决大数据处理等技术问题，发表了三篇论文，其中描述了 GFS、BigTable、MapReduce 三种技术 [13–15]，但是 Google 并没有公布这三种技术的实现方法。因此雅虎等互联网公司联合起来，并在 2006 年建立了开源项目 Hadoop，目的是根据 GFS、BigTable、MapReduce，实现开源分布式数据处理系统。除了 Hadoop 和 MapReduce，目前 Hadoop 开源项目还包括一系列自动化部署、管理 Hadoop 和 MapReduce 的辅助工具，如 YARN、ZooKeeper，还包括基于 Hadoop 实现的关系型数据库 HBase。另外，基于 Hadoop 的算法引擎 Spark 以及数据挖掘算法库 Mahout 为大数据处理提供极大便利。

为了从用户上网流量记录中分析用户的兴趣爱好，我们还需要分析用户曾经浏览过的网页。分析网页的主要内容则必须使用自然语言分析相关的算法。自然语言分析是一个包涵计算机科学、人工智能以及计算机语言学的学科领域，它主要处理计算机与人类之间的交互问题。目前自然语言分析大多数算法是基于机器学习实现的。机器学习使得自然语言分析算法能自动地从基于现实语言构建的大型语料集中学习语言的规则。目前，自然语言分析研究的主要问题能分成三类：理解自然语言，从自然语言输入集中延伸相关的意思以及生成自然语言。语义分析是自然语言分析中的一个子集，它的主要目的是从计算机的角度，提取关键词，通过一系列非线性函数将关键词之间的联系展现出来，达到理解的目的。

本文所介绍的系统从互联网提供商（ISP）的流量分析入手。由于大多数用户通过 ISP 上网，ISP 的节点路由或者交换机能采集到用户访问各种类型的网站的流量。这与之前提到的公司不同，他们只关注与公司业务相关的流量。而从 ISP 的角度看，通过分析用户访问各种类型网站的流量，我们可以获得用户的兴趣爱好。利用 ISP 在网络接入的优势与特点，我们可以向互联网用户全天候地提供全方位的推荐服务。

随着全球互联网进入 HTTPS 时代，许多互联网服务已经改为使用 HTTPS，即将网页等网络信息进行 TSL 或者 SSL 加密传输。这就意味着许多网络数据虽然流经 ISP 的路由或者交换机，但是 ISP 无法查看数据包中的内容。要读取用户访问网站的流量似乎越来越难。但是作为 HTTPS 领导者的 Google，也承认其目前有 25% 的数据传输没有加密。并且，以色列大学的研究团队最近发表一篇论

文表明他们使用统计方法分析 https 加密流量，能够以 96% 的准确性确定用户的操作系统、浏览器以及他们正在浏览的网页。这就表明有大部分的网络数据是可以读取的。

本文就从学术的角度，探讨推荐系统实现的可行性及其中的关键技术。

第二节　本文结构

本文研究的主要目标是构建用户模型，即得出代表用户模型的 User-Item 矩阵。根据这个目标，本文提出了基于大数据分析的推荐系统。其中第一章主要介绍了相关背景及研究现状。第二章主要介绍相关算法及简单叙述 Hadoop 分布式数据处理的概念。第三章主要介绍用户上网数据采集，网站语义分析，数据处理及存储，基于 Hadoop 的推荐算法等模块的工作细节。第四章主要介绍整体系统工作流程，以及实证分析。第五章总结以上内容。

第二章　相关算法及 Hadoop 分布式数据处理

本文提出的推荐系统基于语义分析、协同过滤算法、Hadoop 和 MapReduce 等几个基本算法及基本概念。以下简单介绍这几个算法及概念。

第一节　语义分析

字、词是自然语言中最重要的一种独立、有意义的元素。一句话离不开字、词，整句话的意思也由字、词的独自意思组合变形而成。亚洲语言，特别是中文，它们和拉丁语言，如英语、西班牙语，之间有巨大的差异，因为亚洲语言缺少界定符号来标定字、词的边界，同时许多亚洲语言的字、词没有特定的，明确的意思。在中文语言处理中，最基础的工作是划分字、词，也就是把汉语句子划分成一连串的字词，因为，这是自然语言处理接下来的工作，如词义类型标记、语法分析等，的基础。为了从用户上网流量记录中分析用户的兴趣爱好，我们需要通过语义分析的方法对用户曾经浏览过网页的内容进行提取、标记和分类。语义分析包含以下主要步骤：

1. 句尾检测

   这一步主要是把一段文字分成一个有意义句子的集合 [16]。因为句子一般都含有思想的逻辑单元，所以句子的语法是可以预测的。而切词工作是建立在单个句子上的，所以分析文章一般从句尾检测开始。

2. 切词

   这一步主要在单个句子上进行操作，将句子分为单个词。切词工作对于不同的文字有不同的切法，比如英文切词工作主要按空格切分，但还要注意不能混淆点号的意义；而中文没有空格，切词时要按照语料库是否存在该词来切分。

3. 词义类型标记

   这一步主要是在切好的词语中标记词语的类型，比如是动词还是名词或者是长名词中的一部分。对于社交网络中的语言，比如推特或者微博，语言比较口语化，会出现大量动词名词混用情况 [17–19]。通过标记，机器甚至能指出名词或者是长名词是属于什么类型，比如是人，地

点，或者组织。

4. 分块

这一步主要是在有逻辑意义的整句和复合标记的基础上分析词语。根据预先定义好的语法，将标记了类型的词语分成块。

5. 提取主要内容

这一步主要是进一步为分好的块标记名称或者类型，这与第三步词义类型标记不同，因为这一步以块为整体标记名称或者类型。

其中每一步都很关键，因为一旦其中一步出错，误差会逐步传播，使得语义分析出严重错误。语义分析可以用经典的 F1 score 来计算其准确程度。下面简单介绍计算 F1 score 方法。在语义分析完成后，统计以下数量：

- TP：被正确识别为实体的词语；
- FP：本来不应该被正确识别为实体的但确实被正确识别为实体的词语；
- TN：本来不应该被正确识别为实体的，同时没有被正确识别为实体的词语；
- FN：本来应该被正确识别为实体的，但没有被正确识别为实体的词语。

此时再定义精确度 *precision* 与召回率 *recall* 如下：

$$precision = \frac{TP}{TP+FP} \tag{2.1}$$

$$recall = \frac{TP}{TP+FN} \tag{2.2}$$

则可以用下式来描述语义分析模型的准确度为

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{2.3}$$

第二节 Hadoop 和 MapReduce

随着互联网的发展，用户与物品间交互的数据越来越多，基于单个机器的推荐系统的处理效率和可扩展性已经开始阻碍极速增长的数据的快速处理。

在生产环境中，运行一个推荐系统所必需的离线计算必须遵守严格的时间和资源的限制，同时这又是更大的分析流程中的一部分，所以其必须定期被执行。此外，必须通过重复运行推荐系统，以在交叉验证集中寻找良好模型参数。但这在经济上和操作上往往又是不可取的，因为在一台机器上执行这种离线计

算十分有可能会失败。并且面对着不断增长的数据量，要不断进行硬件升级来提高机器的性能，以满足时间上的限制。正是由于这些缺点，使用单台机器运行推荐系统变得昂贵并且难以操作。而在多个机器组成的集群上运行推荐系统，既能提高容错水平，又能解决数据密集的问题。而 Hadoop 和 MapReduce 正是保存大数据以及并行处理数据的的最佳选择。

Hadoop 由多个部分组成，其最基础的组件是 Hadoop 分布式文件系统（HDFS）。HDFS 的运行机制是把要写入的数据分发到大量计算机集群中，仅一次写入，就可以多次读取用于分析。但它与其它分布式系统相比，最大的区别是不能删除，只能进行格式化。

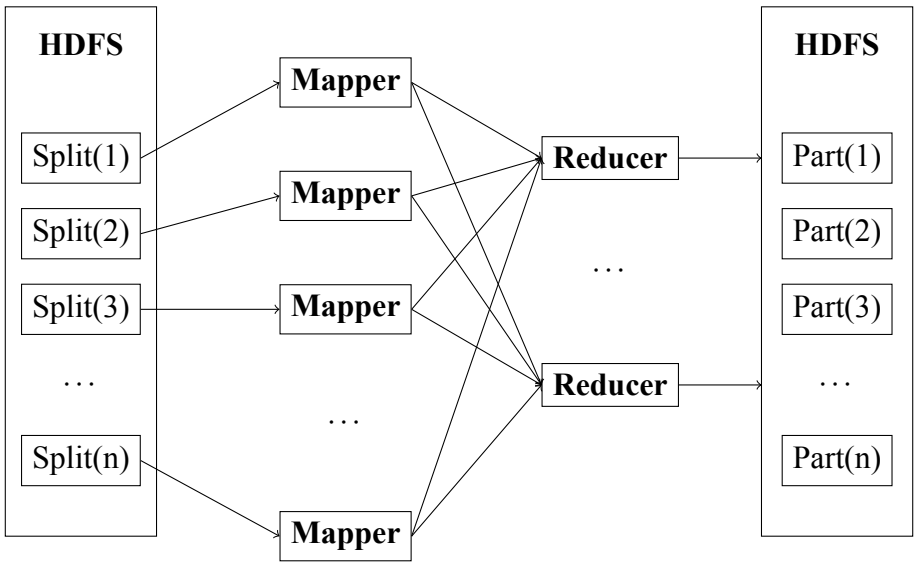

图 2.1　Hadoop 的 MapReduce 工作流程图

Hadoop 集群由一个主机和多个从机组成。其中，主机是最重要的，它保持整个 HDFS 文件系统的正常运行，同时保持与每台从属机器的心跳连接。HDFS 中存储的文件被划分为的固定大小的存储块。当文件被放置在 HDFS 时，主机将分配给每个存储块不同的全局标签。为了保证数据的可靠性，每个存储块被多次复制，然后保存到不同的从机中去。由于主机保持整个 HDFS 文件系统的正常运行，所以主机保存每个存储块的全局标签，并保存存储块的地址，还保存存储块和原始文件之间的映射关系。

通常 Hadoop 集群只有一台主机，这不仅简化大型机管理策略，同时也给主机整个系统全局信息，以方便储存和安置存储块。有时上述 HDFS 除了有一个主机以外，还有一个影子主机，它缓存了主机的运行信息，这种设计使得它很

容易从操作故障中重新启动并恢复。

Hadoop 的主要执行框架即 MapReduce。对于保存在分布式文件系统的数据，MapReduce 在每个数据节点上，获取本地的数据副本，然后进行本地运算。

图2.1为 Hadoop 的 MapReduce 工作流程图。从该图中可以发现，MapReduce 框架包含两个阶段，map 阶段和 reduce 阶段。输入数据和计算后的输出数据都是一个 (key,value) 集合。用 $< k, v >$ 来表示这些键值对。key 主要用在 reduce 阶段，用来决定哪个 value 结合在一起。用两个函数来表示这两个过程：

$$\textbf{Mapper} \quad : \quad < k1, v1 > \rightarrow list < k2, v2 > \tag{2.4}$$

$$\textbf{Reduce} \quad : \quad < k2, list < v2 >> \rightarrow list < k3, v3 > \tag{2.5}$$

计算过程从 map 阶段开始，在该阶段中，各从机接受主机的分配，然后同时地、并行地执行 map 函数。此时保存在分布式文件系统的输入数据将被切分成很多个部分。进行每次切分的同时主机就会对从机指定一个 map 任务。在 map 阶段的后期，每个 map 函数的输出键值对将会对 map 函数中生成的键进行哈希分区。紧接着进入 reduce 阶段。在 reduce 阶段的初期，每个分区中的键值对将基于键分别排序，接着按照键的排序进行合并。有同样的键的分区将被分配到同一个 reduce 任务中。最后，在 reduce 阶段的后期，reduce 函数输出键值合并的最终结果。

在 Hadoop 的 MapReduce 工作流程中，一个工作记录服务器作为主节点把数据分成几部分作为 map 阶段的输入。同时任务记录服务器作为数据节点保存 map 函数中间生成的结果到 HDFS 中。

Hadoop 集群对组成集群的计算机性能要求不高，所以即使古老的、配置低的服务器也能作为集群的一部分。Hadoop 会基于地点计划分配 MapReduce 运算，同时提高集群整体的输入输出效率来减少运算的成本。这种工作方式将原本单独机器的运算工作合理地分配到多个机器中，不仅提高了数据运算的可靠性，还大大地提高了运算的效率。

第三节　基于用户历史的协同过滤算法

协同过滤算法基于三个假设：人们有共同的兴趣爱好；他们的兴趣爱好是稳定的；可以根据兴趣爱好的历史数据预测用户的选择。协同过滤算法主要比较用户之间的行为，找到最近的相邻用户，根据最近的相邻用户的行为预测用

户的兴趣爱好。

对于 m 个用户的用户集 U，和 n 个物品的物品集 I，用户 $u_i \in U$ 与物品 $i_j \in I$ 的交互信息记录为一个数值 $r_{i,j} \in R$。若用户 i 对物品 j 无交互行为，则 $r_{i,j}$ =?。最基本的协同过滤算法所研究的问题为：基于 R，给每位用户推荐一个不包含该用户已经评价过物品的列表，其中的物品按照符合用户兴趣程度递减排序。

协同过滤算法能分为主要两个步骤：第一步是计算用户 x 与用户 y 之间的相似程度：

$$S_{x,y} = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}} \tag{2.6}$$

其中 $r_{x,i}$ 为用户 x 对物品 i 的评分，$\bar{r}_x$ 是用户 x 的平均评分，I_{xy} 为与用户 x 和用户 y 曾发生交互行为的物品集。

第二步计算用户 u 对物品 i 评分的预测值 $r_{u,i}$：

$$r_{u,i} = \bar{r}_u + \frac{\sum_{u' \in U^k} (r_{u',i} - \bar{r}_{u'}) S_{u,u'}}{\sum_{u' \in U^k} S_{u,u'}} \tag{2.7}$$

其中 U^k 为与用户 u 距离最近的 n 个用户。最后，只需从物品集中筛选出预测分数前几的物品，推荐给用户 u。以上被称作基于用户的协同过滤算法，其中从以下几个方面，可以看出它是从用户的角度来工作的：

- 首先在计算相似度的时候，它是计算两个用户之间的相似度，这与基于物品的协同过滤算法正好相反，后者是计算两个物品之间的相似度；
- 在计算评分的预测值之后，对每位用户的商品评分进行高低排序，然后推荐给用户排前几名的物品。这与基于物品的协同过滤算法正好相反，后者是对每位商品的用户评分进行高低排序，而评分相对高的用户正是最有可能对该物品感兴趣的人。

就如上面所提到的，基于物品的协同过滤算法要先计算物品之间的相似度，得到物品的相似度矩阵。然后在相近的物品之间计算用户可能评价的评分。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上是一样的，其中最大的区别是其计算的顺序不同，一个更多使用 User-Item 矩阵的行，另一个更多使用 User-Item 矩阵的列。

但是，User-Item 的行数与列数不同，会对这两种算法的性能有很大影响，特别是在单个机器上执行算法的时候。如果用户数远远大于物品数，此时如果使用基于用户的协同过滤算法，计算的用户相似度矩阵就会很大，即很多行（相

似度矩阵的行与列相等）。相反，如果使用基于物品的协同过滤算法，计算的物品相似度矩阵就比较小。两种算法对于单个机器的固定内存大小来说尤其重要，因为相似度矩阵过大，有可能超过内存大小，或者占用过多内存使接下来的计算变慢。

以上两种算法都称作基于历史数据的协同过滤算法，因为他们都使用了用户与物品间交互的历史数据。基于历史数据的协同过滤算法存在两个缺点：

- 在计算用户之间、物品之间的相似度矩阵时，时间复杂度为平方级别，即计算成本很高；
- 推荐算法的精度取决于所用的相似度计算函数。如果没有按照用户与物品间历史交互数据选择合适的相似度计算函数，推荐算法的效果会不太理想。

第四节 基于模型的协同过滤算法

为了解决上面的问题，新的协同过滤算法被提出来，比如基于模型的协同过滤算法。与基于历史数据的协同过滤算法不同的是，基于模型的协同过滤算法首先从 User-Item 矩阵中训练出一个模型，然后以这个模型再进行计算，推荐物品给用户。这个模型可以用公式描述：

$$f(p_i, q_j) \to R_{i,j}, i = 1,2,\ldots M, j = 1,2,\ldots N \tag{2.8}$$

其中 p_i 和 q_j 代表用户 i 和物品 j 的一对模型参数，f 是把模型参数映射到已知的 User-Item 矩阵数据的一个模型参数。所以基于模型的协同过滤算法的主要任务就是在模型函数 f 下，通过已知的 User-Item 矩阵数据 $R_{i,j}$，预测参数 p_i 和 q_j。

在众多的基于模型的协同过滤算法中，潜在因子模型运行性能突出，因为它解决了协同过滤算法中最常见的两个问题：

- User-Item 矩阵过于稀疏，使得矩阵之间的运算时间、空间复杂度高；
- User-Item 矩阵虽然行列众多，但是其秩非常低，也就是说在这个 User-Item 二维空间中，许多信息是重合的，冗余的。

下面就来说明潜在因子模型的运行规则。令 R 为 $|C| \times |P|$ 的 User-Item 矩阵，其中包含了用户与物品间交互的历史信息。C 代表用户集，P 代表了物品集。如果用户 i 曾经与物品 j 间有交互动作发生，并产生了某个值 $r_{i,j}$，代表着这个交互动作的强度。潜在因子模型就是把 R 渐近地分解为两个秩为 r 的特征矩阵 U

和 M，使得 $R \approx UM$。$|C| \times r$ 的矩阵 U 代表用户集的潜在因子，即 R 的行向量。$r \times |P|$ 的矩阵 M 代表物品集的潜在因子，即 R 的列向量。而用户和物品之间的交互强度的预测值可以在低维的矩阵空间中进行，即 $u_i^{\tau} m_j$，其中 u_i 是代表用户 i 的向量，m_j 是代表物品 j 的向量。所以，接下来的主要问题就是计算矩阵 U 和矩阵 M。

比较常见的算法是随机梯度下降算法，即随机选取矩阵 U 和矩阵 M 的初始值，然后计算 $u_i^{\tau} m_j$ 与 $r_{i,j}$ 之间的误差，通过迭代的方式顺着梯度最大方向下降而得出最优解。

$$U^*, M^* = \underset{U,M}{\operatorname{argmin}} \{ \frac{1}{2} \sum_{i=1}^{C} \sum_{j=1}^{P} I_{i,j} (r_{i,j} - u_i^{\tau} m_j)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_M}{2} \|M\|_F^2 \} \tag{2.9}$$

其中 U^* 和 M^* 代表最小化运算后的最优解，$I_{i,j}$ 代表示性函数，即当其自变量大于零，函数值为一。

另外一种计算方法称为交替最小方差算法。给定一个阈值 ε，通过以下步骤找到矩阵 U 和 M：

1. 随机选择一个矩阵 M；
2. 固定矩阵 M，优化矩阵 U; 对逐个用户 u_i 进行优化，即固定 m_j，需要找到

$$\underset{u_i}{\operatorname{argmin}} \sum_{j \in R_i} (r_{i,j} - u_i \cdot v_j)^2 \tag{2.10}$$

这是线性最小二乘方程，可以得到

$$u_i = (M_{*,i}^{\tau} M_{*,i})^{-1} M_{*,i} R_{*,i} \tag{2.11}$$

其中 $M_{*,i}$ 是有来自用户 u_i 的评价的 M 的子集。

3. 固定矩阵 U，优化矩阵 M，与上一步相似;
4. 循环第 2、3 步，直到相关系数的变化量小于 ε，即可得到收敛解。

对比以上两种计算方法，我们可以发现，对于单个机器而言，随机梯度下降算法更适合于低阶矩阵因子化的计算。这是因为随机梯度下降算法相比于交替最小方差算法更容易实现，并且前者计算量相对来说更小。

但是，随机梯度下降算法是序列算法，因为它在每次迭代后都要更新模型的参数。而交替最小方差算法相比于随机梯度下降算法的优点在于，前者可以运用于并行计算。

虽然交替最小方差算法的计算量比随机梯度下降算法的计算量大得多，但是它的交替迭代方式让他更适用于并行计算。例如，当我们计算用户特征矩阵 U 时，U 的第 i 行 u_i 能同时用于解一个最小二乘方程，这个最小二乘方程中只包含 R 的第 i 行 r_i，即用户 i 的交互信息，以及对应于 r_i 中不为零的位置的矩阵 M 中的所有列 m_j。u_i 的同时重复使用是与矩阵 U 中其它行的计算相互独立的，所以矩阵 U 的计算很容易实现并行化，只要可以保证计算过程中能及时获取矩阵 R 的行向量以及对应的矩阵 M 的列向量。

从数据处理的角度来说，交替最小方差算法要在 User-Item 矩阵 R 与物品特征矩阵 M 之间，实现一个并行的联合操作。类似的，当计算物品特征矩阵 M 时，需要在 User-Item 矩阵 R 与用户特征矩阵 U 之间实现一个并行的联合操作。本文将在下一章实现其具体步骤。

第三章　用户上网数据采集与处理

本章主要介绍用户上网如何留下信息，如何将用户上网信息转变为推荐系统所需要的训练集。其中包括了以下内容：网络信息通信协议及其结构、网页主要内容分布位置、网络流量信息采集、去除标签和过滤内容的策略、格式化、存储采集到信息的方法。

第一节　网络信息通信协议及其结构

互联网用户通过各类软件发送、接受信息。对于不同的软件，有不同的信息发送、接受格式, 称作协议。以大多数网络浏览器使用的基本 HTTP 协议来说，其请求的头部有以下常用的信息

表 3.1　HTTP 协议请求头部常用的信息

头部名称	描述
回应接收类型	接受从对方服务器回应发来文件的类型
请求内容类型	发送请求时请求主体的 MIME 类型（一般由 POST 请求或者 PUT 请求使用）
日期	发送信息时的日期与时间
主服务器	服务器的域名（虚拟服务器）以及服务器所监听的 TCP 端口号（如果使用标准端口号，则忽略端口号）
引用源	前一个网页的地址（该网页上的链接到当前的网页）
用户代理	用户所使用的代理的信息（浏览器的信息等）

表3.1简单介绍了 HTTP 协议请求头部中常用的信息。根据 HTTP 请求头部我们可以获得用户请求网站 URL，请求时间，使用浏览器类型信息。通过网站 URL 可以简单将网页分为几类：

- 综合网页类。一些门户网站的首页通常含有各种关键词，所属范围较大，与用户的兴趣爱好没有多少关系，比如搜狐首页、新浪首页；
- 搜索类网页。这些网页会包含一个或多个搜索关键词，并且这些关键词由用户输入，基本上能代表用户的兴趣爱好。例如在百度搜索等搜索引擎页

面，通常会在其 URL 后接上搜索关键词的 UTF-8 编码；

- 终点页面。这类网页通常包含一个或多个关键词，接着会有大篇幅文字描述相关内容。

根据以上分类可以为信息采集分类系统设计第一层过滤分类规则。另外还能通过引用源的内容获取网页间的联系，经相互对比，增加网页信息抓取的准确度。

第二节　网络内容采集

仅通过分析网络请求与响应头部的信息，获取到的信息对于本系统来说远远不够。所以需要记录并保存所有网络流量包，然后解压信息包，获得其中网页信息。对于使用其他协议的软件来说，只要按照该协议的约定，也能正确解压信息包，获取其中内容。

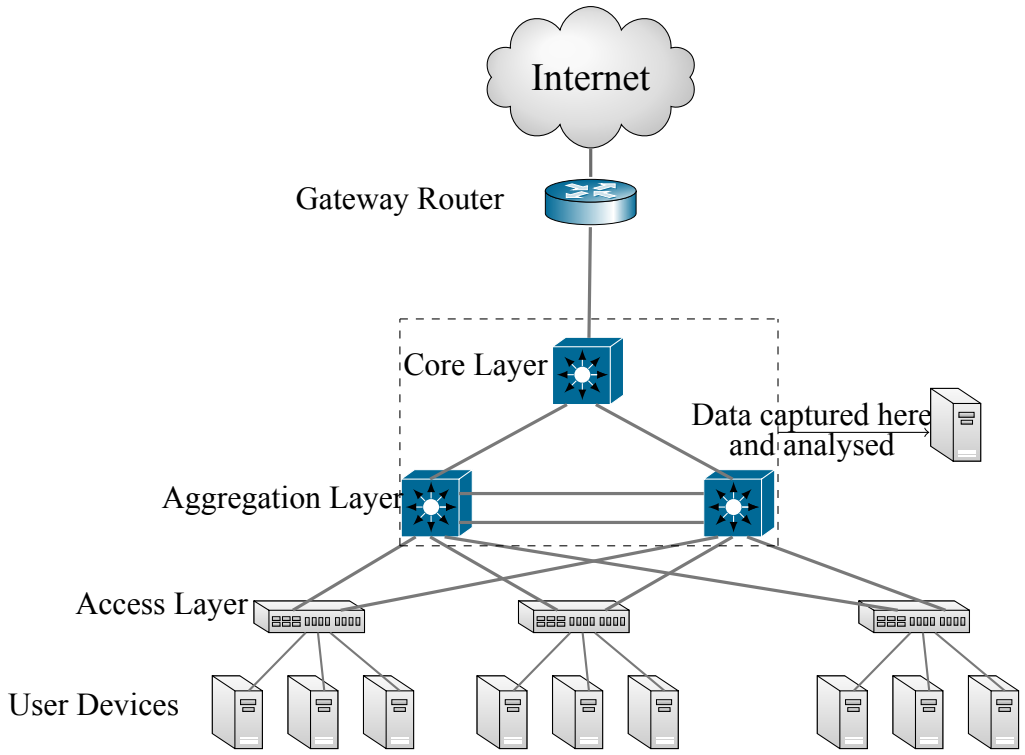

图 3.1　用户数据采集及处理

在处理大量网络流量信息之前，可以先用小型工具抓取网络包进行分析。Wireshark 是一个跨平台的网络流量分析工具，能用于识别与分析多种网络协议包。当网卡处于 Promiscuous 模式时，Wireshark 能监听其所在局域网的所有网

络包。从 Wireshark 的 GUI 窗口可以简明地查看包信息，以及解压包获取其中内容。Wireshark 还能设置强大的过滤条件，从而筛选出目标网络包。Wireshark 适用于 Windows 平台用户，对于使用 Linux 的用户而言，可以使用 Iptables 完成上述分析步骤。

对于大型的路由器或交换网络的情况，如图3.1中的多层级网络，Wireshark 和 Iptables 就不太适用。目前，国内外比较流行使用的大型网络流量信息采集方式有 Cisco 公司开发的 NetFlow 工具。支持 NetFlow 的路由器和交换机能够收集所有接口上 IP 交换数据，并且能把数据导出到 NetFlow 信息分析服务器上。国内研究人员也设计了类似的分布式网络流量分析系统 [20–22]。

本文根据 NetFlow 采集模式设计采集及处理模块，如图3.1所示。对于每个地区的路由器或者交换机，都连接到信息分析服务器。信息分析服务器将对文件的 Content-Type 进行过滤。目前，包含有效信息的网络文件类型大部分为 HTML，另外还有一部分内容通过 JSON 文件与 XML 文件传输。但是随着 Javascript 的极速发展，相当一部分 HTML 由浏览器从 Javascript 文件中生成，即网络内容不再是显式传输，而是隐藏在 JSON 文件与 XML 文件中，填充到 Javascript 生成的 HTML 框架中。这对本文的网页信息分析造成了一定的困难。经过简单的过滤，可以记录保存表3.2中标定的信息。

表 3.2　信息分析服务器记录信息格式

User ID	Request URL	Time	Location	Content	Device

第三节　网络信息处理及分类

通过前两节的分析，混杂的网络信息已经归档为按表3.2中格式的记录。为处理和分类这些有序信息，本节形成策略如图3.2：

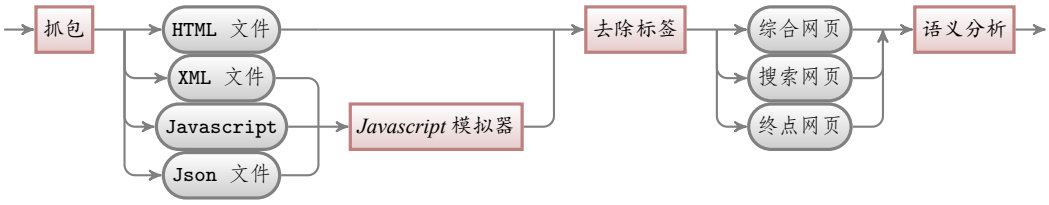

图 3.2　网页信息处理分类策略

该策略的主要描述如下：网页信息处理分类首先将表3.2中的网络协议包解压，然后获得 HTML 文件，或者 Javascript 文件与 XML 文件、JSON 文件。将后

三者放入 Javascript 虚拟机中，可以生成 HTML 文件。接着按照 HTML 文件的请求 URL 或者标题内容进行分类，按前一节所描述的网页分类分为综合网页，搜索网页或者终点网页。由于 URL 包含网站名称，以及访问路径，故 URL 包含部分关键词，可将 URL 作为一个语句进行语义分析。另外，由于 HTML 文件结构可知，其中的内容镶嵌在各种 HTML 标签中。故分析 HTML 文件首先要去除各种标签。通过实现监督学习算法 [23] 去除标签，将剩下的文字内容输入语义分析模块。最后，按网页类型做不同分析重点的语义分析。

按照第二章第一节描述的语义分析原理，本模块处理网页，分析得出所有关键词。从关键词中取出所有名词，并按照出现次数进行排序。将高频名词按照表3.3格式输出保存。

表 3.3 信息分析服务器输出信息格式

User ID	Request URL	Time	Location	Item	Device

第四节 在 Hadoop 上实现基于用户历史的协同过滤算法

根据 Hadoop 与 MapReduce 的原理及特点，我们可以将基于用户历史的协同过滤算法计算过程分解到 Map 阶段与 Reduce 阶段中。

首先要进行简单的计数工作，即计算每个 User-Item 对的数目。将该数目代入逻辑函数 $S(t) = 1/(1+e^{-t})$ 进行简单的归一化。得到 User-Item-Value 格式的一组数据，形成下一步所需的 User-Item 矩阵，记为 $R = \{< i, j, r_{i,j} >\}$，保存于 Hadoop 的 HDFS 中。以下假设 n 个用户，m 个物品，

3.4.1 计算每个用户的平均评分

每个用户 i 的平均评分可以如下计算：

$$\bar{r_i} = \frac{\sum_{j=1}^{n} r_{i,j}}{\sum_{k=1}^{l} 1} \qquad \forall i \in [1,n] \tag{3.1}$$

其中 l 表示该用户评价了几个物品。

生成的平均评分数据集记为 $U_{all} = \{< i, \bar{r_i} >\}, \forall i \in [1,n]$。具体的 Map 与 Reduce 算法如下：

- Map-I：将 i 相同的 $< i, j, r_{i,j} >$ 映射到同一个机器中，简记为 $< j, r_{i,j} >$。对于 k 个机器，本步骤的复杂度为 $o(\frac{nm}{k})$。

- Reduce-I：取出 $<i,j,r_{i,j},\bar{r}_i>$，保存到 HDFS 中，此时将 R 指向 $\{<i,j,r_{i,j},\bar{r}_i>\}$。另外保存 $U_{all}=\{<i,\bar{r}_i>\},\forall i\in[1,n]$ 到 HDFS 中。本步骤的复杂度同样为 $o(\frac{nm}{k})$。

3.4.2 计算用户间相似度

计算用户间相似度矩阵 $S=\{S_{i,j}|\forall i,j\in[1,n]\}$。定义 $U_i=\{<i,j,r_{i,j},\bar{r}_i>|\forall j\in[1,m],<i,j,r_{i,j},\bar{r}_i>\in R\}$ 为用户 i 评价过的物品集。定义 $N_{k,l}=U_k\cap U_l$，为用户 k 与用户 l 共同评价过的物品集。为了计算可行性，$N_{k,l}$ 将分到同一个节点上。具体的 Map 与 Reduce 算法如下：

- Map-II：将 $\{U_i|i\in[1,n]\}$ 映射为 $\{N_i,j|\forall i,j\in[1,n]\}$。如果有 k 个机器，本步骤的复杂度为 $o(\frac{m^2n}{k})$。
- Reduce-II：取出 $\{S_{i,j}|\forall i,j\in[1,n]\}$，保存到 HDFS 中。本步骤的复杂度为 $o(\frac{m^2n}{k})$。

此步骤可以看出可以看出，对于物品数远大于用户数的情况，本算法更适合于用户数远大于物品数的情况。

3.4.3 计算预测矩阵

将矩阵 R 变形为矩阵 I，其中矩阵 I 的数据按物品的序号排序：

$$I=\begin{pmatrix}I_1\\I_2\\\vdots\\I_m\end{pmatrix}=\{I_j|\forall j\in[1,m]\} \tag{3.2}$$

其中

$$I_j=\begin{pmatrix}<1,j,r_{1,j},\bar{r}_1>\\<2,j,r_{2,j},\bar{r}_2>\\\vdots\\<n,j,r_{n,j},\bar{r}_n>\end{pmatrix}=\{<i,j,r_{i,j},\bar{r}_i>|\forall i\in[1,n],<i,j,r_{i,j},\bar{r}_i>\in R\} \tag{3.3}$$

即 I_j 表示与物品 j 相关的所有评分数据集。定义相似度矩阵 S 为

$$S=\begin{pmatrix}S_{1,1}&\dots&S_{1,n}\\\vdots&\vdots&\vdots\\S_{n,1}&\dots&S_{n,n}\end{pmatrix}=\begin{pmatrix}S_1\\\vdots\\S_n\end{pmatrix} \tag{3.4}$$

其中 $S_i = \{S_{i,j} | \forall j \in [1,n]\}$ 表示用户 i 与其它用户间的相似度。具体的 Map 与 Reduce 算法如下：

- Map-III：将 j 相同的 $< i, j, r_{i,j} >$ 映射到同一个机器中，简记为 $< j, r_{i,j} >$，同时将 S_i 映射到同一个机器中。对于 k 个机器，本步骤的复杂度为 $o(\frac{nm}{k})$。
- Reduce-III：取出 $B =< i, j, r_{i,j}, \bar{r}_i, S_i >$，保存到 HDFS 中。本步骤的复杂度为 $o(\frac{m^2 n}{k})$。

这一步是为计算预测矩阵做准备。按照第二章的公式计算用户 i 对物品 j 的评分 $p_{i,j}$。预测矩阵 P 定义为：

$$P = \begin{pmatrix} \{p_{1,1}, p_{1,3}, p_{1,5}\} \\ \vdots \\ \{p_{i,1}, p_{i,3}, p_{i,5}\} \\ \vdots \\ \{p_{n,1}, p_{n,3}, p_{n,5}\} \end{pmatrix} = \begin{pmatrix} P_1 \\ \vdots \\ P_i \\ \vdots \\ P_n \end{pmatrix} \tag{3.5}$$

其中 $P_i = \{p_{i,j} | \forall j \in [1,m], < i, j, r_{i,j}, \bar{r}_i > \notin R\}, \forall i \in [1,n]$ 表示用户 i 对未评价物品集的所有预测分数。具体的 Map 与 Reduce 算法如下：

- Map-IV：将 i 相同的 $B =< i, j, r_{i,j}, \bar{r}_i, S_i >$ 映射到同一个机器中，简记为 $< j, r_{i,j}, \bar{r}_i, S_i >$，同时将 Map-I 生成的 U_{all} 中的 U_i 映射到同一个机器中。对于 k 个机器，本步骤的复杂度为 $o(\frac{nm}{k})$。将
- Reduce-IV：取出 $P =< i, j, p_{i,j}, >$，保存到 HDFS 中。本步骤的复杂度为 $o(\frac{m^2 n}{k})$。

至此所有用户对未评价物品集的分数 $p_{i,j}$ 都已经计算出来。即构建了完整的 User-Item 矩阵，也即用户兴趣模型。此时对于每个用户，对所有物品按评价分数大小排序，排名前 k 的物品组成一个向量。下一步可以根据这个向量推荐用户内容。

第五节 在 Hadoop 上实现基于模型的协同过滤算法

在上一章中已经具体解释了，交替最小方差算法更适合于在 Hadoop 上实现的基于模型的协同过滤算法。

对于交替最小方差算法，主要步骤是矩阵 R 与矩阵 M 和矩阵 U 之间的联合操作。通常来说，User-Item 矩阵 R 要远远大于特征矩阵。但是，本文只考虑矩

阵 M 和矩阵 U 都不需要切分，即矩阵 M 和矩阵 U 都能保存在单个运算节点的内存中，的情况下，实现具体算法。

初步估计在并行计算的过程中，需要 $\max(|C|,|P|)\times r\times 8$ 个字节的内存量。对于 1 千万数量级的用户数或者物品数，特征数为 100 的情况，大概需要 8GB 的内存，就目前的计算机水平，这个条件很容易满足。

基于以上设定，下面介绍交替最小方差算法的具体步骤。首先从用户数据集 U 或者物品数据集 M 中取相对小的一个，复制到集群的每个计算节点中。另外每个计算节点本地都有 User-Item 矩阵 R 的副本。然后，通过一个 map 运算，把矩阵 R 的副本和矩阵 M 结合起来（矩阵 R 的副本和矩阵 U 类似）。另外，在这个 map 运算中，我们还能从结合的结果实现中并行计算特征向量的逻辑，也就是说，我们可以在单独一个 map 运算中，运行一整套矩阵 U 和矩阵 M 的计算。

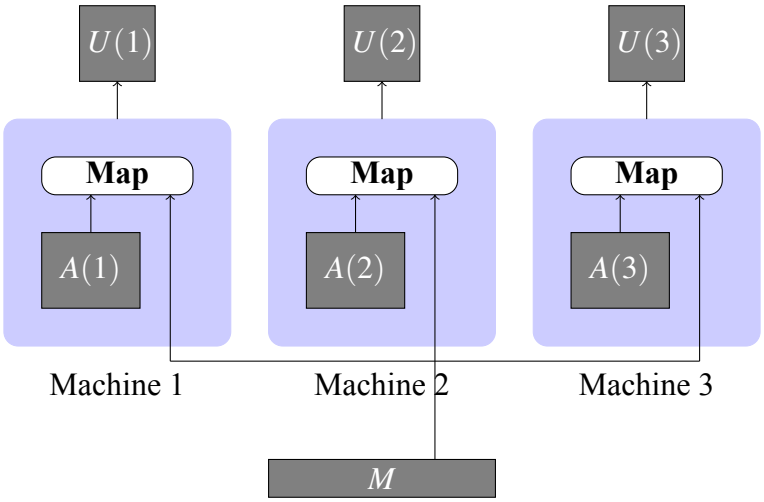

图 3.3　基于模型的协同过滤算法 MapReduce 实现

图3.3描述了在三个计算节点上计算矩阵 U 时的并行结合方法。我们将矩阵 M 分发到每个计算节点中，然后计算节点会给矩阵 M 的特征向量创建一个哈希表。保存在 HDFS 中的矩阵 R 按行分块储存，并作为 map 运算的输入。$R(1)$ 代表矩阵 R 的第一个分块。接着，map 运算读入矩阵 R 其中一行 r_i 然后选择其中部位零的下标，按照这些下标从哈希表找到矩阵 M 中对应的 m_j。然后，map 运算将解一个线性方程，这个线性方程由交互信息 r_i 和物品特征向量 m_j 构成。在得到方程解后，将结果写入 HDFS 中。对于矩阵 M 的计算，其计算步骤类似，唯一的不同点在于，要在每个运算节点分发矩阵 U，以及按照列来对矩阵 R 进行分块储存。

以上的计算方法只使用了 map 运算，因为相对于 map 运算与 reduce 运算一起使用，单独使用 map 运算在时间安排上更加灵活。另外，我们还避免使用耗时很长的混排阶段，因为在混排阶段，大量数据要通过速率不太高的网络进行传输。由于我们能够在一次 map 运算中运行矩阵结合以及计算的步骤，所以矩阵结合的结果就没必要写到 HDFS 中。在计算线性方程的初始化阶段，我们通过 Hadoop 的分布式缓存分发特征矩阵。另外，我们将 Hadoop 设置为在工作节点上重用虚拟机，并把特征向量矩阵缓存在内存中以避免接下来的 map 函数要重新读取数据。

第四章　系统工作流程与实证分析

第一节　系统整合

基于前几章的描述，我们可以综合构建一个数据挖掘系统，发现用户兴趣，推荐相关内容。图4.1展现了整体系统流程。

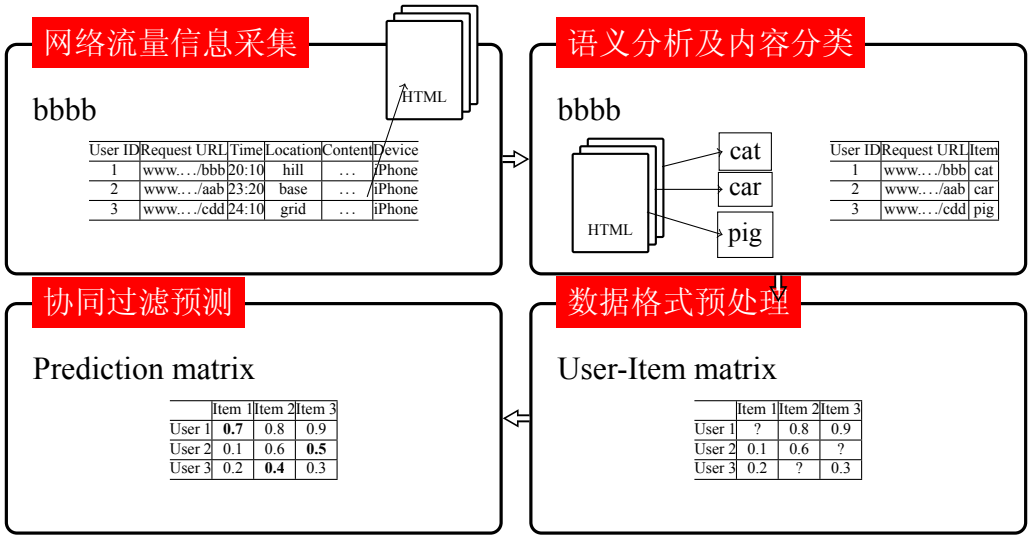

图 4.1　整体系统流程

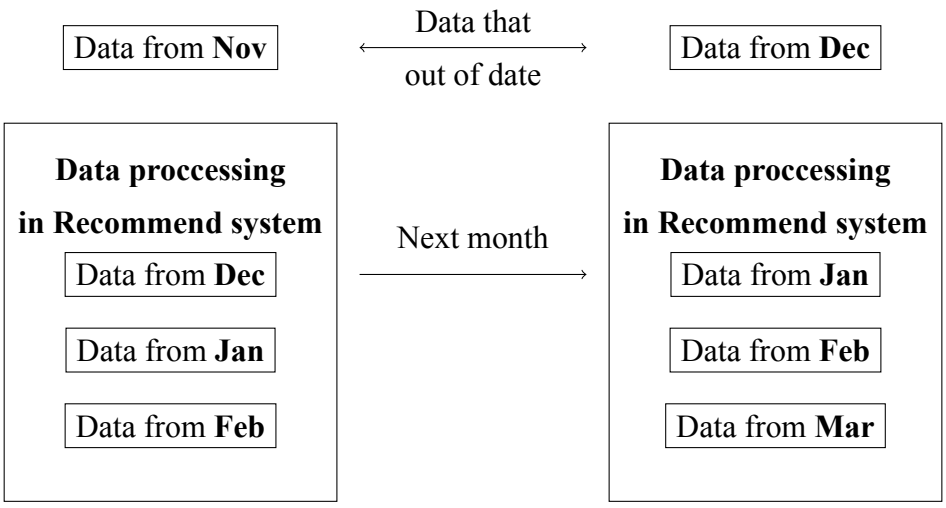

图 4.2　推荐系统数据的流动性

本文构建的推荐系统将按照图4.1中的箭头指向运行。但是在现实情况中，社会的信息是流通的，新的事物会不断出现，而人们的兴趣也会不断地改变。所以推荐系统的运行过程不是一次性的。推荐系统所使用的用户数据应该具有流的形式，即在推荐的过程中获取新的用户数据，并丢弃旧的数据。

如图4.2，在 3 月份时，处理的是 12 月、1 月、2 月的数据，为 3 月的应用访问做推荐；到 4 月份时，处理的是 1 月、2 月、3 月的数据，12 月份的数据被丢弃。或者采取加权的形式，比如最近一个月的数据权重较高，其它历史月份的权重较低。

下面将根据整体系统流程图，简单介绍真实数据测试的方法、流程及结果。

第二节 真实数据测试

本文采用的测试硬件主要包括 4 台安装了 64 位 Ubuntu14.04 以及 Hadoop2.6.4 的服务器，其配置包含 2 个 Intel 的 Xeon 的 CPU，型号为 E5-2630，核心频率 2.30GHz，2GB 内存，40GB 硬盘空间，其中一台服务器作为 Master，另外三台服务器作为 Slave。

由于测试数据的限制，本文无法对系统整体测试，只能按照各模块的功能进行测试。在图4.1中，关键步骤为第二步语义分析及内容分类以及第四步协同过滤预测。在语义分析及内容分类步骤中，本文采用了精确度较高的 ICTCLAS 中文分词系统 [24]。ICTCLAS 中文分词系统的各项测试数据在其下载页面已有详细介绍，本文不再重复。

在协同过滤预测步骤中，采用了 4 个不同的数据集，分别是：MovieLens 的 144MB 的数据集，其中包含了截止到 2016 年 1 月，240000 多位用户对 33000 多部电影的 22000000 个评分数据，以及 58000000 个类型标记；2015 年阿里移动推荐算法比赛的 129MB 的数据集，其中包含 2014 年 11 月 12 日到 2014 年 12 月 11 日间，10000 多名用户在淘宝线上与 1900000 多个商品的互动信息，即浏览、收藏、加入购物车以及购买等动作；雅虎音乐 1.5GB 的 C15 数据集，其中包括 2002 年到 2006 年间，18223179 位雅虎用户对 136736 首歌的 717872016 个评分；另外还有雅虎音乐 1.6GB 的 R2 数据集，其中包含用户对音乐、歌手、专辑的评分，数据量级与 C15 数据集相似。

在 4 个数据集中，阿里移动推荐算法比赛的数据集，相对而言，是最满足本推荐模块测试要求的。因为用户与商品的互动信息，是二值信息，即有没有

发生过该类互动。而互联网用户在浏览网页中，表现对某关键字兴趣爱好也是二值信息，即感兴趣，或者不感兴趣。当然，也可以将该二值信息改为类似评分值的连续值，即按浏览该网页时间，点击该网页等计算出兴趣爱好值。

在测试基于交互历史的协同过滤推荐系统时，要将数据集分为训练集以及测试集。训练集用于训练协同过滤推荐系统中的参数，测试集用于计算推荐的优良程度。由于数据集中的评分记录都有时间戳，所以可以按时间区分训练集与测试集。本文采用，评分记录按时间排序后，前 80% 的记录作为训练集，后 20% 的记录作为测试集的区分方案。

推荐系统的优良程度可以采取 F1 分数来评价。此时 F1 分数的定义如下：

- $PredictionSet$：预测评分大于最大评分一半的记录组成的集合
- $ReferenceSet$：测试集中评分大于最大评分一半的记录组成的集合

此时再定义精确度 *precision* 与召回率 *recall* 如下：

$$precision = \frac{|PredictionSet \cap ReferenceSet|}{|PredictionSet|} \tag{4.1}$$

$$recall = \frac{|PredictionSet \cap ReferenceSet|}{|ReferenceSet|} \tag{4.2}$$

则 F1 分数为

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

表4.1展示了协同过滤推荐系统在四个数据集中的 F1 分数。其中 F1 分数稳定在 0.07 的范围，表明协同过滤推荐系统在 F1 分数这种评价标准中表现并不太好。造成这个结果的主要原因是，F1 分数比较适用于评价二分类问题。而协同过滤算法计算得到的实数值对二分类问题的界定线比较敏感。

表 4.1 协同过滤推荐系统的性能分析

数据集	训练集大小	F1 score
MovieLens	576MB	0.0823
Aliyun contest	516	0.0644
Yahoo! R2	6.4GB	0.0876
Yahoo! C15	6.0GB	0.0713

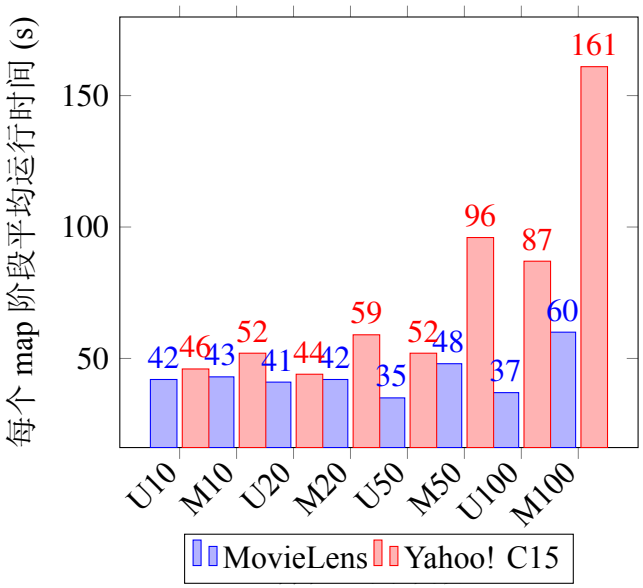

图 4.3 基于模型的协同过滤推荐系统的测试

图4.3是对基于模型的协同过滤推荐系统测试。在该测试中，我们使用了MovieLens 和雅虎音乐 C15 两个数据。对于 MovieLens 数据，无论特征空间维数 r 多大，每个 map 阶段的平均运行时间落在 35 秒到 60 秒的区间。这表明，对于小的数据集，其运行时间主要由 Hadoop 自身的工作调度决定。对于比较大的雅虎音乐数据，我们可以发现特征空间维数 r 较小时，其运行时间主要由 Hadoop 自身的工作调度决定。但是当特征空间维数 r 较大时，其运行时间主要由分发、计算特征矩阵过程决定。另外，我们还注意到，计算物品特征矩阵 M 用时比计算用户特征矩阵 U 用时更多，这是因为用户集 U 中用户数目大概是物品集 M 的 20 倍，所以要花更多时间来分发矩阵 U。实验表明，对于 MovieLens 数据，每小时能计算 37 到 47 次迭代。而对于雅虎音乐 C15 数据，每小时能计算 15 到 38 次迭代。这表明对于包含百万级交互信息的数据，本文的计算方法每天能进行几百次矩阵因子化计算。

除了做性能分析，本文还对 Hadoop 的并行运算能力做了测试。根据 Amdahl[25] 提出的衡量并行计算能力的公式

$$S_p = \frac{T_1}{T_p} \tag{4.4}$$

其中 S_p 代表并行计算加速倍数，T_1 代表仅使用 1 个运算节点处理目标数据集所用时间，T_p 代表使用 p 个运算节点处理目标数据集所用时间。下面我们从

MovieLens 数据集中随机选取某固定大小的子集进行 S_p 的计算。

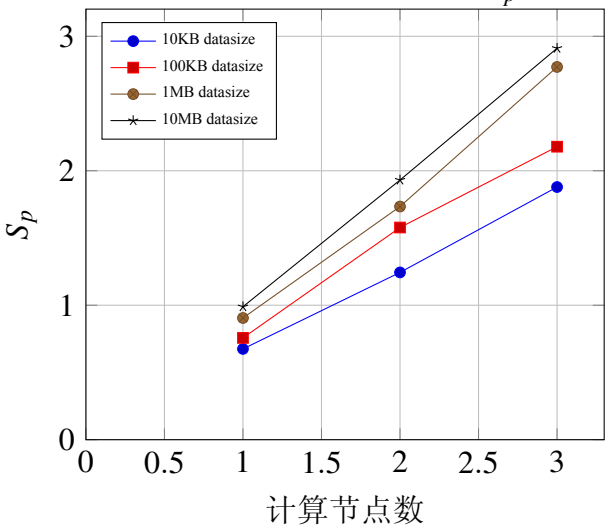

图 4.4 Hadoop 节点并行性能分析

图4.4描述了并行计算加速倍数与 Hadoop 节点数的关系。从图4.4可以发现，并行计算加速倍数随着 Hadoop 节点数的增加而增加，并且两者间是几乎线性的关系。另外我们还可以看出，数据集越大，对应更好的并行计算加速倍数曲线。由此我们可以得出结论：Hadoop 分布式集群在处理相对较大的数据集时，性能更优。

第五章 总结

本文研究的主要目标是构建用户模型，即得出代表用户模型的 User-Item 矩阵。根据这个目标，本文提出了基于大数据分析的推荐系统。

该基于大数据分析的推荐系统，主要由用户上网数据采集，网站语义分析，数据处理及存储，基于 Hadoop 的推荐算法等模块组成。本文在第二章对每个模块相关的算法做了简要介绍，其中包括语义分析算法、协同过滤推荐算法以及 Hadoop 分布式与 MapReduce 数据处理的概念。随后，第三章对各个关键模块的工作细节做了详细的描述，其中还简要地分析了基于 Hadoop 分布式系统的协同过滤推荐算法的计算复杂度。接着本文在第四章介绍了整体系统工作流程，同时进行了实证分析，得到了 Hadoop 分布式系统与协同过滤推荐算法的相关结论。

本文仅从学术的角度，探讨推荐系统实现的可行性及其中的关键技术。但是所提出的相关概念以及实践操作步骤能广泛应用于商业领域，为用户兴趣预测等商业运作系统提供了经验以及指南。

参考文献

[1] 杨清龙. 基于网络日志的互联网用户行为分析 [D]. 武汉: 华中科技大学, 2013.

[2] 史艳翠. 基于通信数据的上下文移动用户偏好动态获取方法研究 [D]. 北京: 北京邮电大学, 2013.

[3] 程辉. 网络用户偏好分析及话题趋势预测方法研究 [D]. 北京: 北京交通大学, 2013.

[4] 张欢. 网络用户偏好分析方法的研究 [D]. 北京: 北京交通大学, 2014.

[5] P. Resnick *et al*. "*GroupLens: an open architecture for collaborative filtering of netnews*" [C].CSCW'1994: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. New York, Jan., 1994: 175 ∼ 186.

[6] G. Linden, B. Smith and J. York. "*Amazon.com recommendations: item-to-item collaborative filtering*" [J]. *IEEE Internet Computing*, 2003, 7(1): 76 ∼ 80.

[7] H. Kondo, H. Nakatani and K. Hiromi. "*The Netflix recommender system: algorithms, business value, and innovation*" [J]. *Acm Transactions on Management Information Systems*, 2015, 6(4): 29 ∼ 47.

[8] J. Davidson *et al*. "*The YouTube video recommendation system*" [C].RecSys'2010: *Proceedings of the fourth ACM conference on Recommender systems*. Barcelona, Sept., 2010: 293 ∼ 296.

[9] G. Adomavicius and A. Tuzhilin. "*Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*" [J]. *Knowledge and Data Engineering, IEEE Transactions on*, 2005, 17(6): 734 ∼ 749.

[10] J. L. Herlocker, J. A. Konstan and J. Riedl. "*Explaining collaborative filtering recommendations*" [C].CSCW'2000: *Proceedings of the ACM conference on Computer supported cooperative work*. Philadelphia, Dec., 2000: 241 ∼ 250.

[11] J. Zhang and P. Pu. "*A recursive prediction algorithm for collaborative filtering recommender systems*" [C].RecSys'2007: *Proceedings of the ACM conference on Recommender systems*. Minneapolis, Oct., 2007: 57 ∼ 64.

[12] Y. Shi, M. Larson and A. Hanjalic. "*Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering*" [C].RecSys'2009: *Proceedings of the third ACM conference on Recommender systems*. New York, Oct., 2009: 125 ∼ 132.

[13] J. D. A. Ghemawat. "*MapReduce: simplified data processing on large clusters*" [C].OSDI'2004: *Proceedings of 6th Symposium on OSDI*. San Francisco, Dec., 2004: 147 ∼ 152.

[14] S. Ghemawat, H. Gobioff and S. T. Leung. "*The Google file system*" [C].SOSP'2003: *Acm Sigops Operating Systems Review*. New York, Oct., 2003: 29 ∼ 43.

[15] F. Chang *et al.* "*Bigtable: a distributed storage system for structured data*" [J]. *Acm Transactions on Computer Systems*, 2008, 26(2): 205 ∼ 218.
[16] T. Kiss and J. Strunk. "*Unsupervised multilingual sentence boundary detection*" [J]. *Computational Linguistics*, 2006, 32(4): 485 ∼ 525.
[17] K. Gimpel *et al.* "*Part-of-speech tagging for Twitter: annotation, features, and experiments*" [C].ACL'2011: *Proceedings of The 49th Annual Meeting of the Association for Computational Linguistics*. Barcelona, Sept., 2011: 42 ∼ 47.
[18] B. O. Owoputi *et al.* "*Improved part-of-speech tagging for online conversational text with word clusters*" [C].NAACL'2013: *Proceedings of North American Chapter of the ACL*. Atlanta, June, 2013: 483 ∼ 485.
[19] L. Derczynski *et al.* "*Twitter part-of-speech tagging for all: overcoming sparse and noisy data*" [C].RANLP'2013: *Processings of Recent Advances in Natural Language*. Hissar, Sept., 2013: 66 ∼ 69.
[20] 乔媛媛. 基于 *Hadoop* 的网络流量分析系统的研究与应用 [D]. 北京: 北京邮电大学, 2014.
[21] 延皓. 基于流量监测的网络用户行为分析 [D]. 北京: 北京邮电大学, 2011.
[22] 董超. 基于网络流量监测的移动互联网特征研究 [D]. 北京: 北京邮电大学, 2013.
[23] C. Kohlschütter, P. Fankhauser and W. Nejdl. "*Boilerplate detection using shallow text features*" [C].WSDM'2010: *Proceedings of International Conference on Web Search and Web Data Mining*. New York, Feb., 2010: 441 ∼ 450.
[24] H.-P. Zhang *et al.* "*Chinese lexical analysis using hierarchical hidden Markov model*" [C].SIGHAN'2003: *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*. Sapporo, July, 2003: 63 ∼ 70.
[25] G. M. Amdahl. "*Validity of the single processor approach to achieving large scale computing capabilities*" [C].AFIPS'1967: *Proceedings of AFIPS Spring Joint Computer Conf* . Atlantic City, Apr., 1967: 483.