文章编号:1001 - 9081(2009)02 - 0424 - 03

一种改进的编辑距离算法及其在数据处理中的应用

赵作鹏[1,2],尹志民[3],王潜平[2],许新征[2],江海峰[2]
(1. 北京大学 遥感与地理信息系统研究所,北京 100871; 2. 中国矿业大学 计算机科学与技术学院,江苏 徐州 221116;
3. 冀中能源集团股份有限公司,河北 邢台 054000)
(cumt. pku@163. com)

摘 要:基于数据处理的需要,在分析原有编辑距离算法的基础上,通过拓展交换操作减少编辑操作的数量。与仅对计算点之前相邻位置字符间的交换操作相比,通过对计算点前后非相邻位置字符间的交换操作改进该算法,能够得到更理想化的编辑距离。将改进的编辑距离算法应用于煤矿隐患数据的处理,提高了隐患数据分类分级的有效性和执行效率。

关键词:编辑距离;字符串相似匹配;数据处理
中图分类号: TP301.6 **文献标志码**:A

An improved algorithm of Levenshtein Distance and its application in data processing

ZHAO Zuo-peng[1,2], YIN Zhi-min[3], WANG Qian-ping[2], XU Xin-zheng[2], JIANG Hai-feng[2]
(1. *Institute of Remote Sensing and GIS, Peking University, Beijing* 100871, *China*;
2. *School of Computer Science and Technology, China University of Ming and Technology, Xuzhou Jiangsu* 221116, *China*;
3. *Jizhong Energy Group Company Limited, Xingtai Hebei* 054000, *China*)

Abstract: Based on the requirement of data processing, after analyzing the existing algorithm of Levenshtein Distance, the number of edit operation was decreased by extending the transposition operation. Compared to the existing the algorithm that can only transpose adjacent symbols before the position of being computed, improving the algorithm by transposing isolated symbols before the position of being computed as well as at after the position of being computed, can gain better edit distance. By applying the improved algorithm to the processing of the hidden trouble data of coal mine, both the validity and efficiency of classifying and grading the hidden trouble data were improved.

Key words: Levenshtein Distance; approximate string matching; data processing

0 引言

煤矿隐患排查管理从隐患级别上,分为重大危险源管理、重大隐患管理、一般隐患管理等。从隐患类别上,又分为顶板、机电、一通三防等众多类别。不同类别和级别的隐患,有不同的处理方式。没有隐患类别和级别的准确判定,不仅无法进行隐患的及时排查,而且无法进行隐患数据的分类分级的统计和安全事故的预测预防。

然而,对于隐患排查系统,每天都有大量的相关数据进入系统。如果采取人工分类分级,一方面,对隐患类别、级别的判定,需要花费很多时间;另一方面,出于主观和客观原因,人工对隐患判定,差错的几率高。为解决以上问题,研究了基于编辑距离的隐患数据自动分类分级处理系统。系统先根据安全规范建立隐患库(记录数量根据安全规范和要求而定),隐患库中的每条隐患记录都有特定的类别和级别。然后利用基于编辑距离的字符串相似匹配技术,计算每次采集到的隐患数据与隐患库中的记录的相似度,取相似度大的记录的类别和级别作为本次采集到的隐患数据的类别和级别。其中的关键是高效的字符串相似匹配技术。

1 字符串相似匹配技术

1.1 字符串相似匹配技术

现有的计算字符串相似度的方法按照计算所依据的特征的不同,可以划分为 3 种方法:基于字面相似的方法、基于统计关联的方法、基于语义相似的方法,以及考虑 3 种方法的多层特征方法[1-3]。其中,基于字面相似的计算方法主要有基于编辑距离的计算方法[4]和基于相同字或词的方法[5]。其中,编辑距离法应用广泛,计算方法相对成熟。近期关于字符串相似匹配技术的研究[6-8]主要集中在多处理器并行处理等方面。

作为汉字字符串相似匹配技术的应用,本隐患排查系统采用改进的编辑距离算法来实现隐患信息的自动分类和分级。

1.2 编辑距离算法

编辑距离(Levenshtein Distance)由 Levenshtein 于 1966 年在文献[9]中提出,通过编辑距离计算源字符串 S 与目标字符串 T 相似度。编辑距离是指由 S 变化到 T 所需要的最小编辑操作的数量,Levenshtein 所提出的编辑操作是指对字符串的某一个位置的字符进行删除、插入、替换的操作。文献[10]对编辑操作进行了扩展,增加了两相邻位置的字符间的交换操作,

收稿日期:2008 - 08 - 13;**修回日期**:2008 - 10 - 08。
基金项目:中国矿业大学青年科研基金资助项目(2007A044);江苏省自然科学基金资助项目(BK2006039)。
作者简介:赵作鹏(1977 -),男,江苏徐州人,讲师,博士研究生,主要研究方向:矿山数字化; 尹志民(1956 -),男,河南濮阳人,正高级工程师,主要研究方向:矿山安全; 王潜平(1964 -),男,安徽潜山人,教授,博士,主要研究方向:CSCW、数据库、WSN; 许新征(1980 -),男,安徽萧县人,讲师,博士研究生,主要研究方向:人工智能; 江海峰(1979 -),男,江苏徐州人,讲师,博士研究生,主要研究方向:嵌入式系统、宽带网络。

以实现编辑操作的最小化。基于编辑距离算法进行动态编程,其算法的时间复杂度为 $O(mn)$,空间复杂度为 $O(mn)$,如果编辑顺序不需要保存的话,空间复杂度为 $O(\min(m,n))$,m,n 分别表示源字符串 S 和目标字符串 T 的长度。

编辑距离 $D(S,T)$ 的计算方法如下所述。首先假设 $\boldsymbol{D}_{ij}=\boldsymbol{D}(s_1\cdots s_i,t_1\cdots t_j)$,$0\leqslant i\leqslant m$,$0\leqslant j\leqslant n$,$\boldsymbol{D}_{ij}$ 表示从 $s_1\cdots s_i$ 到 $t_1\cdots t_j$ 的编辑距离,那么 $(m+1)\times(n+1)$ 阶矩阵 $\boldsymbol{D}_{ij}$ 可通过式(1)计算得到:

$$\boldsymbol{D}_{ij}=\begin{cases}0, & i=0\text{ 且 }j=0\\ \operatorname{Min}\begin{cases}\boldsymbol{D}_{i-1,j-1}+(\text{if } s_i=t_j \text{ then } 0 \text{ else } 1)\\ \boldsymbol{D}_{i-1,j}+1\\ \boldsymbol{D}_{i,j-1}+1\\ \text{if } s_{i-1}s_i=t_jt_{j-1} \text{ then } \boldsymbol{D}_{i-2,j-2}+1\end{cases} & i>0\text{ 或 }j>0\end{cases},\quad(1)$$

式(1)包含删除、插入、替换、交换四种操作,其中交换操作对计算点(i,j)之前相邻位置字符间的交换。该算法是从两个字符串的左边开始比较,记录已经比较过的编辑距离,然后进一步得到下一个字符位置时的编辑距离。矩阵 $\boldsymbol{D}_{ij}$ 能够通过从 $\boldsymbol{D}_{00}$ 逐行逐列计算获取,最终 $\boldsymbol{D}_{mn}$ 表示 $D(S,T)$ 的值,即 S 和 T 的编辑距离。

编辑距离越大,相似度越小。假设源字符串 S 与目标字符串 T 长度的最大值为 $L_{\max}$,编辑距离为 LD,相似度为 SI,那么 $SI=1-LD/L_{\max}$。

2　改进的编辑距离算法

式(1)包含了删除、插入、替换、交换的编辑操作,在大多数情况下,可以有效地计算中英文字符串间的相似度。但中文的某些表达方式,使用式(1)计算时,却得出错误的结果。例如 S = "老师您好",T = "您好老师",使用式(1)计算时,编辑距离为4,相似度为0。实际上 S 和 T 表达的是相同的意思。因而在这种情况下,式(1)不再适用,需要进一步改进。

从式(1)可以看出,交换操作的两个前提条件是:1)对相邻位置字符间的交换操作;2)对计算点之前的字符间的交换。即:if $s_{i-1}s_i=t_jt_{j-1}$ then $\boldsymbol{D}_{ij}=\boldsymbol{D}_{i-2,j-2}+1$。但非相邻位置的字符,计算点之前与之后的字符,都存在交换的操作,即:

1)if $s_{i-k}\cdots s_i=t_j\cdots t_{j-k}$ then $\boldsymbol{D}_{ij}=\boldsymbol{D}_{i-k-1,j-k-1}+1$

2)if $s_{i-k}\cdots s_i=t_j\cdots t_{j+k}$then $\boldsymbol{D}_{ij}=\boldsymbol{D}_{i-k-1,j-k-1}+1$,$0\leqslant k\leqslant\min(m,n)$。

经过对交换操作的拓展,可以得到$(m+1)\times(n+1)$阶矩阵$(\boldsymbol{D}_{ij})$的计算公式如式(2)所示(式中的斜体部分为改进内容):

$$\boldsymbol{D}_{ij}=\begin{cases}0, & i=0\text{ 且 }j=0\\ \operatorname{Min}\begin{cases}\boldsymbol{D}_{i-1,j-1}+(\text{if } s_i=t_j \text{ then } 0 \text{ else } 1)\\ \boldsymbol{D}_{i-1,j}+1\\ \boldsymbol{D}_{i,j-1}+1\\ \text{if } s_{i-k}\cdots s_i=t_j\cdots t_{j-k} \text{ then } \boldsymbol{D}_{i-k-1,j-k-1}+1\\ \text{if } s_{i-k}\cdots s_i=t_j\cdots t_{j+k} \text{ then } \boldsymbol{D}_{i-k-1,j--k-1}+1\end{cases} & 1\leqslant k\leqslant\min(m,n),i>0\text{ 或 }j>0\end{cases},\quad(2)$$

经过对交换操作的扩展,从式(2)可以清楚地看到,当 $k=1$ 时,交换操作就变成相邻位置的交换操作,因而式(1)可以看作式(2)的特例。通过对交换操作的再扩展,进一步减少了编辑操作,得到了更理想的编辑距离。当 $s_{i-k}\cdots s_i=t_j\cdots t_{j-k}$ 或 $s_{i-k}\cdots s_i=t_j\cdots t_{j+k}$ 出现频率较高时,更能体现出式(2)的优势。例如S = "*abcdefgh*",T = "*dbcahfge*",采用式(1)计算时时,编辑距离为4,采用式(2)计算时,编辑距离为2。对之前的例子,S = "老师您好",T = "您好老师",采用式(1)计算时,编辑距离为4,相似度为0;采用式(2)计算时,编辑距离为1,相似度为0.75,能准确地反映字符串间的相似度。

相对于式(1)的时间复杂度:$O(mn)$,空间复杂度:$O(\min(m,n))$,式(2)的空间复杂度不变,仍为 $O(\min(m,n))$,但在时间复杂度方面,式(2)在提高结果的准确度的同时,也增加了时间复杂度,由式(2)可以看出,时间复杂度为:$O(mn\times\min(m,n))$。但在很多情况下,可以一定程度地牺牲时间复杂度来换取结果的准确度,而且在某些特定的应用中,可以提高系统整体的执行效率。以下应用实例就是很好的证明。

3　隐患数据自动分类分级处理的实现过程

隐患自动分类分级算法流程如图1所示,其中:$KDATA_i$ 表示隐患库的某一条记录;$Ksort_i$ 和 $Klevel_i$ 分别表示记录 $KDATA_i$ 的类别字段和级别字段;N_i 表示输入的隐患数据 $SDATA$ 与该条记录 $KDATA_i$ 的相似度;$N_{\max}$ 表示相似度的最大值;$Ksort$ 和 $Klevel$ 分别表示最终要求的隐患数据 $SDATA$ 的类别和级别;"->"表示将该符号左边变量的值赋给右边的变量。

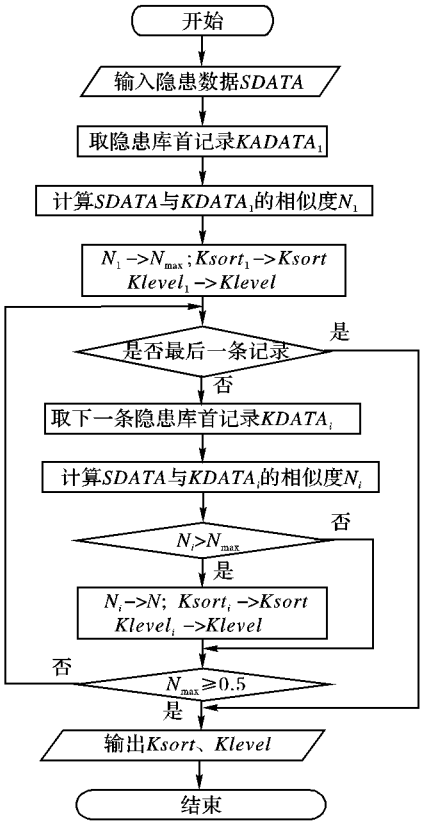

图1　自动分类分级算法流程

首先根据安全规范建立不同类别和级别的隐患库,记录数在几百到几千条之间,对每次采集的实际发生的隐患数据,采用改进的编辑距离算法,和隐患库中的每一条隐患进行相似匹配,并计算相似度。隐患库中相似度最大记录的类别和级别作为该次采集的隐患数据的类别和级别。

从图1可以看出,算法的关键在于字符串相似度的计算,决定了自动分类分级的准确率和效率。

隐患库中的隐患记录之间相对独立,因而为了提高程序运行效率,若 $SDATA$ 与 $KDATA_i$ 的相似度大于等于0.5,则不再继续和其他的隐患库中的记录进行相似度的计算,直接取 $KDATA_i$ 的类别和级别。因此,平均遍历隐患库的记录数,仅为总记录数的1/2。

此外,相对于式(1),利用式(2)能够得到更小的编辑距离,即更大的相似度,因而能够更快地获得大于等于0.5的相

似度,因而效率更高。其次,对于很多意义相同但表达顺序不同的情况下,利用式(2)能得到更准确的相似度。

4 应用实例

4.1 应用情况

利用本改进算法实现的隐患排查管理系统,应用于冀中能源集团股份有限公司某矿。隐患排查管理系统中,*SDATA*(待分类分级的隐患数据)与 $KDATA_i$(隐患库中的模式数据)的长度在5~20个汉字之间,$KDATA_i$ 数量为600条。下面通过一个具体的例子来说明本改进算法的优势。

采集的隐患数据 *SDATA* = "防尘喷雾未使用";隐患库中第18条记录内容为 $KDATA_{18}$ = "未使用防尘喷雾"。应用式(1)计算得到的 $\boldsymbol{D}_{ij}(i \leqslant 7, j \leqslant 7)$ 矩阵如表1所示。

表1 应用公式1计算编辑距离

		未	使	用	防	尘	喷	雾
	0	1	2	3	4	5	6	7
防	1	1	2	3	3	4	5	6
尘	2	2	2	3	4	3	4	5
喷	3	3	3	3	4	4	3	4
雾	4	4	4	4	4	5	4	3
未	5	4	5	5	5	5	5	4
使	6	5	4	5	6	6	6	5
用	7	6	5	4	5	6	7	6

最终得出的编辑距离 $\boldsymbol{D}_{77} = 6$,相似度为(1 - 6/7)约为0.14,因为值小0.5,需要遍历下面的记录进行计算,因而不仅效率低而且会得出不正确的结果,严重影响安全生产。使用式(2)计算得到的 $\boldsymbol{D}_{ij}$ 矩阵如表2所示。

表2 应用公式2计算编辑距离(1)

		未	使	用	防	尘	喷	雾
	0	1	2	3	4	5	6	7
防	1	1	2	3	3	4	5	6
尘	2	2	2	3	4	3	4	5
喷	3	3	3	3	4	4	3	4
雾	4	4	4	1				
未	5	4	5					
使	6	5	4					
用	7	6	5					

表3 应用公式2计算编辑距离(2)

	未	使	用	防	尘	喷	雾
未							
使							
用							
防			1	1	2	3	4
尘			2	2	1	2	3
喷			3	3	2	1	2
雾			4	4	3	2	1

当计算到 $\boldsymbol{D}_{34}$ 时,因为 $KDATA_{1\cdots3} = SDATA_{4\cdots7}$,*SDATA* 执行交换操作,由"防尘喷雾未使用"交换变为:"未使用防尘喷雾",然后继续计算后续的 $\boldsymbol{D}_{ij}(i, j > 3)$,如表3所示。最终得出的编辑距离 $\boldsymbol{D}_{77} = 1$,相似度为(1 - 1/7)约为0.86,大于0.5,不再需要遍历其他的记录,高效地得出正确的结果。

通过100条隐患数据实验,结果表明:用本改进的编辑距离算法,自动分类分级的正确率达到96%,平均执行速度提高20%。充分证明,改进的算法有力地提高了隐患数据分类分级的有效性和执行效率。

4.2 提高编辑距离算法执行效率的其他关键技术

由于本隐患排查系统是基于Web的实现方式,所以响应速度尤为关键。为了提高编辑距离算法的运行效率,首先,算法利用触发器和存储过程实现。一旦有新的隐患数据采集进入数据库时,触发器立即触发计算相似度的存储过程。经过实验证明,平均执行时间可以缩小10%~20%。其次 $KDATA_i$ 要求尽可能简短,使得编辑算法中二维数组LD其元素少,运算次数减少。因而创建隐患库时,尽可能以短句或关键字取代长句。再次,为了在速度和匹配需求方面达到平衡,当硬件采用中低端服务器时,隐患库的记录数控制在600条左右以保持足够快的响应速度。最后,因为中文表达中,$s_{i-k}\cdots s_i = t_j\cdots t_{j+k}$ 的情况较多,例如"老师您好"与"您好老师",但 $s_{i-k}\cdots s_i = t_j\cdots t_{j-k}$ 的情况较少,因而为了提高算法的执行效率,在处理中文字符串时,可以不考虑 $s_{i-k}\cdots s_i = t_j\cdots t_{j-k}$ 的情况,所以式(2)可进一步地简化为式(3),以便简化算法。

$$\boldsymbol{D}_{ij} = \begin{cases} 0, & i = 0 \text{ 且 } j = 0 \\ \mathrm{Min}\begin{cases} \boldsymbol{D}_{i-1,j-1} + (\text{if } s_i = t_j \text{ then } 0 \text{ else } 1) \\ \boldsymbol{D}_{i-1,j} + 1 \\ \boldsymbol{D}_{i,j-1} + 1 \\ \text{if } s_{i-k}\cdots s_i = t_j\cdots t_{j-k} \text{ then } \boldsymbol{D}_{i-k-1,j-k-1} + 1 \end{cases} \\ 1 \leqslant k \leqslant \min(m,n), i > 0 \text{ 或 } j > 0 \end{cases} \quad (3)$$

5 结语

通过扩展计算点前后非相邻字符间的交换操作,改进了编辑距离算法,实现了编辑操作的最小化。在隐患数据的自动分类分级系统中,利用本算法对本隐患数据处理有更高的执行效率、提高了准确性。使得隐患自动分类分级系统可以更好地满足现场需要,系统经过半年的现场运行表明,不仅降低了矿井安全生产事故发生的概率,而且至少可以每月节约企业目标生产成本的5~10%。

参考文献:

[1] HALL P A V, DOWLING G R. Approximate string matching[J]. ACM Computer Survey, 1980, 12(4): 381-402.

[2] WAGNER R A, FISCHER M J. The string-to-string correction problem[J]. Journal of the ACM, 1973, 21(1): 168-173.

[3] 章成志. 基于多层特征的字符串相似度计算模型[J]. 情报学报, 2005, 24(6): 696-701.

[4] MONGE A E, ELKAN C P. The field matching problem: Algorithm and applications[EB/OL]. [2008-06-16]. http://www.cecs.csulb.edu/~monge/Papers/kdd96.ps.

[5] NIRENBURG S, DOMASHNEV C, GRANNES D J. Two approaches to matching in example-based machine translation[EB/OL]. [2008-06-16]. http://www.mt-archive.info/TMI-1993-Nirenburg.pdf.

[6] LIPSKY O, PORAT E. Approximate matching in the L_∞ metric[J]. Information Processing Letters, 2008, 105(4): 138-140.

[7] MICHAILIDIS P D, MARGARITIS K G. Processor array architectures for flexible approximate string matching[J]. Journal of Systems Architecture, 2008, 54(1-2): 35-54.

[8] 余建明, 徐波, 薛一波. 基于网络处理器的高速字符串匹配[J]. 清华大学学报: 自然科学版, 2008, 48(4): 590-592.

[9] LEVENSHTEIN V L. Binary codes capable of correcting deletions, insertions and reversals[J]. Doklady Akademii Nauk SSSR, 1966, 163(4): 707-710.

[10] LOWRANCE R, WAGNER R A. An extension of the string-to-string correction problem[J]. Journal of the ACM, 1975, 22(2): 177-183.