一种基于逐层扫描的频繁字串快速提取算法

张宇萌[1,2]　刘传汉[1]

(上海交通大学计算机科学与工程系　上海 200030)[1]　(宁波大学商学院信息管理系　宁波 315211)[2]

摘　要　串频统计是一种简便有效的抽取未登录词方法。本文提出了一种快速的频繁字串提取和计频方法，通过逐层扫描快速发现频繁字串，修正字串有效出现频次，最后抽取平均互信息量达到阈值的字串。实验结果显示该方法有效可行。

关键词　频繁字串，中文抽词，逐层扫描，互信息

An Algorithm of Fast Frequent String Extracting Based on Level-wise Scan

ZHANG Yu-meng[1,2]　LIU Chuan-han[1]

(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China)[1]

(Department of Information Management, Ningbo University, Ningbo 315211, China)[2]

Abstract　String frequency statistics is a simple and effective method of extraction unlisted word. This paper presents an effective algorithm of extracting frequent strings. It uses a level-wise scan for finding rapidly frequent strings and modifies the valid frequency that string appears in text. Finally, those high-frequent strings that reach the threshold of average mutual information are extracted. Experimental results show that the method is effective and feasible.

Keywords　Frequent string, Chinese automatic word extraction, Level-wise scan, Mutual information

1　引言

未登录词识别是自然语言处理领域的重要课题之一。汉语中未登录词的种类很多、构词规律各异、数量众多，而且还在不断的更新扩大，即使是最大的词典不会也不可能囊括自然语言的所有词汇[1]。

为了自动识别出未登录词，学者们提出了基于规则和统计的方法实现未登录词抽取[2-4]，后者是当前研究的主流，如文献[5]在分词的基础上进行频度统计得到大量的候选新词，文献[6]利用 $\chi 2$ 统计量方法和广义似然比方法计算字之间的相关度，从语料库中获得未登录词，用于自动分词或自动编制词典，文献[7]从符号串的内部结合紧密程度判断是否成词或短语。

研究表明，串频统计[8-11]是一种简便有效的识别未登录词方法。中文中，句子是连续的字符串，中间无明显分隔符号，如果某个连续的字符串在文本中出现的频率(简称串频)越高，则构成"词"的可能性越大[12]。采用串频统计进行未登录词抽取的过程一般可分为三步[13]：(1) 使用各种分隔标记对文本进行文本预切割，分割成若干个短句；(2) 从分割好的短句集合中找出高频字串；(3) 对高频字符串进行统计加权，判断高频字串是否为有意义的"词"。步骤(2)是最为耗时的，它决定着未登录词抽取的整体性能。文献[8,9]的方法基本相同，后者在算法上进行了改进，使得运算速度有所提高。为进一步提高运算速度，文献[13]提出了位置记忆跳跃匹配算法，虽然其算法处理速度大幅度提高，但存在部分重复字串遗漏的问题，未登录词抽取质量不尽人意。除了运算速度较慢外，"串频统计"法难以滤除共现频度高的非词语常用搭配[14]，如"有个"、"这一"等。

在前人研究基础上，本文提出了一种快速的串频统计算法，称为逐层扫描算法。较以往方法，该算法在处理速度和抽词质量上有了较大的提高：(1)本文算法吸取了数据挖掘中 Apriori 关联规则挖掘的思想，利用 Apriori 性质尽可能地压缩搜索空间，大幅度提高高频字符串提取速度。(2)提出有效支持数的统计方式，能较好地过滤高频非"词"字串，提高了抽词的准确度。

2　基于逐层扫描的高频字串提取算法

2.1　基本定义和定理

定义 1(短句)　经过文本预处理后，分隔标记之间的连续字符串。

定义 2(字串 $x_i=\{w_1w_2\cdots w_n\}$)　由短句中若干个连续字符组成的字符串，$w_i(1\leqslant i\leqslant n)$表示一个字符，$n$ 表示字串长度。

定义 3(支持数 $\sigma(x_i)$)　字串 x_i 在短句集合中出现的次数。

定义 4(支持度 $S(x_i)$)　字串 x_i 在短句集合中出现的频率，即 $S(x_i)=\frac{\sigma(x_i)}{A}\times 100\%$，$A$ 表示候选字串总数。

定义 5(频繁字串)　在短句集合中出现的频率大于事先确定的阈值(即最小支持数)的字串，也称作高频字串。

定义 6(子串)　如果 q 字串包含 p 字串，则 p 是 q 的子串，或 p 是 q 的子集。

定义 7(父串)　如果 q 字串包含 p 字串，则 q 是 p 的父串，或 q 是 p 的超集。

定义 8(k-字串($k\geqslant 1$))　也称 k 元字串，即包含 k 个字符的字串。

Apriori 算法是一种很有影响的挖掘布尔关联规则频繁项集的算法，该算法利用频繁项集性质的先验知识，使用一种称作逐层搜索的迭代方法，利用已获得的 k-项集确定($k+1$)-项

张宇萌　副教授，主要研究方向为文本自动分类；刘传汉　博士生，主要研究方向为文本自动摘要。

集。Apriori 有一个重要性质：频繁项集的所有非空子集都必须也是频繁的。这个性质用于压缩搜索空间，可大幅度提高频繁项集逐层产生的效率。该性质也适合频繁字串的快速提取，稍作变化，可以推导出与 Apriori 性质相似的定理 1。

定理 1 如果子字串是非频繁的，其父字串也是非频繁的(或，频繁字串的所有非空子串都必须也是频繁的)。

证明：假设字串 x_1 是字串 x_2 的子串，即 $x_1 \subset x_2 \Rightarrow x_2 = x_1 \cup x'$；如果 x_1 不是频繁的，则 x_1 不满足最小支持度阈值 min_sup，即 $S(x_1)=\frac{\sigma(x_1)}{N} \leqslant$ min_sup。x_2 是 x_1 的父串，是在 x_1 字串的基础上添加了子串 x'，不可能比 x_1 更频繁出现，即 $\sigma(x_2)=\sigma(x_1 \cup x') \leqslant \sigma(x_1) \Rightarrow S(x_2)=\frac{\sigma(x_2)}{N} \leqslant S(x_1) \leqslant$ min_sup。因此，x_2 也不是频繁的。

定理 2 字串被切割成的子字串越短，则生成的候选字串总数就越少。

证明：设字串 X 的长度为 N，被分割成 m 个子字串 $\{x_1, x_2, \cdots, x_m\}$，子字串 $x_i(1 \leqslant i \leqslant m)$ 的长度为 l_i，其中 $\sum_{i=1}^{m} l_i = N$。设 X 生成 k-候选字串个数为 s_1，其子字串集 $\{x_1, x_2, \cdots, x_m\}$ 生成 k-候选字串总数为 s_2，并且假定 $l_i \geqslant k$。

$$s_1 = N-k+1$$

$$s_2 = \sum_{i=1}^{m}(l_i-k+1) = \sum_{i=1}^{m} l_i - m(k-1) = N-k+1-(m-1)(k-1)$$

则

$$\frac{s_2}{s_1} = \frac{N-k+1-(m-1)(k-1)}{N-k+1} = 1-\frac{(m-1)(k-1)}{N-k+1} \tag{1}$$

由式(1)可知，当字串 X 没有被分割时 $m=1$，这时 $s_1 = s_2$。字串 X 被分割后，$m \geqslant 2$，则 $s_2 \leqslant s_1$，并且分割次数越多(m 值)越大，$\frac{s_2}{s_1}$ 就越小。

2.2 文本预处理

由定理 2 可知，要提高统计速度就应该尽可能把文本切割成较短的字串。虽然中文词与词之间没有明显的分隔符号，但有两类分隔符号我们可以利用，它们包括绝对分隔符与条件分隔符。绝对分隔符指标点符号、数字、字母等；条件分隔符指构词能力很差的单汉字、数词和虚词等，如“的”，“很”，“了”，“关于”，“对于”…，它们很难与其他汉字组成短语和词。这样文本就被这些分隔符切割成由短句组成的集合，该集合记作 S。

2.3 基于逐层扫描的频繁字串发现算法

传统的 n 元组提取方式会产生数量巨大的候选元组[15]，如长度为 10 个字符的字串将产生 45 个候选字串。统计所有候选字串出现频次是非常耗时的工作，难以满足海量信息处理。本文提出的基于逐层扫描的频繁字串发现算法可以极大压缩候选字串的产生数量。

为提高频繁字串产生的效率，我们利用定理 1 压缩搜索空间：首先从短句集 S 中找到频繁 1-字串集合，该集合记作 L_1。利用已知的 L_1 去找频繁 2-字串的集合 L_2，利用 L_2 去找 L_3，如此下去，直到不能找到频繁 k-字串集为止。算法由候选字串生成步和剪枝步组成：

(1) 候选字串生成步：为找 L_k，从短句集 S 中提取长度为 k 的字串(k -字串)集合，该集合记作候选集 C_K。根据 L_{k-1} 中子字串的有无，C_K 收集可能频繁的字串。如，字串 $x_i = \{w_i w_{i+1} \cdots w_{i+k-2}\}$ 不在 L_{k-1} 中，则根据定理 1，其父串 $y_i = \{w_i w_{i+1} \cdots w_{i+k-1}\} = x_i \cup w_{i+k-1}$ 也不可能是频繁的，从而大大压缩查找空间。

(2)剪枝步：C_K 是 L_k 的超集，它的成员可能是频繁的也可能不是频繁的，但所有的频繁 k-字串集都包含在 C_k 中。扫描短句集 S，确定 C_K 中每个候选字串的重复次数，从而确定 L_k。C_K 可能很大，这样所涉及的计算量就很大。由定理 1 可知，任何非频繁($k-1$)-字串的父字串 k-字串不可能是频繁的。因此，如果一个候选 k-字串的($k-1$)-子集不在 L_{k-1} 中，则该候选字串也不可能是频繁的，不必去扫描整个短句集计数就被删除了。因此，利用已获知的先验知识可以极大地压缩搜索空间，提高查找速度。

例如，假设待处理的文本为：“构建社会主义和谐社会，社区的和谐，家庭的和谐，是和谐社会的重要组成部分”。

经过预处理，文本被切割成短句集合 S={“构建社会主义和谐社会”，“社区”，“和谐”，“家庭”，“和谐”，“是和谐社会”，“重要组成部分”}，其中 $|S|$ 表示所有短句字符总数，这里 $|S|=29$。

(1) 在算法的第一次迭代，每个字符都是候选 1-字串集合 C_1 的成员，$C_1=\{w_1, w_2, \cdots, w_i\}$，累计它们出现的次数；

(2) 假定最小支持数为 2。可以确定频繁 1-字串集合 L_1 ={{和}，{谐}，{社}，{会}}，它是由具有最小支持度的候选 1-字串集组成。

(3)为发现频繁 2-字串的集合 L_2，算法产生候选 2-字串集合 C_2。从短句集 S 中依次截取长度为 1 的字串 $x_i=\{w_i\}$，如果 $x_i \in L_1$，其父字串 $y_i=\{w_i w_{i+1}\}$ 则可能是频繁的，收录到 C_2 中。最后，C_2={社会，和谐，谐社}。

(4)扫描短句集 S，计算 C_2 中每个候选字串的支持计数。

确定频繁 2-字串集合 L_2={和谐，谐社，社会}，它是由具有最小支持度的候选 2-字串集组成。

(5)重复(3)-(5)步，最终得到 3-字串集和 4-字串集。表 1 给出长度 2 以上、具有最小支持数的字串。

表 1 统计出来的频繁字串

字串	和谐	谐社	社会	和谐社	谐社会	和谐社会
支持数	4	2	3	2	2	2

2.4 有效支持数的修正

研究发现，算法得到一些频繁(高频)字串，但有些字符串是无意义的字串。这些字串虽然具有最小支持数，但并不是其单独出现的真实频次，而是其父串分裂而得的子串，如“谐社”是“和谐社”、“谐社会”的子串；“和谐社”、“谐社会”是“和谐社会”的子串。

定义 8(有效支持数) 字串的有效支持数等于其出现频次减去其最频繁父串的频次：

$$\mathrm{Valid}(x_i)=\mathrm{Fre}(x_i)-\mathrm{Max}\{\mathrm{Fre}(\mathrm{sup}(x_i))\} \tag{2}$$

例如，“和谐”出现 4 次，它的父串“和谐社”、“和谐社会”出现次数都是 2 次，因此它的有效支持数为(4－2)＝2；同理，“谐社”，“社会”的有效支持数分别为 0 次和 1 次。因此，算法输出有效支持数大于阈值的字串(“和谐”，“和谐社会”)，而把“谐社”、“社会”等字串删除，如表 2 所示。

表 2 有效支持数修正后的频繁字串

字串	和谐	谐社	社会	和谐社	谐社会	和谐社会
有效支持数	2	0	1	0	0	2

2.5 基于互信息统计进行字串过滤

经过有效支持数修正后，仍然还有一些由高频字组成的

随机字串，如“中兴”，“也不”。这些字串出现的频次很高，仅用计频方式很难判定它们是否成“词”，需要进一步筛选。互信息(MI)是一种常用的统计模型，是对两个随机变量 X 和 Y 之间统计相关程度的一种度量。字串间互信息反映了两字符串之间结合的紧密程度。MI 的值越大，字串成为词或短语的可能性就越大。字串 $x_i=\{w_1w_2\cdots w_n\}$ 可分割成子字串 $x_{j-}=\{w_1w_2\cdots w_j\}$ 和 $x_{j+}=\{w_{j+1}w_{j+2}\cdots w_n\}$，其中 $1\leqslant j<n$，x_i 的互信息如公式(3)所示。

$$MI(x_i)=\log\frac{P(x_i)}{P(x_{j-})P(x_{j+})}=\log\frac{A\times N(x_i)}{N(x_{j-})\times N(x_{j+})} \tag{3}$$

其中，$P(x_i)$，$P(x_{j-})$和 $P(x_{j+})$分别是 x_i，x_{j-}和 x_{j+}中在候选字串集中出现的概率，$N(x_i)$，$N(x_{j-})$和 $N(x_{j+})$表示 x_i，x_{j-}和 x_{j+}的有效支持数，A 表示候选字串集字串总数。

由于抽取的字串包括二元字串和多元字串，文中采用平均互信息分别对多字串进行度量

$$MI(x_i)=\frac{1}{n-1}\sum_{j=1}^{n-1}MI(x_{j-},x_{j+})1\leqslant j<n \tag{4}$$

3 实验及算法分析

在 T2050，1G 内存，Windows XP 平台上，实验随机选取长度不等的文本，应用本文算法和传统的 n 元组方法分别做了频繁字串的抽取实验，并在二方面做了比较：(1)频繁字串抽取效率，(2)频繁字串抽取数量和质量。

3.1 频繁字串抽取效率比较

在频繁 k-字串基础上产生(k+1)-候选字串，循环往复，直至无候选字串产生。实验结果如表 3 所示。

表 3 长度不同的候选字串数量比较

id	len	Sen	S1	NS1	S2	NS2	S3	NS3	S4	NS4	S5	NS5	S6	NS6	S7	NS7	S8	NS8	S9	NS9	S10	NS10
1	348	29	136	136	33	33	15	27	12	21	6	18	2	15	1	12	0	9	0	6	0	5
6	593	66	194	194	77	77	20	60	5	36	0	22	0	12	0	6	0	3	0	2	0	1
3	810	82	225	225	92	92	37	86	9	57	4	40	0	24	0	11	0	6	0	2	0	0
7	911	94	265	265	134	134	55	134	11	101	6	70	4	48	2	33	0	23	0	13	0	7
2	1208	117	297	297	119	119	50	108	21	72	10	42	7	29	6	17	5	8	3	6	0	3
8	1574	142	475	475	149	149	39	112	13	71	4	44	0	22	0	9	0	6	0	4	0	3
9	2164	246	576	576	295	295	60	218	10	124	3	64	2	31	0	17	0	11	0	8	0	6
4	2633	319	490	490	440	440	134	430	47	331	32	261	9	191	6	137	3	98	2	72	0	46
5	3953	457	770	770	720	720	178	638	42	460	28	329	9	226	2	160	0	111	0	75	0	52
10	5352	703	801	801	1191	1191	256	1098	62	778	36	539	26	367	6	239	4	154	3	97	2	60

其中字段 id、len 分别表示文本编号和文本长度；sen 表示经过文本预处理后，获的短句数量；$S1$、$S2$…$S10$ 表示本文算法统计过程中得到的 1-，2-，…，10-候选字串的数量；$NS1$、$NS2$…$NS10$ 表示传统算法在统计过程中得到的 1-，2-，…，10-候选字串的数量。$S1$ 与 $NS1$ 为单字符字串的个数，这些字符在文本中出现的频次都超过了实验所设置的阈值，由它们组成的 2-字串都作为候选字串，因此，$S2$ 与 $NS2$ 数量相等。2-候选集中已经有许多字串的支持数小于阈值，它们的父串(3-字串)不能被选入 3-候选集，因此，$S3$ 小于 $NS3$。从表 3 上可以发现：随着字串长度的增加，候选字串的数量在急剧减少，使得统计运算量大大降低。候选字串生成总数比较如图 1 所示。图中横坐标为文本长度，纵坐标为算法生成的所有候选字串数量。

在算法的时间复杂性方面，该算法主要时间花在对字串进行排序，如对 k-频繁字串集所有字串快速排序的时间小于 $f(k)\log(f(k))$，其中 $f(k)$为 k-频繁集包含的字串个数，因此整个算法的时间复杂度为 $O(\sum_{k=2}f(k)\log(f(k)))$。从表 3 可以发现，长度 3 以上的频繁字串个数迅速减少，所以该算法的时间主要消耗在对 2-频繁字串的排序上。

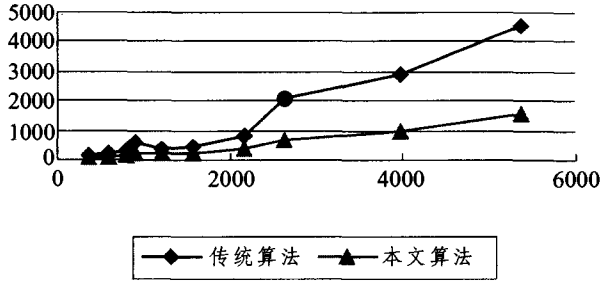

图 1 候选字串生成数量比较图

3.2 重复字串抽取数量和质量比较

以上文的样本为测试对象，算法对最后得到的频繁字串进行有效支持数修正，并计算每个频繁字串的平均互信息，选取平均互信息较大的字串。实验结果如表 4 所示。

表 4 部分样本的实验结果

文本编号	文本长度	频繁词(个)	正确词(个)	可接受率
1	348	9	9	100%
6	593	15	14	93.3%
3	810	16	15	93.8%
7	911	16	15	93.8%
2	1208	17	16	94.1%
8	1574	17	16	94.1%
9	2164	22	21	95.5%
4	2633	25	24	96.0%
5	3953	43	40	93.0%
10	5352	64	61	95.3%

为进一步验证该算法的有效性，我们同文献[9,10]中的算法进行了比较，也采用小说《空中小姐》作为测试数据，取最小平均互信息为 0.8，实验结果如表 5 所示。

表 5 测试结果比较

抽取方法	频繁词	正确词	可接受率
文献[10]中的方法	35	34	97.14%
文献[9]中的方法	45	44	97.78%
本文的方法	73	72	98.63%

抽词结果如下：

阿眉 什么 自己 告诉 知道 女孩 北京 怎么 薛苹 因为 张欣 生活 机场 朋友 时候 小沈 工作 姑娘 眼睛 关义 海军 水

兵 杭州 开始 十分 喜欢 应该 英雄 乘务 东西 码头 空中小姐 高兴 明白 乘务队 希望 几乎 简直 民航 情况 确实 已经 变化 城市 吃饭 而且 九溪 军舰 全部 王眉 影响 报纸 充满 房间 关系 航班 建设 精神 蓝色 泪水 妈妈 重新 乘务员 第二天 开玩笑 老百姓 一架飞机 容易 虽然 写信 信任 衣服 意思

上面的结果中，带下划线的字串是小说中的人名，带波浪线的是地名，带方框的字串"一架飞机"是错误的。与以往的方法相比，本文方法抽取的频繁字串数量和质量都有一定的提高。

结束语 本文吸取了数据挖掘中 Apriori 关联规则挖掘的思想，提出了逐层扫描算法和有效支持数统计方式。与以往算法相比，该算法具有更快的速度和更高的准确性，可广泛应用于自动标引、分类、信息检索、摘要等各种信息处理领域。

参考文献

[1] 王 永成. 中文信息处理技术及其基础[M]. 上海：上海交通大学出版社，1991

[2] Tan Hong-ye. Research on Method of Automatic Recognition of Chinese Place Name based on Transformation[J]. Journal of Software, 2001, 12(11): 1608-1613

[3] Nie Jian-Yun. Unknown Word Detection and Segmentation of Chinese using Statistical and Heuristic Knowledge. Communications of COLIPS, 5(1&2): 47-57

[4] Ling G C, Asahara M, Matsumoto Y. Chinese Unknown Word Identification Using Character-based Tagging and Chunking. In Companion Volume to the Proceedings of ACL 2003, Interactive. Poster/Demo Sessions: 197-200

[5] 崔世起，刘群，孟遥，等. 基于大规模语料库的新词检测[J]. 计算机研究与发展，2006，43(5)：927-932

[6] 黄萱菁，吴立德，王文欣，等. 基于机器学习的无需人工编制词典的切词系统[J]. 模式识别与人工智能，1996，9(4)：297-303

[7] 罗盛芬，孙茂松. 基于字串内部结合紧密度的汉语自动抽词实验研究[J]. 中文信息学报，2003，17(3)：9-14

[8] 刘挺，吴岩，王开铸. 串频统计和词形匹配相结合的汉语自动分词系统[J]. 中文信息学报，1998，12(01)：17-25

[9] 任禾，曾隽芳. 一种基于信息熵的中文高频词抽取算法[J]. 中文信息学报，2006，20(5)：40-90

[10] 韩客松，王永成，陈桂林. 无词典高频字串快速提取和统计算法研究[J]. 中文信息学报，2001，15(02)：23-30

[11] 姜韶华，党延忠. 基于长度递减与串频统计的文本切分算法[J]. 情报学报，2006，25(1)：74-79

[12] 金翔宇，孙正兴，张福炎. 一种非受限中文文档自动抽词方法[J]. 中文信息学报，2001，15(6)：33-39

[13] 马颖华，王永成，苏贵洋. 一种在汉语文本中抽取重复字串的快速算法[J]. 电子学报，2002，12(12)：2177-2179

[14] 刘华. 一种快速获取领域新词语的新方法[J]. 情报学报，2006，20(5)：17-23

[15] 彭 波. 搜索引擎的混合索引技术[J]. 计算机工程与应用，2004，22：16-18

(上接第 86 页)

在确定了最佳延迟时间后，可以根据式(1)将 Internet 平均访问时间的时间序列构造成 m 维的相空间，通过 G-P 算法来确定 Internet 访问时间的关联维数。在滞后时延 $\tau=3$ 下，m 由 3 逐渐递增到 12 时 $\ln(C_m(r))-\ln(r)$ 变化的曲线，如图 4 所示。

从图 4 看到，随着嵌入维数 m 增大，$\ln(C(r))$ 随 $\ln(r)$ 变化的曲线斜率呈逐渐收敛趋势，并且图 4 中从左到右的第 4 条曲线开始，各曲线的线性部分平行性较好，且 m 增大到 9 之后的各曲线几乎重叠，即 $D(m)$ 呈收敛趋势，故认为 $m=9$ 是 Internet 访问时间重构空间的最小嵌入维数。

对 $m=9$ 时 $\ln(C(r))-\ln(r)$ 曲线进行线性回归，得到 Internet 平均访问时间的饱和关联维数为：

$$D(m=9,\ \tau=3)=2.8308 \qquad (7)$$

饱和关联维数存在且为分数，故认为 Internet 访问时间的演化过程具有混沌特征，系统表现出对初始条件敏感的混沌振荡。

4 Internet 访问时间的预测模型

一般认为，某条路径的访问时间受多方面因素的影响，其中主要包括路由节点处理数据包的能力，路由节点所承受的负载，链路带宽以及数据包大小等[10]。文献[3]指出，对关联维数为 D 的混沌系统，必须至少使用 $\lceil D \rceil$ 维模型才可以表征影响目标系统 D 个重要的物理过程。

根据 Internet 访问时间的关联维数为 2.8308，说明至少需要使用 3 个物理特征量来表征 Internet 访问时间的混沌过程。在由多因素引起的 Internet 访问时间变化中，关联维数的确定为建立 Internet 访问时间的理论模型提供了变量个数的标准，而不必将各种有相关联系的因素都考虑在模型方程之中。因此，对 Internet 访问时间的模型建立如下：

$$\begin{cases} x_1=f_1(x_1,x_2,x_3,x_1^{(n)},x_2^{(n)},x_3^{(n)}) \\ x_2=f_2(x_1,x_2,x_3,x_1^{(n)},x_2^{(n)},x_3^{(n)}) \\ x_3=f_3(x_1,x_2,x_3,x_1^{(n)},x_2^{(n)},x_3^{(n)}) \end{cases} \qquad (8)$$

式(8)表征了 Internet 访问时间在混沌轨道运动上的数学模型，其中 $x_1\sim x_3$ 为表征 Internet 访问时间的重要物理量，$x_1^{(n)}$ 为该变量的 n 阶导数，该三维状态微分方程可以作为 Internet 访问时间的预测模型。

由于混沌轨道运动形式十分复杂，考虑到目前 Internet 访问时间的时间序列跨度较小，该样本数据集还不能确切地描述式(8)中方程的具体形式。确定 Internet 访问时间状态微分方程中的具体参数将是本文的下一步工作。

结束语 以 CAIDA skitter 采样得到的 Internet 访问为样本空间，对其进行延迟坐标相空间重构，得到 Internet 访问时间的最佳滞后时间为 3，以此说明了信息损失的时间尺度范围。通过 G-P 算法，得到 Internet 访问时间的相空间最小嵌入维数为 9，饱和关联维数为 2.8308，关联维数存在且为分数，说明了 Internet 访问时间的演化过程具有混沌特征，并以此建立了 Internet 访问时间三维微分方程在混沌运动轨道上的数学模型。该工作对于以混沌动力学来分析 Internet 网络特征量并在此基础上进一步预测及控制网络性能开辟了一条新路。

参考文献

[1] 张家才，周登勇. 从开放的复杂巨系统来看 Internet 中的大范围模式[J]. 系统仿真学报，2002，14(11)：1450-1454

[2] 徐野，赵海. Internet 网络的访问直径分析[J]. 计算机学报，2006，29(5)：690-698

[3] 黄润生. 混沌及其应用[M]. 武汉：武汉大学出版社，2000：183-246

[4] Takens F. Lecture notes in mathematics, 1981, 898: 366

[5] Kamrani E, Hamid H R. Modeling Internet delay dynamics for teleoperation//Proceedings of the 2005 IEEE Conference on Control Applications. 2005: 1528-1533

[6] CAIDA skitter Project. http://www.caida.org

[7] 张宏莉，方滨兴. Internet 测量与分析综述. 软件学报，2003，14(1)：110-116

[8] 吕金虎，陆君安. 混沌时间序列分析及其应用[M]. 武汉：武汉大学出版社，2002：60-63

[9] 李后强，汪富泉. 分形理论及其在分子科学中的应用. 北京：科学出版社，1993：190-195

[10] Han K H, Kim S K Y J, Kim Y J. Internet control architecture for Internet-based personal robot. Autonomous Robots, 2001, 10: 135-147