一种基于 Tri-Gram 模型的拼音输入法的设计与实现

吴鸿智

0272408

本文全文使用鸿智拼音书写

2005 年 1 月 29 日

§1 简介

输入法是人机交互过程中非常重要的一个环节，要使用键盘输入汉字，最根本的是要解决如何利用 26 个英文字母键、10 个数字键和成千上万个汉字对应的问题。目前为止，人们先后开发了以拼音输入法、郑码输入法、自然码输入法和表形码输入法等为代表的多种输入法。

对于任何一个汉字而言，它都具有 3 个基本的要素，即音、形、义。所谓音是指汉字的读音，所谓形是指汉字的结构 (即笔画组成)，而所谓义是指该汉字的意义。因此，要使用键盘输入汉字，所依靠的也不外乎这 3 个要素，即如何通过汉字的音、形和义来建立汉字与键盘按键的对应关系。

- 拼音输入法：使用键盘输入汉字的最直接方法是使用拼音输入来输入汉字，因为我国使用的汉语拼音即脱胎于 26 个英文字母。拼音输入法的优点是易学，一个人只要会汉语拼音，就可以使用拼音输入法，但缺点是输入速度太慢，因为汉字中同音字较多，故用户使用该种输入法始终要面对一个选字的问题，这样就降低了输入速度。拼音输入法的代表有微软拼音输入法、全拼输入法、简拼输入法、双拼输入法、智能 ABC 输入法等。
- 拼形输入法：汉字是由多种偏旁部首组成的，其最小单位为 5 种笔画，即横、竖、撇、捺、折。可以根据汉字的这种特点来实现输入。其方法是将每个英文字母键定义成多个偏旁部首，用户在输入汉字时只需按书写顺序按下各偏旁部首所对应的按键即可。这类输入法的优点是重码率低，同时它还可以输入词组，其缺点是需要记忆不少内容，所以学习起来不如汉语拼音那么容易。因此，这类输入法可供专业打字或录入人员使用。拼形输入法的代表有郑码输入法、表形码输入法和五笔字型输入法等。
- 音形结合输入法：如前所述，汉字除了具有音和形两个基本要素外，还有一个“义”要素。拼音输入法的优点是学习简单，缺点是效率不高。拼形输人法的优点是输入效率高，缺点是学习困难。为此，人们又开发出了音、形、义相结合的编码。这方面比较突出的代表为自然码，其特点是词汇输入采用拼音，而单字输入则采用声、韵、义、形来组字。

虽然有诸多缺点，但目前使用最广泛的中文输入法仍然是拼音输入法，这主要是由于它的直观和易学。再加上微软拼音输入法的出现，拼音输入法与其他输入法相比最大的缺点－重码率高的问题也被大大缓解，这从根本上解除了阻碍拼音输入法广泛流行的最大障碍。

本文实现了一个基于 Tri-Gram 模型的拼音输入法－鸿智拼音，通过对语料库的分析统计得到 Tri-Gram 概率模型，并能由此直接将拼音序列转化成概率最大的对应中文句子，尽可能降低用户选择重码的次数，以提高拼音输入法的效率。

§2 Tri-gram 模型

困扰拼音输入法最大的问题就是重码问题，即一个拼音序列有多个不同的词或句子与之对应。然而往往有意义的句子只占很少的比例，我们就通过 Tri-gram 模型来选出可能性最大的那一个。本节主要介绍 Tri-gram 的背景知识。

对于一个句子，$S = w_1w_2...w_l$，计算 $p(S)$ 概率如下

$$p(S) = p(w_1)p(w_2|w_1)...p(w_l|w_1w_2...w_{l-1}) \tag{1}$$

由于样本集的限制，“历史”不能太长，否则会导致模型参数太多。

§2.1 n-Gram 模型

当前出现哪个词，仅仅与前面的 $n-1$ 个词有关。即假设

$p(w_i|w_1w_2...w_{i-1}) \approx p(w_i|w_{i-n+1}w_{i-n+2}...w_{i-1})$

由此，

$$p(S) = \prod_{i=1}^{l} p(w_i|w_{i-n+1}w_{i-n+2}...w_{i-1}) \tag{2}$$

一般而言，n 越大，区分度就越好；n 越小，可靠性更佳；实际应用中往往需要折中考虑。并且随着n 的增大，数据集大小会按指数级增长，这也是实际应用中需要考虑的一个问题。

当 $n = 3$ 就是 Tri-gram 模型。

§2.2 概率平滑

因为参数空间太大，数据集大小小于参数空间，所以原始的 Trigram 模型估计，会有很多概率为 0 的情况。这会在计算句子概率时带来不好的影响，因为只要有一个概率为 0，整句的概率就变成 0。为了解决这个问题，我们可以从已发生的事件的概率中扣除一部分，然后重新分配这些扣出来的概率给那些未发生的事件（和已发生事件），这就是概率平滑的基本思想。具体的方法有 Good-Turing 估计法、退化法和线性插值法等。

§3 实验方法

§3.1 实验装置

硬件环境：Dell Inspiron 8200 (P4 1.6G Mobile, 512M DDR RAM, 20G HD)

软件环境：Windows XP SP2, Microsoft Visual Studio .Net 2003, MSDN Oct 2004., Adobe Photoshop 6.0, axialis Icon Workshop 5.0

语料库：人民日报 98 年 1 月熟语料、procsina 和 procchina 共三个已分词的中文语料库 (总的 Word Occurence 为 15,831,234)。从 Win98 下的全拼输入法导出的字到拼音的码表。

§3.2 数据结构

对于拼音，我们采用通常的英语 Trie 树来保存。而对于汉字词语，是无法直接用英语的 Trie 树来保存，其内存消耗太大。我们使用了改进型的 Trie 树，具体而言，就是每个节点有一个动态数组，这个数组里保存其子节点的指针和子节点所代表的汉字，并按汉字的 Unicode 大小排序。每次添加一个词组时，从根节点出发按字在相关节点的动态数组里进行二分查找，如果最终全部匹配，则该词组已经在树中。否则沿着查找路径向相关动态数组中进行添加。

§3.3 实验步骤

预处理步骤

1. 从全拼输入法导出的码表极不规范，有许多非标准的拼音 (如 uu 等)。首先人工去掉这些拼音。
2. 然后统计出所有的拼音并记录为 PYCard.txt。
3. 接着，统计语料库中的字频，按照统计出的字频大小，把原始的字到拼音的码表转换成拼音到字的简单码表 SimpleMB.txt，每个拼音中的字按其字频高低降序排列。
4. 统计语料库中的词频，去除词频过小的词语，剩下的词语保存为词表 WordCard.txt。
5. 统计词表中出现的词的 Trigram 概率，记录为 3-gram.txt。
6. 为词表中的所有词语注音。如果词语在原始码表中出现，直接使用原始码表中的拼音。否则用动态规划求用数量最小的原始码表中存在的词语对当前词语进行分解的方案，如果存在多音字，按照该字在原始码表中出现次数最多的拼音进行标注。结果保存为 PYWordList.txt。

我们就输入过程中涉及的一些术语假设如下：CompStr 为以转换的词语序列，PreCompStr 为还未转换的拼音字母。我们规定 PreCompStr 中至多有 3 个拼音。Candidate 为当前 PreCompStr 可能对应的所有词语集。

我们实现了标准 IME 接口的输入法程序，初始时调入所有预处理数据。具体输入过程如下：

1. 输入的拼音字母先保存在 PreCompStr 中。递归求解拼音数量最少的拼音字母序列划分方案。如果拼音数大于规定的数目，拿走头部多余的拼音，和已转换的拼音一起，首先递归求出所有可能的对应词语划分。然后用加约束条件的递归方法，通过访问相应词语 Trigram 的记录，求解最大概率的词语序列保存到 CompStr 中。(我们使用 Additive Smoothing 进行概率平滑，以避免 0 概率的出现。并且为了防止做过多不必要的计

Word1	Word2	Word3	频率
一	不	留神	1.076923e-001
一	个	一	1.092443e-003
一	个	三	6.027273e-004
一	个	上午	1.130114e-004
一	个	上市	4.897160e-004
一	个	下午	3.767046e-004
一	个	不	5.236194e-003
一	个	不可	4.897160e-004
一	个	不容	1.883523e-004
一	个	不折不扣	6.403978e-004

表 1: 部分 Trigram 统计结果

算，我们只对当前输入位置向前六个词的范围重新进行概率计算。)

2. 对当前 PreCompStr 的拼音划分方案，取其所有前缀按照 Trigram 的概率大小排序加入到 Candidate 中去。
3. 用户可以一次输入完一个句子的所有拼音，让输入法自动转换后，再回退修改（如果有错的话）。也可以以词组为单位依次输入，在 Candidate 中进行相应选择（如果默认选择错误的话）。注意我们规定基于 Tri-gram 的算法不处理已被用户选择过的词语。

§4 实验结果和分析

§4.1 预处理

经统计，所有合法的拼音总数为 406 个 (为了保留上海方言" 嗲"，我们多加了一个非标准拼音 dia)。统计词表时，我们设定的频率阀值是 5，一共记录了 48,213 个词。

Tri-gram 统计结果中，一共有 631,091 个概率值：其中 Tri-gram 有 184,645 个，Bi-gram 有 404,988 个，Uni-gram 有 41,458 个。部分结果如表 1。

为了加快输入法载入预处理数据的速度，我们把 3-gram.txt, PYWordList.txt 和 WordCard.txt 由文本文件转换成更为紧凑的二进制文件，并且记录内容的是相应的 Trie 树。

§4.2 输入法

由于 VC 对动态申请大量小块数据的内存利用率非常低，我们不得不自己写了内存分配例程为 Trie 树节点和其他小规模但数量较大的数据结构分配内存。这一改进使输入法总的内存需求从原先的 233M 降到 40M。

语段	错字率	错字数	总字数
1	0%	0	129
2	11%	11	99
3	20.5%	16	78
4	5.4%	2	37

表 2：不同语段输入结果比较

我们对输入结果的测试都是采用一次输入的方法，不进行任何选择，以错字数来衡量输入的效果。不同语段之间输入效果的比较详见表 2。

输入结果 (带下滑线的是错误的部分)：

§4.2.1 第一段

我们要高举马列主义、毛泽东思想和邓小平理论三面伟大旗帜，在江泽民同志三个代表重要讲话的指引下，紧密团结在以胡锦涛同志为核心的党中央周围，同心同德，艰苦奋斗，为早日把我国建设成为中等程度的发达国家，为实现中华民族在二十一世纪的伟大复兴贡献出自己应有的一份力量!

分析：由于我们训练的语料库主要来自人民日报，所以用上面这段文字测试对已训练素材的输入效果。结果一个字也没有错，说明 Tri-gram 模型对已训练的材料的输入结果还是不错的。

§4.2.2 第二段

大家好，我是吴洪志，复旦大学计算机科学与工程系的本科省。我非常喜欢中文信息处理这门客，在短短一学期的课程中，学到乐很多很使用的只是，也作乐不少很有趣的项目，希望这们可能越办越好，黄老师越来越漂亮：）

分析：对一般文字（即不是太接近训练语料，主要由语料库中的词构成，但用词方法有所不同）的输入效果。对未登录词” 吴鸿智” 虽然没有也不可能猜对，但其结果还算情理之中 (之所以出现” 洪志” 应该是因为有关法轮功的报道)。对另外一些词 (实用，知识等) 的区分度不是很好。总体效果一般。

§4.2.3 第三段

与前面无片估计的讨论类似，一般来说，一直最消防查估计是存在的。担当我们局限与无片估计的自类中考察是，由于缩小乐考虑的范围因此常常能找到一直最消防查无片估计。

分析：这是概率论书上的一段文字，测试与语料库相差比较大的材料的输入效果。除了专有名词和” 了” 之外，余下的大多数词语还是正确的，这说明 Tri-gram 方法对未训练数据的输入效果，特别是专有名词的输入效果并不好。

§4.2.4 第四段

鸿智拼音 五星红旗迎风飘扬，展示高唱多么昂扬。歌唱我们亲爱的祖国，共同走向繁荣富强。

微软拼音 3.0 五星红旗迎风飘扬，展示高唱多么昂扬。歌唱我们亲爱的祖国，共同走向繁荣富强。

分析：测试中遇到的有趣现象。两种输入法错的完全一样，这从一方面证明了微软拼音使用了基于 Tri-gram 算法。

§5 未来展望

由于临近期末考试，这次大作业的时间很紧张，很多想做的功能没来得及做。目前是用最小数目匹配把从拼音字母序列转换到拼音，其实应该是做成按概率来转换的。另外对原始码表中未注音词语进行自动标注的结果存在一定错误，未来可以用更好的算法或者直接进行人工标注。而且当前鸿智拼音 40M 的内存需求太大，主要是由于一次把所有 Tri-gram 读入内存造成的。可以通过设置一个 Tri-gram Cache，内存中只存放最近引用的少量记录，其余全部留在磁盘上，需要时再从磁盘中读入。这样可以减少大量内存需求，加快输入法启动速度，而且相应的开销也不会太大。还有一个很重要的方面没有实现就是自学习功能，可以自适应用户输入的新词，并对 Tri-gram 做相应的更新。

§6 软件介绍

安装完鸿智拼音后，不断按 Ctrl+Shift 直到出现如图所示的状态窗口。窗口中” 中/英” 代表当前的中文/英文输入状态，一个整圆/半圆图标代表当前的全角/半角输入状态，整圆/半圆图标上的白色”L” 代表 LATEX 输入风格的状台。

输入窗口分三部分：最上面一行是 CompStr，即已转换好的词语 (它们可能随着上下文的改变而变化)。中间一行是 PreCompStr，即最近键入的拼音字母序列。最底下一行是 Candidate，即 PreCompStr 所对应的词语，按可能概率高低降序排列。

快捷键如下：

切换中文/英文输入状态 Shift

切换全角/半角输入状态 Shift+ 空格

切换 LATEX 输入风格 Shift+Enter

注：用过 LATEX 的人都知道其编辑软件 WinEdit，需要按照西方的习惯，在输入每个标点后键入一个空格才能进行正常排版。这很不符合中国人的习惯。为此，我们特地设计了”LATEX 输入风格” 这样一个功能。当它开启

图 1: 输入窗口

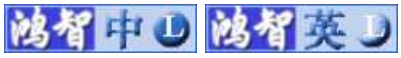

图 2: 状态窗口

时，每一个中文标点后都会自动加上一个空格，方便了中文论文的撰写。

§7 参考资料

1. 黄萱菁：中文信息处理课件
2. FreePY 源代码
3. MSDN Oct 2004