拼写纠正在拼音输入法中的应用

陈正 李开复
微软中国研究院
北京海淀区知春路 49 号希格玛中心五层 100080

摘要：

中文输入法一直是中文语言研究的一个难题，本文以拼音整句输入法为基础，提出了在中文输入过程中的拼写自动修改。通过对用户输入过程中所犯各种错误进行分析，我们建立了一种有效可行的打字模型，通过收集用户真实输入的数据，统计得到用户的打字模型的参数；同时基于大量的中文文本，训练得到一个强大的中文语言模型，并与中文的打字模型相结合，采用类似语音识别的技术，修改用户输入中的各种错误，并得到最合适的汉字。同时，拼写纠正不仅可以进行用户自适应，而且还适用于各种语言。

关键字：

打字模型，语言模型，Viterbi Beam Search，Trigram，自适应

一、背景

中文输入法一直是中文语言研究的一个难题，传统的中文输入法有两种主流技术，一类是以音为基础的输入，如微软拼音输入法、中文之星等，另一类是以形为基础的输入，如五笔字型等。拼音输入法简单易学，因此成为一种最流行的中文输入方法，大约 97%以上的用户使用拼音输入法[1]。但同时也存在很多问题，如输入的速度，中英文混合输入时的模式切换，以及较高的输入错误对系统的影响等等。

我们提出了一种打字模型来解决以上问题。打字模型是从用户输入的数据中训练得到。在音字转换中，打字模型与中文的语言模型相结合，通过 Viterbi Search 能够得到最合理的句子输出。音字转换系统既能接受用户正确的拼音输入，同时也能容忍合理的输入错误，并能在拼音层次上自动地检测和修改用户的输入错误。

一般来说，用户输入的是一串罗马字母，系统需要把它转换成一串中文字符。用户在输入过程中存在较多的错误，引起这些错误的原因有以下几个方面：

（1）中国人书写时使用的都是汉字，而不是拼音，虽然从小就开始接受拼音的教育，但是并没有外国人使用英语那样频繁。

（2）受到地域的限制，中国有许多方言，不同地方的人的发音存在一定的差异，例如南方人对卷舌－－平舌、前鼻音－－后鼻音的区分就不同于北方人，这也就是常说的“南方模糊音”。

（3）拼音输入不是一种“所见即所得”的输入方法，用户输入的是拼音，而看到的却是系统转换后的汉字，因此一般情况下用户检查的不是输入的拼音，而是转换后系统显示的汉字。

（4）对键盘的熟练程度以及输入的速度都直接影响到用户输入的准确度。在过去的一段时间内，许多心理学家都通过实验得到用户输入的混淆矩阵。在整句输入中，如果用户敲错一个拼音，不仅这个拼音对应的汉字会出错，

而且还会影响到周围的一些汉字，引入了更多的错误；而且，当用户看到错误的汉字时，并不知道是系统发生的错误还是用户输入存在错误，用户需要找到错误的地方进行修改，才能得到正确的结果。这给用户带来很大负担，如果系统能够自动地在拼音层次上进行拼写检查和修正，那就可以减少用户修改的次数，提高输入的速度。

二、理论基础

解决“拼写纠正”的方法是采用“语音识别”的技术。在语言识别中，需要解决的目标是给定一段语音 A，寻找一串汉字 W，使得 $\Pr(W)$ 达到最大。利用贝叶斯公式，可以把问题转换为 $\Pr(W \mid A) = \Pr(A \mid W) * \Pr(W) / \Pr(A)$。其中 $\Pr(W)$ 是语言模型，$\Pr(A \mid W)$ 是声学模型，由于不同的人对同一个汉字的发音不同，因此引入声学模型来刻画不同的人读一个汉字时的不同特征。

拼音汉字转换的目标则是给定一串拼音 P，寻求一组汉字串 H，使得 $\Pr(H)$ 达到最大。类似语音识别算法，利用 Bayes 公式，可以把问题转换为 $\Pr(H \mid P) = \Pr(P \mid H) * \Pr(H) / \Pr(P)$。由于在具体问题求解时，$P$ 是给定的，也就是说 $\Pr(P)$ 是一个常数，因此问题可以简化为寻找 H，使得 $\Pr(P \mid H) * \Pr(H)$ 达到最大。理论上，我们可以穷举所有可能的 H 组合，当然在具体系统实现时采用的是一些有效的搜索算法，如 Viterbi Beam Search [2,3]。

$\Pr(H)$ 是语言模型，它能给出一个给定的汉字串的概率，通常情况下，可采用“基于统计的语言模型”来实现。$\Pr(P \mid H)$ 是打字模型，表示对于给定的汉字 H，用户打为 P 的概率。从本质上来说，打字模型和声学模型十分相似，他们刻画的都是用户对于一个汉字的不同表示方法，声学模型刻画的是发音的特征，而打字模型刻画的则是用户输入的特征。

由于 H 是一些汉字的组合，可分解为 w_1, w_2, Λ, w_n，其中 w_i 可以是字或者是词，因此 $\Pr(P \mid H) \approx \prod \Pr(P_{f(i)} \mid w_i)$，其中 $P_{f(i)}$ 是对应于 w_i 的拼音序列。传统的拼音汉字转换系统不考虑用户敲错键的概率，因此如果把多音字看作不同的字，那么 $\Pr(P_{f(i)} \mid w_i) \equiv 1$，其中 $P_{f(i)}$ 是 w_i 的正确发音，因此求解问题转换为寻找 H，使得 $\Pr(H)$ 最大。另外还有一些系统采用“南方模糊音”来解决这个问题，把一些常混淆的音强制捆绑在一起，这虽然能够解决一部分问题，但是仍然存在大部分的输入错误无法得以解决，我们是从实际用户的输入数据中统计得到 $\Pr(P_{f(i)} \mid w_i)$。

打字模型可有多种描述方法，理论上我们可以通过统计得到所有的 $\Pr(P_{f(i)} \mid w_i)$，但问题的规模太大，因此我们将问题简化为寻找每一个可能的中文音节可能输入变形，中文共有 406 个音节，因此统计 $\Pr(PinyinString \mid Syllable)$ 是可以实现的。当然，这种表示方法问题的问题还是很大，需要很多的训练数据才能够得到可信的结果，因此在实际系统中采用我们下面提到的方法将更为有效可行。

根据心理学家对输入人员所犯的错误的分析结果可以得知，用户在输入过程中常犯的错误有以下几种：

（1） **替代错误**：即输入中把一个键敲成另外一个键，这种错误一部分是由于键盘的布局造成的，输入者容易把正确的键击打成键盘上与其相邻的那些键。另外一部分错误是由输入人员的思维引起的，他们在输入时容易把左右手映射的键敲错，即把左（右）手应该击打的键击打成相应右（左）手击打的键。心理学家通过实验统计发现，键盘上横向相临的键被敲错的概率占所有输入错误的 43%，纵向相临的键被敲错的概率占 15%，左右手映射错误的占 10%，其余大约占 32% [4]。

（2） **插入错误**：即输入中有可能在某些地方插入一些不应该出现的字符。这种错误一部分是由键盘的布局造成，对键盘不太熟悉的用户（在中国这种现象比较普遍）在输入时手指可能按在两个键的中间，同时击打两个或多个键；另外一部分错误是由于输入人员对拼音的熟练程度引起的，例如南方人对‘zh’－‘z’、‘sh’－‘s’、‘ch’－‘c’、‘ng’－‘n’的混淆，导致输入中可能引起一些插入错误。

（3） **删除错误**：即输入中某些字符被漏敲，引起这种错误的原因主要来源于输入人员，输入人员对拼音的熟练程度，以及他们思维的跳跃都可能引起一些字符被漏敲。

（4） **其他类型的错误**：除了以上三种常见的错误之外，输入中还有可能遇到这些错误，如输入者思维先后次序的颠倒引起输入的字符的先后次序发生颠倒，实验统计结果表明这种类型的错误占总错误的 15%。

由于打字模型和声学模型具有很多相似之处，因此在构造打字模型是我们可以采用类似声学模型的构造方法。在语音识别中，目前比较流行的是采用隐元马尔可夫模型（HMM）方法，对于每一个音素，用一个 HMM 来表示，同一个音素的每个发音样本对应于 HMM 的一个状态序列，状态与状态之间的迁移可以通过实际的数据训练得到。同样，在构造打字模型时，我们也可以把用户输入拼音时的每个键看成是一个状态，用户在输入同一个音时击打的键可能不同，我们可以根据用户实际输入的数据训练各个状态之间的转移概率，得到用户击打不同的拼音的特征。因此我们就可以得到用户输入所有拼音的打字模型，由于中文共有 406 个音节，因此实际训练中就需要很多的训练数据，为了简化模型，我们把不同音节中相同的字母捆绑在一起看成一个状态[5]，因此整个模型的状态简化为 27 个，（其中 26 个状态为英文字母，另外一个代表一些不认识的字符）。

根据上面提到的几种错误类型，同时考虑到系统实现的有效性，我们设计了一个简单有效的打字模型，模型分为三个部分：

（1） **替代模型：**

$\Pr(a\ typed\ as\ b)$，代表字符 a 被敲成字符 b 的概率，可用实际统计得到的数据来表示，$\Pr(a\ typed\ as\ b)=\frac{Count(a\to b)}{Count(a)}$，其中 $Count(a\to b)$ 表示在实际数据中字符 a 被敲成 b 的次数，$Count(a)$ 表示真实数据中字符 a 出现的次数。

（2） **插入模型：**

$\Pr(a\ inserted\ after\ b)$，代表字符 b 之后插入字符 a 的概率，可用实际统计得到的数据来表示，$\Pr(a\ inserted\ after\ b)=\frac{Count(b\to ba)}{Count(b)}$，其中 $Count(b\to ba)$ 表示在实际数据中在字符 b 之后插入一个字符 a 的次数，$Count(b)$ 表示真实数据中字符 b 出现的次数。

（3） **删除模型：**

$\Pr(a\ deleted\ after\ b)$，代表字符 b 之后删除字符 a 的概率，可用实际统计得到的数据来表示，$\Pr(a\ deleted\ after\ b)=\frac{Count(ba\to b)}{Count(ba)}$，其中 $Count(ba\to b)$ 表示在实际数据中在字符 b 之后删除一个字符 a 的次数，$Count(ba)$ 表示真实数据中字符 ba 出现的次数。

虽然以上的打字模型不能刻画所有的输入错误，但他十分有效，模型的参数个数只有 27*27*3=2187 个，因此可以从较少的真实数据中统计得到。在实际训练时，我们用 Viterbi Search 把正确的拼音和用户输入的拼音进行匹配，得到一个最优的切分，然后分别训练上面三个打字模型。即使是这样，我们仍然遇到了数据不足的问题，有些情况在真实数据中仅发生少数几次，其数据并不可信；因此我们采用了类似“语音识别”中的 MAP 技术 [2]，先利用一些先验的知识构造一个“打字模型”SPM_A，然后从真实数据中统计得到真实的“打字模型”SPM_B，然后把这两个模型进行插值，得到平滑后的打字模型 $SPM=\lambda SPM_A+(1-\lambda)SPM_B$。

三、中文语言模型

在语言模型方面，我们采用的基于统计的 N 元马尔可夫语言模型，通常使用的有二元文法（Bigram）和三元文法（Trigram）的语言模型[6]。

Bigram: $p(w_i \mid w_{i-1}) = \lambda_1 f(w_i \mid w_{i-1}) + \lambda_2 f(w_i)$

Trigram: $p(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 f(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 f(w_i \mid w_{i-1}) + \lambda_3 f(w_i)$

对英文来说，Trigram 已被广泛使用，然而对中文来说，许多人认为 Bigram 已经足够，没有必要引入 Trigram。通过收集大量的中文文本，其中包括常见的报刊杂志，如人民日报、科技日报等，平衡的语料库，以及大量 web 上下载的数据(约 1.6G)，通过筛选得到训练语料，然后训练得到 Trigram 语言模型。在实际应用中，我们采用困惑度 PP（perplexity）[2,6]来评估统计语言模型的性能，困惑度的度量是从信息熵的角度得到。困惑度可以表示为：

$PP = 2^{LP} = 2^{-\frac{1}{n}\log p(w_1, w_2, \Lambda, w_n)}$，其中 $p(w_1, w_2, \Lambda\ w_n)$ 可以简化为 Trigram 模型，

$$p(w_1, w_2, \Lambda, w_n)$$
$$= p(w_1 \mid < s >) \cdot p(w_2 \mid w_1, < s >) \Lambda\ p(w_i \mid w_{i-2}, w_{i-1}) \Lambda\ p(< / s > \mid w_{n-1}, w_n)$$

其中，<s>、</s>代表句子的开始和结束。

由于中文存在分词问题，因此 PP 即可以用字的 PP 来表示（表示为 PPc），也可以用词的 PP 来表示（表示为 PPw）。如果用 PPc 来度量，那么公式中的 n 就表示为文本中所含的中文字的个数；如果用 PPw 来度量，那么公式中的 n 就表示为文本中所含的中文词的个数。应该说，PP 值越小，就表示语言模型用于识别时的候选字/词的个数就越少，对系统的帮助就越大。

通过实验，我们证明当语言模型的大小超过一定的限制（10M 左右），Trigram 就比 Bigram 有比较显著的进步，而且通过对确定领域进行自适应，我们能够得到更好的语言模型。

我们选择了五种类型的文本语料(A - E)，然后分别训练得到五个语言模型，测试它们之间的困惑度，然后把他们混合起来训练得到一个混合的语言模型(Mixed)，然后测试这个模型对各个领域(A - E)的困惑度。结果如表一所示：同一个领域的困惑度比跨领域的困惑度要小很多，而且语料越多，模型的效果也就越好。

语料库	A	B	C	D	E	Mixed
A	21	34	65	83	74	36
B	50	25	55	71	62	27
C	98	42	32	73	74	33
D	187	139	175	58	156	87
E	96	54	72	84	57	43
Average PPc	71	49	68	73	79	41

表一：不同领域之间的困惑度(LM: Trigram)

四、实验

打字模型是从用户真实输入的数据中统计得到的。我们设计了一个打字实

验，屏幕上显示需要输入的汉字，让用户在一定时间内输入其对应的拼音，系统自动记录用户击键的次序以及击键的时间。通过统计得到用户输入速度的曲线，然后去掉两头不合理的数据，把剩余的数据进行统计得到打字模型。我们在北京收集了 100 个用户*8 小时的输入数据，我们把其中 90%的数据作为训练，剩余的 10%的数据作为测试，搜索的方法采用 Viterbi Beam Search，中文语言模型采用的是 Trigram 模型，测试文本的字的困惑度为 66.69，词的困惑度为 653.71。

实验一：

首先，我们测试系统原型（不含有拼写纠正的功能）音字转换的错误率。输入数据分为两组，一个输入是完美的拼音（不含有任何错误），另外一个输入是用户实际输入的拼音。实验结果如表二所示：

	Error Rate
Perfect Input	6.82%
Actual Input	20.84%

表二：不含有拼写纠正功能的系统

统计测试数据，把用户输入的拼音和正确的拼音进行动态匹配，发现用户输入的拼音的罗马字母有 3.4%发生错误，拼音被敲错的中文字符占总字符的比例为 4.6%。但是这些被敲错的拼音不仅自身转换会发生错误，而且还会影响到其周围的汉字转换也发生错误，引起错误的传播，最后转换的错误率的上升幅度基本上是原先的 3 倍。

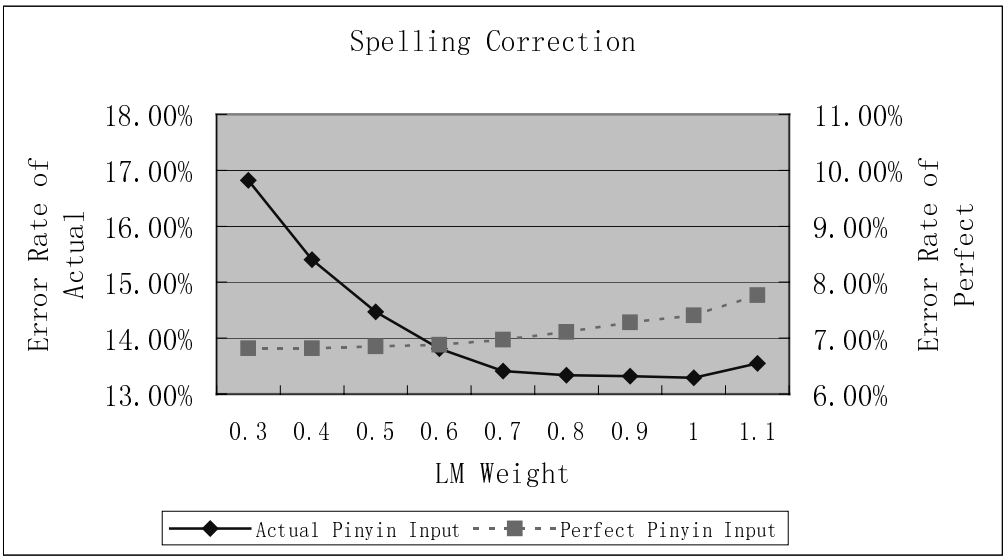
由于实际应用中为了平衡打字模型的概率和语言模型的概率，我们引入一个因子，称之为“语言模型权重”(LM Weight)[6,7,8]，因此求解问题转换为寻找 H，使得 $\Pr(P \mid H) * \Pr(H)^{\alpha}$ 达到最大，其中 α 为 LM Weight。如果转换为对数空间，即需要使得 $\log \Pr(P \mid H) + \alpha * \log \Pr(H)$ 得到最大。

实验二：

我们对具有拼写纠正功能的系统进行测试，输入数据与实验一的输入数据相同，实验结果如表三、图一所示：

LM Weight	Error Rate	
	Perfect Input	Actual Input
0.3	6.82%	16.82%
0.4	6.82%	15.40%
0.5	6.85%	14.47%
0.6	6.88%	13.81%
0.7	6.97%	13.41%
0.8	7.11%	13.34%
0.9	7.28%	13.32%
1.0	7.41%	13.29%
1.1	7.77%	13.55%

表三：含有拼写纠正功能的系统



图一：不同的 LM Weight 以及不同质量的输入对系统性能的影响

选择不同的α，对系统造成的影响也不同。α越大，系统就更相信语言模型的作用，能够纠正更多的错误，当然引入的副作用也就越大；α越小，则系统更相信打字模型的作用，那样造成的副作用就很小，但是概率较小的错误就难以得到纠正。从图一中我们可以看出，随着α的增加，系统的副作用也随着增加，但是对于实际输入的数据，系统的错误率是先下降到一定程度，然后才开始上升，这也就说明在开始的一段时间内，系统纠正错误的能力大于副作用，但当α增加到一定程度，副作用达到一定程度，系统的总体性能就开始下降。

从上面的数据我们也可以得出另外一个结论，对于不同的用户我们可以通过调整α来达到最优的效果。不同的输入人员的输入能力不同，输入的错误率相差也较大，如果对所有的用户采用统一的α，那么结果并不是最好的。如果能够自动判断用户的输入能力，自动调整α，那么能够得到最优的转换结果。因此我们可以在输入系统中达到用户的自适应。

我们从实验用户中挑选了 3 个用户，加上一个理想用户（假设这个用户在输入过程中不存在任何错误），我们选用不同的α进行测试，结果如表四所示：

	α_1=0.5	α_2=0.8	α_3=1.1	*Dynamic* α
User 0	6.85%	7.11%	7.77%	6.85%
User 1	8.15%	8.23%	8.66%	8.15%
User 2	13.90%	12.86%	12.91%	12.86%
User 3	19.15%	18.19%	17.77%	17.77%
平均	12.01%	11.6%	11.78%	10.16%

表四：用户自适应

其中：User0 是理想用户，输入中不存在任何错误，User1 输入平均错误率为 0.77%，User2 输入平均错误率为 4.41%，User3 输入的平均错误率为 5.73%。

从表四的结果我们可以看出，如果对不同的用户采用统一的α，并不能得到

最优的结果，如果我们能够根据用户在输入过程中的修改操作来判断用户的输入技能，自动的调整α，就能得到最好的转换结果。在实际应用中，对于一些熟练而且不容易犯错的用户，系统可以设置较小的α，而对于那些新手或者是经常击键错误的用户则可以设置较大的α。判断用户输入技能可以根据以下几个因素：

（1） 用户在输入过程中修改的次数占击键次数的比例。

（2） 根据用户输入的文本，统计输入时使用的手指的难易程度，计算出平均需要的输入时间，然后根据用户实际输入时间判断其熟练程度。

五、结论

拼音输入法是一种被人们广泛采用的中文输入方法，在整句输入的基础上，本文引入了输入过程中的拼写纠正，使得系统可以容忍一定程度的错误输入，并能自动地纠正这些错误。实验表明拼写纠正不仅提高了系统的转换准确率，转换错误率比原先下降 35%，而且提高了用户的输入速度。并且系统能够自动地适应用户的输入，达到最优的效果。此外，拼写纠正不仅可以应用于中文输入，对于日文、英文等其他语言也一样适用，通过收集不同的输入数据，构造不同的打字模型，就可以达到多语言共用。

参考文献：

[1] 陈原 主编，汉语语言文字信息处理，上海教育出版社，1997.12，pp.4。
[2] Kai-Fu Lee, Automatic Speech Recognition, Kluwer Academic Publishers, 1989.
[3] Chin-Hui Lee, Frank K. Soong, Kuldip K. Paliwal, Automatic Speech and Speaker Recognition -- Advanced Topics, Kluwer Academic Publishers, 1996.
[4] William E. Cooper, Cognitive Aspects of Skilled Typewriting, Springer-Verlag New York Inc., 1983.
[5] Hwang, M.Y.;Huang, X., Subphonetic modeling with Markov states-Senone, ICASSP-92., 1992, vol.1, pp33-36.
[6] Frederick Jelinek, Statistical Methods for Speech Recognition, The MIT Press, Cambridge, Massachusetts, 1997.
[7] Bahl,L., Bakis, R., Jelinek, F., and Mercer, R. Language Model / Accoustic Channel Balabnce Mechanism. IBM Technical Disclosure Bulletin, vol.23(1980), pp. 3464-3465.
[8] X. Huang, M. Belin, F. Alleva, and M. Hwang, Unified Stochastic Engine (USE) for Speech Recognition, ICASSP-93., 1993, vol.2, pp. 636-639.

Spelling Correction in Pinyin Input

Chen Zheng, Kai-Fu Lee
Microsoft Research ,China
5F, Beijing Sigma Center
No.49, Zhihun Road Haidian District
Beijing 100080, P.R.C.

Abstract:

Chinese input method is one of the difficult problem of Chinese Language Processing. Base on sentence-based pinyin input method, we propose a new typing model to solve this problem. After analyze the most popular errors which made by typists, we build a typing model. The typing model is trained on real data, and learns probabilities of typing errors. We also train a powerful Chinese language model based on large corpus. In the Pinyin-to-Hanzi conversion, the probabilities of typing model are combined with the language model probabilities, to find the most probable interpretation of a sequence of Roman letters typed. Further more, spelling correction can automatically adapt to typist, and it is applicable to any language.

Keywords:

Typing Model, Language Model, Viterbi Beam Search, Trigram, adaptation