基于有限状态自动机的蒙古文同形词校对方法研究*

巩政，廉冰

（内蒙古大学计算机学院，内蒙古自治区 呼和浩特 010021）

摘要：蒙古文在书写时使用变形显现字符来表示文字，由于变形显现字符的形同音异现象，使得在传统蒙古文文本中存在着大量的同形词拼写错误。而导致单词拼写错误的主要原因，通常是输入者把形状相同读音不同的变形显现字符错误的输入到单词中造成的。本文针对蒙古文文本中存在的同形词错误，将词典和构词规则融合到有限状态自动机模型中，提出了一种基于有限自动机的蒙古文同形词校对方法。实验结果表明，该方法可以快速的校对蒙古文文本，平均纠错率为 91. 5%。

关键词：蒙古文；同形词；有限状态自动机；拼写校对

中图分类号：TP391　　**文献标识码：**A

RESEARCH ON PROOFREADING ALGORITHM OF MONGOLIAN HOMOGRAPH ERRORS BASED ON FINITE STATE AUTOMATA

Gong Zheng[1], Lian Bing[1]

(College of Computer Science Inner Mongolia University , Hohhot, Inner Mongolia 010021, China)

Abstract: Mongolian uses presentation character to express text in writing, due to the phenomenon of the same shape but different pronunciation of presentation character leads to a large number of homograph spelling errors in traditional Mongolian texts. However, the main cause of spelling errors is that writers often wrongly input presentation character with the same shape but different pronunciation to word. For homograph errors in Mongolian texts, this paper integrates dictionary and word-formation rules into finite state automata model, and introduces a proofreading method of Mongolian homograph errors based on finite state automata. Experimental results indicate that this method can quickly proofread Mongolian texts and the average correction rate is 91.5%.

Key words: mongolian；homograph；finite state automata；spelling correction

1 引言

蒙古文是一种拼音文字，区别于其他拼音文字的是，在书写蒙古文时，文字是通过变形显现字符来表示的[1]。基于国家标准编码存储的蒙古文书写错误主要是同形词的拼写错误，这类拼写错误主要是因为录入者对国家标准编码中的蒙古文字母中的形同音异变形显现字符使用不当造成的。因此，对蒙古文同形词校对的研究有重大的意义。

蒙古文文本错误的类型包含非词错误、真词错误和句法语义错误[2]，而蒙古文非词错误又包括字形错误和读音错误[3]。传统的检查非词错误的有效方法是查找词典，如果指定单词不在词典中，则认为非词。然而，传统蒙古文是典型的黏着语，其构词和构形都是通过在词根或词干后追接不同的附加成分来完成的[4]，其形态变化丰富、拼写系统复杂。词典中包含的单词数量有限，为扩大词汇的覆盖面，可使用词典结合于构词规则的方法，这也是处理粘着性语言时常用的方法[3]。用国家标准[5]编码表示的蒙古文非词错误大多数属于读音错误，即同形词错误，这类拼写错误人工校对是很难发现的，而字形错误很少，也易于发现。

以往有关蒙古文单词拼写校对方面的论文大多针对的是对非国家标准编码表示的蒙古文拼写错误的处理。由于国家标准[5]编码的蒙古文字母是按读音给出的，录入者发生的拼写错误大多数是读音不准或控制符使用不当造成的。因此，按国家标准[5]编码表示的蒙古文拼

* **收稿日期：**　　**定稿日期：**

基金项目：本体驱动的英蒙机器翻译研究，内蒙古自然科学基金（2013MS0902）

作者简介：巩政（出生年 1965），男，教授，自然语言处理；廉冰（出生年 1989），女，硕士，自然语言处理。

写错误特点和非国家标准表示的蒙古文拼写错误的特点有所不同。目前，有关讨论国家标准[5]编码表示的蒙古文拼写校对方面论文很少见到。

本文将词典和构词规则融合到有限状态自动机模型中，介绍了一种基于有限状态自动机的用国家标准[5]编码表示的蒙古文同形词的校对方法。首先，以《蒙古文正字法词典》[6]为依据，根据蒙古语单词的语法特征以及词的结构，构造词法分析器。其次，根据传统蒙古文国家标准[5]编码基本字符集、传统蒙古文名义字符的变形显现字符以及变形规则，建立同形字符规则库。最后，根据同形字符规则库，在词法分析器上进行启发式搜索，如果输入单词正确，则不做任何处理；如果输入单词错误，则用与其同形的单词对其纠错。

2 词法分析器的构造

在构造词法分析器的时候，本文将词分成静词类、动词类和无变化词类三种情况来考虑。对于静词类和动词类又分为词干和结尾后缀，而对于无变化词类，由于它们没有形态变化，所以只要考虑无变化词类本身即可。将它们分别构造自动机，然后根据蒙古语构词特点以及词的结构，组装成词法分析器。

2．1 构造静词自动机

本文以《蒙古文正字法词典》[6]为依据，将《蒙古文正字法词典》[6]里的所有静词类提取出来，即将名词、形容词、数词、量词、时位词、代词提取出来构造一个静词类词干自动机，再将蒙古文静词类结尾后缀构造一个静词类结尾后缀自动机，最后将这两个自动机连到一起，构造一个静词类自动机。

2.1.1 构造静词词干自动机

蒙古文静词词干自动机 DFA(Deterministic Finite Automata) M1 是一个五元组：

$$M1=(\sum ,Q ,\delta, q_0 ,F)$$

其中，∑是传统蒙古文静词名义字符和控制字符输入字符集；Q 是状态的有限集合，状态分为非终结状态和终结状态两大类，在状态转移图中，非终结状态用单线圈来表示，终结状态用双线圈来表示，有时候，终结状态可能成为非终结状态，这要根据输入字符串的当前字符来判断，如果当前字符是字符串的结尾字符，那么就把它看成是终结状态，否则看成是非终结状态；$q_0 \in Q$ 是初始状态；F 是终止状态集合，包含了所有的终结状态，$F \subseteq Q$；δ是 Q 与∑的直积 $Q \times \sum$到 Q 的映射，有时也称为状态转移函数，$\delta(s_i, c)=s_j$ 表示当前状态 s_i 下，若遇到输入字符 c，则转移到状态 s_j。

图 1 为传统蒙古文静词词干状态转移的部分示意图：

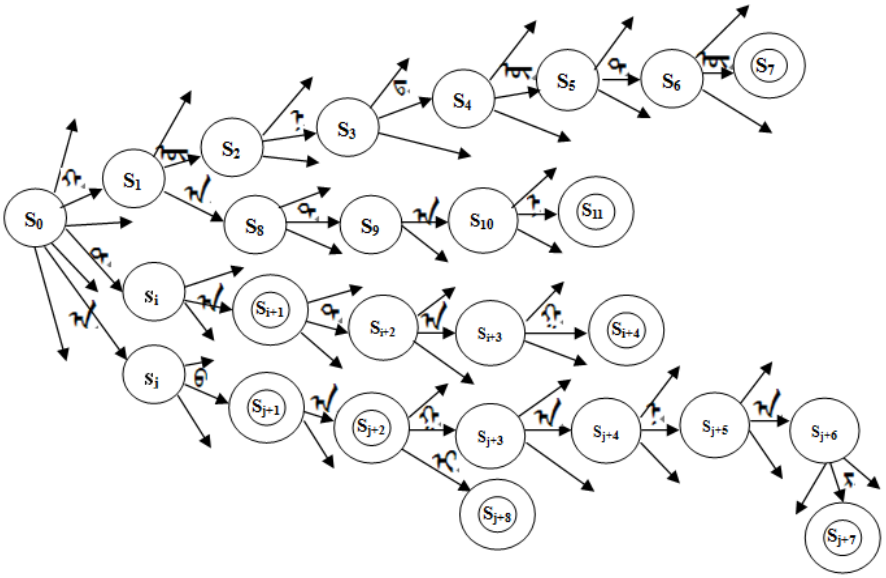


图 1　静词词干的状态转移图

2.1.2 构造静词结尾后缀自动机

蒙古文静词结尾后缀自动机 DFA M2 的构造如同 M1，所不同的是，∑是传统蒙古文静词结尾后缀名义字符和控制字符集。

2.1.3 构造静词自动机

将静词词干自动机 M1 和静词结尾后缀自动机 M2 连到一起，就可构成不确定的有限自动机 N1，然后将 N1 确定化和最小化得到静词自动机，N1 的构成如图 2 所示：

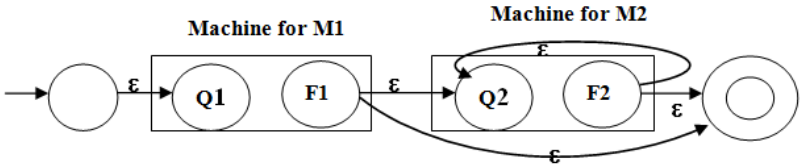


图 2　静词自动机的结构

2．2 构造动词自动机

动词自动机的构造过程，类似于静词，本文根据内蒙古大学计算机学院总结的动词结尾后缀，将《蒙古文正字法词典》[6]里的所有动词的词干切出来，然后分别构造动词词干自动机和动词结尾后缀自动机，最后将它们连起来，构成动词自动机。

2.2.1 构造动词词干自动机

蒙古文动词词干自动机 DFA M3 的构造方法与 DFA M1 基本相同，所不同的是，∑是传统蒙古文动词名义字符和控制字符集。

2.2.2 构造动词结尾后缀自动机

蒙古文动词结尾后缀自动机 DFA M4 的构造方法与 DFA M1 基本相同，所不同的是，∑是传统蒙古文动词结尾后缀名义字符和控制字符集。

2.2.3 构造动词自动机

将动词词干自动机 M3 和动词结尾后缀自动机 M4 连到一起，构成不确定的有限自动机 N2，然后将 N2 确定化和最小化得到动词自动机。N2 的结构如图 3 所示：

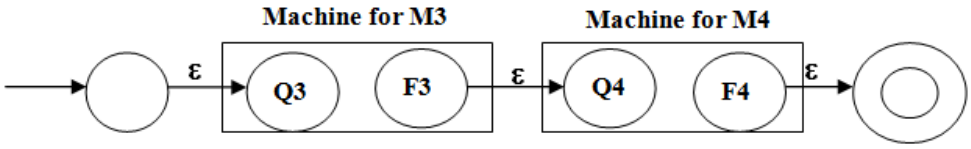


图 3　动词自动机的结构

2．3 构造无变化词自动机

本文将《蒙古文正字法词典》[6]里的所有无变化词提取出来，即将副词、情态词、后置词、摹拟词、语气词、感情词和连接词提取出来构造无变化词自动机 M5，其构造过程与 M1 相似，所不同的是，∑是传统蒙古文无变化词类名义字符和控制字符集。

2．4 词法分析器的构造

下面描述如何将 N1、N2、M5 组合成词法分析器 LA（Lexical Analyzer），图 4 为词法分析器的结构：

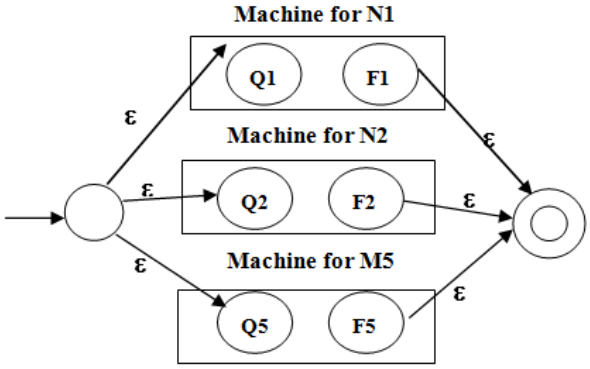


图 4　词法分析器的结构

最后，将图 4 所示的自动机使用确定化算法和最小化算法得到词法分析器 LA。

3 同形字符规则库的建立

蒙古文单词出现同形异音现象的主要原因是传统蒙古文名义字符存在着编码不同，但是它们的变形显现字符形状相同的情况，输入者常常按照字形是否相同来选择字符，而不是按照读音选择相应的字符。

本文中，根据传统蒙古文名义字符的变形显现字符集和它们相应的变形规则，总结出了哪些名义字符的变形显现字符是相同的，以及在什么情况下它们使用相同的字形，从而建立了传统蒙古文同形字符规则库，如表 1 所示：

表 1　同形字符规则库

序号	同形字符（名义形式）	同形字符(输入形式）	变形显现形式	构成同形的条件
1	+FVS1 和	a’ 和 e		独立成词
2	和 + FVS1	a 和 e’		词首
3	和	a 和 e	（词中）、（词末）	词中或词末且前面不是 h、不是 g
4	和	a 和 n	（词中）、（词末）	词中、后为辅音、不与其前面字符构成强制性合体字或词末、不与其前面字符构成强制性合体字
5	和	e 和 n	（词中）、（词末）	词中、后为辅音、不与其前面字符构成强制性合体字或词末、不与其前面字符构成强制性合体字
6	和 +FVS1	a 和 n’		词中、后为元音、不与其前面字符构成强制性合体字
7	和 +FVS1	e 和 n’		词中、后为元音、不与其前面字符构成强制性合体字
8	+FVS1 和 +FVS1	a’ 和 e’		词末
9	202F 和 MVS	-和_		词中且其后所跟字母为 a 或 e
10	和 +FVS1	e 和 n’		词首
11	和	w 和 v	（独立）、（词首）、（词中）、（词末）	独立、词首、词中、词末
12	+FVS1 和 +FVS1	w’ 和 v’		词中
13	和	W 和 w	（词中）、（词末）	词中或词末且不与其前面字符构成强制性合体字

14	和	W 和 v	（词中）、（词末）	词中或词末且不与其前面字符构成强制性合体字
15	和和和和	w 和 v 和 o 和 u		词末
16	和	o 和 u	(独立)、(词首)(词中 0	独立形式或词首或词中
17	+FVS1 和+FVS1	o’ 和 u’	（词中）、（词末）	词中或词末
18	+FVS2 和+FVS2	o” 和 u”		词中
19	和	t 和 d	（词首）、（词中）	词首或词中而且词中时后面为元音
20	和	h 和 g		词首或词中，并且与其后面的字符构成强制性合体字
21	和+FVS1	h 和 g’	（词首）、（词中）	词首或词中且后为阳性元音
22	+FVS1 和+FVS1	h’ 和 g	（词首）、（词中）	词首或词中且后为阳性元音
23	和	E 和 W	（词中）、（词末）	词中或词末且不与其前面字符构成强制性合体字
24	和	i 和 y	（词中）、（词末）	词中或词末且不与其前面字符构成强制性合体字
25	和+FVS1	j 和 y’		词首
26	和	j 和 y		词末并且其前面字符为 MVS
27	和	H 和 Z		词中

其中，’代表控制字符 180B；”代表控制符 180C；_代表控制字符 180E，-代表控制字符 202F。

4 校对算法

有限自动机包含有用的信息，这些信息可以很容易的提取出来而且很适合作为搜索过程的可靠的向导，避免了宽度和深度优先搜索所带来的搜索复杂度的指数增长。我们使用的额外的信息—从有限自动机的某些状态延伸出去的未来的路径可能遇到的符号—可以作为探索合适路径的一个猜测，因此正适合使用 A*算法。

在 A*算法中，实际上，我们需要两类代价：已经搜索的路径的代价（g）,和对未来代价估算（h）,它不能超过未来代价的实际值。在选择接下来沿着哪条路径扩展的每一步，我们考虑到目前为止所花费的实际代价（g），和对未来代价的一个估计值（h），产生 f=g+h。

在有限自动机里，搜索同形字符的匹配自然产生 g，g 是指在有限自动机里到达状态 s 时，和输入字符串比较，到目前为止所发生的同形字符替换的数目。对 h 的估算是基于我们已经简单介绍的启发信息。

4．1 启发信息

对于每一个状态，从该状态出发长度为 n 的所有可能的路径上的字母作为启发信息存储起来。图 5 给出了非循环的有限自动机，每条弧的上面为名义字符，下面为其拉丁转写，它的每一个状态都被向前 n 步遇到的所有可能的字母信息标记，在这里 n=2。

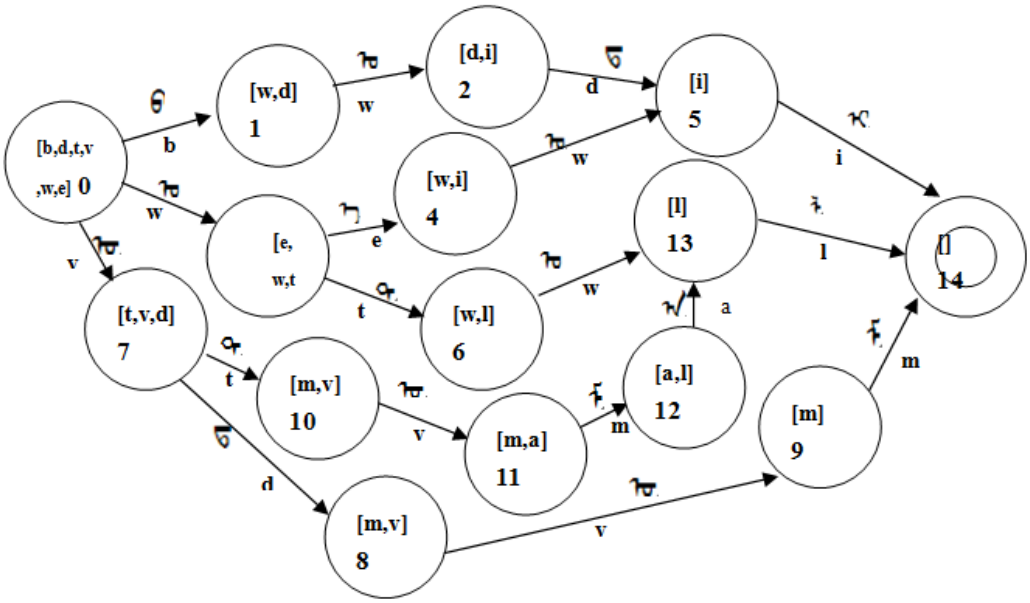


图 5　带启发信息的有向图（n=2）

在图的搜索过程中，这个向量和我们输入单词将要匹配的字母进行比较。在状态里面存储的字母和输入单词将要匹配的字母的差异作为启发函数的代价。也就是说，对于输入单词当前匹配位置向前 n 个字母的每一个，如果在当前节点状态的向量里没有找到，则启发函数 h 的代价加 1。这就反映了一个估计，在未来的某个时刻，任何字母不在我们正在搜索的路径中，那么它相对于输入单词来说就是同形字母替换。

例如，假设我们正在匹配的蒙古文单词拉丁转写形式是 vdvmal，而且输入单词匹配到字母 v,有向图匹配到了图 5 中的状态 9，对于 n=2 的情况，将要匹配的字母是 ma，那么此时，启发函数 h 的值为 1。因为接下来将要匹配的两个字母是 m 和 a，而状态 9 里面的向量不包括 a。

在 A*算法中，h 对于任何的 n 都是合理的，因为它不会过高估计到达目标的代价。如果，在接下来的几步，未来的路径和输入单词将要匹配的字符串中有 i 个符号不同，这些符号将要通过同形字符替换产生，因此 h 不会超过到达目标的代价。而且 h 的值可以提前计算出来，对于每一个状态，我们简单的执行界限深度优先搜索，界限值为 n，将我们在边上遇到的所有的符号存储在我们开始搜索的那个状态里。

Mans Hulden 在有限自动机上寻找快速近似匹配的字符串的实验中发现[7]：使用 n=2 和 n=∞组合值要优于其他的 n 值，考虑空间的问题，对于拼写检查来说，n=2 效果会更好。在这里我们使用 n=2 作为我们的启发信息。我们的搜索算法也是在 Mans Hulden 的启发式搜索算法的基础上，根据蒙古文的特点，进行修改得到的纠错算法。

4．2 纠错方法

我们把对蒙古文单词的查错和纠错放到了一起来判断，如果单词正确则不做处理；如果单词错误则对其进行纠错。我们的纠错方法是在词法分析器上进行启发式搜索，在当前状态下，要么沿着和待匹配的字符串完全相同的路径前进，要么沿着和待匹配的字符串同形的路径前进，每当有多条路径可走时，利用启发函数的函数值来判断，先沿着具有最小 f 值的节点往下延伸。在算法中，如果第一个输入就出错，则沿着在词首与其同形的路径前进。如果单词正确，那么先找到的字符串肯定是该正确的单词；如果单词错误，那么先找到的字符串是与输入单词字形一样，而且差距最小的字符串，若有多个同形词，可以根据规则来判断或者提供候选建议。

初始，搜索开始于一个称为初始状态的节点和单词的开始位置 0（假设字符串的起始位

置从 0 开始）。然后，我们考虑在自动机上，从状态 v 出发的每一条可能的边，如果该边和输入单词的当前位置的字母相同，则创建一个新的节点，在节点上标记到达下一状态 v’，输入字符串前进一个位置，重新计算代价 f=g+h，并存储到该节点里；如果该边和输入单词的当前位置的字母不相同，则判断在当前条件下，自动机上的字母和输入单词的字母是否同形。如果同形，则创建一个新的节点，在节点上标记到达下一状态 v’，输入字符串前进一个位置，g 加 1，因为发生了一次同形字符替换，重新计算代价 f=g+h，并存储到该节点里；如果不同形，则放弃对该路径的搜索。接下来，我们选择具有最低 f 值的节点扩展，以此类推，直到我们找到答案位置。自动机到达终点状态并且输入单词也到达词末时，如果 f=0，则说明没有发生同形替换的情况；如果 f≠0，则说明输入单词错误，需要校对，那么第一个找到的是与待校对单词同形且差距最小的一个。图 6 是在图 5 的自动机中，对输入单词ᠸᠳᠸᠮᠠᠯ（vdvmal）进行的扩展搜索。

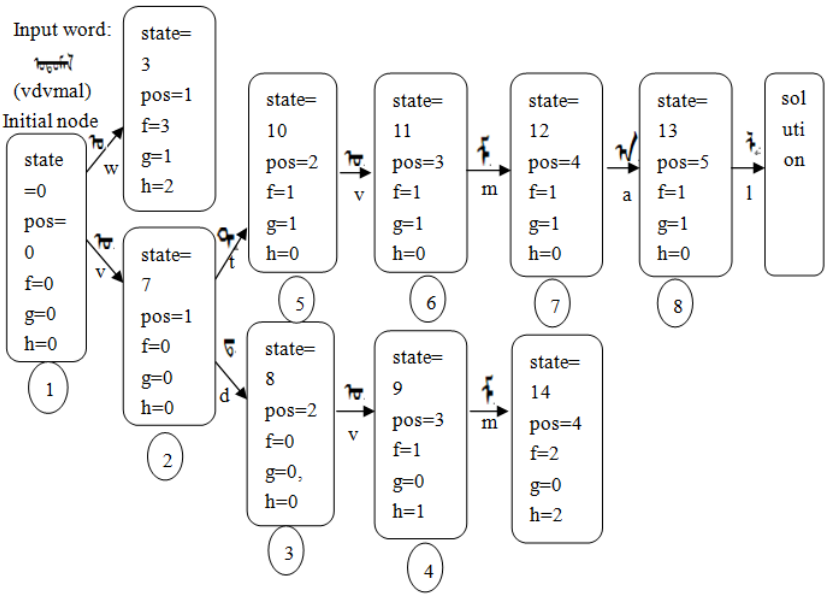


图 6　对输入单词进行扩展搜索

图 6 中的圆圈序号代表节点被扩展的先后次序，节点里包含的信息包括：节点所处的状态 state、输入字符串比较到的位置 pos、比较到当前状态所发生的同形字符替换的次数的 g 值、未来走两步可能遇到的符号是否在当前路径上的 h 值、以及 f 值，f=g+h。每次都找具有最小 f 值的节点扩展，如果 f 值相同，那么首先扩展走的最远的节点。纠错算法如下：

```
SEARCH (word, A)
OPEN (v, pos, g, h) ←— (start state, 0, 0, calculate_h() )
while OPEN not empty
 do
 {
        (v, pos, g) ←— remove-cheapest (OPEN)
        for each edge (v —→v’ )
        do
        {
              if v is final and pos is end of word
                    then SOLUTION (node)
              if word (pos)=edge
                    then ADD (v’ , pos+1, g, calculate_h() )
              if word (pos)与 edge 同形
                    then ADD (v’ , pos+1, g+1, calculate_h())
        }
}
```

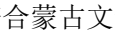
5 实验结果与分析

蒙古文国家标准[5]编码及变形显现字符及控制符使用规则 2011 年底才正式颁布，因此用国家标准编码存储的蒙古文语料很少。本实验中使用的测试语料是使用国家标准[5]编码存储的传统蒙古文。实验中从语料中随机抽取了十个测试文本，这些测试文本有的来自于输入人员录入的蒙古文（1 万多字），有的来自于蒙古文网站（2 万多字），这些文本中包含了国家标准编码表示的蒙古文单词中常见的拼写错误，如蒙古文第四、第五，第六、第七元音使用混淆错误，辅音 d、t 混淆使用错误，双元音使用错误，元音阴阳性错误、控制符使用错误等类型。我们对本文中提出的传统蒙古文同形词错误校正方法进行了测试，用纠错率对算法的性能进行了评价，实验结果如表 2 所示，其中：

$$\text{纠错率} = \frac{\text{正确校正的单词个数}}{\text{需校正的单词个数}} \times 100\%$$

表 2　校对方法测试结果

样本号	蒙文词(个)	纠错（个）	正确纠错（个）	纠错准确率（%）
1	11236	1543	1389	90.0
2	2197	592	534	90.2
3	2234	223	208	93.3
4	2105	252	238	94.4
5	2724	258	235	91.1
6	1421	189	170	89.9
7	1603	162	147	90.7
8	2514	261	242	92.7
9	2073	268	244	91.0
10	2080	282	259	91.8

从表 2 的数据可以看出，用国家标准编码表示的蒙古文录入的错误率比较高，这是因为从传统意义上讲蒙古文单词只要字形正确就行，因此用非国标编码表示的单词拼写错误相对较少[3]，反之用国家标准的读音编码表示的单词拼写错误反而较多，且错误中大多数都是同形词错误。表 2 的数据也说明利用本文提出的方法对使用国家标准[5]编码存储的传统蒙古文进行同形词校对是可行的，平均纠错准确率为 91.5%。另外，在测试中，除了常见错误类型外，我们发现有阴阳性混用的情况。例如，校对文本中存在这样的单词：  （alvs-eqe），它不符合蒙古文语音和谐规则，正确的写法应该是  （alvs-aqa）,对于这种同形词错误，我们的校对算法也能校对。从纠错率上看，本方法的纠错能力与其它方法比较还不算高，但其它已有方法主要针对的是非国家标准编码表示的蒙古文，与本文讨论的问题可比性不强。

测试中发现，出现没有实现校正的单词的主要原因是词典中包含的单词有限，对于词典中没有包括的单词不能正确的纠正；词典中的某些单词的词性可能划分的有误，导致不能正确的纠错；对于某些外来词，一般是根据单词读音构词，不遵循蒙古文构词规则，可能会判断错误。

6 总结

本文对应用国家标准[5]编码存储的传统蒙古文文本中存在的同形词错误的校正方法进行了研究，将词典和规则融合到有限自动机模型中，首次利用同形字符规则库，在词法分析器上进行启发式搜索，对输入单词进行查错和纠错。该方法主要适用于非词错误的查错和纠错，而不太适用于真词、句法和语义错误的校对。今后，可以通过扩大词典规模，进一步完善同形字符规则库来提高算法的准去性和智能性，而且对于有多个候选单词出现时，可以使用语言模型来对每个候选词打分，然后选出得分最高的候选词来纠错。

参考文献

[1] 清格尔泰.现代蒙古语语法[M].修订版，呼和浩特：内蒙古人民出版社，1999.

[2] 张仰森，俞士汶.文本自动校对技术研究综述[J].计算机应用研究，2006,6:8-12.

[3] 斯•劳格劳.基于不确定有限自动机的蒙古文校对算法[J].中文信息学报，2009,23(6):110-115.

[4] 张仰森.中文校对系统中纠错知识库的构造即纠错建议的产生算法[J].中文信息学报，2001,15(5):33-39.

[5] 国家质量监督检验检疫总局、国家标准化管理委员会. GB 25914-2010 信息技术　传统蒙古文名义字符、变形显现字符和控制字符使用规则[S]. 北京：中国标准出版社, 2011.

[6] 《蒙古文正字法词典》编委会.蒙古文正字法词典[M].内蒙古人民出版社，1999 年，呼和浩特.

[7] Hulden M. (2009b). Fast approximate string matching with finite automata. Procesamiento de Lenguaje Natural 43(1):57-64.