

基于 Spark 的压缩近邻算法

张素芳¹ 翟俊海² 王婷婷² 郝璞² 王聪² 赵春玲²

(中国气象局气象干部培训学院河北分院 河北保定 071000)¹

(河北大学数学与信息科学学院河北省机器学习与计算智能重点实验室 河北保定 071002)²

摘要 K-近邻(K-Nearest Neighbors,K-NN)是一种懒惰学习算法,用 K-NN 对数据分类时,不需要训练分类模型。K-NN 算法的优点是思想简单、易于实现;缺点是计算量大,原因是在对测试样例进行分类时,其需要计算测试样例与训练集中每一个训练样例之间的距离。压缩近邻算法(Condensed Nearest Neighbors,CNN)可以克服 K-NN 算法的不足。但是,在面对大数据集时,由于自身的迭代计算特性,CNN 的运算效率会变得非常低。针对这一问题,提出一种名为 Spark CNN 的压缩近邻算法。在大数据环境下,与基于 MapReduce 的 CNN 算法相比,Spark CNN 的效率大幅提高,在 5 个大数据集上的实验证明了这一结论。

关键词 压缩近邻,大数据,样例选择,迭代计算,懒惰学习

中图分类号 TP181 文献标识码 A

Spark Based Condensed Nearest Neighbor Algorithm

ZHANG Su-fang¹ ZHAI Jun-hai² WANG Ting-ting² HAO Pu² WANG Cong² ZHAO Chun-ling²

(Hebei Branch of China Meteorological Administration Training Centre,China Meteorological Administration,Baoding,Hebei 071000,China)¹

(Key Lab. of Machine Learning and Computational Intelligence,College of Mathematics and Information Science,

Hebei University,Baoding,Hebei 071002,China)²

Abstract K-nearest neighbors (K-NN) is a lazy learning algorithm. It is unnecessary to train classification models, when one uses K-NN for data classification. K-NN algorithm is simple and easy to implement. The disadvantages of K-NN is that it requires large number of computations, which is introduced by calculating distances between testing instance and every training instance. Condensed nearest neighbors (CNN) can overcome the drawback of K-NN mentioned above. However, CNN is an iterative algorithm, when it is applied in big data scenario, its efficiency becomes very low. In order to deal with this problem, this paper proposed an algorithm named Spark CNN. In big data circumstances, Spark CNN can significantly improve the efficiency of CNN. This paper experimentally compared the Spark CNN with MapReduce CNN on 5 big data sets, the experimental results show that the Spark CNN is very effective.

Keywords Condensed nearest neighbors, Big data, Instance selection, Iterative calculation, Lazy learning

1 引言

K-近邻^[1]是一种懒惰学习算法,用 K-NN 对数据分类时,不需要训练分类模型。K-NN 算法的优点是思想简单、易于实现;缺点是计算量大,原因是在对测试样例进行分类时,其需要计算测试样例与训练集中每一个训练样例之间的距离。当面对大数据环境时,K-NN 的这一缺点更加突出。针对 K-NN 的这一缺点,研究人员进行了大量的研究,提出了许多改进的方法,这些方法大致可分为 3 类:1)基于样例选择的方法;2)基于近似近邻搜索的方法;3)基于并行化的方法。

基于样例选择的方法是从原训练集中选择一个子集来代

替原训练集后用 K-NN 对数据进行分类。这种方法研究的历史最悠久,早在 1968 年,Hart^[2]就提出了一种著名的样例选择算法,即著名的压缩近邻算法。CNN 算法选择的样例大多分布在分类边界附近,这条准则几乎成了样例选择的基本准则。近几年,研究人员提出的样例选择方法大多也采用这一准则。例如,基于交叉验证思想,Zhai 等人^[3]提出了交叉选择样例方法;针对 K-NN 回归问题,Song 等人^[4]提出了基于排序的样例选择方法;基于模糊粗糙集技术,Onan^[5]提出了样例选择方法,等。

顾名思义,基于近似近邻搜索的方法是在整个训练集中搜索目标样例的近似近邻,这类方法目前研究得最多的是基

本文受国家自然科学基金项目(71371063),河北省自然科学基金项目(F2017201026),河北大学自然科学研究计划项目(799207217071),河北大学大学生创新训练项目(2017071)资助。

张素芳(1966—),女,硕士,副教授,主要研究方向为机器学习;翟俊海(1964—),男,博士,教授,主要研究方向为机器学习,E-mail:mczjh@hbu.com(通信作者);王婷婷(1991—),女,硕士生,主要研究方向为云计算与大数据处理。

于哈希技术的方法。例如,基于局部敏感哈希技术,Alvar 等人^[6]提出了计算时间复杂度为线性级的样例选择算法,这种方法能实现大数据集的样例选择;基于树的紧哈希技术,Hou 等人^[7]提出了近似近邻搜索方法;基于谱哈希技术,Wan 等人^[8]提出了针对高维数据的近似近邻搜索方法;文庆福等人^[9]提出了基于分布式哈希技术的近似近邻搜索方法,等。

基于并行化的方法大多利用开源的大数据平台(如 Hadoop 和 Spark)并行地进行 K-近邻的搜索。由于大数据是近几年最火热的研究方向,因此研究人员提出了许多这类算法。例如,刘义等人^[10]提出了基于 MapReduce 和 R-树的 k-近邻连接算法;针对高维大数据分类问题,Muja 等人^[11]提出了一种可扩展的 K-NN 分类算法,其在计算机视觉应用中取得了良好的效果;Maillo 等人^[12]提出了一种基于 Spark 的大数据 K-近邻分类方法;基于 MapReduce 和随机权网络,Zhai 等人^[13]提出了一种大数据投票样例选择方法。Song 等人^[14]对基于 MapReduce 的大数据 K-NN 算法进行了理论和实验分析,证明了其具有较高的参考价值。

因为 CNN 从训练集中选择样例的过程是一种迭代计算的过程,而 Spark 最适合处理大数据迭代计算和交互式计算,所以基于这种想法,本文提出了基于 Spark 的 CNN 算法,简称 Spark CNN。在大数据环境下,Spark CNN 可以显著提高 CNN 算法的效率。将其与基于 MapReduce 的压缩近邻算法(MapReduce CNN)在 5 个数据集上进行了实验比较,结果证明提出的算法是非常有效的。

2 基础知识

本节介绍相关的基础知识,包括压缩近邻算法 CNN 和 Spark 大数据内存并行计算框架。

2.1 CNN 算法

压缩近邻算法 CNN 使用的是 1-近邻,也称为最近邻,其思想非常简单。设 T 是训练集, T' 是选择的样例集合。开始时,从训练集 T 中随机选择一个样例,并将其从 T 中移动到 T' 中。然后,用 T' 分类 T 中的样例,凡是被错误分类的样例,就从 T 中移动到 T' 中。这样, T 中的样例不断减少,而 T' 中的样例不断增加。当 T 变为空集或 T 中的所有样例(剩余的样例)都能被 T' 正确分类时,算法终止。压缩近邻算法的伪代码如算法 1 所示。

算法 1 压缩近邻算法

输入:训练集 $T = \{(x_i, y_i) \mid x_i \in R^d, y_i \in Y, 1 \leq i \leq n\}$

初始化 $T' = \emptyset$;

从 T 中随机选择一个样例,并将其从 T 中移动到 T' 中;

repeat

 for (each $x_i \in T$) do

 for (each $x_j \in T'$) do

 计算 x_i 到 x_j 之间的距离;

 寻找 x_i 在 T' 中的最近邻(1-近邻) x_j^* ;

 end

 end

 if (x_i 的类别和 x_j^* 的类别不同) then

$T' = T' \cup \{x_i\}$;

$T = T - \{x_i\}$;

 end

end

until ($T = \emptyset$ 或 T 中的所有样例都能被 T' 用最近邻(1-近邻)正确分类);

输出: T'

从算法 1 可以看出,压缩近邻算法 CNN 是一种迭代算法,其计算时间复杂度为 $O(n^3)$ 。当训练集 T 的规模较大时,压缩近邻算法的效率会变得非常低,面对大数据集,它甚至会变得不可行。针对这一问题,本文提出了基于 Spark 的压缩近邻算法,简称 Spark CNN。本文提出的算法充分利用了 Spark 内存计算及适于迭代计算的特点,可大幅提高压缩近邻算法的效率,在 5 个数据集上的实验结果证实了这一结论。

2.2 Spark 并行计算框架

Spark 是基于内存计算的大数据并行计算框架,是在克服另一个大数据开源计算框架 Hadoop 不足的基础上提出的^[15]。Spark 的首要设计目标是避免运算时出现过多的网络和磁盘 IO 开销,为此它将核心数据结构设计为弹性分布式数据集(Resident Distributed Dataset, RDD)。用户可以使用 RDD 将一部分数据集缓存在内存中,使其能在并行操作时被有效地重复使用。Spark 为 RDD 提供了一系列算子,以对 RDD 进行有效的操作。此外,为了避免 Hadoop 启动和调度作业消耗过大的问题,Spark 采用基于有向无环图(Directed Acyclic Graph, DAG)的任务调度机制进行优化,这样就可以将多个阶段的任务并行或串行执行,无需将每一个阶段的中间结果存储到 HDFS(Hadoop Distributed File System)上。

Spark 的两个核心概念是 RDD 和算子。RDD 是一个抽象的数据结构,它将一个大数据集以分布式方式保存在机群服务器的内存中。从物理上讲,RDD 将一个大数据集划分为若干个数据块,这些数据块以分布式方式存储在机群的各个节点上,可以存储在节点的内存中,也可以存储在节点的外存中。每一个数据块有一个标识 BlockID,通过 BlockID 元数据实现对数据块的管理。从逻辑上讲,这些数据块被划分为若干个分区。RDD 通过算子对分区数据进行操作,结果会得到一个新的 RDD。算子是 Spark 中定义的函数,用于对 RDD 中的数据进行操作。Spark 中的算子可以分为以下 4 类:

1) 创建算子,用于将缓存在内存中的数据或外存文件创建为 RDD 对象;

2) 变换算子,用于将一个 RDD 变换为另一个 RDD;

3) 缓存算子,用于将 RDD 缓存在内存中或外存中,以便后续运算继续使用;

4) 行动算子,会触发 Spark 作业执行,并保存计算结果 RDD。

图 1 以大数据词频统计为例,给出了 Spark 处理大数据的流程。从图 1 可以看出 RDD 和算子之间的关系。

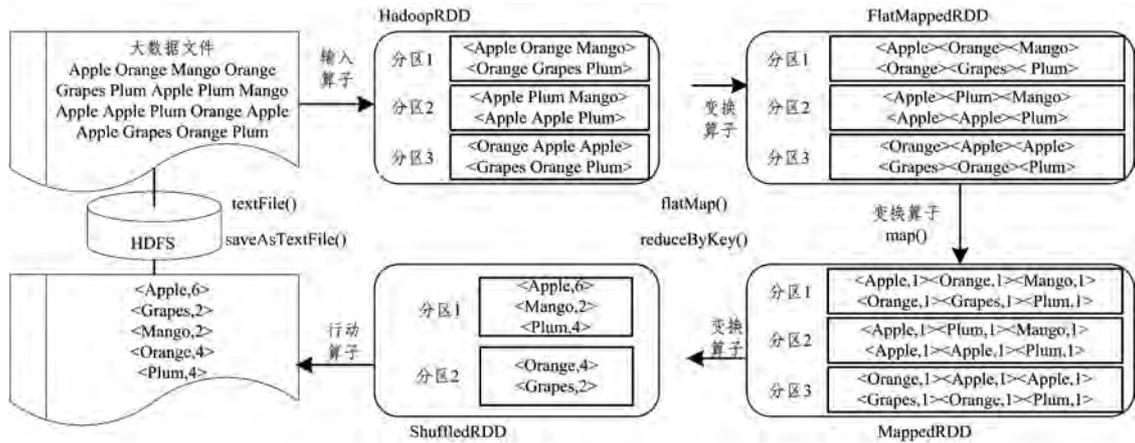


图1 Spark 处理大数据的流程(以词频统计为例)

3 基于 Spark 的压缩近邻算法

本节首先介绍本文提出的 Spark CNN 算法的基本思想, 然后给出 Spark CNN 算法的伪代码。Spark CNN 算法的思想如图 2 所示, 它主要利用了 Spark 中 map 算子分而治之的思想和 Spark 内存计算的特性。具体地, 首先将大数据集合 T 划分为若干子集, 并部署到 m 个 Spark 云计算节点上; 然后在这些节点上应用 CNN 算法并行地从相应的子集中选择样例; 最后将各个节点选择的样例合并到一起, 得到最终选择的样例子集。因为 CNN 算法是一种迭代算法, 所以在各个节点上应用 CNN 算法选择样例时, 利用 Spark 的内存计算特性将每一次迭代得到的中间结果缓存进内存中, 在之后 CNN 算法的不断迭代过程中, 可以不断重复利用缓存的中间结果, 大大减少算法的运行时间, 大幅度提高压缩近邻算法 CNN 的效率。Spark CNN 算法的伪代码如算法 2 所示。

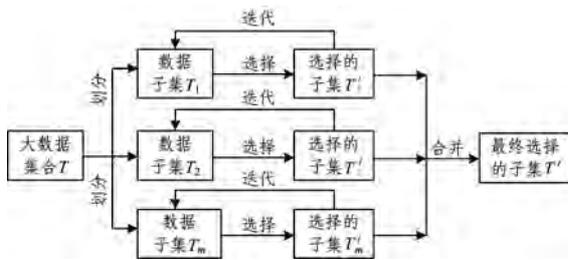


图2 Spark CNN 算法的基本思想

算法 2 Spark CNN 算法

输入: 训练集 $T = \{(x_i, y_i) \mid x_i \in \mathbb{R}^d, y_i \in Y, 1 \leq i \leq n\}$

初始化 $T' = \emptyset$;

从 T 中随机选择一个样例, 并将其从 T 中移动到 T' 中;

repeat

//用算子 textFile 读取压缩样例子集 T'

var lines: RDD[String] = sc.textFile(T');

//用 map 算子将大数据集划分为若干个子集, 并部署到不同的 Spark 节点。各个节点并行地用 T' 按最近邻对子集中的样例进行分类, 并选择错误分类的样例

var T' : RDD = lines.map($x_j \Rightarrow \{$

for ($i=1; i \leq \text{subT.size}(); i=i+1$) do

Distance = EuclideanDistance(x_j , subT.get(x_i));

if (Distance = minDistance) then

if (x_i 的类别和 x_j 的类别不同) then

$T' = T' \cup \{x_i\}$;

$T = T - \{x_i\}$;

end

end

end

)

//用 saveAsTextFile 算子将最新的压缩样例子集缓存到 T' 中

result.cache().saveAsTextFile(T');

until ($T = \emptyset$ 或 T 中的所有样例都能被 T' 用最近邻(1-近邻)正确分类);

输出 T'

4 实验结果

为了验证本文所提算法的有效性, 将其与基于 MapReduce 的 CNN 算法^[16] 在 5 个大数据集上进行实验比较。5 个大数据集中, 包括 2 个人工数据集和 3 个 UCI 数据集。第一个人工数据集是它们二维数据集, 每类 100000 个样例, 共 300000 个样例。它们服从的概率分布为:

$$p(\mathbf{x}|\omega_1) \sim N(\mathbf{0}, \mathbf{I}), p(\mathbf{x}|\omega_2) \sim N\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{I}\right),$$

$$p(\mathbf{x}|\omega_3) \sim N\left(\frac{1}{2}\left(\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \mathbf{I}\right) + \frac{1}{2}\left(\begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}, \mathbf{I}\right)\right)$$

第二个人工数据集是一个 2 类二维数据集, 每类包含 200000 个样例点, 共 400000 个样例点。它们服从的高斯分布为 $p(\mathbf{x}|\omega_i) \sim N(\mu_i, \Sigma_i), i=1, 2$, 参数如表 1 所列。

表 1 高斯分布的相关参数

i	μ_i	Σ_i
1	$\begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$	$\begin{bmatrix} 0.6 & -0.2 \\ -0.2 & 0.6 \end{bmatrix}$
2	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.2 & -0.1 \\ -0.1 & 0.2 \end{bmatrix}$

5 个大数据集的基本信息列于表 2 中; 实验环境是具有 6 个节点的云计算平台, 节点的基本配置信息如表 3 所列; Spark 云计算平台中节点的具体规划如表 4 所列; 在 5 个数据集上的实验结果分别列于表 5—表 9 中, 从中可以看出, 基于 Spark 的 CNN 算法的效率远远高于基于 MapReduce 的 CNN 的效率。

表 2 实验所用数据集的基本信息

数据集	样例个数	属性个数	类别个数
Gaussian1	300000	2	3
Gaussian2	400000	2	2
Skin Segmentation	240000	3	2
Poker Hand	1000000	10	10
Forest CoverType	580000	54	7

表 3 云计算节点的基本配置信息

软硬件项目	配置情况
处理器(CPU)	Inter Xeon E5-4603 2.0GHz (双核)
内存	8GB RDIMM
硬盘	1 TB
网卡	Broadcom 5720 QP 1GB 网络子卡(四端口)
网络设备	华为 S3700 系列以太网交换机
操作系统	CentOS 6.4
云计算平台	spark-1.6.0-bin-hadoop2.4.tgz
JDK 版本	Jdk1.8

表 4 Spark 云计算节点的功能规划

节点号	主机名	IP 地址	节点类型
1	host1	10.187.84.50	Master
2	host2	10.187.84.51	Worker
3	host3	10.187.84.52	Worker
4	host4	10.187.84.53	Worker
5	host5	10.187.84.54	Worker
6	host6	10.187.84.55	Worker

表 5 在数据集 Gaussian1 上的实验结果

(单位:s)

迭代次数	Spark CNN	MapReduce CNN
第 1 次	127.21	269.7
第 2 次	46.01	220.8
第 3 次	45.78	209.5
第 4 次	44.67	208.9
第 5 次	43.90	207.7

表 6 在数据集 Gaussian2 上的实验结果

(单位:s)

迭代次数	Spark CNN	MapReduce CNN
第 1 次	178.25	340.85
第 2 次	54.55	315.1
第 3 次	49.00	310.0
第 4 次	47.34	309.7
第 5 次	46.19	309.4

表 7 在数据集 Skin Segmentation 的实验结果

(单位:s)

迭代次数	Spark CNN	MapReduce CNN
第 1 次	18.620	134.201
第 2 次	9.031	127.807
第 3 次	8.213	132.576
第 4 次	8.012	130.243
第 5 次	8.011	135.011

表 8 在数据集 Poker Hand 上的实验结果

(单位:s)

迭代次数	Spark CNN	MapReduce CNN
第 1 次	59.013	243.023
第 2 次	12.485	247.724
第 3 次	11.384	238.834
第 4 次	10.527	252.324
第 5 次	10.634	237.087

表 9 在数据集 Forest CoverType 上的实验结果

(单位:s)

迭代次数	Spark CNN	MapReduce CNN
第 1 次	38.673	422.867
第 2 次	15.324	410.393
第 3 次	10.383	435.013
第 4 次	10.245	402.891
第 5 次	9.865	432.081

结束语 本文提出了一种基于 Spark 的压缩近邻算法,通过研究我们得出两个结论:1)因为 Spark 是一种大数据内存计算框架,所以基于 Spark 的压缩近邻算法的运行效率远高于基于 MapReduce 的压缩近邻算法;2)Spark 在第一次迭代的过程中需要从 HDFS 上读取数据,并通过 Spark 引擎将其转化成 RDD 后缓存进内存中,这需要花费较多的时间,但在以后的运行过程中可以重复利用缓存的 RDD 数据集,Spark 压缩近邻算法的运行时间会大大减少。实验结果显示,与基于 MapReduce 的 CNN 算法相比,Spark CNN 具有非常高的效率。

参考文献

- [1] COVER T, HART P. Nearest neighbor pattern classification [J]. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.
- [2] HART P. The condensed nearest neighbor rule [J]. IEEE Transaction on Information Theory, 1968, 14(5): 15-516.
- [3] ZHAI J H, LI T, WANG X Z. A cross-selection instance algorithm [J]. Journal of Intelligent & Fuzzy Systems, 2016, 30 (2): 717-728.
- [4] SONG Y S, LIANG J Y, LU J, et al. An efficient instance selection algorithm for k nearest neighbor regression [J]. Neurocomputing, 2017, 251: 26-34.
- [5] ONAN A. A fuzzy-rough nearest neighbor classifier combined with consistency-based subset evaluation and instance selection for automated diagnosis of breast cancer [J]. Expert Systems with Applications, 2015, 42(20): 6844-6852.
- [6] ALVAR A G, JOSE-FRANCISCO D P, RODRIGUEZ J J, et al. Instance selection of linear complexity for big data [J]. Knowledge-Based Systems, 2016, 107(C): 83-95.
- [7] HOU G, CUI R, PAN Z, et al. Tree-based compact hashing for approximate nearest neighbor search [J]. Neurocomputing, 2015, 166(C): 271-281.
- [8] WAN J, TANG S, ZHANG D D, et al. HDIdx: High-dimensional indexing for efficient approximate nearest neighbor search [J]. Neurocomputing, 2017, 237: 401-404.
- [9] 文庆福, 王建民, 朱晗, 等. 面向近似近邻查询的分布式哈希学习方法 [J]. 计算机学报, 2017, 40(1): 192-206.
- [10] 刘义, 景宁, 陈萃, 等. MapReduce 框架下基于 R-树的 k-近邻连接算法 [J]. 软件学报, 2013, 24(8): 1836-1851.
- [11] MUJA M, LOWE D G. Scalable nearest neighbor algorithms for high dimensional data [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 36(11): 2227-2240.
- [12] MAILLO J, RAMÍREZ S, TRIGUERO I, et al. kNN-IS: An Iterative Spark-based design of the k-nearest neighbors classifier for

big data [J]. Knowledge-Based Systems, 2017, 117: 3-15.

[13] ZHAI J H, WANG X Z, PANG X H. Voting-based instance selection from large data sets with mapreduce and random weight networks[J]. Information Sciences, 2016, 367: 1066-1077.

[14] SONG G, ROCHAS J, BEZE L E, et al. K nearest neighbour joins for big data on mapreduce: a theoretical and experimental

analysis[J]. IEEE Transactions on Knowledge & Data Engineering, 2016, 28(9): 2376-2392.

[15] 刘军, 林文辉, 方澄. Spark 大数据处理-原理、算法与实例[M]. 北京: 清华大学出版社, 2016.

[16] 翟俊海, 郝璞, 王婷婷, 张明阳. MapReduce 并行化压缩邻近算法[J]. 小型微型计算机系统, 2017(12): 2678-2682.

(上接第 397 页)

定一个较低数值的 λ (如 0.1), 然后随机抽取几次 (如 10 次) 一定规模的词对样本 (如 100 个) 交由专家判定。若专家判定样本中的词对确实具有语义关联性, 则此时的 λ 即可作为阈值输出; 否则, 对 λ 递增一定步长, 再重复随机抽取和专家判定的过程。

事实上, 随着 λ 的递增, 所抽取样本中专家判定具有语义关联性的词对比率会同步递增。因此, 初期可将 λ 递增的步长设置为一个较大的数值 (如 0.01), 随着 λ 的递增再将步长逐步减小 (如 0.005)。

3.5 词簇中心的选取

假设 $W_1 = \{w_i | i=1, 2, \dots, s, s \leq t\}$ 是一个连通词簇, 即目标领域主题中的一个子主题。定义词簇中某一个词汇 w_i 与词簇 W_1 的语义关联度 $O(w_i)$ 如下:

$$O(w_i) = TFIDF(w_i) + \sum_{\substack{j=1 \\ j \neq i}}^s Q(w_i, w_j)$$

通过归一化, 可得到词汇 w_i 对词簇 W_1 的语义贡献度

$$O(w_i | W_1) = \frac{O(w_i)}{\sum_{w_i \in W_1} O(w_i)}$$

从而, 选取对词簇 W_1 语义贡献度最大的词汇作为词簇中心 w_1^* , 即:

$$w_1^* = \max_{w_i \in W_1} O(w_i | W_1)$$

词簇中其他词汇即作为词簇中心词汇的连接词汇。

3.6 主题词表的生成

由 3.1 节得到目标领域主题候选词, 由 3.3 节得到基于词共现的主题词簇图, 由 3.4 节得到每个词簇的中心词汇, 从而给出如下主题词表生成算法。

Step1 在主题词簇集 W 中选择任意初始词汇节点, 设为 w_0 ;

Step2 遍历词汇 w_0 所在的词簇, 设为 $W_0 \subset W$, 并计算词簇 W_0 的词簇中心, 设为 w_0^* ;

Step3 以词汇 w_0^* 为中心词汇, 以词簇 W_0 中其他词汇为连接词汇, 构建目标领域主题词表 $(w_0^*, O(w_0^*)): \{(w_i, O(w_i)) | w_i \in W_0\}$;

Step4 若主题词簇集 $W \setminus W_0 = \emptyset$, 则退出; 否则选择词汇节点, 设为 $w_1 \neq w_0$, 重复 Step2 和 Step3。

4 算例

为验证本文所提方法对目标领域主题词表抽取的效果, 选用某时政类报刊上 715 篇新闻报道进行实验。每篇报道不短于 1000 字, 并标注其中 223 篇是关于主题“美国海外军事活动”的。

以这 715 篇新闻报道作为样本语料, 以标注具有主题性的 223 篇新闻报道作为领域文本集。抽取出目标领域主题词表共 377 个词汇, 29 个词簇, 表 1 列出了部分抽取结果。

表 1 算例计算结果

核心词汇	链接词汇
局势(0.18)	危机(0.11) 原则(0.09) 政治(0.07) 事态(0.06) 立场(0.06) ...
外交部(0.23)	发言人(0.13) 外长(0.10) 公约(0.07) 应对措施(0.04) 大使(0.02) ...
航空母舰(0.3)	护卫舰(0.21) 战机(0.17) 海军基地(0.05) 战斗群(0.03) 武器(0.01) ...
水域(0.27)	军事行动(0.15) 主权(0.14) 安全(0.12) 岛链(0.03) 射程(0.03) ...
演习(0.4)	联合军事演习(0.33) 作战单位(0.05) 演练项目(0.04) 编队(0.04) 救援(0.03) ...
...	...

通过与该领域专家人工抽取结果的比较表明, 该词表具有较好的准确率和召回率。

结束语 对特定领域主题词表的提取, 一方面能为进一步优化自然语言的处理性能和效果提供基础, 另一方面能为本体构建提供概念的构成要素和关系结构^[8]。

本文提出一种面向给定语料的领域主题词表提取算法, 构建了一种从语料中自动获取主题词表的语言处理模型, 实现了对特定专业或细分领域主题词表的统计学习。该方法是一种基于计算的经验主义方法, 不受人为主观因素的影响, 计算结果具有客观性。

参考文献

- [1] 常春, 卢文林. 叙词表编制历史、现状与发展[J]. 农业图书情报学刊, 2002(5): 25-28.
- [2] 肖健, 徐建, 徐晓兰, 等. 英中可比语料库中多词表达自动提取与对齐[J]. 计算机工程与应用, 2010, 46(31): 130-134.
- [3] 陈炯, 张永奎. 一种基于词聚类的文本特征描述方法[J]. 计算机系统应用, 2011, 20(2): 211-215.
- [4] 葛宁, 王军. 领域 Ontology 的自动丰富——基于 ADL 地名表的实例研究[J]. 计算机科学, 2007, 34(9): 156-162.
- [5] 奉国和, 郑伟. 国内中文自动分词技术研究综述[J]. 图书情报工作, 2011, 55(2): 41-45.
- [6] SALTON G, CLEMENT T Y. On the construction of effective vocabularies for information retrieval[C]// Proc. of 1973 Meeting on Programming Languages and Information Retrieval. New York, USA: ACM Press, 1973.
- [7] 丁国栋, 白硕, 王斌. 一种基于局部共现的查询扩展方法[J]. 中文信息学报, 2006, 20(3): 84-91.
- [8] 李勇, 李苹. 主题词表到领域本体的转化研究[J]. 现代计算机, 2013(5): 12-15.