基于近邻传播的文本数据流聚类算法研究

李一鸣[1,2] 倪丽萍[1,2] 方清华[1,2] 刘慧婷[3]
(合肥工业大学管理学院 合肥 230009)[1]
(合肥工业大学过程优化与智能决策教育部重点实验室 合肥 230009)[2]
(安徽大学计算机科学与技术学院 合肥 230601)[3]

摘 要 随着大数据时代的到来,网络上产生了大量非结构化文本数据流,这些文本数据流具有动态、高维、稀疏等特征。针对这些特点,首先将传统的AP算法及流式文本数据特征相结合,然后提出文本数据流聚类算法——OAP-s算法。该算法通过在AP算法上引入衰减因子,对聚类中心结果进行衰减,同时将当前时间窗口的聚类中心带入到下一时间窗口中进行聚类。针对OAP-s算法的不足,又提出了OWAP-s算法。该算法在OAP-s算法模型的基础上定义了加权相似度,并通过引入吸引度因子,使得历史聚类中心更具吸引性,得到更精确的聚类结果。同时,两种算法均采用滑动时间窗口模式,使算法既能体现数据流的时态特征,又能反映数据流的分布特征。实验结果表明,两种算法在聚类精确度、稳定性方面均高于OSKM算法,而且具有较好的伸缩性和可扩展性。

关键词 数据挖掘,近邻传播聚类,文本数据,滑动时间窗口,权重
中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.041

Research of Text Data Streams Clustering Algorithm Based on Affinity Propagation

LI Yi-ming[1,2] NI Li-ping[1,2] FANG Qing-hua[1,2] LIU Hui-ting[3]
(College of Management, Hefei University of Technology, Hefei 230009, China)[1]
(Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education, Hefei University of Technology, Hefei 230009, China)[2]
(School of Computer Science and Technology, Anhui University, Hefei 230601, China)[3]

Abstract With the advent of the era of big data, a large amount of unstructured text data streams have emerged online. Those data streams are dynamic, high-dimensional and sparse. For these characteristics and on the basis of the traditional AP algorithm, a text data stream clustering algorithm, called OAP-s algorithm, was proposed in this paper. By introducing attenuation factor in the AP algorithm, OAP-s algorithm passes the clustering center of the current window to the next window, while attenuating the results. However, this OAP-s algorithm also has some shortcomings. Therefore, we proposed another text data stream clustering algorithm, called OWAP-s algorithm. Based on the OAP-s algorithm, OWAP-s algorithm defines the weighted similarity, introduces attractive factor and makes the historic clustering center more attractive, thus obtains more accurate clustering results. Meanwhile, both algorithms adopt the sliding time window model, which reflects the temporal characteristics as well as the distribution of the data stream. Experimental results show that both algorithms are flexible and extensible, and they are more accurate and stable than OSKM algorithm.

Keywords Data mining, AP clustering, Text data, Sliding window, Weight

1 引言

随着大数据时代的到来,网络上产生了大量的非结构化数据。面对这些实时产生、数据量庞大、结构复杂的非结构化数据,人们迫切需要从中提取有价值的信息和知识。文本数据流聚类技术是分析这些非结构化数据的常用方法,它在新闻过滤、话题检测及跟踪(TDT)、用户特征推荐等方面取得了很好的应用效果,迅速成为当前的研究热点[1-3]。

与普通数据流相比,文本数据流也具有如下两个方面的特点:(1)在处理时间上要求算法具有实时性、高效性,即算法对大量数据集不能进行多遍扫描;(2)算法需要具有自适应性,因为随着数据的流入,数据模式可能会发生改变。除此之

到稿日期:2015-03-15 返修日期:2015-07-22 本文受国家自然科学基金(71301041,61202227,71271071),国家自然科学基金重点项目(71490725)资助。

李一鸣(1990—),男,硕士生,主要研究方向为数据挖掘、机器学习,E-mail:756165381@qq.com;倪丽萍(1981—),女,博士,副教授,主要研究方向为数据挖掘、智能决策技术;方清华(1990—),女,硕士生,主要研究方向为群智能算法、数据挖掘;刘慧婷(1978—),女,博士,副教授,主要研究方向为数据挖掘、机器学习。

外，文本数据流比普通数据流具有更高的维数和稀疏性，这使得对文本数据流的处理具有更大的挑战性[4,5]。

在现有的研究中，大多数文本数据流算法利用 tf-idf 量化单词的效用，并在此基础上将传统的聚类算法修改成在线算法以适应新的数据流场景。文献[6]提出 CluStream 算法，该方法引入衰减函数，给予每篇文档权重，权重随时间进行衰减。其对类球形簇能取得较好的聚类结果，但较难聚成任意形状的簇。文献[7]提出 ConStream 算法，这种方法利用一个称为簇滴的数据结构记录聚类的相关信息，并通过用户设置不同的参数来构建新的聚类。为了解决高维数据聚类问题，文献[8]提出 HPStream 算法，该算法采用高维投影技术选择子空间进行聚类，同时使用衰减函数表示演化信息，但先验参数——平均聚类维数难以确定。文献[9]将语义平滑模型扩展到文本数据流中；但基于语义的方法要花费很多的人力建立语料库，算法的时间、空间复杂度较高。文献[10]提出 OSKM 算法，该方法是 k-means 方法的扩展，其将不断流入的数据流分成小份，每一份都可以在内存中进行高效的处理。随后，在这些数据上进行 k-means 迭代，得到聚类结果。该算法因为要事先确定聚类个数，不能随着类别的变化对聚类个数进行变化。

AP 算法[11]的优点在于不需要像 k-means 算法那样事先确定聚类的个数。它在迭代过程中不断搜索合适的聚类中心，自动从数据点间识别类中心的位置及个数，使所有的数据点到最近的类代表点的相似度之和最大。而且与传统的 k-means 算法对初始类中心的选择具有敏感性相比，AP 算法是一种确定的聚类算法，多次独立运行的聚类结果一般都非常稳定。

本文将 AP 算法结合权重的运算扩展应用到文本数据流聚类中，针对文本数据流的动态特征，本文将文本数据流分割成一个个滑动时间窗口，然后对每个滑动时间窗口中到来的数据进行处理。将先前滑动时间窗口的聚类中心用在下一滑动时间窗口的聚类中，用于影响下一滑动时间窗口的聚类结果。利用滑动时间窗口进行聚类的优势在于每个滑动时间窗口内的数据都可以保存在内存中，这样就可以对该窗口内的数据点进行多次处理，提高聚类准确率。同时，根据可用的缓冲区大小或者实验需求等，还可以变换滑动时间窗口大小。针对此模式，本文通过引入衰减因子提出了 OAP-s 算法，在此基础上引入了吸引度因子，提出了 OWAP-s 算法。实验结果表明，这两种算法的聚类效果均要优于 OSKM 算法。

本文第 2 节介绍了 AP 算法；第 3 节介绍了 OAP-s 算法的具体内容和步骤；第 4 节介绍 OWAP-s 算法的具体内容和步骤；第 5 节是实验结果与分析；最后总结全文。

2 AP 算法

AP(affinity propagation)算法是 Frey 和 Dueck 于 2007 年在《Science》上提出的一种新的聚类算法[11]。它的特点是快速、高效，不必事先指定聚类数目，并且能够很好地解决非欧空间问题以及大规模稀疏矩阵计算问题等。算法将每个样本点都视为网络中的一个节点，吸引力信息沿着节点连线递归传输，直到找到最优的类代表点集合。算法中用到的基本概念定义如下。

2.1 基本概念

定义 1(相似度 s)　数据点 i 和点 j 的相似度记为 $s(i,j)$，是指点 j 作为点 i 的聚类中心的相似度，一般使用欧氏距离来计算，如 $-\|(x_i-x_j)^2+(y_i-y_j)^2\|$。文中，所有点与点的相似度值全部取为负值，因为可以看到，相似度值越大说明点与点的距离越近，便于后面的比较计算。$s(i,j)$ 代表数据点 j 适合作为数据点 i 的聚类中心的程度。

定义 2(参考度 p)　数据点 i 的参考度称为 $p(i)$ 或 $s(i,i)$，是指点 i 作为聚类中心的参考度，一般取相似度值的中值。近邻算法对于每一个数据点 i 来说都需要输入一个实际的数值 $s(i,i)$，$s(i,i)$ 越大，该数据点越有可能被选为聚类中心。这个数值被称为“偏好”。聚类的数量受这个输入的“偏好”影响，在消息传递过程中呈现出来。

定义 3(吸引度矩阵 r)　$r(i,j)$ 指从数据点 i 到候选聚类中心点 j，反映了在考虑到点 i 的其他潜在候选聚类中心后，j 适合作为点 i 的候选聚类中心所积累的证据。

定义 4(归属度矩阵 a)　$a(i,j)$ 指从候选聚类中心点 j 到点 i，反映了考虑到对于其他数据点来说 j 应该作为候选聚类中心以外，点 i 选择点 j 作为其候选聚类中心做积累的证据。

2.2 AP 聚类算法步骤

Step 1　算法初始化，计算初始相似度矩阵 s；对 p 赋初值。

Step 2　计算样本点间的吸引度值。

$$r(i,j)\leftarrow s(i,j)-\max\{s(i,k)+a(i,k)\}(k\in\{1,2,\cdots,n,\text{但 }k\neq j\}) \tag{1}$$

$$r(j,j)\leftarrow p(j)-\max\{s(j,k)+a(j,k)\}(k\in\{1,2,\cdots,n,\text{但 }k\neq j\}) \tag{2}$$

Step 3　计算样本点间的归属度值。

$$a(i,j)\leftarrow\min\{0,r(j,j)+\sum_{j}\{\max(0,r(k,j))\}\}(k\in\{1,2,\cdots,n,\text{但 }k\neq i\text{ 且 }k\neq j\}) \tag{3}$$

$$a(j,j)\leftarrow\sum_{k}\{\max\{0,r(k,j)\}\}(k\in\{1,2,\cdots,n,\text{但 }k\neq j\}) \tag{4}$$

Step 4　吸引度矩阵和归属度矩阵的更新。

$$r_i=(1-\lambda)*r_i+\lambda*r_{i-1} \tag{5}$$

$$a_i=(1-\lambda)*a_i+\lambda*a_{i-1} \tag{6}$$

λ 是收敛系数，主要用于调节算法的收敛速度及迭代过程的稳定性。

Step 5　如果迭代次数超过设定的最大值或者聚类中心在若干次迭代中不发生改变时终止计算，确定类中心及各类的样本点；否则返回 Step 2，继续计算。

AP 算法根据 $\arg\min_{k}(a(i,k)+r(i,k))$ 得到聚类中心[11,12]。

3 OAP-s 算法

3.1 算法设计思想

OAP-s 算法是 AP 聚类算法在文本数据流上的一个实现。其采用半连续处理过程，即把整个文本数据流分成一个个滑动时间窗口，然后对每个滑动时间窗口中到来的数据进行处理。这种方式的优势是它在限定的时间范围内将一部分文本存储在内存中，可以对这部分文本数据进行多次扫描读取，在不影响时间效率的基础上提高处理的准确性。而且，可以根据可用的缓冲区大小或者实验需求等变换滑动时间窗口的大小[10]。

OAP-s 算法的基本思想是将整个文本数据流分成一个个滑动时间窗口($t=0,1,2\cdots$),对每个时间窗口中流入的数据 N 和历史数据 C 进行聚类,并将聚类结果用到下一个时间窗口中的数据聚类上,期间引入衰减因子 γ,对历史数据的重要性进行衰减。其聚类模型如图 1 所示。

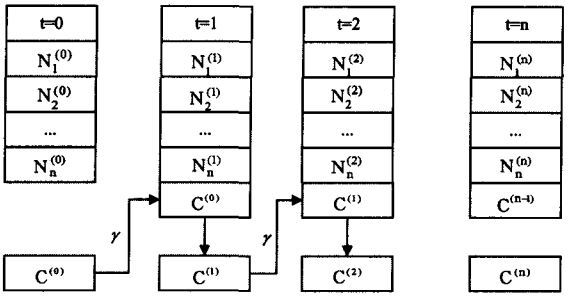


图 1 聚类模型

3.1.1 相关定义

根据图 1 所示的模型,引入如下定义。

定义 5(相似度矩阵) 对于高维文本数据来说,用余弦相似度来表示两个文本的相似性要优于用欧几里德距离的表示[13]。因此,定义相似度矩阵如式(7)。

$$S(i,j)=\begin{cases}-\dfrac{1}{\cos(i,j)}, & i\neq j\\ s^*, & i=j\end{cases} \tag{7}$$

其中,$\cos(i,j)$为数据 i 和数据 j 之间的余弦相似度。当数据 i 和数据 j 不相等时,$\cos(i,j)$越大,i 与 j 的相似度越大;当数据 i 和数据 j 相等时,$P(i)=s^*$,s^* 为所有相似度的中值。

定义 6(归一化的余弦相似度[4])

$$\cos(A,B)=\frac{\sum_{i=1}^{n}a_i\times b_i}{\sqrt{\sum_{i=1}^{n}(a_i)^2}\times\sqrt{\sum_{i=1}^{n}(b_i)^2}}=\frac{A\cdot B}{|A|\times|B|} \tag{8}$$

这里对向量 A 和 B 进行归一化,则公式转化为:

$$\cos\theta=\sum_{i=1}^{n}(a_i\times b_i)=A\cdot B \tag{9}$$

定义 7(历史数据与当前流入数据的权重) 根据图 1 所示,$t+1$ 时刻,参与聚类的数据为新流入的数据 N_i^{t+1} 和历史数据 C_i^t。

假设 WN_i 表示数据点 N_i 的权重;WC_i 表示第 i 个类中数据的个数,对于每次新流入的数据 N_i 来讲,WN_i 权重设为 1。权重越大表明数据的重要性越高,权重越小表明数据的重要性越低。

定义 8(衰减因子) 在此过程中引入衰减因子,衰减因子 γ 的引入可以让历史数据以指数形式被遗忘。设 WN_i^{t+1} 为 $t+1$ 时刻 N_i^{t+1} 聚类到第 i 类中数据的权重,WC_i^t 是 $t+1$ 时刻聚类到第 i 类数据中历史向量 C_i^t 的权重。那么 $t+1$ 时刻聚类向量 C_i^{t+1} 的权重为:

$$\begin{aligned}WC_i^{t+1}&=WN_i^{t+1}+\gamma WC_i^t\\&=WN_i^{t+1}+\gamma(WN_i^t+\gamma WC_i^{t-1})\\&=WN_i^{t+1}+\gamma WN_i^t+\gamma^2WN_i^{t-1}+\cdots+\gamma^nWN_i^{t-n+1}\\&=\sum_{l=0}^{t+1}\gamma^lWN_i^{t+1-l}\end{aligned} \tag{10}$$

因为 γ 一般是小于 1 的,在 t_1 时刻数据对 t_2 时刻数据的影响是呈现指数递减的,为 $\gamma^{(t_2-t_1)}$。

定义 9(聚类中心) 对于 $t+1$ 时刻得到的每个聚类中心 C_i^{t+1},权重为 WC_i^{t+1}。

3.2 OAP-s 算法步骤及算法实现

综上,OAP-s 算法步骤描述如下。

Step 1 初始化最大迭代次数 maxits、收敛系数 λ、衰减因子 γ、滑动时间窗口大小 N,并设 $t=0$。

Step 2 依次读取 N 个新的数据向量,$t=t+1$。

(a)将每个数据向量的权重设为 1;

(b)构造相似度矩阵;

(c)通过 AP 算法得到相应的聚类中心;

(d)更新聚类中心,分别将聚类中心赋予权重,并进行衰减。

Step 3 重复 Step 2,直到数据流的结束或者人为结束。

Step 4 输出结果为聚类中心和权重。

算法 1 OAP-s 算法

输入:最大迭代次数,震荡因子,衰减因子,要进行处理的数据
输出:聚类中心和权重
步骤:

```
1. 读取 N 个文本数据
2. while N≠Ø do
3.   t←t+1
4.   foreach X_n∈N^{t+1} do
5.     WX_n←1;
6.   end foreach
7.    if 数据及历史数据权重构建完毕 then
8.      ComputeSimilarity();
9.        if 相似度矩阵构建完成 then
10.           Apcluster(s,p);
11.           if C_i^{t+1}不为空 then
12.             UpdateClusterCenter();
13.           end if
14.       end if
15.    end if
16.    输出聚类结果 C_i^{t+1};
17. end while
```

3.2.1 相似度计算

针对文本数据高维、稀疏的特点,本文采用如下降维方法。首先对整个文档建立一个单词索引;然后将得到的〈*key*, *value*〉转换为〈*index*, *value*〉,其中 *index* 指的是单词的序号,*value* 指的是数值。由于所有文档的 *index* 都是从小到大进行排列的,进行相似度计算时在两篇文档的向量中按顺序寻找 *index*,如果两篇文档的 *index* 值相等,那么将两篇文档相应 *index* 下的 *value* 相乘,按此累加,直到计算出两篇文档之间的相似度。这种处理方式降低了 AP 算法中相似度计算的复杂度,使得算法的计算效率更高,其具体实现如算法 2 所示。

算法 2 计算相似度过程

```
ComputerSimilarity()
1. foreach i∈N' do
2.   foreach j∈N' && j≠i do
3.     if(i≠Ø && j≠Ø ) then
4.       将文本数据 i 和文本数据 j 中的所有相等的 index 对应的
```

value 值相乘，然后相加，得到文本数据 i 和文本数据 j 的相似度值。

5. end if
6. end foreach
7. end foreach
8. S(i,i)←所有的相似度值中值

算法 2 中 N' 为每时刻处理的数据 N 与上一时刻历史聚类中心 C_i 个数相加之和。其中步骤 4 的过程可描述如下：

l_{index}[ ]、l_{value}[ ]分别存储的是 i 的 index 和 value

j_{index}[ ]、j_{value}[ ]分别存储的是 j 的 index 和 value

while(ia<i_{index}[]的长度 && jb<j_{index}[]的长度)

do

if(i_{index}[ia]==j_{index}[jb]) then

S←S+i_{value}[ia]·j_{value}[jb];

ia←ia+1;jb←jb+1;

else if (i_{index}[ia]>j_{index}[jb]) then

jb←jb+1

else ia←ia+1;

end if

end if

3.2.2 更新聚类中心

为了使得到的聚类中心在下一次聚类过程中更具有代表性，将每次得到的聚类中心根据其内部包含的数据信息进行更新，然后统一进行归一化处理，最后将处理结果用在下一次的聚类过程中。经过归一化处理的聚类中心更能代表本次的聚类结果。具体实现如算法 3 所示。

算法 3 更新聚类中心

UpdateClusterCenter()

1. 初始化 k 个全维空数组 $V_1 \cdots V_k$
2. foreach C_i^{t+1} do
3. foreach $X_n \in N_i^{t+1}$ do
4. $V_i \leftarrow V_i + X_n \cdot WX_n$
5. end foreach
6. foreach $Y_n \in C_i^t$ do
7. $V_i \leftarrow V_i + Y_n \cdot WY_n$
8. end foreach
9. 将得到的 V_i 中的 value 值进行归一化

C_i^{t+1}←归一化的 V_i

$WC_i^{t+1} \leftarrow WN_i^{t+1} + \gamma WC_i^t$

10. end foreach

3.3 时间复杂度分析

根据算法 1，设所处理的文本数据集维度为 M，每次处理的文档数为 N，则 OAP-s 算法的时间复杂度如下。

(1)构建相似度矩阵 *ComputeSimilarity*( )的时间复杂度为 $O(M' \cdot N \cdot N)$，其中 M' 指的是两篇文档的 *index* 长度之和。对于两篇文档 i 和 j，计算时间复杂度为 O(文档 i 的 *index* 个数+文档 j 的 *index* 个数)，一般来讲 $M' \ll M$。

(2)*Apcluster*(s, p)的时间复杂度主要在于信息量的迭代。其算法复杂度为 $O(\rho \cdot N \cdot N)$，其中 ρ 指的是算法迭代的次数。一般来说，算法通常不会达到最大迭代次数，除非算法不收敛，因此 $1 < \rho \ll N$[14]。

(3)更新聚类中心 *UpdateClusterCenter*( )的时间复杂度为 $O(k \cdot M)$，其中 k 为聚类得到的中心数量。

则整个算法的时间复杂度为：$O((M'+\rho) \cdot N \cdot N + k \cdot M)$。

4 OWAP-s 算法

4.1 OWAP-s 算法设计思想

考虑到 OAP-s 算法更新聚类中心需要较高的时间复杂度，提出一种 OWAP-s 算法，即在算法中不更新聚类中心，而是给予聚类中心一个权重。由于 AP 算法得到的聚类中心是数据集中的一个数据点，因此将此点定义为代表点来参与下一时刻的聚类。代表点的权重越高，表明代表点的吸引度越高，此聚类越活跃，越容易成为聚类中心。基于此思想，根据文献[15]中带权重的 AP 算法(Weighted AP，WAP)，构建了带权重的不对称相似度矩阵。

定义 10 带权重的相似度矩阵：

$$S(i,j)=\begin{cases} -(1+\dfrac{w_i}{w_j}) \cdot \dfrac{1}{\cos(i,j)}, & i \neq j \\ (1+\dfrac{1}{w_i}) \cdot s^*, & i=j \end{cases} \tag{11}$$

其中，w_i 指的是文本数据点 i 的权重，w_j 指的是文本数据点 j 的权重。$\cos(i,j)$指的是文本数据点 i 和文本数据点 j 之间的余弦相似度。

定理 1 s^* 指的是各个代表点之间相似度中值的 1/2。

证明：当求自相似度时，由于 $P(i)$一般取相似度的中值，当 $w_i=1$ 时，$P(i)=2s^*$，$s^*=\dfrac{1}{2}P(i)$。

定理 2 权重越大的数据点越稳定，越不容易被聚到其他类中。

证明：根据文献[11]中 $S(i,j)$的定义，$S(i,j)$代表数据点 j 适合作为数据点 i 的聚类中心的程度。由式(11)可知，在数据点 i 和数据点 j 不相等时，如果 $w_i=w_j$，那么 $S(i,j)=\dfrac{-2}{\cos(i,j)}$。如果 $w_j \neq w_i$，w_j 越大，相应的$\dfrac{w_i}{w_j}$越小，即 $S(i,j)$越大，数据点 j 作为数据点 i 的聚类中心的可能性越高。

定理 3 权重越大，$P(i)$越大。

证明：当 $w_i>1$ 时，$1+\dfrac{1}{w_i}<2$，因 s^* 是负值，则 $P(i)$变大。

根据样本点间的吸引度和归属度的计算公式(1)－(6)，可以得到以下结论：

(1)$S(i,j)$较大时，使得 $R(i,j)$也较大；

(2)$P(i)$较大使得 $R(j,j)$较大，$A(i,j)$也就较大。

综上所述，在代表点的判定中，$e(i,j) \leftarrow A(i,j)+R(i,j)$，决策值 $e(i,j)$越大，文本数据点 j 作为最终聚类中心的可能性就越大。

本文引入吸引因子 δ，吸引因子 δ 的取值范围为[0,1]，即若某次聚类中，选为聚类中心的点在下一次聚类中的权重为 $1+\delta$，即 $WC_i=1+\delta$。在权重设置上不宜过大，这样可以避免由于聚类中心权重过大而影响聚类效果。δ 越小表明衰减程度越高，历史数据所占的比重越低；反之衰减程度越低，历史数据所占的比重越高。

4.2 算法步骤及算法实现

Step 1 初始化最大迭代次数 maxits、收敛系数 λ、吸引因子 δ、滑动时间窗口大小 N、$t=0$。

Step 2　依次读取 N 个新的数据向量，$t=t+1$。

(a)将每个数据向量的权重设为 1；

(b)构造加权相似度矩阵；

(c)通过 AP 算法得到相应的聚类中心；

(d)分别将聚类中心赋予权重。

Step 3　重复 Step 2，直到数据流的结束或者人为结束。

Step 4　输出结果为聚类中心和权重。

算法 4　OWAP-s 算法

输入：最大迭代次数，震荡因子，吸引因子，要进行处理的数据

输出：聚类中心和权重

```
1. 读取 N 个文本数据
2. while N≠Ø do
3.   t←t+1;
4.   foreach X_n∈N^{t+1} do
5.     WX_n←1;
6.   end foreach
7.   if 数据及历史数据权重构建完毕 then
8.       ComputeSimilarity();
9.     if 相似度矩阵构建完成 then
10.       Apcluster(s,p);
11.     end if
12.  end if
13. 输出聚类结果 C_i^{t+1};
14. 对聚类中心 C_i^{t+1} 赋予权重，WC_i^{t+1}←1+δ;
15. end while
```

由于不需要进行聚类中心的更新，因此 OWAP-s 算法的时间复杂度为：$O((M'+\rho)\cdot N\cdot N)$。

5　实验与结果分析

本节首先介绍实验采用的数据集以及聚类评价标准，然后进行 OAP-s 算法、OWAP-s 算法和 OSKM 算法等算法之间的聚类结果比较，并用其中样本最大的数据集分析不同参数对实验结果的影响。

5.1　文本数据集

实验中采用 NG20 数据以及 CLUTO 中的数据集[16]。表 1 列出了这些数据集的基本信息。对于每个数据集，N 代表文档的总数，d 代表单词的总数量，K 代表类数，$\frac{N_{nz}}{Nd}$ 代表文本矩阵中非零的比例。由表 1 可以看出这些数据集均具有高维稀疏性[10]。

表 1　文本数据集的基本信息

Data	N	d	K	$\frac{N_{nz}}{Nd}$
NG20	19949	43586	20	0.0018
Classic	7094	41681	4	0.0008
Ohscal	11162	11465	10	0.0053
k1b	2340	21839	6	0.0068

5.2　聚类结果比较

文中采用平均 NMI 值作为聚类评价标准。

平均 NMI 的计算公式为 $\frac{I(Y;\hat{Y})}{\sqrt{H(Y)\cdot H(\hat{Y})}}$，其中 Y 代表聚类结果标签，$\hat{Y}$ 代表类标签。平均 NMI 数值越接近 1 表示聚类结果越好，越接近 0 表示聚类结果越差。

对 4 个文本数据集进行 random 处理，分别得到 3 种不同的数据流，然后取得平均值，结果如表 2 所列。其中 SPKM、OSKM、OSKM-s、CLUTO 算法的处理结果来自文献[10]。

表 2　NMI 值

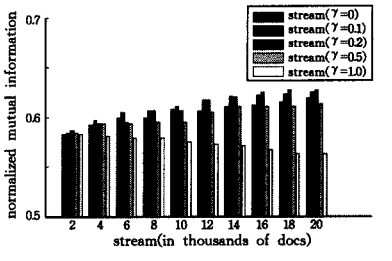
	NG20	ohscal	classic	k1b
K	20	10	4	6
SPKM	0.54±0.02	0.42±0.03	0.55±0.04	0.56±0.05
OSKM	0.59±0.01	0.44±0.01	0.63±0.03	0.66±0.03
OSKM-s	0.59±0.01	0.45±0.01	0.59±0.03	0.65±0.03
CLUTO	0.58±0.01	0.44±0.00	0.55±0.01	0.62±0.03
OAP-s 算法	0.62±0.01	0.50±0.02	0.66±0.01	0.71±0.01
OWAP-s 算法	0.62±0.01	0.51±0.01	0.66±0.02	0.72±0.01

从表 2 中可以看出，OAP-s 算法和 OWAP-s 算法在聚类精确度方面要优于其他几种方法。

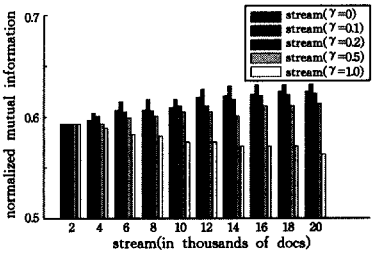
5.3　算法性能分析

为了进一步分析算法性能及参数对算法的影响，采用样本数据最多、稀疏性最强的 NG20 数据集进行分析。为了模拟真实流数据环境，将 20-newsgroup 文本到来顺序进行 random 处理，得到 3 条不同的文本数据流：NGStream1、NGStream2、NGStream3。对这 3 条数据流分别运行 OAP-s 和 OWAP-s 算法，最终结果取 3 次平均值。

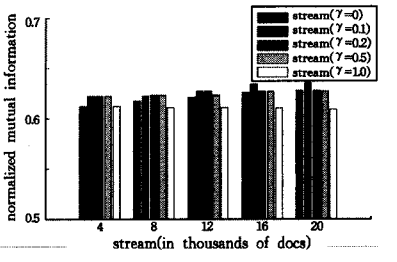
图 2 示出 OAP-s 算法分别在 $N=1000$、2000 和 4000 时，衰减因子 γ 分别为 0、0.1、0.2、0.5、1.0 时的聚类 NMI 值。



(a)$N=1000$ 时 OAP-s 算法的 NMI 值



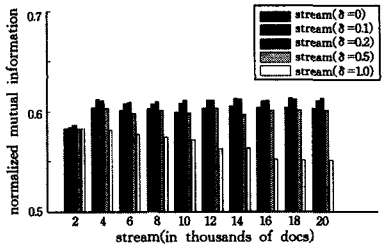
(b)$N=2000$ 时 OAP-s 算法的 NMI 值



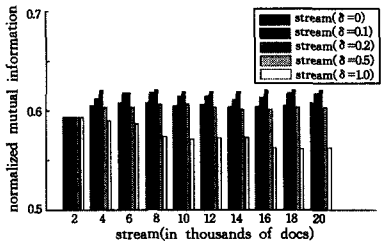
(c)$N=4000$ 时 OAP-s 算法的 NMI 值

图 2

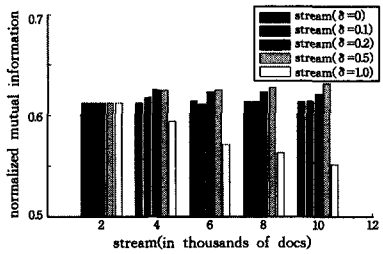
图 3 示出 OWAP-s 算法分别在 $N=1000$、2000 和 4000 时，吸引度因子 δ 分别为 0、0.1、0.2、0.5、1.0 时的聚类 NMI 值。实验中将权重投影到区间[1,2]上，即 $1\leqslant WC_i=1+\delta\leqslant 2$，选为聚类中心的点在下一次聚类中的权重为 $1+\delta$。



(a)N=1000 时 OWAP-s 算法的 NMI 值



(b)N=2000 时 OWAP-s 算法的 NMI 值



(c)N=4000 时 OWAP-s 算法的 NMI 值

图 3

下面分别从准确率、稳定性和算法效率 3 个方面说明两个算法的性能。

(1)准确率

从运行结果看，OAP-s 算法和 WAP-s 算法由于不依赖于初始聚类，因此在聚类准确率上均优于 OSKM。

其次，对于 OAP-s 算法，衰减因子 γ 越大说明历史数据占的比重越大，对 NMI 结果的影响越大；反之越小。从图 2 中可以看到，在固定的时间窗口下，OAP-s 算法在衰减因子 γ 为 0.1 或 0.2 时的 NMI 值要高于 γ 为 0.5 和 1 时的 NMI 值。

在固定衰减因子的情况下，时间窗口内样本数据量 N 越大，OAP-s 算法的准确率越高且鲁棒性越强。例如当 γ 取0.1 时，图 2(c)的整体 NMI 均值比图 2(b)高，图 2(b)的整体 NMI 均值比图 2(c)要高。图 2(c)的 NMI 值的标准差要明显比图 2(a)和图 2(b)小，且随着数据点的不断流入，聚类中心包含的信息越来越多，得到的聚类结果呈现越来越好的趋势。

最后，对于 WAP-s 算法，吸引度因子 δ 越大，表明历史数据所占的比重越大，即重要性越高；反之越小。从图 3(a)—(c)中可以看到，在时间窗口固定的情况下，OWAP-s 算法在吸引度因子 δ 为 0.2 左右时的 NMI 值较高。在 δ 固定后，每次处理的数据量越多，得到的聚类中心代表性越强，相应的聚类准确率越好。即图 3(c)的整体 NMI 值优于图 3(b)，图 3(b)的整体 NMI 值优于图 3(a)。但在时间窗口固定的情况下，如果对聚类中心的吸引度赋予的值非常大，即 δ 越大，那么会使得聚类结果越来越差，因为这些聚类中心点会吸引原本不属于此聚类中心的数据点。

(2)稳定性

由于 AP 算法可以随着数据点类别的变化自动调节得出结果的聚类个数，避免了 k-means 算法需固定聚类个数所带来的弊端；且 OAP-s 和 WAP-s 算法不仅比较新来的数据点和历史数据点之间的相似度，而且还比较各个新数据点之间和历史数据点之间的相似度值，对每个数据点之间的关系综合考虑，对权重进行衰减，进一步降低权重的影响，使得聚类结果不会太差。所以两种算法的稳定性均优于 OSKM 算法。

其次，对于 OAP-s 算法，其参数 γ 对算法的影响不是很大。对于 OWAP-s 算法，吸引度因子 δ 过小会导致 t 时刻聚类中心在 $t+1$ 时刻聚类过程中的作用不明显；过大则会导致把本应属于其它类中的数据聚到另外一聚类中心中，进而影响聚类准确率。比如图 3 在吸引度因子 δ 为 1 时随着数据的不断流入效果越来越差。总之，吸引度因子 δ 过大或者过小都可能导致聚类结果不理想，而且 OWAP-s 算法中的吸引度因子的取值大小会受到不同类别数据之间距离的影响，所以 OAP-s 算法的稳定性要好于 OWAP-s 算法的稳定性。

(3)时间效率

OWAP-s 算法的时间复杂度为：$O((M'+\rho)\cdot N\cdot N)$。OAP-s 算法由于需要更新聚类中心，算法的时间复杂度为：$O((M'+\rho)\cdot N\cdot N+k\cdot M)$。很明显，$O((M'+\rho)\cdot N\cdot N)<O((M'+\rho)\cdot N\cdot N+k\cdot M)$，所以 OWAP-s 算法的时间效率要高于 OAP-s 算法。

整体上，由于先前滑动时间窗口的聚类中心用在下一滑动时间窗口的聚类中，影响了下一滑动时间窗口的聚类结果，因此滑动时间窗口下的聚类中心的选取非常重要，而且要赋予比较合理的权重。滑动时间窗口中包含的数据量越多，得到的聚类中心代表性越强。

结束语 本文针对目前网络上产生的大量的非结构化文本数据，将传统的 AP 算法及流式文本数据的特征相结合，首先引入衰减因子，提出了 OAP-s 算法，在此基础上根据 AP 算法相似度的基本概念对 WAP 算法进行改进，提出OWAP-s算法，并将两者用到文本数据流中。实验结果表明，这两种算法的聚类精确度均高于 OSKM 算法。由于实验中数据流入类别数目会影响聚类结果，下一步的研究工作是如何在数据流入类别数目变化较大的情况下分析聚类结果。

参 考 文 献

[1] PhridviRaja M S B, Chintakindi, Srinivasb, et al. Clustering Text Data Streams-A Tree Based Approach with Ternary Function and Ternary Feature Vector [J]. Procedia Computer Science, 2014(31): 976-984

[2] Huang Guang-yan, He Jing. Mining Streams of Short Text for Analysis of World-wide Event Evolutions[J]. World Wide Web, 2015, 18(5): 1201-1217

[3] Zhang Jian-peng, Chen Fu-cai, Li Shao-mei, et al. Online Clustering Algorithm for Evolutionary Data Stream Based on Affine Propagation [J]. Pattern Recognition and Artificial Intelligence, 2014, 27(5): 443-451(in Chinese)
张建朋，陈福才，李邵梅，等. 基于仿射传播的进化数据流在线聚类算法[J]. 模式识别与人工智能，2014，27(5)：443-451

[4] Aggarwal C C. Mining Text Streams[M]// Mining Text Data, 2012: 297-321

[5] Gong Ling-hui, Zeng Jian-ping, Zhang Shi-yong. Text Stream

Clustering Algorithm Based on Adaptive Feature Selection[J]. Expert Systems with Applications, 2011, 38(3): 1393-1399

[6] Aggarwal C C, Han J W, Wang J Y, et al. A Framework for Clustering Evolving Data Streams[C]// Proc of the 29th International Conference on Very Large Data Bases. Berlin, Germany, 2003: 81-92

[7] Aggarwal C C, Yu P S. On Clustering Massive Text and Categorical Data Streams[J]. Knowledge and Information Systems, 2010, 24(2): 171-196

[8] Aggarwal C C, Han J W, Wang J Y, et al. A Framework for Projected Clustering of High Dimensional Data Streams[C]// Proc of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 852 -863

[9] Liu Y B, Cai J R, Yin J, et al. Clustering Text Data Streams[J]. Journal of Computer Science & Technology, 2008, 23(1): 112-128

[10] Shi Zhong. Efficient Streaming Text Clustering[J]. Neural Networks, 2005, 18(5/6): 790-798

[11] Frey B J, Dueck D. Clustering by Passing Messages between Data Points[J]. Science, 2007, 315(5814): 972 -976

[12] Guo Xiu-juan, Chen Ying. Analysis and Application of AP Clustering algorithm [J]. Journal of Jilin Jianzhu University, 2013, 30(4): 58-61(in Chinese)
郭秀娟, 陈莹. AP 聚类算法的分析与应用[J]. 吉林建筑大学学报, 2013, 30(4): 58-61

[13] Strehl A, Ghosh J, Mooney R J. Impact of Similarity Measures on Web-page Clustering[C]// AAAI Workshop on AI for Web Search. 2000: 58-64

[14] Zhang Zhen, Wang Bin-qiang, Yi Peng, et al. A Hierarchical Combination of Semi Supervised Neighbor Propagation Clustering Algorithm [J]. Journal of Electronic & Information Technology, 2013, 35(3): 645-651(in Chinese)
张震, 汪斌强, 伊鹏, 等. 一种分层组合的半监督近邻传播聚类算法[J]. 电子与信息学报, 2013, 35(3): 645-651

[15] Zhang X L, Furtlehner C, Sebag M. Data Streaming with Affinity Propagation[C]// Proc of the European Conference on Machine Learning and Knowledge Discovery in Databases. Antwerp, Belgium, 2008: 628-643

[16] Karypis G. CLUTO-a clustering toolkit [OL]. 2002. http://www-users. cs. umn. edu/~karypis/cluto

(上接第 197 页)

模型元素的聚类分析[8]，都未考虑子流程构成行为之间的控制流相关性，生成的候选子流程中的构成行为结构松散，甚至无法保证模型的保序性[17]，因此，对生成的候选子流程进行控制流修正以及构建之间控制流关系是下一步的研究内容。另外，在第二部分实验中发现，人工修正后的流程片段之间出现很多重复的行为甚至其它流程片段，这也是本文方法存在的问题，即初始模型中的每个行为只属于一个抽象行为，因此，接下来的改进算法将重点研究解决行为或流程片段重用问题。

参 考 文 献

[1] Mendling J, Reijers H A, van der Aalst W M P. Seven Process Modeling Guidelines (7pmg) [J]. Information and Software Technology, 2010, 52(2): 127-136

[2] Smirnov S, Dijkman R, Mendling J, et al. Meronymy-based aggregation of activities in business process models[J]. Conceptual Modeling-ER 2010, Lecture Notes in Computer Science, 2010, 6412: 1-14

[3] Smirnov S, Reijers H A, Weske M H, et al. Business process model abstraction: a definition, catalog, and survey[J]. Distributed and Parallel Databases, 2012, 30(1): 63-99

[4] Smirnov S. Business Process Model Abstraction[D]. Germany: University of Potsdam, 2012

[5] Polyvyanyy A, Smirnov S, Weske M. Reducing Complexity of Large EPCs[C]// MobIS. Saarbrücken, Germany, 2008: 195-207

[6] Polyvyanyy A, Smirnov S, Weske M. On Application of Structural Decomposition for Process Model Abstraction[C]// Proceedings of the BPSC 2009. Leipzig, 2009: 110-122

[7] Vanhatalo J, Völzer H, Koehler J. The Refined Process Structure Tree[C]// Proceedings of the 6th International Conference on Business Process Management (BPM 2008). Milan, Italy, 2008: 100-115

[8] Smirnov S, Reijers H A, Weske M. A Semantic Approach for Business Process Model Abstraction[M]// Advanced Information Systems Engineering: 23rd International Conference(CAiSE 2011). London, UK, June 20-24, 2011. Springer, 2011: 497-511

[9] Weidlich M, Dijkman R, Mendling J. The ICoP framework-Identification of correspondences between process models, Advanced Information Systems Engineering[M]// Advanced Infromation Systems Engineering: 22nd International Conference (CAiSE 2010). Hammamet, Tunisia, June 7-9, 2010. Springer, 2010: 483-498

[10] Reijers H A, Mendling J, Dijkman R M. On the Usefulness of Subprocesses in Business Process Models: BPM Center Report BPM-10-03[R]. BPMcenter. org, 2010

[11] Qu Y, Hu W, Cheng G. Constructing virtual documents for ontology matching[C]// Proceedings of the 15th International Conference on World Wide Web. Edinburgh, Scetland, UK, 2006: 23-31

[12] Porter M F. An algorithm for suffix stripping [J]. Program, 1980, 14(3): 130-137

[13] Euzenat J, Shvaiko P. Ontology matching[M]. Springer-Verlag, 2007

[14] Schae_er S E. Graph Clustering[J]. Computer Science Review, 2007, 1(1): 27-64

[15] Zhou S, Xu Z, Tang X. New method for determining optimal number of clusters in K-means clustering algorithm [J]. Computer Engineering and Applications, 2010, 46(16): 27-31

[16] Franzblau A N. A Primer of Statistics for Non-statisticians [M]// Harcourt, Brace & World New York. 1958

[17] Liu D, Shen M. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach[J]. Information Systems, 2003, 28(6): 505-532