网络信息审计系统中的多模式相似匹配算法*

高 鹏+, 张德运, 孙钦东, 翟亚辉, 卢伍春

(西安交通大学 电子与信息工程学院,陕西 西安 710049)

A Multiple Approximate String Matching Algorithm of Network Information Audit System

GAO Peng+, ZHANG De-Yun, SUN Qin-Dong, ZHAI Ya-Hui, LU Wu-Chun

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

+ Corresponding author: Phn: +86-29-82668096, E-mail: g622@xanet.edu.cn, http://www.xjtu.edu.cn

Received 2003-05-21; Accepted 2003-09-26

Gao P, Zhang DY, Sun QD, Zhai YH, Lu WC. A multiple approximate string matching algorithm of network information audit system. *Journal of Software*, 2004,15(7):1074~1080.

http://www.jos.org.cn/1000-9825/15/1074.htm

Abstract: This paper shows a simple, efficient, and practical algorithm for locating all occurrences of a finite number of keywords in a char/Chinese character string allowing k chars inserting errors. The algorithm consists of constructing multiple finite state single-pattern matching machines from keywords and a state-driver applied to drive all finite state single-pattern matching machines, and then using the state-driver to process the text string in a single pass. Speed of the matching is independent of the amount of the inserting errors. Generally, the algorithms do not need to inspect every character of the string. They skip as many characters as possible by making full use of the information in matching failure and text window mechanism. This algorithm can be widely applied to network information auditing, database, information retrieval, and etc.

Key words: information audit; matching allowing errors; multiple approximate string match; finite state machine

摘 要: 针对网络信息审计系统的需要,提出一种新颖的基于 Episode 距离的快速多模式相似串匹配算法.该算法把模式串集合转换为多个有限自动机,然后利用模式串集合建立一个状态驱动器.依次用待匹配串的字符驱动状态驱动器,由状态驱动器驱动各个有限自动机,实现了中英文混合的允许插入错误的相似多模式匹配.该算法不需要匹配每个字符,能充分利用匹配过程中本次匹配不成功的信息并结合改进的文本窗机制,跳过尽可能多的字符;能够控制每个模式串的允许错误上限;匹配速度与允许插入的错误字符数 k 无关.该算法在信息审计、数据库、信息检索等领域有着广阔的应用前景.

关键词: 信息审计;允许错误的匹配;多模式相似匹配;有限自动机

* Supported by the National Security Fund from the Ministry of Information Industry of China under Grant No.2001-1-010 (国家信息产业部计算机网络与信息安全基金)

作者简介: 高鹏(1973－),男,陕西三原人,博士,主要研究领域为网络安全,算法设计;张德运(1941－),男,教授,博士生导师,主要研究领域为计算机网络,信息安全;孙钦东(1975－),男,博士生,主要研究领域为计算机网络;翟亚辉(1977－),女,硕士,主要研究领域为计算机网络;卢伍春(1976－),男,硕士,主要研究领域为计算机网络.

中图法分类号: TP393 文献标识码: A

网络信息审计系统的功能可简单描述为,在网络关键节点监听所有出入的数据包,解析重组后,对数据包中的文本按内容性质进行分类,根据分类结果给出报警并指导相应的设备进行屏蔽.文本分类需要统计各个关键字出现的频率,它是整个系统效率的瓶颈.随着应用的扩展,出现了干扰审计系统正确性的一些方法、技术,迫切需要网络信息审计系统能对关键字集合作快速多模式相似匹配.

多模式相似匹配是指,按照一定的相似标准,在串 $S_{0\ldots n-1}$ 中找出所有与模式串 $P_0,P_1,\ldots,P_{Q-1}$ 相似的子串[1].它是模式匹配领域中最近才被注意到的问题.讨论多模式相似匹配的文献很少[2],我们尚未发现涉及中英文混合的多模式相似匹配算法的文献.已有针对英文字符集的文献有:文献[1]利用 two-level hashing 提出了允许一个错误的多模式相似匹配算法;文献[3~5]对文献[1]进行了扩展,使之可以适应多个错误;文献[2]总结并提出了3种算法,分别基于并行比特串搜索状态向量、文本窗中的模式字符计数、对已有多模式精确匹配的扩展(分解模式串),实现了针对英文字符串的相似匹配.这些算法的错误数目不能大于模式串长度,不能控制每个模式串的允许错误上限,在允许的错误上限接近模式串长度时,匹配时间会急剧增加[5].中文一般采用双字节编码方式,一个字符由两个字节组成.已有多模式相似匹配算法允许在模式串的任意位置插入字符.表示一个中文字符的两个字节的中间不允许插入字符,在包含 h 个汉字的字符串中存在 h 个禁止插入位.已有多模式相似匹配技术会导致错误的匹配结果(特别是在包含非文本数据的网络包中).

针对实际应用的需要,本文提出中英文混合字符串的多模式快速相似匹配算法,利用匹配不成功信息和改进的文本窗机制对算法进行了改进.最后分析了该算法的复杂度,给出了相应的实验结果.

1 信息审计系统对多模式相似匹配的要求

串相似标准主要有:编辑距离[6](表示为 ed,允许插入/删除/替代错误)、海明距离(表示为 hd,允许替代错误)、Episode 距离[7](表示为 epd,允许插入错误).英文字符表小、词较长,存在自然分词边界,一定距离的删除/替代错误对词义的影响不大.中文字符表小、词较短(2~4 汉字),且没有明显的分词标准,删除/替代错误会导致显著的词义改变.为了在干扰信息审计系统的同时不影响语义的传递,信息发送方主要采用插入干扰字符的方法.在网络信息审计系统中,我们采用 Episode 距离作为相似标准.

模式相似不能保证其语义的相似,为了提高文本分类的准确率,审计系统要求对大部分关键字作精确匹配,对部分敏感关键字作相似匹配(一个关键字是否为敏感关键字可以由文本特征选择算法或经验确定).所以要求多模式相似匹配算法在一次匹配中能够控制每个模式串的允许错误上限.最后,在应用中发现,插入的干扰字符数目可能远大于模式串的长度,特别是在利用 HTML 标记作为干扰字符时.

2 基于 Episode 距离的多模式相似匹配算法

基于 Episode 距离的多模式相似匹配是指:对于给定模式串集合 P_0,P_1,P_{Q-1},待匹配串 S 和错误上限 k,若存在 P_i,T 是 P_i 的一个子串,$epd(P_i,T)<k$,则认为出现了一次相似匹配.本文中的字符有如下定义:中文字符占2字节,英文字符占1字节.所有模式串均来自大小为δ的字符集$\{a\}$.所有模式串中的字符组成$\{c\}$.

多模式相似匹配算法按照模式串集合构造多个有限状态机;按照字符在各个模式中出现的位置构造状态驱动器;依次用 $S_{0\ldots n-1}$ 中的字符驱动状态驱动器,由状态驱动器驱动各有限状态机,得到匹配结果.

状态机的构造如下:扫描模式串的每个字符 $Char_i$,构造状态转移表 $Ttable$,用来描述其在各个模式串中出现的情况.状态驱动器的构造如下:根据$\{c\}$建立可用字符完全 Hash 表 $Tchar(char_j)$,存储各个字符对应状态转移表的指针,没有对应状态转移表的字符的指针初始化为 $NULL$.建立模式串状态表 $Tpattern(P_i)$记录当前各个状态机的位置.

这里给出算法匹配函数的C语言描述(多模式相似匹配算法1),可以看出,算法最多允许的错误数是可以大于模式串长度的.

算法 1. 多模式相似匹配算法(**AMP1**).

```
/*状态转移表数据结构*/
item of Ttable={
        int keyID;//对应模式串的序号
        int end;//模式串的结尾标志
        int charid;//本字符模式串的位序号
 /*字符驱动状态驱动器数据结构*/
 Item of Tchar(char_j)={
       pointer of Ttable(char_j);
}
 /*模式串状态表数据结构*/
 Item of Tpattern(P_i)={
       int state;//对应状态机当前所在状态号
        int matchid;//用来记录本次算法匹配的结果
        int m_output;
        int startpos;//首字符匹配的位置
        int maxk;//本模式串的错误上限
}
 void AMP1(char *s,int datalen)
{
    int i,j,d,k,c,c_i;
    Ttable *titem;
    for(i=0;i<datalen,d=i;i++)
    {
      if(s[i]<127) c_i=s[i];
     else   c_i=s[i]*256+s[++i];
      if(Tchar(c_i==NULL)
      {
        for(j=0;j<Ttable(Tchar(c_i)).size;j++)
        {
           titem=Ttable(Tchar(c_i)).item[j];
            k=titem->keyid;
            c=titem->charid;
            if(c==(Tpattern(k).state−1))
            {
               Tpattern(k).state++;
               if(c==0) Tpattern(k).startpos=d;
               if(titem->end && (d−Tpattern(k).startpos<=Tpattern(k).maxerr))
                {
                  Tpattern(k).m_output++;
                  Tpattern(k).state=1;
                }
            }
        }
      }
    }
 }
```

3 利用 QS 算法原理对相似匹配算法的改进

QS 算法的基本思想是利用本次匹配不成功的信息,尽可能多地跳过字符.当 $P_{1\cdots m}$ 与 $S_{i\cdots i+m-1}$ 对齐进行匹配时,如果匹配失败,则分析 S_{i+m} 这个字符,以决定 P 右移的距离 $skip1(S_{i+m})$.在多模式精确匹配中,$skip1(char)$的计算如下[8],其中 minlen 是模式串集合中最短模式串的字符长度,P_{kj} 表示第 k 个模式串的 j 个字符:

$$skip1(char)=\begin{cases}\min\{\min len-j+1\mid P_{kj}=char,1\le j\le \min len,0\le k\le Q-1, & char在\{P\}中\\ \min len+1, & char不在\{P\}中\end{cases} \tag{1}$$

在相似匹配中,*skip*1(*char*)的计算与最短匹配长度以及允许最大插入字符数目 k 相关.一个模式串被插入 k 个字符后被分为多个片断,这些片断中最长片断的长度可称为最大模式片断长度 *SP*.

$$SP = \text{int}(\min len / k) + 1 \tag{2}$$

定理 1. 在多模式跳跃匹配中,当匹配失败时,能保证最短模式串不被遗漏的最大移动距离为 *SP*.

证明:根据抽屉原则,*S*[1:*n*]中的模式串在插入了 k 个字符后,至少有一个片断的长度大于等于 *SP*.如果到来字符不在任何一个模式串中,向前跳跃 *SP* 的距离后,可保证至少会落在某个串的一个片断中. □

根据定理 1,在多模式相似匹配中,*skip*2(*char*)为

$$skip2(char) = \begin{cases} 1, & char\text{在}\{P\}\text{中} \\ SP+1, & char\text{不在}\{P\}\text{中} \end{cases} \tag{3}$$

为了保证算法的匹配速度,*skip*2 采用完全 Hash 表存储.以下是跳跃匹配的的多模式相似算法 C 语言描述.

算法 2. 中英文混合的多模式相似匹配算法(**AMP2**).

```
void AMP2(char*s,int datalen)
{
   int i,j,d,k,q,c,ci,f;lag=FALSE;
   Ttable *titem;
   for(i=0;i<datalen,d=i;i++)
   {
      if(s[i]<127) ci=s[i];
     else   ci=s[i]*256+s[++i];
      if(Tchar(ci)!=NULL)
      {
        if(flag)
        {
           i=q ;
           flag=False
        }
       …….//和算法 1 相同
      }
      else
      {
        q=i;
        flag=TRUE;
         i=i+skip2(ci);
      }
   }
}
```

4 利用文本窗机制对相似匹配算法的改进

已有的文本窗机制多采用编辑距离.它首先依次统计一定长度的 S 的子串中是否包含足够多的模式串中的字符,根据以下定理判断是否可能发生了一次匹配,然后再用常规的算法对可能的匹配进行验证[2].

文献[2]指出,如果δ为字符集的大小,在中等错误率的环境中两个串的相似距离小于 k 的概率为

$$f(ed(a,b)<k)=O(\gamma^m), length(a)=length(a)=m, \alpha=k/m.$$

$$\gamma = \left(\frac{1}{\delta\alpha^{\frac{2\alpha}{1-\alpha}}(1-\alpha)^2}\right)^{1-\alpha} \le \left(\frac{e^2}{\delta(1-\alpha)^2}\right)^{1-\alpha} \tag{4}$$

可见,在中文环境中(δ=65 535),验证的代价基本可以忽略.

引理 1. 假设 $i<j, ed(T_{1..j},P)<k, length(P)=m$,则 $T_{j-m+1..j}$ 至少应包含 $m-k$ 个 P 中的字符[9].

本文采用 Episode 距离,得到以下定理,确定文本窗的大小 winsize 和滑动规则.

定理 2. 假设 $P, T_{1..j}, epd(T_{1..j},P)<k$,则 $T_{j-m-k+1..j}$ 中包含 m 个 P 中的字符.

证明:因为 $epd(T_{i..j},P)<k$,则在 P 中最多插入 k–1 个字符后形成的串等于 $T_{i,j}$,所以文本串 T_j 开始向前的 $m+k$–1 个字符所形成的子串必然包含所有 P 中的字符.

对于英文而言,一个字符在同一关键字中、不同关键字中出现多次的概率较大,需要为每一个模式串维护一个文本窗口,匹配每个字符要更新所用窗口,其效率受到影响[2].中文同一字符在同一关键字中、不同关键字中出现多次的概率很小,所以本文为所有模式串建立一个文本窗口,根据定理 2,其大小为 max*len*+*k*;max*len*/min*len* 分别表示最大/最小的模式串长度,*k* 表示所用模式串允许错误上限的最大值.已有的文本窗算法采用一个固定的数组记录前 *winsize*–1 个字符的匹配情况,窗口滑动后,必须遍历这个数组以获得统计结果,我们采用改进的基于循环链表的文本窗算法(如图 1 所示),每次仅需要根据当前字符的检查 Hash 表的结果来更新尾指针的表项,避免了遍历,提高了效率.

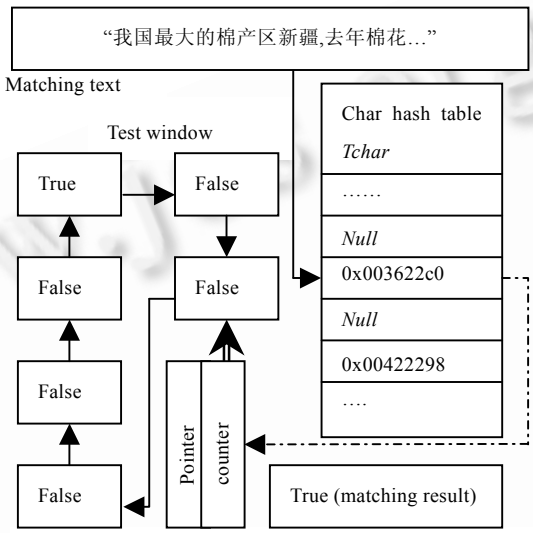

Fig.1 Illustration of the text window algorithm

图 1 文本窗算法图示

以下是基于文本窗扩展的多模式相似匹配算法C语言描述.

算法 3. **中英文混合的多模式相似匹配算法(AMP3)**.

```
void AMP3(char*s,int datalen,int ws,int mink;,textW textw)
{
   int i, ci,countk=0;
   for(i=0;i<datalen,d=i;i++)
   {
     if(s[i]<127) ci=s[i];
     else   ci=s[i]*256+s[++i];
     if(Tchar(ci)!=NULL)
     countk++;
        if(i==(winsize+textw.lastmatch))
        {
           countk--;
           lastmatchpos=updatelastmpos(textw);
        }
            if(countk==mink)
            AMP2(textw.lastmatch,winsize);
   }
}
```

5 算法复杂度分析

假设模式串的总个数为 Q,第 i 个模式串的长度为 Len_i,所有模式串所包含的无重复的字符集合$\{c\}$大小为 C,均来自大小为δ的字母表$\{a\}$;每个字符出现在所有模式串中的次数为 PC_i,平均为 PC,则构造所用状态机部分的算法时间复杂度为 $O(\Sigma Len_i)$,空间复杂度为 $O(\Sigma Len_i)$;构造状态机驱动器的算法时间复杂度为 $O(C+\Sigma PC_i)$,空间复杂度为 $O(\delta+\Sigma PC_i)$,因为使用双子节字符完全 Hash 表,所以,这里δ=64K.

在匹配过程中,假设待匹配串的长度为 n.在最差情况下,待匹配串中的所有字符都有对应的状态转移表,每个状态转移表大小为 $\mathrm{Max}(Len_i)$,则算法时间复杂度为 $O(n\times\mathrm{Max}(len_i))$.在最优情况下,算法复杂度为 $O(n/SP)$.假设所有串中各字符是等概率出现的,则在长度为 n 的串中,需要进行遍历状态转移表的字符有 $n\times C/\delta$个,每次操作需要遍历平均 PC 个表项;不需要遍历的字符为 $n\times(\delta-C)/\delta$个,因为这里进行跳跃匹配,只需要进行 $n\times\delta-C/(\delta\times SP)$.平均算法复杂度为 $O(n\times C\times PC/\delta+n\times(\delta-C)/(\delta\times SP))$.因为$\delta$=64K,一般情况下,$C<<\delta$,在大部分的字符中都是跳跃匹配的,所以本算法实际上是很有效的.

在加入文本窗机制后,对于每个字符都要先进行一次记数并判断是否在窗口范围内,然后对可能的匹配进行验证.因为本算法对所用模式串仅使用一个窗口,计算平均复杂度需要更为复杂的模型,这里不再讨论.

根据算法的特点,算法的匹配速度与下面 3 个因素有关:

(1) $\{c\}$中每个字符在待匹配串中出现的概率.显然,出现的概率越低,移动 SP 距离的概率越大,匹配时间越少;文本窗发现可能匹配的概率越低,算法所需要的整体时间越趋近于文本窗记数的时间.而改进的文本窗统计算法采用 Hash 表实现,时间复杂度在最差情况下为 $O(n)$,与模式串数目无关.

(2) 中文、英文在模式串中各占有的比例.中文语言是大字符集语言,它具有两个特点:① δ值较大,② 中文词长度较短(平均词为 1.83~2.09)[8].中文语言的这两个特点决定了:① 大多数情况下,来到的中文字符都不属于$\{c\}$,所以移动 SP 距离的概率比较大;② 每个中文字符对应的状态转移表都比较小,即使当前字符属于$\{c\}$,需要遍历的次数也很少.英文字符的特点是:① 字母表小;② 词比较长.英文语言的这两个特点决定了:① 到来字符属于$\{c\}$的概率高于中文的情况;② 转移表比较大.所以,当中文字符较多时,匹配速度较快.

(3) $\{c\}$中每个字符在每个模式串中出现的次数.出现次数越少,对应的状态转移表越小,遍历越快,平均算法复杂度越低.

6 实验与结果

实验使用 PIV 2G/128M 的 PC 机,采用 VC6.0 实现算法.待测文本来自 1998 年人民日报文本,共 6 494 931 字节,然后根据已有的分词标注从中切分出 3 000 个 2 字汉语词作为模式串集合.模式串集合中不存在包含关系(一个模式串是另一个的子串).在待测文本的前 1 000 个模式串子串中的随机位置上插入了 2 000 个干扰字符,用于模拟针对网络信息审计系统的干扰.在文本后追加了 windows2000 系统目录 system 下的 27 个二进制文件,模拟网络中的实际包内容,待测文本总长度为 8.4M 字节.

表 1 是各种算法的匹配时间比较,其中 Counting filter 来自文献[2](为了有可比性,我们改变了该算法的终止条件:匹配整个文本串,而不是在匹配到任一模式串后即停止).算法 2、算法 3 的允许错误上限等于模式串的长度,Counting filter 算法的允许错误上限设为 2,可以看出,算法 2、算法 3 均优于已有算法.特别是算法 3,当关键字数目从 200 个增加到 1 000 个时,其匹配时间基本与模式串多少无关.

Table 1 Comparison of performance of algorithms under various keyword numbers

表 1 不同数目关键字的各种算法性能对比

Number of keywords	CPU time of counting filter algorithm	CPU time of algorithm 2	CPU time of algorithm 3	Matched times
200	0.83	0.5	0.3	213 371
400	1.17	0.56	0.3	347 907
600	1.40	0.62	0.3	484 935
800	1.62	0.64	0.32	591 615
1 000	1.84	0.66	0.32	705 471

在表 1 中,当模式串为 200~600 个、800~1000 个时,算法 2、算法 3 出现了所用时间变化很小的现象.这是

因为,增加的模式串所包含的字符已在前面模式串中出现的概率增大,匹配过程中查表的次数增长较慢.

最后需要指出,Counting filter 算法在每次实验中,均出现了多次的误匹配现象.这些误匹配现象主要集中于前 6.4M 文本串中,后 2M 的二进制文件中也有出现.

7 结 语

本文分析了网络信息审计系统对快速多模式相似匹配算法的要求,提出了一种基于 Episode 距离的多模式相似串匹配技术.该算法把所有模式串转换为多个有限自动机;利用模式串建立一个状态驱动器;依次用待匹配串的字符驱动状态驱动器,再由状态驱动器驱动各个有限自动机.该算法有如下特点:

(1) 实现了基于 Episode 距离的多模式中英文相似匹配算法,允许错误 k 的上限由待匹配串的长度可以远大于模式串的长度;

(2) 利用文本窗机制加速匹配过程,并采用循环链表进行文本窗的更新;

(3) 充分利用本次匹配不成功的信息,在多模式相似匹配中实现了跳跃匹配;

(4) 在一次多模式匹配中,可以针对不同的模式串设定不同的允许错误上限.

该算法的时间复杂度与允许错误字符数目 k 无关,最优情况下为 $O(n/PS)$.实验显示,该算法控制灵活、有着良好的性能.在网络信息审计、数据库、文本分类和基因检测等系统中有着广阔的应用前景.本文的下一步工作是利用多个处理器实现多模式的并行相似匹配算法,以获得更快的匹配速度.

References:

[1] Muth R, Manber U. Approximate multiple string match. In: Hirschberg D, Meyers G, eds. Proc. of the CPM'96. LNCS 1075, Springer-Verlag, 1996. 75~86.

[2] Ricardo B-Y, Gonzalo N. New and faster filters for multiple approximate string matching. Random Structures and Algorithms, 2002,20(1):23~49.

[3] Gonzalo N, Ricardo B-Y. Fast multidimensional approximate pattern matching. In: Crochemore M, Paterson M, eds. Proc. of the CPM'99. LNCS 1645, Springer-Verlag, 1999. 243~257.

[4] Ricardo B-Y. Regnier M. Fast algorithms for two dimensional and multiple pattern matching. In: JR Gilbert BR, Karlsson R, eds. Proc. of the SWAT'90. LNCS 447, Springer-Verlag, 1990. 332~347.

[5] Gonzalo N. Multiple approximate string matching by counting. In: Proc. of the WSP'97. Carleton University Press, 1997. 125~139.

[6] Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl, 1966. 707~710.

[7] Das G, Fleisher R, Gasieniek L, Gunopulos D, Karkainen J. Episode matching. In: Apostolico A, Hein J, eds. Proc. of the CPM'97. LNCS 1264, Springer-Verlag, 1997. 12~27.

[8] Xu YZ, Wang YC, Shen Z. A fast algorithm for matching multiple patterns. Journal of Shanghai Jiaotong University, 2002,36(4):516~520 (in Chinese with English abstract).

[9] Ukkonen E. Approximate string matching with q-grams and maximal matches. Theoretical Computer Science, 1992,1:191~211.

附中文参考文献:

[8] 许一震,王永成,沈洲.一种快速的多模式字符串匹配算法.上海交通大学学报,2002,36(4):516~520.