第40卷 第6期
Vol.40 No.6

计 算 机 工 程
Computer Engineering

2014年6月
June 2014

·人工智能及识别技术· 文章编号：1000-3428(2014)06-0195-06 文献标识码：A 中图分类号：TP18

一种基于簇相合性的文本增量聚类算法

陶舒怡[1]，王明文[1]，万剑怡[1]，罗远胜[2]，左家莉[3]

(1. 江西师范大学计算机信息工程学院，南昌 330022；2. 江西财经大学网络信息管理中心，南昌 330013；
3. 江西师范大学初等教育学院，南昌 330027)

摘 要：传统文本聚类方法只适合处理静态样本，且时间复杂度较高。针对该问题，提出一种基于簇相合性的文本增量聚类算法。采用基于词项语义相似度的文本表示模型，利用词项之间的语义信息，通过计算新增文本与已有簇之间的相合性实现对文本的增量聚类。增量处理完部分文本后，对其中错分可能性较大的文本重新指派类别，以进一步提高聚类性能。该算法可在对象数据不断增长或更新的情况下，避免大量重复计算，提高聚类性能。在20 Newsgroups数据集上进行实验，结果表明，与k-means算法和SHC算法相比，该算法可减少聚类时间，提高聚类性能。

关键词：文本聚类；增量聚类；语义相似度；簇相合性；文本再分配

An Incremental Text Clustering Algorithm Based on Cluster Congruence

TAO Shu-yi[1], WANG Ming-wen[1], WAN Jian-yi[1], LUO Yuan-sheng[2], ZUO Jia-li[3]

(1. School of Computer Information Engineering, Jiangxi Normal University, Nanchang 330022, China;
2. Network Information Management Center, Jiangxi University of Finance and Economics, Nanchang 330013, China;
3. School of Elementary Education, Jiangxi Normal University, Nanchang 330027, China)

【Abstract】 Traditional text clustering methods are only suitable for static sample, and their time complexity is too high. Aiming at these problems, this paper proposes a new Incremental Text Clustering Algorithm Based on Congruence(ITCAC) between text and cluster. The new algorithm can avoid a lot of double counting to improve the performance of clustering. It uses text representation model based on semantic similarity of lexical items, fully takes the semantic information between terms into account and computes the congruence between new documents and existing clusters. After processing part of the documents, the algorithm reassigns the categorization of documents that has large possibility of misclassification to further improve the clustering performance. Experimental results on 20 Newsgroups datasets show that, compared with the k-means algorithm and SHC algorithm, the new algorithm not only has less clustering time, but also has better performance of clustering.

【Key words】 text clustering; incremental clustering; semantic similarity; cluster congruence; text redistribution

DOI: 10.3969/j.issn.1000-3428.2014.06.042

1 概述

随着互联网技术的发展和普及，当前发布的在线网页已达到亿数量级，且仍以每天数百万的速度增长。网上信息大多又以文本的形式存在。因此，人们迫切需要一些对文本进行组织和管理的工具。文本聚类正是其中的一种技术。在如今数据日益膨胀的背景条件下，绝大多数实际应用都需处理高维、大规模数据。其中有些数据规模太过庞大，不能一次完全处理，例如微博数据；有些则处于时刻更新的状态，例如银行交易数据。传统聚类方法往往不能很好地处理这种动态数据。而增量聚类在数据更新时，不需对整个数据集重新聚类，只处理增量部分，即可实现聚类结果的更新和扩充。因此，本文提出一种基于簇相合性的文本增量聚类算法(Incremental Text Clustering Algorithm Based on Cluster Congruence, ITCAC)。

2 相关工作

当前大部分文本聚类算法使用向量空间模型。近年来出现了一些基于Word Net[1]作为本体以提高文本聚类性能的研究。例如，文献[2]提出的基于词项语义相似度的文本

基金项目：国家自然科学基金资助项目(61272212)。

作者简介：陶舒怡(1988－)，女，硕士研究生，主研方向：信息检索，数据挖掘；王明文，教授、博士生导师；万剑怡，教授；罗远胜，讲师、硕士；左家莉，讲师、博士。

收稿日期：2013-02-28 **修回日期**：2013-07-23 **E-mail**：taosy12@163.com

表示方法，它通过捕获词项之间的语义信息，以能够更准确地分配词项频率权重。

传统聚类算法一般采用批量处理的模式，即每当数据更新时，算法必须对整个数据集进行重新聚类，其中最典型的就是 k-means 算法。但随着文档数的增加，传统聚类算法的时间复杂度也随之增加，聚类效果却大大降低。在信息数据日益膨胀的背景条件下，人们逐步开始了对增量聚类的研究。常见增量聚类算法有基于密度(如 DBSCAN)和基于网格(如 STING)等。文献[3]提出了基于直方图的聚类算法(HC)，并使用簇内文档间相似度分布情况代表簇；此后文献[4]又提出了基于短语语义相似度直方图的增量聚类算法(SHC)。该算法首先计算每个簇的直方图比例，即簇中相似度超过一定阈值的文档对占所有文档对的比例；然后判断新增文本是否会增加簇的直方图比例。如果是，则将文本加入该簇，否则该文本将单独成簇。

文献[5]对文本增量聚类进行研究，提出了基于簇特征的文本增量聚类算法，首先利用 k-means 进行初始聚类，保留聚类后每个簇的簇心、均值、方差、3 阶和 4 阶中心矩等特征信息；然后根据判断新增数据能否提高簇的特征信息，从而实现有效聚类。文献[6]提出了领导者算法，根据文档与领导者之间的相似性进行增量聚类。文献[7]利用领导者算法初始聚类，然后使用层次凝聚聚类算法对得到的结果进行调整和修正，由此提出了一种结合增量和层次聚类的混合算法。文献[8]提出了图增量聚类方法，通过考虑图的拓补结构及属性间的相似度，达到降低时间复杂度的效果。文献[9]等在聚类过程中由用户指定约束条件，提出了一种高效约束增量聚类方法。

本文提出的 ITCAC 算法，主要通过计算新增文本与已有簇之间的相合性对文本进行聚类。不但定义新的簇相合性计算方法，此外还提出在增量聚类过程中对错分可能性大的文本重新指派类别的处理方法。

3 基于簇相合性的文本增量聚类

3.1 文本表示方法

向量空间模型被广泛地应用于文本聚类技术中，它利用特征空间向量表示文本，一般将文中独立词项的出现频率作为特征，具体可表示为：

$$\boldsymbol{d}_j=\{w_1,w_2,\cdots,w_n\}$$

其中，w_i代表文档 j 中词项 i 的频率权重。这种模型只根据词项出现频率确定其权重显然不够准确，并不能获取文本中各词项之间的语义关系。近年出现了将 WordNet 作为本体用以提高聚类效果的研究。

本文使用基于词项语义相似度的文本表示模型[2]，根据词项之间的相似度，对词项频率权重进行调整。该模型表示为：$\boldsymbol{d}_j=\{sw_1,sw_2,\cdots,sw_n\}$，其中，$sw_i$表示文本 j 中词项 i 的语义权重，计算方法为：$sw_i=w_i+\sum_{k\neq i}^{n}w_k\cdot Sim(i,k)$，其中，$Sim(i,k)$表示词项 i 和 k 基于 WordNet 的相似度[10]。

3.2 簇信息度量

对于每个簇保存其簇心、均值、方差和文档数这 4 种信息。除文档数外，其余簇信息的计算方法如下：

$$M_j=\frac{1}{|C_j|}\sum_{d_i\in C_j}d_i \tag{1}$$

$$Mean_j=\frac{1}{|C_j|}\sum_{i=1}^{|C_j|}Dis(d_i,M_j) \tag{2}$$

$$Var_j=\frac{1}{|C_j|}\sum_{i=1}^{|C_j|}\left(Dis(d_i,M_j)-Mean_j\right)^2 \tag{3}$$

其中，$|C_j|$表示 C_j 簇中包含的文档数；$Dis(d_i, M_j)$表示文档 d_i 与 C_j 簇心之间的距离，计算方法如下：

$$Dis(d_i,M_j)=\sqrt{\sum_{x\in d_i,y\in C_j}(x-y)^2} \tag{4}$$

其中，x 和 y 分别表示文档 d_i 与 C_j 簇心中包含的词项权重。

3.3 文本与簇的相合性

相合性的概念首先由文献[11]提出，而文献[12]定义了一种新的计算文本与簇相似性的方法。在此基础上，本文提出了一种新的计算文本与簇相合性的方法，它不仅计算了文本与簇的相似度，而且还考虑了簇分布的统计特征。因此，它能够更准确地衡量文本与簇的相关性。

3.4 增量聚类阶段

首先根据式(5)计算新增文本与每个簇之间的相合性；然后根据式(8)选出与文本相合性最高的簇；最后判断最大相合性的值是否大于阈值β，如果大于，就将文本加入到簇 $C_{\max}$ 中，并更新簇特征信息；否则该文本将成为一个新的簇，簇类总数加 1。循环上述过程，直至处理完所有文本。

$$Score(d_i,M_j)=Join(d_i,M_j)+\sum_{1\leqslant k\leqslant N}^{k\neq j}Sim(M_j,M_k)\times Join(d_i,M_k) \tag{5}$$

其中：

$$Sim(M_j,M_k)=Cos(M_j,M_k) \tag{6}$$

$Join(d_i, M_j)$表示文本与簇之间的连接性，计算方法为：

$$Join(d_i,M_j)=\frac{Sim(d_i,M_j)}{1+Var_j} \tag{7}$$

$$C_{\max}=\arg\max_{1\leqslant j\leqslant N}Score(d_i,M_j) \tag{8}$$

上述聚类过程即本文提出的 ITCAC 算法，具体描述如下：

算法 1 ITCAC 算法

输入 M 篇新增文档

输出 N 个文档聚类结果

1. 得到第 i 篇新增文档 d_i；
2. 使用式(8)选取 $C_{\max}$ 簇；
3. If Score(d_i, $M_{\max}$) > β then
   将 d_i 加入到簇 $C_{\max}$ 中；

更新簇 C_{max} 的簇信息；

Else

d_i 单独成为一个簇；

Endif

3.5 文本再分配

增量聚类算法往往存在一个问题，即一旦将文本分配到特定簇后，聚类结果就不能改变。这意味着如果文本被错分，错误会一直延续下去，并最终导致簇心偏离，聚类性能下降。增量聚类处理的数据是动态更新的，有些文本当前最可能属于这个簇，但加入新增文本后，它可能更适合另一个簇。因此，在增量聚类过程中，文本错分是不可避免的，但可以对部分不确定类别的文本再分配，以起到调整和修正的作用。

对此，本文在得到与文本相合性最大的簇 C_{max} 的同时，也计算出取得次大值的簇 C_{sec}，然后判断最大值和次大值之间的差异是否大于阈值 δ。若大于，证明该文本最适合加入到簇 C_{max} 中；否则说明还需更多信息进一步判断该文本最适合的簇。但文本的簇标仍记为 C_{max}，且簇 C_{max} 各特征信息不变，将该文本标记为离开簇的候选文本。批量处理完增量数据后，对簇中不确定分类的文本进行重新分配。此即为增加文本重新分配的调整后的算法AITCAC，具体描述如下：

算法 2 AITCAC 算法

输入 M 篇新增文档

输出 N 个文档聚类结果

1. 得到第 i 篇新增文档 d_i；
2. 使用式(8)选取 C_{max} 和 C_{sec} 簇；
3. If Score(d_i, M_{max}) > β then

If (Score(d_i, M_{max}))–Score(d_i, M_{sec})) > δ

将 d_i 加入到 C_{max} 中；

更新簇 C_{max} 的簇信息；

Else

将 d_i 加入到簇 C_{max} 中，标记为离开该簇的候选文档；

Endif

Else

d_i 单独成为一个簇；

Endif

4 实验结果及分析

4.1 实验数据及预处理

实验中使用了 20 Newsgroups 数据集，该数据集包含了 20 个类别共 19 997 篇文档，其中，soc.religion.christian 类别只包含 997 篇文档，其余每个类都有 1 000 篇。

在进行实验之前，首先要对所有的文本进行预处理，包括去除停用词、不频繁词项，词项归一化等；然后计算每篇文本的词项频率并得到文本词项语义向量表示模型。为保证实验结果的正确、有效，实验中所有方法都使用该预处理方式。

实验主要分成 2 个部分：(1)对随机选取的 10 个类别共 10 000 篇文档进行聚类；(2)对全部 20 个类别共 19 997 篇文档进行聚类。将本文提出的 ITCAC 和 AITCAC 算法与 k-means 算法、SHC 算法进行实验结果对比。

4.2 评价指标

实验采用文本聚类文献中最广泛使用的 3 种评价指标，即纯度、熵、归一化互信息。

准确率是检索结果中相关文档数占所有结果的比例，簇的准确率可以定义为：

$$P(i,j)=\frac{N_{ij}}{N_i}$$

其中，N_{ij} 表示簇 i 中类 j 的文档数；N_i 表示簇 i 中的文档数。

纯度则是计算每个类 j 最大准确率加权平均值，纯度越高，聚类效果越好。

$$Purity=\sum\frac{N_i}{N}\max P(i,j)$$

其中，N 为总的文档数。

熵表示单个随机变量不确定性的均值，其值越大，则说明变量不确定性越大。每个簇的熵计算方法为：

$$E(i)=\sum p(i,j)\cdot\lg(p(i,j))$$

其中，$p(i,j)$表示簇 C_i 和簇 C_j 共现概率。所有簇的整体熵的计算方法为：

$$Entropy=\sum\frac{N_i}{N}E(i)$$

其中，N_i 表示类别 i 中的文档数；N 为总文档数。

归一化互信息常用于计算 2 个事件之间的相关性，其值越大，聚类效果越好；反之聚类效果越差，计算方法如下：

$$NMI=\frac{MI(C,C')}{\max(H(C),H(C'))}$$

其中，$MI(C,C')=\sum_{c_i\in C,c_j\in C'}p(c_i,c_j)\lg\frac{p(c_i,c_j)}{p(c_i)p(c_j)}$；$p(c_i, c_j)$，$p(c_i)$和 $p(c_j)$表示簇 C_i 和簇 C_j 共现概率、簇 C_i 概率和簇 C_j 概率；$H(C)$, $H(C')$分别表示类别 C 和类别 C'的熵，计算式为：

$$H(C)=-\sum_{c_i\in C}p(c_i)\lg p(c_i)$$

4.3 结果分析

4.3.1 10 个类别的实验

k-means 初始质心是随机挑选的，该算法极易受初始点选择的影响，并造成聚类结果不稳定。因此，本文取 k 为 10 时 5 次运行的平均结果作为 k-means 的最终聚类结果。

表 1 分别列出了 k-means、ITCAC 算法及 AITCAC 算法对 10 个类别共 10 000 篇文档聚类结果的比较。其中粗体字表示当前指标的最优值。表 1 中的 ITCAC 是算法阈值 β 取 0.045 时的聚类结果，而 AITCAC 则是阈值 β 取 0.045，

δ 取 0.02 时的结果(在此阈值条件下算法取得最好性能)。ITCAC 算法共聚得 15 个簇，而 AITCAC 共聚得 16 个簇。实验结果显示，ITCAC 算法总体性能要优于 k-means；而调整后的 AITCAC 算法又比 ITCAC 算法性能有了明显提高。这也证明文本重新分配策略确实起到了修正可能错分对簇心产生的偏离影响，并提高聚类性能的作用。

表 1 3 种算法对 10 000 篇文档的聚类结果对比

文档数	纯度			熵			归一化互信息		
	ITCAC	AITCAC	k-means	ITCAC	AITCAC	k-means	ITCAC	AITCAC	k-means
1 000	0.587	**0.689**	0.473	1.483	**1.109**	2.122	0.553	**0.666**	0.357
2 000	0.644	**0.721**	0.527	1.431	**1.102**	1.830	0.569	**0.668**	0.444
3 000	0.687	**0.744**	0.584	1.314	**1.040**	1.669	0.605	**0.687**	0.492
4 000	0.709	**0.757**	0.578	1.256	**1.012**	1.577	0.622	**0.695**	0.519
5 000	0.721	**0.762**	0.566	1.218	**1.001**	1.591	0.633	**0.699**	0.518
6 000	0.730	**0.767**	0.593	1.194	**0.989**	1.487	0.641	**0.702**	0.549
7 000	0.734	**0.770**	0.642	1.178	**0.986**	1.313	0.645	**0.703**	0.601
8 000	0.651	**0.681**	0.592	1.482	**1.311**	1.518	0.554	**0.605**	0.540
9 000	0.579	**0.605**	0.601	1.837	1.694	**1.421**	0.447	0.490	**0.572**
10 000	0.525	0.547	**0.593**	1.985	1.852	**1.466**	0.402	0.442	**0.559**

4.3.2 20 个类别的实验

20 个类别的文本聚类，ITCAC 和 AITCAC 每次增量处理 2 000 篇文本(第 10 次增量处理 1 997 篇)，共执行 10 次增量聚类；k-means 同样取 5 次运行的平均值，每次分别对 2000 至 1 9997 篇文本进行聚类，共 10 次。实验对比结果如表 2 所示，其中粗体字表示当前指标的最优值。表 2 中 ITCAC 是算法在阈值 β 取 0.075 时的聚类结果，而 AITCAC 则是阈值 β 取 0.075，δ 取 0.015 时的结果。此外 ITCAC 算法共聚得 62 个簇，而 AITCAC 共聚得 51 个簇。实验结果显示，20 个类别时 ITCAC 总体效果同样比 k-means 算法好；AITCAC 算法在 3 个指标上都要优于 ITCAC，且优势比 10 个类别时的对比情况更明显。

结合 2 次实验结果看，ITCAC 和 AITCAC 前几次聚类性能既稳定又高效，但后几次性能却骤然下降。其可能的原因是：增量聚类过程中文本错分是不可避免的，且错误会一直传播，引起簇心逐渐偏离，最终导致聚类效果降低。随着文本数量增加，聚类错误也随之累积，进而严重影响聚类性能。即使采取文本再分配策略，也只能对部分错误进行修正，并不能完全消除文本错分的影响，以至于聚类后期性能下降。

表 2 3 种算法对 19 997 篇文档的聚类结果对比

文档数	纯度			熵			归一化互信息		
	ITCAC	AITCAC	k-means	ITCAC	AITCAC	k-means	ITCAC	AITCAC	k-means
2 000	0.566	**0.703**	0.405	1.853	**1.169**	2.699	0.541	**0.712**	0.357
4 000	0.572	**0.673**	0.513	1.915	**1.390**	2.118	0.514	**0.641**	0.510
6 000	0.568	**0.658**	0.519	1.969	**1.475**	2.024	0.495	**0.608**	0.532
8 000	0.570	**0.645**	0.511	1.973	**1.556**	1.958	0.486	**0.587**	0.547
10 000	0.575	**0.639**	0.525	1.973	**1.600**	1.998	0.480	**0.575**	0.538
12 000	0.573	**0.635**	0.520	1.986	**1.624**	2.013	0.473	**0.569**	0.534
14 000	0.577	**0.628**	0.539	1.976	**1.659**	1.926	0.470	**0.560**	0.554
16 000	0.574	**0.626**	0.592	1.986	**1.674**	1.818	0.466	0.556	**0.579**
18 000	0.515	**0.560**	0.530	2.300	2.041	**1.910**	0.400	0.477	**0.558**
19 997	0.466	0.507	**0.531**	2.568	2.340	**1.930**	0.345	0.413	**0.553**

4.3.3 与 SHC 算法聚类结果的对比

本文对基于语义相似度直方图的聚类算法(SHC)[4]使用相同的数据集进行对比实验。由于 SHC 算法要计算簇中所有文档对之间的相似度，当文档数增加到一定程度时，算法的时间复杂度就会变得相当得高，即 $O(N\times N)$。且文献[4]中使用的实验数据也非常小，最大的文档集只包含了 2 619 篇文档。因此，本文只对 3 000 篇文档进行 SHC 聚类，并与 AITCAC 和 k-means 算法结果进行对比。

表 3 列出了详细的对比结果。其中，SHC 算法共产生 104 个簇，而 AITCAC 只产生了 16 个簇，k-means 则是 5 次运行的平均结果。从产生的簇数看，AITCAC 算法明显少于 SHC 算法。此外，实验结果显示，除第 1 次增量处理时，SHC 得到的纯度指标略优于 AITCAC，其余指标及之后的增量处理性能都明显低于 AITCAC，尤其是归一化互信息。这也正是因为 SHC 算法聚成的簇类太多。相比于 AITCAC 算法性能的上升趋势，SHC 算法随着文档数的增加，聚类性能不断下降。因此，可推断出 SHC 算法不适合处理大规模数据。综上所述，本文提出的 AITCAC 算法不仅能够达到比经典 k-means 算法更好的聚类性能，而且也明显优于 SHC 算法。

表3 与SHC算法的聚类结果对比

文档数	纯度			熵			归一化互信息		
	SHC	AITCAC	k-means	SHC	AITCAC	k-means	SHC	AITCAC	k-means
1 000	**0.716**	0.689	0.473	1.019	**1.109**	2.122	0.419	**0.666**	0.357
2 000	0.677	**0.721**	0.527	1.262	**1.102**	1.830	0.342	**0.668**	0.444
3 000	0.651	**0.744**	0.584	1.397	**1.040**	1.669	0.314	**0.687**	0.492

4.3.4 时间加速比

为进一步比较各算法的聚类性能，本文统计分析了运行时间。表4列出了算法对10个类别10 000篇文档聚类时间对比。其中，ITCAC是取阈值β为0.045时的运行时间，AITCAC是取阈值β为0.045、δ为0.02时的运行时间，k-means则是取5次运行的平均时间。增量时间是指算法第1次~第10次处理1 000篇新增文档的时间，而总运行时间则是指算法完成对所有数据处理的时间。从表4中增量时间上看，ITCAC10次增量处理时间都比AITCAC少。这是因为AITCAC增加了对不确定类别文档进行重新分配的处理过程。每增量处理完1 000篇文档，都要对其中标记为离开簇的候选文档重新判断最合适的簇，这个过程无疑会增加算法的运行时间。而对于总运行时间，本文提出的增量算法运行时间比k-means少得多：全部处理完10 000篇文档，k-means所用时间是AITCAC的7倍之多。且这时AITCAC算法还得到了前1 000至前9 000篇文档的聚类情况，而k-means算法要想得到全部的这些信息，则要花费AITCAC算法33倍的时间。当已知前9 000篇文档的聚类结果，增量聚类算法更能发挥它的优势。在这种情况下，k-means算法的处理时间是AITCAC算法的246倍之多。实验证明，本文算法相比于k-means，大大降低了聚类处理时间。且文档数越多，优势越明显。

表4 算法运行时间对比 s

文档数	增量时间		总运行时间		
	ITCAC	AITCAC	ITCAC	AITCAC	k-means
1 000	**55**	79	**55**	79	370
2 000	**133**	198	**188**	277	1 429
3 000	**179**	233	**367**	510	3 061
4 000	**243**	312	**610**	822	4 956
5 000	**269**	334	**879**	1 156	6 944
6 000	**266**	323	**1 145**	1 479	10 094
7 000	**281**	347	**1 426**	1 826	11 954
8 000	**276**	323	**1 702**	2 149	14 771
9 000	**310**	336	**2 012**	2 485	21 600
10 000	**322**	393	**2 334**	2 878	21 739

4.3.5 阈值影响

图1~图3分别给出了阈值在20个类别聚类时对纯度、熵、归一化互信息3种指标的影响，其中阈值β分别取0.03~0.09之间的10个值。图中显示当阈值取0.075时，算法取得最好的聚类结果，这也是用于与k-means进行比较的结果。

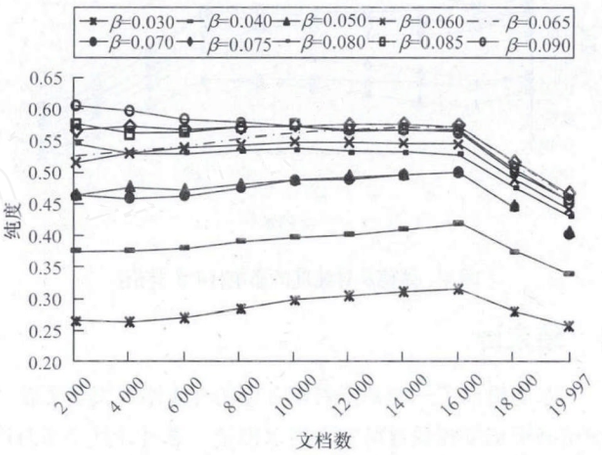

图1 阈值β对纯度的影响(20个类别)

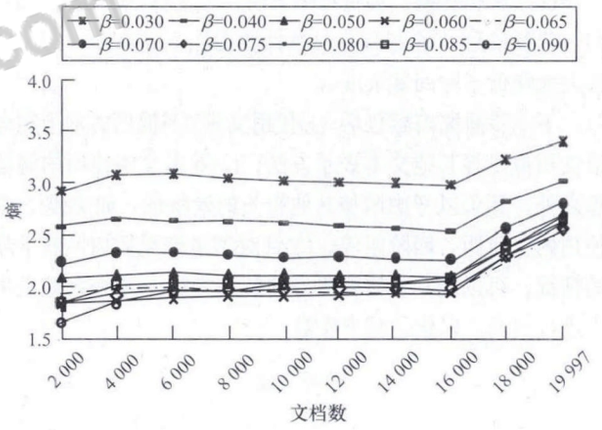

图2 阈值β对熵的影响(20个类别)

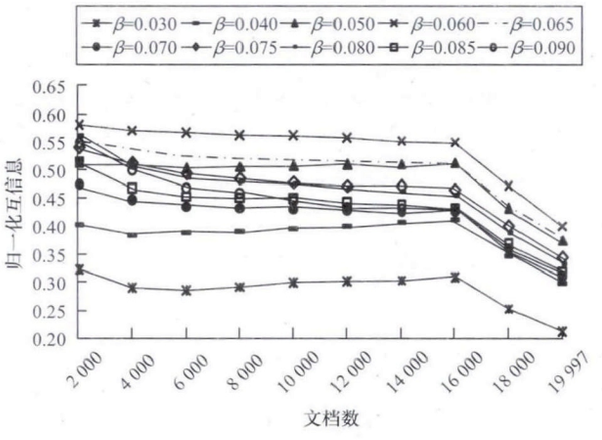

图3 阈值β对归一化互信息的影响(20个类别)

图4是10个类别聚类时阈值对纯度的影响(对熵和归一

化互信息的影响类似)，其中，阈值β分别取0.03~0.08期间的11个值。从图中可看出阈值取0.045时能够得到最好的聚类结果。

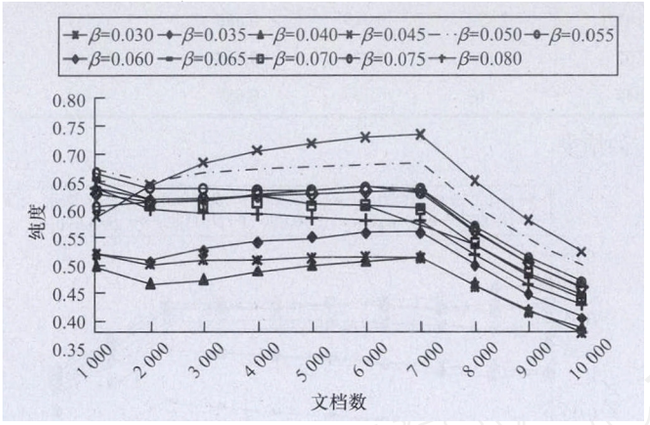

图4 阈值β对纯度的影响(10个类别)

5 结束语

本文提出了一种基于簇相合性的文本增量聚类算法。使用基于词项相似度的文本表示模型，通过计算文本与簇之间的相合性进行聚类，同时采用文本再分配策略，以进一步提高聚类性能。实验结果表明，本文算法在纯度、熵、归一化互信息3个指标上总体性能要优于k-means算法，且大大降低了时间复杂度。

下一步研究内容包括:(1)使用文献[13]提出的基于短语语义相似度等其他文本表示方法；(2)除本文中使用的簇信息之外，还可以考虑增加其他相关的簇特征，如文献[5]中使用的三阶距、四阶距等；(3)针对增量聚类后期性能下降的情况，可以进一步研究更好的解决方法；(4)在大数据集上进行实验，以优化模型参数。

参考文献

[1] Fellbaum C. WordNet: An Electronic Lexical Database[M]. Cambridge, USA: MIT Press, 1998.

[2] Gad W, Kamel M. New Semantic Similarity Based Model for Text Clustering Using Extended Gloss Overlaps[C]// Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition. Berlin, Germany: [s. n.], 2009: 663-677.

[3] Gad W K, Kamel M S. Incremental Clustering Algorithm Based on Phrase-Semantic Similarity Histogram[C]// Proceedings of the 9th International Conference on Machine Learning and Cybernetics. Qingdao, China: [s. n.], 2010: 2088-2093.

[4] Hammouda K, Kamel M. Incremental Document Clustering Using Cluster Similarity Histograms[C]//Proceedings of IEEE/WIC International Conference on Web Intelligence. Beijing, China: [s. n.], 2003: 597-601.

[5] 潘 敏. 基于簇特征的文本增量聚类研究[D]. 南昌: 江西师范大学, 2012.

[6] Vijaya P, Murthy M N, Subramanian D K. Leaders-Subleaders: An Efficient Hierarchical Clustering Algorithm for Large Data Sets[J]. Pattern Recognition Letters, 2004, 25(4): 505-513.

[7] Srinivas M, Mohan C K. Efficient Clustering Approach Using Incremental and Hierarchical Clustering Methods[C]// Proceedings of IJCNN'10. Barcelona, Spain: [s. n.], 2010: 1743-1749.

[8] Zhou Yang, Cheng Hong, Yu J X. Clustering Large Attributed Graphs: An Efficient Incremental Approach[C]//Proceedings of 2010 IEEE International Conference on Data Mining. Sydney, Australia: IEEE Press, 2010: 689-698.

[9] Davidson I, Ravi S S, Ester M. Efficient Incremental Constrained Clustering[C]//Proceedings of KDD'07. San Jose, USA: [s. n.], 2007: 240-249.

[10] Banerjee S, Pedersen T. Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet[C]//Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Text Processing. Mexico City, Mexico: [s. n.], 2002: 136-145.

[11] Lin Chengru, Chen M S. A Robust and Efficient Clustering Algorithm Based on Cohesion Self-Merging[C]//Proceedings of SIGKDD'02. Edmonton, Canada: ACM Press, 2002: 582-587.

[12] Luo Yuansheng, Wang Mingweng, Le Zhongjian, et al. An Improved KNN Text Categorization Algorithm Based on Cluster Distribution[J]. Journal of Computational Information Systems, 2012, 8(3): 1-8.

[13] Gad W, Kamel M. Ph-ssbm: Phrase Semantic Similarity Based Model for Document Clustering[C]//Proceedings of KAM'09. Wuhan, China: [s. n.], 2009: 197-200.

编辑 金胡考

---

26. TP-mine:基于分割簇的RFID轨迹数据增量聚类算法

27. 基于人工免疫增量的聚类算法

28. 一种有效的增量聚类算法

29. 一种基于网格的增量聚类算法

30. 一种基于TextRank的文本二次聚类算法

31. 基于蚁群算法的文本聚类算法的参数优化

32. 基于相对密度的多耦合文本聚类算法

33. 一种基于Mahalanobis距离的增量聚类算法

34. 基于MapReduce的改进k-means文本聚类算法

35. 一种中文文本聚类算法的研究

36. 基于Single-Pass算法的网络舆情文本增量聚类算法研究

37. 基于簇特征的增量聚类算法设计与实现

38. 一种基于距离的增量聚类算法

39. 文本聚类算法研究

40. 基于K-means文本聚类算法研究

41. 基于k-均值的文本聚类算法及改进

42. 一种新的基于SVM的文本分类增量学习算法

43. 一种基于族相合性的文本增量聚类算法

44. 基于向量空间的文本聚类算法

45. 基于Kolmogorov复杂性的文本聚类算法改进

46. 一种基于网格密度的聚类算法

47. 一种基于PSO&PAM的聚类算法

48. 一种基于聚类密度的文本分类算法研究

49. 一种中文文本聚类算法的研究

50. 一种基于模拟退火的遗传聚类算法