Java 拼音输入法的整句输入实现

彭一凡 张猛
北京大学言语听觉研究中心

摘　要：拼音输入法是现今最主流的中文输入法实现方式。本文使用 N-gram 语言模型，通过拼音切分、构建词网、Viterbi 解码等过程，设计并实现了基于 Java 输入法框架的 Java 拼音输入法。实验证明，本文采用的解码方法能够快速、有效的处理整句输入问题。并且借助 Java VM 的特性，该输入法具有跨平台、易用、可移植的特点。
关键词：Java 拼音输入法，N-gram 语言模型，Viterbi 算法

Design and Implementation of Whole Sentence Java Pinyin Input Method

Abstract: Pinyin input method is currently the main Chinese Input Method. The paper proposes a novel Java Pinyin Input Method which has Pinyin segmentation, word net building and Viterbi decoding. From the experiment, this decoding method can process the whole Pinyin input sentence fast and efficiently. Based on advantages of Java VM, the Java Pinyin Input Method described in this paper is cross-platform, easy-use and portable.
Keywords: Java Pinyin input method, N-gram language model, Viterbi algorithm

1 引言

中文输入方法是中文信息处理的重要课题。现有流行中文输入法是以键盘为输入设备的拼音输入法。输入法应用程序依赖于操作系统提供的接口框架。在 Windows 操作系统中，一般使用 IMM-IME（输入法管理器-输入法生成器）[1]。在 Linux 操作系统中则为 SCIM（Smart Common Input Methods）。另外还有支持 Solaris/OpenSorlaris 的 IIIMF（Internet/Intranet Input Method Framework）和支持 Mac OS 的 OpenVanilla 等。可以看出，在操作系统层面上，尚没有一种通用的输入法框架。

在虚拟机层面，Java VM 的提出为跨平台输入法的实现提供了一种可能。Java 开放、跨平台、易用的特点，使其迅速成为全球主要的软件开发平台之一。但是，如果 Java 应用程序仅采用系统输入法，会限制其跨平台的特性，无法满足国际化应用程序的需要。因此作为 Java VM 的一部分，Java 输入法框架为文本编辑组件提供了独立于平台的 API。由此开发的输入法可以在任何 Java 应用程序中运行，并且无需修改或重新编译文本编辑组件。正式基于以上的优点，本文设计并实现了 Java 拼音输入法，并希望其对以 Java 为核心的移动平台输入法也具有参考价值。

拼音输入法的核心问题是同音字多，重码率高，输入效率低。针对这个问题，本文采用的解决策略是借助自然语言处理技术，引入语言模型进行整句解码。对于用户输入的拼音串，本输入法首先根据规则将拼音串进行切分，然后通过发音词典构建词网，最后使用主流的 N-gram 语言模型，通过 Viterbi 算法进行解码，得到最优路径。

本文以下部分按照输入法的解码顺序组织：第 2 节描述 Java 输入法框架，第 3、4、5 节介绍拼音流的切分方法，然后根据 N-gram 语言模型描述本文采用的解码算法。第 6 节给出实验结果，最后是本文的结论。

2 Java 输入法框架

Java 开发工具包（JDK）提供了 Java 输入法框架。该框架使文本编辑组件的开发人员能够与输入法进行交互，从而能够进行大字符集语言的文本输入。Java 输入法框架如图 2 所示。其中加重框表示 JDK 已经实现的部分。

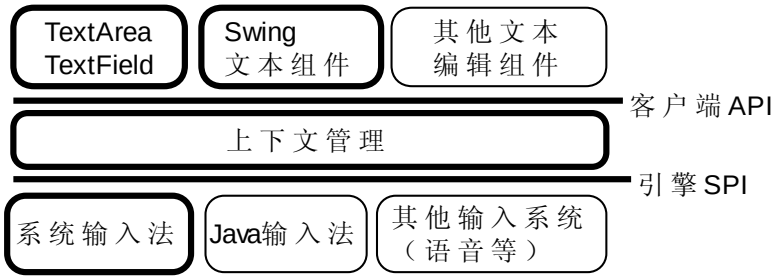


图 2　　Java 的输入法框架[2]

输入法 API 定义了用于文本编辑组件实现文本输入的类和接口。AWT 文本组件 TextArea 和 TextField 和 Swing 文本组件是输入法的客户组件。上下文管理功能管理着文本编辑组件与输入法之间的通讯路径。JDK 主要使用与主机输入法管理器集成的本地输入法。适配器将对本地输入法所用数据模型和输入法框架之间的信息进行转化。

本文输入法的界面（User Interface, UI）设计与现今主流的输入法类似，而且同时支持键盘和鼠标操作。输入法界面如图 3。输入法通常包括 3 种界面。

- 写作窗口，用于显示用户的输入以及产生的候选，并且可以让用户选择正确答案。
- 状态窗口提供了输入法的当前转台信息，比如当前的语言、采用的字符集等。
- 控制面板，用于用户改变输入法的状态，比如选择不同的语言和字符集等。因此它通常附属于状态窗口。

图 3　　输入法界面

Java 输入法框架提供了输入法引擎 SPI，使开发者可以使用 Java 语言开发输入法。使用它，只需要实现两个接口 InputMethodDescriptor 和 InputMethod。InputMethodDescriptor 用于提供一些必要的信息来帮助输入法框架初始化输入法程序，比如返回本输入法能够处理的语言，返回输入法名称等。另外在 InputMethodDescriptor 需要实例化 InputMethod。当输入法实例创建以后，它通过 InputMethodContext 与输入法框架、客户端组件等窗口进行交互。这其中包括如下几个重要的接口：

setInputMethodContext 方法用于初始化上文提及的 InputMethodContext。通常输入法会根据其内容生成对应的状态窗口来向使用者展示输入法的当前状态。

activate 和 deactivate 方法分别用于客户端获得或者失去文本输入焦点的情况。一般来说只有当输入法处于 active 状态时，它才会工作。在本文中，activate 用于重新绘制输入法界面，初始化算法参数；而 deactivate 用于取消界面并释放空间。

另一个重要的方法是 dispatchEvent，它用于分发需要处理的各种鼠标、键盘事件。事件流的处理过程如图 4 所示。输入法需要判断哪些事件需要自己处理，而将不需要处理的继续发送给客户端组件。

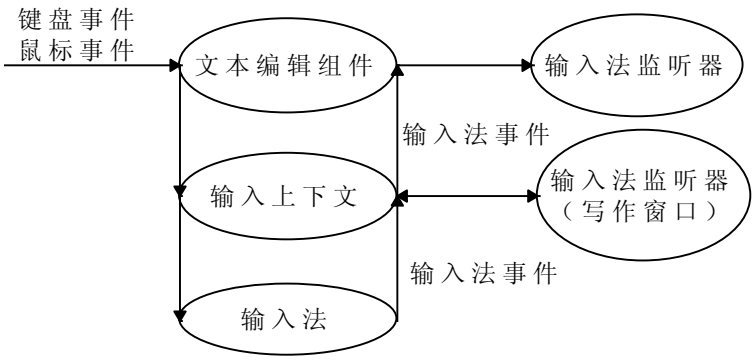
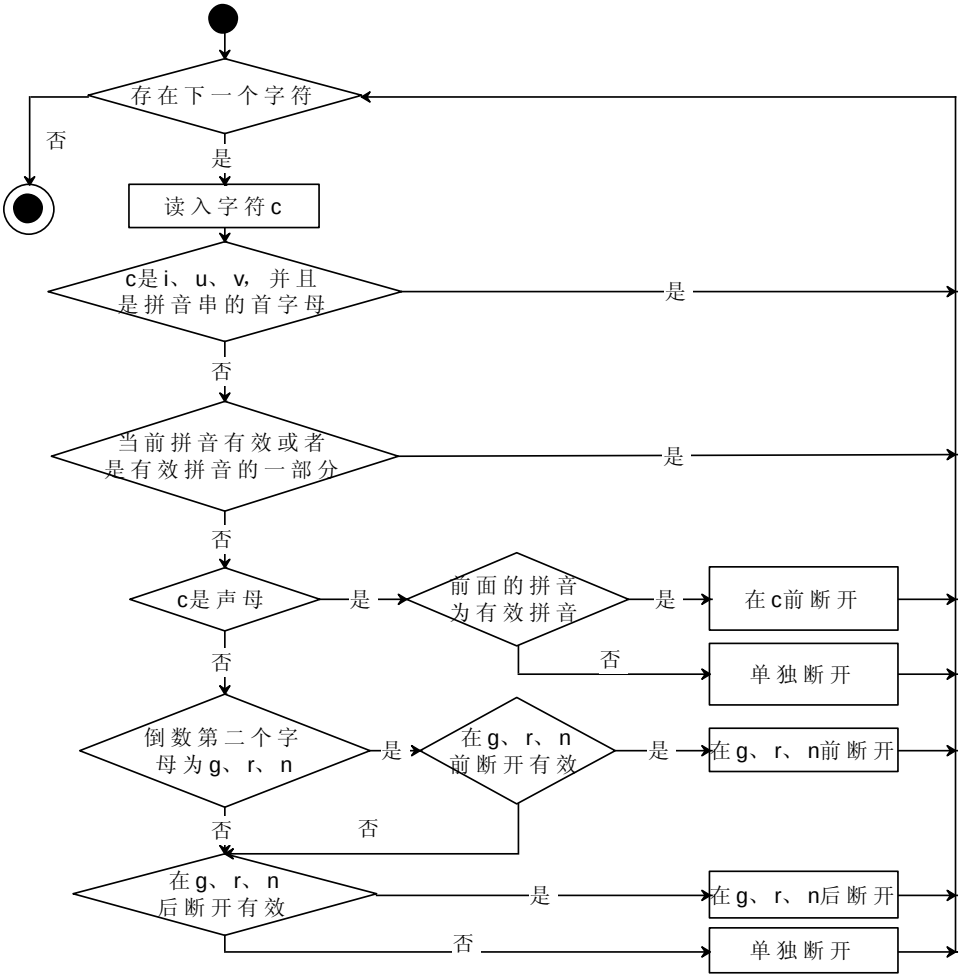


图 4　　事件流示意图[2]

3 拼音流的切分

音节是发声的自小单位。一个音节由元音（Vowel）和辅音（Consonant）构成。在汉语普通话中，每个音节由“辅音-元音”构成（其中包括只有元音而没有辅音的纯元音音节）[3]。拼音流的切分就是将拼音串中的各个音节区分开来。例如，当用户输入拼音来实现“北京大学”的输入时，需要输入的拼音串是“beijingdaxue”，切分后的结果为“bei’jing’da’xue”（“’”表示拼音切分位置）。当拼音流中出现 g、r、n 时，容易发生切分歧义，因为 g、r、n 既可以与前面的部分字符组成韵母，也可以与后面的纯元音音节组成一个有效的拼音。比如“yinge”既可以切分为“ying’e（硬腭）”，也可以切分为“yin’ge（银鸽）”。算法 1 着重对拼音中 g、r、n 进行了相应的处理，能够处理一部分切分问题[4]。在此基础之上，借助前缀树，输入法可以支持混拼输入。



算法 1　拼音流的切分算法

4 N-gram 语言模型

语言模型是语音识别、机器翻译和自然语言处理等任务的重要组成部分，也是输入法整句输入的模型基础。现今最主流的语言模型是 N-gram 语言模型，它是一种简单而非常有效的语言建模方法。N-gram 语言模型的原理是根据前 n-1 个词预测当前词的概率[5]。利用概率的乘积公式，句子 $w_1w_2…w_n$ 的概率 $P(w_1w_2…w_n)$表示为

$$P(w_1w_2…w_n) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1\ w_2)\ …P(w_n|w_{n-N+1}\ …w_{n-1}) \quad (1)$$

实际使用的通常是 N=2 或 N=3 的二元文法（bi-gram）或三元文法（tri-gram）。以二元文法为例，(1)式可以近似为

$$P(w_1w_2…w_n) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_2)\ …\ P(w_n|w_{n-1}) \quad (2)$$

5 词网解码

从拼音到词的转换是通过发音词典实现的。对于没有出现在发音词典中的词，本文首先将其转换为汉字，然后利用语言模型对汉字进行组合。

由于同音字、同音词的出现，相同的拼音串存在多种可能的词序列，而词网就是所有这些可能的词序列的一种压缩表示形式。例如，“ce’shi’pin’yin’shu’ru”对应的词序列是“测试拼音输入”（图中蓝色路径），但拼音串对应的可能词序列如图 5 所示：

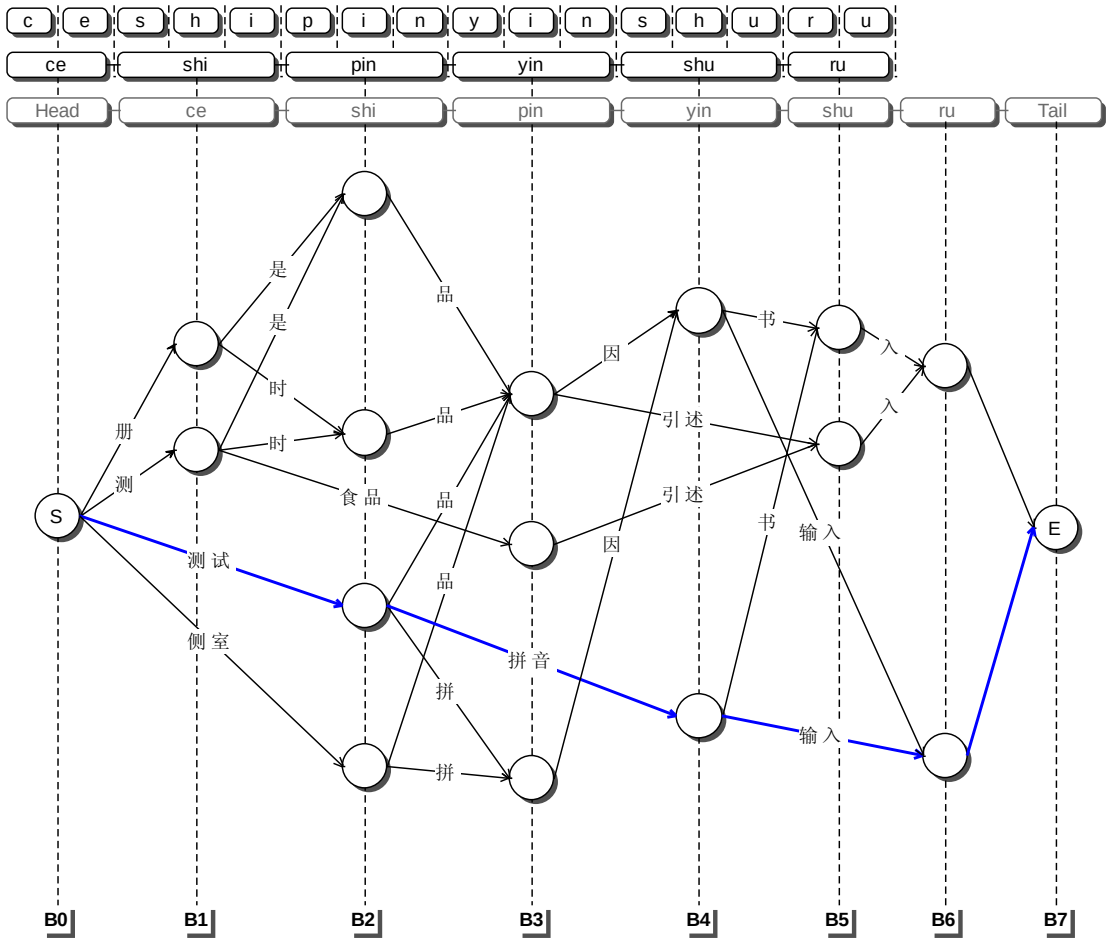


图 5 “ce shi pin yin shu ru”的词网示意图

在词网中，本文利用 N-gram 语言模型，对词网中的路径进行打分，分值表示该条路径对应句子的概率。然后，利用动态规划算法在词网中挑选出一条概率最大的路径，其对应的词序列即为转换结果。由于解码空间比较大，出于时空考虑，本文采用边构建词网边裁剪的算法来实现。

本文采用的方法结合了 CKY（Cocke-Kasami-Younger）算法[5]和 HMM 中的 Viterbi 算法[6]。这种方法既可以动态的找到最优路径，又结合了句法分析中经典的自底向上的解码方法。因此为利用完全句法分析的信息来进一步提高系统性能打下了良好的基础。图 6 示意了图 5 中词网实际的解码过程。其中蓝色箭头对应了图 5 中最优路径的解码过程。本文采用的解码方法如算法 2：

```
for j = 0 to len(words)-1:
    table[j, j] = new_cell
    for i = j-1 to 0:
        for k = I to j-1 :
            table[i,j] = table[i,j] + union(table[i,k], table[k+1,j])
```

算法 2　拼音输入法解码算法

其中 words 表示切分后的音节个数，table 表示图 6 中对应的表格。union 为合并操作。该操作既添加了可能的新词，也合并了对应两个单元中的路径，并对结果进行重打分、排序和剪裁。

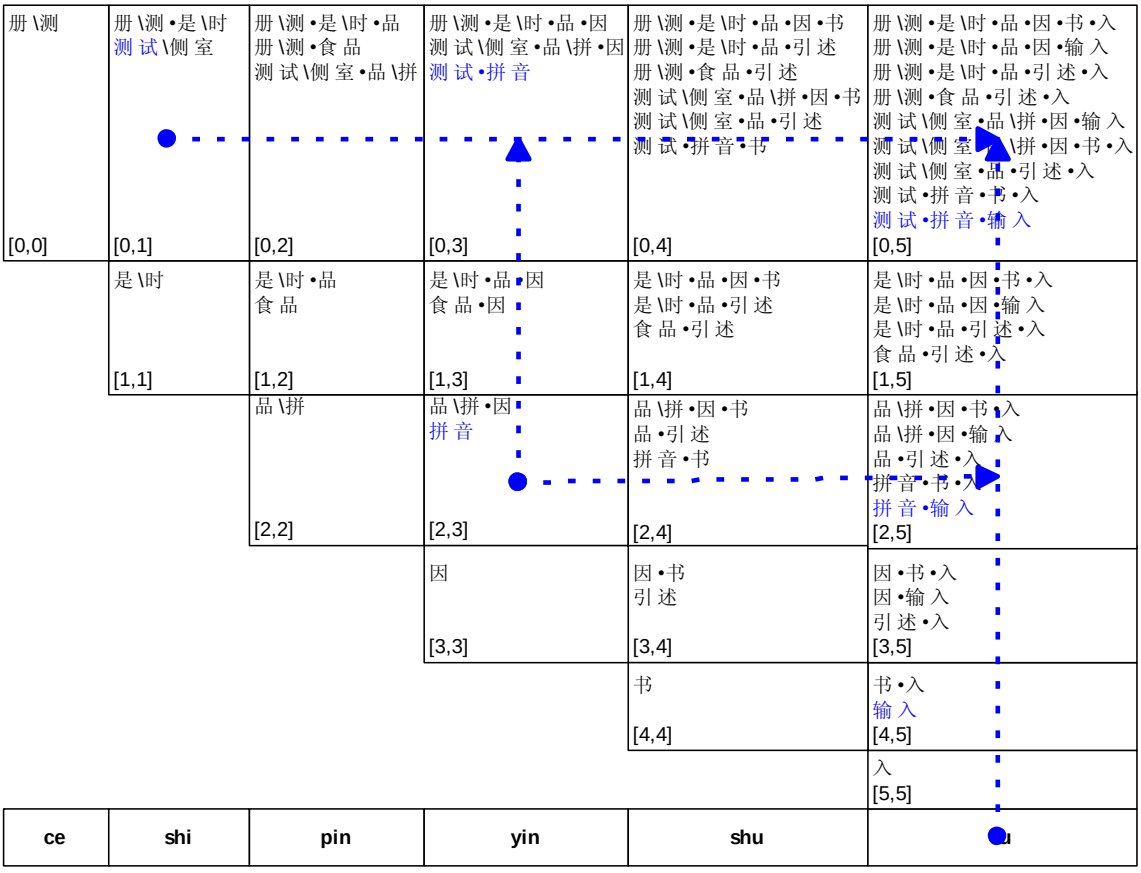


图 6　“ce shi pin yin shu ru”的解码过程

6 实验

本输入法采用的语言模型，训练语料来自北京大学计算语言所标注的 1998 年 1 月的人民日报语料[7]，共 45000 句话。使用 SRILM[8]训练三元文法，平滑算法为 KN 平滑。训练得到一元文法 60003 个，二元文法 396081 个，三元文法 103242 个。

解码时，对每一步得到的所有路径进行打分，然后根据概率分数进行重排序，最后保留得分最高的 N 条路径（N-best）进入下一步。最后一步取得分最高的路径作为最终结果。测

试语料来自 Hub-4，共 1098 句话。评价方法采用字错误率（WER），实验结果如表 1：

表 1　不同 N-best 下解码结果的字错误率

N-best	Corr(%)	Del(%)	Sub(%)	Ins(%)
1-best	85.04	0.03	14.92	0.05
5-best	86.79	0.03	13.18	0.05
10-best	86.88	0.01	13.12	0.05
20-best	86.86	0.02	13.12	0.06
30-best	86.65	0.02	13.13	0.06

通过实验可以看出，对于小语料，每次保存的最优路径数目不必过多，就可以得到最优结果。

7 总结

Java 输入法框架为设计者提供了方便、清晰的接口，接管了内存共享、窗口生成、上下文处理等底层操作，便于设计者更加关注算法设计。本文根据 Java 输入法框架设计并实现了基于整句输入的输入法系统。借助 Java 的跨平台性，改输入法有望移植到移动开发平台中。

在算法上，本文利用 N-gram 语言模型，采用 CKY 和 Viterbi 算法相结合的解码方式，收到了良好的效果。这种方法简单而有效，并且与句法分析中概率上下文无关文法（PCFG）的解码方式有天然的相似之处，因此如何在解码过程中利用句法信息是本文下一步工作的重点。

感谢

在输入法的实现过程中，本文也参考了一些网上开源的工作，比如 CityInputMethod[10]、neoeime[11]等，这些工作给了我们很大的启发。在此表示感谢。

参考文献

[1] 胡宇晓，马少平，夏莹. 基于 IMM-IME 输入法接口的实现方法. 计算机工程与应用. 2002 年 1 月

[2] http://java.sun.com/j2se/1.5.0/docs/guide/imf/spec.html

[3] 杨行峻，迟惠生. 语音信号数字处理. 电子工业出版社. 北京: 1995

[4] 李炜，贾庆成，刘政怡. 汉语拼音输入法中拼音流的切分.《现代计算机（专业版）》. 2007 年 08 期

[5] Daniel Jurafsky and James Martin. An introduction to speech recognition, computational linguistics and natural language processing. New Jersey: Prentice Hall, 2000.

[6] Lawrence R. Rabinera. Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc. IEEE 77(2):257-286. 1989
[7] http://www.icl.pku.edu.cn/icl_groups/corpus/dwldform1.asp
[8] A. Stolcke. SRILM - an extensible language modeling toolkit. In Proc. Intl. Conf. on Spoken Language Processing, 2002.
[9] http://java.sun.com/j2se/1.5.0/docs/guide/imf/spi-sample/CityIM.jar
[10] http://ostatic.com/neoeime