基于 MapReduce 的改进的 Apriori 算法及其应用研究

赵 月 任永功 刘 洋
(辽宁师范大学计算机与信息技术学院 大连 116029)

摘 要 随着移动通信和互联网技术的迅猛发展,如何高效地分析移动用户的需求并及时推送有用信息成为数据挖掘领域的热点之一。针对上述问题,提出一种基于云计算 Hadoop 平台的分布式关联规则 MRS-Apriori 算法。该方法在经典 Apriori 算法的基础上优化了数据库编码规则,增加了判断标记 Judgemark 来判断事务项是否频繁,提高了 MRS-Apriori 算法在连接时扫描数据库的效率。在编码的基础上,采用 Hadoop 平台下的 MapReduce 编程框架模型实现并行化处理,提高了迭代时连接步骤的效率,降低了大规模数据样本运算的时间开销。实验结果表明,改进的 MRS-Apriori 算法可以有效地减少运算时间,在处理大规模数据集上具有较高的准确性。

关键词 编码规则,关联规则,频繁项集,MapReduce 框架
中图法分类号 TP39 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.043

Improved Apriori Algorithm and Its Application Based on MapReduce

ZHAO Yue REN Yong-gong LIU Yang
(School of Computer and Information Technology, Liaoning Normal University, Dalian 116029, China)

Abstract With the rapid development of mobile communications and Internet technology, it becomes one of the hot issues in the field of data mining that how to analyze the requirements of mobile users efficiently and send useful informations in time. In order to recommend the analysis result to users efficiently and timely, a mining method named MRS-Apriori algorithm based on MapReduce was proposed. This method defines a kind of coding rule to optimize database based on classical Apriori algorithm. A judging mark named Judgemark is added to database to decide whether the transaction database is frequent. This mechanism improves the efficiency of MRS-Apriroi algorithm in connecting database to scan database efficiently. On the basis of encoding rules, the MRS-Apriroi algorithm uses MapReduce programming framework model under Hadoop to achieve parallel processing. It improves the performance of iteration when connecting process and reduces the time in dealing with large-scale data. The experiment results show that MRS-Apriroi algorithm can effectively reduce time and have high accuracy in handling large data sets.

Keywords Coding rules, Association rules, Frequent itemsets, MapReduce framework

1 引言

随着移动通信技术和互联网技术的融合发展,新兴移动设备和服务也相应产生[1]。人们越来越多地通过移动应用获取生活娱乐、导航、在线社交等各种服务[2]。人们在享受种类繁多的移动应用给生活带来便利的同时,也面临着海量无关信息困扰的问题。如何根据访问习惯高效地挖掘移动用户潜在的兴趣点是亟待解决的热点问题。现有技术侧重于使用关联规则(Association Rules)挖掘方法解决这一问题。经典的关联规则算法 Apriori[3]和 AprioriTid[4]可以通过用户访问页面的事务数据库,发现被频繁访问的网页项之间的关联规则关系。但是,在移动应用数据量庞大的现代社会中,Apriori 算法有着明显的缺点:

1)频繁项连接步骤产生大量非频繁项集;

2)多次扫描全局事务数据库极大地增加了时间开销。

为了弥补以上缺点,以更高效、快捷地进行关联规则挖掘,许多改进的 Apriori 算法[5,10]被提出。Benjamin 等人[5]提出一种可扩展的 pcApriori 算法,即通过改进生产者-消费者处理方案,在加工和销售的过程中划分数据给可用线程,并将其扩展到多处理器大型数据集上。郭建等人[6]提出 MapReduce 模型与编码操作相结合的分布式关联规则挖掘算法。罗丹等人[7]提出基于压缩矩阵的 Apriori 算法。Wang 等人[8]提出基于布尔矩阵的改进的 Apriori 算法。Ming-Yen 等人[9]提出 3 种基于 MapReduce 的改进的 Aprioir 算法。Enrique 等人[10]提出两步改进算法技术来剪切非频繁候选项集。唐家维等人[11]根据现代 GPU 大规模并行化结构的单结构多数据特点来改进 Apriori 算法。刘瑞阳等人[12]将逻辑规则过滤方法引入频繁项集挖掘。上述算法或改进频繁项集挖

到稿日期:2016-03-22 返修日期:2016-05-15 本文受国家自然科学基金项目(F020806),辽宁省高等学校优秀人才支持计划项目(LR2015033),辽宁省科技计划项目(2013405003),大连市科技计划项目(2013A16GX116)资助。

赵 月(1990—),女,硕士生,主要研究方向为数据挖掘;任永功(1972—),男,博士,教授,主要研究方向为数据库技术、数据挖掘、智能信息计算等,E-mail:renyonggong@gmail.com;刘 洋(1991—),女,硕士生,主要研究方向为数据挖掘。

掘方法或引进并行结构提升效率，依然存在时间消耗大、不能同时解决 Apriori 算法的全部缺点的问题。

本文提出的 MRS-Apriori 算法定义了数据库编码规则，增加了判断事务项是否频繁的标记（JudgeMark），弥补了 Apriori 算法多次扫描全局数据库的缺点。同时，基于MapReduce编程框架模型能够快速生成候选项集，解决了连接步骤中消耗时间过多的问题，提高了算法效率，可以实现具有巨大实际数据量的移动应用网页推荐服务。

2　相关知识

2.1　移动应用网页推荐形式化描述

基于移动应用网页推荐的关联规则问题可以表述为：设项目（Item）集合 $I=\{i_1,i_2,i_3,\cdots,i_n\}$ 是移动应用中所有被访问的页面的集合，$U=\{u_1,u_2,u_3,\cdots,u_m\}$ 是一个用户访问移动应用页面的事务集合（URL Transaction），其中 u_i 是一个项目集合且满足 $u_i\subseteq I$。

定义 1（支持度，Support）　页面项 URL_a，URL_b 同时发生的概率称为网页关联规则的支持度，即

$$Support(URL_a\Rightarrow URL_b)=P(URL_a\cup URL_b)=\frac{Support\_count(URL_a\cup URL_b)}{Total\_count(URL_a)} \tag{1}$$

其中，$Support\_count(URL_a\cup URL_b)$ 为 URL_a 和 URL_b 同时发生的事务个数，$Total\_count(URL_a)$ 是所有事务个数。

定义 2（置信度，Confidence）　若页面项集 URL_a 发生，则页面项集 URL_b 也发生的概率称为网页关联规则的置信度，即

$$Confidence(URL_a\Rightarrow URL_b)=P(URL_b|URL_a)=\frac{Support\_count(URL_a\cup URL_b)}{Support\_count(URL_a)} \tag{2}$$

移动应用网页推荐的关联规则表述形式为：$URL_a\Rightarrow URL_b$，其中 URL_a 和 URL_b 分别为规则的前项和后项，$URL_a\subset I$，$URL_b\subset I$ 且 $URL_a\cap URL_b=\varnothing$。规则满足的支持度和置信度不低于最小支持度（mini_sup）和最小置信度（mini_conf）[13]。

定义 3（网页数据库 D 编码规则）　以项集 $I=\{i_1,i_2,i_3,\cdots,i_n\}$ 为列，以事务 $U=\{u_1,u_2,u_3,\cdots,u_m\}$ 为行对网页事务数据库编码，令 $f:D\rightarrow a_{ij}$，若 $I_n\in U_m$，则 a_{ij} 赋值为 1，否则赋值 0。编码形式如表 1 所列。

表 1　网页事务数据库编码形式

U_{id}	I_1	I_2	…	I_n
U_1	a_{11}	a_{12}	…	a_{1n}
U_2	a_{21}	a_{22}	…	a_{2n}
⋮	⋮	⋮	⋮	⋮
U_m	a_{m1}	a_{m2}	…	a_{mn}

性质 1　任意频繁集的子集都是频繁集，任意非频繁集的超集都是非频繁的[12]。

性质 2　事务集合中项数若少于 K，则无法产生频繁 K 项集[14]。

2.2　MapReduce 编程模型

MapReduce 模型[15]是一种支持并行程序化的编程模型，可以实现海量数据的分布式并行计算，具有较好的可扩展性和容错能力。MapReduce 将待处理的较大规模的数据分区到若干规模较小的数据块，然后通过函数式编程语言来建立映射（Map）和归约（Reduce）。每个 Map 任务从文件中读取一个数据块，并且输出 Map 函数指定的〈*key*，*value*〉键值对列表，同时创建分区并根据键值来确定记录存放的分区。每个 Reduce 任务接收 Map 端的〈*key*，*value*〉后，对相同 *key* 下的所有 *value* 进行运算处理，并将组后的键值作为结果输出。MapReduce 的任务执行过程如图 1 所示。

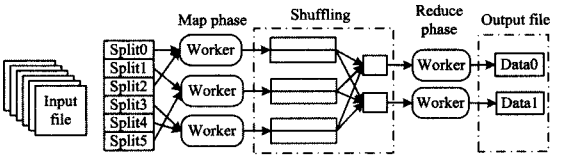

图 1　MapReduce 的任务处理流程

3　基于 MapReduce 的移动应用网页推荐 MRS-Apriori 算法

3.1　移动应用网页推荐原理

如何高效地发现频繁项集是移动应用网页智能推荐应用关联规则挖掘的主要任务。移动应用网页智能推荐包括样本数据采集、数据预处理、构建推荐模型和智能推荐 4 个过程，如图 2 所示。

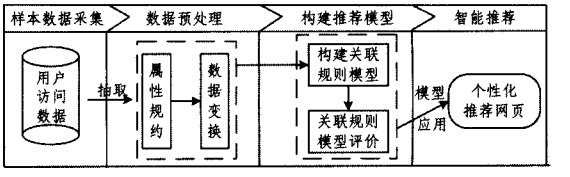

图 2　移动应用网页智能推荐流程

3.2　数据预处理

通过移动应用后台抽取用户访问的详细数据，记录形成数据样本，数据包括用户 ID、访问页面、访问时间、来源网站、页面标题和关键词等属性，即

〈*NO.*〉〈*id*〉〈*accesspage*〉〈*time*〉〈*sourcepage*〉〈*title*〉〈*keywords*〉

在进行移动应用网页关联规则挖掘之前，为了提高数据挖掘的效率，先要对数据进行预处理。将与用户网页推荐建模不相关的属性归约掉，包括访问时间、来源网站、页面标签、关键词等属性。同时，将单次访问事件整合划分。数据预处理算法如算法 1 所示。

算法 1　数据预处理

输入：移动应用后台抽取的用户访问数据
输出：清理整合后的规范化数据文件
1. delete irrelevant attributes
2. for i=1 to sizefile
3. if firstID == secondID
4. Join(firstID. accesspage)
5. else end if
6. firstID == secondID
7. end for

3.3　构建关联规则推荐模型

基于 MapReduce 编程模型和编码优化的改进的 MRS-Apriori 算法步骤如下。

步骤 1　产生频繁 1 项集。将预处理后的数据导入到

Hbase 中，对数据进行分割，传输到不同映射器上。在 Hadoop 集群框架下进行映射和归约操作，由于频繁 1 项集是简单的项目计数，由 Map 操作生成，Reduce 操作将其统计生成频繁 1 项集，如图 3 所示。基于 MapReduce 并行化模型产生频繁 1 项集的算法如算法 2 所示。

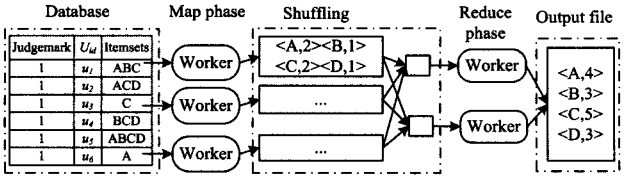

图 3 MapReduce 处理生成频繁 1 项集

算法 2 生成频繁 1 项集

输入：事务集〈U_{id}，list〉

输出：频繁 1 项集

1. Map (key，value)
2. for $u_i \in D$
3. for $i \in u_i$
4. output (i，1)；
5. end
6. end
7. Reduce(key=item，value=count)
8. for key≠∅
9. for value in valuelist
10. key. count+=value；
11. end
12. if (key. count≥min_sup count)
13. output (key，key. count)；
14. end

步骤 2 事务数据项编码。为了提高关联规则挖掘的效率，减少扫描数据库的开销，根据 2.1 节的定义和性质，对步骤 1 中生成的频繁 1 项集〈U_{id}，*list*〉添加删除标记(Judgemark)并将初始值设为 1，如表 2 所列。

表 2 数据库添加删除标记

Judgemark	U_{id}	A	B	C	D
1	u_1	1	1	1	0
1	u_2	1	0	1	1
1	u_3	0	0	1	0
1	u_4	0	1	1	1
1	u_5	1	1	1	1
1	u_6	1	0	0	0

根据频繁 K 项集将 list 中不满足支持度计数的项删除，同时，若只有 U_m 的 $k-1$ 项是频繁的(性质 2)或任意 k 项是频繁的并且任意 $k+1$ 项是非频繁的(性质 1)，将 Judgemark 设为 0，如表 3 所列。下一次扫描数据库时，根据 Judgemark 值只需扫描值为真的 U_{id}，从而提高了算法的时间效率。Judgemark 标记事务项算法如算法 3 所示。

表 3 Judgemark 筛选频繁事务项

Judgemark	U_{id}	A	B	C	D
1	u_1	1	1	1	0
1	u_2	1	0	1	1
0	u_3	0	0	1	0
1	u_4	0	1	1	1
1	u_5	1	1	1	1
0	u_6	1	0	0	0

算法 3 Judgemark 标记事务项

1. num=getCount ()
2. if (num≥mini_sup)
3. Codeitem()；
4. else
5. Deleteitem()；
6. for(i=1；i≤m；i++)
7. for(j=1；j≤n，j++)
9. if (前 k 项均为非频繁 or 任意 k 项是频繁的且任意 k+1 项是非频繁的)
10. Judgemark=0；

步骤 3 产生频繁 K 项集。在 Hadoop 集群框架下，每个映射器读取频繁 1 项集 L_1 生成候选 2 项集 C_2，对候选 2 项集进行归约操作生成频繁 2 项集。然后迭代上述过程，实现频繁 K 项集的挖掘。图 4 示出了生成频繁 2 项集的过程。频繁 K 项集生成算法如算法 4 所示。

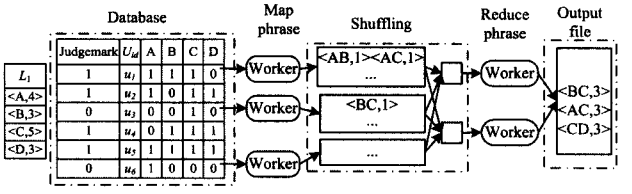

图 4 MapReduce 处理生成频繁 2 项集

算法 4 生成频繁 K 项集

输入：编码完成后的事务集和频繁 k−1 项集 L_{k-1}(k≥2)

输出：频繁 k 项集 L_k

1. Map(key，value)
2. read L_{k-1}
3. C_k=candidate(L_{k-1})
4. for $u_i \in D$
5. C_t=subset(C_k，u_i)
6. for $c \in C_t$
7. output〈c，1〉；
8. end
9. end
10. Reduce(key=item，value=count)
11. for key≠∅
12. for value in valuelist
13. key. count +=value；
14. end
15. if key. count≥min_sup count
16. output〈key，key. count〉；
17. end

步骤 4 循环步骤 3 直到所有频繁项集均生成，根据频繁项集得到强关联规则，如算法 5 所示。

算法 5 MRS-Apriori

1. 运行算法 1//规范数据
2. 运行算法 2//生成频繁 1 项集
3. 调用算法 3//对数据编码
4. for(k=2；L_{k-1}≠∅；k++)
5. 算法 4//生成频繁 k 项集
6. end

4　实验及结果分析

4.1　实验环境

本次实验在 Hadoop 分布式文件系统下进行，集群系统由一个主服务器 NameNode 和 3 个 DataNode 组成。在每个节点上配置安装 hadoop-1.1.2，sun-JDK，openssh，并给各个节点分配相应的 IP 地址。配置完成后在 hadoop01 节点端口监控 NameNode 的运行情况，如图 5 所示。

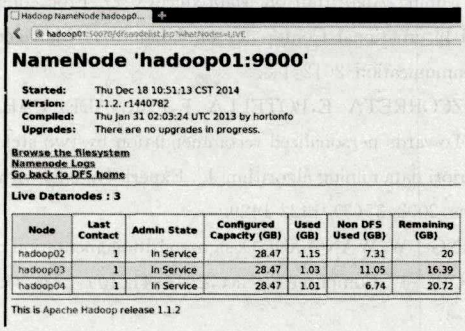

图 5　配置 Hadoop 各节点的运行情况

4.2　结果分析

本文样本数据采集自北京某健康科技有限公司出品的一款移动应用，选取 2015 年第 2－3 季度用户实时访问采集的数据，设置最小支持度为 1%，最小置信度为 70%，在 4 个节点的 Hadoop 集群上进行测试。为了证明算法在实践上的高效性和准确性，将 MRS-Apriori 算法与 MCM-Apriori 算法[6]和并行 Apriori 算法进行对比，分别在不同数量的集群点上进行实验，取数据量分别为 0.8GB，1.6GB，2.4GB 和 3.2GB 时的运行时间制成如表 4 所列的运行时间表。

表 4　3 种算法的运行时间表

算法	数据/GB	1 节点/s	2 节点/s	3 节点/s	4 节点/s
并行 Apriori	0.8	612.1	361.4	280.5	209.5
	1.6	891.8	496.5	359.8	289.6
	2.4	1474.9	784.5	556.4	406.1
	3.2	1674.3	876.1	602.7	453.2
MCM-Apriori	0.8	515.4	302.1	223.8	163.4
	1.6	812.5	452.3	330.3	248.6
	2.4	1390.6	739.7	523.9	373.8
	3.2	1504.5	783.6	555.1	406.7
MRS-Apriori	0.8	491.6	284.2	211.1	154.4
	1.6	820.9	431.9	305.7	219.6
	2.4	1260.7	670.2	474.6	328.5
	3.2	1346.1	701.5	480.7	349.1

本文综合多次实验结果后取平均值，以减小实验误差。为了直观地理解，选取表中 4 节点的对比情况进行分析，对比情况如图 6 所示。

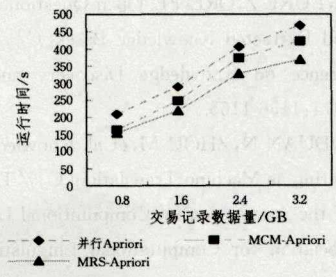

图 6　3 种算法在数据记录下的时间对比

根据图 6 的实验结果显示，本文算法在不同的交易记录数据量的情况下的运行时间明显优于一般并行 Apriori 算法和 MCM-Apriori 算法；MRS-Apriori 算法在扫描事务数据库时使用标记项判断是否扫描，大大减少了扫描的时间消耗，并且随着数据量增大其效果越明显，可以扩展到数据量庞大的移动应用网页智能推荐服务中。

图 7 示出了 MRS-Apriori 算法和 MCM-Apriori 算法生成 K-频繁项集的时间对比。MRS-Apriori 算法在时间上明显优于 MCM-Apriori 方法，这是由于 MRS-Apriori 算法在扫描数据库时使用一种通过标记项标记非频繁事务项的方法，显著减少了频繁项集的生成时间，可以很好地扩展到数据量庞大的应用中。

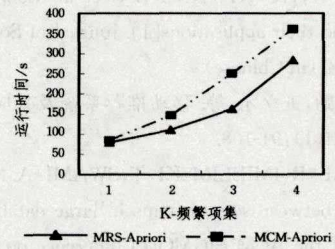

图 7　两种算法生成 K-频繁项集的时间对比

同时，在交易数据量相同的情况下，以数据量 3.2GB 为例，分别对集群点个数进行实验，结果如图 8 所示。

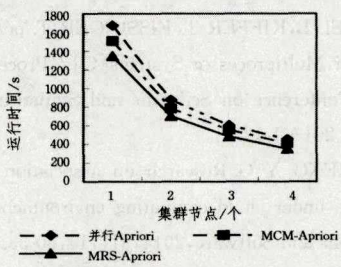

图 8　集群节点个数与运行时间的关系

由图 8 可知，算法优势在于当集群点达到一定数目时运行时间趋于稳定，可以考虑将这一结论应用到实际中，找到最优的集群点个数，实现资源的有效利用。

3 种算法的对比实验结果表明，基于 MapReduce 模型和编码改进的 MRS-Apriori 算法应用在移动应用数据上的时间效率和资源利用率均更高，并且随着 URL 事务集的增加，算法的优势越明显。

结束语　本文在基于 MapReduce 编程模型的基础上改进 Apriori 算法，引入判断频繁事务项标记的数据库编码策略，进而提出 MRS-Apriori 算法。针对传统 Apriori 算法在生成频繁项目集时必须多次扫描数据库的问题，首先提出一种优化数据库编码规则的方法，在编码的同时增加一种判断标记(Judgemark)，该标记用来判断事务项在下次迭代中是否被扫描并计数，提高了连接数据库的效率。然后，针对经典 Apriori 算法大量生成候选项集的时间消耗问题，采用 Hadoop 平台下的 MapReduce 编程框架模型实现并行化处理，提高了迭代时连接步骤的效率，降低了大规模数据样本运算的时间开销。在 Hadoop 集群环境下进行了实验与测试，结

果表明，该算法能高效地产生频繁项集，解决了 Apriori 算法产生大量候选项集和多次扫描全局事务数据库所产生的时间开销的问题。理论分析和对比实验表明，本文算法是有效可行的，相比传统的关联规则挖掘算法，本方法可以推广到数据规模庞大的移动应用网页推荐中。

参 考 文 献

[1] HUANG Y B,CHEN M Y. Architecture Characteristics and Analysis of Mobile Device Applications[J]. Chinese Journal of Computers,2015,38(2):386-396. (in Chinese)
黄永兵，陈明宇. 移动设备应用程序的体系结构特征分析[J]. 计算机学报，2015，38(2)：386-396.

[2] MENG X W,HU X,WANG L C,et al. Mobile recommender systems and their applications[J]. Journal of Software,2013,24(1):91-108. (in Chinese)
孟祥武，胡勋，王立才，等. 移动推荐系统及其应用[J]. 软件学报，2013，24(1)：91-108.

[3] AGRAWAL R,IMIELIMSKI T,SWAMI A. Mining Association Rules between sets of items in large databases[C]// Proceedings of the ACM SIGMOD Conference on Management of Data. Washington DC,1993:207-216.

[4] AGRAWA A,SRIKANT R. Fast algorithms for mining association rules[C]// Proceedings of the VLDB International Conference. 1994:487-499.

[5] SCHLEGEL B,KIEFER T,KiSSINGER T. pcApriori:Scalable Apriori for Multiprocessor Systems[C]// Proceedings of International Conference on Scientific and Statistical Database Management. 2013:1-12.

[6] GUO J,RENG Y G. Research on association rule mining in Book sales under cloud computing environment[J]. Computer Applications and Software,2014,31(11):50-53. (in Chinese)
郭健，任永功. 云计算环境下的关联规则挖掘在图书销售中的研究[J]. 计算机应用与软件，2014，31(11)：50-53.

[7] LUO D,LI T S. Research on Improved Apriori Algorithm Based on Compressed Matrix[J]. Computer Science,2013,40(12):75-80. (in Chinese)
罗丹，李陶深. 一种基于压缩矩阵的 Apriori 算法改进研究[J]. 计算机科学，2013，40(12)：75-80.

[8] WANG B L,SHEN Y G. Improvement of Apriori algorithm based on boolean matrix[J]. Adwanced Materials Research,2011,159:144-148.

[9] LIN M Y,LEE P Y,HSUEH S C. Apriori-based Frequent Itemset Mining Algorithm on Mapreduce[C]// Proceedings of the 2nd International Conference on Ubiquitous Management and Communication. 2012:1-8.

[10] LAZCORRETA E,BOTELLA F,FERNÁDEZ-CABALLERO A. Towards personalized recommendation by two-step modified Apriori data mining algorithm[J]. Expert Systems with Applications,2008,35(3):1422-1429.

[11] TANG J W,WANG X F. Design and Implementation of Apriori on GPU[J]. Computer Science,2014,41(10):238-243. (in Chinese)
唐家维，王晓峰. 基于 GPU 的并行化 Apriori 算法的设计与实现[J]. 计算机科学，2014，41(10)：238-243.

[12] LIU D Y,FENG J,LI X F. Logic-based Frequent Sequential Pattern Mining Algorithm[J]. Computer Science,2015,42(5):260-264. (in Chinese)
刘端阳，冯建，李晓粉. 一种基于逻辑的频繁序列模式挖掘算法[J]. 计算机科学，2015，42(5)：260-264.

[13] 韩家炜，等. 数据挖掘概念与技术(第 3 版)[M]. 范明，等译. 北京：机械工业出版社，2012：158-162.

[14] OLIVEIRA S R M,ZAIANE O R. A unified framework for protecting sensitive association rules in business collaboration [J]. International Journal of Business Intelligence and Data Mining,2006,1(3):247-287.

[15] JEFFREY D,SANJAY G. Mapreduce:Simplified Data Processing on Large Clusters[J]. Proceedings of the Sixth Symposium on Operating System Design and Implementation,2004,51(1):107-113.

(上接第 221 页)

[14] FERRUCCI D,BROWN E,CHU-CARROLL J,et al. Building Watson:an Overview of The Deep QA Project[J]. AI Magazine,2010,31(3):59-79.

[15] UNGER C,CIMIANO P. Pythia:Compositional Meaning Construction for Ontology-based Question Answering on the Semantic Web[C]// International Conference on Natural Language Processing and Information Systems. Salford:Springer Berlin Heidelberg,2011:153-160.

[16] BERANT J,CHOU A,ROY F,et al. Semantic Parsing on Freebase from Question-Answer Pairs[C]// The 2013 Conference on Empirical Methods on Natural Language Processing. Seattle:Association for Computational Linguistics,2013:1533-1544.

[17] BERANT J,LIANG P. Semantic Parsing via Paraphrasing[C]// The 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore:Association for Computational Linguistics,2014:479-485.

[18] BORDES A,CHOPRA S,WESTON J. Question Answering with Subgraph Embeddings[C]// The 2014 Conference on Empirical Methods on Natural Language Processing. Stroudsburg:Association for Computational Linguistics. 2014:1535-1545.

[19] YAO X,DURME B. Information Extraction over Structured Data:Question Answering with Freebase[C]// The 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore:Association for Computational Linguistics. 2014:753-770.

[20] FADER A,LUKE Z,OREN E. Open Question Answering Over Curated and Extracted Knowledge Bases[C]// Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD). 2014:1156-1165.

[21] BAO J W,DUAN N,ZHOU M,et al. Knowledge-Based Question Answering as Machine Translation[C]// The 52nd Annual Meeting of the Association for Computational Linguistics,Baltimore:Association for Computational Linguistics. 2014:1272-1294.