基于 Hadoop 的电子商务个性化推荐算法——以电影推荐为例①

马瑞敏, 卞艺杰, 陈 超, 吴 慧

(河海大学 商学院, 南京 211100)

摘 要: 以电影为推荐对象, 选择了基于内容的推荐算法和基于协同过滤的推荐算法相结合的混合推荐算法, 并在 Hadoop 平台下进行 MapReduce 并行化; 通过线性组合的方式将两种推荐算法计算得到的电影相关性系数进行组合, 实现本文系统的计算电影相关性的混合推荐算法, 得到最终的电影相关性矩阵, 构建电影关系网; 最后对本文的提出的新型混合推荐算法工作模式进行了测试分析, 证明改进方案具有良好的推荐精度.

关键词: Hadoop; 推荐算法; MapReduce; 混合推荐算法

E-Commerce Personalized Recommendation Algorithm Design Based on Hadoop: Taking Move Recommendations as an Example

MA Rui-Min, BIAN Yi-Jie, CHEN Chao, WU Hui

(Business School, Hohai University, Nanjing 211100, China)

Abstract: This paper implements recommendation algorithm based on Item collaborative filtering recommendation algorithm and content-based parallelization, and uses the hybrid recommendation algorithm based on these two complementary recommendation algorithm for project-related calculations, and accesses network of the projects' relationships, namely the correlation coefficient matrix of each project. Then we do the relevant test analysis to the improved algorithm.

Key words: Hadoop; recommendation algorithm; MapReduce; hybrid recommendation algorithm

电子商务的迅猛发展为消费者带来了各种便利, 但是, 面对海量的商品信息, 用户很难从中搜寻到对自己有价值的部分, 从信息自身的角度来讲, 一些好的信息资源, 可能由于某种原因, 不能被用户熟知而埋没掉, 该现象被称为“信息超载”[1]. 为此, 互联网企业做了很多的尝试, 比如 Google、百度等, 通过网络爬虫技术搜集网页信息, 对其进行筛选、分析、提取、组织和处理, 为用户提供搜索服务. 这需要用户输入关键字, 反馈的信息自己基本都有一个了解, 不了解的可能很感兴趣的商品, 仍然不能展示到用户面前. 而且很多情况下, 用户根本不清楚自己的需求具体是什么, 更谈不上用关键字描述. 面对搜索引擎的各种弊端, 个性化推荐系统应运而生. 电子商务的个性化推荐系统注重用户的“个性化”需求和对未知商品的推荐, 根据用户的历史消费记录及用户的基本信息来学习用户的兴趣爱好, 为不同的用户推荐不同的商品. 随着用户体验的要求越来越高, 系统的响应速度必须被纳入 Web 应用的用户体验重要指标, 传统的电子商务推荐系统扩展性差和计算耗费大量时间是让人难以接受的. 虽然不断有各种高效的推荐算法提出, 但是面对海量数据的表现仍然差强人意, 通过改进算法的本身来提高计算效率的提升空间越来越小. 所以针对目前的个性化推荐系统的改进, 需要考虑平台存储、高性能和扩展性问题. Hadoop 平台的分布式文件系统 HDFS 和分布式批处理框架 MapReduce 不仅能够存储不断增长的海量数据, 也能对数据进行并行化处理, 提高算法性能的和系统的响应速度, 使个性化推荐系统更能适应海量数据的发展要求.

① 收稿时间:2014-09-03;收到修改稿时间:2014-10-08

1 以电影为例选择推荐算法

1.1 数据资源

推荐系统中的原始数据是决定采用何种推荐算法的基础, 电子商务推荐系统的初始数据资源主要分为三类: 用户信息、商品信息、用户评分信息.

(1)用户信息

是指用户的基本信息, 主要是通过用户注册时填写和登录后补充的个性信息来获得. 用户信息包括: 用户标识、登录密码、年龄、性别、职业、住址、电子邮件等.

(2)商品信息

电子商务个性化推荐系统需要商品信息形成商品关系网, 并以此作为用户可能感兴趣的商品的基础资源, 本文的推荐算法以电影作为推荐对象, 电影的基本信息主要包括: 电影编号、电影名、上映时间、电影风格、电影简介等.

(3)用户评价信息

电子商务个性化系统通过搜集用户对商品评价的数据, 将其作为推荐算法的重要输入内容. 用户对商品的评价类型是多样性的, 如模糊评价或直接评分的形式, 本文采用用户对电影评分的方法, 评价的主要信息包括: 用户标识、电影编号、评分、时间戳.

1.2 选择推荐算法

推荐算法是整个推荐系统最为核心的部分, 推荐算法的选择直接决定了推荐结果的质量和系统的性能, 推荐算法主要有以下几种:

(1)基于关联规则的推荐算法利用关联规则搜寻不同商品之间的相关性, 当用户浏览或购买了某个商品后, 主动地向其推荐相关性高的商品, 但是它存在系统不能实现动态更新的缺陷, 其中关联规则的发现是最关键且最耗时的过程, 随着规则数量的增多, 系统的管理也变得更加复杂[2].

(2)基于内容的推荐算法主要来源于信息过滤, 搜集用户浏览、购买的商品的数据, 对其进行结构化和内容特征的关键字提取, 建立用户模型, 并与商品描述文件的相比较, 将相似度最高的商品推荐给用户. 很容易理解, 但它要求用户需求和商品信息能以特征词的方式提取, 所以它不适合图片、音频、视频等多媒体数据, 并且语义的缺乏容易造成推荐准确度不高的问题.

(3)基于协同过滤的推荐通过最邻近(具有相似兴趣的用户)或相似商品(相似度最高的商品)的意见或评分来预测某用户对某商品的意见或评分[3]. 与前两种算法相比更加灵活, 只需要依靠用户对商品的评分, 推荐的商品的形式不受限制, 推荐的内容除了常规的文档, 也包括图片、音频、视频等, 而且可以推荐潜在的用户感兴趣的商品, 但是协同过滤算法也遇到了商品评分数据稀疏的问题.

(4)基于混合推荐算法的推荐系统, 是将以上的推荐技术相结合使用, 传统的混合推荐算法模式是在不同的情况下使用某一种相应的推荐技术, 以产生更好的推荐效果.

通过对推荐对象——电影的初始数据的分析, 以及对各类推荐算法的对比可以看出单独选用一种推荐算法并不能得出令人满意的推荐结果. 因此, 本文采用基于内容的推荐算法和基于协同过滤的推荐算法相结合的混合推荐算法, 这两种方法分别进行推荐, 将两种算法的推荐结果以线性组合的方式融合, 得到最终推荐的结果. 其中, 协同过滤推荐算法弥补了基于内容推荐算法不适合推荐电影这类多媒体数据的缺陷, 同时, 基于内容的推荐算法解决了协同过滤算法中用户对电影评分少的数据稀疏的问题.

2 基于Hadoop的混合推荐算法设计

Hadoop 主要由两大核心模块组成: 分布式系统 HDFS 和分布式批处理框架 MapReduce. 其中, HDFS 提供了大规模文件存储系统以及可靠的备份管理机制, 解决了海量数据的存储问题; 而 MapReduce 提供了分布式应用开发接口, 为解决算法效率问题提供了突破口, 充分利用框架中每台服务器的计算能力, 提高了算法的效率和扩展性. 本文主要讨论 MapReduce 并行化来提高算法的效率.

2.1 并行框架 MapReduce

MapReduce 整个过程分为数据映射(Map)和数据化简(Reduce)两个阶段, 一个任务被分解成为多个任务的“Map”实现, 另一个是分解后多任务处理的结果被汇总起来的“Reduce”实现, 得出最后的分析结果. 在这两个阶段中所有的数据都是以<key, value>键值对形式存储数据, MapReduce 具体的执行流程如下:

(1)启动阶段

MapReduce 作业以块为单位, 将输入数据通过 MapReduce 中 Splitter 分割成 M 块, 大小可以被程序员修

改参数自由设定, 然后拷贝程序被集群启动, 拷贝各个数据块.

(2)任务指派阶段

从这些拷贝程序中选出一个, 作为本次作业的 Master, 即主控制程序, 负责作业运行的调度和监控. Master 分配任务到剩下的拷贝程序, 即 Workers. 空闲的 Workers 被 Master 分配一个 Map 或者 Reduce 任务.

(3)Map 映射生成键值对阶段

分配到 Map 任务的 Worker 从输入分块中读取数据, 通过 MapReduce 提供的 RecordReader 提取出 key/value 对, 通过 map()函数的映射处理后, 生成键值对缓存在内存中.

(4)写入本地磁盘阶段

将缓存中的键值对通过分块函数分成 R 块, 周期性的写入本地磁盘, 并且将每块在磁盘上的位置信息传给 Master, 再由 Master 传递给负责 Reduce 任务的 Workers.

(5)远程读取排序阶段

负责 Reduce 任务的 worker 根据 Master 传送来的位置信息, 开始远程读取这些数据. Worker 在读取完所有的中间数据之后, 会将这些中间数据排序, 使得相同 key 值的 value 排在一起.

(6)写到输出文件阶段

排序工作结束后, worker 将同一个 key 对应的中间 value 传递给用户事先定义好的 reduce 函数, 将 value 值合并起来, 形成新的 key/value 对, 写到输出文件中.

2.2 基于内容的推荐算法 MapReduce 并行化

基于内容的推荐最常用的方法是: 词频率-逆向文档频率 TF-IDF(term frequency – Inverse document frequency)方法[4]. TF-IDF 方法定义如下:

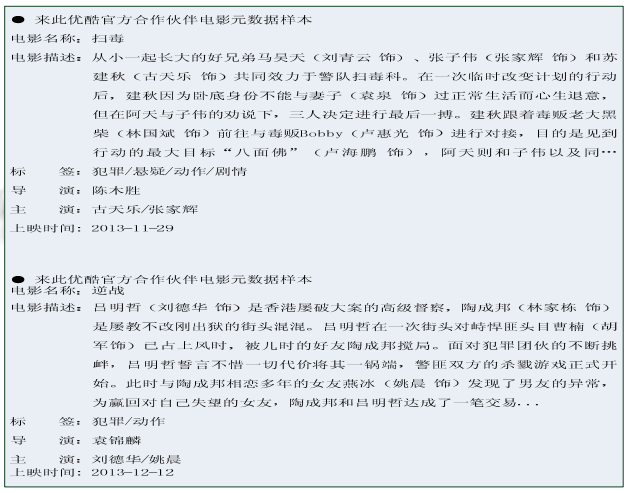
$$w_{ij}=TF_{ij}\times IDF_i=\frac{f_{ij}}{\max_z f_{zj}}\times\log\frac{N}{n_i} \tag{1}$$

其中, 关键词 k_i 在文档 d_j 中出现的词频为 TF_{ij}, 逆向文档频率为 IDF_i, 设有 N 个文档$(d_1,d_2,...,d_n)$, n_i 个文档中出现了关键词 k_i, f_{ij} 表示 k_i 在文档 d_j $(1\le j\le n)$中出现的次数, z 表示在文档 d_j 中出现的关键词, $\max_z f_{zj}$ 表示在文档 d_j 中所有关键词出现次数的最大值.

那么一个文档 d_j 可以用向量 $\overrightarrow{d_j}=(w_{1j},w_{2j},...,w_{mj})$ 表示, 其中 m 是关键词的个数. 用户 a 配置文件和商品 s 配置文件的相似度夹角余弦方法可以被用作计算用户 a 关键字说明文件和商品 s 关键字说明文件的相似度, 计算公式如下:

$$r_{as}=\cos(\overrightarrow{d_a},\overrightarrow{d_s})=\frac{\overrightarrow{d_a}\cdot\overrightarrow{d_s}}{\left\|\overrightarrow{d_a}\right\|_2\times\left\|\overrightarrow{d_s}\right\|_2}=\frac{\sum_{i=1}^{k}w_{ia}w_{is}}{\sqrt{\sum_{i=1}^{k}w_{ia}^2}\sqrt{\sum_{i=1}^{k}w_{is}^2}} \tag{2}$$

通常情况下, 我们在网络上看到的电影都有一些描述文本, 这就是本文所说的电影初始数据. 一个电影的初始数据如图 1 所示.



● 来此优酷官方合作伙伴电影元数据样本
电影名称: 扫毒
电影描述: 从小一起长大的好兄弟马昊天（刘青云 饰）、张子伟（张家辉 饰）和苏建秋（古天乐 饰）共同效力于警队扫毒科。在一次临时改变计划的行动后，建秋因为卧底身份不能与妻子（袁泉 饰）过正常生活而心生退意，但在阿天与子伟的劝说下，三人决定进行最后一搏。建秋跟着毒贩老大黑柴（林国斌 饰）前往与毒贩Bobby（卢惠光 饰）进行对接，目的是见到行动的最大目标“八面佛”（卢海鹏 饰），阿天则和子伟以及同…
标 签: 犯罪/悬疑/动作/剧情
导 演: 陈木胜
主 演: 古天乐/张家辉
上映时间: 2013-11-29

● 来此优酷官方合作伙伴电影元数据样本
电影名称: 逆战
电影描述: 吕明哲（刘德华 饰）是香港屡破大案的高级督察，陶成邦（林家栋 饰）是屡教不改刚出狱的街头混混。吕明哲在一次街头对峙捍匪头目曹楠（胡军饰）已占上风时，被儿时的好友陶成邦搅局。面对犯罪团伙的不断挑衅，吕明哲誓言不惜一切代价将其一锅端，警匪双方的杀戮游戏正式开始。此时与陶成邦相恋多年的女友燕冰（姚晨 饰）发现了男友的异常，为赢回对自己失望的女友，陶成邦和吕明哲达成了一笔交易...
标 签: 犯罪/动作
导 演: 袁锦麟
主 演: 刘德华/姚晨
上映时间: 2013-12-12

图 1 电影初始数据

本文主要针对中文电影的推荐, 英文先暂不考虑, 所以要先对系统中电影初始数据进行中文分词操作. 本文的基于内容的推荐系统采用中科院计算所的张华平、刘群索开发的中文分词系统 ICTCIAS[5]. 电影初始数据分词后, 可以用文档 $D_i(W_{i1},...,W_{ij},...,W_{im})$来表示第 i 个电影的 m 个关键字, W_{ij} 是关键字 K_j 的权重, 计算公式如下:

$$Wij=TF\text{-}IDF*Weight(POS) \tag{3}$$

其中 TF-IDF 就是由公式 1 得来的, 而 Weight(POS)是由表 1 决定的[6].

表 1 启发式的词性/类型权重列表

元数据类型或词性	权重
名词	0.8
动词	0.7
形容词	0.5
副词	0.2
标签	1
标题	0.7
描述	0.5

通过上面的方法, 将电影转换成以关键字的权重为因素的向量后, 利用公式 2 能够计算出基于内容的电影间相关性. 但是当电影数量庞大时, 而且该计算任务的

时间复杂度至少为 $O(N^2)$. 所以, 对基于内容的推荐算法 MapReduce 并行化处理, 步骤描述如下:

Step1: 本地任务: 生成电影对

输入: 电影向量列表:

$$\{(word_{i1},...,word_{in})\},...,\{(word_{n1},...,word_{nn})\}$$

对于每一个电影 $V_i(1\le i\le n-1)$, 输出电影 V_i 和其他电影的向量对

输出: 电影向量对列表:

$$\{(word_{i1},...,word_{in}),(word_{j1},...,word_{jn})\}...$$

Step2 Map-Reduce 任务: 计算电影间的相关性

输入: 电影向量对列表:

$$\{(word_{i1},\cdots,word_{in}),(word_{j1},\cdots,word_{jn})\}\cdots$$

Map: $(key:(V_i,V_j),$

$value:\{(word_{i1},\cdots word_{in}),(word_{j1},\cdots,word_{jn})\})$

计算两个电影向量的夹角余弦 $\#relevancy$

Reduce: idle opetation

输出: 视频间的相关性列表:

$$\{(V_i,V_j),\#relevancy\}$$

上面算法的最后得到的输出结果是一个 $N\times N$ 的矩阵 R_1, 其中 r_{ij} 表示电影 i 和电影 j 的相关性. 值得说明下的是这里的 $r_{ij}=r_{ji}$, 得到的矩阵 R_1 是个对称矩阵.

2.3 基于商品的协同过滤算法 MapReduce 并行化

基于商品的协同过滤推荐的总体思路是根据用户的评分数据建立商品间相似的模型[7]. 计算流程分为以下 3 个步骤:

1) 搜集数据并初始化, 形成用户-商品评分矩阵

用一个 m*n 阶矩阵 U(m*n)来表示用户-商品评分矩阵, 其中, n 表示商品个数, m 表示用户个数, R_{ij} 表示用户 i 对商品 j 的评分.

2) 计算商品相似度, 形成目标商品的最近邻集合

本文采用相关相似性的方法进行计算: $R_{c,i}$ 为用户 c 对商品 i 的评分, $\overline{R}_i$ 和 $\overline{R}_j$ 分别表示商品 i 和 j 的平均评分, U_{ij} 表示对商品 i 和 j 都评分过的用户集合, 则商品 i 和 j 之间的相似性可以通过皮尔森相关系数来计算:

$$sim(i,j)=\frac{\sum_{c\in U_{ij}}\left(R_{c,i}-\overline{R}_i\right)\left(R_{c,j}-\overline{R}_j\right)}{\sqrt{\sum_{c\in U_{ij}}\left(R_{c,i}-\overline{R}_i\right)^2}\sqrt{\sum_{c\in U_{ij}}\left(R_{c,j}-\overline{R}_j\right)^2}} \tag{4}$$

3) 产生评分预测, 给出推荐列表

设 N 表示目标商品 k 的邻居集合, 则用户 u 对商品 k 的预测评分 $P_{u,k}$ 为:

$$P_{u,k}=\overline{R}_k+\frac{\sum_{n\in N}sim(k,n)\times\left(R_{u,n}-\overline{R}_n\right)}{\sum_{n\in N}\left(\left|sim(k,n)\right|\right)} \tag{5}$$

根据 MapReduce 的编程模型规则, 输入输出都必须是键值对. 在计算电影相似度的过程中, 算法的输入是<null, (用户, 电影, 评分)>, 算法的输出是<(电影 1, 电影 2), 相似度>.

因为需要从用户对电影的评分数据得到<(电影 1, 电影 2), 相似度>, 所以需要得到每个用户对两个电影的评分数据, 即<(电影 1, 电影 2), (用户 1 对电影 1 的评分, 用户 1 对电影 2 的评分)>这样的用户-电影映射集合. 而将电影根据用户关联起来, 就要得到所有被该用户评分过的电影, 即<用户 1, (电影 1, 电影 2, 电影 3, …)>.

由以上的想法, 本文计算电影相似度的 MapReduce 作业需要 2 个 MapReduce 任务来完成. 首先, 用一个 MapRedcue 任务搜集各个用户对电影的评分, map 过程将数据转换成以用户编号为键的键值对, reduce 函数将相同用户编号的键值对进行合并. 详细见表 2 和表 3 的例子.

表 2　搜集用户对电影评分 map 阶段

Map 输入	Map 输出
<null, (用户 1, 电影 1, 评分)>	<用户 1, (电影 1, 评分)>
<null, (用户 1, 电影 2, 评分)>	<用户 1, (电影 2, 评分)>
<null, (用户 2, 电影 1, 评分)>	<用户 2, (电影 1, 评分)>
<null, (用户 2, 电影 3, 评分)>	<用户 2, (电影 3, 评分)>
<null, (用户 3, 电影 1, 评分)>	<用户 3, (电影 1, 评分)>
<null, (用户 3, 电影 2, 评分)>	<用户 3, (电影 2, 评分)>
<null, (用户 3, 电影 3, 评分)>	<用户 3, (电影 3, 评分)>

表 3　搜集用户对电影评分 reduce 阶段

Reduce 输入	Reduce 输出
<用户 1, (电影 1, 评分)>	<用户 1, ((电影 1, 评分), (电影 2, 评分))>
<用户 1, (电影 2, 评分)>	
<用户 2, (电影 1, 评分)>	<用户 2, ((电影 1, 评分), (电影 3, 评分))>
<用户 2, (电影 3, 评分)>	
<用户 3, (电影 1, 评分)>	<用户 3, ((电影 1, 评分), (电影 2, 评分), (电影 3, 评分))>
<用户 3, (电影 2, 评分)>	
<用户 3, (电影 3, 评分)>	

这个还不是电影之间的相似度, 所以还需要一次 MapReduce 任务, 将用户和各电影之间的键值对转换成电影与电影之间的键值对, 这样就可以计算电影之间的相似度. 第二次 MapReduce 任务的 map 阶段是获得各个电影两两之间同一用户的评分对比, reduce 阶段是计算两个电影的相似度, 计算公式 4. 计算电影之间相似度的 MapReduce 任务详细过程见表 4 和表 5.

表 4 计算电影两两之间相似度 map 阶段

Map 输入	Map 输出
<用户 1, ((电影 1, 评分), (电影 2, 评分))>	<(电影 1, 电影 2), (评分, 评分)>
<用户 2, ((电影 1, 评分), (电影 3, 评分))>	<(电影 1, 电影 3), (评分, 评分)>
<用户 3, ((电影 1, 评分), (电影 2, 评分), (电影 3, 评分))>	<(电影 1, 电影 2), (评分, 评分)>
	<(电影 1, 电影 3), (评分, 评分)>
	<(电影 2, 电影 3), (评分, 评分)>

表 5 计算电影两两之间相似度 reduce 阶段

Reduce 输入	Reduce 输出
<(电影 1, 电影 2), (评分, 评分)>	<(电影 1, 电影 2), 相似度>
<(电影 1, 电影 2), (评分, 评分)>	
<(电影 1, 电影 3), (评分, 评分)>	<(电影 1, 电影 3), 相似度>
<(电影 1, 电影 3), (评分, 评分)>	
<(电影 2, 电影 3), (评分, 评分)>	<(电影 2, 电影 3), 相似度>

通过上面的计算, 可以得到一个 $N\times N$ 的矩阵 R_2, 其中 r_{ij} 为电影 i 和电影 j 的相关性. 同样, 这个矩阵也是对称的.

2.4 混合推荐算法

在前两小节得到了两个相关性矩阵 R_1 和 R_2, 由于两个矩阵的尺度不一样, 必须先将 R_1 和 R_2 通过行归一化操作规划到同一尺度下, 再进行线性组合. 本文还是通过一个 MapReduce 任务完成矩阵的行归一化计算, 具体算法描述如下:

MapReduce 任务: 相关性矩阵的行归一化

输入: 电影对的相关性列表:

$\{(V_i,V_j),\#relevancy\}\ldots$

Map:

$(key:V_i,value:(V_j,\#relevancy))$

Reduce:

$(key:V_i,value:(V_m,\#relevancy_m),\cdots,(V_n,\#relevancy_n))$

$sum=\sum\#relevancy_k \qquad (m\le k\le n)$

对于 V_i 相关联的所有视频, 即 V_m 到 V_n

$\#relevancy_{ij}=\#relevancy_{ij}/sum$

输出 $key:(V_i,V_j),value:\#relevancy_{ij}$

输出: 经过行归一化后的电影相关性列表: $(V_i,V_j,\#relevancy_{ij})$

矩阵经过行归一化的规范后进行线性组合. 系统中依然是一个 MapReduce 并行化计算, 最终输出线性组合之后的电影相关性矩阵.

MapReduce 任务: 电影相关性矩阵的线性组合

输入: 视频对的相关性列表:

$\{(V_i,V_j,\#relevancy_{ij})\}\ldots$

Map: $(key:(V_i,V_j),value:\#relevancy_{ij})$

Reduce:

$(key:(V_i,V_j),value:(\#relevancy_{ij1},\#relevancy_{ij2}))$

获得电影 V_i 和 V_j 线性融合的相关性

$\#relevancy_{ij}=\partial\times\#relevancy_{ij1}+(1-\partial)\times\#relevancy_{ij2}$

输出: 线性组合之后的电影相关性列表: $(V_i,V_j,\#relevancy_{ij})$

至此, 电影关系网构建的数据准备工作已经结束, 最终的输出是一个 $N\times N$ 的矩阵 R, 其中 r_{ij} 为电影 i 对电影 j 的相关性系数.

3 个性化推荐服务的实现

3.1 电影关系网的构建

构建电影关系网最重要的任务就是确定关系网的边权重, 即关系网中两个关联电影的关系系数, 在 2.4 节混合推荐算法得到的矩阵 R 中, r_{ij} 即为电影 i 对电影 j 的相关性系数. 本文对关系网的构建也用 MapReduce 框架进行计算. 首先遍历电影的相关性矩阵, 然后进行排序筛选. 对电影进行排序筛选是由于不可能将所有与目标电影有关系的电影都纳进关系网中, 而且与目标电影相关性系数很小的电影, 推荐的意义也不大. 本文所构建的电影关系网, 顶点为电影, 边的权重为电影之间的相关性. 考虑上述原因, 为每个电影保留与之最相关的 K(通常为 25)个电影. 同时, 考虑电影相关性系数要有一定的推荐意义, 设定一个相关性阈值 T(通常为 $1/(K\times100)$), 权重比 T 小的边将被忽略.

电影关系网构建的具体算法描述如下:

MapRedcue 任务: 电影关系网的构建

输入: 电影对的相关性列表:

$\{(V_i,V_j),\#relevancy_{ij}\}\cdots$

Map: $(key:V_i,value:(V_j,\#relevancy_{ij}))$

Reduce:

$(key:V_i,value:(V_m,\#relevancy_m),\cdots(V_n,\#relevancy_n))$

$sum=\sum\#relevancy_k \qquad (m\le k\le n)$

对于 V_i 相关联的所有视频, 即 V_m 到 V_n

$\#relevancy_{ij}=\#relevancy_{ij}/sum$

If $\#relevancy_{ij}<10\times(1/N)$

Delete $\{(V_i,V_j),\#relevancy_{ij}\}$

输出权重 TOP K 的边

输出: 电影间的相关性列表:
$\{(V_i,V_j),\#relevancy_{ij}\}$

这样电影关系网构建工作完成, 需要对各个电影的相似度做一定的整理和搜集工作, 获得每个电影的相似度列表, 持久化的存储到文件系统中. 这个 MapReduce 任务描述如下:

MapReduce 任务: 搜集各个电影的相似度列表, 存储到 HDFS.

输入: 电影对的相关性列表:
$\{(V_i,V_j),\#relevancy_{ij}\}...$

Map: $(key:V_i,value:(V_j,\#relevanvy_{ij}))$

Reduce:
$(key:V_i,value:(V_m,\#relevancy_m),\cdots,(V_n,\#relevancy))$

输出: 每个电影与其他电影的相关性列表
$(V_i,(V_m,\#relevancy_m),\cdots,(V_n,\#relevancy))$

将上述各个电影的相似度列表, 存储到Hadoop的分布式文件系统中. 之所以做这一步 MapReduce 任务, 是为了方便利用公式 5 来计算目标用户对于没有看过的电影的预测评分.

3.2 推荐电影类表

在以上电影关系网的基础上, 我们就可以为目标用户实现个性化推荐服务. 一般用户刚登陆系统时, 用户的兴趣和分布式系统中存储的用户兴趣并没有明显的变化, 所以我们可以通过一个 MapReduce 任务, 事先将用户登录时的推荐电影的预测评分计算好, 按预测评分给推荐的电影排序, 取其中的 Top N 作为推荐结果. 具体的 MapReduce 任务描述如下:

MapReduce 任务: 计算推荐电影列表

输入: 各电影相似度列表
$\{(key:V_i,value:(V_m,\#relevancy_m),...,(V_n,\#relevancy))\}...$

通过公式 5 计算电影 V_i 的预测评分#prescore.

说明: 这里计算的时候, 会用到表 3 中 Reduce 输出的保存在临时存储空间的文件数据.

Map: $(key:UserID,value(V_i,\#prescore))$

Reduce:
$(key:UserID,value:(V_m,\ \#prescore_m),...,(V_n,\#prescore_n))$

输出: 每个用户各电影的预测评分列表:

$(UserID,(V_m,\#prescore_m),...,(V_n,\#prescore_n))$

计算出针对每个用户的推荐列表后, 将其保存到 HDFS 文件存储系统, 并且通过 Sqoop 工具将推荐列表拷贝到 MySQL 数据库中的预测评分表, 包含用户编号(UserID)、电影编号(MovieID)和预测评分(PreScore). 当用户登录时, 系统接收到推荐请求, 根据用户的ID, 在MySQL中取出用户的推荐电影列表, 根据预测评分排序, 取出 TOP N 的电影推荐给用户.

4 算法测试与分析

4.1 实验数据与环境

本文在计算电影相似度时, 不是使用一种算法, 而是将基于商品的协同过滤推荐算法和基于内容的推荐算法分别得到的结果进行线性融合, 为了检验改进后的算法是否提高了推荐质量, 本文选择美国明尼苏达州立大学 GroupLens 研究小组提供的公用数据集 MovieLens 作为实验源数据. MovieLens 是一种电影评分数据集, 因其数据集丰富, 并且数据属性清晰、准确、真实, 被广泛应用于研究推荐算法技术. 考虑到实验的目的是测试改进算法的精确度, 所以实验不需要在分布式的情况下进行.

实验选取 MovieLens 数据集中的 MovieLens-10M 作为实验的数据来源, 数据集合的特征包括用户数量、电影数量、评分数量以及数据的稀疏程度, 如表 6 所示.

表 6　MovieLens-10M 数据集特征

数据集	用户数量	电影数量	评分数量	数据稀疏程度
MovieLens-10M	71567	10681	10000054	1.31%

由于数据量过大, 本实验只随机选取 MovieLens-10M 数据集合中 1%左右的电影数量, 包含 943 名用户对 1682 部电影的 100000 个评价数据. 其中电影表中电影的简介(MovieIntroduction)是通过中文分词系统 ICTLAS 分词后的关键词列表.

4.2 评价指标

本文采用统计精度度量方法评价推荐系统质量. 统计精度度量方法中的平均绝对偏差 MAE(Mean Absolute Error)是预测用户评分与实际用户评分之间的差值绝对值的平均值, MAE 越小, 表明推荐的精度越好.

假设 p_i 是预测值, q_i 是实际评分, 共有 n 个预测商品, 平均绝对误差的计算公式如下:

$$MAE=\frac{\sum_{i=1}^{n}|p_i-q_i|}{n} \tag{6}$$

4.3 实验结果及分析

为了测试改进后的算法的推荐精度, 将其与目前主流的推荐算法做比较, 主要是基于内容的推荐算法、基于商品的协同过滤推荐算法、传统工作模式的混合推荐算法以及本文线性融合的混合推荐算法. 其中, 传统的电子商务推荐系统采用的混合推荐算法的工作模式是: 根

据计算对象状态不同采用不同的推荐算法, 具有计算对象相关性考虑因素不够全面的缺点. 本文的混合推荐算法将两种方法分别进行推荐, 将两种算法的推荐结果以线性组合的方式融合, 得到最终推荐的结果. 在计算用户对电影的预测评分时, 电影关系网中目标电影的关系电影个数影响着算法的 MAE, 所以在实验中, 将目标电影的关系电影数从 5 增加到 40, 间隔为 5, 查看不同的关系电影个数对预测精度的影响, 这样也为电影关系网筛选时, 保留相关电影个数做一个测试. 实验结果如表 7 所示.

表 7 本文改进算法统其他推荐算法的推荐误差比较

相关电影个数	基于内容的推荐算法	基于商品的协同过滤推荐算法	传统工作模式的混合推荐算法	本文改进的推荐算法
5	0.8264	0.7763	0.7623	0.7529
10	0.8139	0.7581	0.7207	0.7283
15	0.8125	0.7322	0.7354	0.7172
20	0.8077	0.7617	0.7366	0.7263
25	0.8151	0.7457	0.7291	0.7033
30	0.8010	0.7391	0.7088	0.6851
35	0.7591	0.7136	0.6871	0.6331
40	0.7441	0.6765	0.6491	0.6074

通过表 4-2 可以看出, 本文改进后的混合推荐算法有更高的推荐精度, 并且随着相关电影数量的增加, 精度也越好. 这是因为本文在计算电影相关度时, 综合考虑了电影本身的关联和用户电影的消费关系, 得到更准确的电影之间的相关性系数来进行评分预测的计算. 综上所述, 本文的改进推荐算法具有较高的推荐精度, 可以应用于电影商务的个性化推荐.

5 结论

本章首先分析了以电影推荐为例的个性化推荐系统存储资源, 根据存储资源的性质, 选择了基于电影内容的推荐算法和基于电影评分的协同过滤推荐算法; 然后提出了两种推荐算法基于 Hadoop 平台的 MapReduce 并行化方法, 并通过线性组合的方式将两种推荐算法计算得到的电影相关性系数进行组合, 实现本文系统的计算电影相关性的混合推荐算法, 得到最终的电影相关性矩阵, 并且通过条件筛选掉关系不密切的电影之间的关系系数, 获得电影关系网; 最后对本文的提出的新型混合推荐算法工作模式进行了测试分析, 证明改进方案具有良好的推荐精度.

参考文献

1 刘建国,周涛,汪秉宏.个性化推荐系统能够的研究进展.自然科学进展,2009,19(1):1–15.

2 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of itemsin large database. Proceedings of ACM SIGMOD, 1993: 207–216.

3 Eirinaki M, Vazirgiannis M, Kapogiannis D. Web path recommendations based on page ranking and Markov models. Proceedingsof the 7th Annual ACM International Workshop on Web Information and Data Management. Bremen, Germany,2005.

4 Salton G.. Automatic Text Processing. Addison–Wesley, 1989.

5 分词系统研究完整版(ICTCLAS)–百度文库.

6 蹇易.基于混合方法的个性化视频推荐技术研究与实现[学位论文].武汉:华中科技大学,2012.

7 邓爱林,朱扬勇,施伯乐.基于商品评分预测的协同过滤推荐算法.软件学报.2003,14(9):1621–1628.

8 Apache Hadoop. Welcome to ApacheTM HadoopR. http://hadoop.apache.org.

9 Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. US ENIX Association, 2004: 10–10.

10 吴发青,贺樑,夏薇薇等.一种基于用户兴趣局部相似性的推荐算法.计算机应用,2008,28(8):1981–1985.

11 张学胜.面向数据稀疏的协同过滤推荐算法研究[学位论文].合肥:中国科学技术大学,2011.

12 熊宇.协同过滤的电子商务个性化服务推荐系统的研究[学位论文].成都:电子科技大学,2013.

13 李施施.面向电子商务推荐系统的研究与应用[学位论文].长沙:湖南大学,2013.

14 王国霞,刘贺平.个性化推荐系统综述.计算机工程与应用,2012,48(7): 66–76.

15 邓雄杰.基于 Hadoop 的推荐系统的设计与实现[学位论文].广州:华南理工大学,2013.