基于滑窗不等长时间序列 STS 距离的聚类算法

刘琴[1)]，王恺乐[2)]，饶卫雄[3)]

[1,2,3)] 同济大学 软件学院 上海市曹安公路 4800 号 中国 201804

[1)] qin.liu@tongji.edu.cn [2)] wangkaile@outlook.com [3)] wxrao@tongji.edu.cn

摘 要 时间序列的聚类算法是分析预测互联网搜索对象搜索指数和社交网络话题热度随时间变化趋势的重要过程，但目前时间序列聚类算法的研究存在两点不足。首先国内外的时间序列聚类的研究都采用等长划分的时间序列，这往往会丢失许多重要特征点，对数据挖掘的结果产生一定负面影响。其次现有工作均直接使用时间序列观测值不能准确的度量时间序列的形状相似度。因此，本文通过标准分数预处理消除时间序列观测值数量级差异影响，并设计基于滑窗的不等长时间序列 STS 距离和类 k-means 聚类算法中心曲线计算算法，最终提出基于滑窗不等长时间序列 STS 距离的聚类算法，从而解决了不等长时间序列聚类问题。本文采集互联网上的真实数据集作为测试样本，进行大量实验。实验结果表明基于滑窗不等长时间序列 STS 距离的聚类算法在消除时间序列观测值数量级差异影响，并解决了不等长时间序列聚类问题的同时，比现有算法取得更优的聚类效果。

关键词 聚类；时间序列；k-mean 算法

中图法分类号 DOI 号：

A STS Distance-based Non-equal Time Series Data Clustering Algorithm

Qin Liu[1)], Kaile Wang[2)], Weixiong Rao[3)]

[1,2,3)](School of Software Engineering, Tongji University, Cao'an Road 4800, Shanghai, China, 201804

[1)] qin.liu@tongji.edu.cn [2)] wangkaile@outlook.com [3)] wxrao@tongji.edu.cn

Abstract Time-series clustering is an important algorithm by many applications. For example, the analysis and forecast of topics on social media and search words on search engine. However, the current time-series clustering algorithms suffer from two shortcomings. First, existing time-series clustering algorithms mostly work only for isometric time-series with equal length. Secondly, existing time series similarity are not able to compare the shapes similarity of time series data that are irrelevant to the absolute values metrics. Both shortcomings have major negative impact on clustering results. To address the problem, in this paper, we propose a novel computation framework to cluster time series data with unequal length. At first, we use z-score standardization to normalize the absolute values of time series data. Next, we extend STS distance and design a new distance measure for time-series with unequal time length. After that, we adapt the classic k-mean algorithm to develop a clustering algorithm for the time series data by finding a time series centroid. We conduct experiments based on two real datasets that are collected from real search engines and public data. Extensive experimental results show that our time-series clustering algorithm can outperform the state of arts in terms of clustering accuracy and quality.

Key words Clustering; Time-Series; K-mean;

————————————————

1 引言

当前互联网迅猛发展，搜索引擎和社交网络得到广泛应用，已经成为人们日常的使用工具。这导致互联网每时每刻都会产生大量的搜索对象或社交网络中的讨论话题。这些搜索对象的搜索次数或讨论话题的关注热度随着时间不断变化，形成典型的时间序列数据。

如何分析搜索对象或讨论话题的时间变化规律，预测其发展趋势，提前掌握互联网流行动向并检测网络舆情，学术界开展广泛的研究。目前的研究方法主要分为(*i*)基于内容的分析方法和(*ii*)基于时间序列的分析方法两类。其中基于内容的分析方法[1, 2]通常需要先事提取影响搜索次数或话题热度的影响因素，然后通过回归、贝叶斯网络和朴素贝叶斯等方法对提取的影响因素数据进行分析。这种基于内容的分析方法往往难以刻画搜索次数或话题热度的时间发展趋势。

而时间序列数据因其趋势信息的直观展现形式，已经广泛应用于社交网络、互联网搜索和新闻媒体数据的分析。比如，Google 公司曾用其搜索引擎中搜索流感相关信息的时间序列预测流感的爆发趋势[3]。

不同的时间序列数据（如多个搜索关键字对应搜索频次时间序列数据，多个 Twitter 讨论话题对应的热度时间序列数据），其形状趋势具有不同的规律性[4]，通过分类或聚类的方法，我们可以区分不同类型的时间序列数据。比如通过分类 Twitter 话题对应的热度时间序列数据，同一类簇的 Twitter 话题具有相同或相似的发展趋势，进而应用于预测话题的发展趋势预测。MIT 的 Devavrat Shah 教授通过其开发的二元时间序列分类算法成功预测推特的热门话题[5]。相较于依赖训练集的分类算法，时间序列聚类算法的研究更具有普遍意义。

不过当前时间序列聚类的研究还存在如下两点不足。第一，目前国内外时间序列聚类的研究中时间序列的划分都采用等长划分，但是现实生活的中时间序列的长度往往是不等长(例如不同互联网搜索对象和社交网络话题的生命周期一般不同，所以描述它们的时间序列长度也通常不同)，这往往会丢失许多重要特征点，对数据挖掘的结果产生一定负面影响[6]。因此，针对不等长的时间序列聚类研究具有更加现实意义。

第二，目前时间序列的相似性度量和聚类算法往往是针对等长时间序列，而不等长时间序列距离的度量和聚类算法等目前还没有成熟的研究成果。这里以关键字搜索频次的时间序列数据为例，不同的关键字尽管其搜索频次差异较大，但是可能呈现相同或相似的时间发展趋势。如图 1 所示三个搜索对象“埃博拉出血热”、“卫生部”和“教育部”的时间序列数据，尽管“埃博拉出血热”、“卫生部”对应的搜索频次每个点的**数值**相对接近，但是“卫生部”和“教育部”的时间序列曲线明显具有更加相似的**形状趋势**。因此，如何针对时间序列的形状趋势、而非观测值，设计时间序列之间的距离，是时间序列数据聚类算法的关键内容之一。 Carla 等人[7]提出的 STS 距离能忽略时间序列的观测值数量级差异，相对欧式距离更好地描述时间序列间的形状相似度，但和欧式距离一样，STS 距离并不适用于不等长时间序列。

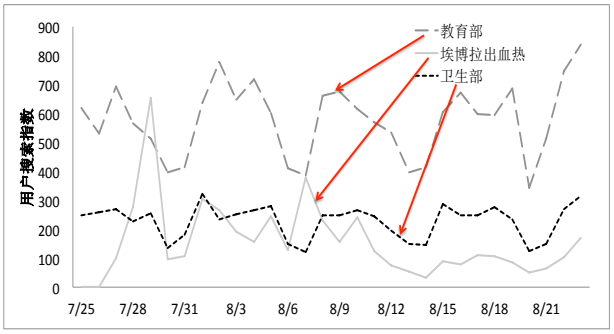


图 1 “埃博拉出血热”、“卫生部”、“教育部”3 个搜索对象的搜索指数序列图

针对以上的不足，本文设计不等长时间序列数据的聚类算法，首先采用标准分数对时间序列观测值进行预处理，以消除时间序列数据观测值数量级差异的影响；其次，基于滑窗在 STS 距离的基础上提出不等长时间序列距离的度量方法；最后通过改进 k-means 聚类算法，设计出一个新颖的基于滑窗不等长时间序列 STS 距离的聚类算法。

基于真实数据的实验结果表明：(*i*) 基于滑窗不等长时间序列 STS 距离的聚类算法可以有效解决时间序列不等长问题和观测值数量级差异问题；(*ii*) 在同一实验条件下，基于滑窗不等长时间序列 STS 距离的聚类算法与传统补零以取得等长时间序列的 k-means 聚类算法相比，其聚类质量及准确率更高。

本文第 2 节回顾相关研究工作；第 3 节描述相似度公式和聚类过程；第 4 节介绍本文的实验设置

与实验结果分析；第 5 节进行总结并探讨下一步的研究方向。

2 相关研究

时间序列聚类具有无需训练数据集、完全根据自身信息进行数据挖掘的特点。文献[8]提出面向话题时间序列的 K_SC 聚类算法，能有效地描述聚类结果中各簇的中心曲线(中心曲线的定义参考 3.1 章节)，较好地刻画话题内在发展趋势特征，精确度较高。不过 K_SC 聚类算法存在两个缺点：对初始类矩阵中心敏感且时间复杂度较高。针对这两个缺点，文献[4]对 K_SC 聚类算法进行优化，得到时间复杂度更低，聚类效果更好的 WKSC 聚类算法。不过，这些聚类算法[4, 8]均假定等长时间序列，没有考虑现实生活中大量存在不等长时间序列的情况。

综述[9]将时间序列的聚类算法分为两类：(1)基于原始数据的时间序列聚类方法；（2）基于特征的时间序列聚类方法。基于特征的时间序列指用时间序列的形态特征（极值点位置、分段斜率等）、结构特征（统计特征，包括平均值、方差等统计特征值）和模型特征数据替代时间序列观测值本身，从而进行针对这些特征值计算的聚类方法。基于特征的时间序列聚类方法可以解决不等长时间序列的聚类问题，但这类方法通常忽略或减弱原始数据值的影响，聚类结果的形状趋势信息往往比较粗糙。有关基于原始数据的时间序列聚类方法的，文献[9]中总结的距离度量方法和聚类算法并不适用于不等长时间序列数据。

针对欧式距离度量时间序列数据间的相似度时，存在忽略时间序列局部特征且易受观测值数量级差异影响的缺点，文献[7]提出 STS 距离。 但是，和欧式距离等一样，STS 距离也不适用于度量不等长时间序列。

DTW 距离是最常见的不等长时间序列距离度量方法，最早被提出用于解决语音识别中发音长短不一的模板匹配问题。文献[10]通过实验分析了 DTW 距离应用在时间序列聚类中的效果，但是 DTW 距离不适合 k-means 聚类算法，无法有效地描述聚类结果各簇的中心点。文献[5] 提出一种不等长时间序列距离的计算方法，应用于时间序列的二元分类算法；不过该方法同样缺乏簇中心点的描述方法，该距离度量方法同样不能直接使用在 k-means 类型的聚类过程中。

此外，文献[7, 11, 12]都使用标准分数 z_score 对时间序列观察数据进行预处理，去除观察值数量级差异对计算过程的影响。文献[12]通过实验比较证明了标准分数 z_score 预处理过程对去除观察值数量级差异的有效性。

最后，文献[13]提出 Average Silhouette 函数用于度量聚类效果，函数值和聚类的簇数相关。函数可以用于选择聚类簇数。文献[8]采用 Average Silhouette 函数作为选择聚类簇数的标准之一。

3 背景介绍

这里我们先介绍本文相关的背景介绍，包括相关的定义和距离计算公式等。

3.1 相关定义

1 **搜索指数**：对于一个搜索关键字查询对象（如“埃博拉出血热”），我们对在指定时间间隔内发生的搜索频次感兴趣。搜索指数用来表达每个搜索对象的搜索频度，目前不少互联网搜索引擎均推出各自的相关搜索指数，如谷歌趋势、百度指数、淘宝指数和搜狗指数等。

2 **搜索指数序列**：针对一个特定的搜索对象，通过记录一定时间范围内得到相应搜索指数的时间序列，称为搜索指数序列。根据搜索指数序列可以描述该搜索对象的搜索指数时间序列图（如图 1），表达该搜索对象在互联网上被搜索的趋势，即随时间推移，该搜索指数的变化规律。

3 **生命周期**：给定一个特定的时间范围，对应的搜索索指数序列长度可能与该搜索对象本身的生命周期长度并不一致。例如，在图 1 给定得时间范围中（2014 年 7 月 26 日到 2014 年 8 月 24 日），搜索对象“埃博拉出血热”在 7 月 27 日前搜索指数为零，“埃博拉出血热”不在其搜索对象生命周期中。在搜索索指数序列长度大于生命周期长度的情况下，其超出生命周期时间点的数值通常被标记为零。因此，在搜集得到的时间序列数据中，我们将第一个数值非零点到最后一个非零数值点间的时间范围，定义为时间序列的***生命周期***。互联网中搜索对象的搜索指数序列生命周期长度通常不一致。图 1 中，“埃博拉出血热”的时间序列生命周期就小于其他两个搜索对象的生命周期不同。

4 **中心点（曲线）**：针对聚类结果中每一个簇的所有时间序列成员，我们参考所有成员构造成一个特定的时间序列数据，该时间序列曲线称为该簇的中心点（曲线）。中心点（曲线）描述了一类时间序列的共同形状趋势特征。

5 **时间序列的局部特征**：时间序列的全局特征描述了整个生命周期内时间序列的形状信息。时间序列的局部特征描述了时间序列生命周期内某一阶段的形状信息。相比时间序列的全局特征，时间序列的局部特征在度量相似度的过程中更有参考价值[7, 14]。如图 2 所示的 3 条时间序列曲线，假定以欧式距离为度量标准(欧式距离定义如 3.2 章节所示)，g1、g2 和 g2、g3 时间序列间的距离都是 10.5 个单位，以此为依据，g1、g2 和 g2、g3 间的总体形状差异相同。但在第 1 个时间单位内 g1 和 g2 间的距离是 1.5 个单位，而 g2 和 g3 间的距离是 1 个单位。同理，在第 3 个时间点到第 10 个时间点间，g1 和 g2 间的距离是 8 个单位，而 g2 和 g3 间的距离是 8.5 个单位。在这些特定时间段内 g1、g2 和 g2、g3 间的欧式距离不等，这些信息反映了时间序列的局部形状信息差异，即时间序列的局部特征。

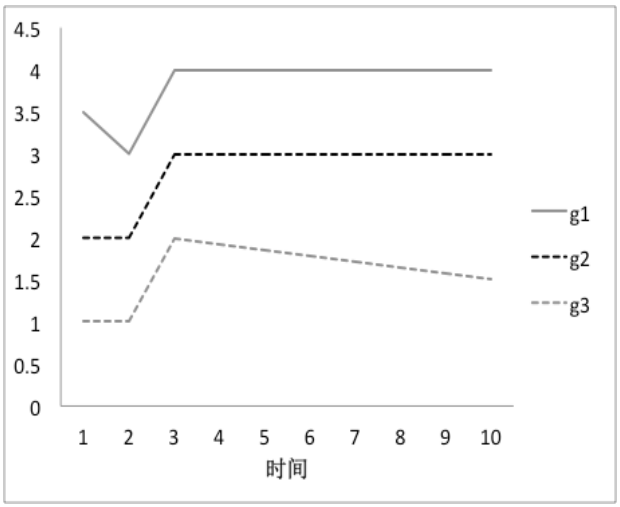


图 2 三条形状不同的时间序列曲线

6 **滑窗**：滑窗是提取时间序列上所有固定长度子序列的方法。对于生命周期长 m 的时间序列 T，其上所有长 l 的子序列都可以被长 l 的滑窗提取，记为S_p^l。其中 l 表示滑窗长度，p 表示滑窗在时间序列 T 上的起始位置。时间序列 T 上所有长 l 的子序列集合表示为$S_T^l = \{S_p^l \text{ of T, for } 1 \le p \le m - l + 1\}$**[14]**。

3.2 距离公式

时间序列的欧式距离指两条等长时间序列所有对应点之间欧式距离的累加和。公式如下：

$$d_E(x,y) = \sqrt{\sum_{k=1}^{P}(x_k - y_k)^2} \quad (1)$$

其中，时间序列 x 和 y 生命周期长为 P，x_k 和 y_k 分别代表时间序列 x 和 y 在时间点 k 所对应的数值。

STS 距离：欧氏距离所计算的时间序列间距离描述了时间序列间全局特征，但忽视了时间序列的局部特征，且其在度量形状差异时受观测数值数量级差异影响明显。文献[7] 提出了 STS 距离来描述时间序列间的形状差异，同时保证准确描述时间序列间全局特征的同时，还考虑时间序列的局部特征，且形状差异和时间序列间各个点观测数值的数量级差异无关。图 2 中 g1、g2 和 g2、g3 时间序列曲线间的欧式距离相等，但如表 1 所示，g2 和 g3 的 STS 距离小于 g2 和 g3 的 STS 距离。

STS 距离计算并累加时间序列间每个时间间隔斜率差的平方，公式定义如下：

$$d_{STS}^2(x,v) = \sum_{k=0}^{n_t-1}\left(\frac{v_{(k+1)} - v_k}{t_{(k+1)} - t_k} - \frac{x_{(k+1)} - x_k}{t_{(k+1)} - t_k}\right)^2 \quad (2)$$

其中 x 和 v 代表两条等长的时间序列，n_t 代表时间序列上时间间隔点的数量，t_k（k=0,1,… ,n_t）代表时间序列上的时间间隔点，v_k（k=0,1,… ,n_t）和 x_k（k=0,1,… ,n_t）分别代表在 k 时刻时间序列对应的值。

表 1 为 g_1、g_2 和 g_3 曲线经过标准分数预处理后 g_1 和 g_2、g_2 和 g_3 曲线间的欧式距离和 STS 距离。数据表明 STS 距离能够更好地描述时间序列的局部形状信息。

表 1 图 2 中 g_1 和 g_2、g_2 和 g_3 的欧式距离和 STS 距离

	欧氏距离	STS 距离
(g_1, g_2)	0.756	1.103
(g_3, g_2)	0.756	0.386

3.3 标准分数z_score

简单的用 STS 距离替代欧式距离并不能完全解决受时间序列观测数值数量级差异影响的问题，所以我们引入标准分数对时间序列进行预处理。

标准分数 z_score 用数据观测值与观测值平均值间的距离替换原观测值，并以标准差作为单位计量。经过标准分数预处理后的数据平均值为 0，标准差为 1，这样可以忽略观测值直接数值差异将数据放在等距量表中。经过标准分数预处理的时间序列不再受观测数值数量级差异影响，可以直接在等距量表中度量比较。时间序列的标准分数公式如下：

$$z_i = \frac{\left(x_i - \bar{x}\right)}{s_x} \quad (3)$$

其中 x_i 代表时间序列在时间点 i 的观测值，$\bar{x}$ 代表时间序列观测值的平均值，s_x 代表时间序列观测值的标准差，z_i 代表经过标准分数处理后的时间点 i 的数值。

图 1 中 3 个搜索对象的搜索指数序列经标准分数预处理后对应得到的时间序列曲线如图 3 所示。

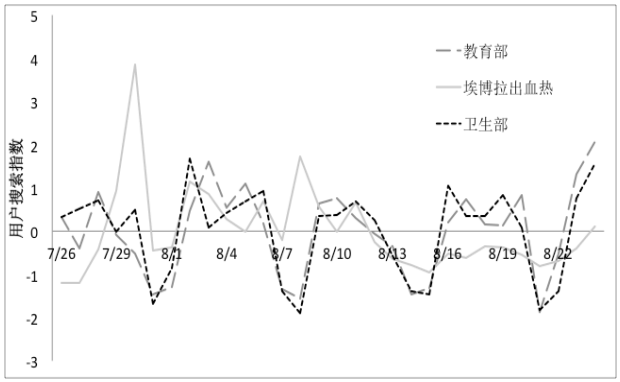


图 3 “埃博拉出血热”、“卫生部”、“教育部”三个搜索对象经过标准分数预处理后的时间序列图

图 3 中，经过标准分数预处理后，可以更加明显看出搜索“卫生部”和“教育部”的时间序列曲线明显具有相似的形状趋势。

4 算法设计

本章提出基于滑窗不等长时间序列 STS 距离的聚类算法，其基本思想是采用 k-mean 方法，首先将不等长时间序列分类到给定数量(如 k 个)的类簇中，每个类簇中的时间序列具有相似的形状趋势规律；然后拟合同一类簇中所有时间序列进行得到该簇的中心曲线，簇中心曲线则描述了所在簇所有时间序列的形状趋势特征。

以上过程的关键和困难之处在于设计适用于不等长时间序列的距离计算公式，以及如何拟合同一类簇中所有时间序列以得到该簇的中心曲线。因此，我们在 4.1 章节中设计基于滑窗不等长时间序列的 STS 距离以计算不等长时间序列的距离，然后在 4.2 章节分析如何拟合一簇时间序列形成中心曲线，最后我们设计基于滑窗不等长时间序列 STS 距离的聚类算法。

4.1 基于STS距离的不等长时间序列距离

文献[7]证明 STS 距离度量经过标准分数预处理后的时间序列可以精确地描述**等长时间序列**的形状趋势信息。STS 距离和标准分数联合使用解决了欧式距离度量时间序列局部特征信息缺失和受观测数值数量级差异影响较大的问题，但却无法用于度量不等长时间序列的距离。

针对不等长时间序列的距离度量，本文设计了基于滑窗不等长时间序列的 STS 距离。在计算两个不等长时间序列时，假定较长时间序列为 r，较短时间序列为 s。设 L_r 为时间序列 r 的长度，L_s 为时间序列 s 的长度，且 $L_r \geq L_s$。使用长度为 L_s 的滑窗提取时间序列 r 中所有长 L_s 的子序列，时间序列 r 上所有长 L_s 的子序列的集合记为 $S_r^{L_s} = \{S_p^{L_s} \; of \; r, for \; 1 \leq p \leq L_r - L_s + 1\}$。其中 p 表示滑窗在时间序列 r 上的起始位置。取集合中子序列到 s 的 STS 距离最小值，将该距离作为不等长时间序列 r 到 s 的 STS 距离。

上述计算结果和较短时间序列 s 的长度 L_s 正比例相关。针对这一问题，本文参考文献[15]将上述距离根据较短时间序列长度长度进行标准化改造，最终得到的基于滑窗不等长时间序列 STS 距离具体公式如下：

$$d(r,s) = \frac{\min\limits_{p=1,\ldots,L_r-L_s+1} d_{STS}^2\left(r_{p:p+Ls-1}, s\right)}{L_s} \quad (4)$$

$d_{STS}^2\left(r_{p:p+Ls-1}, s\right)$ 表示 s 到 r 上 k 到 $k+N_s-1$ 段子序列间的 STS 距离。

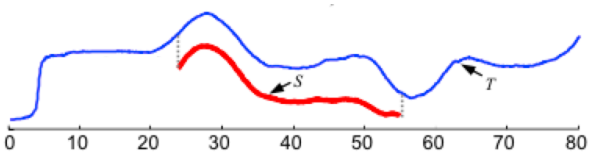


图 4 不等长时间序列 r 和 s 间距离计算过程

图 4 描述了基于 STS 距离的不等长时间序列距离计算的物理含义，当计算较长时间序列 r 和较短

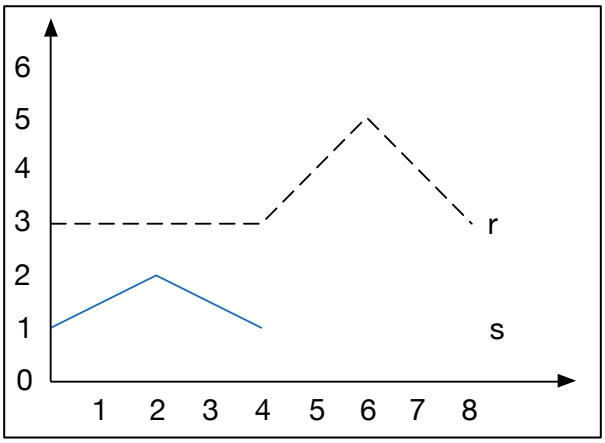
时间序列 s 间的距离时，首先不断平移时间序列 s，找到时间序列 r 到时间序列 s 的 STS 距离最近的子段。计算该子段到时间序列 s 间的 STS 距离并进行长度标准化改造后，作为时间序列 r 到时间序列间 s 的距离。

4.2 中心曲线计算

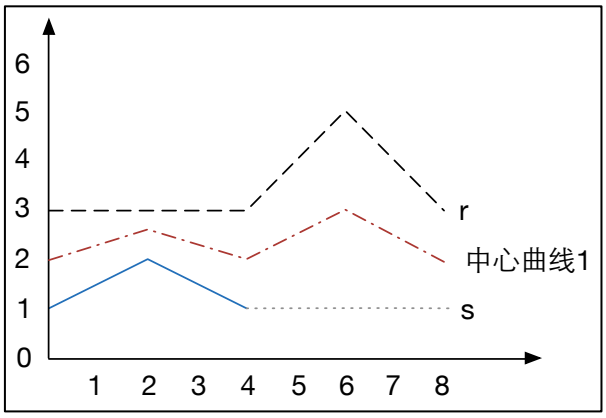
和 k-means 聚类算法相似，基于滑窗不等长时间序列 STS 距离的聚类算法也是一种迭代算法。k-means 聚类过程包括分配聚类对象和更新簇中心两步。基于滑窗不等长时间序列 STS 距离的聚类算法也同样迭代上述两个过程。因此，我们需要对同一类簇中的不等长时间序列数据进行拟合，形成该类簇对应的中心曲线。其关键在于如何对不等长的时间序列进行拟合。

在提出我们拟合方法之前，一个简单的方法就是计算同类簇中对象的均值作为新的簇中心。不过，这一方法不适用于不等长时间序列数据。这里我们以假设我们要计算图 5（a）中时间序列 s 和 r 的中心曲线为例，s 和 r 的生命周期不等长，无法采用类似 k-means 聚类算法中更新簇中心的方法直接求 s 和 r 的均值时间序列。针对这一情况，目前最常用的处理方法就是对较短的时间序列做补零处理，延长其生命周期，然后再计算二者的平均值时间序列。如图 5（b），中心曲线 1 是时间序列 s 和补零处理后与时间序列 r 的平均值时间序列。但这样的做法增加了人为干扰因素，破坏了原时间序列的**形状信息**。

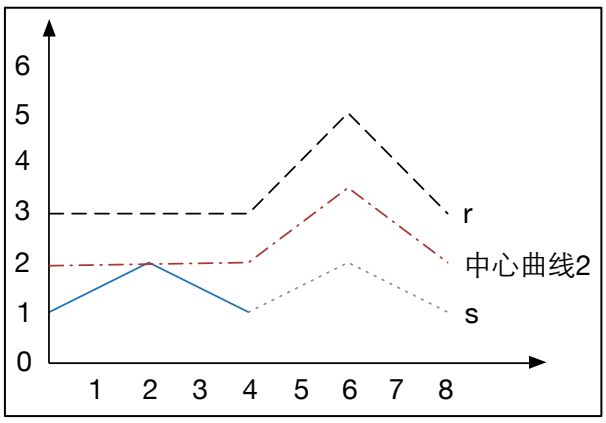
针对以上问题，我们提出的时间序列拟合计算方法，不增加任何数据，不改变任何时间序列原始生命周期。当拟合两条不等长时间序列 r 和 s 时，设 L_r 为时间序列 r 的长度，L_s 为时间序列 s 的长度，且 $L_r \geq L_s$。同样使用长度为 L_s 的滑窗提取时间序列 r 中所有长 L_s 的子序列，时间序列 r 上所有长 L_s 的子序列集合记为 $S_r^{L_s} = \{S_p^{L_s} \, of \, r, for \, 1 \leq p \leq L_r - L_s + 1\}$。其中 p 表示滑窗在时间序列 r 上的起始位置。取集合中到 s 的 STS 距离最小的子序列记为 $S_{\acute{p}}^{L_s}$，$\acute{p}$ 为该子序列的起始位置，$S_{\acute{p}}^{L_s}$ 是与 s 生命周期长度相等。计算 $S_{\acute{p}}^{L_s}$ 与 s 的均值时间序列记为 $\bar{s}$，并用 $\bar{s}$ 替换时间序列 r 中子序列 $S_{\acute{p}}^{L_s}$，其他子段保持不变，得到新的时间序列 $\acute{r}$，将 $\acute{r}$ 作为时间序列 r 和 s 的拟合时间序列。



（a） 不等长时间序列 r 和 s



（b） 补零补齐的中心曲线



（c） 基于滑窗的拟合中心曲线

图 5 计算不等长时间序列的簇中心曲线

图 5（c）表示经新拟合中心曲线算法计算得到图 5（a）中两条时间序列的拟合时间序列。新拟合算法计算得到的中心曲线明显保留原时间序列的更多形状趋势信息，优于曲线图 5（b）结果，并无改变原有时间序列数据的形状信息。

另外，拟合多条时间序列时，为了减少一条时间序列对整体拟合结果的影响，降低噪声点干扰，本文在拟合时间序列时为每条时间序列设定了拟合权重值 w。且初始情况下，每条时间序列拟合权重值 w_i（i=1,…,n，n 为需拟合时间序列总数）等于 1。设时间序列 r 和时间序列 s 的拟合权重值分别为 w_r 和 w_s，则 r 和 s 的拟合时间序列的拟合权重值为 w_r 与 w_s 的和。计算上述 $S_{\acute{p}}^{L_s}$ 与 s 的均值时，需要考虑时间序列 r 和 s 的拟合权重，即 $\bar{s}_k = \frac{S_{\acute{p}}^{L_s}{}_k \times w_r + s_k \times w_s}{w_r + w_s}$，k=1,…,$L_s$。

计算中心曲线的具体算法如下：

算法 1. 中心曲线计算算法 updateCentre$(D, cls[|D|])$.

输入：D：时间序列数据集；$cls[|D|]$：时间序列簇类别标签.

输出：$CTS[k]$：k 个类簇分别对应的中心曲线.

1. $cluster[k] \leftarrow getCluster(D, cls[|D|])$ // 获取一簇时间序列
2. $CTS[k] \leftarrow cluster[k][1]$ // 簇中第一条时间序列作为初始中心曲线
3. $w[k] \leftarrow \{1\}$ // 初始化 k 个簇中心曲线的拟合权重
4. ***for*** $i \leftarrow 1$ ***to*** k
5. ***for*** $j \leftarrow 2$ *to* $|cluster[i]|$
6. $CTS[i] \leftarrow mergeTS(CTS[i], cluster[i][j], w[i])$ // 拟合时间序列和现中心曲线
7. $w[i] \leftarrow w[i] + 1$ // 更新拟合权重
8. ***end***
9. ***end***
10. ***return*** $CTS[k]$

4.3 聚类算法

通过上述簇中心曲线计算方法，我们可以使用 k-means 的聚类过程对不等长时间序列进行聚类。另外，算法需要指定聚类簇数 k 并且随机选取 k 个时间序列作为 k 个类簇的初始中心曲线。基于滑窗不等长时间序列 STS 距离的聚类算法描述如算法 2 所示。

算法 2. 基于滑窗不等长时间序列 STS 距离的聚类算法.

输入：D：时间序列数据集；k：聚类簇数.

输出：$cls[|D|]$：数据集 D 中每个时间序列的类簇标记.

1. $DIS[k][|D|] \leftarrow \{\}$ // 时间序列到 k 个中心曲线的距离矩阵
2. $CTS[k] \leftarrow random(k)$ // 中心曲线
3. ***for*** $i \leftarrow 1$ ***to*** n // 迭代 n 次
4. $DIS[k][|D|] \leftarrow computeDistance(D, CTS[k])$ // 计算所有时间序列到 k 个中心曲线的距离
5. $cls[|D|] \leftarrow min(DIS[k][|D|])$ // 选取最小距离簇标记当前时间序列
6. $CTS[k] \leftarrow updateCentre(D, cls[|D|])$ // 更新中心曲线
7. ***end***
8. ***return*** $cls[|D|]$

基于滑窗不等长时间序列 STS 距离的聚类算法是一个迭代算法，迭代的次数除了可以由用户人工指定外，还由描述聚类结果各簇内成员紧凑度的 F-Value 函数值[8]变化情况决定。F-Value 函数定义公式如下：

$$F = \sum_{k=1}^{K} \sum_{x_i \in C_k} d\left(x_i, u_k\right) \quad (5)$$

其中 C_k 代表聚类结果簇 k，u_k 是 C_k 簇的中心曲线，x_i 是 C_k 簇中的时间序列，$d\left(x_i, u_k\right)$ 表示 x_i 和 u_k 间的基于 STS 距离的不等长时间序列距离。当一次迭代后本次 F-Value 函数值和上一次迭代的 F-Value 函数值差值小于阀值时，则停止迭代，聚类过程结束。

5 实验比较与分析

5.1 实验设计

5.1.1 数据说明

实验使用两个数据集，具体如下。

- 实验数据集 1 由两部分组成，第一部分搜索对象的选择参考了权威机构中国互联网协会的“中国网站排名”网站。“中国网站排名”将网站按照国内访问流量进行排序，得到中国互联网访问流量前 100 名榜单。我们在实验中先对访问流量前 100 名的网站名单进行清理，除去其中恶意引流网站，将剩余的 78 个网站作为搜索对象，并收集其搜索指数序列数据加入数据集 1。第二部分搜索对象来自百度搜索风云榜中的热门搜索、热门流行词汇、热门网页游戏和热门歌曲共计 216 个。最后，我们以 78

个网站名称和百度热门词汇所为搜索对象，记录这些搜索对象对应的搜狗指数的时间序列数据。搜狗指数记录了搜索对象自 2012 年 10 月 1 日起至查询日期（数据收集时间为 2014 年 8 月 26 日），每日的搜索指数。不在搜索对象生命周期内的时间点对应的搜索指数为 0，所有对象搜索指数序列的初始长度都为 694。数据集 1 中的搜索对象生命周期较长，生命周期长度可以覆盖搜索指数序列的初始长度一半以上。

- 实验数据集 2 包含实验数据集 1 中所有搜索对象，此外还包括搜狗热搜榜中的每日热搜词汇榜。搜狗每日热搜词汇榜记录了其搜索引擎上每日新产生最热门的 10 个搜索词汇（如“世界杯四强出炉”），这些搜索对象的生命周期较短。实验数据集 2 共包含 1040 个搜索对象。因此，相比于实验数据集 1，实验数据集 2 具有更加分散的数据特征。

5.1.2 数据预处理

实验中数据的预处理包括截取搜索指数序列生命周期和标准分数预处理两个过程。

从搜狗指数采集到的所有搜索指数序列的初始长度都为 694，但其中大部分搜索对象的生命周期长度小于 694。如在图 1 中，对“埃博拉出血热”的搜索出现在 7 月 27 日后，“埃博拉出血热”的生命周期小于 694，“埃博拉出血热”的搜索指数序列在其生命周期前存在大量 0 值。这些 0 值会影响我们对时间序列形状信息的判断。本文设计程序找到搜索指数序列上第一个和最后一个对应非 0 搜索指数的点，截取两个时间点中间的时间序列，得到搜索对象完整生命周期的搜索指数序列。程序同时排除搜索指数序列生命周期太短（实验限定搜索对象的生命周期大于 50）的搜索对象。

实验还通过标准分数的方法对搜索指数序列进行预处理，将所有观测数值数量级差异明显的搜索指数序列放在等距量表中度量。

5.2 实验结果与分析

实验首先通过计算数据集 1 和 2 在聚类簇数 K 取不同值时 Average Silhouette 函数的值，并以此为依据确定最佳的聚类簇数。其次，实验比较基于滑窗不等长时间序列 STS 距离的聚类算法与通过补零取得等长时间序列的 k-means 聚类算法，两个算法的聚类正确率和聚类质量。

5.2.1 选择最佳聚类簇数

本文提出的基于滑窗不等长时间序列 STS 距离的聚类算法和 k-means 聚类算法相似，针对不同数据集，都需要在聚类开始前指定聚类结果的类簇数 K。通常选择聚类簇数的方法比较主观，本文通过计算类簇数 K 取不同值时的 Average Silhouette 函数值，确定两个数据集最佳的聚类簇数 K。

Average Silhouette 函数被提出用于度量聚类质量，并且可以用于选择聚类簇数[13]。Average Silhouette 函数的值等于所有聚类对象 Silhouette 函数值的平均值，Silhouette 函数值公式定义如下：

$$s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \quad (5)$$

对于聚类对象 i，$a(i)$ 表示聚类结果中，聚类对象 i 与所在簇所有其他聚类对象距离的平均值。$b(i)$ 表示聚类结果中，聚类对象 i 与除所在簇外且距离其最近的簇中所有聚类对象距离的平均值。根据公式可知 $-1\le s(i)\le 1$，$s(i)$ 的值越大，说明聚类对象 i 的分配结果越好。Average Silhouette 函数计算所有对象的 Silhouette 函数值的平均值，反映了分配聚类对象的整体效果，即聚类质量。

同一聚类算法，选择不同的聚类簇数对应的 Average Silhouette 函数值通常不同。图 6 记录了在类簇数 K 取不同值时，数据集 1 和数据集 2 使用基于滑窗不等长时间序列 STS 距离的聚类算法的 Average Silhouette 函数值变化情况。

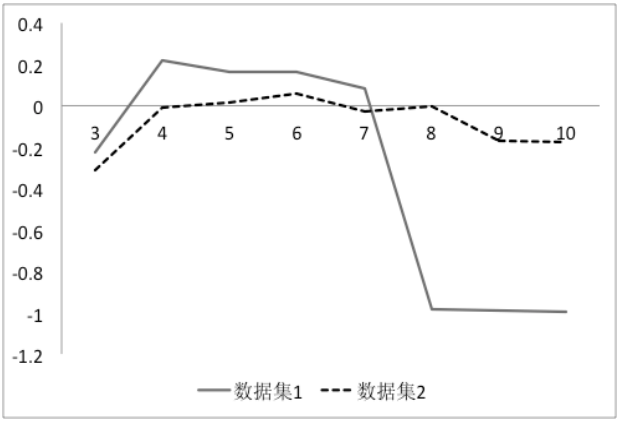
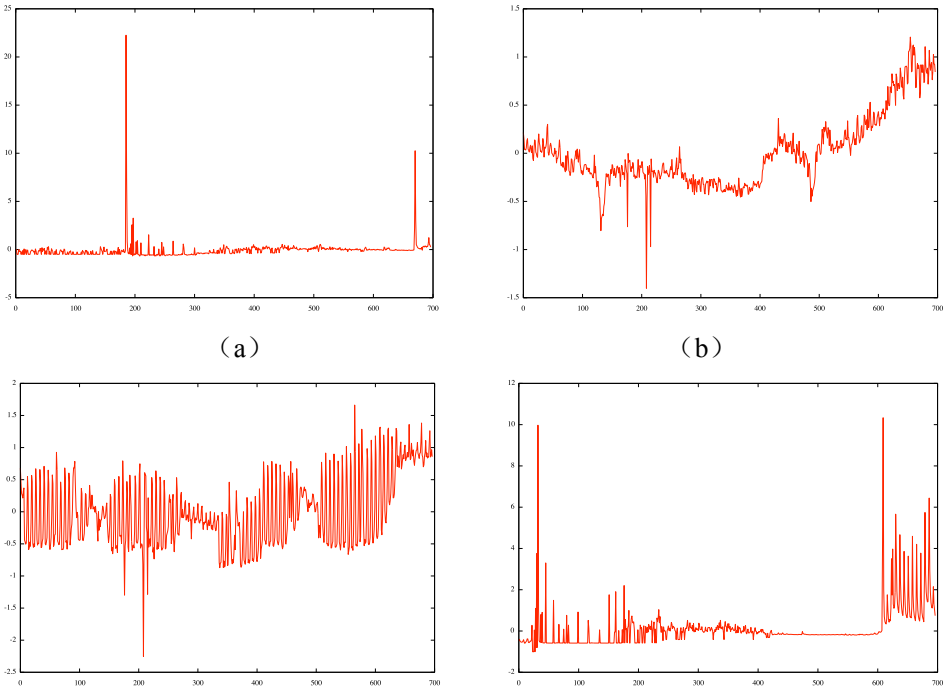


图 6 选择不同聚类簇数时数据集 1 和数据集 2 对应的 AverageSilhouette 值

根据图 6 中信息，实验数据集 1 在聚类簇数 K 取 4 到 7 之间时聚类效果较好。 聚类簇数 K 为 4 时 Average Silhouette 函数值最大，聚类效果最好。实验数据集 2 在聚类簇数 K 取 4 到 8 之间时聚类效

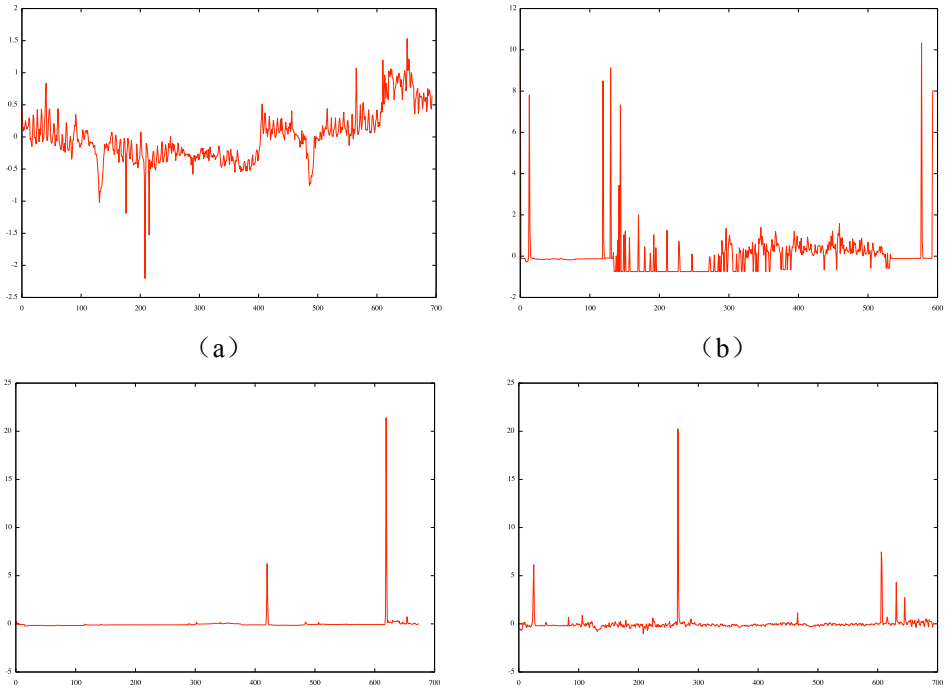
果较好。聚类簇数 K 为 6 时 Average Silhouette 函数值最大，聚类效果最好。该结果与文献[13]中以 k=6 的经验结果一致，这侧面验证了图 6 结果的正确性。图 7 和图 8 给出了实验数据集 1 和 2 使用基于滑窗不等长时间序列 STS 距离聚类算法分别在在簇数 K 为 4 和 6 时，算法迭代 20 次得到的聚类结果各簇的中心曲线图。图中横坐标表示时间序列的长度，纵坐标单位是搜索指数数据经标准分数预处理后的等距量表单位。通过图 7 和图 8 的拟合中线曲线，我们将将进一步完成最终的数据聚类，其结果将在下文进一步报告聚类结果的精度和质量。



（a） （b）

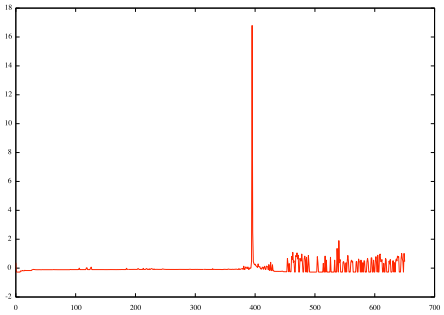
（c） （d）

图 7　数据集 1 聚类结果中心曲线图

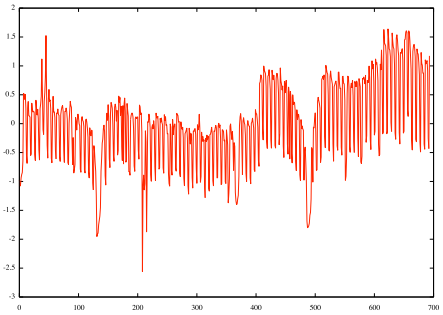


（a） （b）

（c） （d）



（e）



（f）

图 8　数据集 2 聚类结果中心曲线图

5.2.2　聚类正确率分析

综述[16]将评价聚类结果优劣的方法分为两类：（1）度量正确率；（2）度量聚类总体质量。目前最常见的聚类正确率定义如下：

$$r=\frac{\sum_{i=1}^{k}a_i}{n} \quad (6)$$

其中，a_i 是出现在第 i 个类簇（执行算法得到）及其对应的类（初始类）中的样本数，n 表示数据集中样本总数，k 表示指定的类簇数。

目前聚类正确率的计算方法要求已知全部聚类对象的类型，本文将这一类正确率度量的方法称为**聚类全局正确率**。在实际情况下，真实的聚类结果往往不易获知，难以将聚类结果类簇和聚类对象初始类一一对应。针对这一情况，本文提出更为可行的用聚类局部正确率来度量聚类正确率。

假设在聚类前，我们已知部分聚类对象归属同一类，我们可以用这些对象被聚类到同一个类簇的最大比例来衡量聚类结果的好坏。公式如下：

$$r=\frac{\max_i^k\left(a_i\right)}{n_{known}} \quad (7)$$

其中，a_i 表示已知对象在同一类簇 i 中的个数，n_{known} 表示已知属于同一初始类对象的总数。$r\le 1$ 且 r 越接近 1 聚类效果越好。

相对已知全部聚类对象的类型，获取属于同一类型的部分聚类对象更加可行。实验对互联网搜索对象按照搜索量变化的趋势进行聚类。因为大多数互联网搜索对象的搜索量变化情况除了由自身性质决定外，更多受到如突发事件等外部不确定因素影响，所以不易直观区分它们的类型。但互联网热门网站的搜索量较大且总能保持在较稳定的范围内，极少受突发事件影响，互联网热门网站搜索量变化的趋势相似，按照搜索量变化的趋势分类，这类对象应该被划分在同一类中。

实验采集互联网热门网站对象（中国互联网访问流量前 100 名网站）的搜索指数序列作为参考数据集，计算它们被聚类到同一个类簇的最大比例值作为局部聚类正确率。实验设定数据集 1 的类簇数 K 为 4，使用基于滑窗不等长时间序列 STS 距离的聚类算法和通过补零取得等长时间序列的 k-means 聚类算法（分别使用 STS 距离和欧氏距离）的局部聚类正确率如表 2。

表 2　数据集 1 使用 3 种时间序列聚类算法的局部正确率

基于 STS 距离不等长聚类算法	补齐时间序列长度的欧式距离 k-means 聚类算法	补齐时间序列长度的 STS 距离 k-means 聚类算法
0.962	0.423	0.513

基于滑窗不等长时间序列 STS 距离的聚类算法将大多数热门网站对象划分到一簇，形成图 7 中第四条中心曲线。热门网站的搜索指数因为极少受突发事件影响且每日被搜索的总量比较稳定，所以其时间序列图通常没有尖锐的峰值，所有数值点都在平均值上下波动，并总体呈现上升趋势。图 7 中（b）中心曲线也符合上述规律。

综上，基于滑窗不等长时间序列 STS 距离的聚类算法较通过补零取得等长时间序列的 k-means 聚类算法聚类正确率更高。

5.2.3　聚类总体质量分析

聚类的总体质量描述了聚类算法给各类簇划分的成员是否合理，可以由聚类结果类间差异和类内差异度量，通常包括两个指标[4, 8, 16]。

（1）F-Value（F 值）：F-Value 的计算方法如公式（5）。F-Value 刻画了每个类内部成员的紧凑

度，F-Value 越小表示聚类效果越好。

（2）D-Value（D 值）：$D=\sum d(u_i,u_j)$，其中 u_i 是类 i 的中心曲线，u_j 是类 j 的中心曲线。D-Value 代表类与类之间的差异性，D-Value 越大聚类效果越好。

表 3 给出了数据集 1（类簇数 K 为 4）和数据集 2（类簇数 K 为 6）使用基于滑窗不等长时间序列 STS 距离的聚类算法和通过补零取得等长时间序列的 k-means 聚类算法（分别使用 STS 距离和欧氏距离）的聚类总体质量。

表 3 三种时间序列聚类算法的聚类总体质量

	数据集 1		数据集 2	
聚类算法	F 值	D 值	F 值	D 值
基于 STS 不等长	47.67	22.19	35.62	49.32
k-means（欧式距离）	185.56	10.19	774.36	38.05
k-means（STS 距离）	93.39	7.49	1056.26	72.57

在数据集 1 和数据集 2 上，基于滑窗不等长时间序列 STS 距离的聚类算法都较通过补零取得等长时间序列的 k-means 聚类算法（分别使用 STS 距离和欧氏距离）聚类总体质量更高，聚类效果更好。

6 结论与未来工作

分析互联网中的搜索对象或社交网络中话题随时间变化的规律，预测其发展趋势极具研究意义，而聚类搜索对象或社交网络中话题的时间序列则为解决该问题提供了一种有效手段。针对时间序列聚类中时间序列不等长和时间序列观测值数量级差异两个问题，本文首先采用标准分数对时间序列观测数值进行预处理，并在 STS 距离的基础上提出新的不等长时间序列距离和基于滑窗不等长时间序列 STS 距离的聚类算法。

本文通过实验首先确定两个实验数据集适合的聚类簇数 K，而后分别比较了使用基于滑窗不等长时间序列 STS 距离的聚类算法法和通过补零取得等长时间序列的 k-means 聚类算法的聚类正确率和总体质量。实验证明了基于滑窗不等长时间序列 STS 距离的聚类算法在解决了时间序列聚类中时间序列不等长和时间序列观测值数量级差异两个问题的同时聚类效果较好。

本文主要从聚类正确率和总体质量上验证了我们提出的基于滑窗不等长时间序列 STS 距离的聚类算法的效果。我们注意到现实生活中存在海量的搜索对象及其对应的时间序列大数据，如何设计高效率的不等长时间序列聚类算法，提高对应聚类算法的计算性能，将是我们未来工作之一。此外，如何设计针对实时不等长时间序列数据的在线聚类算法也是我们积极考虑的下一步工作。因此，我们未来工作将在计算效率和质量精度这两个层面开展更加深入的不等长时间序列聚类算法研究工作。

参 考 文 献

1. 谢婧, *中文微博的话题检测及微博预警*, 2013, 上海交通大学.
2. 姚海波, *微博热点话题检测与趋势预测研究*, 2013, 华南理工大学.
3. Ginsberg, J., et al., *Detecting influenza epidemics using search engine query data.* Nature, 2008. **457**(7232): p. 1012-1014.
4. 韩忠明, 陈妮, 乐嘉锦, 段大高, 孙践知, *面向热点话题时间序列的有效聚类算法研究.* 计算机学报, 2012(11): p. 2337-2347.
5. Nikolov, S., *Trend or no trend: a novel nonparametric method for classifying time series*, 2012, Massachusetts Institute of Technology.
6. 陈湘涛, 李明亮, 陈玉娟, *基于时间序列相似性聚类的应用研究综述.* 计算机工程与设计, 2010(03): p. 577-581.

7. Möller-Levet, C.S., et al., *Fuzzy clustering of short time-series and unevenly distributed sampling points*, in *Advances in Intelligent Data Analysis V*. 2003, Springer. p. 330-340.
8. Yang, J. and J. Leskovec. *Patterns of temporal variation in online media*. in *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011. ACM.
9. Warren Liao, T., *Clustering of time series data—a survey*. Pattern recognition, 2005. **38**(11): p. 1857-1874.
10. Liao, T., et al. *Understanding and projecting the battle state*. in *23rd Army Science Conference, Orlando, FL*. 2002.
11. Roy, D., et al., *Prototyping a global algorithm for systematic fire-affected area mapping using MODIS time series data.* Remote sensing of environment, 2005. **97**(2): p. 137-162.
12. Bollen, J., H. Mao, and A. Pepe. *Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena*. in *ICWSM*. 2011.
13. Rousseeuw, P.J., *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.* Journal of computational and applied mathematics, 1987. **20**: p. 53-65.
14. Ye, L. and E. Keogh. *Time series shapelets: a new primitive for data mining*. in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009. ACM.
15. Mueen, J.Z.A. and E. Keogh, *Clustering time series using unsupervised-shapelets*. 2012.
16. 孙吉贵, 刘杰, 赵连宇, *聚类算法研究*. 软件学报, 2008(01): p. 48-61.