Journal of Computer Applications
计算机应用,2013,33(1):186-188
ISSN 1001-9081
CODEN JYIIDU
2013-01-01
http://www.joca.cn

文章编号:1001-9081(2013)01-0186-03
doi:10.3724/SP.J.1087.2013.00186

基于文本聚类与分布式 Lucene 的知识检索

冯汝伟*,谢 强,丁秋林
(南京航空航天大学 计算机科学与技术学院,南京 210016)
(*通信作者电子邮箱 fruwei12@gmail.com)

摘 要:针对传统集中式索引处理大规模数据的性能和效率问题,提出了一种基于文本聚类的检索算法。利用文本聚类算法改进现有的索引划分方案,根据查询与聚类结果的距离计算判断查询意图,缩减查询范围。实验结果表明,所提方案能够有效地缓解大规模数据建索引和检索的压力,大幅提高分布式检索性能,同时保持着较高的准确率和查全率。

关键词:非结构化知识;分布式索引;文本聚类;全文检索;并行检索
中图分类号:TP391.3 **文献标志码**:A

Knowledge retrieval based on text clustering and distributed Lucene

FENG Ruwei*, XIE Qiang, DING Qiulin
(*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu* 210016, *China*)

Abstract: To solve the low performance and efficiency issues of the traditional centralized index when processing large-scale unstructured knowledge, the authors proposed the retrieval algorithm based on text clustering. The algorithm used text clustering algorithm to improve the existing index distribution method, and reduced the search range by judging the query intent through the distance of query and clusters. The experimental results show that the proposed scheme can effectively alleviate the pressure of indexing and retrieval in handling large-scale data. It greatly improves the performance of distributed retrieval, and it still maintains relatively high accuracy rate and recall rate.

Key words: unstructured knowledge; distributed index; text clustering; full-text search; parallel retrieval

0 引言

随着信息技术的发展,非结构化知识呈几何级数增长。传统的 Lucene 检索框架采用高度优化的倒排索引结构,大大地提高了非结构化文本知识的检索效率。但是 Lucene 面对大规模数据,索引时间急速增长,巨大的索引文件也给搜索带来性能瓶颈[1]。使用分布式技术对索引进行存储划分可以有效地克服集中式索引的缺点。现有的划分方案分为基于关键词或者基于文档,后者在多关键词搜索时的性能较优,更符合实际用户搜索需求[2]。但是目前文档划分的方案是按随机算法均衡划分,对于任意检索需要启动所有的索引服务器,给集群带来很高的负载;并且各个查询结果集的返回和归并的时间随着索引的数目而增长,影响了搜索的效率。

针对以上问题,本文在深入分析文本聚类和 Lucene 索引机制的基础上,引入文本聚类技术改进基于文档的划分方案,在 Hadoop 平台上结合 Mahout 分布式机器学习框架处理海量文本的分类,并行地建立索引,并提出基于查询意图的检索,该方案将有效地缓解对海量非结构化文本知识的建索引压力,提高检索性能。

1 分布式索引集群框架

分布式索引集群框架主要由聚类模块、索引集群组成,底层使用 Hadoop 分布式文件架构。聚类模块的任务是在分布式平台上执行聚类算法,划分存储在 HDFS(Hadoop Distributed File System)上的海量文本知识。索引集群根据聚类划分的结果并行地构建分布式索引,并向检索层提供检索服务。检索服务层分析用户的查询并调用索引集群进行检索。整个系统框架如图 1 所示。

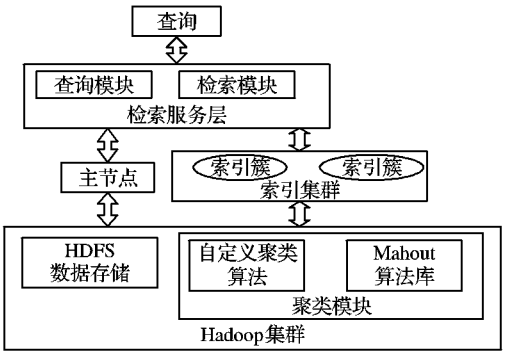

图 1 分布式索引集群框架

2 基于文本聚类的分布式索引构建技术

文本聚类将文本集合划分成不同的簇,在同一簇内的文档相似性极大,而在不同簇之间的文档相似性极小。聚类是一种无监督的机器学习问题,由于不需要训练过程,也不需要预先对文档手工标注类别,因此具有较高的灵活性和自动化处理能力[3]。文本聚类的流程如图 2 所示。

2.1 建立文本信息特征

文本聚类的首要问题是如何将非结构化的文本内容表示

收稿日期:2012-07-27;**修回日期**:2012-08-22。

作者简介:冯汝伟(1988-),男,江苏江阴人,硕士研究生,主要研究方向:分布式计算; 谢强(1972-),男,四川自贡人,副教授,博士,主要研究方向:知识工程、信息系统、信息安全; 丁秋林(1935-),男,江西抚州人,教授,博士生导师,主要研究方向:航空宇航制造工程、管理与信息化。

成为数学上可分析处理的形式，本文采用向量空间模型(Vector Space Model, VSM)来表示文本信息的特征。VSM 将文本表示为空间中的向量(T_1,W_1,T_2,W_2,…,T_n,W_n)，其中：T_i 为特征向量词汇，W_i 为 T_i 的权重。本文使用 TF-IDF(Term Frequency-Inverse Document Frequency)函数计算词汇权重[4]，以此来最大限度地区别不同文档。对于词汇 t 在文档 d 中的权重 $W_{t,d}$ 的计算如式(1)所示：

$$W_{t,d} = tf_{t,d} \times \log(n/df_t) \tag{1}$$

其中：词汇频率 $tf_{t,d}$ 为词 t 在文档 d 中的频率，词汇频率越大说明越重要；n 为文档总数；逆文档频率 df_t 为包含词 t 的文档数，逆文档频率越大说明越不重要。

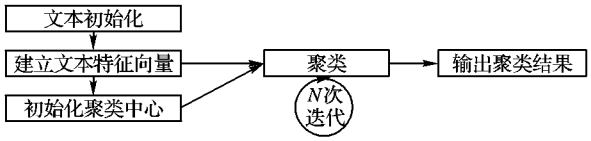

图 2　文本聚类处理流程

VSM 存在的一个问题是文档特征向量具有惊人的维数，因此本文采用向量的稀疏表示方法来表示特征向量。另外设定词频的阈值来缩减文本向量的维度，去除出现次数过少与次数过多的常见词汇。Mahout 默认采用 Lucene 的 StandAnalyzer 作为词汇分析器[5]，由于中文分词不同于英文分词，不能简单地以空格和字切分，在向量化过程本文使用 IKAnalyzer 中文分析器对文本进行分词。

2.2　文本聚类

在建立文本信息特征后，接下来的工作就是在此数学形式的基础上，对其进行基于距离的聚类处理。由于文本向量的高维性，若采用欧氏距离度量文本的相似度，距离数值将会比较大，在聚类的过程中很难设定较合理的距离阈值，所以本文采用余弦距离度量，两个向量($\boldsymbol{A} = \{a_1,a_2,\cdots,a_n\}$，$\boldsymbol{B} = \{b_1,b_2,\cdots,b_n\}$)之间的余弦距离计算如式(2)：

$$Cosine\_distance(\boldsymbol{A},\boldsymbol{B}) = 1 - \sum_{i=1}^{n}(a_i \times b_i) \Big/ \left(\sqrt{\sum_{i=1}^{n} a_i^2} \times \sqrt{\sum_{i=1}^{n} b_i^2}\right) \tag{2}$$

本文使用 *K*-means 算法进行聚类，但对于海量的非结构化知识，很难给出较合理的聚类结果簇的数量 k 与初始簇心[6]。所以本文结合 Canopy 算法，初始化 *K*-means 的输入。整个聚类过程如图 3 所示。

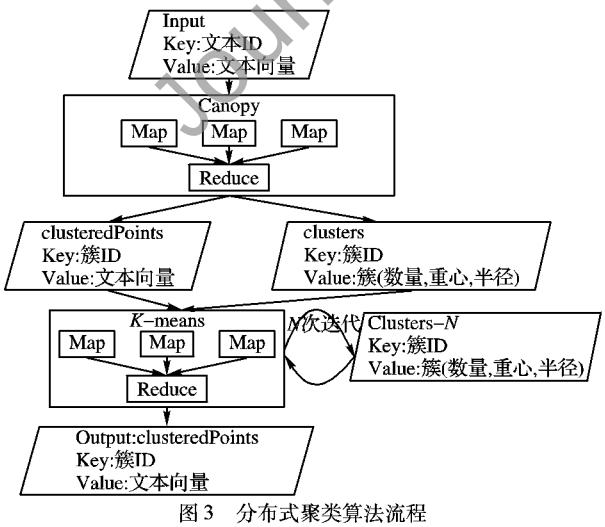

图 3　分布式聚类算法流程

Canopy 算法的过程如下。

1) 向量集加入初始 list，选择两个距离阈值：*T*1 和 *T*2，其中 *T*1 > *T*2；

2) 从 list 中任取一点 *P*，计算点 *P* 与所有簇的距离(如果当前不存在簇，则把点 *P* 作为一个簇)，如果点 *P* 与某个簇距离在 *T*1 以内，则将点 *P* 加入到这个簇；

3) 如果点 *P* 曾经与某个簇的距离在 *T*2 以内，则需要把点 *P* 从 list 中删除，这一步是认为点 *P* 此时与这个簇已经够近了，因此它不可以再做其他簇的中心了；

4) 重复步骤 2) ~ 3)，直到 list 为空结束。

在 Hadoop 上 Canopy 聚类的 Map/Reduce 实现如下。

Map 阶段　对本地向量进行 Canopy 聚类，输出 Canopy 各个簇的中心向量。

Reduce 阶段　对输入的各个簇的中心向量进行 Canopy 聚类，输出全局的 Canopy 簇。由于 Reduce 阶段聚类的数据集是 Map 结果的各个簇的中心向量，为了避免簇数减少过多，造成各簇间区别不明显，经实验测试将 Reduce 阶段的距离阈值 *T*1，*T*2 都减小一定数值，来获得一个较优的聚类结果。

在进行过 Canopy 聚类后，将得到的簇作为初始重心，k 值为 Canopy 聚类后簇数，进行下一步 *K*-means 聚类[7]。*K*-means 算法的过程如下：

1) 计算每个点到各个簇重心的距离，将它加入到最近的簇；

2) 重新计算每个簇的重心；

3) 重复步骤 1) ~ 2)，直到各个簇重心在精度范围内或达到最大迭代次数。

在 Hadoop 上 *K*-means 聚类的 Map/Reduce 实现[8] 如下：

Map 阶段　对本地向量依据当前聚簇信息，将每个点加入到与其距离最近的簇，输出结果为〈与当前点距离最近簇的 ID，当前点〉；

Reduce 阶段　将输入的〈簇 ID，点〉进行汇总，重新计算重心，并判断收敛。

2.3　索引建立

在得到聚类的结果后，主节点保存各个簇的中心、半径等信息，并根据聚类结果，将文档分发给各个索引节点并行创建 Lucene 索引。Lucene 的 IndexReader 不支持直接从 HDFS 文件系统初始化[9]，并且从 HDFS 上下载庞大的索引文件到本地将耗费大量时间。本文将索引文件存储在本地，主节点记录各个簇对应索引的位置。由于聚类的结果并不一定在数量上均衡(见后面的实验)，采用一台索引服务器构造一个簇的索引并不适合，并且聚集结果的簇数可能超过已有节点的数量。本文采用图 4 的策略建立索引，通过负载均衡将索引建立在索引服务器上，为每个索引分配独立的端口，通过多线程并发访问索引，提高存储与检索性能。

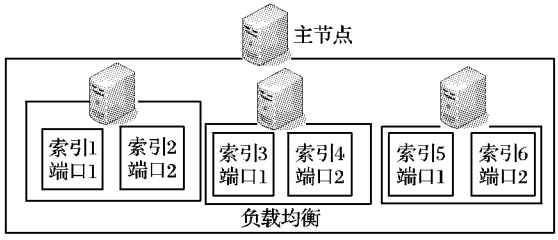

图 4　索引分布策略

3　文本内容检索算法

本文的文档划分方案将相似的文档聚成一类，在检索时根据用户的查询意图，选择有可能的类进行检索，避免检索所有索引，减少索引节点的启动数量。

虽然 Luncene 有 MultiSearcher 与 ParallelMultiSearcher 支持在多个索引搜索,但仅限于索引都存在本地的情况,将庞大的索引文件从其他节点复制到本地非常影响性能[10]。另外 RemoteSearcher 仅适合小规模的分布式远程索引。本文通过计算全局的逆文档频率(Inverse Document Frequency, IDF)值,改写 Lucene 的打分机制实现分布式的检索[11]。检索的过程分为以下 4 个步骤。

1)通过查询的意图来选择索引节点。

计算查询串在各个索引的得分,排序得分确定需要查询的索引集合 S。选择超过特定阈值的簇或者根据得分排序选取特定数目的簇。假设查询串为 $q = \{tq_1, tq_2, \cdots\}$,索引簇 *index*-1 的簇心 $sc = \{t_1:w_1, t_2:w_2, \cdots, t_n:w_n\}$。本文根据式(3)计算查询串在索引 *index*-1 的得分:

$$Score(q, index\text{-}1) = \sum_{t \text{ in } q} w_t \tag{3}$$

2)计算各个查询词的全局文档频率(Document Frequency, DF)值,然后再计算全局 *IDF*。

3)查询 S 中各个索引,通过全局的 *IDF* 值计算结果得分。

IndexSearcher 是 Lucene 搜索中的最主要的类,Similarity 会调用 IndexSearch 中的统计函数来获取词条的 *DF* 值和文档总数,本文方法继承并覆盖 IndexSearcher 中相应的方法 docFreq()和 maxDoc(),分别返回对应的全局 *DF* 值和文档数。

4)返回检索结果归并排序输出。

4 实验及分析

本文的测试数据来自新浪博客 220 万篇博客文章,总共 15 GB大小。经过本文的文本聚类算法,测试数据集的聚类结果如图 5 所示。

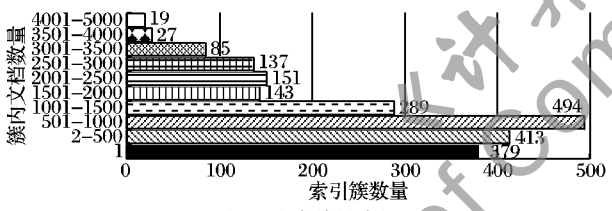

图 5 聚类结果分析

聚类得到的 2 137 个簇中有 379 个文档各自单独形成一个簇,对于这些孤立点包括簇中数量小于 50 的簇,不适合单独为其建立索引,所以在划分文档时将其归并到同一个簇中,来减少簇的数目,最后缩减为 1 678 个簇。

在目前国内外中文搜索引擎的研究中,得到用户的平均查询串中词数为 2.27[12]。提取搜索日志中包含 1 ~ 6 个关键词的查询串各 100 个进行查询测试,得到需查询索引个数的最大值、最小值和平均值如图 6 所示。结果显示查询索引的个数随着查询词数量的增加而递减,但在实验中也发现随着查询字数 > 10 之后,由于查询串在各个索引的概率都偏低,使得查询索引的个数不稳定地上升。但可以发现,引入聚类使得每次查询仅仅只需查找 2% 左右的索引节点,大大缓解了查询压力,提高了查询性能。

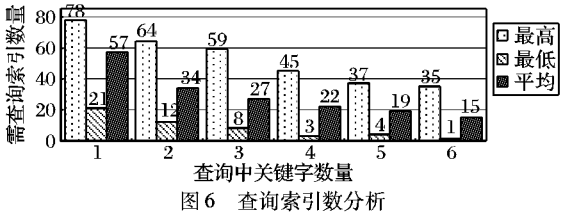

图 6 查询索引数分析

最后本实验对检索的查全率进行分析,因为使用 Lucene 单独建立索引,因此认为检索的查准率还是可靠的,而分布式检索对文档的召回有较大影响。由于分析显示大多用户只查看搜索的前 100 项结果,所以本实验分别计算全局的查全率和排名前 100 和前 30 的查全率,分析结果如图 7 所示。结果显示查全率随着关键词的个数增加而少量提高,并且 Top100 和 Top30 的查全率都比较高。

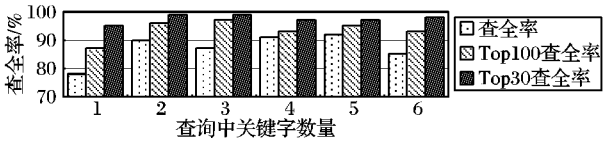

图 7 查全率分析

5 结语

本文针对海量非结构化知识的检索问题,将文本聚类技术引入到分布式索引的设计中,提出了一种在云平台下基于聚类的分布式索引存储模型,解决当前集中式索引的存储性能问题,通过查询意图的推断减少并行检索服务数量,提高了分布式索引技术的性能。通过实验分析,基于 Canopy 与 *K*-means 聚类的分布式检索,在有效缩小检索范围的情况下,并没有对检索结果的准确率和查全率带来较大影响,大幅提高了信息检索的效率。

参考文献:

[1] 蒋明原,孔令德,宁静静. 一种海量数据下的 Lucene 全文检索解决方案[J]. 电脑开发与应用,2011,24(4):32 - 35.

[2] MOFFAT A, WEBBER W, ZOBEL J. Load balancing for term-distributed parallel retrieval [C]// SIGIR'06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2006: 348 - 355.

[3] 曹宇,尹刚,李翔,等. 聚类搜索引擎研究进展浅析[J]. 电脑知识与技术,2011,7(22):5398 - 5400.

[4] 徐文海,温有奎. 一种基于 TFIDF 方法的中文关键词抽取算法[J]. 情报理论与实践,2008,31(2):298 - 302.

[5] OWEN S, ANIL R, DUNNING T, *et al*. Mahout in action [M]. Greenwich: Manning Publications, 2010: 123 - 137.

[6] ESTEVES R M, PAIS R, RONG C. *K*-means clustering in the cloud—a Mahout test [C]// Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications. Washington, DC: IEEE Computer Society, 2011: 514 - 519.

[7] ESTEVES R M, RONG C. Using Mahout for clustering Wikipedia's latest articles: a comparison between *K*-means and fuzzy *C*-means in the cloud [C]// Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science. Washington, DC: IEEE Computer Society, 2011: 565 - 569.

[8] 李应安. 基于 Map/Reduce 的聚类算法的并行化研究[D]. 广州:中山大学,2010.

[9] BUTLER M H, RUTHERFORD J. Distributed Lucene: a distributed free text index for Hadoop [EB/OL]. [2012-03-25]. http://www.hpl.hp.com/techreports/2008/HPL-2008-64.pdf.

[10] SAJJA K. Performance study of Lucene in parallel and distributed environments [D]. Boise: Boise State University, 2011.

[11] HATCHER E, GOSPODNETIC O, McCANDLESS M. Lucene in action [M]. Greenwich: Manning Publications, 2009.

[12] 王浩,姚长利,郭琳,等. 基于中文搜索引擎网络信息用户行为研究[J]. 计算机应用研究,2009,26(12):4665 - 4668.