文章编号:1671-9352(2008)01-0060-05

一种用于文本聚类的改进 k-means 算法

索红光[1,2],王玉伟[2]
(1. 北京理工大学计算机科学技术学院, 北京 100081;
2. 中国石油大学计算机与通信工程学院, 山东 东营 257061)

摘要:k-means 是目前常用的文本聚类算法,针对其最终搜索的局部极值与全局最优解偏差较大的缺点,采用一种基于局部搜索优化的思想来改进算法,并推导出目标函数的变化公式。根据目标函数值的改变对聚类结果作再次划分后,继续 k-means 迭代,拓展其搜索范围。理论分析和实验结果表明修改后的算法能有效地提高聚类的质量,且计算复杂度仍与数据集文本总数呈线性变化。

关键词:文本聚类;k-means;向量空间模型;局部迭代
中图分类号:TP391 **文献标志码**:A

An improved k-means algorithm for document clustering

SUO Hong-guang[1,2], WANG Yu-wei[2]
(1. School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China;
2. School of Computer & Communication Engineering, China University of Petroleum, Dongying 257061, Shandong, China)

Abstract: The k-means algorithm is a popular method for document clustering, but it often gets stuck at a local maximum far from the optimal solution. A procedure based on local search was used to improve this algorithm. The formula about object function change was also deduced, which can be used to again partition the clustering. This procedure makes appropriate iterations to enlarge the search space. Theory analysis and experimental results show that the improved algorithm efficiently improves k-means clustering and its computation is also linear in the size of document collection.

Key words: document clustering; k-means; vector space model; local iteration

0 引言

聚类是将物理或抽象对象集合按有关特性的相似程度进行分组的过程。聚类产生每一组数据称为一个簇,簇中每一个数据称为一个对象。聚类的目的是使同一簇中对象的特性尽可能地相似,而不同簇对象之间的差异尽可能地增大。聚类作为一种无监督的学习方法,能从数据集中发现数据的分布情况,是一种强有力的信息处理方法。随着 www 以及各种文本资源的不断增长,文本聚类也得到越来越多的重视。

k-means 算法[1]是由 MacQueen 提出,该算法及其推广是数据挖掘及知识发现领域中的一种重要的方法。该算法简单且收敛速度快,但也有着明显的缺点[2]:该算法是基于目标函数的算法,通常采用梯度法求解极值,由于梯度法的搜索方向是沿着能量减小的方向进行,使得算法很容易陷入局部极值。文献[2]同时还利用免疫规划改进 k-means 算法,克服其易于陷入极值的特点。

在文本聚类的应用方面,Dhillon[3]将其用于文本聚类中,并利用余弦相似度来计算对象间的距离,也是目前常用的文本聚类算法。但对于文本这种特殊的对象,其维度高,具有稀疏性,不同簇之间相似度的差异性比较大,因此可能导致聚成一簇的文本之间的非相似性[4],传统 k-means往往更容易陷入

收稿日期:2007-09-05
作者简介:索红光(1966-),男,博士研究生,副教授,主要研究方向为自然语言处理. Email: suohg@163.com

局部最值,导致较差的聚类结果。本文的目的是提高 k-means 文本聚类结果的质量,并保持其较快的执行效率。

1 文本预处理

文本聚类可以描述为:对一个给定的文本集合 $D=\{d_1,d_2,\cdots,d_n\}$,最终要得到一个簇的集合 $C=\{C_1,C_2,\cdots,C_k\}$,$\bigcup_{i=1}^{k} C_i=D$,使得对每一个 $\forall d_i(d_i\in D),\exists C_j(C_j\in C),d_i\in C_j$,同时还要使得目标函数 $Q(C)$达到最值,其中 n 为文本的总数目,k 为聚类最终的个数,且 $C_j\cap C_l=\varnothing, j\neq l$。

1.1 文本的特征选取及表示

通常采用向量空间模型(vector space model, VSM)[5]来表示每个文本。在该模型中,每个文本 $\boldsymbol{d}$ 被认为是向量空间中的一个向量。本文使用 tfidf[6] 作为特征向量的度量,该度量给出文本中每个词 t 的权重,权重的计算如下:

$$\text{tfidf}(\boldsymbol{d},t)=\text{tf}(\boldsymbol{d},t)*\log_2\frac{N}{\text{df}(t)}。\tag{1}$$

其中,$\text{tf}(\boldsymbol{d},t)$是词 t 在文本 $\boldsymbol{d}$ 中的词频,$\text{df}(t)$是文本集合 D 中包含词 t 的所有文本的数目,N 为文本总数。特征选取之后,文本 $\boldsymbol{d}\in D$ 表示为向量的形式,各维的值为相应的 $\text{tfidf}(\boldsymbol{d},t)$权重值,则文本可表示为

$$\boldsymbol{d}=\{(\text{t}_i,\text{tfidf}(\boldsymbol{d},t_i))\mid 1\leqslant i\leqslant m\}。\tag{2}$$

其中 t_i 为词条,m 为特征向量的维度。但经过特征选取后 m 仍是非常庞大的,少则几千维,多达几万维;而对应每个文本向量的非零词频却较少,这就导致了文本 VSM 表示模型的高维、稀疏性。

1.2 相似度的定义

本文使用余弦距离度量[7]文本之间的相似度,它定义 2 篇文本 $\boldsymbol{d}_1,\boldsymbol{d}_2$ 的相似度如式(3)。

$$\text{Sim}(\boldsymbol{d}_1,\boldsymbol{d}_2)=\cos(\boldsymbol{d}_1,\boldsymbol{d}_2)=\frac{\boldsymbol{d}_1\cdot\boldsymbol{d}_2}{\|\boldsymbol{d}_1\|\cdot\|\boldsymbol{d}_2\|}。\tag{3}$$

为了减小不同长度的文本对于计算文本相似度的影响,每个文本向量都被归一化到单位长度,即 $\boldsymbol{d}=\boldsymbol{d}/\|\boldsymbol{d}\|=$

$$\{\text{tfidf}(\boldsymbol{d},t_1),\text{tfidf}(\boldsymbol{d},t_2),\cdots,\text{tfidf}(\boldsymbol{d},t_m)\}/$$

$$\sqrt{\text{tfidf}(\boldsymbol{d},t_1)^2+\text{tfidf}(\boldsymbol{d},t_2)^2+\cdots+\text{tfidf}(\boldsymbol{d},t_m)^2}。$$

于是 $\|\boldsymbol{d}\|=1$,其余弦相似度即为 2 文本向量的点积,即 $\text{Sim}(\boldsymbol{d}_1,\boldsymbol{d}_2)=\boldsymbol{d}_1\cdot\boldsymbol{d}_2$。

2 用于文本聚类的 k-means 算法

k-means 方法是基于划分的聚类方法。其基本思想为:对于给定的聚类数目 k,首先随机选择 k 个文本作初始的类中心,然后根据每个文本与各个类质心的相似度,将它赋给最相似的类。然后重新计算每个类的质心。不断迭代以上过程,直到目标函数收敛。在文本聚类中一般采用的目标函数[8]为:

$$Q(C)=\sum_{j=1}^{k}q(c_j)=\sum_{j=1}^{k}\sum_{d\in C_j}\boldsymbol{d}^{\mathrm{T}}Z(C_j)。\tag{4}$$

其中 $q(C_i)$[3]为对应簇 C_i 内部聚类(intra-cluster)的相似度(similarity)。

$$q(C_i)=\sum_{d\in C_i}\boldsymbol{d}^{\mathrm{T}}Z(C_i)=\|S(C_i)\|。\tag{5}$$

式(4)的目标函数是基于余弦相似度的。如果文本 $\boldsymbol{d}$ 及中心作归一化后,采用欧式距离和采用余弦相似度的方法效果是等同的[9]。

如第 1 章中介绍,算法输入为:文本集合 D;输出为:簇的集合 C;定义 C 的复合向量 $\boldsymbol{S}(C_i)=\sum_{d\in c_i}d$,集合 C 的中心 $Z_i=Z(C_i)=\boldsymbol{S}(C_i)/\|\boldsymbol{S}(C_i)\|$;记第 t 次迭代产生的簇的集合记为 $C^{(t)}$,$t=0,1,2\cdots$,下面给出传统文本聚类 k-means 算法的流程[3]:

传统 k-means 算法:

(1) 随机选择初始中心向量 $Z_1^{(0)},Z_2^{(0)},\cdots,Z_k^{(0)}$,初始迭代次数 $t=0$;

(2) 每个文本向量 $\boldsymbol{d}\in D$ 与中心向量 $Z_j^{(t)}$ 比较,根据其相似程度将其分配到最近的一类中,即:$j^*=\arg\max_j\boldsymbol{d}^{\mathrm{T}}Z_j^{(t)}$,$1\leqslant j\leqslant k$,$t$ 为迭代次数。

(3) 由步骤(2)得到新的簇分区集合 $C^{(t+1)}=\text{nextKM}(C^{(t)})$,计算新的中心向量 $Z_j^{(t+1)}$,$1\leqslant j\leqslant k$。

(4) 如果 $Q(C^{(t+1)})-Q(C^{(t)})>\varepsilon$($\varepsilon>0$,为判断终止的阈值),则 $t=t+1$,转到(2);否则终止算法,输出聚类最终簇的集合 C^*。

3 算法修改

由于 k-means 算法容易陷入局部最值的缺点,本文对 k-means 算法进行二次分区调整,调整完毕后,重新进行 k-means 启发式的搜索过程。调整过程的基本思路为:当 k-means 算法陷入局部极值时,将聚类分区中的文本向量与 k-means 算法生成的其它中心比较,如果将该点从当前簇移动至其它簇中时,会使目标函数改变满足一定条件,则移动该向

量。因此,本文对从簇 i 移动到簇 j 中去目标函数的变化作了计算。

3.1　目标函数的变化

对分区进行修改,将一个向量 y 从其所属簇 C_i 移至另一个簇 C_j 中去,则目标函数的变化记为:

$$\Delta Q = Q(C^{(t+1)}) - Q(C^{(t)}) = \Delta q_i + \Delta q_j。$$

由式(5)可推得

$$\begin{aligned}\Delta q_i &= q(C_i - \{y\}) - q(C_i) = \\ &\| S(C_i - \{y\}) \| - \| S(C_i) \| = \\ &\| \sum_{d\in C_i} d - y \| - \| \sum_{d\in C_i} d \| = \\ &\| S(C_i) - y \| - \| S(C_i) \| = \\ &((S(C_i) - y)(S(C_i) - y)^{\mathrm{T}})^{1/2} - \| S(C_i) \| = \\ &(\| S(C_i) \|^2 - 2y^{\mathrm{T}} \cdot S(C_i) + 1)^{1/2} - \| S(C_i) = \\ &(\| S(C_i) \|^2 - 2 \| S(C_i) \| \cdot y^{\mathrm{T}} Z(C_i) + 1)^{1/2} - \\ &\| S(C_i) \|。\end{aligned} \tag{6}$$

同理可得

$$\begin{aligned}\Delta q_j &= q(C_j \cup \{y\}) - q(C_j) = \\ &(\| S(C_j) \|^2 + 2 \| S(C_j) \| \cdot y^{\mathrm{T}} Z(C_j) + 1)^{1/2} - \\ &\| S(C_j) \|。\end{aligned} \tag{7}$$

3.2　修改算法及复杂度分析

(1) 利用传统 k-means 算法得到簇的集合。

(2) 由公式(6)(7)计算文本向量 $\boldsymbol{d} \in D$ 从簇 C_i ($\boldsymbol{d}$ 所在的簇)移动至簇 C_j 时的 $\Delta Q_j (1 \leqslant j \leqslant k, j \neq i)$,若 $\max \Delta Q_j > \varepsilon' (\varepsilon' > 0)$,则将相应文本移动到当前簇 C_j 中,记录当前 C_i, C_j 为发生改变的簇,否则不作改变。重复步骤(2),直到遍历完所有文本,其中每个文本只遍历一次。

(3) 由步骤(2)得到新的簇分区集合记为 $C^{(g+1)} = \mathrm{nextMov}(C^{(g)})$,更新经过移动修改的簇的中心向量 $Z_j^{(g+1)}, 1 \leqslant j \leqslant k$,其中 g 为算法调整的次数。

(4) 如果所有 $\Delta Q < \varepsilon' (\varepsilon' > 0$,为判断终止的阈值),终止算法,输出聚类最终簇的集合 C;否则 $g = g + 1$,转到步骤(1)。

该算法涉及这样几个关键问题:

(a) 调整算法中对 ΔQ 的计算都是基于当前的数据(公式(6)(7)),不需要预先进行下一步的迭代。即:从第 t 次迭代直接得出 $t+1$ 次的结果。

(b) 步骤(2)中会对多个向量同时进行移动,避免对 k-means 产生的分区改变的效果不明显和消耗过多的时间进行迭代,所以步骤(2)过程中并不更新中心向量,而在步骤(3)中更新经过移动的簇的中心向量。

(c) 上述调整算法步骤(2)~(4)与传统 k-means 算法是相似的,都是与某一值进行比较,不同的是传统 k-means 是将文本向量与聚类中心的相似度 $x^{\mathrm{T}} C_j^{(t)}$ 比较,而修改算法是将文本向量移至对应聚类中心后 ΔQ 的比较,因此其计算复杂度也应为 $O(knG)$,其中 k 为聚类的个数,g 为修改算法的迭代次数。

当考虑到步骤(1)~(4)后整个改进算法迭代次数为:

$$T = t + g + \sum_{i=1}^{g} t'。 \tag{8}$$

其中 t' 为调整算法后步骤(1)的迭代次数。所以算法的复杂度应为 $O(knT)$, $k \ll n$, $T \ll n$,试验结果表 4 中将讨论算法的复杂度。

3.3　算法迭代的自动确定

由于搜索到 k-means 最优解的问题属于 NP-完全问题[10],因此在应用中搜索到一个局部的最优值即可。

根据实验结果分析,当调整算法已经无法跳出局部极值时,会出现 3.2 节中算法步骤(1)的迭代次数仅为一次,以及分区调整算法中出现文本向量反复移动的问题,如向量 $\boldsymbol{y}$ 从其所属簇 C_i 移至另一个簇 C_j 中去,下一次调整迭代过程又将向量 $\boldsymbol{y}$ 移动回去。如果这样的情况出现次数较多(本文实验中设为 3 次),算法即可提前终止。记录步骤(1)的迭代次数以及相应向量编号的移动,进行上述分析,可设定迭代次数。

因此,对整个修改算法的迭代次数 g,可根据 3.2 节中算法步骤(4)的条件终止或限制最大迭代次数,或根据上述设定自动调整,具有较高的效率。

4　实验和算法评估

本文将传统的聚类算法与修改后的算法在聚类效果以及计算复杂度等方面进行比较。

4.1　测试数据集

本文测试语料采用了搜狗实验室 2006 年 11 月发布的"文本分类语料库"。该语料库来源于 Sohu 新闻网站保存的大量经过编辑、手工整理与分类的新闻语料。其分类体系包括几十个分类节点,网页规模约为十万篇文本。本文从其中抽取了 10 个数据集进行测试。

另外,实验中采用中科院 ICTCLAS 中文分词系统,对文本进行分词处理。

4.2　评估标准

F 度量[11]是一种参照信息检索的评测方法，将每个聚类结果看作是查询的结果，这样对于最终的某一个聚类类别 r 和原来的预定义类别 i，得出：

$$\text{准确率 precision}(i,r) = n_{ir}/n_r, \quad (9)$$

$$\text{召回率 recall}(i,r) = n_{ir}/n_i。 \quad (10)$$

这里，n_{ir} 是聚类 r 中包含类别 i 中的文本的个数，n_r 是聚类类别 r 中实际对象的数目，n_i 是原来预定义类别 i 应有文本数。则聚类 r 和类别 i 之间的 f 值计算如式(11)。

$$f(i,\ r) = \frac{2\text{recall}(i,r) * \text{precision}(i,r)}{\text{recall}(i,r) + \text{precision}(i,r)}。 \quad (11)$$

最终聚类结果的评价函数为：

$$F = \sum_i \frac{n_i}{n} \max\{f(i,r)\}。 \quad (12)$$

其中 n 为所有测试文本的个数。

4.3 实验结果

由于项目需求本文采用 Java 语言编写，在 CPU3.06 GHz，512 MB 内存环境下测试。两算法的初始化聚类中心选择相同，人工指定相应的初始化中心。测试语料中抽取了 10 个文本数目（见表 1）递增的数据集进行测试，每个数据集包含 5 个类别。测试语料中的文本篇幅大小差异较大，长文本的字数可达 3 000 左右，而短的只有 290 左右，具有一般性。

表 1 文本测试数据集

Table 1 Document collection for test

数据集	文本数目
DS1	50
DS2	150
DS3	200
DS4	300
DS5	400
DS6	500
DS7	750
DS8	1 000
DS9	2 000
DS10	3 000

传统算法和改进的文本聚类算法总体 F 值比较，如图 1 所示。其中文本数目为 750 的数据集聚类效果较差，是因为初始聚类中心的选择较差，导致整体聚类效果降低。

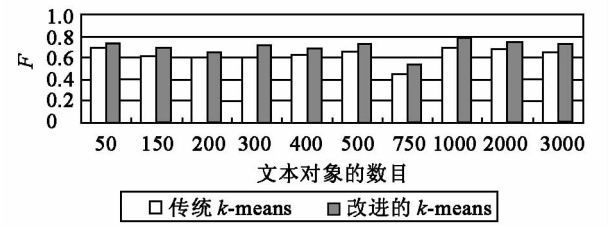

图 1 聚类效果的比较

Fig.1 Comparison of clustering results

实验结果用表 2 和表 3 中的数据矩阵的形式说明，$E(i,r)$ 表示实际聚类 r 包含预定义类别 i 中文本的个数，即 $E(i,r) = n_{ir}$；对角线中的数据即为正确聚类的个数。表 2、表 3 是文本数为 1 000 篇数据集的数据矩阵表，其中 Max$f$ 表示局部 f 值（见公式(11)）。

表 2 传统 k-means 聚类结果

Table 2 Result of k-means clustering

E	$r1$	$r2$	$r3$	$r4$	$r5$	Max$f$
$i1$	76	74	21	28	1	0.505 0
$i2$	13	133	23	11	20	0.633 3
$i3$	5	4	141	22	28	0.715 7
$i4$	3	4	4	184	5	0.808 8
$i5$	4	5	5	10	176	0.818 6

表 3 修改算法聚类结果

Table 3 Result of improved k-means clustering

E	$r1$	$r2$	$r3$	$r4$	$r5$	Max$f$
$i1$	122	49	17	11	1	0.734 9
$i2$	5	159	11	10	15	0.726 0
$i3$	2	5	150	19	24	0.777 2
$i4$	0	11	2	183	4	0.853 1
$i5$	3	4	6	6	171	0.824 1

表 4 为两算法迭代次数和调整次数的比较，实验中限制 g 的最大迭代次数为 20 次。如数据集 DS3 所示，当 $g = 20$，其聚类效果实际与 $g = 10$ 时相同，也就是会碰到这样的情况，调整算法调整后，又被 k-means 算法迭代回去了，然后反复；此时应采用 3.3 中的方法自动确定算法调整次数 g 的值。

在 4.3 节中注释(c)中讨论了算法的复杂度为 $O(knT)$，其中算法整体迭代次数 T 如公式(8)所示。从表 4 中的数据可以看出，当遇到如数据集 DS3 所示的情况后，经过调整后，发现通常 k-means 的迭代次数仅为 1，即 $t' = 1$；此时 $T = t + 2g$。

而数据集 DS8 代表当调整次数 $g = 11 < 20$，算法自动终止时的情况；经过统计发现，仅有几次调整后 k-means 的迭代次数超过一次，此种情况是因为调整时能够让 k-means 算法跳出此时的局部最值，所以 k-means 迭代次数较多，而 k-means 迭代次数少的情况只不过是调整算法在进行局部的最优化处理。

经过上述 2 种情况的讨论，可以得出：当数据集较大时，$T \ll n$，所以改进后的算法仍然与文本总数 n 成线性变化，从表 4 中数据也说明了这点。另外从表 4 中还能看出当文本数目较大时调整算法不易陷入反复迭代的过程，说明改进后的算法也是适用于数据集较大的情况。

表 4　两聚类算法迭代次数比较

Table 4　Iteration times comparison of two kinds of clustering algorithms

数据集	文本数目 n	传统迭代次数 t	算法调整次数 g	算法改进后总次数 T
DS1	50	3	20	43
DS2	150	3	5	15
DS3	200	5	20	45
DS4	300	8	5	20
DS5	400	5	20	45
DS6	500	6	13	32
DS7	750	7	20	47
DS8	1000	8	11	37
DS9	2000	8	9	35
DS10	3000	9	10	38

通过实验,还发现初始中心的选择在 k-means 算法中非常重要:初始中心的选择不仅决定了聚类效果的好坏,还直接决定了 k-means 以及改进算法的迭代次数。中心点选择较差的时候,算法迭代次数明显增多,如对数据集 DS7 的聚类结果。因此,可以得出这样的结论:较优初始中心选择的设计算法,要优先于对 k-means 算法进行复杂地局部迭代,这也是本文没有采用较复杂的局部迭代算法的原因。本文的下一步工作将对初始中心的选择进行改善。

5　结论

本文介绍了一种基于局部迭代调整改进的文本聚类方法,由于文本数据高维稀疏的特点,k-means 方法较易陷入局部极值,对其进行二次划分,让其重新搜索最优解,以提高聚类质量,同时对改进算法的迭代次数进行自动调整。理论分析和实验结果表明改进算法能有效的提高聚类的质量,且仍具有与数据文本总数呈线性变化的时间复杂度,同时对较大的文本集合也是有效的。

参考文献:

[1] MACQUEEN J. Some methods for classification and analysis of multivariate observations[C]// Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, 1967: 281-297.

[2] 行小帅,潘进,焦李成. 基于免疫规划的 K-means 聚类算法[J]. 计算机学报, 2003, 26(5):605-610.

[3] DHILLON I S, MODHA D S. Concept decompositions for large sparse text data using clustering[J]. Machine Learning, 2001, 42(1):143-175.

[4] 张猛,王大玲,于戈.一种基于自动阈值发现的文本聚类方法[J].计算机研究与发展,2004,41(10):1748-1753.

[5] SALTON G, WONG A, YANG C S. A vector space model for automatic indexing[J]. Communication of the ACM, 1975, 18(5):613-620.

[6] 刘涛.用于文本分类和文本聚类的特征选择和特征抽取方法的研究[D].天津:南开大学计算系,2004.

[7] STEINBACH M, KARYPIS G, KUMAR V. A comparison of document clustering techniques[C]// Proceedings of the 6th ACM-SIGKDD International Conference on Text Mining. Boston, MA, USA: ACM Press, 2000: 103-122.

[8] LARSEN B, AONE C. Fast and effective text mining using linear time document clustering[C]// Proceedings of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. San Diego, California: ACM Press, 1999: 16-22.

[9] SHI Zhong. Efficient online spherical K-means clustering[C]// Proceedings of the 2005 IEEE International Joint Conference on Neural Networks. Montreal, Canada: IEEE Press, 2005: 3180-3185.

[10] DHILLON I S, GUAN Y, KOGAN J. Iterative clustering of high dimensional text data augmented by local search[C]// Proceedings of the 2002 IEEE International Conference on Data Mining. Maebashi, Japan: IEEE Press, 2002: 131-138.

[11] 刘远超, 王晓龙, 徐志明,等. 文本聚类综述[J].中文信息学报, 2006, 20(3): 55-62.

(编辑:孙培芹)