

基于纠错候选集和似然匹配的自动纠错算法*

张仰森 丁冰青

山西大学计算机科学系 (030006)

摘要 针对待校对文本中的常见错误类型, 本文介绍了一种基于纠错候选集和似然匹配的自动纠错算法, 并用字字同现信息、词性同现信息对得出的纠错建议进行排序, 充分利用了出错字串的特征, 并结合上下文启发信息, 可有效地对文本中的错字、漏字、多字、易位、多字替换等错误提供纠错建议。

关键词 纠错候选集 似然匹配 纠错算法

A automatic correcting algorithm based on correcting suggestion sets and likelihood match

Zhang Yansen Ding Bingqing

Computer science department, Shanxi University 030006

Abstract This paper introduce a automatic correcting algorithm based on correcting suggestion sets and likelihood match according to common error types in pre-proofreading text. The algorithm makes a full use of the characteristics of wrong strings and context heuristic information and sorts the suggestions on character-character and part-of speech occurrence information. It can provide correcting suggestions for such errors as ghost word, missed Chinese characters, superfluous Chinese characters, reversed Chinese characters and substituted Chinese characters etc.

Keyword Correcting suggestion sets; Likelihood match; Correcting algorithm

一、自动纠错研究概述

自动纠错是文本自动校对的一个重要组成部分, 它为自动查错时侦测出的错误字串提供修改建议, 辅助用户改正错误。修改建议的有效性是衡量自动纠错性能的主要指标, 它有两点要求, 一是提供的修改建议中应该含有正确或合理的建议; 二是正确或合理的修改建议应尽可能排列在所有修改建议的前面。因此, 纠错修改建议的产生算法及其排序算法是自动纠错研究的两个核心课题。待校对文本中的错误类型一般有以下几类^{[1][2][3]}:

①容易混淆的字词错误

如: 宋庆龄于 8 月 14 日凌晨 1 时抵达雅加达。(凌晨)

②由于编码相同或相近(包括拼音、五笔编码)引起的错字错误

如: 我们这些化(hua)夏子孙。(华 hua 夏); 就难免必(ny)理不平衡。(心 nt 理)

③由于编码相同或相近(包括拼音、五笔编码)引起的多字替换错误

* 山西省自然科学基金资助项目

如:对于大规模真实广西(yysg)的研究。(文本 yysg)

④漏字、多字、易位造成的错误

如:文稿中仍会遗留许多误。(错误); 维护建筑市场场秩序。(市场)
忽视发挥利率的杆杠作用。(杠杠)

⑤英文单词拼写错误

如:95年4月联想推出新版 office 办公软件。(office)

文献[4]采用模式匹配方法对长词进行纠错处理,但没有充分利用出错字串的特征,算法计算量大;文献[3]提出一种替换字表结合主词典,通过加字和换字对侦测出来的错误字串提供修改建议的纠错算法,但该算法的纠错建议局限于替换字表,没有考虑上下文启发信息,主要考虑对错字这种错误类型进行纠错,对漏字、多字、易位、多字替换、英文单词拼写等错误类型的纠错能力较弱。

二、纠错思想及纠错知识库的构造

2.1 纠错思想

本文采用纠错候选集和似然匹配相结合的纠错方法,根据字字同现、词性同现等信息排除不合理建议,对纠错建议进行排序,使合理的纠错建议尽量排在前面。这样既充分利用了输入文本中错误的特征,又能突破候选字集的局限性,可纠正一些不能由纠错候选集纠正的错误(如由漏字、多字、易位等引起的错误),同时增强了纠错建议的可读性。

对易混淆的字词错误,纠错时查易混淆词词典,直接给出纠错建议;对输入编码相同或相近(包括拼音、五笔编码)的字词错误,用纠错候选字、词替换出错字词,给出纠错建议;对漏字、多字、易位错误,采用似然匹配法,得到纠错建议。

2.2 纠错知识库的构造

2.2.1 易混淆词词典的构造

收集常见的易混淆字词,建立易混淆词词典,在纠错时先查该词典,可直接得到部分易错字词的纠错建议,另外,可以动态地将人工给出的纠错建议加入该词典中。该词典的格式为:

错误字串	纠错建议
零晨	凌晨

2.2.2 输入编码相同或相近(包括拼音和五笔编码)字词候选集的构造

1. 拼音编码相同或相近的字词候选集的构造

拼音编码相同或相近的字取自 UC DOS 中文平台的拼音方案,其中共有编码 11894 组,取一个拼音对应多个字或词的 9922 组编码为拼音编码纠错建议候选集 $S_{py} = S_{pyzi} \cup S_{pyci}$

(1) 音同、音近候选字集 S_{pyzi} 6714 组:

如:音同“bei 北被倍备背辈贝杯卑臂悲碑”等;

(2) 音同、音近候选词集 S_{pyci} 3208 组:

如:“baohan 包含 饱含 包涵”。

2. 五笔字型输入编码相近字词候选集的构造

我们设计了一个五笔字型编码相似函数,用于从 UC DOS 中文平台的五笔字型输入方案

(其中共有编码 22482 组)中提取编码相同或相近的字词共 36837 组, 构成五笔字型编码纠错建议候选集 $S_{wb}=S_{wbzi} \cup S_{wbci}$, 其中候选字集 S_{wbzi} 5916 组, 候选词集 S_{wbci} 30921 组。

如: 字(pb) — 安(pv) 落伍(aiwg) — 范例(aiwg)

3. 纠错候选集的排列格式

令纠错候选字集为 $S_{zi}=S_{wbzi} \cup S_{pyzi}$, 候选词集为 $S_{ci}=S_{wbci} \cup S_{pyci}$, 其内容排列方法为:

a. 纠错候选字集 $S_{zi} = S_{pyzi} \cup S_{wbzi}$, 按字的内码大小排序, 每个字的候选字集占一行; 字在候选字集中的行号以哈希函数 $addr=(Byte1-0xb0)*94+(Byte2-0xa1)$ 来定位, 其中 Byte1Byte2 为该字的内码。设 Z 为字, 它及其候选字的排列格式为:

$Z \quad C_{z1} \dots C_{zi} \dots C_{zj} \dots C_{zm}$ 这里 C_{zi} 为第 i 个纠错候选字。

在整个候选字集中, 平均候选字数为 38 个, 最多为 97 个; 候选字按字频排列。

b. 纠错候选词集 $S_{ci}=S_{pyci} \cup S_{wbci}$, 按词的首字内码大小排序, 所有首字相同的词按平均字字同现频率大小排列在同一行; 若某一词有多个候选词, 则这些词之间用“;”隔开, 其次序按平均字字同现频率大小排列。在整个候选词集中, 平均纠错候选词个数为 2.7 个, 最多为 49 个。

2.2.3 漏字、多字、易位词的纠错建议获取

漏字、多字、易位错误情况比较复杂, 没有一定的规律可循, 不能通过简单的换字操作得到纠错建议, 本文取错误字串的首字和尾字做为启发信息, 采用似然匹配的方法, 将与出错词“相似”的词作为纠错建议。

1. 似然匹配算法

对字串 $Z_1Z_2\dots Z_n$ 和词 $C_1C_2\dots C_m$; 令函数

$$\text{Same}(Z_i, C_j) = \begin{cases} 1 & \text{当 } Z_i=C_j \\ 0 & \text{当 } Z_i \neq C_j \end{cases}$$

则当 $m \geq n$ 时, 有:

$$\sum \text{Same}(Z_i, C_i) \geq n-1 \quad (1 \leq i \leq n; Z_i=C_i)$$

$$\text{或 } \sum \text{Same}(Z_{n-i+1}, C_{m-i+1}) \geq n-1 \quad (1 \leq i \leq n; Z_n=C_m)$$

$$\text{或 } \sum \text{Same}(Z_i, C_i) + \sum \text{Same}(Z_{n-i+1}, C_{m-i+1}) \geq n-1 \quad (1 \leq i \leq n; Z_i=C_i; Z_n=C_m)$$

当 $m < n$ 时, 有

$$\sum \text{Same}(Z_i, C_i) + \sum \text{Same}(Z_{n-i+1}, C_{m-i+1}) = m \quad (2 \leq i \leq m-1; Z_i=C_i; Z_n=C_m)$$

那么, 称字串 $Z_1Z_2\dots Z_n$ 与词 $C_1C_2\dots C_m$ 似然匹配, 记为 $\text{Match}(Z_1Z_2\dots Z_n, C_1C_2\dots C_m)$; 若 $\text{Match}(Z_1Z_2\dots Z_n, C_1C_2\dots C_m)$, 则 $C_1C_2\dots C_m$ 可作为字串 $Z_1Z_2\dots Z_n$ 的纠错建议。

似然匹配可纠正任一字与原串不同的词条、首尾漏字、词中多字、少字等错误。

如: match (“当务之争”, “当务之急”), match (“而走险”, “铤而走险”)等。

2. 字驱动双向词典的构造

为便于查找和对词进行似然匹配, 重组词典信息, 构造以字为驱动的双向词典。该词典以词的首字内码大小排列。词典的数据结构定义如下:

```
struct Item
{
    char* Character [2]; //字
    int    Frequency;    //字频
    int    IsDanZiCi;    //是否为单字词, 是为 1, 否则为 0
    int    BeFirst;      //能否做词典中某词的首字, 是 1, 否则 0
}
```

```

int    BeSecond;    //能否做词典中某词的第二字,是1,否则0
int    BeThird;    //能否做词典中某词的第三字,是1,否则0
int    BeFourth;   //能否做词典中某词的第四字,是1,否则0
long*  BiHead;     //指向以该字为首字的二字词
long*  BiTail;     //指向以该字为尾字的二字词
long*  TriHead;    //指向以该字为首字的三字词
long*  TriTail;    //指向以该字为尾字的三字词
long*  QuarHead;   //指向以该字为首字的四字词
long*  QuarTail;   //指向以该字为尾字的四字词
long*  MultiHead;  //指向以该字为首字的多字词
long*  MultiTail;  //指向以该字为尾字的多字词
}Zi;
如:Character="安"    Frequency=5067    IsDanZiCi=1    BeFirst=1
    BeSecond=1      BeThird=1        BeFourth=1
    BiHead->插 定 顿 放 分 抚 好 家 静 居 康 乐 眠 民 宁 排 培 全 然 设
           身 生 适 泰 危 慰 稳 息 闲 祥 详 享 歇 心 逸 葬 置 装 甌
    BiTail->保 不 伏 公 苟 毫 建 柳 偏 平 请 偷 晚 微 问 相 招 治
    TriHead->乐 窝 理 会 眠 药 全 部 全 带 全 阔 全 帽 全 线
    TriTail->NULL
    QuarHead->安 静 静 分 守 己 家 落 户 居 乐 业 民 告 示 全 系 数 然 无 恙 营 扎 寨 于 现 状
    QuarTail->NULL      MultiHead->NULL      MultiTail->NULL

```

三、纠错建议产生算法和纠错建议排序算法的实现

3.1 纠错建议产生算法

定义纠错建议存放的结构为:

```

struct Suggestion
{
    char Suggestion[MAX][20]; //存放纠错建议
    int  weight[MAX];        //相应纠错建议的评价值
}Jianyi;

```

设字 Z_i 的候选字为 $Z_{ci1}, Z_{ci2}, \dots, Z_{cim}$, 则纠错候选建议的产生算法如下:

(1)纠错建议计数器 i 初始化为 0; 若错误为英文单词拼写错误, 将出错单词按骨架生成算法生成骨架键, 查骨架键词典, 若命中, 对应的单词为纠错建议。

(2)查易混淆词词典, 若命中, 则将该纠错建议放入纠错建议缓冲区 $Jianyi.Suggestion[i]$; 纠错建议计数器 i 加 1。若纠错建议不止一个, 循环做 2, 直至取完所有纠错建议。

(3)查编码候选词集 S_{ci} , 若命中, 将该纠错建议放入纠错建议缓冲区 $Jianyi.Suggestion[i]$; 纠错建议计数器 i 加 1。若纠错建议不止一个, 循环做 3, 直至取完所有纠错建议。

(4)若字符串 Z 的长度 $Length(Z)=1$ (以字计), 字符串 $Z=Z_1$

若 $Z_1.IsDanZiCi=1$, 取候选字集 S_{zi} 中相应的候选字 $Z_{ci1}, Z_{ci2}, \dots, Z_{cim}$ 放入纠错建议缓冲区 $Jianyi.Suggestion[i]$, 计数器 i 加 1, 纠正“音同、音近、形近”的错字。

(5)若字符串 Z 的长度 $Length(Z)=2$ (以字计), 字符串 $Z=Z_1Z_2$

若 Z_2Z_1 成词, 则将 Z_2Z_1 放入纠错建议缓冲区 $Jianyi.Suggestion[i]$; 纠错建议计数器 i 加 1, 可纠正“易位”错误。若 $Z_{ci1}.BeFirst=1$.and. $Z_{ci1}Z_2$ 成词或 $Z_{ci2}.BeSecond=1$.and. Z_1Z_{ci2} 成词或

$Z_{c1i} \cdot \text{BeFirst}=1$ 且 $Z_{c2j} \cdot \text{BeSecond}=1$ 且 $Z_{c1i}Z_{c2j}$ 成词则将 $Z_{c1i}Z_2$ 或 Z_1Z_{c2j} 或 $Z_{c1i}Z_{c2j}$ 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1, 可纠正“音同、音近、形近”的错字。

(6)若字串 Z 的长度 $\text{Length}(Z)=3$ (以字计), 字串 $Z=Z_1Z_2Z_3$

若 $Z_{c1i} \cdot \text{BeFirst}=1$ 且 $Z_{c1i}Z_2Z_3$ 成词, $Z_{c2j} \cdot \text{BeSecond}=1$ 且 $Z_1Z_{c2j}Z_3$ 成词 或 $Z_{c3k} \cdot \text{BeThird}=1$ 且 $Z_1Z_2Z_{c3k}$ 成词, 则将 $Z_{c1i}Z_2Z_3$ 或 $Z_1Z_{c2j}Z_3$ 或 $Z_1Z_2Z_{c3k}$ 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1; 可纠正错字错误。若 $Z_{c1i} \cdot \text{BeFirst}=1$ 且 $Z_{c1i}Z_2$ 成词或 $Z_{c2j} \cdot \text{BeSecond}=1$ 且 Z_1Z_{c2j} 成词或 $Z_{c3k} \cdot \text{BeThird}=1$ 且 Z_2Z_{c3k} 成词, 则将 $Z_{c1i}Z_2$ 或 Z_1Z_{c2j} 或 Z_2Z_{c3k} 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1; 可纠正多字错误。

(7)若字串 Z 的长度 $\text{Length}(Z)=4$ (以字计), 字串 $Z=Z_1Z_2Z_3Z_4$

若 $Z_{c4l} \cdot \text{BeFourth}=1$ 且 $Z_1Z_2Z_3Z_{c4l}$ 成词 或 $Z_{c3k} \cdot \text{BeThird}=1$ 且 $Z_1Z_2Z_{c3k}Z_4$ 成词 或 $Z_{c2j} \cdot \text{BeSecond}=1$ 且 $Z_1Z_{c2j}Z_3Z_4$ 成词或 $Z_{c1i} \cdot \text{BeFirst}=1$ 且 $Z_{c1i}Z_2Z_3Z_4$ 成词, 则将 $Z_1Z_2Z_3Z_{c4l}$ 或 $Z_1Z_2Z_{c3k}Z_4$ 或 $Z_1Z_{c2j}Z_3Z_4$ 或 $Z_{c1i}Z_2Z_3Z_4$ 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1; 可纠正错字错误。若 $Z_1Z_2Z_4$ 成词或 $Z_1Z_3Z_4$ 成词, 则将 $Z_1Z_2Z_4$ 或 $Z_1Z_3Z_4$ 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1; 可纠正多字错误。

(8)取以 Z_1 为首字和以 Z_n 为尾字且与字串 Z 似然匹配的多字词作为纠错建议, 放入纠错建议缓冲区 $\text{Jianyi.Suggestion}[i]$; 纠错建议计数器 i 加 1, 可纠正多字、漏字、易位、多字替换和部分不能由候选集纠正的错误。

(9)如果 $i=0$; 提示由用户给出纠错建议, 并将纠错建议加入易混淆词词典。

(10)将所得纠错建议排序(见候选建议的排序算法), 取前 20 个作为纠错建议。

(11)转 1。

3.2 候选建议的排序算法:

由纠错候选集和经过似然匹配得出的纠错建议并非都是有效的。在纠错时根据用字信息及上下文相关字词的同现信息计算纠错建议的优先值 ρ , 并根据优先值 ρ 对纠错建议进行排序, 将优先值大的纠错建议放在前面, 可提高纠错效率和可读性。

最小字同现频率排序在一定程度上反映纠错建议的“词频”信息, 设 $S_1S_2 \dots S_m$ 为纠错建议, $r(S_iS_{i+1})$ 为相邻字 S_i 与字 S_{i+1} 的同现频率, 我们令最小字同现频率为优先值 ρ 记为

$$\rho = \underset{i=1}{\overset{m-1}{\text{Min}}} r(S_i, S_{i+1})$$

四、实验结果与纠错盲区

4. 1 实验结果

对含有 284 个真实错误的文本进行纠错实验, 这 284 个错误分别用人工标记和机器自动查错标记, 来计算纠错算法的纠错正确率。纠错正确率定义为纠错算法提供的前五选纠错建议正确的个数占总错误数的百分比。

1. 纠正人工标记出的错误:

对人工标记的 284 个错误点的测试文本进行测试, 纠错算法对 196 个错误点给出的前五选纠错建议正确, 纠错正确率为 69%。

2. 纠正机器自动查错得到的错误:

对经查错算法处理后的含有 284 个真正错误的文本使用该纠错算法, 查错算法查对 199 个错误, 纠错算法给出的纠错建议中前五选正确的有 129 个, 纠错正确率为 43.6%。

4. 2 纠错盲区

通过实验, 我们发现纠错算法存在下列不足或盲区:

(1) 对某些情况不能给出正确的纠错建议

① 姓名、地名错误

如: 蔡元培 1917 年任北大校长。(蔡元培)

爱尔兰 总统决定把给赞比亚的援助减少 30%。(爱尔兰)

② 日期、数字错误

如: 一九二一年二月四五日, 是一个难忘的日子。(四日)

1999 年 10 月我去了北京。(1999)

③ 语法语义级错误和查错算法不能侦测出的错误

如: 使用吸尘器以后, 大大处长了电机寿命。(延长)

看到一位老师深有体会地说。(听)

(2) 候选建议的排除算法有可能排除掉正确的修改建议

如: 但至少这方机的知识能帮助他们在查阅字典后对单词进行理解。(方面)

候选建议排序算法, 给出“方机”的前五选修改建议依次为: “分机、文集、分系、分期、分级”, “方面”列第二十二候选, 会被排除掉。

自动校对要求纠错算法给出纠错建议速度快、准确且数量不能太多。因此, 本文在对错误特征充分分析的基础上, 以纠错候选集结合似然匹配算法, 计算产生候选建议, 并利用最小字字同现频率, 对候选建议进行排序和排除。实验表明, 所给的纠错算法对拼音、五笔字型录入时产生的错误进行纠错是切实可行的。若在实际的纠错过程中, 不断扩大和优化易混淆词典、编码纠错候选集, 纠错效果会更好。

参考文献

- [1] 张仰森、丁冰青, “中文文本自动校对的技术现状及展望”, 中文信息学报, 98. 3.
- [2] 孙才、罗振声, “汉语文本校对字词级查错处理的研究”, 《第四届计算语言学会议论文集》, 97.
- [3] 郭志立等, “中文校对系统中的修改建议提供算法”, 《第四届计算语言学会议论文集》, 1997.
- [4] 于勤、姚天顺, “一种混合的中文文本校对方法”, 中文信息学报, Vol. 12, No. 2, 1998.
- [5] 常新功、夏莹, “用语料库语言学知识指导文本识别研究” 计算语言学进展与应用, 1995 年.
- [6] 张仰森、丁冰青, “一种英文单词拼写查错和纠错的方法——骨架键法”, 电脑开发与应用, 99. 3.
- [7] Joseph J. Pollock “Automatic spelling correction in scientific and scholarly text”
Communication of the ACM 84. 4.
- [8] James L. Peterson “computer programs for detecting and correcting spelling errors”
Communication of the ACM 80. 12.