基于 AC 自动机和双数组 Trie 树的藏文多模式匹配算法

崔 晨[1] 彭 展[1,2,3]
CUI Chen PENG Zhan

摘 要 多模式匹配算法在网络舆情监管中发挥重要作用，这些算法可以有效地监测筛选出与特定事件相关的敏感词，进而可以快速地对事件做出响应，更好地应对网络舆情的动态变化。然而在处理涉藏网络舆情所涉及的海量藏文数据时，传统针对 ASCII 字符集的多模式匹配算法，常常无法满足性能要求。在这一背景下，文章针对藏文多模式匹配问题，提出一种适用于藏文字符集的多模式匹配算法——TAC-DAT 算法（tibetan Aho Corasick-double array trie），该算法将 AC 自动机（Aho Corasick automaton）与双数组 Trie 树（double array trie）结合，减少内存空间占用，同时利用藏文音节字之间以音节点为分隔的特点，优化自动机跳转过程，降低失败字符匹配次数，从而提升了匹配的效率。实验结果表明，该算法在藏文多模式匹配任务中表现出色，相较于传统多模式匹配算法性有明显提升。

关键词 AC 自动机；双数组 Trie 树；藏文处理；多模式匹配

doi：10.3969/j.issn.1672-9528.2025.01.026

0 引言

互联网的普及和广泛应用使得网络舆情传播速度变得极为迅速，信息传递的途径也愈加多样化和复杂化。与此同时，大量的藏文内容出现在网络上，如何迅速且精准地对这些藏文信息予以过滤、深入分析，直接关系到能否及时对涉藏网络舆情做出响应。模式串（字符串）匹配，作为网络安全领域的核心技术之一，可以将一个或多个模式串作为一个文本中的子串查找并返回其位置 [1]，同时作为一种基于规则的方法，其具有简洁、高效，易于编程实现的优点，尤其适用于敏感词检测这类不涉及复杂语义分析，但对响应时间极为敏感的任务。若将涉藏敏感词作为模式串，结合针对藏文的高效多模式匹配算法，可以更快速地在海量信息中过滤及定位敏感词，有助于及时监测处理涉藏突发事件或涉藏舆情热点，提升涉藏网络安全保障体系和能力 [2]。

1975 年，Aho 等人 [3] 提出了 AC 自动机算法，通过构建一个有限状态自动机，能够在一个文本中同时搜索多个模式串，并找出所有匹配的位置。将所有模式串存储在一个自动机中，构建成类似于 Trie 树的结构，当匹配失败时根据节点信息及时进行状态转移并继续匹配。但是 AC 自动机在存储大量模式串时会增大内存的占用，而双数组 Trie 树可以在保证 Trie 树搜索速度不变的同时提高空间利用率，内存占用更低 [4]。将两者进行结合可以使得 AC 自动机能够在降低内存占用的情况下实现快速的多模式匹配。为进一步提高两者结合之后对藏文的处理效率，本文利用藏文音节字以音节点分隔的语言特点提出一种基于 AC 自动机和双数组 Trie 树的藏文多模式匹配算法——TAC-DAT 算法。

1 研究现状

模式匹配算法的研究历史悠久，许多经典算法被广泛使用，如：1977 年，Knuth 等人 [5] 提出了著名的 KMP 算法，该算法通过利用已知部分匹配的信息，尽可能减少回溯次数来实现字符串匹配；同年，Boyer 等人 [6] 共同提出了 BM 算法，该算法利用模式串与文本串从右往左进行比较，并通过预处理模式串构建坏字符规则和好后缀规则来快速定位匹配位置；1980 年，Horspool 等人 [7] 提出了 BMH 算法，该算法对 BM 算法进行简化处理，在移动模式串时仅考虑了坏字符启发；1987 年，Karp 等人 [8] 共同提出 Rabin-Karp 算法，该算法对于给定的文本串与模式串利用哈希函数对字符串进行编码，通过比较哈希码来判断是否可能发生匹配，并在发生哈希碰撞时进一步验证字符串是否匹配；1990 年，Sunday[9] 提出了基于 BM 算法的一种改进简化的算法 Sunday 算法，该算法在发生不匹配时，通过查找文本串中下一个可能匹配的字符来确定模式串向右移动的距离，从而加速匹配过程；

1. 西藏民族大学信息工程学院 陕西咸阳 712082
2. 西藏自治区光信息处理与可视化技术重点实验室 陕西咸阳 712082
3. 西藏网络空间治理研究基地 陕西咸阳 712082
[基金项目] 西藏自治区自然科学基金项目“藏文模式匹配与文本索引关键技术研究”（XZ202101ZR0089G）

1994 年，Wu 等人 [10] 提出的 Wu-Manber 算法，该算法将所有模式串转换为一个哈希表，并在文本串中使用哈希表进行匹配。在匹配过程中，利用前缀数组来快速跳过不匹配的情况。

AC 自动机是一种高精度的多模式匹配算法，在各领域中都有广泛应用。然而，当模式串数量庞大时，AC 自动机的内存占用增加，匹配速度下降，性能降低。为了解决内存空间占用问题，研究者不断探索并提出多种内存空间紧缩技术。1984 年，Dencker 等人 [11] 提出将非空转移路径对应的字符和下一跳状态存储至状态转移表的每一行中，这种方法更适应于稀疏的状态转移表，压缩效果更为显著；1988 年，Aoe 等人 [12] 提出使用两个数组表示 Trie 树，有效结合了数字搜索树高效检索和 Trie 树链式表示的空间紧凑的特点，降低了内存占用；2015 年，熊刚等人 [13] 提出基于数据访问特征的混合自动机建构算法 HybridFA，对访问频率和层次两方面进行改善，存储空间较 AC 自动机降低 5%。

由于藏文字符编码、字体结构等方面差异，应用于藏文的模式匹配算法的研究相对落后于中文和英文，现有的藏文模式匹配算法有：TVM 算法，该算法利用藏文音节特点，通过元音字符查找匹配，有效提高藏文字符串匹配效率 [14]；TAC 算法，该算法有效利用藏文中音节点结尾的特点改进了 AC 自动机，有效提高匹配效率 [15]；BMT 算法，该算法针对藏文 Unicode 编码以及藏文音节特征，修改优化 BM 算法的匹配过程，能够获得较少的比较次数以及增大跳跃举例 [16]。因此，进一步加强对藏文模式匹配算法的研究十分必要。

2 藏文音节字结构介绍

藏文与汉语、英语相比属于拼音文字，包含 30 个辅音字母和 4 个元音字母，其结构为以一个基字为核心，其余字母作为上加字、下加字、前加字、后加字、再后加字和元音部分，与基字前后或者上下叠加构成完整的藏文音节字结构 [17]，藏文音节字结构如图 1 所示。

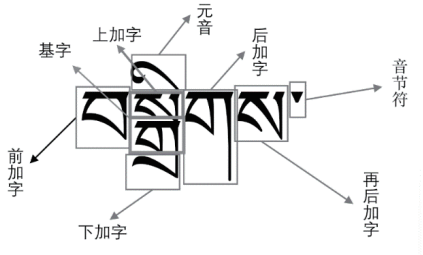

图 1 藏文音节字结构

一个藏文音节字最少情况下包含一个基字，最多情况下包含 6 个辅音字母和 1 个元音字母。在两个音节字之间使用分隔符“·”分隔，此分隔符也称为音节点 [18]，计算机在读入藏文音节字后会将音节字拆分成一系列字符进行处理。若将传统的多模式匹配算法直接应用到藏文，当匹配失败时会继续对当前音节字剩余字符进行匹配降低了匹配速率，本文针对每一音节字以音节点为结尾这一特点，在 AC 自动机与双数组 Trie 树结合的基础上进行改进，减少失败匹配次数，节省匹配时间。

3 AC 自动机与双数组 Trie 树

3.1 AC 自动机

AC 自动机对所有的模式串集合构建成 Trie 树结构，Trie 树用于存储模式串，其根节点不存储任何字符，除根节点外的每个节点有且仅有一个字符和状态，叶子结点状态为对应模式串的结束状态。在 Trie 树的基础上为每个节点添加 Fail 指针，用于在字符匹配失败时指示 AC 自动机快速跳转至下一个可能匹配的节点，因此 Trie 树中的每一个节点都有一个指向 Trie 树中其他节点的 Fail 指针。AC 自动机匹配时依次读入文本串中的各字符，通过查找自动机中各节点的状态进行转移 [19]，状态之间转移则表示字符的转移，若当前节点处于终止状态则将其所对应模式串输出，直到文本串匹配完成。

以模式串集合 {he，she，his，hers} 为例，其 AC 自动机结构如图 2 所示，其中 7、8、9 号节点表示终止状态，虚线箭头为 Fail 指针。

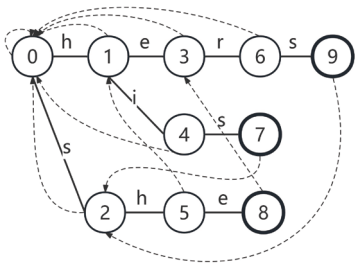

图 2 模式串集合所构建的 AC 自动机

以模式串“his”为例，将其拆分成字符序列“h”“i”“s”，从根节点开始将拆分后的字符序列由左至右依次插入 Trie 树中，插入结果如图 3 所示。

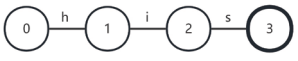

图 3 插入模式串“his”后的 Trie 树

其他模式串采用相同的方式插入至 Trie 树中，插入时模式串可共享公共前缀，如“he”和“his”可共享前缀“h”。

完成 Trie 树构建后，为其各节点构建 Fail 指针。以构建模式串“his”对应节点 (1, 4, 7) 的 Fail 指针为例，首先将根节点及节点 1 的 Fail 指针指向根节点。节点 4 其父节点 1 的 Fail 指针指向根节点，而根节点的子节点中不包含节点 4 的字符“i”，因此节点 4 的 Fail 指针指向根节点。而节点 7 其

父节点 4 的 Fail 指针指向节点的子节点 1 中包含节点 7 的字符“s”，因此节点 7 的 Fail 指针指向节点 1，其他节点采用相同方式完成 Fail 指针构建。

3.2 双数组 Trie 树

双数组 Trie 树是 Trie 树的一种高效实现，该算法的核心思想是使用“base”和“check”两个数组表示 Trie 树中的节点（即状态），减少了内存占用。其中 base 数组可存储 Trie 树中状态转移信息，check 数组可检查状态转移正确性 [20]。若 base[*i*] 和 check[*i*] 值均为 0 则表示节点 *i* 位置为空，即未存储任何节点信息可插入节点信息，若 base[*i*] 值为负数时则表示当前节点处于终止状态，即成功匹配到某模式串，可将其输出 [21]。双数组 Trie 树通过使用数组压缩存储节点和状态信息，提高了空间利用率。状态转移操作为：假设当前所处状态为 *s*，若输入字符为 *c* 且同时满足式（1）（2），则由状态 *s* 转换至状态 *t*，如图 4 所示。

$$\text{base}[s] + c = t \quad (1)$$

$$\text{check}[t] = s \quad (2)$$

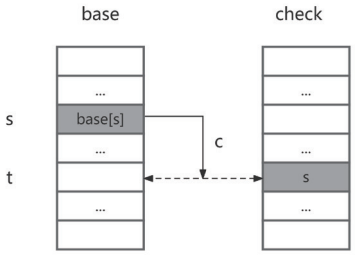

图 4 状态转移图

4 TAC-DAT 算法介绍

TAC-DAT 算法主要思想：通过将 AC 自动机状态转移表和双数组 Trie 树紧凑存储结构结合，在匹配时通过双数组 Trie 树进行状态转移，并利用 AC 自动机的 Fail 指针处理不匹配情况，在保持 AC 自动机匹配速率不变的同时减少内存消耗。同时根据藏文两个音节字之间使用音节点“·”分隔的特点进一步改进，若有字符匹配失败则当前藏文音节字匹配失败，改变读入当前藏文音节字下一个字符的操作，转换为读入匹配失败字符右侧最近一音节点后一位字符继续进行匹配，即从下一个藏文音节字拆分后第一个字符开始匹配，可以有效减少失配字符的匹配从而提高匹配效率。

4.1 构建过程

假设模式串集合 *P*={བཀ，ཐབ，ཐག}，TAC-DAT 算法构建所需数据结构的过程共分为三步：

（1）构建 Trie 树。与图 4 中构建过程相同，将 *P* 构建成 Trie 树，构建结果如图 5 所示，其中节点 12、13、15 表示终止状态。

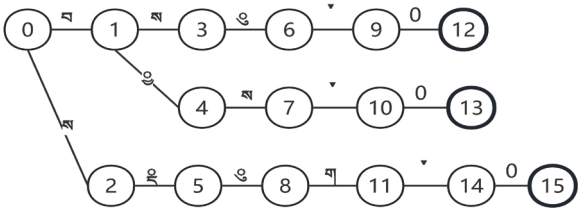

图 5 Trie 树结构图

（2）构建双数组。首先，将 base 数组和 check 数组初始化为 0。根节点所对应数组下标为 0，令 base[0]=1、check[0]=0。根节点共包含 ཀ、ཐ 两个子节点（节点 1、2），令 d=1，由于 check[*d*+3927]=0 且 check[*d*+3943]=0 成立，其中 ཀ、ཐ 的 Unicode 分别为 3927 和 3943，令 check[*d*+3927]=1 和 check[*d*+3943]=1。ཀ 共有两个子节点 ཁ、ྒ（节点 3、4），由于 check[*d*+3943] ≠ 0，令 *d*=*d*+1，即 *d*=2，检查 check[*d*+3943]=0 和 check[*d*+4018]=0，check[*d*+3943]=2 和 check[*d*+4018]=2，此时 ཀ 无子节点，base[1+3927]=2。重复上述操作，完成双数组表的构建，其中 ཀ、ཁ、ག、ྒ、ང、ཅ、ཆ、· 各字符的 Unicode 值分别为 3927、3943、3957、4018、3955、4002、3907、3852，双数组构建结果如表 1 所示。

（3）构建 Fail 数组。将 AC 自动机的状态转移表嵌入到双数组 Trie 树的节点中，对于每个节点，记录其 Fail 指针所指状态并将其转换成数组形式存储。根节点以及第一层节点 Fail 指针均指向根节点，再使用广度优先遍历依次遍历剩余 Trie 树节点。第二层节点 3、4 的父节点 1 的 Fail 指针为根节点，根节点的子节点中不包含节点 4 的字符ྒ，而包含节点 3 的字符ཁ，因此，节点 4 的 Fail 指针指向根节点，节点 3 的 Fail 指针指向节点 2。重复上述操作，完成 Fail 表构建，构建结果如表 2 和图 6 所示。

表 1 base 和 check 数组

	0	ཀ	ཁ	ཁ	ྒ	ཁ	ྐ	ྒ	·	·	ཀ	0	0	·	0	ྒ
i	0	3928	3944	3945	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4020
base	1	2	3	46	155	156	49	102	4010	4011	160	−1	−2	4013	−3	61
check	0	1	1	2	46	61	3	49	155	156	102	4010	4011	160	4013	2

表 2 Fail 数组

	0	ཀ	ཁ	ཁ	ྒ	ཁ	ྐ	ྒ	·	·	ཀ	0	0	·	0	ྒ
i	0	3928	3944	3945	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4020
Fail	0	0	0	3944	0	3944	0	0	0	0	0	0	0	0	0	0

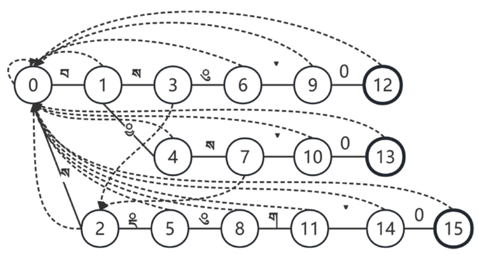

图 6 Trie 树 Fail 指针

4.2 匹配过程

在对文本串匹配时，依次读入文本串中的字符，根据公式（1）（2）、表 1、表 2 进行字符状态转移。匹配过程包含以下两种情况：若与数组表中转移后的节点字符匹配，则根据 base 数组和 check 数组沿着当前 Trie 树路径自上至下匹配，直至结束状态节点；若与根据 base 数组和 check 数组中转移后的节点字符不匹配，无法找到下一个匹配节点，则根据 Fail 数组指向节点或根据音节点跳转节点，继续进行匹配至结束状态节点，重复上述步骤将文本串中所有字符匹配完成。对于文本串 T=“ཛེས་སྨུག་ལྡན་པའི་བོད་སྦྱོངས་དགའ་བསུ་ཕྱེད་ལ་བཞུ།”，模式串集合 P={བསུ་，བྱུས་，སྨུག་}，TAC-DAT 算法的匹配过程如下：

（1）读入字符ཙ进行匹配，其 Unicode 值为 3929，计算 base[0]+3929=3930，由于 base[0] ≠ check[3930]，状态转移失败。由于 Fail[0]=0，状态转移至 0，继续读入右侧最近一音节点后一位字符ས进行匹配。

（2）依次读入字符ས、ྨ、ུ、ག、་进行匹配，节点ས的 Unicode 值为 3943，计算 base[0]+3943=3944，由于 base[0] = check[3944]，状态转移至 3944，剩余字符依次匹配，经历状态转移 0 → 3944 → 4005 → 4006 → 4009 → 4012 → 4013 至结束状态，输出模式串སྨུག་。

（3）依次读入字符ལ、ྡ、ན、ི、པ、ད进行匹配，根据 base 数组和 check 数组可知，均状态转移失败。

（4）依次读入字符བ、ས、ུ、་进行匹配，节点བ的 Unicode 值为 3927，计算 base[0]+3927=3928，由于 base[0] = check[3928]，状态转移至 3928，剩余字符依次匹配，经历状态转移 0 → 3928 → 3945 → 4003 → 4007 → 4010 至结束状态，输出模式串བསུ་。

（5）依次读入字符ཁ、ལ、བ、ཞ进行匹配，根据 base 数组和 check 数组可知均状态转移失败。

（6）无剩余字符，匹配结束。

5 实验及结果分析

本次实验通过分别固定模式串数量和文本串长度，对 TAC-DAT 算法以及 AC 自动机的运行时间和内存占用进行对比分析。实验软硬件环境为 12th Gen Intel(R) Core(TM) i5-12500H，内存为 16 GB，64 位 Windows 11 系统，全部算法使用 JAVA 实现，编译软件为 IDEA 2023，jdk 版本为 1.8。测试使用文本串来源为真实藏文书籍，长度为 1~512 MB。模式串集合来源为复旦大学自然语言处理实验室发布的藏语新闻数据集中随机生成藏文词，数量为 1 万个至 20 万个。

本次实验共为三组，第一组固定模式串数量为 20 万个，文本串长度由 1 MB 递增至 512 MB，测试 TAC-DAT 算法和 AC 自动机运行时间，实验结果如表 3 所示。第二、三组文本串长度均固定为 512 MB，模式串数量由 1 万个递增至 20 万个，测试 TAC-DAT 算法和 AC 自动机运行时间及内存占用，实验结果如表 4、表 5 所示。为减少外部因素对实验结果的影响，每组实验均运行 10 次，取平均值作为最终实验结果。

表 3 模式串数量固定为 20 万，算法在不同长度文本串下的运行时间

模式串数量：20 万个　　时间单位：s		
文本串长度（MB）	TAC-DAT	AC
1	0.03	0.38
4	0.08	0.59
8	0.13	0.75
16	0.25	4.02
32	0.53	4.44
64	5.09	8.63
128	15.20	23.90
256	29.29	54.18
512	177.14	294.33

表 4 文本串长度固定为 512 MB，算法在不同数量模式串下的运行时间

文本串长度：512 MB　　时间单位：s		
模式串数量（万个）	TAC-DAT	AC
1	7.70	12.53
3	11.28	31.22
6	21.50	50.97
9	44.36	79.28
12	59.14	88.20
15	118.77	136.28
18	131.72	154.06
20	180.89	225.72

表 5 文本串长度固定为 512 MB，算法在不同数量模式串下的内存占用

文本串长度：512 MB　　内存单位：MB		
模式串数量（万个）	TAC-DAT	AC
1	746	930
3	1440	1851
6	1700	2368
9	2157	2506
12	2369	2566
15	2873	3080
18	2943	3307
20	4289	4439

由表 3 和表 4 可以观察到，当模式串数量固定为 20 万个时，文本串长度为 16 MB 时，TAC-DAT 算法运行时间相较 AC 自动机运行时间减少约为 93%，文本串长度为 64 MB 时，运行时间减少约为 36%。当文本串长度固定为 512 MB 时，模式串数量为 3 万个时，运行时间减少约为 63%，模式串数量为 15 万个时，运行时间减少约为 12%。产生此种现象的原因为：AC 自动机在对藏文进行多模式匹配时，若一个藏文音节字拆分后某个字符匹配失败，会继续对当前音节字剩余字符进行匹配，增加失败匹配次数，使得匹配时间增加。而 TAC-DAT 算法会从下一个藏文音节字拆分后第一个字符开始匹配，减少失败匹配次数，进而降低匹配时间，提高匹配效率。

由表 5 可以观察到，当文本串长度固定为 512 MB 时，模式串数量为 3 万个时，TAC-DAT 算法内存占用相较 AC 自动机内存占用减少约为 22%，模式串数量为 20 万个时，内存占用减少约为 3%，因此使用双数组 Trie 树有效降低 AC 自动机内存占用空间。

根据以上实验结果可知，在处理相同大小的文本串和模式串时，TAC-DAT 算法匹配准确且效率更高，占用内存更少。因此，相较于将传统算法直接应用于藏文多模式匹配，TAC-DAT 算法更适合藏文多模式匹配。

6 结语

本文介绍了藏文音节字结构、AC 自动机和双数组 Trie 树的原理，并在 AC 自动机与双数组 Trie 树结合基础上根据藏文音节字以音节点分隔的特点改进失败匹配后状态转移过程，减少失败匹配次数。实验结果已证实：TAC-DAT 算法相较于 AC 自动机不仅能减少内存占用空间，而且有效提高匹配的效率，更适应于藏文多模式匹配。

参考文献：

[1] EQUI M, MÄKINEN V, TOMESCU A I, et al. On the complexity of string matching for graphs[J]. ACM transactions on algorithms,2023,19(3): 1-25.

[2] 李亚真，刘黎明 . 涉藏网络舆情演化逻辑与治理路径探析 [J]. 云南警官学院学报 ,2023(3):27-33.

[3] AHO A V, CORASICK M J. Efficient string matching: an aid to bibliographic search[J]. Communications of the ACM, 1975, 18(6): 333-340.

[4] HOU M, SONG Y H, XU D L, et al. A multi-pattern matching algorithm based on double arraytrie[J]Lecture notes in electrical engineering,2014,279(1): 863-868.

[5] KNUTH D E, MORRIS J H, PRATT V R. Fast pattern matching in strings[J]. SIAM journal on computing, 1977(6):2.

[6] BOYER R S, MOORE J S. A fast string searching algorithm[J]. Communications of the ACM,1977,20(10):762-772.

[7] HORSPOOL R N. Practical fast searching in strings[J]. Software: practice and experience,1980,10(6): 501-506.

[8] KARP R M, RABIN M O. Efficient randomized pattern-matching algorithms[J].IBM journal of research and development,1987,31(2):249-260.

[9] SUNDAY D M. A very fast substring search algorithm[J]. Communications of the ACM, 1990, 33(8): 132-142.

[10] WU S, MANBER U. A fast algorithm for multi-pattern searching[EB/OL].[2024-06-28].https://www.cs.arizona.edu/sites/default/files/TR94-17.pdf.

[11] DENCKER P, DÜRRE K, HEUFT J. Optimization of parser tables for portable compilers[J]. ACM transactions on programming languages and systems ,1984, 6(4): 546-572.

[12] AOE J I, YASUTOME S, SATO T. An efficient digital search algorithm by using a double-array structure[C/OL]//The Twelfth Annual International Computer Software & Applications Conference.Piscataway:IEEE, 2002[2024-01-19].https://ieeexplore.ieee.org/document/17222.

[13] 熊刚，何慧敏，于静，等 .HybridFA: 一种基于统计的 AC 自动机空间优化技术 [J]. 通信学报 ,2015,36(7):31-39.

[14] 王蒙，彭展，杨涵刈 . 基于藏文元音构件的字符串匹配算法 [J]. 电子技术与软件工程 ,2022(18):137-142.

[15] 王蒙，彭展 . 一种基于 AC 自动机的藏文多模式匹配算法 [J]. 电子技术与软件工程 ,2023(1):143-148.

[16] 春燕，曲珍，许宁 . 面向藏文基本集编码的单模式匹配算法研究 [J]. 西藏科技 ,2017(3):78-80.

[17] 洋宗，更太加，魏建国，等 . 藏语安多方言语音合成语料库的设计与构建 [J]. 青海科技 , 2023, 30(5): 163-169.

[18] 拉巴顿珠，珠杰，顿珠次仁 . 藏语音素转写的算法研究 [J]. 计算机仿真 ,2024,41(6):370-374+434.

[19] 姜海洋，李雪菲，杨晔 . 基于距离比较的 AC 自动机并行匹配算法 [J]. 电子与信息学报 ,2022,44(2):581-590.

[20] 徐聪，张丰，杜震洪，等 . 基于哈希和双数组 trie 树的多层次地址匹配算法 [J]. 浙江大学学报 (理学版), 2014, 41(2): 217-222.

[21] 李慧，杨炳儒，潘丽芳，等 . 一种基于双数组 Trie 的 B2B 规则串提取方法 [J]. 计算机科学 ,2013,40(5):206-208+223.

【作者简介】

崔晨（1999—），女，山东青岛人，硕士研究生，研究方向：信息内容安全。

（收稿日期：2024-09-25）