基于 Apache Spark 的协同过滤系统设计与实现

陈 斌,李淑琴,曾星宇,帅凯严

(北京信息科技大学 计算机学院,北京 100101)

摘 要:随着 Web 技术的发展,互联网用户数量持续增长,快速为用户生成精确推荐变得愈加困难。提出了项目协同过滤算法,采用余弦相似性计算项目间相似度并用加权平均值的方法为用户生成推荐结果。在 Apache Spark 上构建该系统,对抓取到的某电商商品数据进行测试。实验结果表明,基于 Spark 的推荐系统可以显著提高推荐生成的速度和有效性。

关键词:Apache Spark;协同过滤;推荐系统

DOI:10.11907/rjdk.143871

中图分类号:TP319 **文献标识码**:A **文章编号**:1672-7800(2015)001-0097-03

0 引言

协同过滤推荐在信息过滤和信息系统中正迅速成为一项很受欢迎的技术。与传统的基于内容过滤进行推荐不同,协同过滤分析用户兴趣,在用户群中找到指定用户的相似(兴趣)用户,综合这些相似用户对某一信息的评价,形成系统对该用户的喜好程度预测。

Spark[1]是一个通用的并行计算框架,由 UCBerkeley 的 AMP 实验室开发,是 MapReduce 模型的实现之一,它提供的框架自动执行任务分解、发送、执行、归并、容错工作,免去了二次开发和定制专用的分布式调度系统。和 Hadoop 一样,建立在 HDFS 基础上,但它的 Job 中间输出和结果可保存在内存中,减少了硬盘 I/O 次数,因而可以有较高的速度,对迭代和多步骤运算有更好的支持能力。鉴于此,很多机器学习相关工作者都在研究和使用它。本文在学习 Spark 基础上,搭建了一个 Spark 平台,并在其上设计实现了一个协同过滤系统。

CMMI 评判的等级审阅评审结果草案,采取措施纠正、完善,直到客户满意为止。

(7)持续过程改进。根据评估报告、客户需求及软件交付后运行的实际情况,提出持续改进建议并制定相应计划。通过加强软件配置管理,严格控制软件更改,加强更改后的回归测试和二次评审,使软件质量缺陷得到系统解决。

5 结语

军事气象水文软件的研制应该牢固树立“质量第一”的观点。在信息化高速发展的今天,只有加强质量管理,才能保证军事气象水文软件质量可靠性,提高军事气象保障的精确度和效率。本模型将 CMMI 关键过程域的理念融入到军事气象水文软件开发常规流程中,通过软件开发过程管理,提升军事气象水文软件的质量及服务能力,进而提高武器装备的质量。

参考文献:

[1] WATTS S HUMPHREY. Managing the software process[J]. Addison Wesley,2002,19(4):58-63.

[2] 质量管理体系要求[Z]. GJB 9001B-2009.

[3] 郑人杰. 基于软件能力成熟度模型的软件过程改进[M]. 北京:清华大学出版社,2003.

[4] DENNIS M. AHERN, RICHARD TURNER, AARON CLOUSE. CMMI Distilled:a practical introduction to intergrated process improvement[J]. Pearson Education,2003,21:113-115.

[5] 万江平,孔学东,杨建梅. 集成能力成熟度模型(CMMI)的研究[J]. 计算机应用研究,2001,18(10):10-13.

[6] 林锐,彭国明. CMMI 和集成化软件研发管理[M]. 北京:电子工业出版社,2008.

(责任编辑:陈福时)

基金项目:北京市教委教育教学—本科生科研训练项目(PXM2014_014224_000079)

作者简介:陈斌(1993—),男,河南开封人,北京信息科技大学计算机学院学生,研究方向为数据挖掘;李淑琴(1963—),女,北京人,北京信息科技大学计算机学院教授,研究方向为人工智能。

1 协同过滤算法设计

协同过滤系统一般分为基于用户的协同过滤、基于项目的协同过滤和基于模型的协同过滤[2]。基于用户的协同过滤假设:对同样的项目,如果用户间的评分结果较为相似,则他们对其它项目的评分也比较相似。此假设弊端是运算量随用户量和项目数量的增长而急剧增长。基于项目的协同过滤系统则通过计算基本稳定的项目间相似性作出推荐,意味着用户的偏好由系统中的项目唯一确定,因而有更好的可扩展性。

在一个典型的协同过滤场景中,有 n 个项目 $I=\{i_1,i_2,\cdots,i_n\}$ 和 k 个用户 $U=\{u_1,u_2,\ldots,u_k\}$,令 M_{k*n} 为基本用户对各项目的偏好矩阵。其中,$M_{u,i}$ 表示用户 u 对项目 i 的偏好,如评分,其值为实数或空,空表示当前用户尚未对项目作出评价。推荐系统的任务就是根据目标用户 $u\in U$ 的评分历史,预测出用户最喜欢的项目。本文采用基于项目的协同过滤算法,使用余弦相似性计算项目间相似性,然后采用加权平均值的方法为用户生成推荐结果。

1.1 相似性度量方法

本文主要采用余弦相似性度量方法[2]计算用户间、项目间的相似性,这里以计算用户间相似性为例,其计算方法可表示为:

$$\mathrm{sim}(u_x,u_y)=\frac{\sum_{i\in C_{u_x,u_y}} r_{u_x,i}\,r_{u_y,i}}{\sqrt{\sum_{i\in C_{u_x,u_y}} r_{u_x,i}^{\,2}}\sqrt{\sum_{i\in C_{u_x,u_y}} r_{u_y,i}^{\,2}}} \tag{1}$$

其中,C_{u_x,u_y} 表示用户 u_x 和 u_y 共同评价过的项目 $i\in I$ 的集,$r_{u_x,i}$ 表示用户 u_x 对项目 i 的评分,$r_{u_y,i}$ 表示用户 u_y 对项目 i 的评分。相似性即两向量 u_x,u_y 的夹角余弦值,因为用户评分均为正数,故相似性取值范围为[0,1]。

1.2 加权平均值

在协同过滤算法中,计算出所有用户间的相似度后,就可以采用一种方法为给定的用户 $u\in U$,预测出用户对所有尚未评价项目的评分。这里介绍一种最为简单、常用并且效果良好的方法:加权求和,表示为:

$$p_{u_x,i}=\bar{r}_{u_x}+\frac{\sum_{u_y\in N_x}(r_{u_y,i}-\bar{r}_{u_y})*sim(u_x,u_y)}{\sum_{u_y\in N_x}sim(u_x,u_y)} \tag{2}$$

其中,N_{u_x} 表示用户 u_x 的邻居,$sim(u_x,u_y)$ 表示用户 u_x 和 u_y 的相似度。

2 推荐系统部署与设计

由于推荐系统的高度计算密集性,将推荐系统部署到单台机器上会有很大的限制,例如,CPU 运算速度、内存大小和硬盘 I/O 能力等,更为重要的是如果此计算机发生故障,将面临数据丢失的风险。使用分布式计算环境可以大大缓解此类问题,如可在分布式运算框架 Hadoop/MapReduce 下实现推荐系统[4-6]。Spark 是一个开源的 Hadoop/MapReduce 并行计算框架,本文在 Spark 平台上构建一个协同过滤系统。

2.1 软硬件配置

实验平台硬件采用 Intel Core i5 双核的普通 PC,网络带宽为 100Mbps,为提高虚拟机 I/O 速度,采用 1 条 4G 和 1 条 8G DRR1333 内存条,并搭载 128GSanDisk 固态硬盘。主机采用 Windows 8.1 x64 位系统,虚拟机采用开源的 Oracle VM VirtualBox 4.3,虚拟主机系统采用开源的 Linux/CentOS6.5,具体如表 1 所示。

表 1 配置详情

序号	软件名	版本号	说明
1	虚拟机	VirtualBox 4.3.12	开源,VirtualBox 官网下载
2	CentOS	CentOS 6.5	开源,CentOS 官网下载
3	OpenSSH	OpenSSH 5.3p1	开源,OpenSSH 官网下载
4	Java 套件	JDK 1.7.0_40	开源,Oracle 官网下载
6	Hadoop	Hadoop 2.2.0	开源,Apache 官网下载
7	Spark	Apache Spark 0.9.2	开源,Spark 官网下载

2.2 网络规划

在 VirtualBox 中建立 Host-Only 网络,虚拟主机规划如下:(master/worker,169.254.80.3);(Worker1,169.254.80.4);(Worker2,169.254.80.5)。

2.3 SSH 通信配置

SSH 通信协议(Secure Shell)[7]是集群中各个节点通信所采用的协议,主节点可以通过 SSH 启动或关闭集群中节点。在配置 SSH 之前,需要保证各节点能相互通信。在各节点的终端中通过 ssh-keygen 命令生成公私钥,将各节点的公钥内容写入 authorized_keys 文件中,并发布到各节点的~ ${USER}/.ssh 目录下,权限为 600。

2.4 Java 环境安装

从官网下载安装包文件,使用 rpm-i jdk.rpm 命令安装,之后编辑 ~ user/.bash_profile 文件,添加 JAVA_HOME 环境变量为 jdk 安装目录,添加 $JAVA_HOME/bin 到 Path 变量。

2.5 Spark 安装配置

首先需要配置 Hadoop。从官网下载合适版本的 Hadoop,安装后修改配置文件:conf/core-site.xml,conf/hdfs-site.xml,conf/yarn-site.xml 和 conf/mapred-site.xml,并添加对应的环境变量,具体参考官网对应版本配置介绍。然后从官网下载 Hadoop 版本编译过的 Spark 程序并安装。配置 Spark 最为简单的方式就是以独立方式部署。成功部署并启动后,Spark 会运行 master 服务器和 worker 服务器,注意:所有节点必须从 master 节点启动。然后将 worker 节点信息添加到 master 节点中的 conf/slaves 文件中,具体请参考官网文档。本文中配置如下:

SPARK_MEM=2g # 指定单个 worker 节点所使用的内存值

SPARK_JAVA_OPTS="-Dspark.storage.memo-

ryFraction＝0.4 -Xmx3g" # 第 1 个参数：可用来作为 cache 的内存比例，剩余内存用来满足任务运行内存空间的需要；第 2 个参数：设定 jvm 最大可用内存为 3G。

3　实验结果分析

3.1　性能评估

为了验证基于 Spark 分布式平台对协同过滤推荐引擎的执行性能，我们从单节点到多节点进行实验。实验数据由 10W 用户和 3900 个商品构成的 69W 条记录构成，使用随机抽样将数据分为测试数据和训练数据两部分，比例为 1∶9。采用 Top20 方法实验，最近邻居个数为 50 个。

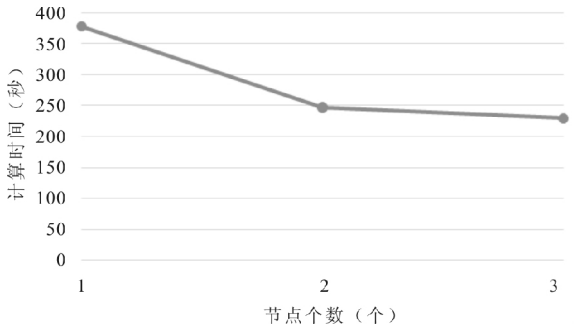

图 1　不同节点数推荐计算时间

图 1 显示执行时间开始随节点增加而减少，但是当节点增加为 3 个时，加速效果则不明显。我们认为是 CPU 瓶颈，因所采用 CPU 为双核酷睿 i5 型号；另外，根据 Amdahl 定律，并行化的程序所获得的加速比，与程序中可并行执行代码所占比例有直接关系，所以加速效果并不是线性增长的。

3.2　推荐质量评估

推荐质量评估一般采用平均绝对误差 MAE(mean absolute error)方法，MAE 越小则推荐质量越高[8]，MAE 计算公式可表示为：

$$MAE = \frac{\sum_{a \in U} |R_{a,i} - P_{a,i}|}{N} \tag{3}$$

其中，N 表示总推荐个数，$R_{a,i}$ 表示用户 a 对项目 i 的实际评分，$P_{a,i}$ 表示系统预测用户 a 对项目 i 的评分。

采用余弦相似度方法，通过修改不同个数的最近邻居，得出相应的 MAE，结果如图 2 所示。

从图 2 可以看出，MAE 整体随邻居个数增加而降低，但其下降速度逐渐降低，即过多增加最近邻居个数对推荐质量的提高贡献有限，反而消耗了更多的计算资源，因而需要在计算资源和推荐质量之间取一个折衷值。

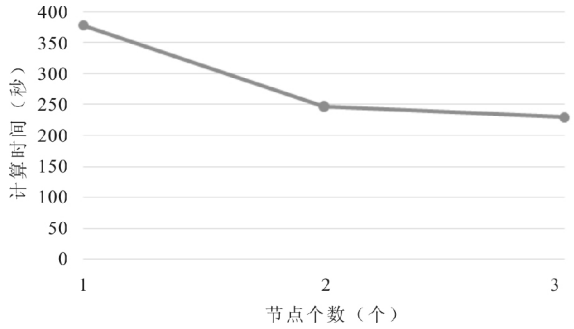

图 2　不同邻居个数的 MAE 值

4　结语

本文首先给出一种协同过滤推荐算法的设计和实现方法，然后介绍了 Apache Spark 框架以及本文实验环境搭建过程，并进行了算法实验。由于本文所采用设备性能的限制，未能进行较大数据集的实验测试。在接下来的工作中，应继续研究如何对 Spark 进行优化，并研究较为适合的应用场景。

参考文献：

[1] Apache Spark. [EB/OL]. https://spark.apache.org/2014.

[2] SARWAR B, KARYPIS G, KONSTAN J, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceedings of the 10th international conference on World Wide Web. ACM, 2001: 285-295.

[3] WebMagic. [EB/OL]. http://webmagic.io/2014.

[4] J DEAN, S GHEMAWAT. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[5] SCHELTER S, BODEN C, MARKL V. Scalable similarity-based neighborhood methods with mapreduce[C]//Proceedings of the sixth ACM conference on Recommender systems. ACM, 2012: 163-170.

[6] SecureShell[EB/OL]. http://zh.wikipedia.org/wiki/Secure_Shell, 2014.

[7] HERLOCKER J L, KONSTAN J A, TERVEEN L G, et al. Evaluating collaborative filtering recommender systems[J]. ACM Transactions on Information Systems (TOIS), 2004, 22(1): 5-53.

（责任编辑：杜能钢）