计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
50(7): 1452-1463, 2013

# 面向信息检索的快速聚类算法

刘 铭 刘秉权 刘远超
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)
(语言语音教育部-微软重点实验室(哈尔滨工业大学) 哈尔滨 150001)
(mliu@insun.hit.edu.cn)

# A Fast Clustering Algorithm for Information Retrieval

Liu Ming, Liu Bingquan, and Liu Yuanchao
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)
(*MoE-MS Key Laboratory of Natural Language Processing and Speech (Harbin Institute of Technology), Ministry of Education, Harbin* 150001)

**Abstract** Due to the fast advance of information retrieval technique, information overload has become a headache problem to Internet users. In order to alleviate user's inconvenience to distinguish useful information from massive junk information, the research for improving retrieval system has gradually become hotter and hotter. Up to now, many techniques have been proposed for automatically categorizing and organizing Web information for users. Among them, clustering is one of the most extensively employed tools. Through clustering retrieval information, Internet users can quickly find out where their interesting retrieval results locate. Unfortunately, traditional clustering algorithms are either ineffective or inefficient for this task. As a result, a novel algorithm specially designed for clustering retrieval information is proposed. This algorithm applies maximum-minimum principle to extract accumulation points to form initial clusters at first. Experiment results show that, this initial cluster partitioning is approximate to the optimal partitioning and only needs small iterative adjustment steps to get convergence. After that, it iteratively adjusts feature set of each cluster to let cluster partitioning more and more precise. Simultaneously, it hierarchically separates the clusters, which don't meet convergence condition, into some sub-clusters to possess the merit of hierarchically representing information. Experiment results also demonstrate that time complexity of this algorithm is close to the recent techniques for clustering retrieval information. Besides, because of iteratively adjusting feature sets, it enables clustering results to be more precise and reasonable.

**Key words** clustering technique for information retrieval; weight adjustment; maximum- minimum principle; fast clustering; self-organizing-mapping (SOM)

**摘 要** 随着信息检索技术的迅猛发展,针对检索系统的改进已逐渐成为研究的热点.聚类是一种有效的改进策略,通过对检索结果进行聚类,可以使用户快速地定位到自己感兴趣的检索信息所在的类别.然而,传统的检索聚类算法要么运行效率低下,要么类别划分能力不强,使它们无法真正地用于检索系统中.针对此问题,提出了一种新颖的检索聚类算法,该算法首先通过极大极小值理论从检索返回的文档集中抽取多个聚点,并依此形成初始文档类划分结果.在此基础上,算法对初始文档类的特征集合进

**收稿日期**:2011-10-14;**修回日期**:2012-06-08
**基金项目**:国家自然科学基金面上项目(61073127);中央高校基本科研业务费专项基金项目(HIT.NSRIF.2013066);中国博士后科学基金面上项目(2013M530156);教育部-微软语言语音重点实验室开放基金项目

行细化调整以使类别的划分更加精确;同时对不满足收敛条件的文档类进行层次分裂以解决信息的分层描述问题.实验表明:此算法的时间复杂度与现有的检索聚类技术相差不多,并且由于对特征集合进行迭代调整使得类别的划分更加准确合理.

**关键词**　信息检索聚类技术;权值调整;极大极小理论;快速聚类;自组织映射

**中图法分类号**　TP391

为使用户能够从大量的检索信息中快速地找到自己感兴趣的内容,可以将聚类技术应用于搜索引擎中,通过对检索返回的信息进行聚类,可以使用户快速地定位到自己感兴趣的信息所在的类别,方便用户的查询.此类聚类技术不同于经典的聚类算法,其要求聚类时间较快,以降低用户的等待时间;类别的划分更加清晰,以防止多个类别的信息互相交错而降低用户的检索效率.目前针对信息检索的聚类技术大致可以分为以下2类:一类是对现有的聚类算法进行改进,使之适用于检索信息的聚类,此类方法能够充分利用现有的聚类技术,通过高质量的聚类算法获得非常好的聚类结果,但其缺点是算法的时间复杂度稍高,并且由于此类聚类算法依赖于文档间的客观相似度以形成聚类结果,很容易形成无法对聚类结果中各文档类的主题进行描述的问题[1-2],典型的代表算法有WEBSOM[1],Scatter/Gather[2]等.另一类针对检索信息的聚类技术是通过主题分析的方法,首先获得检索返回的文档集合中包含的多类信息,并抽取相应的特征来代表这些信息,然后合并与每个特征相关的文档以形成一个类别,此类方法能够很好地描述聚类后每个类别所反映的主题,但是由于其没有进行类别内部及类别间的相似性分析,因此聚类后每个类别的内部凝聚性不强,类间区分度不大,多个类别间的信息相互交错[3-5],典型的代表算法有频繁项集算法FTC[3]、后缀树算法STC[4].

本文将上述2类方法结合起来,首先根据文档的分布情况将文档集合划分为多个类别,并且构造出能够代表每个类别的特征集合.然而,按照上述方法获得的类别划分是比较粗糙的,因此本文通过迭代调整的方式,通过调整特征集合中的特征以及特征的权值,逐步将文档集合划分为多个类内凝聚度高、类间区分性大的文档类.为解决同一篇文档可能归属于多个类别的模糊划分问题以及信息的分层描述问题,本文将模糊思想引入到聚类中,根据文档包含的多个侧重点,将文档归属到多个类别中去,并对聚类结果进行分层表示,使算法能够在一定程度上发现信息间的层次关系.

## 1　文本特征向量预处理

如文献[6]所述,只需从文档集合中抽取少量的有效特征即可获得良好的聚类结果.因此为了降低特征空间的维度以加快聚类速度,本文通过线性回归融合特征第1次出现的位置、特征的频率、特征的文档频率等统计量来计算特征的权值,并从每篇文档中选取权值超过一定阈值的特征去构造特征空间,具体计算方法可参见文献[7].之后即可初始化特征矩阵$\boldsymbol{A}$,$\boldsymbol{B}$,并为每篇文档建立一个指示向量,假设第$i$篇文档的指示向量为$\boldsymbol{VecDirect}_i$.

矩阵$\boldsymbol{A}$的行为文档集合列为特征空间,$A_{ij}$为特征空间中的第$j$个特征在文档集合中的第$i$篇文档中的权值.

计算矩阵$\boldsymbol{A}$中的各文档对应的行向量与第$i$篇文档对应的行向量之间的相似度,将文档集合中的各文档按照其与第$i$篇文档的相似度由大到小的顺序存放于$\boldsymbol{VecDirect}_i$中.矩阵$\boldsymbol{B}$的行列均为文档集合,$B_{ij}$代表第$j$篇文档在第$i$篇文档对应的指示向量$\boldsymbol{VecDirect}_i$中的位置.文档间相似度的具体计算方法为

$$Cos(\boldsymbol{A}_i,\boldsymbol{A}_j)=\frac{\sum_{m=1}^{r}A_{im}A_{jm}}{\sqrt{\sum_{m=1}^{r}A_{im}^2}\sqrt{\sum_{m=1}^{r}A_{jm}^2}},\qquad(1)$$

其中,$\boldsymbol{A}_i$和$\boldsymbol{A}_j$分别为第$i$篇文档和第$j$篇文档对应的特征向量,$r$为特征向量的长度.

上述初始化的矩阵及向量即为算法IRSOM的输入,其具体的使用方法可见2.1节和2.2节.

## 2　基于自组织映射的信息检索聚类算法

本文提出一种基于自组织映射的信息检索聚类算法(information retrieval clustering algorithm based on self-organizing-mapping, IRSOM),该算法首先采用极大极小值理论[8]将文档集合划分为多个类

别,同时构造出能够代表每个类别的特征集合,然后迭代地调整这些作为类别代表的特征集合,并根据调整后的特征集合将文档集合划分为多个类内凝聚度高、类间区分性大的文档类.

### 2.1 聚点选择

如文献[8]所述,在文档集合中具有极大极小值的两篇文档属于不同文档类的可能性是非常大的,同时当选择的具有极大极小值的文档数小于真实类别数时,其极大极小值的变化幅度较小.因此本文即通过文档间相似度的极大极小值去寻找属于不同类别的文档以作为初始类别划分的聚点.

基于极大极小值原则的聚点选择方法的基本原理是:首先使相距最远(余弦相似度最小)的 2 篇文档 $Doc_1$ 和 $Doc_2$ 作为初始的两个聚点 $AC_1$ 和 $AC_2$,而其余聚点的选取标准则可以用递推公式来表达.若已经选取了 $m$ 个聚点,则第 $m+1$ 个聚点的选取原则为

$$AC_{m+1}=\arg\min_{Doc_k\notin Select}\{\max[Cos(Doc_k,AC_r)]\}. \quad (2)$$

其中,$r=1,2,\cdots,m$,$AC_r\in Select$, $Select$ 代表已经选取的聚点集合.

基于极大极小值原则的聚点选取方法的详细步骤如下:

1) 设文档集合为 $DOC$、最多可选取的聚点数为 $p$、已经选取的聚点集合为 $Select$、已经选取的聚点数为 $s$、$VecMin$ 中存放了 $Select$ 中的每个聚点对应的极大极小值、初始时 $Select=$Null,$s=0$.

2) 从文档集合中选取相似度最小的 2 篇文档 $Doc_1$ 和 $Doc_2$ 作为初始聚点 $AC_1$ 和 $AC_2$,记这 2 篇文档间的相似度为 $Sim(Doc_1,Doc_2)$,以 0 和 $Sim(Doc_1,Doc_2)$分别作为聚点 $AC_1$ 和 $AC_2$ 的极大极小值,并将其放入到 $VecMin$ 中.通过矩阵 $\boldsymbol{B}$ 将 $Doc_1$ 和 $Doc_2$ 从所有文档对应的指示向量中予以删除,$s=s+2$.

3) 如果 $s+1\leqslant p$,转步骤 4),否则转步骤 8).

4) 设 $NS$ 为即将选取的新聚点,设 $SH$ 为其对应的极大极小值,初始设 $SH=\infty$.

5) 从头至尾扫描聚点集合 $Select$,设此时正在扫描的聚点为 $AC_k$、$AC_k$ 对应的指示向量 $\boldsymbol{VecDirect}_k$ 中的第 1 篇文档为 $Doc_v$,记 $AC_k$ 与 $Doc_v$ 的相似度为 $Sim(AC_k,Doc_v)$,如果 $Sim(AC_k,Doc_v)\leqslant SH$,则 $NS=Doc_v$,$SH=Sim(AC_k,Doc_v)$.

6) 如果 $AC_k$ 不为 $Select$ 中的最后一个聚点,则转步骤 5),否则转步骤 7).

7) 此时的 $NS$ 即为按式(2)选择的新聚点,将 $NS$ 放入 $Select$ 中,将 $SH$ 放入 $VecMin$ 中,通过矩阵 $\boldsymbol{B}$ 将 $NS$ 从所有文档对应的指示向量中予以删除,$s=s+1$,转步骤 3).

8) 计算 $Select$ 中的每个聚点的深度,并找到最大深度,保留位于最大深度前的聚点.

聚点深度的计算方法为

$$Depth(i)=|VecMin(i)-VecMin(i-1)|+|VecMin(i+1)-VecMin(i)|. \quad (3)$$

上述的聚点选取算法中最多可以产生 $p$ 个聚点.显然,此参数也限制了划分文档集合可以得到的类别数.文献[8]表明:聚类中的类别数一般小于文档集合大小的平方根,因此本文以文档集合大小的平方根为 $p$ 赋值.

本节通过聚点选取算法得到的类别数可能与真实的类别数存在一定的差距,此问题可以由 2.5 节提出的内聚度及区分度分析方法得以解决,通过判断类别的内聚度及类别间的区分度来增加或合并类别,以使类别的划分逐步精确.

上述聚点选取算法的时间复杂度为 $O(p\times p)$,其中算法循环 $p$ 次,每次循环时在步骤 4)~7)中最多需要计算 $p$ 个文档的最小值,由于 $p=\sqrt{n}$,$n$ 为文档数.所以通过极大极小值选取聚点的时间复杂度为 $O(n)$.

### 2.2 初始化类别特征集合

在按照 2.1 节聚点选取算法得到作为类别代表的聚点后,即可将文档映射到每个聚点所代表的类别中去,以形成初始文档类划分,同时也可构造出作为每个类别代表的特征集合.其具体步骤如下:

1) 设聚点集合为 $Select$,设当前正在处理 $Select$ 中的第 $i$ 个聚点 $AC_i$.

2) 找到与 $AC_i$ 最相似的 $d$ 个文档(为保证算法速度,本文设 $d=5$,其中 $d$ 的选取来源于实验部分的统计结果).取这 $d$ 个文档和 $AC_i$ 构成一个文档类,设其为 $C_i$,并从 $Select$ 中删除 $AC_i$.

3) 计算 $C_i$ 中的各文档所包含的每个特征的权值,由于特征的权值能够代表特征的重要性,因此如果某个特征的权值大于所有特征权值的平均值,则视其代表的信息较为重要,将其插入到 $C_i$ 的特征集合中.

4) 检测 $Select$ 是否为空,为空则停止,否则按步骤 2)~4)处理 $Select$ 中的第 $i+1$ 个聚点.

设 $n$ 为文档数、$s$ 为 $Select$ 中包含的聚点数,由 2.1 节可知 $s<p$,且 $p=\sqrt{n}$,即 $s<\sqrt{n}$,则步骤 1)、步骤 4)的时间复杂度为 $O(1)$,步骤 2)为 $O((d+1)\times n)$,步骤 3)为 $O((d+1)\times B)$,$B$ 为特征排序时间,与特征空间有关,步骤 1)~4)共循环 $s$ 次,则特征

集合初始化的时间复杂度为 $O(2s+s\times(d+1)\times n+s\times(d+1)\times B)$,其中 $d+1$ 为常数可忽略,则其时间复杂度为 $O(s\times n+s\times B)$.

特征 $f$ 在类别 $C_i$ 中的权值计算方法为

$$Weight(f,C_i)=\frac{CF(f,C_i)}{DF(f)}\times AW(f,C_i),\quad(4)$$

其中,$CF(f,C_i)$代表特征 $f$ 被类别 $C_i$ 中的多少篇文档所包含;$DF(f)$为特征 $f$ 的文档频率;2 个值的商反映了特征 $f$ 在类别 $C_i$ 和整个文档集合中的分布情况的差异度;$AW(f,C_i)$为特征 $f$ 在类别 $C_i$ 中的平均权值,反映了特征在该文档类中的重要性. 可见,式(4)结合了特征的权值以及特征的分布来确定特征的重要性.

按上述步骤获得作为每个类别代表的特征集合后,即可计算文档与类别特征集合之间的相似度,然后将文档映射到与其具有最大相似度的那个特征集合所代表的文档类中去. 文档与类别特征集合之间的相似度计算方法为

$$Sim(Doc_k,FC_i)=\sum_{inf\in IFki}\frac{DW(Doc_k,inf)\times CW(FC_i,inf)}{DW(Doc_k,inf)+CW(FC_i,inf)}\times\log(|\ IFki\ |),\quad(5)$$

其中,文档 $Doc_k$ 的特征集合和类别特征集合 $FC_i$ 的交集为 $IF_{ki}$;大小为$|IF_{ki}|$;该交集中的某个特征 $inf$ 在 $Doc_k$ 中的权值为 $DW(Doc_k,inf)$;在 $FC_i$ 中的权值为 $CW(FC_i,inf)$.

式(5)从两方面衡量了文档和类别特征集合之间的相似度:1)如果特征 $inf$ 在文档 $Doc_k$ 和类别特征集合 $FC_i$ 中的权值均较大,说明特征 $inf$ 反映的信息在文档和类别特征集合中均位于重要地位,表明文档 $Doc_k$ 所描述的信息和类别特征集合 $FC_i$ 所反映的信息大体相同,则文档和类别特征集合之间的相似度应较大;2)如果文档 $Doc_k$ 的特征集合和类别特征集合 $FC_i$ 之间的交集很大,说明文档 $Doc_k$ 所描述的信息和类别特征集合 $FC_i$ 所反映的信息大致相似,则文档和类别特征集合之间的相似度应较大.

由实验部分可知,按上述方法获得的文档类与真实的类别划分结果的差距很大,类别的内部凝聚度和类间区分度均非常低. 针对此问题,本文将自组织映射的思想引入到算法中,通过迭代的调整特征集合以增大特征集合所代表的文档类的内部凝聚性,并减少类别间信息的交叉性. 其具体实现方法如 2.3 节和 2.4 节所述.

### 2.3　神经元拓扑结构的初始化

自组织映射算法是 1991 年由 Kohonen 首先提出[9],并随后成为广泛使用的一种无导师的自组织和自学习网络[10]. 该算法将高维空间中的文档数据投影到二维平面上,并且输入文档,彼此之间的相似性在二维离散空间中得到了很好地保持. SOM 算法能够根据文档的分布,逐步收敛到最佳的类别划分. 与其他聚类方法相比,SOM 算法的优点在于可以实现实时学习,算法具有自稳定性,无需外界给出评价函数;然而其缺点也非常明显,就是迭代时间较长. 本文正是利用其迭代调整的优点以获得良好的聚类结果,并通过压缩类别特征集合以减少算法的迭代时间.

本文以 2.2 节构造的每个文档类对应于一个神经元,将文档类的特征集合视为神经元的特征集合,之后如图 1 所示的扇形结构组织神经元. 扇形结构是一个闭环结构,其插入、删除神经元的数目与结构无关,这样能够避免传统的矩形结构中由于插入过多的神经元而造成神经元训练不足,出现神经元的"欠利用"问题[11]. 其插入方式如图 2 所示:

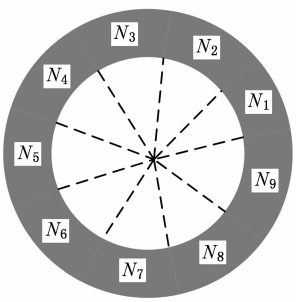



Fig. 1　Round neuron topology.

图 1　扇形结构组织神经元

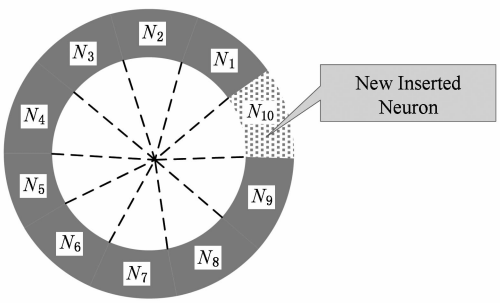



Fig. 2　Neuron insertion in round topology.

图 2　扇形结构插入神经元

神经元扇形结构的初始化步骤如下:

1) 设文档集合为 *DOC*,神经元集合为 *NEURON*.

2) 按式(5)计算 *DOC* 中的每篇文档与 *NEURON*

中的每个神经元之间的相似度,并为每个神经元建立一个 ***MAP*** 向量,其中第 $i$ 维的值为 $DOC$ 中的第 $i$ 篇文档与此神经元之间的相似度.

3) 按式(1)计算任意 2 个 ***MAP*** 向量间的相似度,以反映神经元之间的相似度,同时通过索引 $Index_l$ 记录与神经元 $N_l$ 最相似的神经元在 $NEURON$ 中的位置.

4) 从 $NEURON$ 中随机选择一个神经元放到扇形结构的任意位置上,设此神经元为 $N_i$,并将 $N_i$ 从 $NEURON$ 中删除.

5) 按 $Index_i$ 找到与 $N_i$ 具有最大相似度的神经元,设此神经元为 $N_j$,将 $N_j$ 放到与 $N_i$ 相邻的位置上,同时将 $N_j$ 从 $NEURON$ 中删除.

6) 检测 $NEURON$ 是否为空,如为空则结束,否则按步骤 6)处理 $N_j$.

按如上方法构造的神经元拓扑结构保证了相似的神经元位于相邻的位置上,即保证了自组织映射算法中的拓扑有序要求[12-13].

设初始化时共有 $s$ 个神经元,由 2.1 节可知:$s<p$,且 $p=\sqrt{n}$,则 $s<\sqrt{n}$,$n$ 为文档数,则上述扇形结构初始化算法中,步骤 1)的时间复杂度为 $O(1)$,步骤 2)构造 ***MAP*** 向量的时间复杂度为 $O(s\times n)$,步骤 3)计算 ***MAP*** 向量间的相似度的时间复杂度为 $O(s\times s)$,步骤 4)~6)构造扇形结构的时间复杂度为 $O(s)$,所以上述初始化算法的时间复杂度为 $O(s\times n+s\times s+s)=O(s\times n)$.

### 2.4 迭代调整

传统的自组织映射算法实现的是“硬聚类”即一篇文档只属于一个类别,因此只是根据文档调整与其最相似的神经元及其邻域内的神经元的权值.但是现实生活中一篇文档可能描述了多种信息,可以归属于多个类别[14].本文即按照上述思路对自组织映射算法进行改进,将模糊思想引入到聚类中[15],使一篇文档可以根据其描述的信息隶属于多个类别.具体方法为构造一个隶属度矩阵 $\boldsymbol{U}$ 以表明不同文档属于不同神经元所代表的文档类的可能性.例如,$U_{ij}$ 即代表第 $i$ 篇文档属于第 $j$ 个神经元所代表的文档类的隶属度,以第 $i$ 篇文档与第 $j$ 个神经元之间的相似度为 $U_{ij}$ 赋值,如果此相似度小于平均值,则将 $U_{ij}$ 置为 0.将 $\boldsymbol{U}$ 的行向量归一化以保证同一篇文档对于不同神经元的隶属度的平方和为 1.

在获得文档对神经元的隶属度矩阵后,即可随机选择文档对神经元进行迭代调整,通过调整神经元中的特征及特征的权值使得类别的划分逐步精确.其迭代调整的具体步骤如下:

1) 从文档集合 $DOC$ 中随机选取一篇文档,设此文档为 $Doc_k$.

2) 计算 $Doc_k$ 与神经元集合 $NEURON$ 中的每个神经元之间的相似度,并使用数组 $IndexArray_k$ 存放与 $Doc_k$ 的相似度超过平均值的多个神经元.

3) 调整 $IndexArray_k$ 中的每个神经元的权值,同时调整位于该神经元的邻域范围内的神经元的权值.

4) 将 $DOC$ 中的文档映射到与其具有最大相似度的神经元所代表的类别中去.

5) 检测算法是否达到最大循环次数,如达到,则停止,否则运行步骤 6).

6) 检测算法是否满足收敛条件,如满足,则停止,否则运行步骤 1)~6)直至收敛.

式(6)介绍了如何根据文档 $Doc_k$ 中的某个特征 $f$,调整神经元 $N_b$ 中的相应特征的权值.如果特征 $f$ 被 $N_b$ 的特征集合 $FS(N_b)$ 所包含,则增大该特征的权值,说明此特征在文档类中的重要性有所增加,否则将 $f$ 插入到 $FS(N_b)$ 中,并赋初始值 0.1.具体的调整方法为

$$NW(N_b,f)(t+1)=\begin{cases}NW(N_b,f)(t)+a(t)\times U_{kb}\times\left(\dfrac{DW(Doc_k,f)}{NH+0.5}\times dist+1\right), & \text{If } f\in FS(N_b);\\ 0.1, & \text{If } f\notin FS(N_b);\end{cases}\tag{6}$$

其中,$N_i$ 代表与文档 $Doc_k$ 的相似度超过平均值的某个神经元,$N_b$ 代表位于 $N_i$ 的邻域范围内的某个神经元,$NW(N_b,f)$ 为特征 $f$ 在神经元 $N_b$ 中的权值,$DW(Doc_k,f)$ 为特征 $f$ 在文档 $Doc_k$ 中的权值,$NH$ 为邻域范围,$a(t)$ 为学习速率,均随着算法迭代调整次数的增加而单调下降.$Dist$ 记录了神经元 $N_b$ 和 $N_i$ 之间的距离,上述参数的具体含义可参见文献[12].

设 $s$ 为神经元数即类别数,$n$ 为文档数,则步骤 1),5),6)的时间复杂度为 $O(1)$,步骤 2)的时间复杂度为 $O(s)$,步骤 3)的时间复杂度和与文档的相似度超过平均值的神经元数有关,由于神经元数为 $s$,因此步骤 3)的时间复杂度最多为 $O(s)$,步骤 4)的时间复杂度为 $O(sn)$.设算法循环 $k'$ 次,则上述迭代调整的时间复杂度为 $O(k'sn)$.可见其时间复杂度与循环次数 $k'$、神经元数 $s$ 和文档数 $n$ 有关,其中 $s$ 和 $n$ 在算法运行前已经确定,也就是说 $k'$ 的大小

直接决定着算法的时间性能.而由文末的实验部分可知,当算法的迭代调整次数达到一定程度时,类别划分结果已经很好.因此本文如文献[12]所述,以文档集合大小的平方根作为最大循环次数,当算法的迭代调整次数达到最大循环次数时,即停止算法的运行.

本文以 $MQE$ 作为算法收敛的判别条件,具体的计算方法为

$$MQE=\frac{\sum_{i=1}^{C}\sum_{\boldsymbol{D}_j\in C_i}\frac{U_{ji}\times|\boldsymbol{D}_j-\boldsymbol{V}_i|^2}{|C_i|}}{C},\qquad(7)$$

其中,$C$ 代表聚类后生成的类别数;$C_i$ 代表映射到第 $i$ 个神经元的文档类;$|C_i|$ 为其包含的文档数;$\boldsymbol{V}_i$ 代表第 $i$ 个神经元的特征向量;$\boldsymbol{D}_j$ 代表第 $j$ 篇文档的特征向量;$U_{ji}$ 代表第 $j$ 篇文档属于第 $i$ 个神经元所代表的文档类的隶属度.

当 $|MQE(t+1)-MQE(t)|<e$ 时,即停止算法的运行.其中 $t$ 代表迭代次数,$e=0.01$.如文献[12]所述:$MQE$ 能够反映类别的整体凝聚程度,因此当相邻两次迭代调整的 $MQE$ 的差值小于一定值时,可说明算法已经获得比较好的类别划分结果,此时即可停止迭代调整.

由文中 2.1~2.4 节的时间复杂度分析可知:算法 IRSOM 的时间复杂度为 $O((n)+(s\times n+s\times B)+(s\times n)+(k'sn))$,其中 $s$ 为神经元数即类别数,$n$ 为文档数,$B$ 仅与特征空间有关.如果忽略特征空间的影响,算法的时间复杂度为 $O(k'sn)$,与传统的自组织映射算法相同.同时注意到传统的自组织映射算法中,神经元的特征集合为特征空间中所有特征[16-17],而本文采用权值大于平均值的特征去构造神经元的特征集合,从而限制了特征集合的大小,因此在计算文档和神经元间的相似度时比传统的自组织映射算法快很多.

### 2.5　内聚度及区分度分析

传统的自组织映射算法的运行速度过慢,主要原因在于每次迭代调整时,仅通过训练文档对局部的神经元进行调整,并且由于学习速率随迭代调整次数的增多而单调下降,因此迭代次数越多权值调整幅度越小,使得算法的后期需要进行大量的小范围调整.针对此问题,本文引入了文档类的内聚度分析和文档类间的区分度分析,即通过判断特征集合所代表的文档类是否被很好地区分,是否有某些文档类被错误地分割为多个类别来对神经元进行全局调整,以减少算法迭代调整的次数.

#### 2.5.1　类别内聚度分析

依赖于客观相似度的聚类算法得到的文档类中很可能存在着某些噪声文档,这些文档严重地影响了聚类的效果,因此应该快速发现这些噪声文档,并且从类别特征集合中把引起这些噪声的特征剔除掉以提高类别的内聚度.本文采用局部密度[18]来判断文档类内的噪声文档,并通过调整特征集合的结构以改变文档类的内聚度.

基于局部密度的聚类是 DBSCAN[19]的改进算法,其只需扫描一遍文档集合即可将文档集合划分为多个类别,同时这种基于局部密度的聚类方式改变了以往密度聚类中对于邻域半径 $E$ 和 $E$ 邻域内的最少对象数 $MinPts$ 的依赖性,其缺点是算法的精度不高.

本文以迭代调整后生成的每个文档类作为一个待聚类的文档集,以文档代表密度聚类中的点,以文档间特征向量的余弦相似度代表两点间的距离.取每个文档类经过局部密度聚类后生成的具有最多文档数的子类作为该文档类的代表,将其他文档视为噪声文档予以过滤,然后清空这个文档类的特征集合,之后如 2.2 节所述,按式(4)计算具有最多文档数的子类中每个特征的权值,排序后输出权值大于平均值的多个特征作为此文档类的特征集合.

可见,按如上方法构造的特征集合仅保留了与文档类(经过密度聚类后具有最多文档数的子类)描述的信息相关的特征.在采用局部密度对文档类进行类别划分时并不需要精确的划分结果,因为可以如 2.4 节所述,通过迭代调整特征集合使得类别的划分越来越精确.

#### 2.5.2　类别间区分度分析

本文以神经元的 ***MAP*** 向量间的相似度来代表两个文档类属于同一类别的可能性.如 2.3 节所述,***MAP*** 向量记录了文档与神经元间的相似度,如果两个神经元的 ***MAP*** 向量非常相似,则说明映射到这两个神经元的文档类所反映的信息是相似的.

由于每次迭代调整时仅通过训练文档对局部的神经元进行更新,因此没有必要在每次迭代调整后都进行内聚度和区分度分析,这样本文在进行 100 次迭代调整后才进行一次类别的内聚度和区分度分析.

按照此方法,可在每经过 100 次迭代调整后即将所有神经元的 ***MAP*** 向量中所有维度的权值置为 0.设在接下来的 100 次迭代调整中某次随机选取的

文档为 $Doc_k$,则将 $Doc_k$ 与每个神经元间的相似度记录在此神经元的 ***MAP*** 向量中文档 $Doc_k$ 对应的维度上,之后计算任意两个神经元的 ***MAP*** 向量间的余弦相似度.选择相似度最大的两个神经元予以合并,并按照 2.5.1 节介绍的局部密度方法确定合并后生成的神经元所代表的文档类的特征集合.如果合并的是扇形结构中第 $i$ 个位置和第 $j$ 个位置上的神经元,则将合并后生成的神经元插入到这 2 个神经元的中间位置即 $(i+j)/2$,并将这 2 个进行合并的神经元予以删除.

## 3 层次表示方式

现实生活中的许多信息具有抽象的层次结构,为了表示信息的层次结构,本文将聚类的结果分层表示.

现有的层次聚类算法可以分为 2 个方向:分裂或凝聚.

1) 分裂.将文档集合视为一个大类,对文档集合进行分裂,如果有某些类别不满足收敛条件则继续分裂,直到所有类别均满足收敛条件,例如 GHSOM[20].

2) 凝聚.将每篇文档视为一个类别,对相似的类别进行合并,直到生成的类别划分结果满足收敛条件,例如层次聚合聚类[21].

本文采用分裂的方法对经过迭代调整后内聚性不够好的文档类进行处理.如文献[12]所述:MQE 反映的是聚类的整体凝聚程度.满足整体收敛条件的聚类结果,其单个类别的凝聚性并不一定能够达到收敛要求.因此本文判断单个类别的凝聚度,当其不满足收敛阈值时则将此文档类进行分裂.此时的分裂即是将不满足条件的神经元代表的文档类作为待聚类的文档集,采用 IRSOM 算法对其聚类,如果有多个神经元需要分裂可以并行执行.

单个类别的凝聚性的判别条件如下:

$$Agglomeration(C_i)=\sum_{\boldsymbol{D}_j\in C_i}\frac{U_{ji}\times|\boldsymbol{D}_j-\boldsymbol{V}_i|^2}{|C_i|},\tag{8}$$

其中各符号的具体含义如式(7)所述.

在对需要分裂的文档类进行初始化时,使用代表该文档类的神经元及其在扇形结构中临近的 2 个边界神经元作为初始神经元,并采用插值方式,初始化 2 个新的插值神经元.每个插值神经元的特征集合为需要分裂的文档类的代表神经元与其边界神经元的特征集合的并集,此并集中的特征权值为特征在这 2 个神经元中的权值平均值.采用此初始化方法的原因是由于需要分裂的神经元所代表的文档类大致集中于此神经元与其边界神经元的周围,因此可以采用如上方法将这 5 个神经元首尾相连构成初始扇形结构,然后按照本文 2.3~2.5 节所述的方法进行权值调整训练.

## 4 实验及分析

为验证本文算法的性能,实验中采用了两类语料:一类为标准测试语料,来源于 Trec 评测数据,Trec(text retrieval conference)是文本检索领域人气最旺、最权威的评测会议,该会议细分为几大主要方向:问题回答(QA)、特定领域检索(Legal,Genomics,Enterprise,Blog)、传统 Web 检索等.本文即使用其 2011 年的中文问句检索语料作为算法 IRSOM 的标准测试数据.随机选择其中的 50 条问句,每个问句对应 1 000 条相关文献.

本文同时构造了一类大规模的网络语料以表明算法 IRSOM 对大规模真实数据的处理能力.为使评测数据能够尽量真实,本文使用百度公司公布的 2011 年的查询日志作为数据源,使用检索频率排名前 50 位的问句模拟用户经常输入的查询,之后再从中随机选取 50 条问句以模拟用户的随机输入.使用搜索引擎"Google"检索这些问句,将检索到的前 1 万篇,共 1 百万篇文档作为测试数据集.观察此数据集可知:每个数据集中包含的文档均过于分散,例如以"旅游"作为检索词得到的数据集中包含有"路线"、"价格"、"地图"、"景点"等多方面的文档.因此需要对检索信息进行聚类,以加快用户寻找相关信息的速度.

聚类是依靠文档间的内在相似性将文档集合划分为多个类内凝聚性高、类间区分性大的类别,因此本文即采用类内准确率及类间准确率来衡量聚类算法的性能.设算法得到的聚类结果中共有 $z$ 个类别,记这 $z$ 个类别为 $C_1,C_2,\cdots,C_z$,则算法的类内准确率和类间准确率如下:

1) 类内准确率.将每个文档类按其内部含义人工划分为多个子类,设文档类 $C_i$ 经过人工划分后得到的子类别为 $Sc_{i_1},Sc_{i_2},\cdots,Sc_{i_m}$,$m$ 为子类别的个数.以具有最多文档数的子类作为 $C_i$ 所描述的信息的代表,设这个子类别为 $Sc_{i_{\max}}$,记其文档数为 $|Sc_{i_{\max}}|$,将其他的子类别包含的文档视为 $C_i$ 的不相关文档,则 $C_i$ 的类内准确率为 $|Sc_{i_{\max}}|/|C_i|$.

2) 类间准确率. 检测这 $z$ 个文档类中的每个文档类是否含有与其他 $z-1$ 个文档类所描述的信息具有相关性的文档,以 $ITDoc_i$ 代表文档类 $C_i$ 内与其他文档类具有相关性的文档数,则 $C_i$ 的类间准确率为 $ITDoc_i/|C_i|$.

以所有类别的类内准确率和类间准确率的平均值作为算法的类内准确率和类间准确率. 由于本文构造的测试数据集中使用多个检索词获得了多类测试数据,因此下文所说的准确率即对应于算法在所有测试数据集上的准确率的平均值.

本文将层次聚类[21]、$K$-means[22],GHSOM[20],GSOM[23],FTC[3],STC[4] 应用于上述测试数据集中,并按照上述方法分别计算每种算法的类内准确率和类间准确率. 同时计算了仅通过极大极小值进行类别划分、分层划分、不分层划分、分层模糊划分 4 种情况下 IRSOM 的类内准确率和类间准确率. 将各种算法的类内准确率列于图 3 中,类间准确率列于图 4 中:

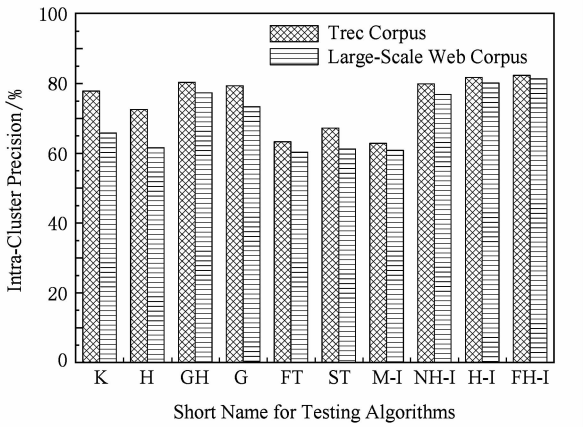



Fig. 3　Intra-cluster precisions of different algorithms.

图 3　不同算法的类内准确率

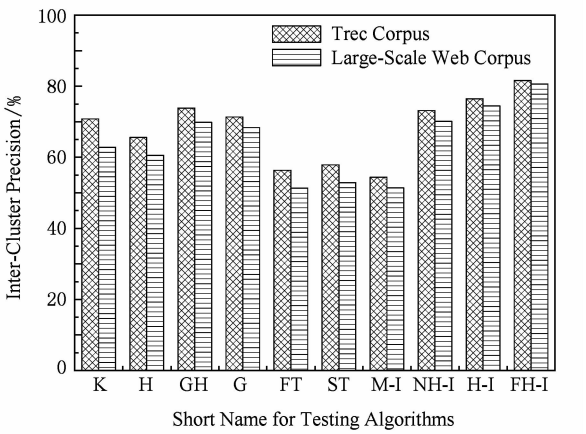



Fig. 4　Inter-cluster precisions of different algorithms.

图 4　不同算法的类间准确率

在图 3 和图 4 中符号 K 代表 $K$-means;H 代表层次聚类;GH 代表 GHSOM;G 代表 GSOM;FT 代表 FTC;ST 代表 STC;M-I 代表仅通过极大极小值得到的类别划分结果;NH-I 代表不使用层次分裂的 IRSOM 算法;H-I 代表使用层次分裂的 IRSOM 算法;FH-I 代表分层模糊划分的 IRSOM 算法. 在本文的后续实验中同样采用如上符号来代表上述算法.

图 3 和图 4 将本文介绍的算法 IRSOM 和已有的聚类算法进行了对比. 其中层次聚类、$K$-means,GSOM 及 GHSOM 是目前比较通用的经典聚类算法,而 FTC 和 STC 是比较流行的检索聚类算法. 通过将图 3 和图 4 中的准确率结果进行对比后可知,算法 FTC 和 STC 的类内准确率和类间准确率均较低,这是由于上述算法是通过信息挖掘的方法来抽取文档集合所包含的多个特征,然后根据这些特征将文档集合划分为多个类别,其得到的聚类结果由于没有进行类内凝聚性和类间区分性的分析,使得类别间信息的交叉性很大从而降低了算法的准确率. 相对地,由于通过调整文档在类别间的分布,经典聚类算法具有较高的类内准确率和类间准确率. 算法 IRSOM 结合了上述两类算法的思想,由图 3 和图 4 可知,在按照极大极小值进行类别划分后,IRSOM 得到的聚类结果的类内准确率和类间准确率均很低(符号 M-I 对应的准确率),此时通过迭代调整,使得算法的准确率有了很大程度的提升(符号 NH-I 对应的准确率),同时 IRSOM 还结合了层次划分以及模糊聚类的思想使其准确率得以进一步提升(符号 H-I,FH-I 对应的准确率).

上述准确率的计算是基于主观分析的,而作为算法的收敛判断条件 $MQE$ 可作为客观的评价标准. 本文将层次聚类、$K$-means,GHSOM,GSOM,FTC,STC 应用于上述测试数据集中,并在图 5 中记录了各种算法的 $MQE$ 值. 同时记录了仅通过极大极小值进行类别划分、分层划分、不分层划分、分层模糊划分 4 种情况下算法 IRSOM 的 $MQE$ 值.

如文献[12]所述:$MQE$ 能够反映聚类的整体凝聚程度,其值越小说明算法生成的类别的凝聚性越好. 由图 5 可知,$MQE$ 值的大小与图 3 和图 4 的准确率是相互对应的,较高的准确率对应了较低的 $MQE$ 值. 可以看出,经典聚类算法具有很低的 $MQE$ 值,即算法具有很好的类别凝聚性. 其中 GHSOM 和 GSOM 的效果最好,这是因为它们是传统的自组

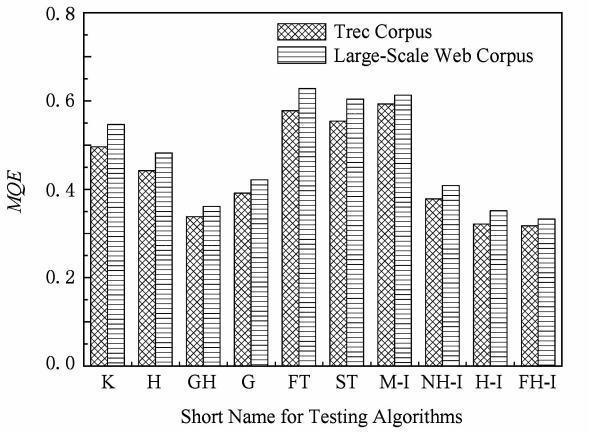



Fig. 5 *MQE* values of different algorithms.

图 5 不同算法的 *MQE* 值

织映射算法的改进算法,保留了自组织映射算法的优点:通过动态调整神经元的权值将文档集合划分为多个内聚度高的文档类. 而本文介绍的算法 IRSOM 也结合了自组织映射算法的思想,同时对自组织映射算法进行了改进,将神经元的特征集合进行压缩,仅选择与神经元代表的文档类具有相关性的特征去构造神经元的特征集合,使得类别的内聚度更高. 并且还通过计算神经元间的相似性将过于细分的文档类进行合并,从而加大了类别间的区分性,因此 IRSOM 的 *MQE* 值最低. 由图 5 还可知,当将层次划分和模糊聚类的思想引入到算法后,IRSOM 可对信息进行分层描述,同时使得文档能够被与其相关的多个文档类所包含,从而进一步提升了其性能.

本文检测了在对网络数据集进行聚类时,IRSOM 的 *MQE* 值随算法逐层推进的变化情况,并在图 6 中记录了这种变化:

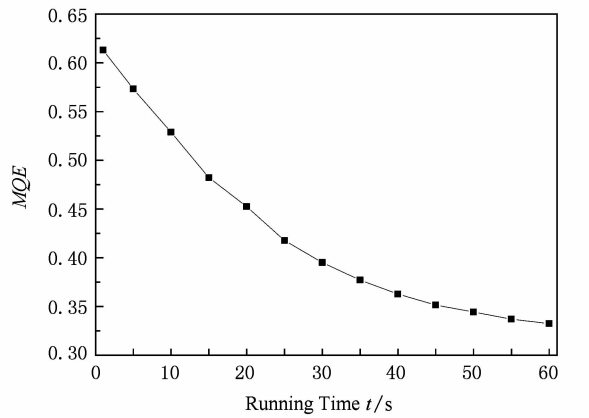



Fig. 6 Curve of *MQE* value of IRSOM along with time passing.

图 6 算法 IRSOM 的 *MQE* 值随时间的变化曲线

由图 6 可见,算法 IRSOM 首先通过极大极小值将文档集合划分为多个类别,但是通过此方法得到的聚类结果中,其类别的内部凝聚性和类间区分性均很低,因此算法运行初期的 *MQE* 值很高. 在此基础上,IRSOM 对神经元的特征集合进行迭代调整,由图 6 可见,只需经过少量的迭代调整后算法的 *MQE* 值即有了很大程度的降低,即算法的准确率有了大幅度的提升. 由图 6 还可知,随着时间的推移,*MQE* 值的改变幅度逐渐降低,算法成收敛状态. 这是由于随着算法的运行,类别的划分逐渐接近于收敛状态,同时由于学习速率的降低使得算法对神经元的调整幅度逐渐减小. 因此当 *MQE* 值的降低幅度小于一定值时即可停止算法,此时的类别划分已经处于比较好的收敛情况,本文即以此方法作为算法的结束判别条件.

如本文引言所述,在对检索信息进行聚类时,聚类时间也是非常重要的指标. 因此本文记录了层次聚类、*K*-means, GHSOM, GSOM, FTC, STC 在不同测试数据集上的聚类时间,同时记录了仅通过极大极小值进行类别划分、分层划分、不分层划分、分层模糊划分 4 种情况下算法 IRSOM 的聚类时间,并将结果列于图 7 中.

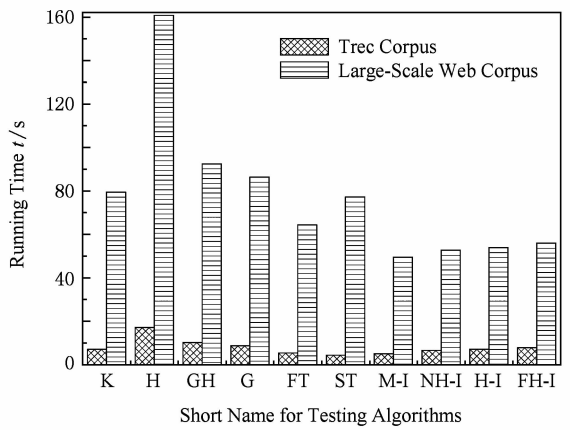



Fig. 7 Running time of different algorithms.

图 7 不同算法的聚类时间

由图 7 可见,在小规模的测试数据集上(例如 Trec 测试数据集),算法 FTC 和 STC 具有较短的聚类时间,这是因为上述算法仅需扫描一遍特征空间即可进行类别划分,因此时间性能很好. 而经典聚类算法中的 *K*-means, GSOM 和 GHSOM 的时间性能均较差,这是因为经典聚类算法需要不断地调整文档在类别间的分布,直到算法满足收敛判别条件时才停止运行,使得算法的运行时间相对较长. 其中 *K*-means 的时间复杂度为 $O(kln)$, $k$ 为循环次数,

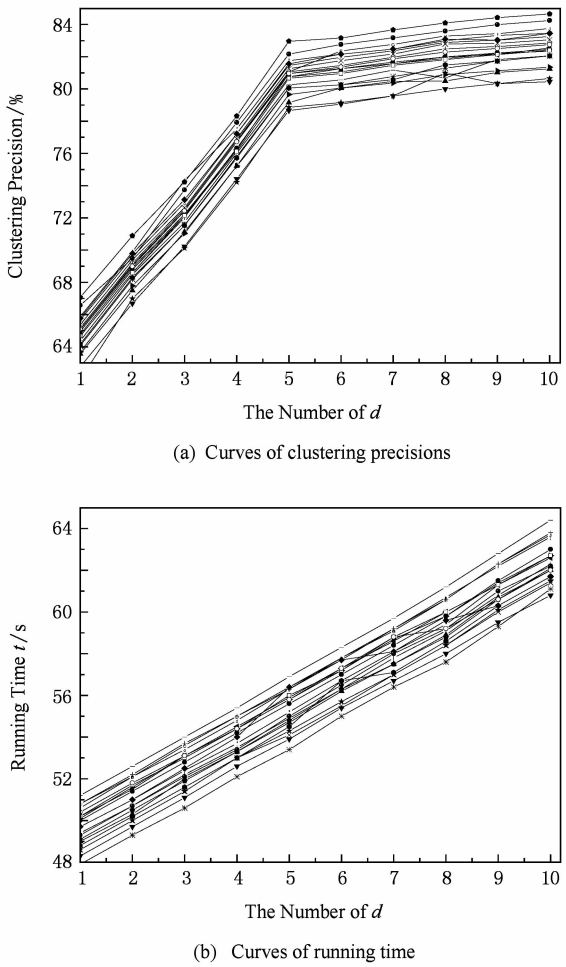
$l$ 为类别数,$n$ 为文档数;GSOM 的时间复杂度为 $O(k'sn)$,$k'$ 为循环次数,$s$ 为神经元数即类别数,$n$ 为文档数,通常来说 GSOM 中 $k'$ 和 $s$ 的值均比 $K$-means 中的 $k$ 和 $l$ 的值大[22],因此 GSOM 的聚类时间要长于 $K$-means. GHSOM 在 GSOM 的基础上进行了层次分裂,因此其聚类时间要长于 GSOM[20]. 层次聚类的时间复杂度最高为 $O(n^2)$. 由本文 2.4 节对 IRSOM 的时间复杂度分析中可以得出:算法 IRSOM 的时间复杂度也为 $O(k'sn)$,然而由于 IRSOM 首先通过极大极小值获得了比较接近于真实聚类结果的初始类别划分,降低了迭代次数,同时通过压缩神经元的特征集合减少了相似度计算时的时间,因此无论是在小规模的 Trec 测试数据集上,还是在大规模的网络测试数据集上,IRSOM 的时间性能均好于经典的聚类算法. 但是,当文档数较少时,IRSOM 的时间性能要弱于 FTC 和 STC,其原因在于当文档数较少时,IRSOM 采取的特征集合压缩方法的效果还没有被有效地体现出来,并且由于其比 FTC 和 STC 多了迭代调整步骤,因此聚类时间较长. 然而,当文档数增多后,其特征集合压缩方法的效果被有效地体现出来,此时的聚类时间明显少于 FTC 和 STC,是所有测试算法中表现最好的方法.

本文在 2.2 节中需要选择聚点周围的 $d$ 个文档以构成初始类别,其中 $d$ 的选取来源于统计实验的结果. 为在一定程度上展示选择的依据,这里随机选取百度查询日志中的 20 条检索问句,及每个问句对应的 1 万条检索结果,以记录不同的 $d$ 对应的聚类准确率和聚类时间变化曲线图. 结果如图 8 所示.

由图 8 可见,算法的聚类时间显然随着 $d$ 的增加而单调上升,而准确率曲线却是随着 $d$ 的增加而逐渐趋于平缓. 由曲线的分布可总结得出,当 $d\geqslant 5$ 后,所有检索问句对应的聚类准确率曲线基本上都比较平稳,因此本文以 $d=5$ 作为选择的依据.

事实上,$d$ 的选择也可通过直观的分析得出,对于一个类别,能够描述其信息的代表特征显然集中地位于与该类别的中心紧密依靠的部分文档中. 当 $d=5$ 时,即选择与类别中心最相似的 5 个文档来构造类别的特征集合后,已经足够将类别中的文档所描述的信息表现出来. 因此本文设置 $d=5$ 以保证在获得良好聚类效果的前提下得到较快的聚类速度.

在这里需要指明的一点是,当处理大规模的数据时,虽然本文算法的性能在所有测试算法中是最好的,但是其聚类时间也较长,例如对于包含 1 万篇




(a) Curves of clustering precisions

(b) Curves of running time

Fig. 8　Clustering precisions and running time of different algorithms with different numbers of $d$.

图 8　不同的 $d$ 对应的算法的聚类准确率和聚类时间

文档的测试数据集进行聚类时,其聚类时间约为 60 s,因此如果将其直接用在搜索引擎上还是无法满足用户的需求.

为保证本文算法可以真正用于检索系统中,必须要进行一些处理. 本文采用了类似于检索中的策略,通过牺牲系统的存储空间来提升算法的效率,随着硬件设备性能的提高及其成本的下降,这些空间的牺牲是完全可以接受的. 事实上,在聚类系统中,相似度计算占用了算法的绝大部分时间,而相似度计算的速度直接影响着聚类的效率. 为提升聚类的效率,可以事先计算文档间的相似度以构造一个文档相似度矩阵,并将其存储于服务器中,当需要对检索返回的文档进行聚类时,可以使用 Hash 表,通过定位文档在相似度矩阵中的位置来快速地确定文档间的相似度. 虽然本文所介绍的算法 IRSOM 通过压缩神经元的特征集合提高了文档与神经元间的相似度计算的效率,但是还是无法解决在聚点选择阶

段、特征集合初始化阶段以及类内凝聚性和类间区分性分析阶段中,文档间的相似度计算过慢以及过于频繁的问题.由图 9 也可看到,这些步骤中的相似度计算几乎占用了大约 85%的聚类时间,因此如果能够提前获得文档间的相似度,显然可以提高聚类的效率.

图 9 即显示了是否事先构造文档相似度矩阵后,各种算法在大规模网络测试数据集上的聚类时间的对比柱状图.

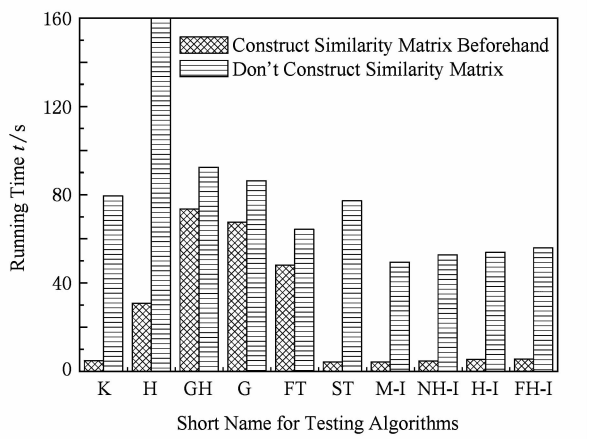



Fig. 9 Histograms of running time of different algorithms whether similarity matrix is constructed beforehand or not.

图 9 构造相似度矩阵前后不同算法的聚类时间的对比柱状图

由图 9 可见,各种算法的聚类时间的下降幅度不同,其中 *K*-means,STC 和层次聚类的下降幅度最大,这是因为这些算法完全依赖于文档间的相似度来完成类别划分,算法 FTC,GSOM 和 GHSOM 的下降幅度很少,这是因为这些算法不依赖于文档间的相似度来完成类别划分.其中 FTC 是通过计算文档与类别中心的特征集合的交集来计算文档和类别间的相似度,而类别中心的特征集合在聚类前是无法预估到的.GSOM 和 GHSOM 是通过计算文档和神经元间的相似度来完成类别划分,而神经元相当于类别中心,因此在聚类前也是无法预估到的.算法 IRSOM 的聚类时间的下降幅度在上述两类算法之间.由图 9 可见,仅通过极大极小值进行类别划分(由符号 M-I 表示)的运行时间的下降幅度非常大,这说明,通过事先构造相似度矩阵后,可以大大减少初始类别构造时所耗费的时间,而该时间几乎占用了算法的绝大部分运行时间.这也说明了当采用本文介绍的压缩神经元的特征集合的方法后,可以大大减少文档和神经元间的相似度计算时所耗费的时间,与文档间的相似度计算时所耗费的时间相比,这部分时间几乎可以忽略不计.由图 9 可见,通过事先构造相似度矩阵后,对 1 万篇文档进行聚类时,IRSOM 所耗费的时间不到 5 s,而最快的 *K*-means 和 STC 的聚类时间也就是 4 s 左右,相差不是很大.同时通过比较图 3 和图 4 中的准确率可知,IRSOM 的聚类准确率要远远好于 *K*-means 和 STC,这说明了算法 IRSOM 具有较好的性能,完全能够应用于真实的检索系统中.

## 5 结 论

本文提出了一种针对信息检索的聚类算法 IRSOM,该算法首先通过极大极小值将文档集合划分为多个类别,同时构造出代表每个类别的特征集合.但是经由上述方法得到的类别划分结果,其类内凝聚性和类间区分性均很差.为解决此问题,本文将自组织映射的思想引入到算法中,通过迭代调整特征集合的结构及特征的权值逐步提高类别的内聚度.同时为了解决属于同一文档类的文档被错误地划分为多个类别而造成类间区分性差的问题,本文通过计算神经元的 ***MAP*** 向量间的相似度,并将相似度较大的文档类进行合并以增大类别间的区分性.为了能够对信息进行分层描述,本文对达不到收敛阈值的文档类进行层次分裂.同时将模糊思想引入到算法中,使得文档能够根据其描述的信息映射到多个相关的类别中去.通过实验可以发现:算法 IRSOM 通过压缩类别的特征集合大大缩短了聚类时间,当文档数较少时,其时间性能和现有的信息检索聚类算法差距不大,当文档数增加后,其时间性能要好于现有的信息检索聚类算法,同时由于其逐步调整特征集合的结构及特征的权值,使得算法的准确性有了大幅度的提高.

## 参 考 文 献

[1] Azcarraga A P, Yap T N Jr, Tan J, et al. Evaluating keyword selection methods for WEBSOM text archives [J]. IEEE Trans on Knowledge and Data Engineering, 2004, 16(3): 380-383

[2] Peter P, Patricia S, Marti H, et al. Scatter/gather browsing communicates the topic structure of a very large text collection [C] //Proc of ACM SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 1996: 213-220

[3] Beil F, Ester M, Xu X. Frequent term-based text clustering [C] //Proc of ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2002: 436-442

[4] Zamir O, Etzioni O. Web document clustering: A feasibility demonstration [C] //Proc of ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM, 1998: 46-54

[5] Valdes P R, Pericliev V, Pereira F. Concise, intelligible, and approximate profiling of multiple classes [J]. International Journal of Human Computer Systems, 2000, 53 (3): 411-436

[6] Li Yuanhong, Dong Ming, Hua Jing. Localized feature selection for clustering [J]. Pattern Recognition Letters, 2008, 29(1): 10-18

[7] Xu Yongdong, Xu Zhiming, Wang Xiaolong, et al. Using multiple features and statistical model to calculate text units similarity [C] //Proc of Int Conf on Machine Learning and Cybernetics. Piscataway, NJ: IEEE, 2005: 3834-3839

[8] Tsutsumi K, Nakajima K. Maximum/minimum detection by a module-based neural network with redundant architecture [C] //Proc of Int Joint Conf on Neural Networks. Piscataway, NJ: IEEE, 1999: 558-561

[9] Kohonen T. Self-Organizing Maps [M]. 2nd ed. Berlin: Springer, 1997

[10] Kohonen T, Kaski S, Lagus K, et al. Self organization of a massive document collection [J]. IEEE Trans on Neural Networks, 2000, 11(3): 574-585

[11] Wu Ying, Yan Pingfan. A study on structural adapting self-organizing neural network [J]. Acta Electronica Sinica, 1999, 27(7): 55-58 (in Chinese)
(吴郢, 阎平凡. 结构自适应自组织神经网络的研究[J]. 电子学报, 1999, 27(7): 55-58)

[12] Liu Yuanchao, Wang Xiaolong, Wu Chong. ConSOM: A conceptional self-organizing map model for text clustering [J]. Neurocomputing, 2008, 71(4/5/6): 857-862

[13] Dalal P R, Joshi I V. Clustering of automobile information using self organizing maps [C] //Proc of Int Conf on Convergent Technologies for Asia-Pacific Region. Piscataway, NJ: IEEE, 2003: 15-17

[14] Shailendra S, Lipika D A. Rough-fuzzy document grading system for customized text information retrieval [J]. Information Systems & Management, 2005, 41(2): 195-216

[15] Celikyilmaz A, Turksen I B. Validation criteria for enhanced fuzzy clustering [J]. Pattern Recognition Letters, 2008, 29 (2): 97-108

[16] Fiannaca A, Di F G, Rizzo R, et al. Fast training of self organizing maps for the visual exploration of molecular compounds [C] //Proc of Int Joint Conf on Neural Networks. Piscataway, NJ: IEEE, 2007: 2776-2781

[17] Hung C, Wermter S. A dynamic adaptive self-organising hybrid model for text clustering [C] //Proc of the 3rd IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2003: 75-82

[18] Ni Weiwei, Sun Zhihui, Lu Jieping. *K*-LDCHD—A local density based *k*-neighborhood clustering algorithm for high dimensional space [J]. Journal of Computer Research and Development, 2005, 42(5): 784-791 (in Chinese)
(倪巍伟, 孙志挥, 陆介平. *K*-LDCHD—高维空间 *K* 邻域局部密度聚类算法[J]. 计算机研究与发展, 2005, 42(5): 784-791)

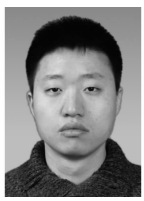
[19] Liu Shang, Dou Zhitong, Li Fei, et al. A new ant colony clustering algorithm based on DBSCAN [C] //Proc of Int Conf on Machine Learning and Cybernetics. Piscataway, NJ: IEEE, 2004: 1491-1496

[20] Andreas R, Dieter M, Michael D. The growing hierarchical self-organizing map: Exploratory Analysis of high-dimensional data [J]. IEEE Trans on Neural Networks, 2002, 13(6): 1331-1341

[21] Reynaldo G G, Aurora P P. Dynamic hierarchical algorithms for document clustering [J]. Pattern Recognition Letters, 2010, 31(6): 469-477

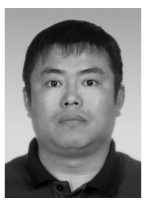
[22] Shahpurkar S S, Sundareshan M K. Comparison of self-organizing map with k-means hierarchical clustering for bioinformatics applications [C] //Proc of Int Joint Conf on Neural Networks. Piscataway, NJ: IEEE, 2004: 1221-1226

[23] Alahakoon D, Halgamuge S K. Dynamic self-organizing maps with controlled growth for knowledge discovery [J]. IEEE Trans on Neural Networks, 2000, 11(3): 601-614



**Liu Ming**, born in 1981. PhD and lecturer. Member of China Computer Federation. His main research interests include text clustering and data mining.

**Liu Bingquan**, born in 1970. PhD and associate professor. Senior member of China Computer Federation. His main research interests include natural language processing clustering and Q&A (liubq@insun.hit.edu.cn).



**Liu Yuanchao**, born in 1971. PhD and associate professor. Member of China Computer Federation. His main research interests include semantic similarity and neuron processing (lyc@insun.hit.edu.cn).