ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.7, July 2010, pp.1503−1514
doi: 10.3724/SP.J.1001.2010.03613
© by *Institute of Software*, *the Chinese Academy of Sciences*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

串匹配算法中模式串与文本之间关系的研究*

刘 萍[1,3+], 刘燕兵[1,2,3], 郭 莉[1,3], 方滨兴[1,3]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

[3](信息内容安全技术国家工程实验室,北京 100190)

Research on Relationship Between Patterns and Text in String Matching Algorithms

LIU Ping[1,3+], LIU Yan-Bing[1,2,3], GUO Li[1,3], FANG Bin-Xing[1,3]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](National Engineering Laboratory for Information Security Technologies, Beijing 100190, China)

+ Corresponding author: E-mail: liuping@ict.ac.cn

Liu P, Liu YB, Guo L, Fang BX. Research on relationship between patterns and text in string matching algorithms. *Journal of Software*, 2010,21(7):1503−1514. http://www.jos.org.cn/1000-9825/3613.htm

Abstract: It was assumed that the pattern and text characters are independent and uniformly distributed over a finite alphabet in classical string matching algorithms, and this assumption differs from real applications and causes many problems. Considering the probability distributions, the contexts of the characters, and the convenience of applications, this paper gives a concept hit rate and four extended concepts about it. Then it gives the theory analysis and detailed experiments with hit rate on the four classical algorithms. The map of the relationships is obtained between the hit rate and the algorithms' performance, and at the same time some valuable conclusions are made through above work. As a character variable, hit rate describes the relativity of patterns and text and can serve as guidelines in the algorithms design, analysis and some other extended research fields of the string matching.

Key words: string matching; probability distributions of character; relativity of strings

摘 要: 经典的串匹配算法设计和分析中假设"字符互相独立并且等概率出现",这与实际应用环境差异很大,导致出现很多问题.考虑了字符的概率分布和上下文的关联,同时兼顾应用的方便,提出了命中密度的概念.在给出基本定义和扩展定义后,通过对4种类型的代表性算法的理论和实验分析,给出了命中密度与算法性能之间的关系.同时,在对命中密度的分析中得出一些极具价值的结论.对命中密度概念的多角度理解以及对它与算法性能关系的深入剖析都说明,命中密度作为一个特征量,可以从一个侧面刻画模式串和文本之间的相关性,它对算法的设计和分析以及串匹配领域研究工作的扩展都具有指导意义.

关键词: 串匹配;字符概率分布;字符串相关性

中图法分类号: TP301 **文献标识码**: A

* Supported by the National Basic Research Program of China under Grant No.2007CB311100 (国家重点基础研究发展计划(973))
Received 2008-09-25; Accepted 2009-03-31

© 中国科学院软件研究所 http://www.jos.org.cn

所谓串匹配(string matching, pattern matching),就是给定一组特定的字符串集合 P,对于任意的一个字符串 T,找出 P 中的字符串在 T 中的所有出现位置.本文研究的是精确多模式串匹配(简称串匹配)的问题,文献[1]中给出的定义如下:

已知有限字符集合 Σ、模式串集合 $P=\{p_i \mid 1\leqslant i\leqslant r, p_i=p_1^i p_2^i \ldots p_{m_i}^i, p_j^i\in\Sigma^+, 1\leqslant j\leqslant m_i\}$ 和文本 $T=t_1t_2\ldots t_n$ ($t_i\in\Sigma$,$1\leqslant i\leqslant n$),求 P 中的模式串在文本 T 中的所有出现位置,即 $occur(P,T)=\{(p,|x|)|(\exists x\exists y)T=xpy, p\in P\}$.

1 问 题

在经历了数十年的研究后,已有上百种串匹配算法出现,对算法时间复杂度的分析也有很好的理论证明.文献[2]中证明了单模式串匹配的时间复杂度下界是 $\Omega((n\log_{|\Sigma|}m)/m)$,其中,$n$ 是文本长度,m 是模式串长度,$|\Sigma|$是字符集大小.文献[3]中将其推广到多模式串匹配的情形:在字符互相独立并且等概率出现的条件下,精确的多模式串串匹配算法的时间复杂度下限是 $\Omega((n\log_{|\Sigma|}mr)/m)$,其中,$r$ 是模式串个数.文献[4–6]中的算法都已经接近或达到这一最优值.

然而,在现有的串匹配算法设计和性能分析中,都只考虑了模式串集合 P 本身的特性,例如字符集大小、模式串长度(通常考虑最小长度)、模式串的个数等,对文本数据 T 也只考虑了它的长度,并没有充分考虑应用环境的其他因素以及这些因素之间的关系,尤其是模式串和文本之间的关系.此外,在性能的分析和测试中,使用的一个前提条件是“字符互相独立并且等概率出现”,这在实际应用环境中也往往不成立,直接导致的后果是很多串匹配算法的理论分析和实际效果差别较大[7,8],甚至同一种算法在不同文章中体现的性能也有很大差异[9−12].

随着串匹配技术理论研究的日益成熟和串匹配技术应用的不断深入,上述问题已经逐步开始被研究人员所关注.文献[13]分析了串匹配算法与文本熵之间的关系,给出了在界定熵文本上的匹配算法,并证明了其平均时间复杂度,以及最坏情况下的时间复杂度的范围.文章中认识到了传统的均匀分布假设前提的不合理性,但是只分别考虑了模式串的概率分布和文本的概率分布,并未结合考虑实际应用环境下它们之间的关系.文献[14]中设计了一种新算法,它利用概率加权的方法,计算出拥有最大平均跳步的目标串的比较序列,并利用该序列进行字符串的匹配.文中对最大跳跃距离的证明也是以“字符相互独立”为前提的.文献[15]利用模式串在样本文本上的概率分布选取字符集的一个子集,并使用它分别对模式串和文本进行采样,构建 semi-index 数据结构,从而加速串匹配算法.文章运用文本的一部分作为样本,巧妙地运用它们之间的关系设计了通过减小字符集方法来提高匹配速度的算法.文献[16]中以 3 个典型的网络内容安全应用系统为背景,详细介绍了其中使用的串匹配算法,并进行了算法级别和系统级别的性能测试.文章中虽然没有明确提出模式串和文本之间的关系对算法性能会产生一定的影响,但它对算法的跳跃次数、访存次数、cache miss 次数、可能的命中次数等方面都进行了详细的实验分析,工作值得借鉴.

为了描述模式串与文本之间的相关性,本文给出了命中密度(hit rate)的概念,概念的定义简单、清晰,在概率分布上的理解也很明确.将命中密度作为一个因素加入到对串匹配算法的理论分析中,可以得出命中密度与算法性能的关系,由此还可以得出一些极具应用价值的结论.

为了方便描述,下面给出本文中使用的符号:文本 T,长度为 n;模式串集合 P,个数为 r,最小长度为 m,字符集为 Σ;结果集合 R,个数为 K,每个结果为(id,pos)形式,其中,id 为模式串的唯一标识,pos 为匹配成功的模式串位置.

2 命中密度的概念

正如第 1 节中所述,孤立地关注模式串的特征或者文本的特征,都不能很全面地把握串匹配技术.尤其是在实际应用中,环境的变化因素是很多的.以网络应用为例,在病毒检测(例如 ClamAV 开源系统)、网络异常事件检测(例如 SNORT 开源系统)、网络协议识别(例如 L7-Filter 开源系统)等领域中,都大量地使用了串匹配技术,而选择哪种算法才是最合适的、选择的算法是否会造成系统性能瓶颈、系统运行中是否需要进行算法更换以及如何更换等,都是系统设计人员面临的挑战性问题,而这些问题从某些方面来说是动态的,不能只根据模式串集合而得到答案.这种动态性来源于网络数据的变化以及这些变化和模式串之间的关系.图 1 是江民公司公布的

© 中国科学院软件研究所 http://www.jos.org.cn

2004 年排名第一的病毒“欢爱病毒”在一个月内的走势图[17],可以看出,病毒的数量在一定时间范围内的变化是非常剧烈的.可以想象,这对在线查杀毒系统造成的压力是不同的,必定会引起系统性能的振荡.

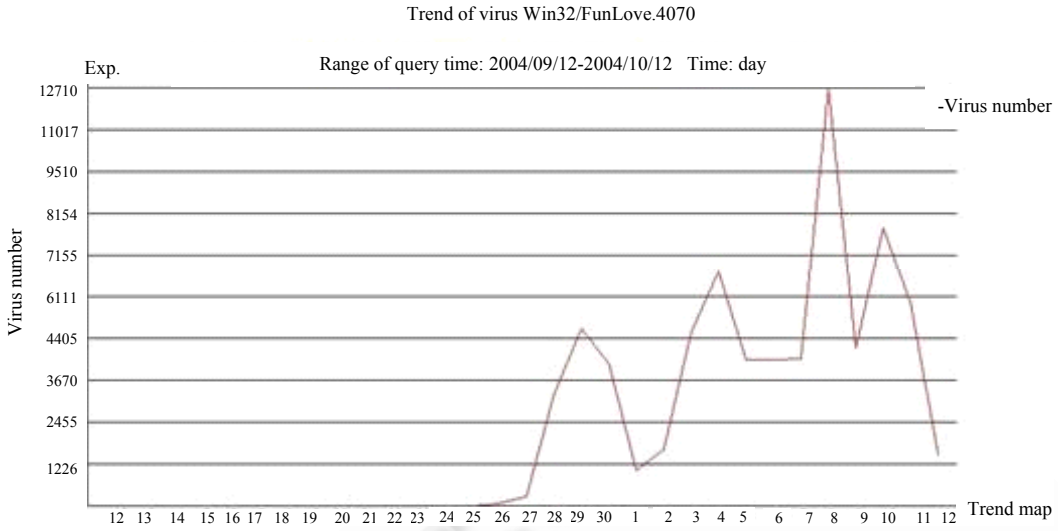


Fig.1 Data burst out trend of virus “FunLove” through 1 month (provided by Jiangmin SciTech Corp. Ltd.)

图 1 “欢爱病毒”在一个月内的爆发数量的走势图(江民公司提供)

造成这种波动现象的原因必然是与模式串和文本都相关,因此研究它们之间的关系很有必要.为了刻画模式串和文本之间的关系,于是我们给出命中密度的概念.

2.1 命中密度的定义

从串匹配算法的定义可以看出,对于模式串集合 P 和输入文本 T,它们之间的关系只有通过匹配结果集合 R 来互相关联(当然,字符集的大小是一个因素,但通常对于确定的(P,T)它是不改变的,只与应用环境相关.例如:在生物信息学领域中,字符集的大小可能为 4).因此,命中密度可以定义如下:

定义 1. 给定模式串集合 P 和文本 T,命中密度ρ是指单位文本长度内命中的模式串次数,$\rho=F(P,T)$.

命中密度的概念是力图描述串匹配算法中模式串和文本数据之间的关系,因此它可以在模式串和文本数据的两个维度上分别进行细定义.如图 2 所示,我们分别定义了文本上的单位置上的命中密度和平均命中密度、模式串集合上的单模式串命中密度和平均命中密度.

定义 2. 文本上的单位置命中密度 ρ_{T_i} 为在文本 T 的位置 i 上命中的模式串个数与文本长度之比,即 $\rho_{T_i}=\frac{mt_i}{n}$.其中,$mt_i$ 为文本位置 i 上命中的模式串的个数,n 为文本 T 的长度.

定义 3. 文本上的平均命中密度 $\overline{\rho}_T$ 为在文本 T 上命中的模式串次数与文本长度之比,即 $\overline{\rho}_T=\frac{K}{n}$.其中,$K$ 为全部模式串命中的次数,n 为文本的长度.

定义 4. 模式串上的单串命中密度 ρ_{P_j} 为模式串集合 P 中的一个模式串 P_j 命中的次数与全部模式串的命中次数之比,即 $\rho_{P_j}=\frac{mp_j}{K}$.其中,$mp_j$ 为模式串 P_j 的命中次数,K 为全部模式串命中的次数.

定义 5. 模式串上的平均命中密度 $\overline{\rho}_P$ 为在全部命中的模式串次数与模式串集合中模式串个数之比,即 $\overline{\rho}_P=\frac{K}{r}$.其中,$K$ 为全部模式串命中的次数,r 为模式串集合中模式串的个数.

从上述定义可以看出,文本方向上的命中密度反映的是命中的模式串在文本中分布的情况,单位置上的是

© 中国科学院软件研究所 http://www.jos.org.cn

个体情况,而平均命中密度是一个平均值,表示命中的模式串在文本上的稠密程度.同样,模式串方向上的命中密度反映的是命中的模式串在模式串集合中的分布情况,单串上的是个体情况,而平均命中密度是一个平均值,它表示命中的模式串在模式串集合中的分布情况.

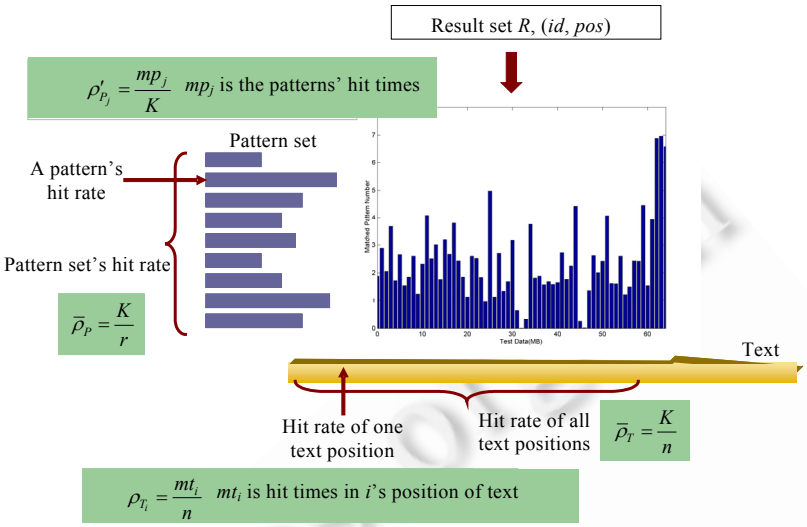


Fig.2 Four extended concepts of hit rate

图 2 4 个扩展命中密度的概念及其定义

2.2 对命中密度概念的理解

命中密度的概念通过串匹配算法中很直观的几个参数值给出了定义,它从一个侧面在文本和模式串之间建立了联系,反映了文本和模式之间的相似程度,进而可以描述它对算法性能产生的影响.下面,从模式串和文本字符概率分布的角度再来理解一下命中密度的概念.

在串匹配中,最基本的操作是单个文本字符和单个模式串字符的比较,比较的结果有两种:相等或者不相等,没有其他结果(这里是相对于近似匹配算法、序列比对算法中的打分系统来说的).文本字符和模式串字符相等的概率越小,那么(跳跃型)串匹配算法实现跳跃的距离就越大,算法的速度就越快.为此,我们引入相关度的概念来度量文本和模式串的相似程度.

定义 6(文本和模式串的 1-阶相关度). 设模式串各个字符出现的概率分布为 $p(i)(0\leqslant i<\sigma)$;文本的各个字符的概率分布为 $q(i)(0\leqslant i<\sigma)$.模式串和文本的 1-阶相关度为 $R_1=\sum_{i=0}^{\sigma-1}p_iq_i$,它是单个模式串字符和单个文本字符匹配的概率.

从定义中可以看出 $0\leqslant R_1\leqslant 1$.$R_1$ 越小,说明文本和模式串的相似程度越小,此时的匹配过程适用于 Horspool, WM(WuManber),SBOM(set backward Oracle matching)这一类的跳跃型算法,通过获取较大的跳跃距离得到较高的匹配速度;相反,R_1 越大,说明文本和模式串相似程度越大,这时跳跃型算法会达到最坏的时间复杂度 $O(mn)$,因此更适合采用线性的算法,如 AC(Aho-Corasick)算法.

1-阶相关度的概念反映的是字符之间相互独立的情况,而在实际的自然语言中,文本或者模式串的前后字符是有关联性的.为了反映这种关联性,我们给出了 k-阶相关度的定义.

定义 7(文本和模式串的 k-阶相关度). 设长度为 k 的字符串 $s_1s_2\ldots s_k$ 在模式串中出现的概率分布为 $p(s_1s_2\ldots s_k)$,长度为 k 的字符串 $s_1s_2\ldots s_k$ 在文本中出现的概率分布为 $q(s_1s_2\ldots s_k)$,那么定义模式串和文本的 k-阶相关度为 $R_k=\sum_{s_1s_2\ldots s_k\in\Sigma^k}p(s_1s_2\ldots s_k)\cdot q(s_1s_2\ldots s_k)$,它反映的是长度为 k 的模式串子串和长度为 k 的文本子串的匹配概率.

© 中国科学院软件研究所 http://www.jos.org.cn

以实际算法中的使用为例:Horspool 算法只根据最后一个字符计算跳跃距离,因此 1-阶相关度 R_1 即能反映它的跳跃能力;Wu-Manber 算法是根据最后 b 个字符进行跳跃,所以 b-阶相关度 R_b 反映的是它的跳跃能力;而 SBOM 算法是根据子串决定跳跃距离的,因此可以使用 $R_1,R_2,\ldots,R_m$ 反映它的跳跃能力.

特别地,m-阶相关度反映的就是本文定义的命中密度.这是因为

$$R_m=\sum_{s_1s_2\ldots s_m\in\Sigma^m}p(s_1s_2\ldots s_m)\cdot q(s_1s_2\ldots s_m)=\sum_{i=1}^{r}p(s_1s_2\ldots s_m=P_i)\cdot q(s_1s_2\ldots s_m=P_i)$$

$$=\sum_{i=1}^{r}(P_i\text{在模式串中出现的频率})\cdot(P_i\text{在文本中出现的频率})$$

$$=\sum_{i=1}^{r}\frac{1}{r}\cdot\rho_i=\frac{\rho}{r}.$$

3 串匹配算法中的命中密度

本节将在定义的 4 种命中密度中选择两个加以分析和实验,进一步阐述命中密度与串匹配算法之间的关系,以及从命中密度的角度探索一些真实的串匹配算法在真实环境中运行的规律.

3.1 文本上的平均命中密度对串匹配算法性能的影响

串匹配算法按运行机制可以分为跳跃型算法和非跳跃型算法两类,按算法使用的数据结构可以分为基于自动机的算法和基于查表(主要是哈希表)的算法两类.根据这两种分类标准,串匹配算法可以分为如下 4 类,我们选取每一类中的代表性算法进行理论分析和实验:

- 基于自动机的非跳跃型算法:代表算法为高级 AC 算法[18];
- 基于自动机的跳跃型算法:代表算法为 SBOM 算法[6];
- 基于查表的非跳跃型算法:代表算法为 SOG(shift-or with q-gram)算法[19];
- 基于查表的跳跃型算法:代表算法为 WM 算法[4].

3.1.1 理论分析

下面以 4 种代表算法为例,分析它们匹配过程的平均时间复杂度与命中密度之间的关系.

高级 AC 算法.

定理 1. 高级 AC 算法匹配的时间复杂度为 $O(n(1+c\rho))$,其中,常数 c 为报告一次模式串匹配所需要的时间.

证明:在高级 AC 算法中,对于每一个字符需要进行一次自动机的状态转换,并检查当前状态是否有命中的模式串.设进行一次状态转换的时间代价为 1,报告一次模式串匹配所需时间为 c.由于对于每个文本位置,有模式串命中的概率为ρ,所以每处理一个文本字符的时间为 $1+c\rho$.因此有:高级 AC 算法匹配的时间复杂度为 $O(n(1+c\rho))$,其中,常数 c 为报告一次模式串匹配所需要的时间. □

SBOM 算法.

定理 2. SBOM 算法匹配的时间复杂度为 $O\left(n\left[\rho(m+c)+(1-\rho)\frac{\log_{|\Sigma|}mr}{m-\log_{|\Sigma|}mr}\right]\right)$,其中,常数 c 为进行一次模式串和文本校验过程的时间代价.

证明:在没有命中的情况下,SBOM 算法的时间复杂度为 $O\left(n\frac{\log_{|\Sigma|}mr}{m-\log_{|\Sigma|}mr}\right)$.对于每个文本位置,当有模式串命中时,需要扫描整个匹配窗口内的字符并进行模式串和文本的校验,时间代价为 $O(m+c)$.当没有模式串命中时,处理一个文本位置的平均时间代价为 $O\left(\frac{\log_{|\Sigma|}mr}{m-\log_{|\Sigma|}mr}\right)$.有模式串命中的概率是$\rho$,没有模式串命中的概率是

© 中国科学院软件研究所 http://www.jos.org.cn

1−ρ,因此有:SBOM 算法匹配的时间复杂度为 $O\left(n\left[\rho(m+c)+(1-\rho)\frac{\log_{|\Sigma|}mr}{m-\log_{|\Sigma|}mr}\right]\right)$,其中,常数 c 为进行一次模式串和文本校验过程的时间代价. □

SOG 算法.

定理 3. SOG 算法匹配的时间复杂度为 $O(n(1+c\rho\log r))$,其中,常数 c 为进行一次模式串和文本校验过程的时间代价.

证明:SOG 算法的运行机制与高级 AC 算法比较相似,处理一个字符的时间代价为 $O(1)$.但是对于可能的模式串命中,SOG 算法需要进行校验.SOG 的校验过程是在整个模式串集合上进行二分查找,校验的时间代价为 $c\log r$.因此有:SOG 算法匹配的时间复杂度为 $O(n(1+c\rho\log r))$,其中,常数 c 为进行一次模式串和文本校验过程的时间代价. □

WM 算法.

定理 4. WM 算法匹配的时间复杂度为 $O\left(n\left[\rho(c_h+c)+(1-\rho)\frac{c_h}{(m-b+1)\left(1-\frac{mr}{2|\Sigma|^b}\right)}\right]\right)$,其中,常数 c_h 是进行一次哈希值计算的时间代价,常数 c 是进行一次模式串和文本校验过程的时间代价.

证明:WM 算法的分析与 SBOM 类似.在没有命中的情况下,WM 算法的时间复杂度为 $O\left(\frac{n}{(m-b+1)\left(1-\frac{mr}{2|\Sigma|^b}\right)}\right)$.对于每个文本位置,当有模式串命中时需要进行一次哈希值的计算并进行模式串和文本的校验,时间代价为 $O(c_h+c)$.当没有模式串命中时,处理一个文本位置的平均时间代价为 $O\left(\frac{c_h}{(m-b+1)\left(1-\frac{mr}{2|\Sigma|^b}\right)}\right)$.有模式串命中的概率是$\rho$,没有模式串命中的概率是 1−$\rho$,因此有:WM 算法匹配的时间复杂度为 $O\left(n\left[\rho(c_h+c)+(1-\rho)\frac{c_h}{(m-b+1)\left(1-\frac{mr}{2|\Sigma|^b}\right)}\right]\right)$,其中,常数 c_h 是进行一次哈希值计算的时间代价,常数 c 是进行一次模式串和文本校验过程的时间代价. □

综合以上分析,我们可以得出以下推论:

推论. 在其他影响因素不变的情况下,串匹配算法的匹配的时间复杂度是命中密度的线性函数.

3.1.2 实验分析

本节中,我们通过实验的方法来验证对算法性能与命中密度之间关系的理论分析结果.

实验数据的构造

因为在真实数据中很难获得不同命中密度的数据,因此对于有关命中密度与匹配性能关系的实验需要在随机构造的数据上进行.使用随机数据进行算法的分析和测试是串匹配领域中比较通用的方法,但是考虑命中密度因素的随机数据构造方法与原有方法仍然有些不同.这是因为原有方法是完全随机生成的数据,通常没有命中,很难控制不同的命中密度水平.例如:当字符集大小$|\Sigma|$=256,模式串长度 m=4 字节,模式串个数 r=5000,文本长度 n=1M 字节时,在完全随机生成数据的情况下,命中模式串个数的期望值为 $EX\leqslant n\frac{r}{|\Sigma|^m}=2^{20}\cdot\frac{5000}{256^4}=1.2$.

© 中国科学院软件研究所 http://www.jos.org.cn

本文构造随机数据的方法如下:

Step 1. 根据设定的参数$|\Sigma|,r,m,n$,随机生成 r 个长度为 m 的模式串,随机生成长度为 n 的文本串.模式串和文本串的字符均匀地取自字符集$\{0,1,2,\ldots,|\Sigma|-1\}$;

Step 2. 根据设定的命中密度ρ和文本长度 n,确定命中的模式串个数 $hit\_num=\rho m$;

Step 3. 随机地从模式串集合中取出 hit_num 个模式串(一个模式串可重复取多次),将这 hit_num 个模式串均匀地、无重叠地撒入到长度为 n 的文本串中.这样就可以确保命中密度的水平是ρ.

需要说明的是,在上述构造方法中,为了确保命中的模式串不互相干扰破坏,必须要求模式串是以无重叠的方式撒入到文本中的,这样就要求构造的命中密度ρ不能超过 $1/m$.这种要求通常是合理的,因为在实际应用中,出现命中的模式串互相重叠的情况比较少,特别是在很多网络应用中,命中密度更小,通常不会超多 1/100.

实验结果及分析

首先,以典型参数配置为例给出算法性能与命中密度的关系,如图 3 所示.

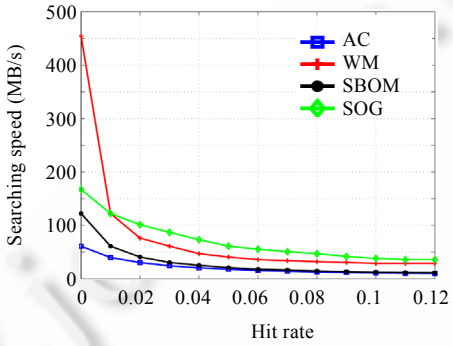


Fig.3 Relationship between the matching speeds and hit rates, $|\Sigma|=256$, $r=5000$, $m=8$

图 3 $|\Sigma|=256,r=5000,m=8$ 时,算法性能与命中密度的关系

典型参数下的实验是给出一个类似真实应用环境的示例,真实应用中,字符集大小为 256,模式串最小长度通常较小,模式串个数适中,此时可以清楚地看到,4 种算法的匹配速度都随着命中密度的增大而显著降低.其中:AC 和 SOG 是非跳跃算法,它们的性能表现较稳定;而跳跃型算法 SOG 和 WM 的性能随着命中密度的增大急剧下降.这是由于命中密度大时将会大幅度地降低跳跃型算法的跳跃距离,从而降低它们的速度,而这对非跳跃算法而言基本没有影响.

其次,为了更详细地说明算法的各主要性能参数与命中密度的关系,我们从 3 个方面进行了实验:模式串长度变化下命中密度与性能的关系,长度取值为 4,8,32 字节,实验结果见表 1($|\Sigma|=256,r=5000$);模式串个数变化下命中密度与性能的关系,个数取值为 500,10 000,50 000,实验结果见表 2($|\Sigma|=256,m=8$);字符集大小变化下命中密度与性能的关系,字符集大小取值为 4,32,128,实验数据见表 3($r=5000,m=8$).实验中生成的文本长度为 20MB.表中数据的单位为 MB/s.

表 1 中给出的是在模式串长度变化情况下,算法性能与命中密度的关系(注:$m=20$ 时,命中密度最大为 $1/m=1/20=0.05$).可以看出,每种算法的匹配速度随着命中密度下降的趋势有所不同.非跳跃型算法 AC 和 SOG 比较匀速,而跳跃型算法在模式串长度增大时,性能下降的速度明显加快.例如:SBOM 算法在 $m=4$ 时,命中密度从 0.01 变化到 0.05 时相应的速度从 67MB 变化到 34MB,性能下降尺度约为 12-12-7-4,共下降了 50%;在 $m=8$ 时,相应的速度从 61MB 变化到 214MB,性能下降尺度约为 21-10-5-3,共下降了 66%;在 $m=20$ 时,相应的速度从 33MB 变化到 8MB,性能下降尺度约为 14-6-3-2,共下降了 76%.这是因为跳跃型算法的性能受模式串长度因素的影响非常大所造成的.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 1 Relationship between the matching speeds and hit rates when the lengthes of pattern is different

表 1 不同模式串长度情况下,串匹配算法的匹配速度与命中率之间的关系

Hit rate	*m*=4				*m*=8				*m*=20			
	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG
0.00	61.133	167.31	**87.226**	152.99	61.133	454.13	**122.27**	167.31	55.446	608.73	**304.37**	167.32
0.01	50.727	76.294	**67.636**	122.26	39.736	122.26	**61.133**	122.27	22.599	151.38	**33.898**	122.27
0.02	43.547	60.873	**55.446**	101.46	30.469	76.294	**40.755**	101.46	14.192	87.226	**19.073**	87.493
0.03	36.68	43.613	**43.547**	82.928	24.432	61.133	**30.534**	86.961	11.098	73.36	**13.27**	76.294
0.04	33.898	39.736	**38.147**	67.958	20.583	46.979	**25.431**	73.36	9.341	61.133	**10.174**	63.157
0.05	30.501	34.553	**34.553**	65.32	17.96	40.755	**21.052**	61.133	8.633	61.133	**8.361**	58.99
0.06	29.046	31.031	31.579	55.446	15.66	35.942	17.96	55.446	—	—	—	—
0.07	24.411	27.75	27.326	50.908	14.192	33.898	16.5	50.817	—	—	—	—
0.08	24.411	26.54	26.54	46.979	12.716	32.11	14.53	46.979	—	—	—	—
0.09	22.907	24.411	24.771	43.547	11.966	30.534	13.363	41.645	—	—	—	—
0.10	21.79	22.599	22.599	40.697	11.098	29.075	12.369	38.147	—	—	—	—

Table 2 Relationship between the matching speeds and hit rates when the size of pattern set is different

表 2 不同模式串个数情况下,串匹配算法的匹配速度与命中率之间的关系

Hit rate	*r*=500				*r*=10000				*r*=50000			
	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG
0.00	114.45	454.13	608.73	167.31	38.147	153.82	101.46	167.32	9.25	114.45	76.294	153.00
0.01	101.87	202.91	304.36	**130.64**	27.723	96.331	46.979	**122.27**	8.847	67.636	33.939	**101.46**
0.02	91.699	151.38	202.91	**107.96**	22.599	76.294	32.11	**101.46**	8.515	50.817	22.599	**76.294**
0.03	87.493	107.97	151.38	**91.699**	19.073	61.133	24.411	**82.928**	8.25	41.645	16.969	**61.133**
0.04	87.226	87.493	122.27	**86.961**	16.349	53.88	19.69	**67.797**	8.028	35.852	13.566	**50.999**
0.05	86.698	76.294	101.46	**73.36**	14.538	46.979	16.49	**61.133**	7.823	31.064	11.304	**44.634**
0.06	76.294	70.643	96.331	**67.636**	12.993	43.547	14.192	**53.88**	7.629	27.75	9.842	**40.64**
0.07	76.294	67.636	87.226	**60.873**	11.818	40.697	12.461	**46.979**	7.503	26.128	8.638	**35.852**
0.08	76.294	67.797	76.294	**55.446**	10.962	38.147	11.233	**43.547**	7.353	24.391	7.728	**32.11**
0.09	76.294	67.797	73.36	**50.817**	10.174	38.147	10.178	**40.64**	7.267	22.599	7.016	**29.526**
0.10	76.294	73.36	67.797	**46.979**	9.537	35.942	9.393	**36.68**	7.208	21.052	6.426	**27.3**

表 2 中给出的是在模式串个数变化情况下,算法性能与命中密度的关系.可以看出,每种算法的匹配速度随着命中密度下降的趋势有所不同,基于自动机类型的算法 AC 和 SBOM 整体相对较匀速,而基于查表的算法在模式串个数增大时,随着命中密度的增大,性能有所下降,下降的速度在密度较低(<0.05)时较快,之后稍缓慢.例如: SOG 算法在 r=500 时,命中密度从 0.01 变化到 0.10 时,相应的速度从 130MB 变化到 46MB,性能下降了 65%,变化步长约为 23-26-4-13-6-7-5-5-4;在 r=5000 时,相应的速度从 101MB 变化到 27MB,性能下降了 72%,变化步长约为 24-25-11-6-4-5-3-3-2.造成这种现象的原因是,基于查表的算法其运行时间随着命中密度的变化受模式串个数影响相对较大.例如,在 SOG 算法中,其影响系数为 $c\log r$,这种关系在上述变化步长中表现得非常明显.

Table 3 Relationship between the matching speeds and hit rates when the sizes of alphabet is different

表 3 不同字符集大小情况下,串匹配算法的匹配速度与命中率之间的关系

Hit rate	*S*=4				*S*=32				*S*=128			
	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG	AC	WM	SBOM	SOG
0.00	33.268	**45.776**	24.391	12.716	55.446	**202.91**	101.82	183.40	52.4	**167.32**	122.27	202.91
0.01	33.939	**43.547**	23.49	11.102	38.147	**153.82**	58.99	122.27	30.566	**101.82**	55.446	122.27
0.02	32.11	**43.613**	22.599	10.233	30.469	**151.38**	41.645	101.46	24.432	**86.961**	38.147	101.46
0.03	30.534	**40.755**	21.79	9.247	25.431	**122.27**	33.898	82.928	21.052	**73.36**	29.075	87.226
0.04	30.501	**40.697**	21.052	8.477	21.807	**101.46**	27.326	68.12	18.494	**65.32**	23.49	76.294
0.05	29.016	**38.147**	18.506	7.927	19.463	**101.46**	23.49	61.133	16.5	**58.99**	19.704	63.157
0.06	26.565	**38.147**	19.279	7.325	17.435	**87.493**	20.349	55.446	15.018	**55.446**	17.435	55.446
0.07	26.565	**35.942**	19.463	6.783	15.921	**82.928**	18.131	50.727	13.983	**50.908**	15.382	50.817
0.08	26.94	**35.852**	18.885	6.492	14.538	**76.294**	16.5	43.613	12.987	**50.817**	14.192	46.979
0.09	26.516	**35.191**	17.759	6.022	13.457	**70.469**	15.018	40.755	12.206	**49.413**	12.993	43.547
0.10	25.431	**33.939**	18.305	5.758	12.455	**67.797**	13.882	38.147	11.44	**48.165**	11.966	38.925

表 3 中给出的是在字符集大小变化情况下,算法性能与命中密度的关系.可以看出,AC 和 SOG 算法的性能随着命中密度的变化基本与字符集大小的变化无关,是很均匀的,而 WM 和 SBOM 算法的变化情况相对大一些.

© 中国科学院软件研究所 http://www.jos.org.cn

尤其是 WM 算法,它的表大小直接与|Σ|和 b 相关,当命中密度不断增大时,WM 算法的匹配时间随着|Σ|的增大而增大,由算法的理论分析也可以看出,这种增大的幅度将会有所加强.从实验数据上看,当|Σ|从 4 增大到 32 和 128 时,算法的匹配性能之比(0.01 时的速度/0.1 时的速度)也由原来的 1.3 增大到 2.28 和 2.1.

由上述分析可以看到,随机数据下的实验结果完全可以验证对算法时间复杂度的结果.我们可以得出如下结论:

结论 1. 串匹配算法的匹配速度随命中密度的增大而降低.

进一步分析实验结果,结合图 3 的曲线图可以看出,在不同的命中密度区域上,不同算法的性能表现很不相同.例如:在密度小于 1%时,WM 算法的性能表现最好;但当密度大于 1%时,SOG 算法的表现就超过了 WM,并且一直比其他算法的性能更优.WM 算法的性能曲线图非常陡峭,这说明它受命中密度的影响非常大,SBOM 次之,AC 和 SOG 算法最平缓.因此我们还可以得出如下结论:

结论 2. 不同的串匹配算法对文本命中密度的敏感程度不同.

3.2 模式串上的命中密度

在模式串方向上,单模式串的命中密度反映了一个模式串个体的命中与全部命中次数之间的关系,它可以从一个较小的颗粒度上观察命中密度的情况.本节中的实验选自真实的数据集:

- 模式串集合:Snort 模式串集合[20]:来自开源系统 SNORT,版本:2006 年 10 月 11 日,选取每条规则中的字符串片断,共计 5 029 个;ClamAV 模式串集合[21]:来自开源系统 ClamAV,版本:0.90.1,选取每条规则中最长的字符串片断,共计 79 561 个;
- 文本数据:选自 MIT 公布的一组用于 IDS 测试的真实网络数据[22],本文中使用了第 1 个星期的共 5 天的数据,每天的数据大约在 600MB~800MB 左右.

实验分为两组,分别用 Snort 模式串和 ClamAV 模式串在 MIT 的 5 天的文本数据上进行匹配,统计全部模式串命中的次数之和,并按照命中次数从大到小,统计了 Top 10 的命中次数之和.实验结果如图 4 和图 5 所示.

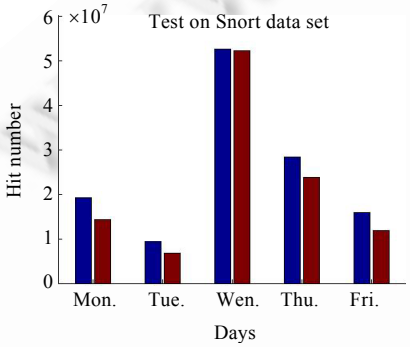


Fig.4 Test of hit rate on patterns with Snort data set

图 4 Snort 数据集上的模式串命中密度

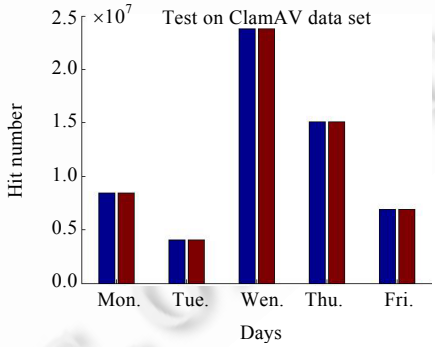


Fig.5 Test of hit rate on patterns with ClamAV data set

图 5 ClamAV 数据集上的模式串命中密度

图中左侧是全部模式串命中次数之和,右侧是 TOP 10 的命中次数之和.两组数据上的命中次数都非常多:Snort 数据集上,命中的模式串个数约有 700~800 个左右,命中的次数从几十万次到百万次不等;ClamAV 数据集上,命中的模式串个数较少,约有 20~30 个左右,命中的次数也是从几十万次到百万次不等.两组数据上的一个共同点是:Top 10 之和与全部命中次数的比例很大,都在 90%以上;在 ClamAC 中,这个比例更高.

上述现象的原因是,在真实的应用环境中,事件的发生在一段时间内(即在一定时间区间的文本数据上)通常是比较集中的.例如:Snort 系统中,一天内甚至是一周内,频繁发生的网络异常事件总是集中在有限的几个上;而在 ClamAV 系统上,表现的则是病毒爆发的集中性.

更进一步地,我们将每组命中数据上 5 天的数据进行更细致的比较,找出其中相同的模式串,发现 Snort 数据

© 中国科学院软件研究所 http://www.jos.org.cn

上共有 32 个,ClamAV 数据上有 21 个.这个相对固定的小模式串集合的存在,反映了真实数据集上的固有特性.由此我们可以得出:

结论 3. 在真实应用环境中,模式串上的命中率符合 2-8 定律,即少数模式串的命中次数占全部模式串命中次数的大部分.

4 命中密度对算法性能影响的深入分析

在给出了命中密度的定义之后,我们通过理论分析对 4 种算法匹配过程的时间复杂度进行了细致的分析,并通过实验验证了理论分析的结果.即:命中密度,尤其是文本上的平均命中密度对串匹配算法的性能有很大的影响.造成这种现象的原因有以下几个方面:

首先,最直接的原因来自于命中导致的代价开销.对于某些串匹配算法,每次命中都需要进行相应的校验,这需要花费较多的时间.例如:对于 SOG 算法,每一次可能的命中都需要在整个模式串集合上进行二分查找,这个开销是比较大的;对于 SBOM 算法,基于子串的匹配可能会有误差,当有一个可能的命中时,也需要在模式串和文本之间进行校验,这个过程的时间开销与模式串长度 m 相关;

其次,对于跳跃型算法,频繁的命中会导致平均跳跃距离的降低.对于跳跃型算法,如 WM,SBOM 等,平均跳跃距离是影响算法性能的最关键的因素.平均跳跃距离越小,算法运行越慢.频繁的命中会导致两个匹配窗口之间的跳跃距离减小.图 6 是不同的命中密度水平下,WM 和 SBOM 算法的平均跳跃距离的变化图($|\Sigma|$=256, r=5000,m=8).可以看出,随着命中密度的增大,算法的平均跳跃距离不断减小;

最后,命中密度的大小反映了文本和模式串之间相关性的大小.命中密度越大,表明文本和模式串之间越相似,因而在每个匹配窗口内需要处理的开销也相应增多.例如对于 SBOM 算法,文本和模式串越相似,在匹配窗口中反向扫描的字符数也越多,相应的自动机中结点访问的层次也越深,算法的 Cache 局部性也会降低.

此外,深层次的原因还可以从以下几个方面来分析:算法匹配过程中产生的可能命中(也称为候选命中)的次数、Cache 的大小、Cache Miss 的次数等.其中,有些因素是串匹配算法领域独有的,而有些则是计算机科学中各种算法研究领域都要长期面对的研究课题.

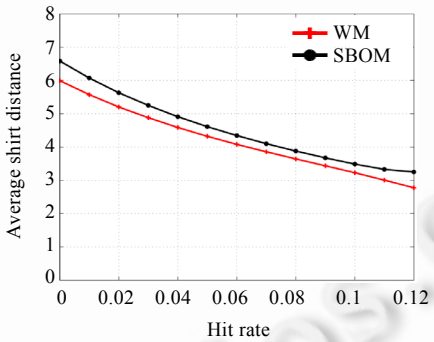


Fig.6 Relationship between hit rates and the skip distance of WM and SBOM

图 6 WM 和 SBOM 算法的平均跳跃距离与命中密度的关系

5 结束语

在串匹配算法研究过程中,模式串与文本之间的关系是不应回避的.本文给出了命中密度的概念,并从概率分布的角度对其进行了解释.运用此概念,我们对 4 类算法中的典型算法的时间复杂度进行了理论分析,得出了算法性能与命中密度的关系,并通过随机数据的实验进行了验证.此外,在研究模式串上的命中密度时,我们发现了非常有趣的 2-8 定律现象.文中通过分析和实验得出了关于命中密度的几个很有用的结论.

对命中密度进行研究的意义体现在两个方面:

© 中国科学院软件研究所 http://www.jos.org.cn

- 可以针对不同的应用设计新算法.设计思路有两种:思路 1 是“量身定做”,例如在网络内容安全应用中,真正命中的网络流量<10%,命中率非常低,因此可以设计高效的过滤型算法,快速处理 no-match 的数据;思路 2 是“以不变应万变”,例如结合 AC 算法和 WM 算法的原理(自动机数据结构)特点,设计自适应文本数据流的算法.此外,充分利用模式串命中密度中的 2-8 定律现象也可以设计基于分组的串匹配算法;
- 在多个候选算法中选择最优算法.思路是“让合适的人干合适的事”,这其中,静态算法选择可以应用在特定应用系统的设计之初.例如:针对病毒检测问题,在系统设计时可以人工地选择一种最优的算法;动态算法选择可以应用在数据未知的情况下,例如:对在线内容过滤系统可实时监测,根据不同的命中密度的变化而动态调整在线算法,从而提高整个系统的吞吐率.

当然,目前给出的命中密度的定义还有一定的局限性,它虽然能从一个稀疏/稠密的角度去描述命中的情况,但是它不能很好地描述全部命中在文本中的分布情况.例如:虽然都是 K 个命中,但分布可能是均匀的,也可能是正态分布的情况.相应地,在构造数据时也使用了均匀分布的情况,这与实际应用环境还有一定的差异.因此,在特定命中密度下,不同的数据分布是否对算法性能有影响,以及如何给出特征量来描述这种分布的不均匀性,都是值得深入探讨的问题.

References:

[1] Navarro G, Raffinot M. Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences. Cambridge University Press, 2002. 41−74.

[2] Yao AC. The complexity of pattern matching for a random string. SIAM Journal on Computing, 1979,8(3):368−387. [doi: 10.1137/0208029]

[3] Navarro G, Fredriksson K. Average complexity of exact and approximate multiple string matching. Theoretical Computer Science, 2004,321(2-3):283−290. [doi: 10.1016/j.tcs.2004.03.058]

[4] Sun W, Manber U. A fast algorithm for multi-pattern searching. Technical Report, TR-94-17, Tucson: University of Arizona, 1994.

[5] Raffinot M. On the multi backward dawg matching algorithm (multibdm). In: Baeza-Yates R, ed. Proc. of the 4th South American Workshop on String Processing. Carleton University Press, 1997. 149−165. http://sunsite.dcc.uchile.cl/~sccc/wsp97/wsp97.html

[6] Crochemore M, Allauzen C, Raffinot M. Factor oracle: A new structure for pattern matching. Theory and Practice of Informatics, 1999,291−306.

[7] Morris JH, Knuth DE, Pratt VR. Fast pattern matching in strings. SIAM Journal on Computing, 1977,6(2):323−350. [doi: 10.1137/0206024]

[8] Boyer RS, Moore JS. A fast string searching algorithm. Communications of the ACM, 1977,20(10):762−772. [doi: 10.1145/359842.359859]

[9] Zhang X, Tan JL, Chen XQ. An improved Wu-Manber multiple patterns match algorithm. Journal of Computer Applications, 2003, 23(7):29−35 (in Chinese with English abstract).

[10] Song H, Dai YQ. A new fast string matching algorithm for content filtering and detection. Journal of Computer Research and Development, 2004,41(6):940−948 (in Chinese with English abstract).

[11] Sun XS, Wang Q, Guan Y, Wang XL. An improved Wu-Manber multiple-pattern matching algorithm and its application. Journal of Chinese Information Processing, 2006,20(2):47−53 (in Chinese with English abstract).

[12] Li WN, E YP,Ge JG, Qian HL. Multi-Pattern matching algorithms and hardware based implementation. Journal of Software, 2006, 17(12):2403−2415 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2403.htm [doi: 10.1360/jos172403]

[13] Chen SF, Reif JH. Fast pattern matching for entropy bounded text. In: Storer JA, Cohn M, eds. Proc. of the DCC’95 Data Compression Conf. Snowbird. UT Los Alamitos: IEEE Computer Society Press, 1995. 282−291. http://openlibrary.org/books/OL20845375M/DCC_’95_Data_Compression_Conference

[14] Chen W, Liu YJ, Lu ZX. Best reordered string-matching algorithms. Computer Engineering and Design, 2004,25(9):1430−1436 (in Chinese with English abstract).

[15] Claude F, Navarro G, Peltola H, Salmela L, Tarhio J. Speeding up pattern matching by text sampling. In: Amir A, Turpin A, Moffat A, eds. Proc. of the 15th Int’l Symp. on String Processing and Information Retrieval. LNCS 5280, Berlin, Heidelberg:

© 中国科学院软件研究所 http://www.jos.org.cn

Springer-Verlag, 2009. 87−98. http://books.google.com/books?id=_GPQj8tOvScC&printsec=frontcover&dq=15th+Symp.+on+String+Processing+and+Information+Retrieval&source=bl&ots=yPUQB8iXl7&sig=efcGCEezue9bWc7nBUEffdCsn2g&hl=en&ei=bPT0S5b_HNGgkQXl9PjFCA&sa=X&oi=book_result&ct=result&resnum=2&ved=0CBUQ6AEwAQ#

[16] Lin PC, Lin ZX, Lin YD, Lai YC, Lin FC. Profiling and accelerating string matching algorithms in three network content security applications. IEEE Communications Surveys and Tutorials, 2006,8(1-4):24−37. [doi: 10.1109/COMST.2006.315851]

[17] http://www.shareware.cn/infoView/Article_313.html. 2004.

[18] Aho AV, Corasick MJ. Efficient string matching: An aid to bibliographic search. Communications of the ACM, 1975,18(6): 333−340. [doi: 10.1145/360825.360855]

[19] Erdogan O, Pei C. Hash-AV: Fast virus signature scanning by cache-resident filters. In: Miller A, ed. Proc. of the IEEE GLOBECOM 2005. St Louis, 2005. http://books.google.com/books?id=rw9WAAAAMAAJ&q=IEEE+GLOBECOM+2005&dq=IEEE+GLOBECOM+2005&lr=&cd=12

[20] http://www.snort.org/dl/

[21] http://www.clamav.net/binary.html#pagestart

[22] http://www.ll.mit.edu/IST/ideval/

附中文参考文献:

[9] 张鑫,谭建龙,程学旗.一种改进的 Wu-Manber 多关键词匹配算法.计算机应用,2003,23(7):29−35.

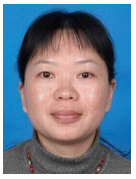
[10] 宋华,戴一齐.一种用于内容过滤和检测的快速多关键词识别算法.计算机研究与发展,2004,41(6):940−948.

[11] 孙晓山,王强,关毅,王晓龙.一种改进的 Wu-Manber 多模式串匹配算法及应用.中文信息学报,2006,20(2):47−53.

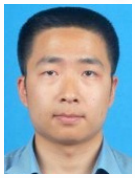
[12] 李伟男,鄂跃鹏,葛敬国,钱华林.多模式匹配算法及硬件实现.软件学报,2006,17(12):2403−2415. http://www.jos.org.cn/1000-9825/17/2403.htm [doi: 10.1360/jos172403]

[14] 程伟,刘玉军,卢泽新.最佳比较序字符串匹配算法研究和应用.计算机工程与设计,2004,25(9):1430−1436.

刘萍(1972－),女,山东济南人,助理研究员,主要研究领域为模式串匹配技术,算法设计,信息内容安全.



郭莉(1969－),女,正研级高工,CCF 会员,主要研究领域为网络安全,信息安全.



刘燕兵(1981－),男,博士生,助理研究员,主要研究领域为算法设计,模式串匹配技术,信息内容安全.

方滨兴(1960－),男,博士,研究员,博士生导师,中国工程院院士,CCF 高级会员,主要研究领域为计算机网络与信息安全,并行处理技术.

© 中国科学院软件研究所 http://www.jos.org.cn