统计语言模型研究及其应用

吴根清

Research and Application on Statistical Language Model

Dissertation Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Engineering

by

Gen-qing WU

(Computer Science and Technology)

Dissertation Supervisor: Professor Wen-hu WU

Associate Supervisor: Associate Professor Fang ZHENG

October, 2004

摘 要

本文针对统计语言模型存在的数据稀疏、领域依赖性强、规模过大，解码速度慢等四个问题进行了深入的分析研究，其成果概括如下：

1. 提出基于泊松过程假设的 Katz 平滑算法。本文提出语言单元在语料中的出现是泊松过程的假设，并用假设检验加以验证。然后针对不同的 *n* 元组求出其期望次数置信区间的上下限。结果表明，次数较低的 *n* 元组，其期望次数的置信区间较大，说明其概率可靠性较低。基于此，本文在平滑中引入了可靠性因子以刻画该信息。为了防止可靠性低的 *n* 元组过于活跃，其概率宁小勿大。实验结果表明该方法显著降低了模型困惑度、音字转换及语音识别的错误率。

2. 提出在线递增式语言模型自适应方法。传统语言模型自适应方法是静态或半静态的，没有将使用过程中用户动态输入的语料利用起来。本文针对这点提出了在线递增式自适应方法。采用改进的 MAP 算法在线更新自适应参数，即用动态加权因子对新语料加权以加快自适应速度。针对参数振荡问题提出抑制振荡的方法，从而保留正确的参数更新并修正错误的参数更新。实验表明该方法能显著降低音字转换的错误率。

3. 提出基于概率剪枝和单元排名结合的统计语言模型压缩方法。由于经概率剪枝处理后保留下来的单元概率分布比较整齐，本文引入了基于排名的压缩算法，仅根据单元的排名位置来确定其概率。实验结果表明采用该方法压缩的模型性能明显好于经传统方法压缩的模型。

4. 提出了分层结构的语言模型应用框架及束网格解码算法。本文提出一种分层结构的语言模型应用框架，并在此基础上提出基于音节或者码元的束网格解码算法，实验结果表明该算法可以使用很少的路径实现很好的解码效果，而且占用内存少，速度快，使用范围广，可以在嵌入式设备上应用。

关键词：统计语言模型 语言模型平滑 语言模型自适应 语言模型压缩 解码算法

Abstract

In this paper, in-depth research on the following aspects of Language Modeling (LM) is described: data sparseness, domain-dependence, huge model size problems and decoding speed. The achievements are as follows,

1. A Poisson Process Hypothesis based on Katz smoothing is proposed. First, hypothesis testing is used to prove that the occurrences of an n-gram unit in the corpus conform to the Poisson Process. Second, the confidence intervals for units with different occurrence counts are constructed, which leads to the conclusion that the confidence intervals for units with low occurrence counts are much bigger than those for units with high occurrence counts, and thereby the reliability of the probabilities for units with different occurrence counts varies greatly. Consequently, a reliability factor for smoothing is proposed to catch this information. In order to prevent those units with low reliability from being overactive, their probabilities are limited. Experiments show that this method can reduce both the perplexity of a language model and the character error rate in both speech recognition and Pinyin-to-Hanzi conversion.

2. An online incremental language model adaptation method is presented. Conventional LM adaptation methods are static or half-static, meaning the corpus produced by the users of an application is not used. In this paper, my online incremental LM adaptation method updates the model parameters online via the proposed MAP method, according to the language data produced by application users. In order to speed up the adaptation, a dynamic weight factor is established. A method for controlling the parametric oscillation is provided to guarantee that the correct updates will remain while the improper updates will be amended. Experiments show that the new adaptation method can reduce the character error rate in Pinyin-to-Hanzi conversion.

3. A novel language model compression method is also put forward. In this paper, a new method based on probability pruning and ranking is applied to language model compression. Because the probabilities of the units surviving pruning are in good order, it is appropriate to utilize the rank-based method. This compression method is proved by experiments.

4. A multi-layer framework for language model-based applications is further illuminated. Moreover, decoding algorithms based on syllable or Pinyin input are brainstormed. Experiments show that these algorithms achieve very good decoding results with a very small beam width; even better, their low memory-requirement and fast decoding speed make them very effective on embedded devices.

Key words: Statistical Language Model, LM Smoothing, LM Adaptation, LM Compression, Decoding Algorithm

目　录

第一章　引　言

难以想象没有语言的生灵会怎样思考，但是人们可能猜想，没有语言的世界在某种意义上会和没有货币的世界差不多——在没有货币的世界里，用作交换的是实际的物品，而不是代表它们价值的金属符或纸符。在这种情况下，最简单的交易尚且慢而麻烦，稍复杂的交易谈何容易！

——Derek Bickerton(比克顿)

夏威夷大学语言学家

语言是人类社会特有的现象。远古时期，人类生产和生活对交流的需求直接导致了语言的萌芽，产生了一些日常性的语音发音。这种原始的语音发音逐渐固定化并赋予特定的含义，就形成了原始的语言。随着原始人类社会的逐步形成和发展，语言成为社会交流的重要工具，并得到了进一步完善。人们对话语进一步进行抽象，形成一套语音、词汇和语法的规则系统。文字的发明把该系统符号化，使得这一系统有了更加确定的传承和交流的载体。

语言作为人类最重要最自然的交流工具，是人类获得信息的最重要的渠道之一，也是文字信息处理领域最重要的方向。中文语言模型作为中文信息处理中最重要的技术之一，有着非常广泛的应用。比如，在语音识别中，为了真正实现从声音到文字的转换，计算机除了需要“听”出是哪个音外，还需要确定该音映射到哪个(些)文字，这就需要依靠语言模型对所有的候选进行打分[1]。在中文输入领域，为了提高输入效率，可以使用语言模型开发出好用的整句输入法。

本论文的研究工作主要集中在统计语言模型建模的研究工作。

本章的内容安排如下：首先简单介绍了语言模型及其研究方法，然后介绍统计语言模型研究所面对的困难和研究现状，最后给出了论文研究工作的整体描述和组织方式。

1.1 统计语言模型研究的现状和难点

语言模型建模方法分为两大类，一种是完全纯粹的依靠大文本的数据，用统计的方法建模；另外一种是以 Chomsky 的形式语言[2, 3]为基础的确定性语言模型，后者更注重语言中语法信息的分析。统计语言学是语言学研究中相对年轻的分支，但是以其简单有效而在实际应用中获得了很大的成功。

1.1.1 统计语言模型的应用

统计语言模型由于具有准确性高、容易训练、容易维护等优点，使其在涉及到文本处理的领域都有非常重要的应用价值。常用的使用方式有两种：一种是作为后处理模块或系统有机组成部分，最常见的是在语音识别、手写识别、OCR(Optical Character Recognition)和机器翻译中使用，另一种是将语言模型作为单独的应用，比如用于中文整句输入法、文本校正等。

统计语言模型在语音识别中的应用

要研究语音识别，首先要明白人对语音的识别过程。人们经过研究发现，获取一段语音(一句话)的意思时，不是简单地通过对声音信号中单个音进行识别然后拼接起来完成的[1, 4]，对某个音识别正确与否与该音所处语境的上下文紧密相关。有时候说话人由于某种原因使得某个音或某几个音发生了一定程度上的畸变(口语中这种现象尤为普遍)，或者听者因环境噪音等因素没有听清说话人所说的一或几个音，但在大部分情况下听者都能够根据各方面的非语音知识，包括当前谈话的主题、上下文信息、语境等来弥补漏掉的音节而获得正确的信息。由此可以得出结论：人在进行语音识别时，不仅使用了耳朵提取到的声学信息，还在很大程度上利用了通过其他手段获得的非声学的信息。这些非声学信息包括词法、句法以及语义等信息。语音识别中语言模型的任务就是充分刻画非声学的信息。

我们可以将大词表连续语音识别系统按照其流程很明确地划分为两大部份：一是声学识别过程，即给定一个语音样本，将该语音样本映射到一个声学单元序列的过程[5]；二是后处理过程，也就是根据第一部分所得

的声学单元序列，映射到正确句子的过程[6]。当然，在实际系统中，这两个部分可能在实现上相互结合，成为一个统一的整体，但是在功能上是分成两个部分(如图 1-1 所示)。

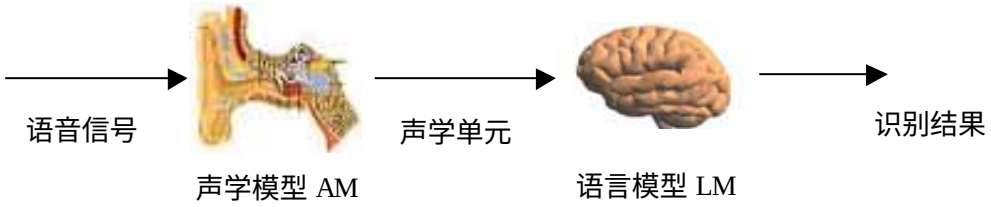

图 1-1 语音识别原理图

如果将输入的声音序列记为 A，最后得到的识别结果句子记为 S^*，则语音识别的任务是努力保证在所有句子候选中，正确的句子 S^* 产生 A 的可能性最大，即

$$S^* = \arg\max_S P(S \mid A) = \frac{\arg\max_S P(A \mid S)P(S)}{P(A)} \approx \arg\max_S P(A \mid S)P(S)$$

其中 $P(A|S)$ 恰好是声学模型评分，而 $P(S)$ 则恰好为语言模型评分。由此可见，语言模型是大词表连续语音识别中不可或缺的模块，其性能直接影响着整个系统的性能。

统计语言模型在整句音字转换中的应用

统计语言模型在中文输入方面也有很大的应用。中文语言的一大特点存在大量的同音字，即汉字的个数远远多于拼音的个数。根据国家标准 GB2312-80[7]，国家规定的一级和二级常用汉字总共有 6763 个，而拼音总共只有大约 400 个(如果算上不常用的用于标注方言词的拼音，则比 400 略多)[8]。平均每个拼音对应 17 个汉字，个别音节(如 yi)对应的汉字达到 100 多个。因此，使用单字拼音输入时，每个汉字需要选择才能得到正确的输入，输入效率比较低下。而基于统计语言模型技术的整句音字转换技术为该问题的解决提供了一个可行的思路。

语言模型在整句音字转换中的应用和在语音识别中的应用类似，原理如图 1-2 所示。

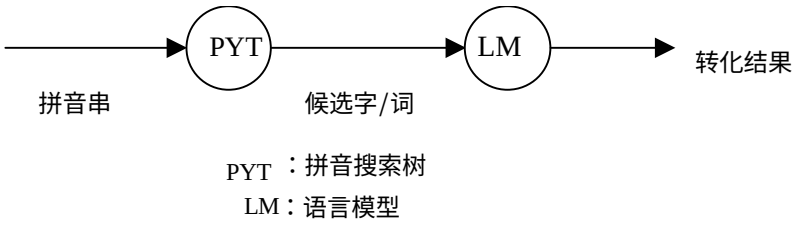

图 1-2 语言模型在中文整句音字转换中应用原理图

1.1.2 统计语言模型建模方法

从根本上说，统计语言模型是用来计算句子概率的模型。给定句子 $S = w_1, w_2, ..., w_k$，根据统计语言模型的**链式原则**，句子的概率可以表示为

$$P(S) = P(w_1)P(w_2 \mid w_1)...P(w_k \mid w_1, ..., w_{k-1}) \tag{1.1.1}$$

该式是统计语言模型的准确概率公式，但实际上由于参数太多，即使对于很小的 k，该式都很难计算。目前有多种方法用于对公式(1.1.1)进行近似计算，比如 n-gram 模型方法[9]、决策树模型方法[10]、最大熵模型方法[11]、基于词类的 n-gram 模型[12-17]等，其中 n-gram 模型由于其简单有效，得到了广泛的应用，也是本文研究的内容。

在 n-gram 建模方法中，为了解决历史过长导致参数过多的问题，对历史长度进行限制，并使用这样的假设：语言是一个马尔科夫过程，第 n 个单词出现仅仅与过去的前面的 n-1 个单词相关，而与其他任何词都不相关。因此，句子 S 的概率可以用更简单的形式表示，即

$$P(S) = P(w_1)P(w_2 \mid w_1)...P(w_k \mid w_{k-n+1}, ..., w_{k-1}) \tag{1.1.2}$$

当 n 取 1、2、3 时，分别成为 unigram、bigram 和 trigram 模型。一般而言，n 很少取 4 或者更大的数字。

在信息论中用信息熵(Entropy)[18]来度量一种语言的复杂程度，下面先

简单介绍信息论中的几个基本概念。

信源是提供消息的人或机器。信源输出是以符号形式出现的具体消息，它载荷消息。

信源的信息量指每个信源输出符号所需的编码的平均个数。采用二进制编码时，它指的是每个信源输出符号所需的平均二进制位数。

对于无记忆信源，即独立的信源 X，其输出符号集为 $\{x \mid x \in X\}$，每种符号的概率记为 $P(x)$，则该信源的熵定义为

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{1.1.3}$$

对于有记忆信源 X，其输出符号集为 $\{x \mid x \in X\}$，观察其连续 n 个输出序列，记为 $X_1, X_2, ..., X_n$，由于记忆长度有限，因此 n 充分大时可以将 $X_1, X_2, ..., X_n$ 视作一个无记忆的联合信源，其输出符号消息集 $\{(x_1, x_2, ...x_n) \mid x_i \in X, 1 \le i \le n\}$，其中单个状态的概率记为 $P(x_1, x_2, ..., x_n)$，该联合信源的联合熵为：

$$\begin{aligned} &H(X_1, X_2, \cdots, X_n) \\ &= -\sum_{x_i \in X} P(x_1, x_2, \cdots, x_n) \log P(x_1, x_2, \cdots, x_n) \end{aligned} \tag{1.1.4}$$

由此引出熵率的概念：

$$B = \frac{1}{n} \lim_{n \to \infty} H(X_1, X_2, \cdots, X_n) \tag{1.1.5}$$

熵率 B 是有记忆信源的任一输出符号的平均信息量。

实际信源的输出都是有记忆的，也就是说，输出符号不仅取决于符号出现的概率，而且也受前面符号的影响。所以，实际信源输出的符号前后之间是统计相关的。如果一个足够长的输出序列能充分反映出信源内在的统计结构，也就是说，如果这个源是统计遍历的，则公式(1.1.5)等价于

$$H(X) = -\lim_{n\to\infty}\frac{1}{n}\log P(x_1, x_2, \cdots, x_n) \tag{1.1.6}$$

即其熵可用一个足够长的源输出序列来估算。

按照信息论的观点，我们可以把语言看成信源，它的输出是词 w_i，那么在一个包含了大量句子的语料库中，每个词的平均信息量就可以由下式估计：

$$H = -\lim_{n\to\infty}\frac{1}{n}\log P(w_1, w_2, \cdots, w_n) \tag{1.1.7}$$

这里的 n 是语料库的大小。

在公式(1.1.7)中，必须知道词序列 $w_1, w_2, \cdots, w_n$ 的概率 $P(w_1, w_2, \cdots, w_n)$，可是实际上它是无法知道的，因此只能用语言模型提供的概率 $\hat{P}(w_1, w_2, \cdots, w_n)$ 对其进行估计。这样，从识别器的角度看，按照 $\hat{P}(w_1, w_2, \cdots, w_n)$ 这种概率机制产生的语言的识别难度就是：

$$LP = -\frac{1}{n}\log \hat{P}(w_1, w_2, \cdots, w_n) \tag{1.1.8}$$

无论语言模型建模技术发展得多么细致完备，也总是不能包容所有的语言知识，因此总有：

$$LP \geq H \tag{1.1.9}$$

这样，我们可以定义

$$PP = 2^{LP} = \hat{P}(w_1, w_2, \cdots, w_n)^{-\frac{1}{n}} \tag{1.1.10}$$

作为语言模型不确定度的一种量度，称为困惑度(Perplexity)[19, 20]，也称复杂度。

可以这样认为，复杂度为 PP 的语言模型，其区分难度等价于让识别器在 PP 个相似的候选词中做出识别，也就是说，复杂度就是语言模型在

每个词结点上的平均分支数。

对语言知识没有建模的语音识别器等效于含有一个认为所有句子分布概率均等的语言模型，则它的识别难度为：

$$\begin{aligned} LP0 &= -\frac{1}{n}\log \hat{P}(w_1, w_2, \cdots, w_n) \\ &= -\frac{1}{n}\log \frac{1}{V^n} \\ &= \log V \end{aligned} \tag{1.1.11}$$

其中 V 是词表的大小。由此可得它的复杂度

$$PP0 = 2^{LP0} = V \tag{1.1.12}$$

也就是说，在没有任何高层次语言知识的情况下，识别器在识别每一个词时都要与词表中的全部 V 个词进行匹配计算。

我们知道，当信源的输出为等概率分布时，它的熵最大。那么在建立了某种语言模型后，它的识别难度总要比等概率情况下的 $LP0$ 小：

$$LP \le LP0 \tag{1.1.13}$$

因此，

$$H \le LP \le LP0 \tag{1.1.14}$$

那么，对于任何语言模型，都满足下式：

$$2^H \le PP \le V \tag{1.1.15}$$

这样一来，就可以看到，语言模型的作用实际上是降低识别难度，等效于缩小词表的实际大小，从而不用在整个词表中进行匹配了。

总结说来，语言的熵 H 反映了语言本身固有的识别难度，困惑度 PP 则反映的是与所用的语言模型有关的识别难度。这二者的差正好反映了一个语言模型还可以改进提高的潜力。

1.1.3 统计语言模型研究面临的难题

数据稀疏问题

数据稀疏性是统计语言模型最大的问题之一。由于在语言模型使用过程中，很多 *n* 元组没能在训练语料中观测到，因此如何有效的估算这些单元的概率就成了很大的问题。中文是典型的语义型语言[21]，相对于英语等语法型语言来说，遣词造句更为随便，这种语言单元搭配的随意性导致 *n*-gram 单元更多更复杂，并且导致语言模型应用场合特点和训练语料性质的差异性可能更大，也就要求性能更好的语言模型建模方法。

领域依赖问题

统计语言模型的性能对训练数据的领域依赖很强。用某领域的语料训练的语言模型在相同领域中使用时性能非常好，但是如果应用领域发生变化(有些领域无法收集语料，只能用一些通用的语料训练模型)，则模型性能急剧下降[22]。传统的自适应方法大多是一种静态的方法，没有将语言模型使用过程中产生的语料利用起来，或者仅仅是将这部分语料作为简单的参考因素(比如用于从通用语料中筛选出和当前文本相关的语料)，因此有其固有的先天缺陷。要解决传统自适应方法带来的问题，最根本的就是需要考虑如何将语言模型使用过程中得到的语料充分利用起来。

规模过大问题

统计语言模型的参数数目非常庞大。人们为了获得更好的模型性能，往往采取增加训练语料的方法来训练模型，带来的后果是模型规模进一步上升，对存储的需求进一步增大。该问题限制了语言模型应用的场合，即使在个人计算机上，几百 MB 甚至 1GB 规模的语言模型也显得有些过大。对于目前蓬勃发展的嵌入式设备(如高端手机)来说，该数字已超出设备存储的极限。因此，解决语言模型规模过大的问题，不仅能使其在个人计算机上的应用更加灵活，而且有可能将语言模型应用到嵌入式设备上。

解码速度问题

由于统计语言模型规模大，运行时对内存、CPU 速度的要求比较高。

大词表的语言模型应用，如中文整句音字转换、连续语音识别、OCR 等应用都由于解码速度问题一直不能在结算能力较差的设备上应用。如何设计有效的解码算法，提高解码算法的速度就成了很重要的一个问题。只有解码速度上去了，语言模型的应用能力才能进一步提高，其适用场合才能进一步拓宽。

1.2 本文研究工作概述

本文研究工作围绕提高统计语言模型的性能和应用能力，将研究焦点定位于对语言模型应用中遇到的四个问题的解决方法的研究。论文研究的整体思路如图 1-1 所示。

1.2.1 研究思路

1) 为了解决统计语言模型的数据稀疏问题，可以采取加大训练语料的方法或者研究更准确的平滑算法。增大训练数据不仅提高了训练语料的收集成本，而且使模型规模增大，不利于语言模型的应用。为此，本文提出一个假设，即语言单元出现事件是一泊松过程，然后采用假设检验的方式验证该假设，并对不同单元的概率可靠性进行刻画。并在此基础上提出了带可靠性参数的 Katz 平滑算法，提高了模型的性能。

2) 为了解决统计语言模型的领域依赖问题，本文针对传统语言模型自适应算法对语言模型使用过程中产生的语料(如语音识别或者音字转换的输出结果)没有被充分利用起来的缺点，提出一种新的语言模型自适应机制，即在线递增式的语言模型自适应框架，使语言模型在使用过程中能够不断提高性能；

3) 为了解决统计语言模型规模过大问题，本文在分析传统压缩方法缺点的基础上，将基于条件概率的剪枝和排名方法结合起来，使得语言模型在压缩到 1M 大小时模型性能还能保持很高的水平。

4) 针对语言模型应用中的解码速度问题，本文提出分层结构的语言模型应用框架，并在此基础上提出基于音节或者码元的束网格解码算法，极

大的提高了解码的速度。

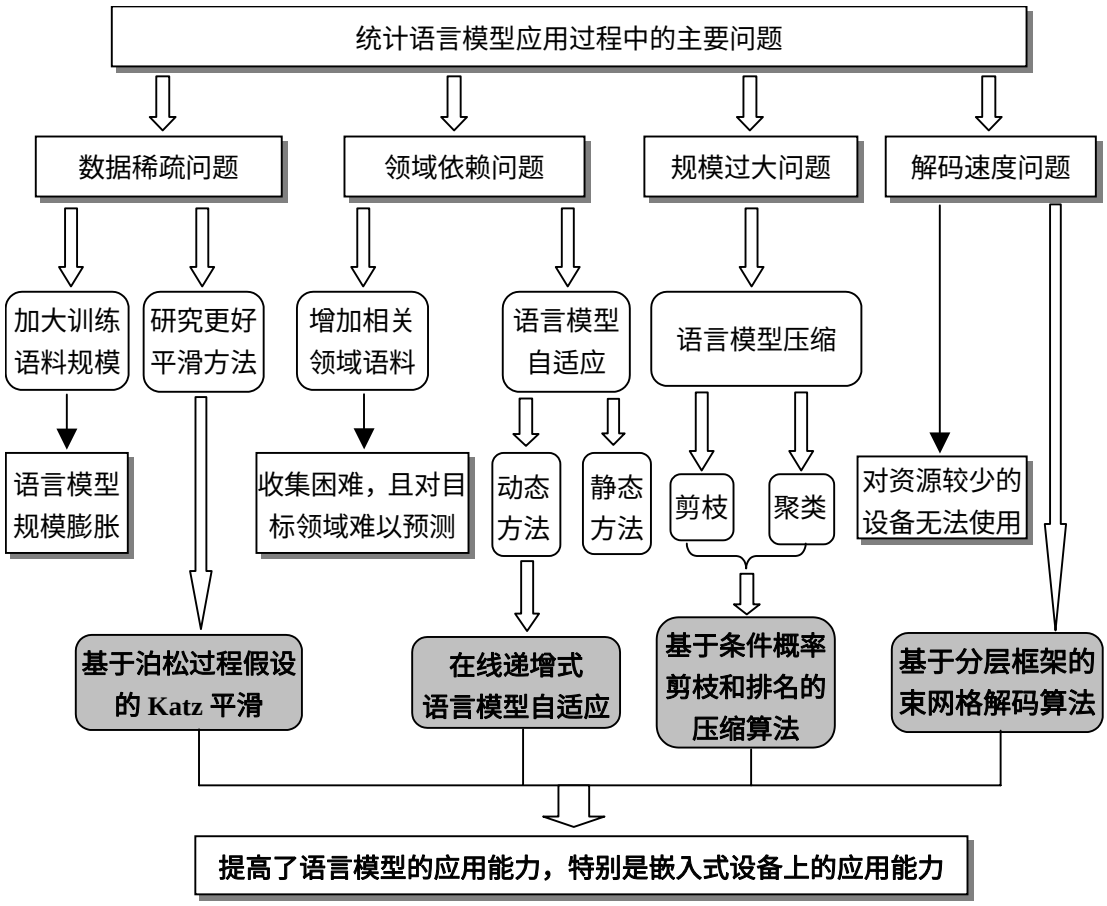

图 1-1 论文的研究思路

1.2.2 论文组织

论文的内容和结构是按照上述 4 个方面的主要工作来组织的。

第二章介绍了传统的语言模型 Katz 平滑，通过分析其不足提出了基于泊松过程假设的 Katz 平滑方法；第三章针对传统语言模型自适应方法没有充分利用使用过程中产生的语料这一问题，提出在线递增式自适应的框架和理论方法，并讨论了该方法具体实现中的一些问题；第四章通过对已有方法的分析，提出了一种基于单元条件概率和单元排名的语言模型压缩算法；第五章以将语言模型应用到嵌入式设备为目标，提出了基于分层结构

框架的束网格解码算法，实验表明该算法速度快，对内存需求少。因此其使用范围非常广，包括 PC、嵌入式设备都可使用，而且该框架的扩展性非常好。

在附录中，简单介绍了博士期间其他方面的工作，主要包括一个嵌入式设备上语言模型开发包的设计，以及使用该开发包在多个嵌入式设备上实现的第一个中文整句输入法。

第二章　基于泊松过程假设的Katz平滑方法

统计语言模型的关键问题是尽可能准确的描绘语言单元连接的概率信息，使得那些正确的句子获得更大的概率。因为正确句子的概率增大，不仅提高语音识别、OCR 等的正确率，而且能极大的限制识别过程中的解码空间[1]。另外，语言模型的性能是语言模型所有其他方面研究的基础。无论是语言模型自适应，还是语言模型压缩，都需要一个性能优越的种子模型作为出发点。

本章的内容安排如下：第 1 小节主要介绍基于图灵估计的 Katz 平滑算法；第 2 小节是以实验的方法分析了传统 Katz 平滑算法存在的问题；第 3 小节对 Katz 平滑的不足做了深入分析，提出了语言单元的泊松过程假设，并加以验证，然后由于基于单元出现次数的置信区间不同，引入单元可靠性因子，提出新的平滑方法；第 4 小节是实验和分析，实验结果表明新方法有效；第 5 小节是本章小结。

2.1 统计语言模型的平滑

统计语言模型通过对海量语料的分析获得词与词之间的连接概率关系。一般实用的中文统计语言模型的词典规模都较大，词条数通常达到 50k 以上。以 50k 为例，采用 trigram 模型，则所有合法的 trigram 单元达到 $50k^3$，即 $1.25*10^{14}$ 个。实际上我们能够获得的训练语料远远少于这个量级。导致的结果是大部分单元在我们的训练语料中都观察不到。这就是统计语言模型的稀疏性问题[9]。如果我们将那些没在训练语料出现的单元概率简单地设为 0，则在使用语言模型时将导致那些没有观察到的单元永远不可能在结果中出现。但在实际应用时，正确句子的单词连接关系往往在训练语料中并不出现。我们发现，对一个使用了 200M 文本(字数)训练的模型进行测试，尽管所用测试语料的领域和训练语料领域相同，仍有高达 20%的 trigram 单元没有在训练语料中出现。

因此，为了通过语言模型将正确的句子选择出来，需要对那些没有在

训练语料出现的单元给出相对合理的概率值，这就是统计语言模型的平滑问题。

为了使叙述和分析更加清楚，严谨，首先定义如下术语：

n 元组 或者叫 *n*-gram 单元，或简称单元，指一个由 n 个单词组成的连接关系组合。写成 w_1^n，或者写做 $w_1..w_n$，单元出现次数则记为 $c(w_1^n)$ 或者 $c(w_1..w_n)$。

单元历史 指 *n*-gram 单元的前 *n*-1 个单词组成的子串，简称历史。对于单元 w_1^n 而言，单元历史是 w_1^{n-1}。

单元概率 指的是在单元历史下的条件概率，w_1^n 的单元概率为 $p\left(w_n \mid w_1^{n-1}\right)$。

2.1.1 基于图灵估计的 Katz 平滑方法

图灵(*Good Turing*)概率估计[23, 24]在解决训练数据稀疏问题的基本思想是：对在训练语料中观察到的数据进行概率调整，把调整出来的概率按照一定的规则，分配到在训练语料中未观察到的数据上，从而消除零概率估计。图灵概率估计的方法实现起来较为简单，而且效率也较高。

首先定义图灵概率估计 P_T。假设训练语料大小为 N，n_r 表示在训练语料中出现 r 次的词(串)的个数，那么

$$N = \sum_r r n_r \tag{2.1.1}$$

对一个在训练语料中出现 r 次的词来说，定义它的图灵概率估计为：

$$P_T = \frac{r^*}{N}$$

其中

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{2.1.2}$$

我们把用一个修正过的计数值 r' 来代替原来的计数值 r 的过程叫做“打折扣”(Discouting)，把 r'/r 叫做折扣系数 d_r。当 $r'=r^*$ 时，就是**图灵折扣**。

如前所述，单元的最大似然概率估计为

$$P_{ML}=\frac{c(w_1^n)}{N} \tag{2.1.3}$$

而图灵概率估计为

$$P_T=\frac{c^*(w_1^n)}{N} \tag{2.1.4}$$

其中

$$c^*(x)=(c(x)+1)\frac{n_{c(x)+1}}{n_{c(x)}} \tag{2.1.5}$$

根据公式(2.1.1)、(2.1.2)、(2.1.4)和(2.1.5)，对出现在训练语料中的词串，它们的图灵概率估计和为

$$\sum_{w_1^n:c(w_1^n)>0}P_T(w_1^n)=1-\frac{n_1}{N} \tag{2.1.6}$$

这样，对在训练语料中未观察到的词串来说，就有了一个概率估计总和 n_1/N，即：

$$\sum_{w_1^n:c(w_1^n)=0}P_T(w_1^n)=\frac{n_1}{N} \tag{2.1.7}$$

另一方面，

$$\sum_{w_1^n:c(w_1^n)>0}[P_{ML}(w_1^n)-P_T(w_1^n)]=\sum_{w_1^n:c(w_1^n)>0}\delta_{c(w_1^n)}=\sum_{r>0}n_r(1-d_r)\frac{r}{N}=\frac{n_1}{N} \tag{2.1.8}$$

其中

$$\delta_c = \frac{c}{N} - \frac{c^*}{N} = (1 - d_c)\frac{c}{N} \tag{2.1.9}$$

因此，

$$\sum_{w_1^n : c(w_1^n) > 0} \delta_{c(w_1^n)} = \sum_{w_1^n : c(w_1^n) = 0} P_T(w_1^n) \tag{2.1.10}$$

Katz 基于此“残余”提出了一种退化(*Back off*)的思想[25]，即：把公式(2.1.9)中的 δ_c 解释为一个出现次数为 $c(w_1^n)$ 的 n 元词串对“未观察到”的 n 元词串的概率的“贡献”。基于这一解释，可进一步推导对条件概率 $P(w_n \mid w_1^{n-1})$ 的估计。假设在训练语料中已经观察到 n-1 元词串 w_1^{n-1}，类似于(2.1.9)中给出的 δ_c，引入 $\delta_c^{(cond)}$：

$$\delta_{c(w_1^n)}^{(cond)} = (1 - d_{c(w_1^n)})\frac{c(w_1^n)}{c(w_1^{n-1})} \tag{2.1.11}$$

现在递归地定义估计因子 $P_s(w_n \mid w_1^{n-1})$。假设已经定义了低阶条件估计因子 $P_s(w_n \mid w_2^{n-1})$。那么，当 $c(w_1^{n-1}) > 0$ 的时候，

$$P_s(w_n \mid w_1^{n-1}) = \tilde{P}(w_n \mid w_1^{n-1}) = d_{c(w_1^n)}\frac{c(w_1^n)}{c(w_1^{n-1})} \tag{2.1.12}$$

给出了在训练语料中词 w_n 出现在 w_1^{n-1} ($c(w_1^n) > 0$)之后的条件概率估计。可以方便地定义一个函数 $\tilde{\beta}$：

$$\tilde{\beta}(w_1^{n-1}) = \sum_{w_n : c(w_1^n) > 0} \delta_{c(w_1^n)}^{(cond)} = 1 - \sum_{w_n : c(w_1^n) > 0} \tilde{P}(w_n \mid w_1^{n-1}) \tag{2.1.13}$$

给出了所有没有出现在 w_1^{n-1} 之后的 w_n ($c(w_1^n) = 0$)的条件概率和的一个估计。我们使用以前(递归)定义的低阶条件分布 $P_s(w_n \mid w_2^{n-1})$ 把(2.1.13)定义的条件“块” $\tilde{\beta}$ 分配到 $c(w_1^n) = 0$ 的那些 w_n 上：

$$P_s(w_n \mid w_1^{n-1}) = \alpha P_s(w_n \mid w_2^{n-1}) \tag{2.1.14}$$

其中

$$\alpha = \alpha(w_1^{n-1}) = \frac{\tilde{\beta}(w_1^{n-1})}{\sum_{w_n:c(w_1^n)=0} P_s(w_n \mid w_2^{n-1})} = \frac{1 - \sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_1^{n-1})}{1 - \sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_2^{n-1})} \tag{2.1.15}$$

是一个归一化常数。当 $c(w_1^{n-1}) = 0$ 的时候，可以定义

$$P_s(w_n \mid w_1^{n-1}) = P_s(w_n \mid w_2^{n-1}) \tag{2.1.16}$$

此时，$\tilde{P}(w_n \mid w_1^{n-1}) = 0$，而且 $\tilde{\beta}(w_1^{n-1}) = 1$。

最后，把公式(2.1.12)、(2.1.14)和(2.1.16)结合起来，用如下的递归表达式表示条件概率分布：

$$P_s(w_n \mid w_1^{n-1}) = \tilde{P}(w_n \mid w_1^{n-1}) + \theta(\tilde{P}(w_n \mid w_1^{n-1})) \cdot \alpha(w_1^{n-1}) P_s(w_n \mid w_2^{n-1}) \tag{2.1.17}$$

其中

$$\theta(x) = \begin{cases} 1, & \text{如果} x = 0 \\ 0, & \text{其它情况} \end{cases} \tag{2.1.18}$$

现在就得到了一个修正过的概率估计式，它由公式(2.1.17)给出。但对于计数值 $c>k$ 的那些 n 元词串来说，Katz 认为它们的概率估计是可靠的，而不想对它们的概率估计打折扣，即不想用图灵概率估计来代替最大似然概率估计，与此同时，仍然要保证所有未观察到的 n 元词串的概率估计和为 n_1 / N。要达到这一点，重新定义

$$d_r = 1, \qquad \text{对} r > k \tag{2.1.19}$$

而且还应该相应地调整原先的 *Turing* 折扣系数 d_r，对 $r \le k$，使得表示“贡献”与未观察到的 n 元词串概率之间平衡关系的等式，仍然可以满足，即：

$$\sum_{w_n:c(w_1^n)>0} \delta_{c(w_1^n)} = \sum_{1 \le r \le k} n_r (1 - d_r) \frac{r}{N} = \frac{n_1}{N} \tag{2.1.20}$$

公式(2.1.20)类似于公式(2.1.7)。按照如下形式求解(2.1.20)中的 d_r：

$$(1-d_r)=\mu(1-\frac{r^*}{r}) \tag{2.1.21}$$

其中 μ 是一个常数。这样，(2.1.20)的唯一解就是：

$$d_r=\frac{\frac{r^*}{r}-\frac{(k+1)n_{r+1}}{n_1}}{1-\frac{(k+1)n_{r+1}}{n_1}},\qquad \text{对}1\le r\le k \tag{2.1.22}$$

公式(2.1.21)就相当于这样一个要求：新定义的计数“贡献”($r/N-r'/N$)与 *Turing* 的“贡献”($r/N-r^*/N$)是成正比的。

于是经过改造之后的基于图灵估计的退化模型概率估计(以 *bigram* 为例)可以表述为

$$P_s(v|u)=\begin{cases}d_r(u)\,f(v|u) & for\ \ r>0\\ \alpha(u)P_s(v) & for\ \ r=0\end{cases} \tag{2.1.23}$$

其中

$$f(v|u)=\frac{C(u,v)}{C(u)}=\frac{r}{c(u)}\ \ , \tag{2.1.24}$$

$$d_r=\begin{cases}1 & for\ \ r>k\\ \dfrac{\frac{r^*}{r}-\frac{(k+1)n_{k+1}}{n_1}}{1-\frac{(k+1)n_{k+1}}{n_1}} & for\ \ k\ge r>0\end{cases}\ , \tag{2.1.25}$$

$$\alpha(u)=\frac{1-\sum_{v:c(u,v)>0}P_s(v|u)}{1-\sum_{v:c(u,v)>0}P_s(v)} \tag{2.1.26}$$

根据 Katz 平滑的文献[25]，公式(2.1.23)、(2.1.25)中的折扣系数 d_r 是带

下标的，即基于单元历史的。否则根据其推导情况，公式(2.1.20)将不再成立。

2.1.2 其他主要平滑方法

除了 Katz 平滑之外，还有一些其他的平滑方法，具代表性的有：

1. 加性平滑

加性平滑[26-28]是最早最简单的平滑算法，这种平滑假设每个 n 元组都至少出现了 δ 次(通常 $0<\delta\le 1$)，δ 可以取 1[29, 30]。这种平滑算法实际使用时效果并不好。

2. Jelinek-Mercer 平滑

Jelinek 和 Mercer 以及后续研究者建议将高阶的最大似然估计概率和已经过 Jelinek-Mercer 平滑处理的低阶 n 元组概率用插值方法结合起来[9]，并且根据每个 n 元组决定其插值系数[29]，即：

$$P_{\text{interp}}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{\text{ml}}(w_i \mid w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) P_{\text{interp}}(w_i \mid w_{i-n+2}^{i-1}) \quad (2.1.27)$$

通过插值，保证那些高阶概率 $P_{\text{ml}}(w_i \mid w_{i-n+1}^{i-1})$ 为零的单元能得到概率，而且它们之间大小关系由低阶的概率大小 $P_{\text{interp}}(w_i \mid w_{i-n+2}^{i-1})$ 来确定。

3. Kneser-Ney 平滑

Kneser 和 Ney 改进了低阶分布对高阶模型的影响因子[31]。举个简单的例子：假设 $w_a w_b$ 是一个比较常用的词组，这样 w_b 出现的概率就不会太少；但是，几乎所有的 w_b 在训练语料中都出现在 w_a 的后面，从直观上说，w_b 在 unigram 级别上不应该有比较大的概率。所以，unigram 的概率不应该完全根据单词出现的次数按比例分配概率，而应该根据单词出现在多少个不同的词的后面来分配概率。

为叙述方便，用 (X,Y) 表示 X 后续连接 Y，且定义：

$$N_{1+}(\bullet, w_{i-n+2}^{i}) = |\{w_{i-n+1} : c(w_{i-n+1} w_{i-n+2}^{i}) \ge 1\}|$$

其中 N_{1+} 表示的是单元至少出现 1 个，而 $\bullet$ 表示所有可能的情况，因此上式表示 w_{i-n+2}^{i} 在训练语料中跟随在多少个不同的单词之后出现。定义：

$$N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet) = \sum_{w_i} N_{1+}(\bullet, w_{i-n+2}^{i-1}, w_i) = \sum_{w_i} N_{1+}(\bullet, w_{i-n+2}^{i})$$

则 Kneser-Ney 平滑(简称 K-N 平滑)的低阶概率采用

$$P_{\mathrm{kn}}(w_i \mid w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet, w_{i-n+2}^{i})}{N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet)} \tag{2.1.28}$$

且整个模型概率用

$$P_{\mathrm{kn}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D}{\sum_{wi} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})\ P_{\mathrm{kn}}(w_i \mid w_{i-n+2}^{i-1}) \tag{2.1.29}$$

其中 D 是一个绝对折扣次数，$\gamma(w_{i-n+1}^{i-1})$ 用于保证概率总和为 1。

Stanley F.Chen 对 Kneser-Ney 平滑又作了一定的改进[32]，对于出现次数较少的这些词进行分类处理，即针对出现次数为 1、2、3 的单元用几个绝对折扣值 D_1、D_2、D_3 代替单个折扣值 D。具体为：

$$Y = \frac{n_1}{n_1 + 2n_2} \tag{2.1.30}$$

$$D_1 = 1 - 2Y\frac{n_2}{n_1} \tag{2.1.31}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2} \tag{2.1.32}$$

$$D_3 = 3 - 4Y\frac{n_4}{n3} \tag{2.1.33}$$

Ronald Rosenfeld 对语言模型的研究做了比较完整的回顾[33]。实际上，当前平滑算法的主流形式都是高阶和低阶相结合的形式，如 Katz 方法，在 n 阶的对应单元为 0 时，试图回退到 n-1 阶，依靠 n-1 阶单元尽可能估计 n

阶单元的概率。

Stanley F. Chen 对这些平滑方法做了详细的总结[32]。从文献的评价结果来看看，目前性能最好的平滑算法当属 Katz 和 K-N，K-N 算法由于回退时使用的模型并非是低阶 K-N 模型本身，即考虑的不是低阶单元的次数，而是和低阶单元连接的不同单词的个数。这样，实际上额外增加了模型的计算和存储，显然对模型压缩不利。

因此，考虑模型的综合性能，Katz 平滑仍然是最具竞争力的平滑算法。

2.2 Katz平滑存在的问题分析

Katz 平滑中最重要的就是平滑因子 d_r 的计算。实际上，每个出现 r 次的单元都被当成只出现了 $d_r \cdot r$ 次，而从这些单元拿到的 $(1-d_r)\cdot r$ 个单元将被分配给那些在训练语料中出现次数为 0 的单元，因此，实际上回退因子 $\alpha(\bullet)$ 是由 d_r 影响的。

在经典的 Katz 平滑中，平滑因子 d_r 的计算是问题的关键。下面分析经典 Katz 平滑的问题中，d_r 计算存在的问题。

2.2.1 全局折扣与局部折扣的对比

首先，Katz 平滑[25]是基于图灵估计[24]的平滑算法，研究 Katz 的推导过程可以发现，他是对每个不同的单元历史 w_1^{n-1} 计算 d_r 的，即对每个历史分别使用图灵估计。这一点可以从公式(2.1.13)和(2.1.20)可以看出，尤其公式(2.1.20)左边是基于给定单元历史 w_1^{n-1}，对任意后续单词 w_n 求和。本文把基于历史的折扣系数记为 $d_r\left(w_1^{n-1}\right)$ 或者 $d_r(h)$，表示其和历史相关。在公式(2.1.25)中，n_1 和 n_{k+1} 也是和历史 w_1^{n-1} 相关，分别记为 $n_1\left(w_n^1\right)$ 和 $n_{k+1}\left(w_1^{n-1}\right)$。

如果训练语料非常充分，这种针对每个历史 w_1^{n-1} 分别使用图灵估计的做法没有任何问题。实际上训练数据稀疏性导致该算法并不非常理想。对于大部分单元历史 w_1^{n-1} 而言，以它为历史且出现次数为 r 的单元个数 $n_r\left(w_1^{n-1}\right)$ 可能会很小，并最终导致 $d_r\left(w_1^{n-1}\right)$ 的估计不准确。

本文使用了 200M(即 2 亿)汉字的中文文本用于 trigram 模型的训练，

实验结果表明，训练总共产生了 700 万的 bigram 单元和 2,900 万的 trigram 单元。

表 2-1 所示的是我们训练的统计语言模型中 *n*-gram 单元出现次数和 *n*-gram 单元历史出现次数的分布情况。为区别 bigram 和 trigram，分别用 $n_{r,2}$ 和 $n_{r,3}$ 表示模型中出现 *r* 次的 bigram 单元和出现 *r* 次的 trigram 单元(都不考虑单元历史，采用单元的全局分布)。而 $n_{r,3}(h_1)$ 和 $n_{r,3}(h_2)$ 则分别表示以 h_1 和 h_2 的为历史且出现次数刚好为 *r* 次的单元个数，$C(h)$ 则表示单元历史 *h* 出现的次数。

表 2-1 *n*-gram 单元分布情况(部分)

r	1	2	3	4	5	6	$C(h)$
$n_{r,2}$	3,815k	1,040k	482k	285k	190k	137k	N/A
$n_{r,3}$	21,529k	3,842k	1,432k	743k	449k	307k	N/A
$n_{r,3}(h_1)$	201	35	17	6	4	3	628
$n_{r,3}(h_2)$	3	1	0	1	0	0	9

可以根据表 2-1 的数据分别计算出每一个系数 d_r，为方便起见，分别用 $d_{r,2}$ 和 $d_{r,3}$ 表示 bigram 和 trigram 的折扣系数，并称为全局折扣系数，与之对应，$d_{r,3}(h)$ 称为局部折扣系数。结果如表 2-2 所示。

表 2-2 根据单元全局分布和基于单元历史分布得到的折扣系数

r	1	2	3	4	5
$d_{r,2}$	0.420	0.611	0.730	0.787	0.828
$d_{r,3}$	0.297	0.518	0.663	0.732	0.804
$d_{r,3}(h_1)$	0.284	0.702	0.419	0.817	0.890
$d_{r,3}(h_2)$	溢出，无法计算				

从结果可以看到，全局折扣因子的结果非常优美。随着 *r* 的增加，被折扣下来的比例 $(1-d_r)$ 呈减小趋势，但是被折扣下来的绝对次数 $r\cdot(1-d_r)$ 呈增加趋势。而考虑了单元历史的 $d_{r,3}(h)$ 则由于参数计算的 $n_{r,3}(h)$ 数值较小，导致结果不稳定，甚至个别单元历史出现了计算溢出的情况。

可见，在 Katz 平滑中需要使用不考虑单元历史的折扣系数，即 d_r，而不应使用 $d_r(h)$。

全局折扣系数的计算实际上只是套用了 Katz 平滑公式中局部折扣因子的计算公式。采用全局折扣因子后，Katz 平滑推导过程中的一些式子将不再成立，比如公式(2.1.20)将不再成立。但是对于所有的历史综合考虑，有类似公式(2.1.20)的限制条件成立，即

$$\sum_{\forall w_1^{n-1}} \sum_{w_n:c(w_1^n)>0} \delta_{c(w_1^n)} = \sum_{1\le r\le k} n_r(1-d_r)\frac{r}{N_T} = \frac{n_{1,T}}{N_T} \tag{2.2.1}$$

其中 N_T 表示不考虑历史时，所有单元出现的次数，$n_{1,T}$ 表示不考虑历史时出现次数为 1 的单元的个数。

如果单考虑折扣系数的计算，可以将基于全局折扣系数的平滑方法和其他方法做个比较。例如，绝对折扣平滑算法[34]不管单元次数多少，都统一扣除掉一个次数 D，实际上十分粗糙。K-N 平滑的改进方法[32]提出对出现次数为 1、2、3 的单元分别扣掉由公式(2.1.31)、(2.1.32)、(2.1.33)给出 D_1、D_2、D_3，本文使用表 2-1 的单元分布数据，针对 trigram，对 K-N 平滑的 D_1、D_2、D_3 进行计算，得出结果，并和 Katz 全局折扣因子比较如表 2-3。

表 2-3 K-N 平滑折扣数和 Katz 平滑折扣数比较

	C	1	2	3
K-N	折扣次数 D	0.737	1.176	1.470
	余下次数 C-D	0.263	0.824	1.530
Katz	折扣次数 $r\cdot(1-d_r)$	0.703	0.964	1.031
	余下次数 $d_r\cdot r$	0.297	1.036	1.969

从表中可以看出，两种方法在对单元次数的折扣上十分相似。其基本特点为：随着 r 的增加，被折扣掉的概率比例 $1-d_r$ 呈下降趋势，但被折扣掉的次数总数 $r\cdot(1-d_r)$ 呈增加趋势。但是相比较而言，Katz 平滑中折扣系数的计算基于图灵估计，由比较严密的数学推导产生；而 K-N 平滑中折扣次数 D_1、D_2、D_3 的计算原文中没有给出推导公式，只是作者的建议公式，

可以认为只是论文作者经验数据的拟合。从这点来说，Katz 平滑的折扣系数优于 K-N 平滑的折扣系数。

基于全局折扣系数的 Katz 平滑算法区别于传统平滑算法，本质上在于全局折扣方法只对全局做了一次图灵估计，而传统方法是对所有的单元历史一一做图灵估计的。采用全局折扣的思想也有文献提到[32]，但目前仍有不少文献在使用 Katz 平滑时采用基于历史的折扣系数[35, 36]，因此从实验上验证并指出这点仍有必要。实验结果表明，采用全局折扣 Katz 平滑的模型其性能远远高于采用局部折扣平滑的模型，语言模型的熵可以降低 40%以上。因此，局部折扣因子性能太差，不应采用。本文后续平滑实验的基准实验(Baseline)采用的是全局折扣因子，即在使用全局折扣因子的基础上再改进其性能。

2.2.2 单元可靠性对折扣系数的影响

使用全局折扣系数的 Katz 平滑还存在一些问题。在语音识别中，当两个单元的历史词是同音词时，用 Katz 平滑产生的概率比较可能会出现一定的矛盾。以 trigram 单元为例，根据折扣原理可知，给定单元历史 $h = w_1 w_2$，有

$$\begin{aligned} P_{\text{discounted}}(w_1 w_2) &= 1 - \sum_{w_3 : C(w_1 w_2 w_3) > 0} P_{\text{Katz}}(w_3 \mid w_1 w_2) \\ &= \sum_{r=1}^{r_T} \frac{(1 - d_{r,3}) \cdot r \cdot n_r}{C(w_1 w_2 w_3)} \end{aligned} \tag{2.2.2}$$

如果将表 2-1 中的数据代入公式(2.2.2)，则可以得到 $P_{\text{discounted}}(h_1)$=0.3226，$P_{\text{discounted}}(h_2)$=0.4606。从这个结果可以知道，虽然单元历史次数 $C(h_1)$和 $C(h_2)$分别为 628 和 9，但是二者折扣下来的概率却相差不大。仔细研究表 2-1 数据可以发现，这是因为双方大部分单元出现次数都非常小，集中在 r_T 以下，都参与了折扣，造成双方被保留下来的概率比例相差也不大，通过计算可得分别为 0.6774 和 0.5494。这种结果导致一个不合理的现象：h_1 保留下来的概率将被 628 个单元分配，而 h_2 保留下来的只被 9 个单元分配。这样可能导致大部分 $P_{katz}(w \mid h_1)$ 可能比 $P_{katz}(w \mid h_2)$ 小的

非常多。比如当 $h_1 = w_1w_2$、$h_2 = w_1'w_2$、$C(w_1w_2w_3)$=10 且 $C(w_1'w_2w_3)$=1 时，平滑可能出现不合理的地方。根据平滑公式，很容易计算得到

$$P_{katz}(w_3 \mid w_1w_2) = 10/628 = 0.016$$

$$P_{katz}(w_3 \mid w_1'w_2) = d_1 \square 1/9 = 0.033$$

显然

$$P_{katz}(w_3 \mid w_1w_2) < P_{katz}(w_3 \mid w_1'w_2)$$

如果 w_1 和 w_1' 在声学上发音相似，则在语音识别中 $P_{katz}(w_3 \mid w_1w_2)$ 就会发生 $P_{katz}(w_3 \mid w_1'w_2)$ 激烈的竞争，而概率的不合理将直接导致解码路径打分出现不合理的情况[37]，图 2-1 是示意图。

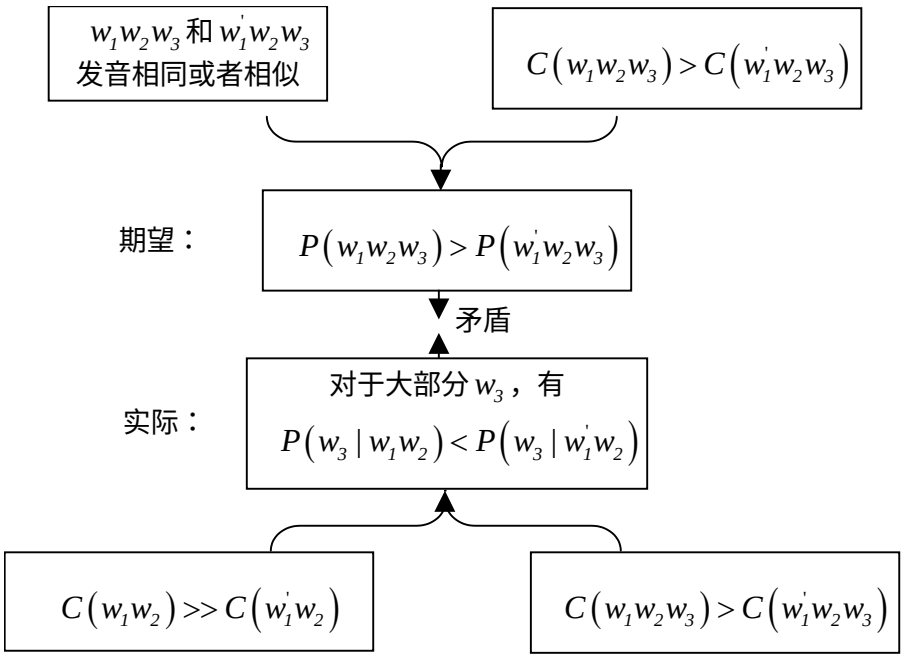

图 2-1 传统 Katz 平滑可能产生的不合理的平滑结果

因为如果把 $w_1w_2w_3$ 和 $w_1'w_2w_3$ 看成是事件空间的两个事件，则显然有 $P(w_1w_2w_3) > P(w_1'w_2w_3)$，是对单元 $w_1w_2w_3$ 有利；而如果考虑 Katz 平滑结果，则发现 $P(w_3 \mid w_1w_2) < P(w_3 \mid w_1'w_2)$，也就是说反而对单元 $w_1w_2w_3$ 不利。

产生这种矛盾的根本原因在于语言模型参数的可靠性问题。图 2-2 给出了两个单元折扣估计的可靠性示意。直观上讲，由于 628 比 9 要大的多，也就相对 9 更加具有统计意义，造成了 10/628 比 1/9 要可靠。具体的可靠性 2.3 小节给出了理论分析。

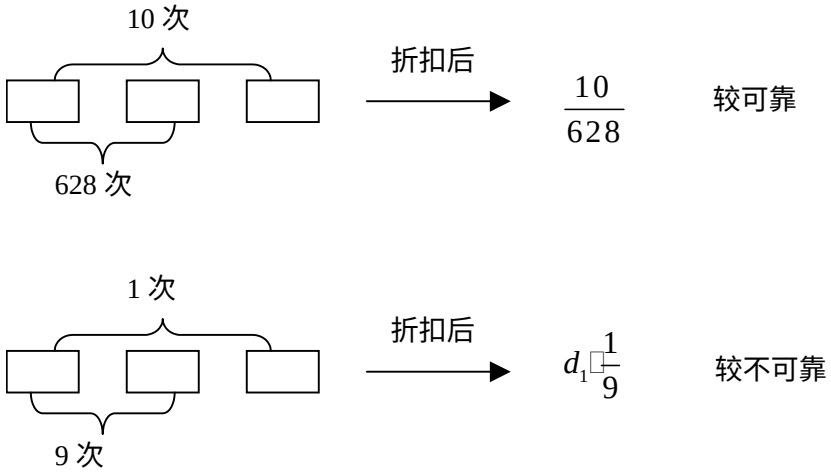

图 2-2 单元概率可靠性示意图

对于可靠性较差的单元，不能给予太大的概率。因为如果其概率过大，在不应该大于其他单元概率的时候大于，将导致其他单元不能领先，造成多个错误；而其概率过小，最坏的情况是只导致该单元概率该领先的时候没有领先，只影响到自身，而不会错误的将属于其他单元的领先位置占领。因此从直观上讲，对于此类可靠性较差的单元，其概率宁可过小，不可过大。

实际上，在增加更多的语料之前，图 2-1 中两个单元的概率 $P(w_3 \mid w_1 w_2)$ 和 $P(w_3 \mid w_1' w_2)$ 的大小关系实际并不能完全确定。但 $P(w_3 \mid w_1' w_2)$ 的可靠性较低，如果概率较大将对模型性能产生负面影响。以语音识别为例，由于音节本身就不是确定的(声学给出很多候选)，因此任意一句话的识别，$P(w_3 \mid w_1' w_2)$ 都可能影响解码路径，如果 $P(w_3 \mid w_1' w_2)$ 过大，将导致那些含 $P(w_3 \mid w_1' w_2)$ 的路径排名提升(实际上可能正确的结果中根本不含该单元，也就是说这种提升很可能是错误的)，最终导致解码出错。因此，理性的方法就是将不可靠单元的概率减少。2.3 小节将基于泊松过程假设检验给出了

单元可靠性的分析和量化措施。

2.3 基于泊松过程假设的Katz平滑方法

首先，本文提出一个假设，即给定任意 *n*-gram 单元 X，其在训练语料流中的出现是一个泊松过程(Poisson Process)，然后采用假设检验加以验证。有研究者对单词在文章中出现的时间间隔采用泊松分布建模[38, 39]，本文将从更大尺度上(即训练语料流，包含海量文献)用假设检验的方式证明任意语言单元的出现事件是一泊松过程。并对于出现次数不同的单元，求出其泊松参数的置信区间，以及其可能出现次数的置信区间，从而对出现次数的可靠性进行描述。

2.3.1 单元出现事件为泊松过程的假设检验

对于计数过程 $\{N(t), t \geq 0\}$，计数开始时时间记为 S_0，且 S_0=0，按顺序记录下事件发生的时间点为 $S_1, S_2, ..., S_n, ...$，两个相邻事件的时间间隔为 $X_i = S_i - S_{i-1}$。

$\{N(t), t \geq 0\}$ 是泊松过程(参数为 λ)的充分必要条件：其两个相邻事件的时间间隔 $\{X_n, n \geq 1\}$ 是独立且参数同为 λ 的指数分布[40]。

根据定义有 $S_n = \sum_{k=1}^{n} X_k$，可证明 S_n 的分布为[40]

$$P(S_n \leq t) = 1 - e^{-\lambda t}\left(1 + \lambda t + ... + \frac{(\lambda t)^{n-1}}{(n-1)!}\right) \tag{2.3.1}$$

如果把语料流的顺序作为时间顺序，语料流中每一个单词算一个时间单位。t 个单词过去后，单元 X 出现的次数记为 $N(t)$，因此要证明单元出现事件是泊松过程，只需要证明随机过程 $\{N(t), t \geq 0\}$ 是泊松过程。

按照泊松过程性质，要检验一个过程是否是泊松过程，可转化为多种检验问题之一。本文选择其中一种检验方法，表述为如下的假设检验命题。

假设检验问题的提出

对于随机过程$\{N(t),t\geq 0\}$，记录事件顺序出现的时间S_n(其中S_0=0)，给定 T>0，要检验过程是否为泊松过程，只需检验在 $N(t)$=n 下，$S_1,S_2,...,S_n$ 的条件分布是否与$[0,T]$上的 n 个独立均匀分布的顺序统计量的分布相同。

假设检验问题的检验

提出统计假设 H_0：$\{N(t),t\geq 0\}$是泊松过程，并记$\sigma_n=\sum_{k=1}^{n}S_k$，当 H_0 成立时，根据泊松性质公式(2.3.1)，可得

$$E\{\sigma_n \mid N(T)=n\}=E\left\{\sum_{i=1}^{n}Y_{(i)}\right\}=E\left\{\sum_{i=1}^{n}Y_i\right\}=\frac{nT}{2} \tag{2.3.2}$$

$$D\{\sigma_n \mid N(T)=n\}=D\left\{\sum_{i=1}^{n}Y_{(i)}\right\}=D\left\{\sum_{i=1}^{n}Y_i\right\}=\frac{nT^2}{12} \tag{2.3.3}$$

其中$\{Y_i,1\leq i\leq n\}$独立同分布，$Y_i\sim U[0,t]$。$Y_{(1)},Y_{(2)},...,Y_{(i)},...,Y_{(n)}$为其顺序统计量。利用独立同分布的中心极限定理，有

$$\lim_{n\to\infty}P\left\{\frac{\alpha_n-\frac{n}{2}T}{T\sqrt{\frac{n}{12}}}\leq x\middle|N(T)=n\right\}$$

$$=\lim_{n\to\infty}P\left(\frac{\sum_{i=1}^{n}Y_i-\frac{n}{2}T}{\sqrt{\frac{nT^2}{12}}}\leq x\right)$$

$$=\Phi(x)$$

$$=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{u}{2}du} \tag{2.3.4}$$

即对充分大的 n，有

$$P\left(\frac{\sigma_n}{T}\le\frac{1}{2}\left[n+x\left(\frac{n}{3}\right)^{1/2}\right]\middle|N(T)=n\right)\approx\Phi(x) \tag{2.3.5}$$

若给定置信水平 α=0.05，则当

$$\frac{\sigma_n}{T}\in\frac{1}{2}\left[n\pm1.96\left(\frac{n}{3}\right)^{1/2}\right] \tag{2.3.6}$$

时，接收假设 H_0，否则，拒绝 H_0。该方法不需要知道泊松过程的参数 λ。

为了检验 H_0，我们随机选取了长度 T 不等的语料流，对 unigram 单元进行统计发现，随着 T 的增加，越来越多的 unigram 单元的 H_0 假设成立，见表 2-4。

表 2-4 满足泊松过程的单元比例与训练语料流的大小关系

T	350,000	700,000	1,400,000	2,800,000
满足泊松过程的单元比例	40.1%	53.1%	67.4%	82.8%

随着统计语料增加，满足泊松过程 H_0 假设的单元比例在增加。有些单元之所以暂时没有满足 H_0 假设，是因为出现次数少，不具备假设检验的条件。因为作为泊松过程的时间间隔参数非常大，需要更多的语料才能检验 H_0。随着测试语料大小增加，发现这部分单元也是泊松过程。

对 bigram 等高阶段单元的检验结果类似。由此可得出结论：从统计上讲，统计语言模型中的任意单元 X，其在训练语料流中的出现事件可以是一泊松过程。

2.3.2 单元出现事件的泊松过程参数估计

本文已经通过上述假设检验证明单元在训练语料流中的出现事件是一泊松过程，但是不同的单元其相邻两次出现的事件间隔不一样，也就是泊松过程的参数不同。

单元 X 的出现事件是泊松过程，则根据泊松过程性质，可知 $N(t+s)$-$N(s)$

是一参数为λ的泊松分布。

对于泊松过程$\{N(t),t\geq 0\}$，λ参数未知，已知条件是 $N(t)=n$，则可以对参数λ的大小和置信区间进行估计。

(1) λ极大似然估计

假设给定 T,在$[0,T]$上顺序观察到 n 个事件的时间$S_1,S_1,...,S_n$取值为$t_1,t_1,...,t_n\leq T$，则似然函数为

$$L(t_1,t_2,...,t_n)=\lambda^n e^{-\lambda T} \tag{2.3.7}$$

令$\frac{dL}{d\lambda}=0$，即得λ的极大似然估计为

$$\hat{\lambda}_L=\frac{n}{T} \tag{2.3.8}$$

n 越大，则$\hat{\lambda}_L$越大，可见λ参数的极大似然估计随 n 增大而增大。

(2) λ区间估计

假设给定 T,在$[0,T]$上顺序观察到 n 个事件的时间$S_1,S_1,...,S_n$，则可以证明S_n的概率密度分布函数为[40]

$$f_n(t)=\frac{\lambda(\lambda t)^{n-1}}{(n-1)!}e^{-\lambda t}=\frac{\lambda^n}{\Gamma(n)}t^{n-1}e^{-\lambda t},\quad (t\geq 0) \tag{2.3.9}$$

其中$\Gamma(\alpha)=\int_0^\infty x^{\alpha-1}e^{-x}dx$，为Γ函数，$\Gamma(n)=(n-1)!$。因此，$2\lambda S_n$的概率密度函数为

$$g_n(t)=\frac{1}{2^{\frac{2n}{2}}\Gamma\left(\frac{2n}{2}\right)}t^{\frac{2n}{2}-1}e^{-\frac{t}{2}},\qquad (t\geq 0) \tag{2.3.10}$$

这与$\chi^2(2n)$的密度相同，故$2\lambda S_n=\chi^2(2n)$。取置信度为$1-\alpha$，则

$$P\left(\chi^2_{\frac{\alpha}{2}}(2n)\le 2\lambda S_n\le\chi^2_{1-\frac{\alpha}{2}}(2n)\right)=1-\alpha \tag{2.3.11}$$

故置信度为$1-\alpha$的区间估计为

$$\left[\frac{\chi^2_{\frac{\alpha}{2}}(2n)}{2S_n},\frac{\chi^2_{1-\frac{\alpha}{2}}(2n)}{2S_n}\right] \tag{2.3.12}$$

χ^2分布的取值可以查表获得。S_n可由实验数据得到。假设 $S_n=T$，即最后一个事件观察到时马上对λ的区间进行估计，区间进一步确定为

$$\left[\frac{\chi^2_{\frac{\alpha}{2}}(2n)}{2T},\frac{\chi^2_{1-\frac{\alpha}{2}}(2n)}{2T}\right] \tag{2.3.13}$$

这样，我们可以获得当 n 分别为不同值时，λ的置信区间(α取 0.05)。又由于$E(N(t))=\lambda t$，因此λ取其置信区间上下时可以得到 $t=T$ 时出现次数 n 的置信区间上下限

$$\left[\frac{\chi^2_{\frac{\alpha}{2}}(2n)}{2},\frac{\chi^2_{1-\frac{\alpha}{2}}(2n)}{2}\right] \tag{2.3.14}$$

而且，可以求得λ的归一化上下限区间[$\lambda_{down}/\hat{\lambda}_L$，$\lambda_{up}/\hat{\lambda}_L$]和 n 的归一化上下限区间[n_{down}/n，n_{up}/n]相同，为

$$\left[\frac{\chi^2_{\frac{\alpha}{2}}(2n)}{2n},\frac{\chi^2_{1-\frac{\alpha}{2}}(2n)}{2n}\right] \tag{2.3.15}$$

综上，结果如表 2-5 所示，上下限分别用下标 *up* 和 *down* 表示。

从表中可以看出，n 很小时，λ和 n 置信区间的下限非常小，上限则很大。比如单元出现次数只有 1 次时，其置信区间的下限只有 0.03。随着 n 增大，归一化区间下限和上限都趋向于 1。

表 2-5 不同 n 取值时 λ 的置信区间及实际可能出现次数置信区间

n	1	2	4	10	20
极大似然估计 $\hat{\lambda}_L$	$1/T$	$2/T$	$4/T$	$10/T$	20/T
λ 区间下限 λ_{down}	$0.03/T$	$0.24/T$	$1.09/T$	$4.80/T$	$12.22/T$
λ 区间上限 λ_{up}	$3.69/T$	$5.57/T$	$8.77/T$	$17.08/T$	$29.67/T$
n 置信区间下限 n_{down}	0.03	0.24	1.09	4.60	$12.22/T$
n 置信区间上限 n_{up}	3.69	5.57	8.77	17.08	$29.67/T$
λ 和 n 归一化下限	0.03	0.12	0.27	0.48	0.61
λ 和 n 归一化上限	3.69	2.79	2.19	1.71	1.48

(续表)

n	40	200	500	1,000	5,000
极大似然估计 $\hat{\lambda}_L$	$40/T$	$100/T$	$500/T$	$1,000/T$	5,000/T
λ 区间下限 λ_{down}	$28.58/T$	$81.36/T$	$457.13/T$	$938.95/T$	$4,862.35/T$
λ 区间上限 λ_{up}	$53.31/T$	$120.53/T$	$544.77/T$	$1,062.90/T$	$5,139.50/T$
n 置信区间下限 n_{down}	28.58	81.36	457.13	938.95	4,862.35
n 置信区间上限 n_{up}	53.31	120.53	544.77	1,062.90	5,139.50
λ 和 n 归一化下限	0.71	0.81	0.91	0.94	0.97
λ 和 n 归一化上限	1.33	1.21	1.09	1.06	1.03

这里需要关注的是 n 的可信度，可以画出 n 的归一化置信区间上下限和 n 的对数关系曲线如图 2-3 所示。从图中可以看出，n 的归一化置信区间上下限随着 n 增大都趋向于 1，也就是说 n 的可靠性越来越强。

可靠性差的单元概率，我们宁可让其稍小，因为其概率即使过分小，最坏的情况是其概率比应有值小的多，最多只影响到该单元自身在解码中竞争不过其他单元，影响了它自己一个单元。如果其概率过大，甚至超过了它不应超过的那些竞争者的概率，则在解码中，该单元的过于活跃将可能屏蔽其竞争者，或者，当其竞争者较多时，可能就会造成了更多的错误。

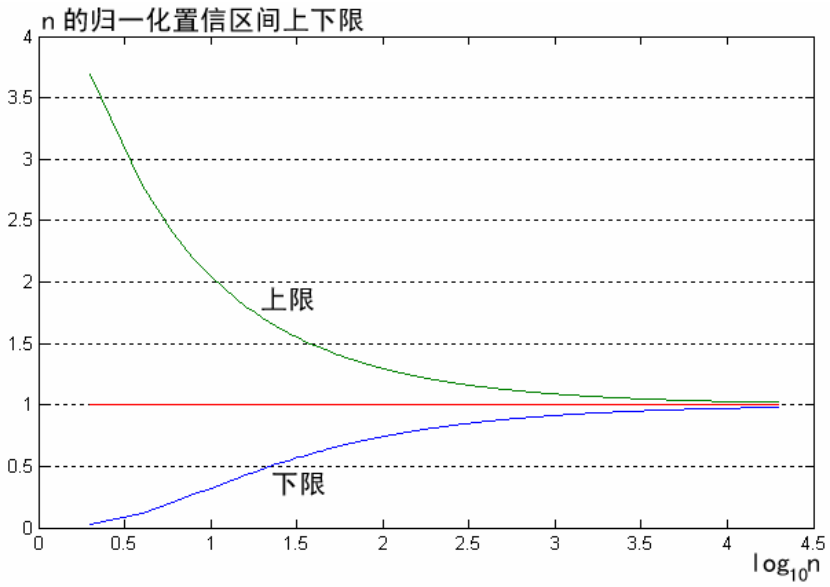

图 2-3 出现次数 *n* 的归一化置信区间上下限随 *n* 变化曲线

2.3.3 基于泊松过程假设的 Katz 平滑方法实现

根据以上分析，本文提出的基于泊松过程假设的 Katz 平滑采用如下方法：

第一，根据实验采用所有单元的分布计算折扣系数，作为折扣的基础。Katz 的原文对计算折扣系数 *dr* 的说明过于简单，特别是公式(2.1.22)中 *dr* 和 n_r 都没有给出下标。但按照其前面的推导，特别是根据公式(2.1.20)的描述可以知道，Katz 平滑的原意是使用基于历史的折扣系数。否则公式(2.1.20)应表述为公式(2.2.1)所示。使用全局因子的思想 Stanley F. Chen 也曾指出[32]，但仍有部分文献[35, 36]在使用 Katz 平滑时采用基于历史的折扣系数，因此指出这点仍非常必要。在后面的实验中，采用全局折扣 Katz 平滑算法训练的模型作为实验的基准(Baseline)。

第二，从泊松过程假设得出结论：单元的出现次数不同，会导致单元概率可靠性的不同。为了描述这种概率可靠性差异，本文提出采用单元可靠性因子，降低那些不可靠单元的概率。对于一个可靠性较差的单元(出现次数较小)，如果其历史出现次数较大，则该单元的概率已经比较小，对其

他单元的威胁并不大；只有那些本身出现次数较少，而且其历史出现次数也较小的单元，按照传统平滑方法其概率将较大，因此需要对其概率作出限制。即可靠性因子应该基于单元历史计算，记为 $R(h)$。

根据表 2-5 的数据对单元概率可靠性进行估计，并综合考虑其对模型性能的影响，提出 $R(h)$采用公式

$$R(h)=\left(1-0.75e^{\frac{-C(h)}{5r_T}}\right) \tag{2.3.16}$$

$R(h)$和 $C(h)$的关系如图 2-4。

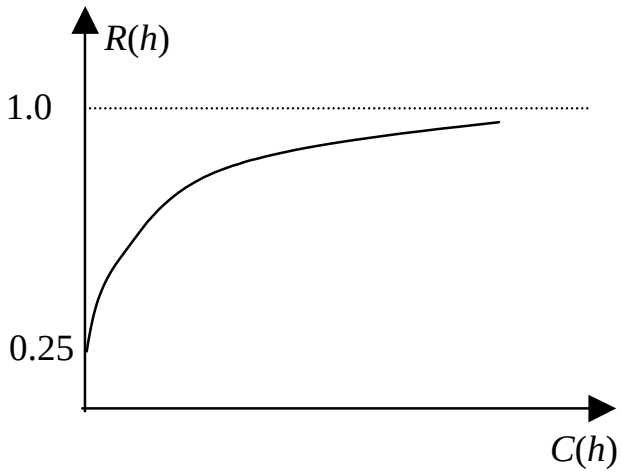

图 2-4 可靠性因子和单元历史次数之间的关系

这样，综合全局折扣系数和可靠性因子，最后得到折扣系数 $d_r(h)$的公式为

$$d_r(h)=\left(1-0.75e^{\frac{-C(h)}{5r_T}}\right)\frac{\frac{r^*}{r}-\frac{(k+1)n_{r+1}}{n_1}}{1-\frac{(k+1)n_{r+1}}{n_1}} \tag{2.3.17}$$

其中 n_r 是所有单元中(单元任意历史)出现了 r 次的单元个数。

在确定了折扣系数 $d_r(h)$ 后，$\alpha(h)$ 按照传统 Katz 平滑的方式计算。

因此，基于泊松过程假设的 Katz 平滑在计算折扣系数时，先抛开了单元历史信息，指出用全局折扣代替基于历史的折扣；然后又引入新的参数 $R(h)$ 描述历史差异。

2.4 实验和分析

本文分别从困惑度(Perplexity)和音字转换正确率两个方面来评价新方法的有效性。

模型训练所用语料大小为 200M 文本(字数)，该语料库来源于 93－94、96－97 四年的《人民日报》，92、94 年的《经济日报》，94 年的《市场报》和 94－96 三年的《新华社》文稿，共计 2 亿字，内容涵盖政治、经济、体育、文化等多个领域。标注语料库由北京工业大学人工智能实验室完成[41]。词表大小为 51,007。

用于测试的语料文本三个。语料 A(1,801 句，23,310 字)为 863 语音库的录音文本，和训练语料很接近，因此其困惑度较小；语料 B(375 句，3,466 字)来自香港凤凰卫视的网站新闻(http://www.phoenixtv.com)；语料 C(25,457 句，231,606 字)是一本政治方面的书籍，和训练语料相差较大。

2.4.1 困惑度比较实验

表 2-6 trigram 模型困惑度比较

	语料 A	语料 B	语料 C
全局折扣的传统 Katz 平滑	54	253	401
基于泊松过程假设的 Katz 平滑	46	223	360
困惑度下降	14.8%	11.9%	10.0%

实验结果表明，不论测试语料和训练数据是否同领域，模型困惑度都有较大下降。表明基于泊松过程假设的 Katz 平滑方法在模型描述方面的能力更强，能从有限的语料中较好的估计出那些没有出现的单元的概率情

况。

2.4.2 音转字评估

我们将传统平滑算法和基于泊松过程假设的 Katz 平滑方法训练所得模型分别进行音字转换的批量实验，采用 trigram 模型，汉字错误率的改善情况见表 2-7 所示。

表 2-7 传统 Katz 平滑方法和基于泊松过程假设的 Katz 平滑方法音转字错误率比较

	语料 A	语料 B	语料 C
全局折扣的传统 Katz 平滑	1.53%	9.49%	15.21%
基于泊松过程假设的 Katz 平滑	1.27%	8.11%	13.87%
错误率度下降	17.0%	14.5%	8.8%

从实验结果看，音转字的错误率明显下降。

2.4.3 语音识别实验

将语言模型用于大词表连续语音识别，测试语料来自 863 的测试集，521 句，6751 字。批量测试结果如表 2-8 所示。

表 2-8 连续语音识别汉字错误率比较

	bigram	trigram
全局折扣的传统 Katz 平滑	7.44%	7.04%
改进的 Katz 平滑	7.38%	3.55%
错误率度下降率	0.8%	49.6%

从表中可以看出,用基于泊松过程假设的 Katz 平滑方法训练的 bigram 模型对语音识别的改进并不大，但用训练的 trigram 模型则性能表现优异。因为 bigram 模型对语言单元连接关系的刻画能力相对较小，而且参数较少，因此使用精细的建模方法并不能提高太多性能；相反，trigram 模型参数多得多，参数之间的竞争更激烈，采用新方法后，提高了描述参数之间差异的能力，使得模型更加准确，从而提高了语言模型的性能。比较表 2-8 和表 2-7 发现，新方法对语音识别正确率的提高比音字转换更明显。其原

因在于，在语音识别中，任何单元都有可能是路径的一部分，其搜索空间比音字转换更大，好的模型平滑算法减少了那些可靠性单元较低的单元对解码的误导。

2.5 小结

本章通过对传统语言模型 Katz 平滑算法的分析，指出传统方法存在的不足：根据 Katz 平滑论文，折扣系数的计算是按照单元历史分别计算的；这种基于单元历史的计算方法非常容易受到数据稀疏的影响，造成折扣系数估计的不准确。本文指出应该使用全局折扣系数的平滑算法(本章实验的 Baseline)，并进一步阐述了全局折扣系数的局限性。首先，从直观上看，出现次数较小的历史单元可靠性较小；接着，本文提出一个假设：语言单元，即 n 元组在训练语料中的出现是一泊松过程。然后用假设检验的方法对该假设加以验证，在置信度为 0.95 的条件下，实验结果表明该假设成立，并求出了出现次数不同的单元其期望次数置信区间的上下限。次数较低的历史单元，其期望次数的置信区间比较大，进而从理论上说明了出现次数不同的历史单元对应的概率可靠性不一样。在平滑时，对于概率可靠性较低的单元，使其概率宁小勿大。因为概率过小只会使其竞争不过其他单元，至多造成该单元不会在候选路径中出现；而使其概率过大则会使该单元在竞争中过于活跃，在不该领先时领先，屏蔽了其他单元，反而造成更多错误。

总之，本章提出的基于泊松过程假设的 Katz 平滑算法，很好的解决了语言模型平滑中折扣系数的估计问题。实验结果也证明了，使用该方法能显著降低模型困惑度。批量测试表明，新方法能显著降低拼音转汉字和连续语音识别的的字错误率。

第三章　统计语言模型在线递增式自适应方法

由于出现了大量可使用的语料库，统计语言模型技术得到了很大的成功。但同时，语料库语言学(Corpus Linguistics)研究发现：词的频度、连接关系等和统计所用的特定语料库有着非常密切的关系，而且在不同的语料库中的表现相差可能非常巨大。人们在获得特定的语料并据此训练成特定语言模型后，有可能将该语料应用到很多不同场合的应用中去，并可能产生应用场合和训练语料不匹配的问题，导致语言模型性能的显著下降[22]。

一般来说，引起语言模型应用场合和训练语料不匹配的原因可能有以下几个：

一、领域不匹配。语言作为人类社会交流的主要工具之一，由此产生的文本资料五花八门、无所不包，它们涉及到人类社会的方方面面。每一个不同的领域，不仅有各自不同的专业领域的词汇，而且其文风、措辞习惯等都相差很远，比如文学批评专著和财经新闻语料之间的差异就非常之大。

二．由地域造成的语料差异。这一点一般和方言相关，中国分八大方言区，虽然绝大部分方言区和标准普通话使用相同的汉字，但由于其发音、措辞习惯等和标准普通话存在较大差距，因此一般称之为“方言普通话”。比如，四川人说普通话往往喜欢以“哈”字结束，根据我们对成都地区发送的手机短消息文本的分析，1.6%的句子以“哈”字结尾，如“不好意思哈”等，远远高于标准普通话。除此之外，政治原因也造成一些不同地域之间语言习惯的不同。典型例子就是大陆、香港和台湾三地语言习惯的差异。比如台湾国语使用“汰换”一词，指“淘汰并更换掉”，大陆用词如“超编”，指“超出人员编制”，这个两个词都有非常强的地域性。这种差异使得用标准汉语文本训练的语言模型，在应用到其他地区时导致性能下降。

三、个人的使用习惯上的差异。每个人由于教育、经历等的不同，造成其在语言使用习惯上的差异。因此训练好的语言模型并不一定最适合当

前用户。

这样，就有必要设计某种调节语言模型参数的算法以提高其适应能力，这种算法就是语言模型自适应(Language Model Adaptation)技术。

本章的内容安排如下：第 1 小节介绍传统语言模型自适应方法及其适用场合；第 2 小节通过分析现有方法的不足提出了在线递增式语言模型自适应框架；第 3 小节讨论了在线递增式自适应理论方法和实现中可能存在的问题及解决方法；第 4 小节给出了实验结果和讨论。第 5 小节是本章小结。

3.1 传统的统计语言模型自适应方法

被广泛使用并深入研究的传统统计语言模型自适应方法主要有以下三种：

3.1.1 基于缓存的语言模型

基于缓存的语言模型(Cache Language Models)自适应技术[42, 43]的基本思想是基于这样一个假设：用户在使用语言模型时话题具有短时稳定性，也就是说在一定时间内一般集中于某一个话题。这样，就可以动态的使用当前文本训练出一个临时模型。由于当前文本语料比较少，一般采用 bigram 模型，较少使用 trigram 模型或者更高次的模型。

如果将单元在静态模型中的概率记为 P_s，动态 Cache 模型中的概率记为 P_{cache}，自适应后概率记为 $P_{adapted}$，则基于缓存的语言模型可以用以下公式描述：

$$\begin{aligned} &P_{adapted}\left(w_i \mid w_{i-n+1}...w_{i-1}\right) \\ &=\lambda_c P_s\left(w_i \mid w_{i-n+1}...w_{i-1}\right)+\left(1-\lambda_c\right)P_{cache}\left(w_i \mid w_{i-n+1}...w_{i-1}\right) \end{aligned} \tag{3.1.1}$$

其中插值系数 λ_c 可以根据语料来选择，如果当前文本规模较大则可以设置的相对大一些。通常情况下，一篇文章中某些特定的词汇和表达方式总是会重复的出现，因此，这个方法在当前文本规模较大时效果比较明显，能保证一些固定组合被正确挑选出来。

另外，如果语言模型的应用场合话题比较固定，只是和训练领域不相匹配，且应用数据难以大量搜集，也可以使用类似基于缓存的自适应技术，即用预先收集的应用领域的有限语料训练出一个和领域相关的模型，然后和静态模型插值使用。

3.1.2 话题自适应模型

在语音识别中、OCR 或者用整句输入法输入文章的过程中，话题通常是非常不固定的[42-46]，

话题自适应(Topic-Adaptive Models)[46-48]也称领域自适应(Domain-Adaptive Models)[49]，其目的是为了抓住当前话题的语言特征。

语言模型使用过程中的话题可能在不断变化，但是话题信息在语言模型中又扮演着非常重要的角色。比如，如果我们知道当前领域是计算机相关领域，则可以知道 *n* 元组(操作，系统)出现的概率应该远远大于(操作，车间)的概率；相反，如果我们知道当前领域和工业生产相关，则可以认为(操作，车间)的概率远远大于(操作，系统)的概率。

话题自适应模型的基本思想是这样：用于训练通用模型(或者静态模型)的语料构成十分庞杂，它一般是各种典型领域语料的复合体。因此，自然而然的想法就是试图将训练语料根据话题划分为几个部分(划分语料的方法可以采用自动聚类方法或者手工划分)，然后针对每个话题的语料分别训练各自话题的语言模型。使用时需要将话题相关的各个语言模型组合在一起，一般使用线性插值的方法。在使用过程中根据话题侦测的结果，动态的调整插值的参数。这种插值可以是 *n*-gram 层面的，也可以是句子层面的。比如 *n*-gram 级别的插值可以表示为

$$\begin{aligned} &P_{adapted}\left(w_i \mid w_{i-n+1}...w_{i-1}\right) \\ &=\sum_{i=1}^{T}\lambda_i P_i\left(w_i \mid w_{i-n+1}...w_{i-1}\right) \end{aligned} \tag{3.1.2}$$

式中必须保证 $\sum_{i}^{T}\lambda_i = 1$。线性插值方法相对于只使用单个话题模型，有以下优点：通过调整权重参数，可以突出某个话题的分量，而其他权重较

小的话题模型则可以防止出现因话题完全不匹配所引起的性能恶化。

也有的话题自适应方法采用非预先分类的方法建模[46]。这种做法的理由是考虑到当前话题的侦测比较困难，而且预先对语料进行划分可能会不全面，划分的不合理可能导致模型应用性能不高。比如说，当前语音识别的话题是关于一个足球运动员的腿部受伤问题，而预先定义的话题包括“足球运动”和“腿部受伤”，那么解决的方法是将这两个话题的语料合并。但是最理想的训练语料集却应该是这两个集合的“交集”，而不是他们的并集。因此，如果当前文档是若干个话题的组合，那就需要将这些集合的语料都组合起来，这种组合的方式随着话题个数呈指数次增长，而且要将这些话题都定义出来也绝非易事。

于是，研究者提出一种略微不同的方法，新方法根据当前文档的历史情况选择最合适的语料集用于构建适合当前文档的语言模型。比如，可以根据当前历史信息采用信息提取技术[50]对所有的训练集语料进行一次“查询”，根据查询的结果，训练集语料文档依当前历史的相关程度排序。选出最相关的语料用于训练一个“话题相关”的语言模型。这个过程可以随着当前文档的更新不断重复进行。

因此，在具体的实现上，本方法由两个重要部分组成。第一步，根据当前文档的历史，采用信息提取的方式，对训练语料进行“查询”，得到当前话题的相关语料集；第二步，用所获得的相关语料对通用模型(或者话题无关模型)进行自适应。可见，当前文档的历史在整个自适应过程中扮演非常重要的角色。因此，在实时系统等有时间延迟要求的系统中，一般就使用当前文档已经处理完毕的部分。而在批处理系统中，比如允许多遍处理的系统，则可以将前一遍处理所得的通篇信息当成历史。比如，在允许双遍搜索的语音识别中，可以将前遍识别的结果用做信息提取查询的输入。

最常用的信息提取技术是 *TFIDF* 方法[50]，该方法用于从训练语料查找相似文档。其中 *TF* 表示词频(Term Frequency)，tf_{ij} 表示第 j 个单词在第 i 篇文档中出现的次数，也就是第 i 篇文档中出现的第 j 个单词的 unigram 个

数。*IDF* 表示倒文档频率(Inverse Document Frequency)，idf_j 表示出现了第 j 个单词的文档个数占所有文档比例的倒数，即

$$idf_j = \frac{\text{所有文档总数}}{\text{出现了第j个单词的文档数}} \tag{3.1.3}$$

而 *TFIDF* 指标则定义为

$$TFIDF_{ij} = tf_{ij} \log(idf_j) \tag{3.1.4}$$

这个组合指标可以这么理解：它是 *TF* 和 *IDF* 两个参数的组合，其中 *TF* 用于保证文档中词频较高的那些词对应的 *TFIDF* 值比较高，而 *IDF* 则用于衡量单词的话题相关性程度。比如，单词“的”具有非常低的话题相关性，因为它几乎出现在任何一个话题的文章中，而单词“仿生学”则具有较高的话题相关性，只有在特定的话题中才会出现。*IDF* 参数保证那些话题相关性较高的单词在话题判定时具有较高的影响力。

类似的，用于判定两篇文档相似程度的指标可以用公式表示

$$Similarity(D_i, D_j) = \frac{\sum_k tfidf_{ik} * tfidf_{jk}}{\sqrt{\sum_k \left(tfidf_{ik}\right)^2 * \sum_k \left(tfidf_{jk}\right)^2}} \tag{3.1.5}$$

因此，可以根据历史的不同(几个单词构成的较短历史或者由第一遍识别得到的整篇文章构成的历史)，分别选择不同的参数用于文档的抽取。将抽取得到的文档按与当前文档的相似度排序，将相似度最高的部分文档用于训练话题相关模型[46]。

这种不预先划分主题的方法处理起来更加灵活，但同时对信息抽取的准确性依赖比较大。

3.1.3 最大熵模型

前面提到的方法在使用自适应信息时都是先训练出几个模型(每个模型和当前话题相关程度不一样)，然后用插值方法获得它们组合的新概率。

最大熵模型(Maximum Entropy Models)这样考虑建模问题：将多个 *n*-gram 模型组合起来，本质上讲，是如何综合使用多个信息源的问题。实际应用中，最大熵方法[11, 51]就是一种更加通用的使用多数据源的方法。最大熵方法的目标是构建一个单一的模型，用该模型将几个信息源提供的信息综合起来。而单个信息源可以看做是对模型的一组限制条件，比如，要求模型满足一组边缘分布条件。因此，所有信息源的组合实际上最后形成一组概率分布的限制函数。而最大熵的目标就是找到在这些限制条件下使得概率分布图形状呈最“扁平”状的分布函数。

在最大熵计算过程中，分两步进行：

一、将所有的信息源限制条件用可计算的方式表示出来，比如表示成一些概率分布的边缘分布条件；

二、在所有符合这些限制条件的分布中，选择具有最大熵的分布作为目标模型。

如果我们用 $\{X\}$ 表示模型讨论的所有事件构成的空间，则 $P(X)$ 表示目标模型在该空间上的概率分布，那么限制条件则和该空间内的某一子空间上的某特定函数 $f_i(X)$ 是相关的。限制条件可以写成下式：

$$\sum_X P(X) f_i(X) = E_i \tag{3.1.6}$$

其中 E_i 是函数 $f_i(X)$ 在限定条件下的期望值，通常就是 $P(X)$ 的边缘分布。比如，在 n-gram 模型中，上式就可以重写成 trigram、bigram、unigram 概率的分布限制。以 unigram 为例子，其限制可以写为

$$f_{w_1}(w) = \begin{cases} 1 & \text{如果} w = w_1 \\ 0 & \text{其他情况} \end{cases} \tag{3.1.7}$$

而 E_{w1} 则就是根据训练语料得到的先验期望值，即

$$\sum_{w\in\text{训练数据}} f_{w1}(w) / N \tag{3.1.8}$$

N 为训练数据的大小。相应的限制条件可以写为

$$\sum_{h} P(h) \sum_{w} P(w|h) f_{w1}(w) = E_{w1} \tag{3.1.9}$$

代入不同的 w_1，则上式构成了一组限制条件。目标模型的要求就是在满足这些限制的条件下，尽可能的不偏离某一给定分布 $Q(X)$，即保证下式分歧函数(Divergence Function)最小

$$\sum_{X} P(X) \log \frac{P(X)}{Q(X)} \tag{3.1.10}$$

由于熵函数和分歧函数就相差一个负号，因此保证分歧最小实际就是要求 $P(X)$ 和 $Q(X)$ 分布的互熵最大。通常情况下 $Q(X)$ 采用平均分布(所有的信息都被看成是限制条件，在限制条件发生作用前没有任何信息，故为平均分布)，因此 $P(X)$ 相当于在满足限制条件的情况下最“ 扁平 ”的分布。可以证明 $P(X)$ 的解形式为[52]

$$P(X) \propto \prod_{i} u_i^{f_i(X)} \tag{3.1.11}$$

其中常数 u_i 需要迭代才能求出。

最大熵方法是 Ronald Rosenfeld 在其博士论文[11]中提出的方法，该方法建模比较复杂。在做自适应时可以将自适应信息当成是一些新的限制条件，但在实际实现时基本没有发现该方法相对于插值方法有什么优势[1]。最大熵模型的更多信息可以从一些论文中[11, 53-58]得到。

总的来说，以上的方法中，前两种都是基于同一个假设，即用户使用语言模型时具有话题短时稳定性。但是在实际应用中，存在两个问题：一是这个假设并不总是成立，很多时候当前文本并不具有固定的主题；二是这种方法只能捕获当前短时内的文本信息用于指导自适应，用户的过去文本没有得到应用。最大熵方法对自适应语料的限制虽然不限于短时文本，但是由于其实现复杂和有限的作用，也在实际系统中应用很少。

3.2 语言模型在线递增式自适应方法

3.2.1 自适应框架流程和通用模型的设计

传统的基于缓存的自适应方法和话题自适应方法可用图 3-1 表示其流程(左图为基于缓存的方法)。从这两个流程可以看出，这类方法有两个缺点，一是短时话题不稳定时自适应无效，二是较久远之前的文本输出对自适应没有起到应有的作用。而且这两类方法对用户语言风格方面的自适应很难起作用。

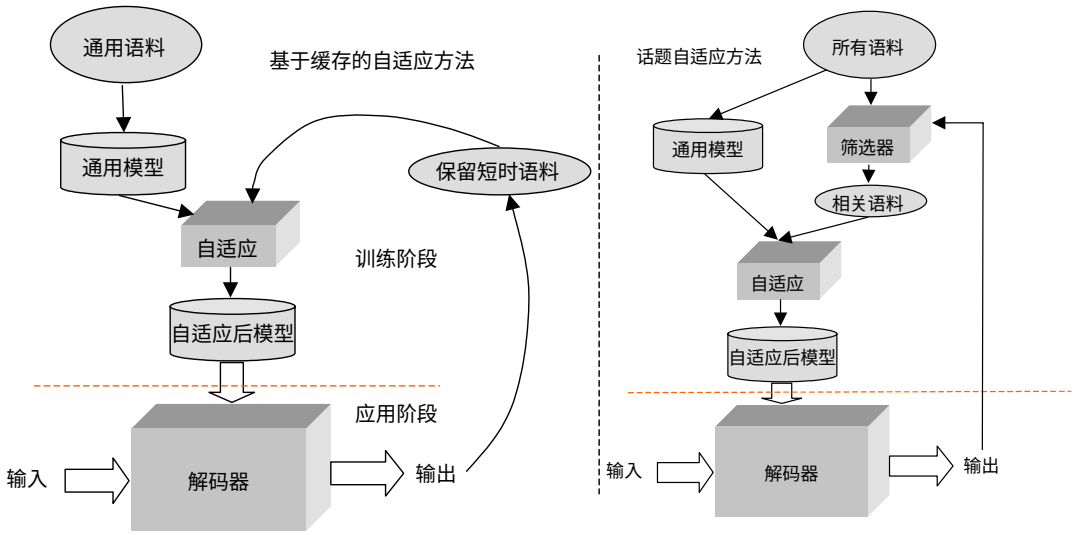

图 3-1 基于缓存的自适应方法和话题自适应方法示意图

针对这些问题，本文提出一种在线的递增式语言模型自适应方法[59]。在线自适应的一个特点是自适应语料在线产生，其典型的应用通常如以下几种：

1)　语言模型用于连续语音听写；

2)　语言模型用于中文拼音整句输入法；

3)　语言模型用于 OCR 识别。

这几种情况的一个共同特点是：将语言模型应用到某个具体的应用场

合，获得结果文本 S，用户校正后得到校正文本 S_M。

因此，最后系统实际上能够得到 S 和 S_M 两个信息源用于自适应。

本文提出的在线递增式自适应模型基于这样一个基本事实：用户应用语言模型时，不管当前话题是什么，不管当前的话题是固定的还是变化的，至少当前的文本对未来(可能是马上，也可能相对遥远一些的未来)是有指导作用的。这样，本文抛开了由于当前话题是否短时稳定而带来的自适应是否有效的问题，通过对语言模型应用过程中输出反馈的学习，递增的提高语言模型的性能。因此，本文将该方法命名为在线的递增式语言模型自适应方法。其框架可以用图 3-2 表示。

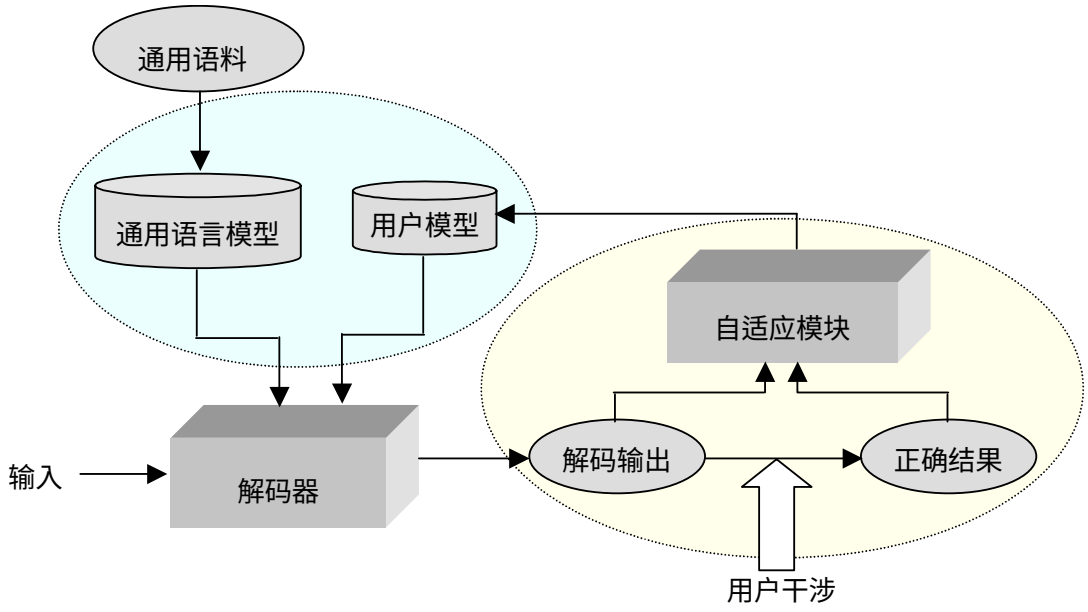

图 3-2 在线递增式语言模型自适应框架

在这个框架中，整个语言模型由通用模型和用户模型构成，通用模型从海量的数据中训练得到，语料构成比较均衡，已经比较充分的反应了一些最常见的语言单元的连接关系。用户模型是根据解码输出和经过用户修正的正确结果通过自适应模型生成。用户模型本身不构成独立的语言模型，它是背景模型的补充，用于补偿背景模型和实际应用之间的差异。这

种结构的好处有：

（1） 模型修改迅速简单，容易满足在线递增式自适应对速度方面的要求；

（2） 模型管理简单。而且由于保留了原来的背景模型，可以方便的估计现有模型和原来模型之间的差异，能够保证在线自适应不会偏离原来模型太多，还能保证在少数情况下当自适应发生错误时，抛弃这种错误。

图 3-2 中的通用模型采用我们第二章提出的改进的语言模型平滑算法，以保证最大可能的挖掘通用模型的潜力。

图 3-2 中的解码器根据具体的应用选择，比如语音识别，OCR 或者中文整句输入法等。

3.2.2 自适应框架有效性实验

通常情况下，训练通用模型的所用的通用语料库非常大，一般达到几百 MB 到几个 GB 的规模 ;而自适应过程中能够收集到的语料却非常有限，相对于通用语料来说几乎是微不足道的，那么少的数据量，对模型性能是否能产生影响呢？或者说少量的领域内数据究竟对模型的性能改善起到多大的作用。

针对该问题我们设计了一个统计语言模型性能和训练数据量关系的实验。

该实验针对目前人们日常生活中普遍使用的手机短消息领域，进行语言模型建模和音字转换测试。以 500 句手机短消息为测试语料，采用拼音转汉字的方式，对模型性能进行检验。训练模型所用语料大小从 0 开始，逐渐增加，最多达到 2.6MB(训练数据不参与测试)，测试语料不参与训练，词表大小为 51,158 个单词，结果如表 3-1。

从表 3-1 看出，在领域内数据从 0 开始增加时，模型的性能提升速度非常迅速。

表 3-1 训练语料大小和模型性能关系

训练语料大小(字节)	音字转换正确率(%)
0	70.44
156	71.95
299	74.43
595	80.68
1,196	82.18
2,530	85.83
5,083	87.37
10,156	87.65
20,331	88.59
40,621	90.52
81,250	91.61
162,507	92.64
325,012	93.28
650,015	93.87
1,300,013	94.87
2,600,006	95.01

其中训练数据量非常少时，训练数据量和模型性能关系如图 3-3。

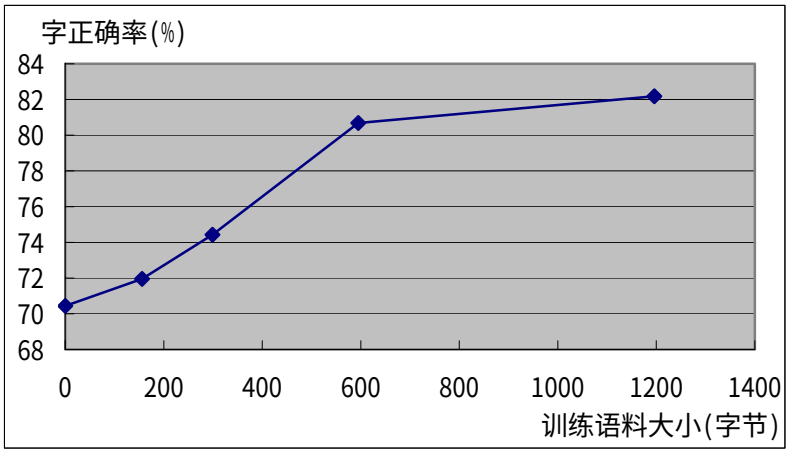

图 3-3 训练数据较少时训练数据量和模型性能关系

手机短消息领域涉及到日常生活的方方面面，包括工作、交通、饮食、娱乐、购物、品牌、旅游、体育等各个方面，测试语料和训练语料都同属手机短消息领域，关系密切，因此其实验结果从某种程度上反应了领域内语料对语言模型性能的重要性。

该实验提供了这样一个重要的信息：**训练语料和测试语料同属一个领域时，非常少(几百字节)的训练语料就能对模型性能起到改进作用**。

该结果对关于在线自适应方法收集到的语料是否太少的问题做了明确的解答，即：在线自适应收集到的同领域的语料虽然少，但是却有重大的意义。而且随着自适应过程的持续进行，得到的自适应语料也将逐渐增加。

也有文献[22]给出类似的分析结果，即：1MB 领域内的数据，其对模型性能的改善超过 30MB 领域外数据的影响。

3.2.3 基于 MAP 方法的自适应参数更新——理论方法

在语言模型自适应中，最重要的是如何将新的自适应语料信息结合到背景模型中去，即需要采用一定方法对语言模型的参数进行重新估计，比如可以采用贝叶斯参数估计方法[60]、最小区别信息方法(Minimum Discrimination Information)[61-63]等。但对在线自适应来说，要求这个参数重估方法的自适应速度尽可能快，而且需要在检测出自适应不合理时，能恢复到原来的参数。基于这个考虑，我们提出采用动态加权因子的最大后验(*Maximum a Posterior*)概率方法[59].

如果将统计语言模型的历史部分记为 h，且记 $P(w|h)$ 为 λ_{hw}，则统计语言模型的参数集可以表示为

$$\phi = \{\lambda_{hw} \mid w \in W, h \in H\} \tag{3.2.1}$$

其中 W 表示所有可能出现的单词集合，即词表；H 表示所有可能出现的历史序列集合。语言模型自适应的目标就是在观察到新的语料数据 X 后对参数集 ϕ 进行重新估计使之更加准确。*MAP* 方法的目标就是获得使 $P(\phi|X)$

最大的解 ϕ_{MAX} [64]

$$\phi_{MAP} = \arg\max_{\phi} P(X \mid \phi)P(\phi) \tag{3.2.2}$$

上式中 X 是用于自适应的语料数据，$P(\phi)$ 是语言模型参数的先验分布。

很明显，MAP 方法的结果依赖于 $P(\phi)$。如果我们选择 $P(\phi)$ 为均匀分布，即对模型参数的取值没有特别的先验偏好，则 ϕ_{MAP} 退化为模型参数在自适应数据 X 上的最大似然估计。这种结果相当于完全抛弃了原先使用大量通用数据训练的通用模型的参数意义，显然不合理。

在 MAP 中广泛使用的是 *Dirichlet* 分布[65, 66]。

如果有 n 个事件，对每个事件的出现分别有相应的概率估计，假设其中一种概率组合为 $(p_1, p_2, ..., p_n)$，则该组合是这个多维分布的一种情况。而 *Dirichlet* 分布则用于计算该组合出现的概率。

对于 n 个事件一种概率组合取值 $(p_1, p_2, ..., p_n)$，则这个多维分布在该处的概率密度为

$$P(\phi) = Dirichlet(P;U) = \frac{1}{Z(U)} \prod_{i=1}^{n} p_i^{u_i - 1} \tag{3.2.3}$$

其中 U 是对应 n 个事件的属性的一组参数，$Z(U)$ 是为了保证总概率为 1 的归一化因子，这里不需要讨论。当 $p_1, p_2, ..., p_n > 0$，$\sum_{i=1}^{n} p_i = 1$，且 $u_1, u_2, ..., u_n > 0$ 时(记为**条件一**)，u_i 可以解释为事件 i 观察到的次数加 1。

实际上，一个事件发生的确切概率几乎无法获得，只能说该事件的确切概率是在 0 与 1 之间的一个分布，当有先验知识时， 可能该概率在某个值(比如 p)附近的密度大一些。因此，概括的说，*Dirichlet* 分布用于描述“概率的概率”。

可以用二维的情况简单解释如下。

对于二维的情况，假设满足事件e_1、e_2的概率为p_1、p_2，且$p_1+p_2=1$，则当$u_1=u_2=1$，$u_1=u_2=10$，$u_1=u_2=100$，及$u_1=71$且$u_2=31$时，可以定性画出分布情况如图 3-4 所示。由于在 *Dirichlet* 分布中，u_i可以解释为事件i观察到的次数加 1，因此情形 1 对应于e_1、e_2都没有发生过的情况，即只知道e_1、e_2发生的总概率为 1(即肯定有一个会发生)，因此(p_1, p_2)是在直线$p_1+p_2=1$第一象限内的均匀分布。在情形 2 和情形 3 中，已观察到的e_1、e_2发生次数相等，因此(p_1, p_2)在(0.5,0.5)附近概率密度相对较大，且由于情形 3 中的次数比情形 2 中的次数大，因此情形 3 中(p_1, p_2)的密度在(0.5,0.5)附近更加集中；情形 4 对应的是e_1、e_2发生次数不等的情况(分别 70 和 30 次)，则(p_1, p_2)在(0.7,0.3)附近分布比较集中。

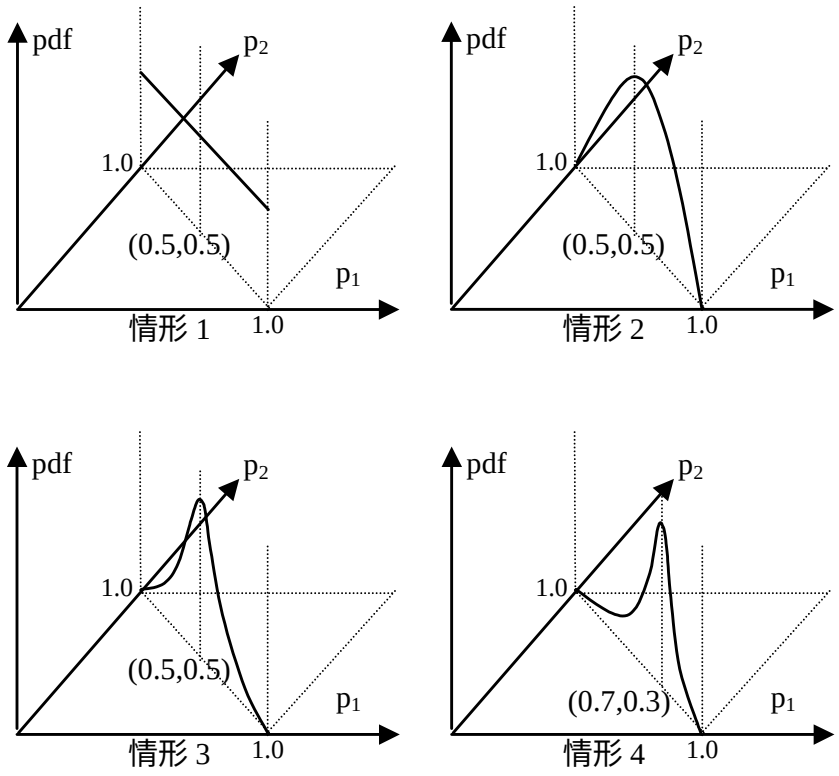

图 3-4 不同情形下的 Dirichlet 分布形状

从图 3-4 可以看出，*Dirichlet* 分布能非常好的刻画多维概率参数的分布概率密度问题。

对于统计语言模型来说，对于给定一个历史 h，满足 $P(w|h)>0$，$\sum_{w\in W}P(w|h)=1$，且 $u_{hw}>0$（u_{hw} 是单元 hw 观察到的次数加 1），因此整个模型概率参数分布相当于可以划分到 $N(h)$ 个（即 h 的个数）满足条件一的 *Dirichlet* 分布，也就是说有

$$P(\phi)=Dirichlet(P;U)=\frac{1}{Z(U)}\prod_{w\in W,h\in H}P_{hw}^{u_{hw}-1} \tag{3.2.4}$$

且

$$u_{hw}=C_{hw}^{(T)}+1\ \left(w\in W,h\in H\right) \tag{3.2.5}$$

其中 $C_{hw}^{(T)}$ 表示单元(h，w)在训练语料中出现的次数。为叙述方便，后文将采用 $hw\equiv(h,w)$。

根据平滑算法，P_{hw} □ $P(w|h)$，采用最大似然估计，即有

$$P_{hw}=\frac{C_{hw}}{\sum_{w\in W}C_{hw}} \tag{3.2.6}$$

当给定一定的自适应语料 X 时，如果单元 hw 在自适应语料种出现了 $C_{hw}^{(A)}$ 次，则有

$$P(X|\phi)=\prod_{h\in H}\prod_{w\in W}P_{hw}^{C_{hw}^{(A)}} \tag{3.2.7}$$

这样， 结合公式(3.2.4)(3.2.5)(3.2.6)(3.2.7)，有

$$\phi_{MAP}=\arg\max_{\phi}\left(\prod_{w\in W}\prod_{h\in H}P_{hw}^{C_{hw}^{(T)}+C_{hw}^{(A)}}\right) \tag{3.2.8}$$

考虑到对于每个具体的 h，有

$$\sum_{w\in W}P_{hw}=1 \tag{3.2.9}$$

于是可以得到 MAP 的结果为

$$P_{hw}^{(MAP)} = \frac{C_{hw}^{(T)} + C_{hw}^{(A)}}{\sum_{w \in W} \left(C_{hw}^{(T)} + C_{hw}^{(A)} \right)} \tag{3.2.10}$$

结果表明：只需要将自适应语料和通用语料一样对待，按最大似然的方式计算概率参数。

3.2.4 基于 MAP 方法的自适应参数更新——实际考虑

如果把以上的结果直接应用于自适应，则在自适应初期，由于自适应语料的不足，造成某些时候其作用如同泥牛入海，淹没在背景语料的汪洋大海中发挥不了修正错误的功能。因此，需要修正这个方法，考虑到自适应是在线完成的，必须满足以下条件：

1)须尽可能多、尽可能快的修改背景模型不合理的地方；

2)为避免自适应对模型造成破坏性损害，自适应须强度适当，不能“过度适应”；

3)随着语料的增加，自适应方法要最终能够收敛到 MAP 方法；

4)为满足在线自适应要求，自适应速度必须能满足在线运行的需要；

假设解码结果串为 $S = w_1 w_2 ... w_n$，用户对其中不正确的地方进行了修改，得到正确的串 $S' = w_1' w_2' ... w_m'$，我们知道，统计语言模型性能一般相当好，即使训练领域和使用领域不相同，一般音转字正确率也能达到 90%，因此实际上被修改的部分并不多。但是，之所以正确的串在打分过程中竞争不过错误串，其原因在于正确串中包含了被修改字词的单元概率过低。如下面的例子：

解码结果：该/地区/的/产/**粮**/米质/等/各项/指标/获/总分/第一

修正结果：该/地区/的/产/**量**/米质/等/各项/指标/获/总分/第一

在这个例子中，只发生了一字错误，而且解码结果的最前部分和最后部分都与正确句子完全相同，因此这两部分包含的单元的概率并不需要修

改，需要修改的是和错误相关的单元的概率。解码的过程可以用图 3-5 表示，横坐标表示音节序列，水平的虚线表示当前步的第一名得分，作为参考。曲线表示解码路径，曲线的起伏表示在解码过程中该路径的得分和排名第一的路径得分之间的差距在变化，曲线越往上表示其得分越接近排名第一的路径得分。路径 1 是到达最后一步时解码排名第一的路径(中间可能不是第一)，因此其到达解码的最后一步时，上升到水平虚线位置。中间两条虚线之间部分表示解码结果和正确结果不符时那些包含了被修改汉字(或者包含该汉字的次)的单元所在区域。路径 2 是**正确结果**在原来解码过程中路径得分起伏情况，由于它被错误的打分(打分过低)，因此到最后时排名在路径 1 之后。路径 2 和路径 1 的差别就在于中间虚线之间部分，因此 2 得分不高的原因就在于其中间虚线之间部分单元的条件概率得分不高(如图所示，得分相对路径 1 来说呈相对下降趋势)。路径 2 的右边部分的形状和路径 1 的对应部分是完全一样的，因为其实际上是相同的单词序列。因此，为保证正确的结果能够出来，我们需要调整中间虚线之间部分的单元得分情况，保证路径 2 排名上浮。因此我们通过修改模型，保证曲线 2 中标注 A 的部分起伏情况发生变化，变成 A'，即路径 2 得分情况呈路径 2’所示。这样即保证了正确结果的出现并最大程度防止模型被过多修改。

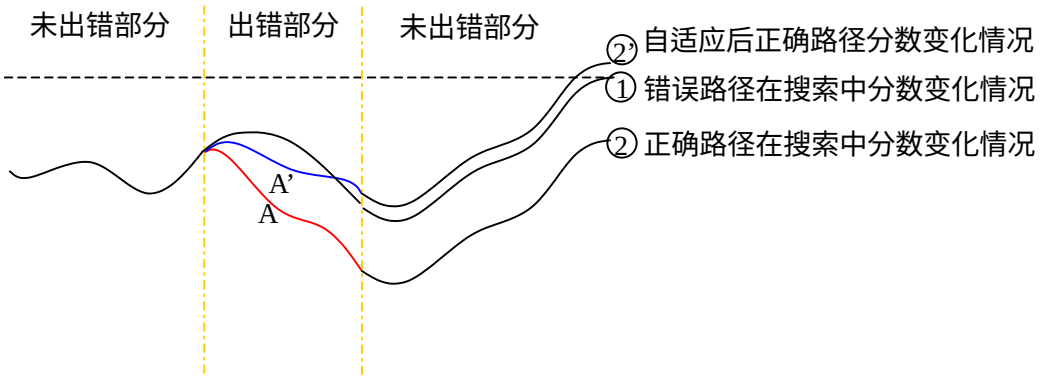

图 3-5 正确路径和错误路径在解码空间中的得分示意图

由于解码结果和修正结果在词边界上可能不完全一致(如这个例子一

样，“产粮”不是词表中词汇(系统词)，而“产量”是，因此，将“粮”字修改成“量”后，应与前面的“产”字合并成单词“产量”)，即正确的词边界为

修正结果：该/地区/的/**产量**/米质/等/各项/指标/获/总分/第一

合并过程采用从被修改字出发，向两边尝试合并的过程。经过这个处理后，概率修改就只涉及包含被修改字词的单元，比如这里只涉及和单词“产量”相关的单元。这种只改变相关单元概率的思想，从哲学上说符合“奥卡姆剃刀原理”[67]，即“无需必要，勿增实体”(Entities should not be multiplied unnecessarily)，在两种方法都能达到同样目的时候，选择简单的那种是明智的。

为了满足模型自适应速度尽可能快的需求(正确的句子 S' 能成为解码的结果)，我们将 MAP 方法修改成采用动态加权因子的 MAP 方法，即

$$P(w|h)=\frac{C_{hw}^{(T)}+\alpha*C_{hw}^{(A)}}{\sum_{w\in W}\left(C_{hw}^{(T)}+\alpha*C_{hw}^{(A)}\right)} \tag{3.2.11}$$

不同的句子发生错误时，其中加权的系数 α 可能并不相同。显然，当 α 增大时，如果输入同一个拼音串，则正确的句子出现的可能性越大。假设 α_{good} 是保证输入该拼音串时，该句子能被正确转换的阈值(可以通过尝试方法快速求出)，则 α 选择如下式：

$$\alpha=\begin{cases}\alpha_{\min}, & 如\alpha_{good}<\alpha_{\min}\\ \alpha_{good}, & 如\alpha_{\min}\le\alpha_{good}\le\alpha_{\max}\\ \alpha_{\max}, & 如\alpha_{\max}<\alpha_{good}\end{cases} \tag{3.2.12}$$

其中 $\alpha_{\min}$ 和 $\alpha_{\max}$ 是系数 α 取值的下限和上限，通常 $\alpha_{\min}$ 取 1，也就是说，即使 α_{good} 小于 1，比如说 0.5(这种情况完全可能)，则 α 取值也不能小于 1；而 $\alpha_{\max}$ 则需要根据训练背景模型的的语料大小来选择合适的值。为了和背景模型匹配的好，可以选择背景模型所有单元平均次数的若干倍数。

3.3 在线递增式自适应方法中的一些问题及其解决方法

在具体实现时，在线递增式语言模型建模方法必须解决很多问题，比如模型振荡、概率归一化、以及新词发现等问题。

3.3.1 模型振荡问题

在训练数据充分的情况下，统计语言模型的性能相当出色，对于音转字，正确率通常能到 99%左右。但是如果训练数据不充分或者当前应用和训练语料不匹配，则性能下降很快，通常只能到 90%甚至更低。对于拼音转汉字的这种应用，由于模型和应用领域不匹配造成的错误大约占(99%-90%)/(100%-90%)，即 90%左右；而由于语言本身的混淆以及 n-gram 建模方法本身的先天缺陷造成的错误大约只占 10%。因此，当我们将一个通用模型应用到一个具体领域，并且一开始就进行自适应。此时每发生一个解码错误，将有 90%的可能是由于模型不匹配造成的，即自适应有 90%的可能性是正确的；由于本文的自适应方法采用了加权因子，在 90%的情况下自适应能够一步到位，大大加快了模型自适应速度。但是，也有 10%的可能性是由于语言本身以及建模方法本身的先天不足造成错误。

当解码发生错误时，实际上无法判断该错误的发生是由于模型和领域不匹配还是由于该领域语言本身存在歧义(对于语音识别来说，即一个发音序列对应多个合法的文字文本序列)。这种情况在输入串较短时，比如单字，显而易见会发生，而且这种情况下如果采用“调整到第一名”的策略显然会导致模型“振荡”，即参与竞争的候选序列的得分排名将总在忽前忽后的不断变化，会引起模型的不稳定。因此，对于单字和单词等片断，α值应该设为 1，保证其真实反应情况。对于较长的句子中间的部分，一般来说自适应正确的可能性很大，但也可能发生振荡，为了防止出现这种现象，需要设置惩罚因子进行控制。

为了解决这个问题，用户模型包含了自适应单元在自适应语料中出现的实际次数C_{real}以及该单元发生振荡的次数C_{sway}。

C_{sway}越大，则需要降低自适应语料的权重。因此公式(3.2.13)中的α需

要做处理，需要调整成如下公式：

$$\alpha' = \alpha * e^{-\frac{C_{sway}}{\alpha}} \tag{3.3.1}$$

并加上$\alpha' \geq 1$的限制条件。其中$e^{-\frac{C_{sway}}{\alpha}}$部分本文定义为自适应的**抑制因子**。抑制因子是为了防止模型参数修改过大而设置的。这样，当发现某些单元的概率调整发生错误时，通过抑制因子降低其动态加权系数的大小，直到动态加权系数为 1。因此，当自适应语料增加到足够充分时(和训练语料同数量级)，该自适应方法将收敛到传统的 MAP。这恰恰也是我们希望的结果。

3.3.2 概率归一化问题

在前面的框架中，我们没有对自适应后的模型进行归一化，原因在于：

1)为了满足在线自适应的要求，自适应速度必须足够快，过于频繁的模型归一化造成大量的计算；

2)当进行模型归一化时，要保证模型的可靠性。从前面的分析可知，一些自适应得到的模型单元或者参数并一定非常可靠，可能存在振荡等问题，因此刚刚通过自适应语料更新了一个单元的参数后，并不适合马上将模型归一化。

基于以上两点考虑，采取批量归一化的方式。即在自适应单元参数积累到一定程度时才进行归一化，而且自适应单元的权重也调整到合适的值(通过α加权和加权抑制因子的双重作用)，保证模型在归一化后比较准确。

3.3.3 新词发现问题

汉语区别于英语的一大特点是汉语文本没有词边界，而且汉语中词和词组之间的界线也是模糊的，有些多字组合可以当成词，也可以当成词组，比如“尊师重教”，可以认为它是一个单词，也可以认为是由“尊师”和“重教”两个单词构成的词组。包含长词的句子候选在语言模型打分中占

有优势。因此，在语言模型在线自适应过程中，可能在某些时候需要将一些单字组合成新词，即所谓新词发现。

自然语言处理中有一些方法用于描述单字和词的关系，如字的统计构词能力 (Word Formation Power, WFP)[68]、汉字的构成模式等。但实际上这些指标的计算都先要依赖于词表本身，比如一种描述汉字 c 的构词能力的指标为

$$WFP(c)=\frac{\text{含c的多字词个数}}{\text{c的总个数}} \tag{3.3.2}$$

只有文本已经经过分词处理了，才有可能计算出 *WFP* 来，也就是先有词表后有 *WFP* 指标，因此，用初始词表产生的 *WFP* 指标是无法用于对新语料中词进行预测的。比如，原始词表中如果没有“的确”这个单词，由于“的”字属于活跃单字，则采用 WFP 方法永远都无法预测出“的确”是个单词。WFP 更多的是一种后验的描述方法，很难用于预测。

可用的做法只能是依靠相连单字之间的相互预测概率来确定他们是否已经紧密到足够的程度。比如可以借用信息论中的“互信息”定量描述任意两个汉字之间的结合力[69]。也有的研究者引入经典统计论中的“四分联立表”及检验联立表独立性的皮尔逊 χ^2 统计量[70]，对长度分别为 2 字、3 字和 4 字的任意汉字串做内部关联性分析。

总的来说，本文的重点在于提出一种修改参数的在线递增式自适应框架，新词发现的问题不是本文讨论的重点，可以使用一些当前比较好的算法解决这个问题。本文只简单的使用临近字相互条件概率的大小作为是否需要合并的依据。

3.4 实验和分析

为了测试我们的语言模型自适应方法的性能，我们设计了相应的测试实验。

通用模型训练所用语料大小为 200M 文本(字数)，该语料库来源于 93

－94、96－97 四年的《人民日报》，92、94 年的《经济日报》，94 年的《市场报》和 94－96 三年的《新华社》文稿，共计约 2 亿字。

测试语料包括三个。语料 *A* 是关于美国总统大选的新闻资料，语料 *B* 是关于中国军事演习的一组新闻资料，语料 *C* 则是一些新闻，没有固定的主题。

测试的方法是采用音字转换的方法。每个语料，先转换成拼音序列，然后分别使用不加自适应功能的语言模型和带自适应功能的语言模型指导解码。解码结果和原始文本进行比较以获得解码的正确率。见表 3-2。

表 3-2 在线自实验性能测试结果

语料	汉字数	不带自适应解码 汉字错误率	带自适应解码 汉字错误率	错误率 下降率
A	4,424	7.64%	6.19%	19.0%
B	3,602	9.08%	6.41%	29.4%
C	3,103	9.64%	9.60%	0.4%

从表中可以看到，语料 *C* 由于没有固定的主题，其前后并没有非常相关的信息，自适应所得到信息并未对后面的解码产生作用，因此其性能并没有得到提高(而且理论上讲，由于前面提到的模型参数振荡等现象的存在，对于语料 *C*，有可能还会出现性能略微下降的情况)。相反，语料 *A* 和语料 *B* 相对有较为固定的主题，因此其前面所得信息对后面产生了正面的指导作用，其性能明显得到提高。错误率下降达 20%左右。

3.5 小结

本章通过对传统语言模型自适应方法原理的分析，指出传统方法在应用方面存在三个方面的不足：一是自适应过程是预先处理好的，或者只使用预先处理好的语料；二是在用户当前话题不固定时不能发挥作用；三是在应用过程中得到的语料没有充分应用起来。针对这三点不足，我们提出了一种在线递增式语言模型自适应方法。在原理上，该方法对自适应的处

理是在线进行且递增式的；在语料使用上，该方法能够不断从用户应用产生的语料中学习有用的信息；在作用上，由于该方法能够将学习到的信息持续保存，因此对于话题多变的应用场合，只要其当前文本和以前文本(可以是很久之前的)有关系，也能保证其从自适应中得到好处。

本文采用了一种改进的 MAP 算法用于在线自适应的参数更新。采用动态加权因子的方式对新语料加权，从而满足模型的快速适应需求。针对语言模型在线自适应中可能出现的振荡问题，我们提出了一种抑制振荡的方法，也就是在发现加权因子过大时采用抑制因子，保证正确的参数修改得以保留，错误的参数修改很快就能得到修正。抑制因子的提出，还保证了随着自适应过程的进行，自适应语料的权重慢慢降为 1，保证真实反应自适应领域的语言连接关系。

总之，本章提出的在线递增式自适应方法，很好的解决了语言模型使用过程中由于领域、地域、用户风格等造成的语言模型性能下降问题。该框架特别适用于整句音字解码等实际应用。实验结果证明，使用该方法能显著降低拼音转汉字的错误率。

第四章　统计语言模型压缩方法研究

语音识别、OCR 等应用都需要使用语言模型。trigram 统计语言模型规模非常大。为了保证语言模型的性能，一般都使用几百 MB 甚至更多的语料训练模型，训练后模型的大小会和训练语料相当。因此一个可用语言模型大小一般达上百 MB 甚至更大。对于存储能力飞速发展的桌面计算机来说，这个规模并不算大，但对于手持设备，比如手机、掌上电脑、PDA 等设备来说，这个规模无法应用。随着近些年嵌入式设备的普及和发展，这类应用在嵌入式设备上的需求越来越大。需求和实现之间的矛盾，导致这类应用系统无法放到嵌入式设备上去。为了解决这个问题，需要多方面努力。对于语音识别来说，需要采用更合适的声学模型并将语言模型压缩到可用的规模。这样，如何尽可能的压缩模型大小并保持模型的准确性成为目前面临的一个难题。

本章就解决如何在保持语言模型可用性的前提下，将语言模型压缩到可在嵌入式设备上使用的规模作深入的研究。

本章的内容安排如下：第 1 小节介绍语言模型压缩的应用背景；第 2 小节分析传统的语言模型压缩算法及其性能，并提出了本文的压缩算法，即基于单元条件概率和排名的压缩算法；第 3 小节是实验和分析，验证本文方法的有效性；第 6 小节是本章小结。

4.1 语言模型压缩的应用背景

4.1.1 嵌入式设备对应用程序的需求快速增长

手持设备是目前增长非常迅速的消费电子类别。据国际数据公司(IDC)数据显示，近几年我国智能手机(Smart Phones)市场的增速极为迅猛，平均年增长率高达 220%。据统计，2000 年，国内智能手机年销量为 14 万部，2001 年为 31 万部，而 2005 年的市场规模预计将达到 3,500 万部[71]。全球智能手机市场也处于高速增长时期，2003 年第 1 季度全球智能手机出货量大约是 170 万部。市场研究公司 ARC Group2003 年九月份报告则指出，智

能手机 2002 年出货约 350 万部，占整体手机市场比重还不到 1%，但预期到 2007 年将大幅成长至 4,500 万部，市场比重增至 5%。

以基于 BREW 技术的智能手机为例(BREW 即 Binary Runtime Environment for Wireless，是著名的 CDMA 芯片产商高通公司开发的手机平台，采用 BREW 平台的手机都能运行 BREW 技术开发的应用程序)，高通(QualComm)公司给出了 2001 年到 2007 年基于该平台的手机增长趋势图[72]，从图 4-1 我们可以看到最近和未来几年市场增长非常迅速，特别是以中国为代表的亚太地区的增长更是引人注目。

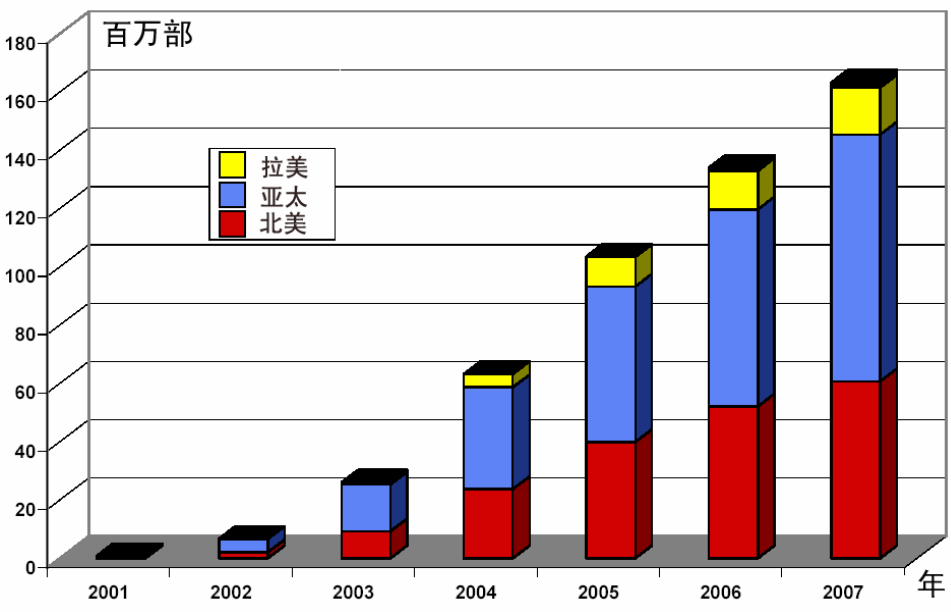

图 4-1 嵌入式设备消费增长幅度示意图

随着手持设备越来越普及，许多桌面计算机上的应用系统需要能够在手持设备上运行。图 4-2 是高通(QualComm)公司给出的从 2002 年下半年到 2003 年上半年其 BREW 平台上应用程序下载次数的增长情况[72]。在不到一年时间内，应用程序的下载次数呈了爆炸式增长。

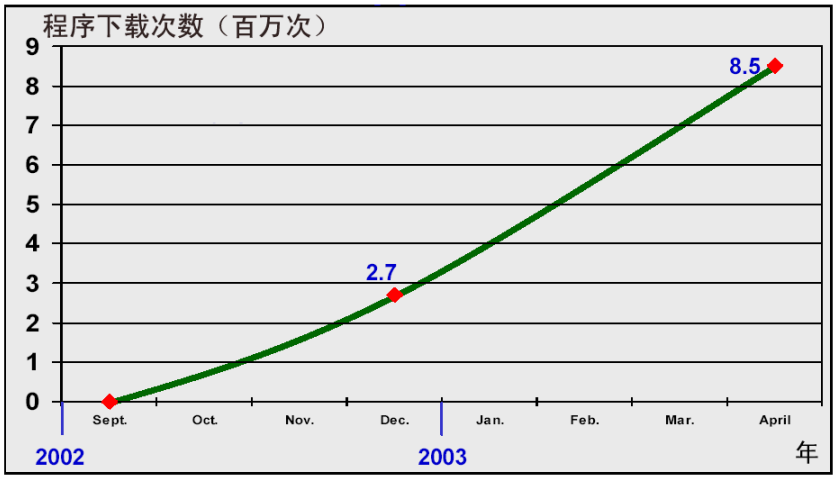

图 4-2 嵌入式设备上应用程序需求增长示意图

这种对手机等嵌入式设备上应用程序的需求直接推动了此类应用程序的开发，也直接推动原先只能在桌面计算机上运行的应用程序从 *PC* 到嵌入式设备的移植工作。

4.1.2 嵌入式设备的计算资源存在不足

嵌入式设备(如智能手机)普遍的特点是存储能力较小，计算能力相对桌面计算机来说较差。表 4-1 给出了目前主流的桌面计算机和和手持设备的配置比较。从表中可以看出，手持设备的 CPU 计算能力和动态内存只有桌面计算机的 1/10 左右，而其静态存储能力则比桌面计算机相差 1,000 倍左右。

表 4-1 桌面计算机和嵌入式设备计算能力比较

	CPU 主频(Hz)	静态存储能力(MB)	动态内存(MB)
桌面计算机	2.5G	80,000	512
嵌入式设备	<400M	32	8

使用语言模型的一些应用，比如语音识别、中文整句输入法、手写整句输入法等，是作为人和计算机交流的人性化界面应用程序的重要组成部分，是移植的重点，但是同时也是难点，因为正如前文提到，高性能的语言模型可能达到几百 MB 的规模，对于目前的手持设备来说，仍然是不可

逾越的鸿沟。而且即使手机的存储能力已经达到了几百 MB 的能力，将全部存储用于一个应用程序也是不现实的。基于以上的原因，迫切需要在保持语言模型性能的前提下，将其大小压缩到手持设备可以承受的规模。

4.2 基于单元条件概率和排名的语言模型压缩算法

4.2.1 常规语言模型压缩算法

一些文献[73-76]介绍了常用的语言模型压缩方法，包括次数剪切(Count cutoff)、规则剪枝(Pruning)和聚类(Clustering)等。

次数剪切(Count cutoff)方法是简单地抛弃次数低于某个阈值 k 的单元，通常 k 是一个预先定义好的值[74, 77]。

规则剪枝则采用较为复杂的规则决定哪些单元次重要，可以抛弃。比如，加权差异方法(Weighted Difference Method)[78]中，trigram 单元和对应的 bigram 单元，或者 bigram 单元和对应的 unigram 单元之间的差异被用做是否保留高阶单元的标准。比如，如果三阶单元概率 $P(w_3 \mid w_1, w_2)$ 和其相应二阶单元概率 $P(w_3 \mid w_2)$ 之间的差异很小的话，就可以认为保留三阶单元是没有必要的，因为丢失它不会对模型的精度造成很大影响。稍微复杂一些的方法是使用以下的公式

$$\left[C(w_1, w_2, w_3) - D(C(w_1, w_2, w_3))\right] * \left[\log P(w_3 \mid w_1, w_2) - \log P(w_3 \mid w_2)\right] \tag{4.2.1}$$

来决定该三阶单元是否应该保留。有的采用更严格的数学规则[75]，用熵理论指导剪枝。具体如下，如果单元 (w_1, w_2, w_3) 从模型中删除，则模型增加的熵为

$$-P(w_1, w_2, w_3)[\log P'(w_3 \mid w_1, w_2) - \log P(w_3 \mid w_1, w_2)] \tag{4.2.2}$$

其中 $P'(w_3 \mid w_1, w_2)$ 表示删除单元 (w_1, w_2, w_3) 后的模型对该被删单元的概率估计。

也有的研究者采用词类方法建模，并同时希望能够改善模型的精度和

鲁棒性[17, 76, 79]，以 bigram 为例，该方法可以简单的用以下公式表示

$$P(W,C)=P(c_1)P(w_1|c_1)\prod_{t=2}^{T}P(c_t|c_{t-1})P(w_t|c_t) \tag{4.2.3}$$

其中 c_t 表示单词 w_t 对应的词类。C 是和词序列 W 对应的词类序列。$P(c)$ 是词类的 unigram 概率，而 $P(w|c)$ 则是单词 w 在词类 c 中的所占的概率比例。 $P(c_t|c_{t-1})$ 是词类之间的转移概率。因为词类的个数应该远远少于词条数目，因此对应的 bigram 模型的单元数目会减少，即模型 $\{P(c_i|c_j),P(w_k|c_i)|i,j,k\}$ 会比较小。词类数目不好选择，而且这种一个词对应一个确定的词类的方法性能也不够好。

对那些重要的单元来说，可能其概率一点点的偏差就会造成模型整体性能的下降；但对某些不那么重要的单元而言，可能其概率只需要定性区分就可以。

4.2.2 单元重要性的各种指标

对于一个 n-gram 单元 (w_1,w_2) 来说，如何衡量其重要程度呢？总结以前的研究文献，常见的有如下的方法：其条件概率值、出现次数、压缩前后模型单元的熵差等。下面一一介绍。本文中使用 $IMP(w_1,w_2)$ 表示 bigram 单元 (w_1,w_2) 的重要性。

(A)条件概率值

对于 bigram 单元来说，条件概率值即

$$IMP(w_1,w_2)=\frac{c(w_1,w_2)}{c(w_1)} \tag{4.2.4}$$

这个假设认为条件概率越大的，其对模型的性能影响也越大。该假设有一定的合理性：因为统计语言模型是个系数异常稀疏的大矩阵，大部分单元都在训练语料中观察不到，这些单元将被赋予一个较小的值，而在训练语料中观察到的单元，则将被赋予较大的值。

因此，可以认为，单元的条件概率越大，则在历史词确定的情况下，

对下一个单词的预测越大，对模型的重要性也越大。

(B)出现次数

如果我们将统计语言模型的词表用V表示，所有的单元组成的集合看成是一个事件的空间E，即

$$E=\{(w_1,w_2)\mid w_1,w_2\in V\} \tag{4.2.5}$$

则单一事件(w_1,w_2)的重要性可以用该事件在空间E中发生的概率来表示，即

$$IMP(w_1,w_2)=\frac{c(w_1,w_2)}{\sum_{w_1,w_2\in V}c(w_1,w_2)} \tag{4.2.6}$$

即某一事件的重要性为该事件出现的次数在整个事件空间中所占比例。在事件空间中所占比例越大的，其重要性越大。

(C) 压缩前后模型的熵差

以上两种定义没有考虑统计语言模型压缩前后的回退特性，即如果某单元没有在训练语料找到，则其概率值将用其对应的低阶单元来平滑。因此，如果我们在定义好单元的重要性，并依据该定义对模型进行剪枝压缩，则压缩后更多模型单元概率的计算需要依靠低阶单元的概率来恢复。因此，如果在定义单元的重要性时就考虑模型的回退性能，可能会更好的指导模型的剪枝。

有一种方法是按如下方式定义模型单元重要性：

$$IMP(w_2,w_1)=\left|\log P(w_2\mid w_1)-\log P'(w_2\mid w_1)\right| \tag{4.2.7}$$

在上式中，$P(w_2\mid w_1)$是原始模型中单元(w_1,w_2)的概率，$P'(w_2\mid w_1)$则是压缩后模型中单元(w_1,w_2)的概率。$P'(w_2\mid w_1)$的计算方法可能多种多样，和具体的压缩方法有关。对于最简单的模型压缩方法——即单元剪枝方法——而言，$P'(w_2\mid w_1)$可以用低阶单元的概率值表示。该方法的合理性在于

考虑了压缩前后单元概率分值的变化。目标是使得所有的单元在压缩前后的概率歧变控制在一定范围内，因此从直观上有其合理性。但是实际实验发现，该方法效果并不好，甚至比前面两种方法差。实际上模型单元的分值差异应该累计，也就是考虑由于该单元的剪枝累计给整个模型造成的熵差。即对不同的单元，即使他们压缩前后的概率分值差是一样的，但由于其出现次数的不同，应该考虑给予不同的重要性度量。其中一种方法是按如下公式：

$$IMP(w_1,w_2)=c(w_1,w_2)*\left|\log P(w_2\mid w_1)-\log P'(w_2\mid w_1)\right| \tag{4.2.8}$$

其中$c(w_1,w_2)$是单元(w_1,w_2)的出现次数，和$P(w_1,w_2)$只相差一个常数。因此公式(4.2.8)实际上描述的是剪枝前后模型熵的变化。这个方法要求那些出现次数很多的单元，在压缩时要保留更大的精度。

4.2.3 各个方法的比较和分析

我们选择了三个测试语料用于评测不同方法的性能。初始模型(种子模型)即采用第二章改进的 Katz 方法训练的模型，所用语料为 200M 汉字。测试语料 1 是一篇政治文章，共 35,025 字；测试语料 2 来自 863 语料，共 1,801 个句子，23,310 字；测试语料 3 是几篇新闻报道，共 4,178 字。本文随机选用这三个语料做拼音转汉字的测试。词表大小为 51,007，采用 bigram 建模。三个语料音字转换的平均错误率如表 4-2 所示。

表 4-2 三种传统剪枝方法处理后模型大小和音转字错误率关系

平均错误率(%) / 模型相对大小	方法 A	方法 B	方法 C
1.0	8.91	7.92	7.32
1.6	8.27	7.64	6.84
3.8	7.63	6.69	6.25
7.5	6.85	5.98	5.53
21.0	4.85	4.63	4.51
43.0(未压缩)	3.64	3.64	3.64

如果将模型相对大小和错误率的关系用曲线表示出来，可以得到如图 4-3 的关系图。

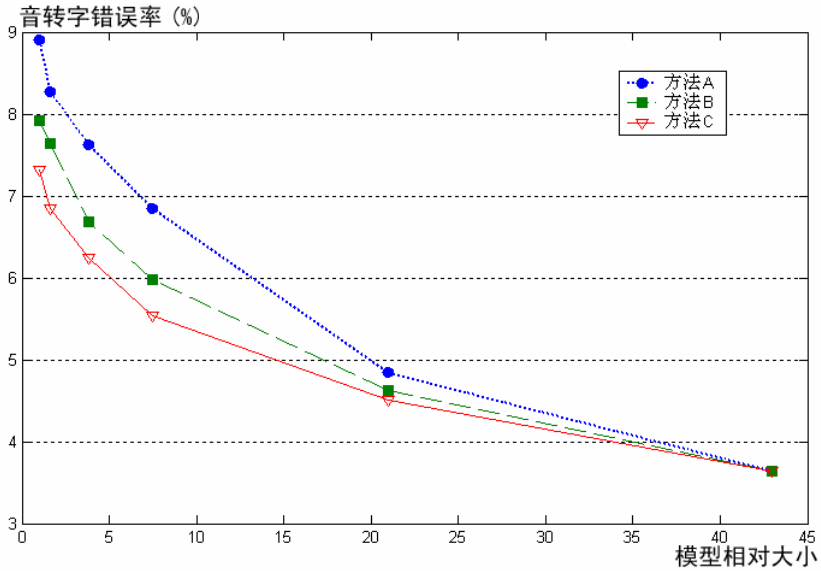

图 4-3 三种传统剪枝方法处理后模型大小和音转字错误率关系曲线

从图 4-3 可以看出，基于单元概率的剪枝性能最差，而基于 $IMP(w_i,w_j)=c(w_i,w_j)*\left|\log P(w_j\mid w_i)-\log P'(w_j\mid w_i)\right|$ 的剪枝方法性能最好。分析其原因如下：

根据语言模型熵率的定义，

$$H(X)=-\lim_{n\to\infty}\frac{1}{n}\log P(x_1,x_2,\cdots,x_n) \tag{4.2.9}$$

其中 x_i 表示在语料流中处于位置 i 的单词编号。如果我们使用 bigram 模型，则上式等价于

$$H(X)=-\lim_{n\to\infty}\frac{1}{n}\left(\log\left(P\left(x_1\right)\right)+\sum_{i=2}^{n}\left(\log\left(P\left(x_i\mid x_{i-1}\right)\right)\right)\right) \tag{4.2.10}$$

如果我们将上式应用到训练数据中，假设训练数据流总长度为 N，则可得

$$H(X)=-\frac{1}{N}\left(\log\left(P\left(x_1\right)\right)+\sum_{i=2}^{N}\left(\log\left(P\left(x_{i-1}\mid x_i\right)\right)\right)\right) \tag{4.2.11}$$

因为对于训练语料流而言，单元$\left(x_{i-1},x_i\right)$是有重复的(即某些 bigram 单元出现多次)，如果我们根据 bigram 单元重新改写上式，可得

$$H(X)=-\frac{1}{N}\left(\log\left(P\left(w_{x_1}\right)\right)+\sum_{i=1}^{V_N}\sum_{j=1}^{V_N}c\left(w_i,w_j\right)*\log\left(P\left(w_j\mid w_i\right)\right)\right) \tag{4.2.12}$$

其中w_{x_1}表示在训练语料中排在第一个位置的单词编号，N_V是词表大小，而且有

$$\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}count\left(w_i,w_j\right)=N-1 \tag{4.2.13}$$

下面我们证明，熵差异方法保证了模型剪枝到预设的大小时，对训练语料流进行打分的分数值变化总和最小。证明如下：

剪枝的过程相当于把训练语料中观察到的 bigram 单元分成两组，一组保留，一组剪掉。假设训练语料中总共出现了N_{Bi-T}个不同的 bigram 单元(每个单元可能出现多次)，根据剪枝的规模，需要保留N_{Bi-R}个，则剪掉的个数为

$$N_{Bi-Pr}=N_{Bi-T}-N_{Bi-R} \tag{4.2.14}$$

将所有的单元按照$c\left(w_1,w_2\right)*\left|\log P\left(w_2\mid w_1\right)-\log P^{'}\left(w_2\mid w_1\right)\right|$的大小，从大到小排序，并从 1 开始编号，第$i$个记为$\left(w_{i1},w_{i2}\right)$，则如果剪掉第$i$个单元，因此的熵率变化为

$$\Delta H(X)\mid_{\left(w_{i1},w_{i2}\right)}=\frac{1}{N-1}\left(c\left(w_{i1},w_{i2}\right)*\left|\log\left(P\left(w_{i2}\mid w_{i1}\right)\right)-\log\left(P\left(w_{i2}\right)\right)\right|\right) \tag{4.2.15}$$

由于采用回退模式，即$P^{'}\left(w_2\mid w_1\right)=P\left(w_2\right)$，该式和我们的剪枝参数$c\left(w_1,w_2\right)*\left|\log P\left(w_2\mid w_1\right)-\log P^{'}\left(w_2\mid w_1\right)\right|$只相差一个常数因子。可见，该方法在剪枝时，保证了保留那些将引起较大熵率变化的单元(编号较小)，而丢

弃那些引起较小熵率变化的单元(编号较大)。

整个剪枝引起的熵率的变化为

$$\Delta H(X)|_{Prune} = \frac{1}{N-1}\sum_{i=N_{Bi-R}+1}^{N_{Bi-T}}\left(c(w_{i1},w_{i2})*\left|\log P(w_{i2}\mid w_{i1})-\log P(w_{i2})\right|\right) \tag{4.2.16}$$

保留下来的单元如果被剪掉，则引起的熵率变化为

$$\Delta H(X)|_{Reserved} = \frac{1}{N-1}\sum_{i=1}^{N_{Bi-R}}\left(c(w_{i1},w_{i2})*\left|\log P(w_{i2}\mid w_{i1})-\log P(w_{i2})\right|\right) \tag{4.2.17}$$

以上两者之和即为 bigram 和 unigram 之间的熵率差。

$$\Delta H(X)|_{Bi-Uni} = \frac{1}{N-1}\sum_{i=1}^{N_{Bi-T}}\left(c(w_{i1},w_{i2})*\left|\log P(w_{i2}\mid w_{i1})-\log P(w_{i2})\right|\right) \tag{4.2.18}$$

我们可以将该剪枝方法直观的用图 4-4 表示。图中粗实线表示用种子模型对训练语料流进行打分时概率分值变化情况，细虚线表示剪枝后模型对训练语料流进行打分时概率分值变化情况，则该剪枝方法保证了两条曲线间所夹面积最小。

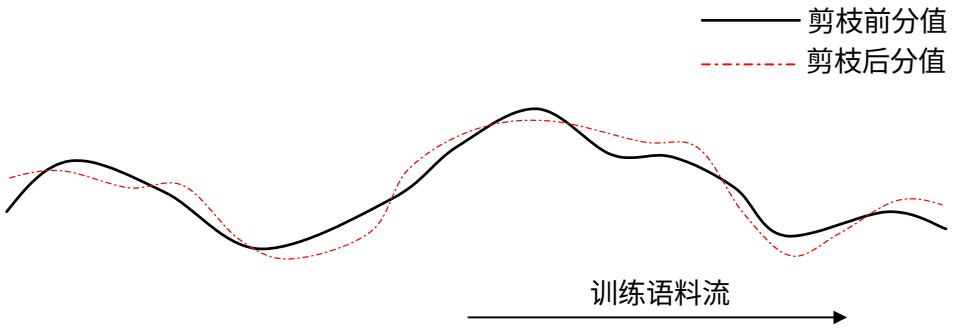

图 4-4 基于模型熵差异的剪枝原理示意图

4.2.4 本文的改进方法

我们发现，传统的剪枝方法对单元的重要性评估各不相同，保留下来的单元特性的分布也各具特色。而这种特点直接决定经该种方法剪枝后的保留下来的单元是否还有进一步压缩的可能。

我们发现，经熵差异方法剪枝后保留下来的单元概率分布比较杂乱，图 4-5 显示了熵剪枝方法后保留下来的单元在各个概率分数值上的分布情况。从图中我们可以看出，保留下来的单元概率分数基本上分布在整个分数允许范围内，

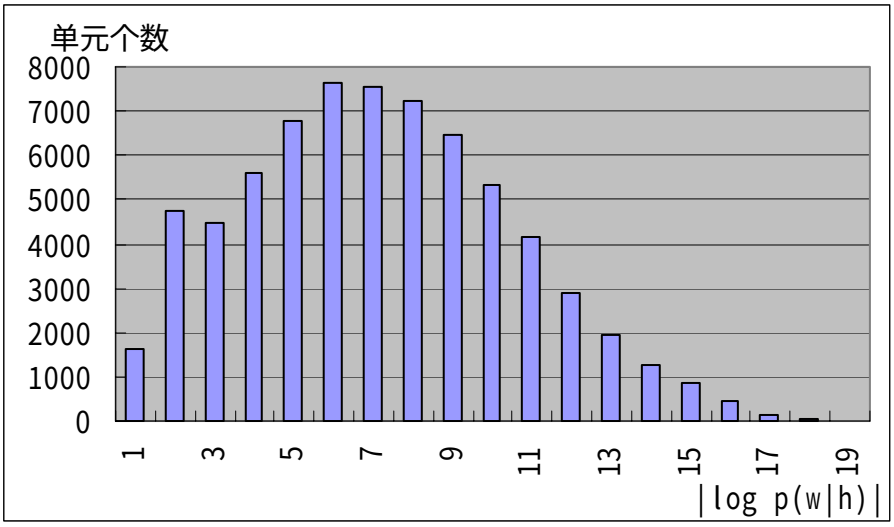

图 4-5 基于模型熵差异的剪枝保留单元概率分数分布图

从图 4-5 可以看出，保留下来的单元分布在允许的整个数值范围，实际上该分布图和模型剪枝前的单元概率分布图相似，大部分单元概率 log 分数都集中在中间。

图 4-6 给出了熵差异方法被剪单元在其排名列表中的分布。从图可以看出，对于给定单元 (h,w)，不管其在 h 对应的所有单元中排名第几，都有可能被剪掉，同样，也都可以被保留下来。这样使得保留下来的单元相对比较杂乱，不利于进一步的压缩。

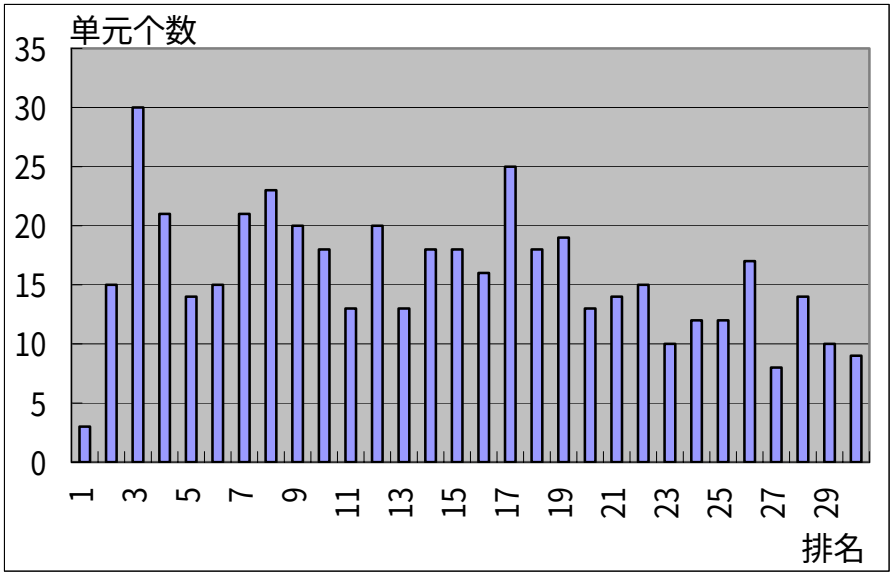

图 4-6 基于熵差异的剪枝方法被剪单元的排名分布图

而基于条件概率的剪枝方法则恰恰相反，其保留的单元相对来说比较整齐划一。因为每一个被保留下来的单元 (h,w)，都是该单元历史 h 对应的最重要的单元。图 4-7 给出的剪枝后保留单元的排名分布也表明了其整齐划一的特点。

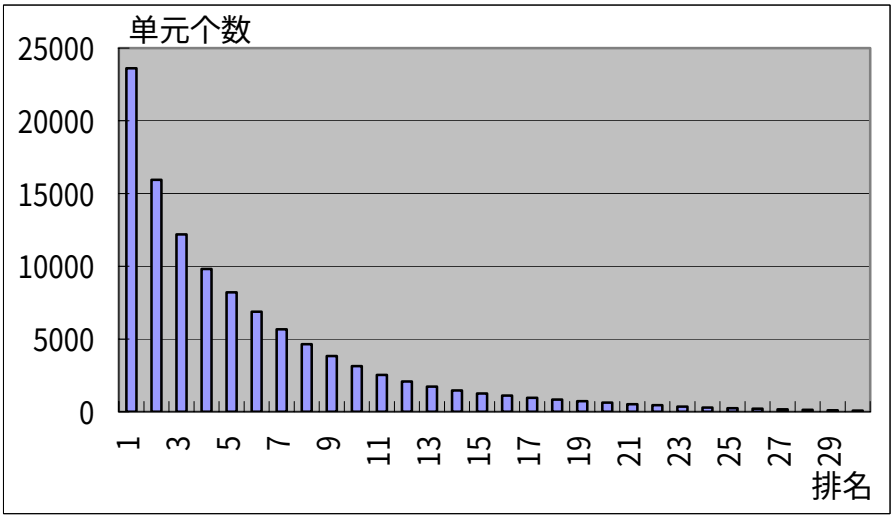

图 4-7 基于条件概率剪枝方法保留单元的排名分布图

由于经过概率剪枝后保留下来的单元分布非常规律，因此可以利用这一特性对保留下来的单元再做进一步的压缩。本文提出了一种基于排名的单元压缩方法，即不需要保留单元的次数信息，而只需要保留单元的排名信息(可以通过存储位置反应排名信息)，利用其排名的位置恢复其概率信息。

首先定义单元或者单元历史的**兄弟系数**。

对于给定历史 h，保留下来的单元个数并不相同。对于给定历史 h，如果保留下来以其为历史的单元个数不为零，则记为 $S(h)$ 个，并将 $S(h)$ 称为相关单元和历史 h 的**兄弟系数**。即由历史 h 开头的单元共有 $S(h)$ 个。图 4-8 给出了 $S(h)$=1、2、3 时排名第一的单元在分数 $|\log(w|h)|$ 上的分布情况。可以看出，$s(h)$ 越小，该分布越靠近 0，而且给定 $S(h)$ 时，分布具有良好的内敛性，即兄弟系数相同且排名位置相同的那些单元概率很接近。

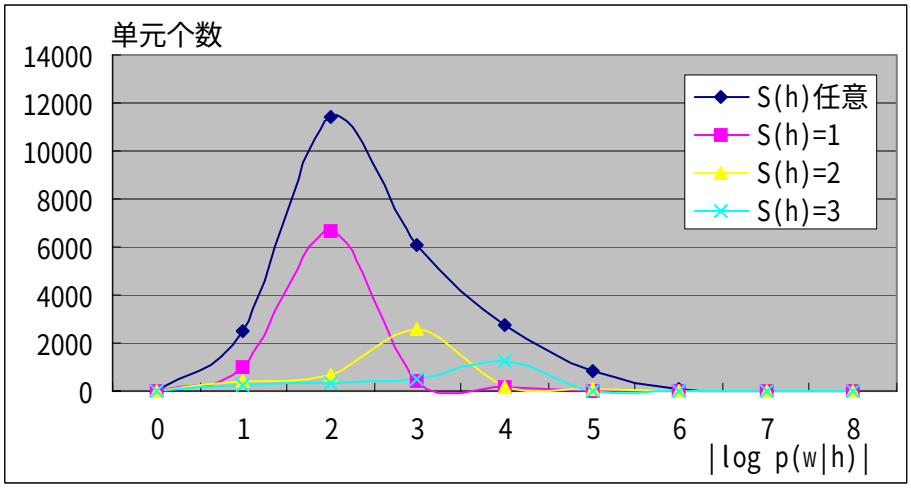

图 4-8 $S(h)$不同时排名第一的单元个数在概率分数$|\log P(w|h)|$上的分布情况

因此，规定兄弟系数相同且排名位置相同的单元属于同一类。且对于 $S(h)=s$ 且排名位置为 i 的单元，其概率统一为

$$P_{s,i}=\sum_{S(h)=s}P_i(\square h)/\sum_{S(h)=s}1 \qquad (4.2.19)$$

其中 $P_i(\square|h)$ 表示剪枝后以 h 为历史的单元列表中排名位置为 i 的单元其在剪枝前概率。

所以，公式(4.3.16)的意义实际上就是对于属于同一类单元，用该类概率的平均值来恢复概率。图 4-9 给出了整个概率求解过程。

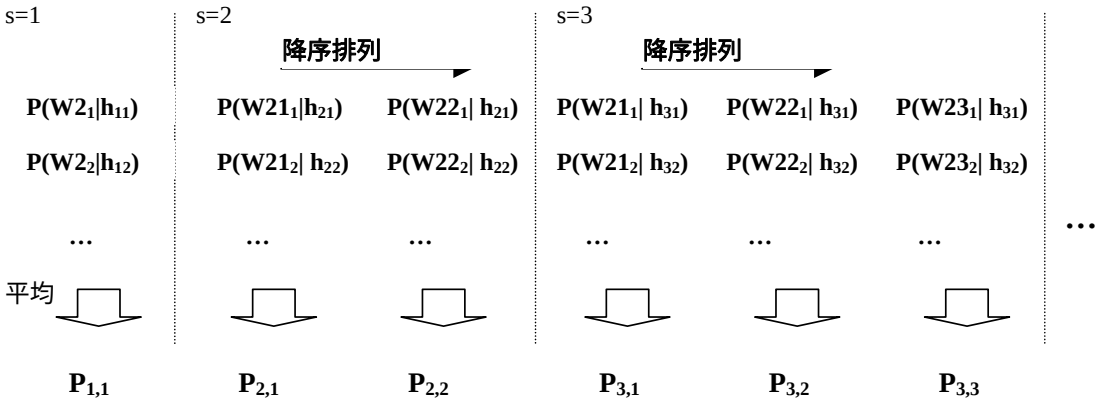

图 4-9 概率 $P_{s,i}$ 的求解过程

使用这种算法，在基本保证性能的情况下，将语言模型大小压缩到 1M 以下(具体的实验结果参考第 4.3 小节)，为语言模型在嵌入式设备上的应用提供了可能。

基于排名的压缩方法是用于高阶单元的压缩，由于 unigram 本身较小，不需要很复杂的压缩方法，可以根据其数据分布特征[80]采用分段弯折压缩[81]。

4.3 实验与分析

实验包括语言模型压缩后保留单元概率码表的计算、压缩模型性能批量测试两个方面。

4.3.1 基于排名的概率码表计算

根据单元的兄弟系数和排名位置对单元聚类，得到的概率码表如表 4-3 所示。

表 4-3 基于排名方法单元概率恢复码表(局部)

s \ i	0	1	2	3	4	5
1	0.385	-	-	-	-	…
2	0.322	0.157	-	-	-	…
3	0.285	0.139	0.091	-	-	…
4	0.273	0.132	0.084	0.060	-	…
5	0.262	0.130	0.083	0.058	0.045	…
6	…	…	…	…	…	…

从表中可以看出，排名靠后的单元概率较小。对于排名位置相同的单元，则兄弟系数较小的单元概率较大。

4.3.2 拼音转汉字正确率测试

测试语料和第 4.2 小节用于分析各个方法性能所用的语料相同。即：测试语料 1 是一篇政治文章，共 35,025 字；测试语料 2 来自 863 语料，共 1,801 个句子，23,310 字；测试语料 3 是新闻报道资料，共 4,178 字。词表大小为 51,007。语言模型使用基于概率剪枝和基于排名的压缩方法。拼音转汉字的解码路径宽度为 100 条。采用 bigram 方式建模。三个语料的音转字平均错误率如表 4-4 所示。

表 4-4 三种剪枝方法结合排名聚类方法后模型大小和音转字错误率关系

错误率(%) / 模型相对大小	方法 A+排名	方法 B+排名	方法 C+排名
1.0	6.65	7.85	7.62
1.6	6.10	7.74	7.23
3.8	5.63	6.65	6.40
7.5	5.05	6.05	5.67
21.0	4.25	4.64	4.54
43.0(未压缩)	3.64	3.64	3.64

对应的模型大小和音转字错误率关系曲线如图 4-10 所示。

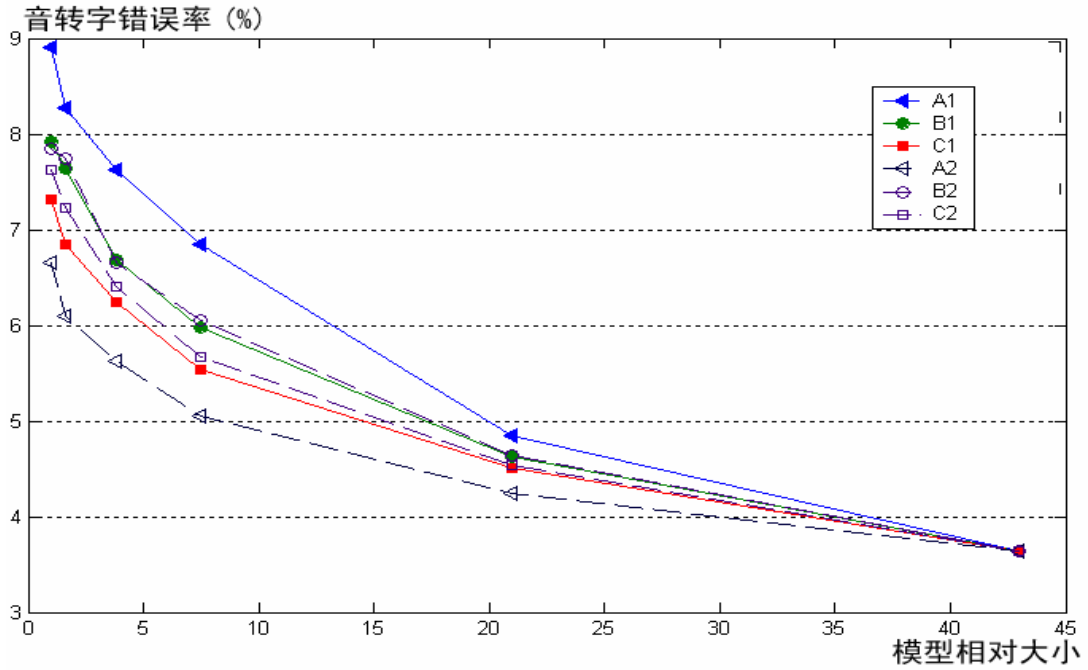

图 4-10 三种剪枝方法分别和排名方法结合后模型大小和音字转换错误率曲线

图 4-10 中，*A*1、*B*1、*C*1 是只使用剪枝方法得到的模型大小和音字转换错误率之间的关系曲线。*A*2、*B*2、*C*2 则是基于剪枝和排名方法压缩后模型大小和音字转换错误率之间的关系。从图中可以看出，基于次数剪枝和基于熵差异剪枝后，采用基于排名的方法对模型压缩基本起不到作用，而概率剪枝的方法和基于排名的方法结合则产生了非常好的效果。在相同模型大小的情况下，音转字错误率明显下降，而且比其他方法都明显要好。

图 4-10 中每条曲线实际上是三个语料测试的平均值。以 *C*2 曲线为例，其对应的三个语料错误率曲线如图 4-11 所示。从图中可以看出，用新方法压缩后的模型对语料 1 和语料 3 的音字转换效果都非常好，错误率曲线随模型大小降低而上升的幅度不是很大；语料 2 的错误率相对上升较多，主要因为语料 2 是 863 的录音文本，过多的考虑了音节之间的搭配关系，因而包含了较多不常见的句子，特别是外国人名较多，导致在模型较小时性能下降的较多。

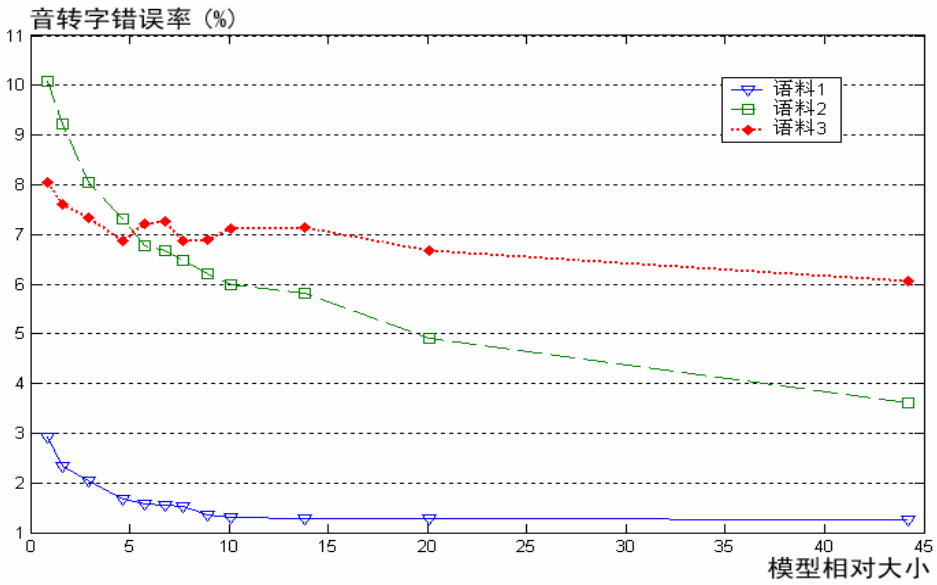

图 4-11 基于概率剪枝和排名方法压缩模型后模型大小和音字转换错误率关系

4.3.3 语音识别批量测试

表 4-5 给出了模型压缩前后语音识别正确率的批量测试结果。测试语料是 863 测试文本集，共 521 句，6751 个汉字。

表 4-5 压缩前后模型用语音识别批量测试的结果

	压缩前	压缩后
模型大小	43MB	1MB
正确率	92.6%	86.9%
压缩前后模型大小比	43/1	
压缩造成错误率上升	78.1%	
不压缩错误率下降	43.5%	

从表中我们可以发现，压缩后模型只有原来的 1/43，而错误率上升完全在可以忍受的范围之内。

4.4 小结

本章通过对传统语言模型压缩方法的研究，提出了一种改进的压缩算

法，即基于概率剪枝和单元排名的模型压缩方法。传统的剪枝方法中，基于模型熵差异的剪枝方法由于较充分的考虑了剪枝前后语言模型在训练语料流上的打分偏差，力图保证剪枝造成的打分差异最小。因此在传统的剪枝方法中熵方法效果较好。但熵方法剪枝后保留下来的单元由于差异较大，不利于进一步压缩。

概率剪枝方法在传统的方法中效果一般，但由于经该方法处理后保留下来单元相对比较有规律，我们引入了基于排名的压缩算法后，丢弃了单元的次数信息，而仅根据单元在列表中的排名位置来确定其概率。该方法和基于概率剪枝的方法结合后效果明显比其他方法要好。一方面是由于基于概率的剪枝方法所保留下来的单元特别适合做排名压缩，另一方面则是因为码表的计算是依靠同类单元的概率平均出来的，具有统计特性，因而排名在某一位置的单元的概率恢复得比较合理。

第五章　统计语言模型应用框架和解码算法

无论是整句音字转换(整句输入法的核心部分)、连续语音识别，还是OCR,使用语言模型都有一个解码的过程。即根据观察到的序列(如拼音串、声音、光学图像等),对所有可能产生该序列的文字串用语言模型进行打分，挑选出得分最高的一个文字串。

在这些应用中，候选组合往往非常多，即搜索的空间非常大，造成了两个后果，一是需要大量的内存，二是搜索的速度比较慢。这两个缺点限制了语言模型的应用范围，特别是使其不能在计算能力较弱的设备上应用。为了解决这个问题，本文以整句音字转换为应用背景，提出一种分层结构的语言模型应用框架，并在此基础上提出基于音节或者码元的束网格解码算法。

本章提出的分层结构和解码算法，非常适合于嵌入式设备上的应用，但它们同时也适合于个人计算机上应用。而且该分层结构和搜索算法具有良好的扩展性，非常容易应用到其他需要使用语言模型的系统中。本文附录给出了使用本章设计的分层结构和束网格解码算法开发的嵌入式设备上的第一个整句输入法，其性能明显优于现有输入法。

本章的内容安排如下：第 1 小节介绍了分层结构的框架设计；第 2 小节提出了解码算法，即基于音节或者码元的解码算法；第 3 小节用实验验证了算法的高效性；第 4 小节为本章小结。

5.1. 语言模型应用框架

5.1.1 逻辑框架

语言模型应用包括音字转换(拼音到汉字的转换)、语音识别、手写整句识别(或者手写识别结果自动校正)等。这些应用都有个“ 解码 ”的过程，即给定输入候选网络，对所有路径进行打分。本文设计的语言模型应用框架其逻辑如图 5-1。其中语言模型内嵌了一个词表，语言模型用于打分，

而解码层则是用来根据输入的候选网格选出最合适的句子。

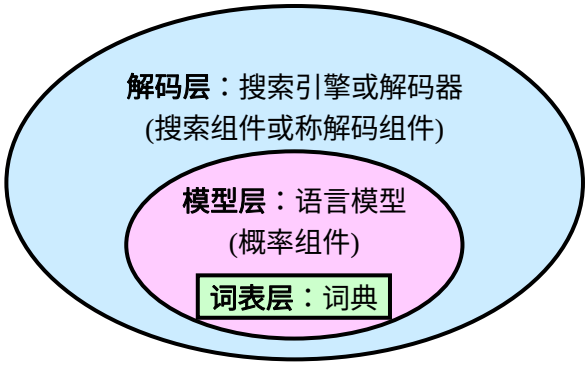

图 5-1 语言模型应用逻辑框架

为了实现这个逻辑框架，本文以音字转换应用框架为例说明如何构建，该框架非常容易修改成 OCR 识别等其他应用的解码框架[1]。

5.1.2 音字转换应用框架的性能需求

音字转换是中文整句输入的核心算法，它将输入的编码串，转换成合适的汉字串。这里的“音”可以推广到类似拼音的其他汉字编码，如笔划编码等，为叙述方便，统一称为音字转换。对于给定汉字编码方式，单字的编码串构成广义的“音节”，所有汉字的编码串组成“音节集”。

国家标准 GB2312-80[7]规定的一、二级汉字共有 6,763 个，GB13000.1[82]规定的汉字则达到 20,902 个，2000 年 7 月颁布的最新标准国家标准 GB18030[83]中收录的汉字则高达 27,484 个。以 400 个拼音计，则 GB2312、GB13000.1 和 GB18030 的汉字重码率分别高达为 17、52 和 69。而实际上由于分布的不均匀，个别音节对应的汉字个数达到一百多个甚至更多。

对于中文输入来说，拼音外的其他编码方式存在另外一个问题：即重码率很低的编码方法，如五笔字形编码，掌握起来比较困难；反之，高重码率的编码方案则一般一个“音节”对应非常多的汉字。总之，汉字输入

[1] 本文设计了一个嵌入式语言模型应用开发包，具体情况请参阅附录

法的好用性和准确性之间存在矛盾。因此，汉字输入一直是中文信息处理研究的一大问题，这方面的研究文献也很多[84-92]。为了解决输入法好用性和准确性之间的矛盾，可以用统计语言模型对所有候选句子打分，给出最优的句子候选[86, 93]，即以整句音字转换为内核，开发出好用的输入法。

由于语言模型非常庞大，整句音字转换解码需要耗费大量的存储空间和 CPU 时间，因此长期以来整句输入法没能在手机等嵌入式设备上应用。

为了解决这个问题，本文试图设计出实用的整句音字转换框架，要求其达到以下目标：

1) 该框架是个基本的框架，能够应用到拼音编码、笔划编码等等场合中。编码方案确定后，能快速开发出基于该编码的音字转换模块用于整句输入法等应用；

2) 该框架对资源需求很小，能够各种计算环境，特别适合掌上设备的计算环境(CPU 较慢，静态存储空间较少，动态内存较小)。

5.1.3 整句音字转换框架分层结构

为了达到前文提出的 2 个条件，本文提出一种分层设计的模式。其具体的目标有两个：1) 语言模型(概率模型)是一个共享的资源，分层次结构能够保证该部分独立于应用； 2) 采用分层次结构能够将框架抽象化，可用于其他类似拼音的编码方式的整句转换模块。

根据以上要求，本文设计的分层次结构如图 5-2 所示：

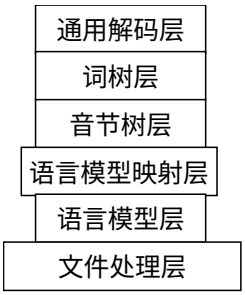

图 5-2 整句音字转换应用分层结构框架

其中文件处理层是一个公共层次，为其他所有层次使用，其功能是为了减少框架的内存使用，将不常用的一些数据结构存储在文件中，当需要时才调入内存中。

语言模型层用于解码时给不同路径打分。对于存储有限的应用系统，可采用压缩的语言模型，规模非常小，但是性能和完整模型相比相差并不大。

映射层的作用就是为不同编码方式的应用系统提供到语言模型统一的访问接口。因为不同的编码方式可能导致汉字和单词的排序、词号等发生变化，而且重码字、词也不同。在拼音编码中，“们”和“门”是重码字，但在笔划编码中，它们互相并不重码。因为可能系统中存在多个输入法或其他应用程序(比如语音识别)同时使用语言模型，因此需要映射层的存在，这样就通过映射层的桥梁作用将语言模型与应用系统隔离，实现了语言模型的共享。

音节树层用于实现音节的切分。对于拼音输入来说，一个拼音就是一个音节，每个拼音对应多个汉字，即存在多音字。实际上，也正是因为存在多音字，才需要语言模型帮助挑选出最恰当的路径作为音字转换的结果候选。同样，如果采用其他的编码方式，也会存在类似的问题。实际上，如果存在一种编码方式既好用(指便于记忆、便于输入)，重码率又低(最理想情况下所有汉字互相不重码)，那么这种编码方式根本不需要语言模型就能获得很不错的整句转换性能。但实际情况却没有那么好，一般重码率低的编码方式都普遍存在难记忆、难使用、单字平均编码长等缺点。因此，采用整句输入，依靠语言模型自动校正的方式是个解决之道，即设计编码方式时注重好用，不强调重码率。这样就存在类似汉语拼音编码中的“多音字”问题。为此，本文定义如下的基本术语：

码元：指构成汉字输入编码的基本单位，每个汉字对应一个有限长度的码元序列。比如 26 个字母 a 到 z 构成了汉字拼音编码的基本码元。在笔划编码方面，横(一)、竖(丨)、撇(丿)、捺(丶)、折(乙)构成了笔划编码的码元，也是笔划编码国家规范[94]。

音节：这里说的音节是拼音编码中音节概念的扩展。任意一个汉字对应的码元序列都对应一个音节，但可能有多个汉字对应同一个音节。比如在由横(一)、竖(丨)、撇(丿)、捺(丶)、折(乙)五个笔划构成的汉字编码方法中，“丨乙一丨一”构成一个音节，对应“由”、“田”等字。为方便起见，每个音节对应一个编号。

在整句音字转换中，输入的是码元串，输出的是整个句子候选，而从码元到句子的转换过程，中间需要经过音节层的处理，即先需要将码元串转换成音节串，然后再将音节串转换成汉字串，类似语音识别中的音节同步解码算法[77, 86]。

将码元串转换成音节串的过程称为**音节切分**。为提高音节切分的速度，需要将所有合法的音节按其码元串建立一棵音节树。以拼音编码为例，音节树如图 5-3 所示：

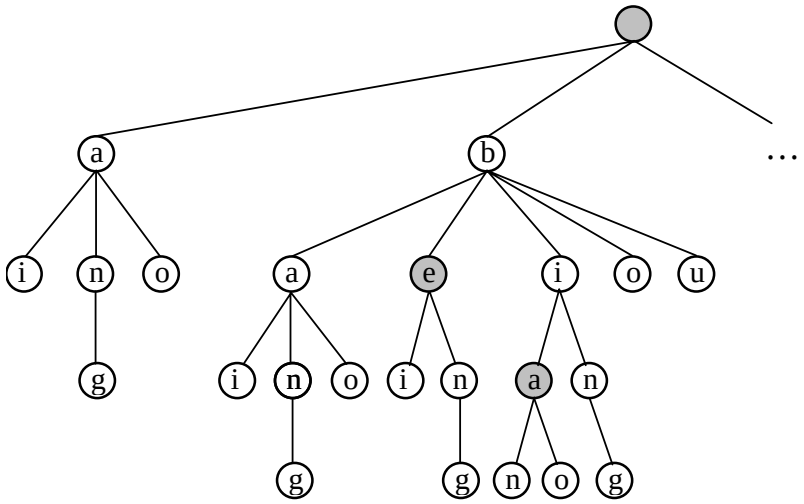

图 5-3 音节树设计

其中灰色的节点表示不可终结，即从根节点到当前节点构不成合法的音节。笔划编码等其他类型的编码也可以构建类似的一棵音节树以实现音节的快速切分。

词树如图 5-4 所示，它和音节树在结构上类似，不同的是，音节树是

以码元为树节点，一个树枝表示**一个**合法的音节；而词树则以音节为树节点，每个树枝表示**一个或者多个**词。

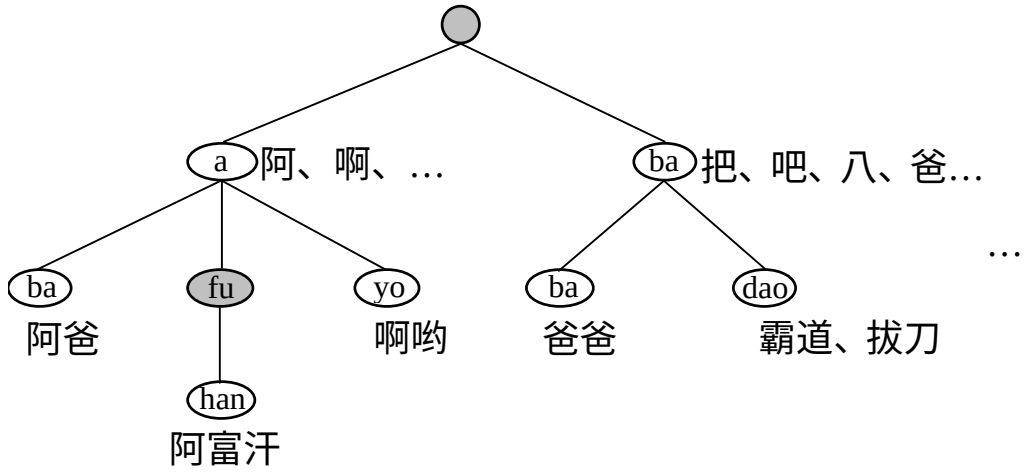

图 5-4 词树设计

词树的用途是在码元串经过切分成为音节串后，对任意起点的音节子串，快速获得其所有可能字、词候选，然后将它们加入到解码路径中[95, 96]。

通用解码层用于实现码元串到汉字串的转换。

从码元串到汉字串的转换过程叫解码(或搜索)，用于实现这个过程的算法叫解码算法(也称搜索算法)。解码层综合使用词树、音节树和语言模型，对用户输入的码元串进行切分，根据词树对所有可能的路径进行打分，输出得分最高的路径。因此，解码层的设计在很大程度上决定了音字转换解码的效率。

5.2 基于分层结构的解码算法设计

一些文献提出了一种音节同步的解码算法[77, 93]，不过该算法设计相对比较复杂，在解决拼音串边界存在混淆的时候效果不好(如，“henai”究竟表示“喝奶”还是“很爱”)。双栈的解码算法[86]使用两个栈分情况处理已到达音节边界的路径和尚未到达音节边界的路径，效率低下，并不适合于通用的音字转换解码算法。

5.2.1 基于音节的束网格(Beam Grid)解码算法

新的解码算法除了能用于各种编码方式的整句解码外，还应满足三点要求：占用内存少、速度快(占用 CPU 少)且是递增式解码。所谓递增式解码指在码元串末尾追加新的码元时，充分利用前次解码的结果，并以其为基础对改变的部分做递增解码。因此其解码速度更快，效率更高。本文提出了一种**基于音节的束网格(Beam Grid)解码算法**。

该算法基本思路如下：给定一个码元串，先对码元串进行切分，形成音节串。解码时以音节为基本扩展单位（步长），形成网格状路径，如图 5-5。每个网格表示到达当前步的一条路径，可以通过网格保存的数据方便的往前回溯这个路径。

算法实现如下：

在束网格解码中，基本的单元是网格节点，假设束的宽度为 W，则每步有 W 个网格节点，代表了 W 条路径。为了将该 W 个节点管理起来，每步分别需要一个额外的指针指向本步得分最大的网格节点。每个网格节点包含如下内容：

该节点对应的词号(wWordID)：即被扩展出来的这个节点代表哪个单词；

到目前为止的得分(dfScore)：即以该节点为结束点的路径得分；

路径前一节点(pPreStepUnit)：指明该节点是从哪个历史节点扩展出来的，据此可以方便的回溯出整个路径；

本步下一高分节点(pHigherUnit)：将在本步终结的所有网格单元按得分高低排序，本指针用于往高分一端遍历；

本步下一低分节点(pLowerUnit)：将在本步终结的所有网格单元按得分高低排序，本指针用于往低分一端遍历；

假设词表中最短的单词为单字词，最长的单词含 $L_{\max}$ 个汉字，即词长范围为 $1\sim L_{\max}$，因为束网格解码中，每一步对应一个音节，因此对于解

码过程中第 i 步某一特定网格单元来说，指向路径前一单元的指针(pPreStepUnit)可能指向第 $i-1$ 步的某个网格单元，也可能指向 $i-L_{\max}$ 步的某个网格单元，或者他们之间某步的某网格单元。如下图所示，第 i 步得分第一的单元是从第 $i-2$ 步扩展出来的，而得分第二的单元是从 $i-1$ 单元扩展出来的。

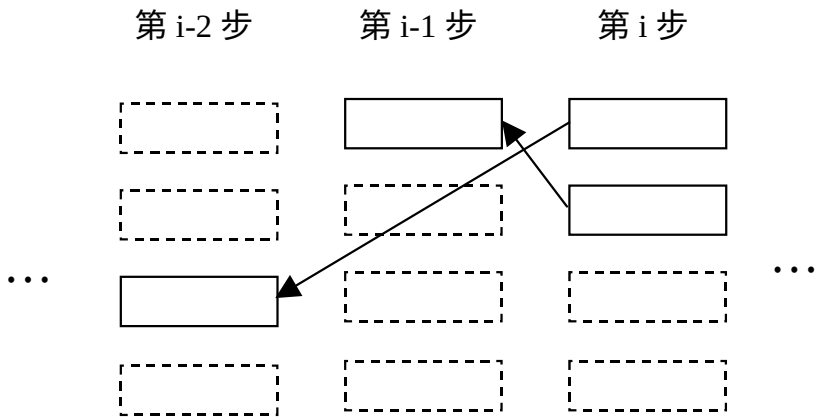

图 5-5 解码算法示意图

核心部分的形式化解码算法如下：

假设码元串追加了一新码元(相当于输入时用户新输入一个编码)，且由于该新码元的增加，经过重新切分码元串得到音节串，将所得音节串和前次解码的音节串比较，对其相同部分，保留其解码结果。从发生变化的第一个音节开始到结束，都需要重新一步步解码。假如发现音节串最后增加了一个新的音节(或者最后一个音节发生变化，如从 a 变为 an)，此时需要扩展路径。假设该新增音节在音节串中是第 i 个，则 $i-1$ 步(包含)之前的音节串都已经经过扩展。

1. $curLen = L_{\max}$；
2. 根据词树，对从 $i-cueLen+1$ 步(含)到 i 步(含)长度为 $curLen$ 的音节子串在词树中找到与之匹配的单词集合 ϕ_{curLen} (0 个、1 个或者多个)，ϕ_{curLen} 不空，转 3，ϕ_{curLen} 空，转 6；
3. 对 $i-cueLen+1$ 步的所有单元对应的每条路径，分别一一尝试扩展

合法单词集合ϕ_{curLen}中的每一单词，并算得分 *score*；根据第*i*步有效网格单元个数选择 4 或者 5 步执行；

4. 如果第*i*步的有效网格单元还未到*W*个(束宽度)，则增加一新单元，根据得分 *score* 排到合适的位置，并将 *pPreStepUnit* 指针指向$i-cueLen+1$步相应的单元，转 6；
5. 如果第*i*步的有效网格单元已经达到*W*个(束宽度)，则判断新路径得分是否比第*i*步已有单元中的最低得分者高，如是，则将原有最低得分单元抛弃，新单元放到第*i*步合适位置，并将 *pPreStepUnit* 指针指向$i-cueLen+1$步相应的单元；否则直接抛弃扩展出的新路径；
6. $curLen = curLen - 1$;
7. 如果*curLen*不为 0，转 2；
8. 结束；

在网格结构的解码过程中，解码路径中不包含那些没有到达词边界的情形，网格中第*i*步保存的全部是第*i*步到达词边界的路径候选，这样保证路径的利用效率很高。当路径扩展到达$i+1$步时，先通过词树的分析结果得到那些跨越多步的路径扩展(跨度最大的路径扩展可以从$i+1-L_{\max}$步节点一次扩展到$i+1$步节点，即跨越$L_{\max}$，相当于扩展了一个长度为$L_{\max}$的词)。因此，通过静态词树(可存放在文件中)的分析，缩小了路径扩展的存储开销，提高了内存的使用率，并且在相同内存消耗的条件下，提高了解码的准确性。

解码过程中记录下来的束网格分步路径如图 5-6 所示(束宽度为 10)。每个节点画出了汉字、词号和到当前节点的路径得分。从图中看出，解码路径到达 2、3、4、5 步时，得分最高的路径分别为“风味”、“冯伟伟”、“风微微吹”、“风微微吹过”，都是最合理的路径。

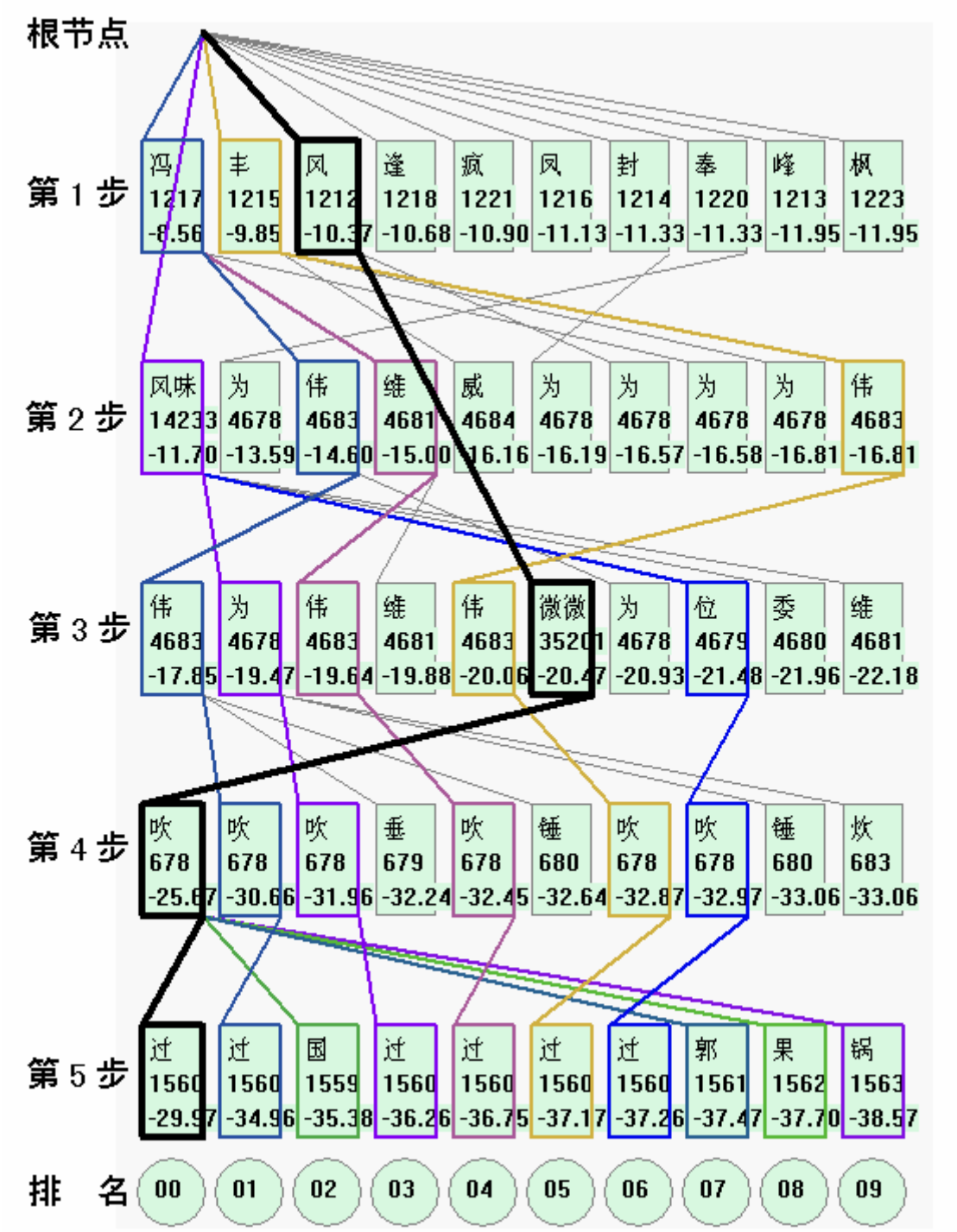

图 5-6 解码分步结果示意图

5.2.2 基于码元的束网格(Beam Grid)解码算法

5.2.1 设计的基于音节的束网格解码方法，满足递增解码、占用内存和 CPU 少等特点，具有非常强的通用性。

该方法的缺点是比较依赖音节的切分。因为扩展是以音节为单位的，那么所有的路径对码元串的切分只能一样。实际上，一个拼音串，如果不含分格符号，则可能存在音节切分歧义。主要存在以下几种情况：

1. 由于 g、n、r 既可以做某些音节的结尾，又能做一些音节的开头，且汉语拼音中存在零声母现象，因此就造成歧义，比如“ henai ”，可以是“ hen ai ”(很爱)，也可以是“ he nai ”(喝奶)。对于 g 和 n，与他们发生歧义的零声母音节主要包括 a/ai/an/ang/ao/e/ei/en/eng/ou 等；对于字母 r 来说，切分歧义主要由 er 引起，比如“ er an ”(而安)与“ e ran ”(愕然)。这种类型的歧义最多，达 1,300 多种组合。
2. 一些双韵音节可以拆分为两个音节。比如“ xian ”，即可能是“ xian ”(对应先、现等字)，也可能是“ xi an ”(西安)。这种类型的歧义有 100 多种组合。
3. 个别零声母音节可以拆分成两个独立音节。如“ ao ”，可以拆分成“ a ”和“ o ”，则几乎所有带 ao 韵母的音节都可以拆分，造成歧义。
4. 多个歧义相邻，造成更复杂的组合歧义。典型的如，句子“ 西安刚… ”对应拼音“ xiangang… ”，如果用最长匹配方法切分音节，得到的结果是“ xiang ang… ”，而正确切分是“ xi an gang … ”，可见没有一个音节和正确切分匹配。而且，该串还存在着“ xian gang… ”、“ xian gan g… ”、“ xiang an g… ”、“ xi ang ang… ”、“ xi ang an g… ”等多达七种不同切分方式。

以上这些例子仅仅是以拼音编码为例，如果考虑到其他的编码方式，比如笔划编码，情况更复杂。对于笔划编码来说，不存在声韵母的问题，

每一个笔划都可能是一个字编码的开始，因此这种音节边界混淆更多。

这类音节切分歧义导致这样的结果：一旦码元串切分成音节串时发生错误，则正确的句子永远出不来。基于音节的束网格解码产生的所有路径都对应某一种音节切分。这样，即使存在对应其他音节切分的路径的得分比当前所有路径都高，这样的路径也不可能在候选中出现。为了解决这个问题，需要改进解码方式，要求音节切分和后续的解码是结合在一起进行的，也就是说，要保证各种合法音节切分的产生的路径都能参与竞争，最后得到的前W(束解码宽度)条路径，是全局最优的。

基于这些考虑，本文提出另外一个改进的解码算法——**基于码元的束网格(Bean Grid)解码算法**。

基于码元的束网格解码算法的根本改进在于，不预先对码元串进行切分，而是将码元切分融合在解码之中，随着路径的推进，自动决定某一位置是否作为音节边界。

在这种新的解码算法中，每一步分别代表一个码元，用户每输入一个码元，解码路径往前扩展一步。解码所需要的网格节点包含如下的数据：

该节点对应的词号(wWordID)：即该节点被扩展出来时究竟代表了哪个单词；如果该节点为有效词号，表明当前步骤到达了音节边界，解码产生一个新的音节；如果词号为无效，则表明当前步骤并没有到达一个音节的边缘，比如用户在输入 wo 之后新输入了码元 h，则当前步骤并没有到达音节边界，仅仅完成了音节的部分输入；

当前音节树节点(pCurSylNode)，即本条路径到达当前步时指向的音节树节点。保留该节点的目的是为了在扩展下一步时能够直接从音节树的某一节点往它的儿子节点扩展，加快扩展速度。对于本步未到达音节边缘的，指向音节树的某个树内节点(非根非叶节点)；对于本步到达音节边缘的，则指向音节树的根节点。

当前词树节点(pCurWTNode)，即本条路径到达当前步时指向的词树的节点。保留该节点的目的是为了在扩展下一步时能够从词树的某一节点直

接往它的儿子节点扩展，加快扩展速度。类似当前音节树节点，对于未到达词边界的，指向词树的内部节点(即非根非叶节点)；而对于到达词边界的，则指向根节点；

到目前为止的得分(dfScore)：即以该节点结束的路径得分情况；

路径前一节点(pPreStepUnit)：指出该节点是从哪个历史节点扩展出来的，可以根据路径前一节点方便的回溯整个路径；

本步下一高分节点(pHigherUnit)：将在本步终结的所有网格单元按得分高低排序，本指针用于往高分一端遍历；

本步下一低分节点(pLowerUnit)：将在本步终结的所有网格单元按得分高低排序，本指针用于往低分一端遍历；

图 5-7 是基于码元的束网格解码算法原理示意图。每个节点的第一部分表示已经到达音节边界的单词(□表示该节点没有到达词边界)，括号内表示当前路径所处的音节树和词树节点(▲表示根节点)。

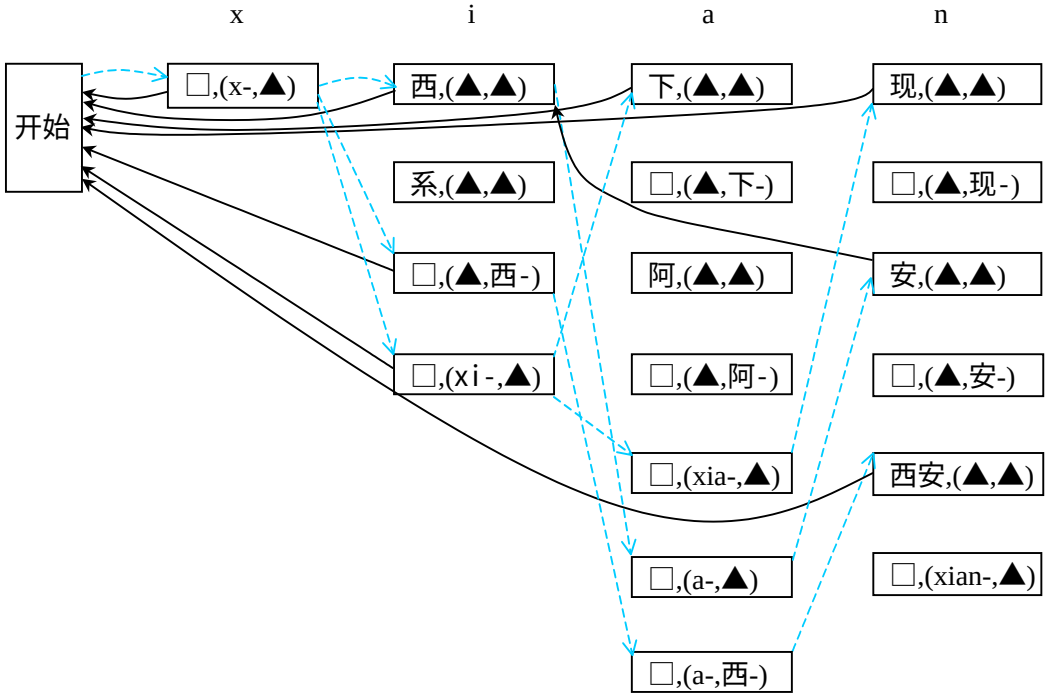

图 5-7 改进的解码算法——基于码元的束网格解码算法原理图

在从码元串到汉字串的解码过程中，任意一步都可能存在分支。比如码元串从“ xi ” 增长到“ xia ”，可能是到达音节边界后重新开始一个新音节(xi a)，也可能是音节尚未结束(xian,xiang 等情况)。解码时自动对两种情况进行处理。图 5-7 中只画出了部分节点的扩展和回溯方向。虚线表示路径扩展，实线表示路径回溯，路径回溯跳过了那些音节树扩展的部分，而直接以词为单位将路径节点接到上一个合适的位置。

从图可以看出，在路径扩展时，每一步都是从其前一步的某个节点扩展出来的。这样，每一步扩展时，可能只是在音节树上前进了一步，也可能只是在词树上前进了一步，或者是到达了词边界并应用 *n*-gram 模型对路径打分。

这种方法的好处有：

1. 每一步都提供了所有可能的选择，包括各种音节切分下的路径扩展，保证解码全局最优；
2. 每一步都需要计算，但是每一步单独的扩展计算都非常少，即计算分布很均匀；而基于音节的解码在音节内部时计算很少，但到达音节边缘时计算会明显增加(用 *n*-Gram 模型打分)，这样在计算能力较低的设备上可能会出现解码不顺畅的情况；
3. 对于用户删除最后一个码元的操作几乎不需要进行任何计算。

为了有效利用搜索路径，需要将路径中一些共同的信息用共享单元表示起来。因为路径中存在两种情况，一种是到达词边界的，这类情况的处理类似于基于音节的束网格解码，由于有语言模型的指导，保留在路径中的都是打分最高的连接关系；另一种是没有到达词边界的(甚至没有到达音节边界的)，如图 5-8 所示，当输入拼音串“ …qukua ”时，“ kua ”部分实际上形成了没有到达词边界或者音节边界的多种可能(假设共有 *A* 种)，在用户输入新的码元之前不知道哪个是正确的。每种可能都可以接在其前趋候选路径“ 去/取/娶/… ”中的任何一个，甚至可以从更前一步的路径节点扩展出来(如单词“ 去苦 ”从更早的节点扩展而来)。假设可扩展的前驱节点个数为 *B* 个，如果对所有的合法前趋节点一一扩展出未达到边界的节点，

则当前步的路径宽度需要增加额外的 $B\times A$ 条才能保证所有可能候选都能保留下来，以保证输入下一个码元时该种可能还保留在路径中。在基于码元的解码中，B 一般都比较大，造成路径开销 $B\times A$ 很大。本文采取的办法是将这些未到达边界的情况用一个共享结构表示出来，先不加入路径。因此额外的开销只有 A，从而有效利用了路径宽度。

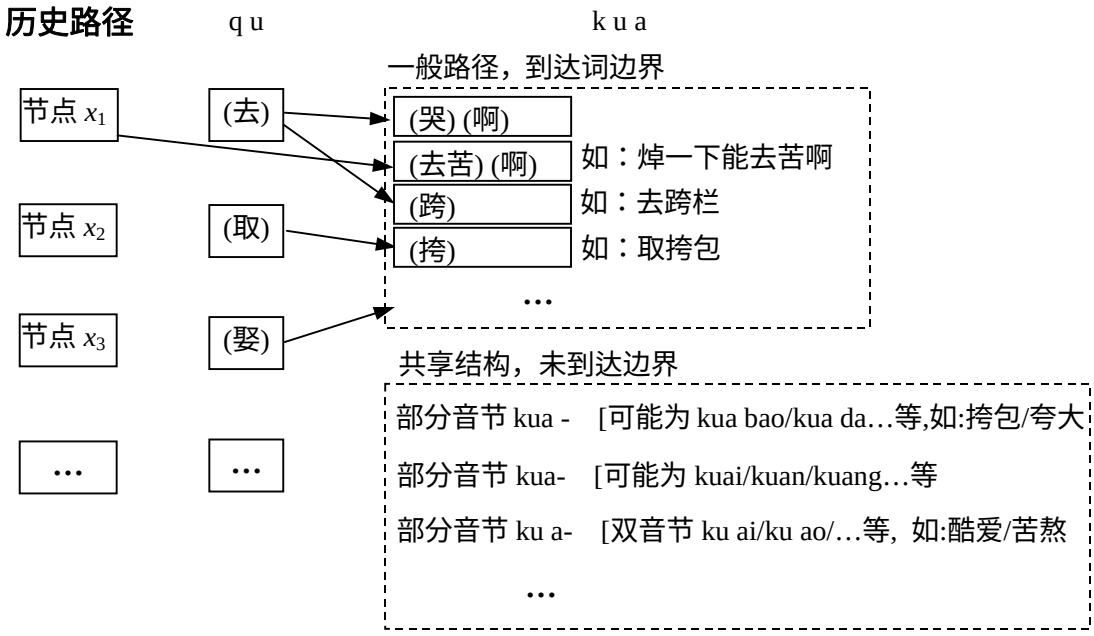

图 5-8 改进的解码算法——基于码元的束网格解码算法的共享结构示意图

语音识别中在路径扩展中也有类似的问题。当对一帧新语音进行处理后，当前状态可能是在基元内部、到达基元边界、到达音节边界或者到达词边界中任何一种情况，如果将那些没有到达边界的情况都一一加入到路径，非常容易造成路径爆炸，而且预测(Look ahead)不准确时可能造成正确候选被错误剪枝。比较好的解决方法是先将未到达边界的部分单独管理，等到新的语音产生并到达边界时，加入到路径中，从而有效的抑制了路径爆炸[1]。

基于码元的解码算法相对于基于音节的解码算法，使用内存要多一些(解码步长为一个码元，因此路径较长，而且需要额外的内存用于保存未到词边界的共享结构信息)。但该方法更加灵活，解码的准确性要更高，对音

节边界的处理更加鲁棒，适合于计算能力较强的高端嵌入式设备。

5.2.3 两种解码算法的推广问题

5.2.1 和 5.2.2 小节提出的解码算法是为整句音字转换这种应用设计的。对于其他的应用，比如语音识别、OCR 等，其前端输出可能是多路的词网格候选，和单路的输入串不同。因此在扩展路径时，需要修改根据输入串(网)从词树中查找候选单词集合的模块。对于给定的起始扩展点，需要从词树中拿出所有可能的候选单词(可能有多路)加入到解码路径中去。此外，可能为了前端识别(如语音识别)的剪枝，需要将语言模型预测打分提前加入前端识别中去[97]。

5.3 实验和分析

基于音节和码元的两种束网格解码算法对解码的组织方式有所不同，基于码元的方法处理音节边界更有优势。

5.3.1 性能和速度测试

表 5-1 和表 5-2 分别列出了使用完整模型和使用压缩模型时，基于音节的束网格解码算法解码宽度和音字转换正确率的关系。完整模型大小为 43MB，压缩模型为 1MB，采用 bigram 建模。测试语料同第四章，即测试语料 1 共 35,025 字；测试语料 2 来自 863 语料，共 1,801 个句子，23,310 字；测试语料 3 是新闻资料，共 4,178 字。为了避免音节切分歧义对解码的影响，强制在输入拼音串中加入音节分隔符(如“ 西安市 ”对应的拼音串为“ xi an shi ”)，保证音节切分完全正确。

表 5-1 解码路径宽度和音转字正确率关系(完整模型)

路径宽度		1	2	3	4	5	不限制
正确率(%)	语料 1	87.08	97.90	98.42	98.48	98.53	98.75
	语料 2	78.37	94.01	95.29	95.71	95.99	96.45
	语料 3	80.68	92.60	93.42	93.30	93.13	93.92

表 5-2 解码路径宽度和音转字正确率关系(压缩模型)

路径宽度		1	2	3	4	5	不限制
正确率(%)	语料 1	87.85	96.45	97.02	97.14	97.37	97.75
	语料 2	79.15	89.57	90.33	90.77	90.88	91.45
	语料 3	82.07	91.10	91.69	91.94	91.96	92.25

对应的解码宽度和性能关系如图 5-9 所示。从图中还可以看到一个有趣的现象，即使用完整模型时，如果路径宽度只有 1，其性能反而不如使用压缩模型。产生这个现象的原因是由于完整模型有非常多的单元参与竞争唯一一条路径，不少不可靠的单元在压缩时被剪掉，因此在使用压缩模型时性能反而上升。但当路径扩大时，路径前后的连接关系能保存的更多更完整，因此完整模型的优势开始显示出来。类似的现象还存在完整模型对语料 3 的解码中，当路径为 4 和 5 时，性能反而略微下降，经过分析，其原因是由于该语料和训练语料差距较大，当路径宽度为 4 和 5 时，路径中刚好包含了部分和该语料不匹配的单元，造成解码错误，随着路径进一步扩大，这种现象就消失了。

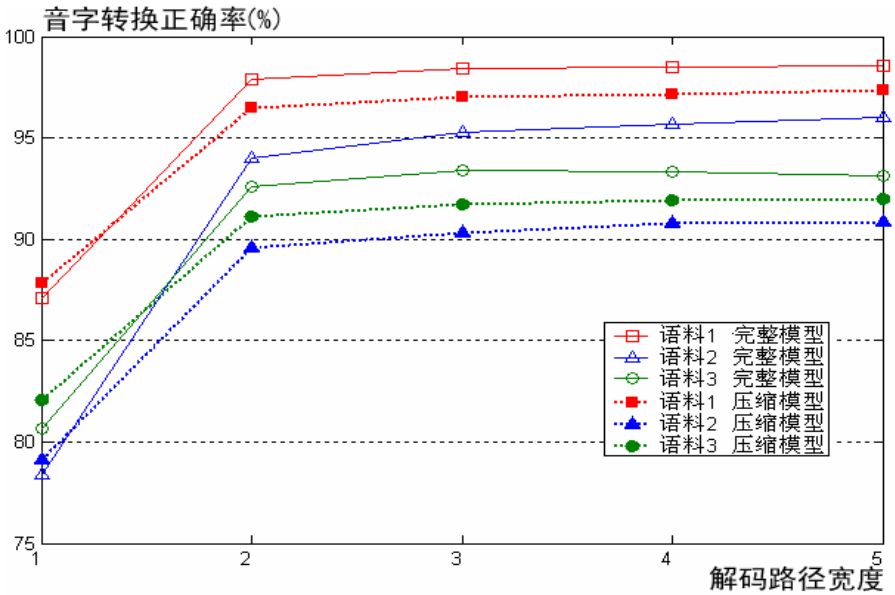

图 5-9 解码路径宽度和音字转换正确率关系图

由表 5-1、表 5-2 和图 5-9 可以看出，由于路径宽度不足造成的错误率的上升非常小，5 条路径基本已经到达解码性能的极限。

图 5-10 给出了使用压缩模型时解码速度和路径宽度的关系曲线。该曲线是解码算法在 PC 机上批量测试的运行结果，机器 CPU 的主频为 P4 2.0G，内存为 512M(实际上内存大小没有影响，因为内存开销不大) 。从图中可以看出，三个语料的解码速度非常相近(解码速度取决于算法本身，与具体的语料无关)。随着路径宽度增加，解码速度降低。路径宽度为 5 条时的解码速度仍高达 5,000 字/秒。使用完整模型时需要访问更多的模型参数，因此降低了解码速度，在路径宽度为 5 时，大约为 1000 字/秒。这种速度也是非常可观的。

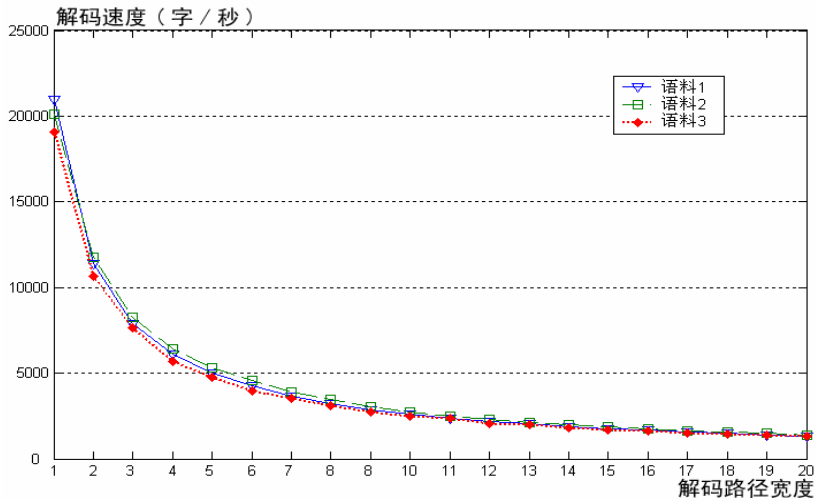

图 5-10 采用压缩模型时解码路径宽度和音字转换速度关系图

另外，由于拼音音节间增加了分隔符，基于码元的解码算法性能和基于音节的解码算法性能完全一样。它们的速度也相近。

根据图 5-9 和图 5-10 结果，本文选择最终应用系统的解码路径宽度为 5 条(具体应用实例见附录)。在该解码宽度下，基于音节的解码算法内核对内存需求只有 30KB 左右，基于码元的解码算法内核的内存需要稍多，大约需要 40KB。这个内存需求对绝大部分嵌入式设备来说都不存在问题。

5.3.2 基于码元解码算法的鲁棒性测试

如果将输入串中的分隔符去掉(如“西安市”对应“xianshi”)，重新使用基于音节和基于码元的两种解码算法对语料解码，然后和含分隔符的解码结果比较，可以考察两种算法对音节边界存在混淆时处理能力的差异。

句子越长，则该句子音节切分出现错误的可能性越大。语料 2 的句子长度较小，句子长度平均为 13 字/句，且句长分布比较合理，如图 5-11 所示，因此选择以语料 2 为实验集。

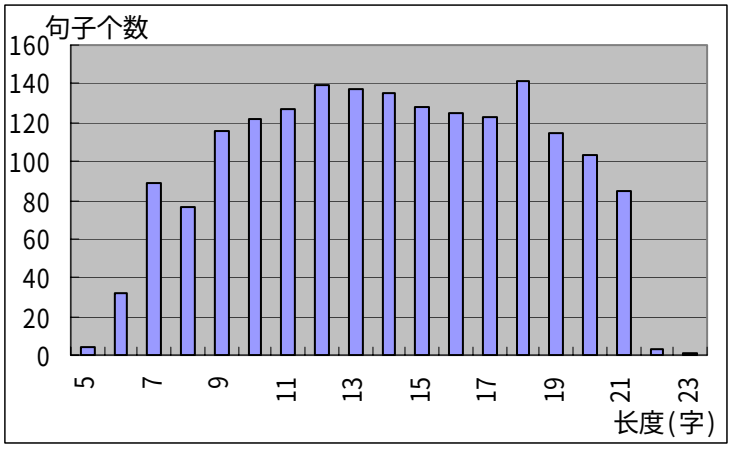

图 5-11 语料句子长度分布图

解码宽度为 5 条，两种算法解码性能(包括切分音节的性能)比较如表 5-3 和表 5-4 所示。

表 5-3 两种算法对音节混淆的处理能力比较(完整模型)

	基于音节的解码		基于码元的解码	
	含分隔	不含分隔	含分隔	不含分隔
正确率(%)	95.99	88.71	95.99	96.03
音节切分错误句子个数	N/A	236	N/A	2
含切分错误的句子比例(%)	N/A	13.1	N/A	0.1

表 5-4 两种算法对音节混淆的处理能力比较(压缩模型)

	基于音节的解码		基于码元的解码	
	含分隔	不含分隔	含分隔	不含分隔
正确率(%)	90.88	83.90	90.88	90.79
音节切分错误句子个数	N/A	236	N/A	6
含切分错误的句子比例(%)	N/A	13.1	N/A	0.3

从表中可以看出，基于音节的解码算法在处理不含分隔符的拼音串时性能明显下降，而且有高达 13.1%的句子音节切分出现错误。而基于码元的解码算法在处理不包含分隔符的拼音串时，性能基本没有下降(其中使用完整模型时反而上升了 0.04%)，而且解码所得句子的音节切分正确率高达 99.7%。如果采用压缩之前的模型进行基于码元的解码，则 1,801 个句子中，只有两个句子的音节切分发生错误，音节切分正确率高达 99.9%。可见，基于码元的解码算法保证了解码的全局最优。

5.4 小结

本章针对语言模型应用中的解码问题，提出了一种分层的应用结构，并以整句音字转换为例，提出一种基于音节的束网格解码算法和一种基于码元的束网格解码算法。实验结果表明，这两种解码算法能用很有限的资源实现很好的搜索效果。前者可用于对音节边界明显的编码方式做整句解码，而后者不需要做音节切分，自动针对编码串做全局解码，提高了音节边界存在混淆时的解码精度。而且后者还可用于边界混淆的编码串的音节自动切分。

采用本章解码技术实现的具体应用系统请参考附录。

第六章　论文工作总结

6.1 语言模型平滑、自适应、压缩以及解码之间的关系

在语言模型平滑、自适应和压缩三个研究方向中，语言模型平滑是语言模型最早研究的问题。

在语言学研究的发展过程中，统计语言学(Stachastic Linguistics)充分吸收了现代数学、计算机科学、控制论以及人工智能等学科的有用思想和方法，用全新的方法和视角研究语言学。比如在统计语言学中最早提出的统计规律之一即齐普夫定律(Zipf's Law)[80](这个定律是因其研究者、美国语文学家齐普夫而得名)。这个时期数理方法对语言的研究实际上只停留在 unigram 上。实际上最早的语言模型平滑技术即加性平滑[26]提出时并不是为了用于对句子进行打分，而是为了研究单词的构成比例关系和分布情况。

二战后，科技文献的爆炸式增长直接推动了统计语言学的发展。人们需要机器来做翻译工作和检索工作，以及试图通过计算机实现人和机器的自由对话，为了达到这个目标，需要对语言进行研究，特别是语言单元之间概率连接关系的研究。但是由于 n 元统计语言模型参数过多，语言模型的平滑问题显得非常重要。

6.1.1 语言模型平滑是自适应和压缩的基础

随着统计语言模型平滑技术的发展和海量数据的出现，语言模型的应用获得了长足的发展，但人们在使用过程中发现统计语言模型对语料的领域性质的依赖非常强。当收集到的训练语料和语言模型的应用领域不匹配时，语言模型性能会急剧下降。实际上，某些领域的语料收集工作非常困难，为了解决语言模型在这些领域的应用问题，研究人员提出了语言模型自适应的概念。即在语言模型使用过程中选择合适的语料训练出合适的模型以保证模型的准确性。

随着可获得语料规模的增加，语言模型的性能得到极大提升，同时，模型大小也随着急剧膨胀。当前，嵌入式设备发展迅猛，越来越多的语言模型应用系统需要移植到嵌入式设备上，但这类设备的计算和存储能力限制了系统的移植工作。为此，在语言模型平滑的基础上，语言模型的压缩显得尤为重要。

因此平滑技术是语言模型自适应和压缩的基础，因为自适应和压缩都需要性能优越的种子语言模型作为出发点，而种子语言模型的建立依赖于出色的语言模型平滑算法。

6.1.2 语言模型自适应和压缩中也需要平滑

语言模型自适应和压缩本身也会遇到平滑的问题。

比如，在本文提出的在线递增式语言模型自适应过程中，随着用户语料的增加，需要考虑用户语料如何结合到整个模型中的问题，因此就需要充分考虑单元之间的平滑。特别是在自适应过程中参数归一化时，需要使用第二章提出的基于泊松过程假设的 Katz 平滑算法用于计算折扣系数等。语言模型压缩中也有类似的问题的，优秀的平滑算法对语言模型参数的刻画更为准确，因此也保证了保留下来的单元确实是最重要的单元。还有，从理论上说，压缩技术的剪枝过程同时需要进行语言模型平滑，否则由于某些单元被剪掉造成概率总和不为 1。

6.1.3 好的语言模型能有效地指导解码

语言模型应用时一般都涉及到解码问题。在没有语言模型指导的情况下，每一步分支是$|V|$(词表大小)，即分支个数和词表大小一样。有了语言模型的指导，可以采取剪枝措施。语言模型刻画的越精细，则剪枝所用的参数越准确，那么在相同的解码宽度下，正确路径越可能保留下来，从而提高了解码的正确性。

6.2 本文的研究贡献

本文针对统计语言模型中的若干难点，以提高语言模型的可用性为根

本目的，在解决语言模型数据稀疏问题、领域依赖问题、规模过大问题以及解码速度问题等四个方面，进行了深入的研究，提出了若干新方法、新策略，并通过实验证明了其有效性。

概括来说，本文的工作重点与贡献主要体现在如下几个方面：

6.2.1 提出基于泊松过程假设的 Katz 平滑方法

平滑方法是解决模型数据稀疏问题的关键。本文通过对传统语言模型 Katz 平滑算法的分析，首先是明确了 Katz 平滑应该采用全局折扣因子(因为部分文献在这个问题上存在混淆)，并以基于全局折扣因子的平滑方法为实验基准(Baseline)；接着本文通过分析 Katz 平滑的不足，提出一个假设，即语言单元在语料流中的出现过程为一泊松过程，然后用假设检验的方式加以验证，并求出了出现次数不同的单元其实际可能出现次数的置信区间上下限。结果表明，次数较低的历史单元，其期望次数的置信区间比较大，从而验证了出现次数不同的历史单元造成的概率可靠性不一样。对于概率可靠性较低的单元，其概率宁小勿大。因为其概率过小只会造成其竞争不过其他单元，至多造成该单元不会在候选路径中出现；而其概率过大则造成其在竞争中过于活跃，在其不该领先时领先，屏蔽了多个其他单元，反而造成更多错误。

基于泊松过程假设的 Katz 平滑方法较好的解决了语言模型平滑中折扣系数的估计问题。本文的实验结果也证明了，使用该方法能显著降低模型困惑度。批量测试表明，新方法能显著降低拼音转汉字和连续语音识别的单字错误率。

6.2.2 提出一种在线递增式统计语言模型自适应框架

传统的语言模型自适应方法，大多采用静态的方法解决语言模型对领域的依赖问题。传统方法没有将使用语言模型过程中得到的语料(即应用系统产生的语料)自适应到模型中去，或者只是简单的使用这些语料从背景语料中筛选出一些相近语料用于改进模型，如果当前话题在原始语料库中没有相近语料，则该方法很难起到较大效果；对于用户风格的适应，传

统的方法适应效果也不是那么令人满意，因为基于 Cache 等方法其不能记忆较长时间之前的信息。为解决这些问题，本文提出了在线递增式统计语言模型自适应框架。在这一框架中，提出了一种采用动态加权因子的参数调整策略，以加快模型自适应速度；为了防止模型在自适应过程中发生振荡，采用了振荡抑制策略，这些策略保证了模型最终收敛于 MAP 自适应方法。通过实验证明，在线递增式语言模型自适应框架比较有效，平均能降低音转字错误率 20%以上。

6.2.3 提出基于条件概率和排名的语言模型压缩方法

高性能统计语言模型所用训练语料较多，训练得到的模型规模较大，比较难以在计算能力较差的嵌入式设备上应用。本文通过对传统方法的分析和实验，提出一种语言模型压缩算法，即将基于条件概率的剪枝和基于排名的压缩方法结合起来。采用排名方法后不需要存储单元出现次数信息，节省了大量空间，而且用统计的方法恢复单元概率，保证了模型参数的准确性。实验表明该方法在压缩性能上好于现有方法。在模型压缩到 1M 以下时性能仍然较好。

6.2.4 提出分层结构的语言模型应用框架及束网格解码算法

针对语言模型的解码时的速度问题，本文提出一种分层结构的语言模型应用框架。以音字转换为例，采用分层结构，便于多个应用之间共享语言模型。本文设计了基于音节/码元的束网格解码算法。该算法在解码宽度为 5 条路径时就基本上达到全搜索的性能，从而有效减少了对计算资源的需求，并提高了解码效率。测试表明其解码速度远远高于实际需求。

本文利用该框架，在 Symbian、Palm OS、WinCE 等嵌入式平台上实现了整句中文输入法，实验表明其性能明显高于现有输入方法(见附录)。

同时，该框架也可应用于 PC 机上，并可推广到 OCR、语音识别等应用中。

6.3 下一步研究的展望

本文虽然对统计语言模型的平滑方法、自适应、压缩等方面进行了一些研究，提出了一些新方法和新思路，取得了一定的成果，但同时也发现一些不足之处。下面将指出这些不足点，以及计划进一步深入开展研究的方向。

1) 语义知识在中文统计语言模型中的应用

长期以来，语言模型的建模方式分确定性语言模型和统计语言模型两大阵营。由于海量语料的出现，基于海量数据的统计语言模型获得了长足的发展并且取得了巨大的成功，但同时也渐渐暴露了统计语言模型的弱点：即统计语言模型由于没有考虑句子的语义信息，有时会给出一些不知所云甚至是令人啼笑皆非的结果。

解决这个问题的可行方法是引入语义信息，目前在这个方面有人做了一些初步工作。比如基于概率的上下文无关文法 PCFGs(Probabilistic Context Free Grammars)方法[98-101]等。这些方法在小领域内的应用取得一定的效果，因为小领域的语义规则简单而有限，可以比较方便的和基于*n*-Gram 的统计语言模型结合起来。对于大词表和大领域的应用，这种方法效果取决于收集的语法是否足够多，而且语法信息和 *n*-Gram 模型之间如何结合、以及打分权重如何确定就显得相当重要。因为语言模型参数众多，如果处理不好，新信息的引入可能会改进部分错误，但也有可能将原先的正确部分处理错误。打个形象的比喻，处理不好有可能产生“按下葫芦起了瓢”的效果。由于中文是一种表意语言，语法很少受限，造成语言更加随意，因此，使用 PCFGs 方法对语法规则的数量和完整性需求更大。在目前条件下，大词表和大领域的语义详细标注信息还难以获得。

要解决这一问题，需要在以下几个方面努力：

(1)加强语料库标注的建设。目前人们所使用的语料库，大多是纯文本语料库，除了使用词表进行分词外，没有进行额外的处理。如果语料库的标注能够进一步细化，对统计语言模型的建模有很大好处；

(2)充分利用语言学家的研究成果，特别是语言学家在词网(Word-Net)[102, 103]方面的一些研究成果，加强中文词网的建设和资源共享。所谓词网，就是将单词之间的概念关系一一表示出来，连成的网状结构。目前英文等语言的词网建设已经取得较好的成果。如，给定一个英文单词apple，词网能给出非常复杂的单词关系网。比如能给出同类词(Cordidate Terms)，结果有：

citrus(柑橘类果实)

apricot(杏)

peach(桃)

plum(李子)

pineapple(菠萝)

banana(香蕉)

……

该同类词达到上百个。除此之外，还可以给出其他很多信息，如包含apple 的上层概念有：

可食用果实(edible fruit)

果实(fruit)

植物器官(plant organ)

自然物品(natural object)

……

这种词之间的关系对语言模型的建模有非常大的指导作用。例如，在本文第二章的实验中，使用 200M 汉字新闻文本训练模型指导解码时，发生如下错误：

正确：她说目前经济条件不好，已经有【耳环】，就不要买项链了。

错误：她说目前经济条件不好，已经有【二环】，就不要买项链了。

因为训练语料由新闻文本组成，文本较多地涉及到各地建设二环的问题，如“北京已经有二环、三环、四环和五环路，现在正筹建六环”，这种语料训练的模型无法避免这种错误。如果中文词网能给出“她”、“耳环”和“项链”之间紧密的关系，则可以避免该错误的发生。

但目前中文词网的建设还没有充分开展起来，可使用的资源非常少，阻碍了其在统计语言模型中的应用。

(3)为了充分运用前面提到的语义信息以及词网，现有的建模方式还需要改进，需要解决 *n*-Gram 模型和其他信息如何结合的问题。

2) 提升语言模型建模的粒度

目前语言模型的建模技术一般都是通过分解，将一个句子的概率通过词之间的预测概率获得。对于某些语言，如英语，人称和动词之间的一致性一般很难通过相邻单词之间的预测概率获得。对于中文来说，一些语义相关的信息用这种方式也无法解决。比如第三人称“他”、“她”、“它”之间的区分，除非知道上下文更多的信息，单靠语音识别很难知道究竟哪个才是正确的选择。

要解决这个问题，只有从更高的层次来描述语言模型，比如从句子级、段落级、甚至是篇章级对语言模型建模[104-106]，充分利用上下文信息，用全局的观点来描述模型。

当然，在这个建模过程中，知识的获取非常重要。比如“它”和“狗”之间的关系比“它”和“工程师”之间的关系要密切的多。给定一个特定的领域，如何获取并描述该领域内概念之间的关系是急待解决的问题。

3) 如何降低语言模型的维数

以 trigram 模型为例，如果词表大小为$|V|$，则语言模型参数空间的维数是$|V|$。给定历史 h，能得到在$|V|$空间内的一个向量。但整个语言模型的参数向量在其空间内分布非常稀疏(数据稀疏性)。这种数据稀疏性导致了空间向量的聚类更加困难。如何有效的降低维数？依靠数据驱动？依靠人的干预？还需要专家知识？

如果以上这些问题能在一定程度上得到解决，则语言模型研究必上一个新的台阶。

参考文献

[1] Huang X D, Acero A, Hon H W, et al. Spoken Language Processing: A Guide to Theory, Algorithm and System Development. New Jersey: Prentice Hall, 2001.

[2] Chomsky N. Aspects of the Theory of Syntax. Cambridge: MIT Press, 1965.

[3] Chomsky N. Syntactic Structures. Cambridge: MIT Press, 1965.

[4] Zheng F, Song Z J, Xu M X, et al. EasyTalk: A Large-Vocabulary Speaker-Independent Chinese Dictation Machine. In: Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech). Budapest, Hungary, 1999, 2: 819~822.

[5] Rabiner L, Juang B H. Fundamentals of Speech Recognition. New Jersey: Prentice Hall, 1993.

[6] Song Z J, Zheng F, Wu W H. Statistical Knowledge Based Frame Synchronous Search Strategies in Continuous Speech Recognition. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Istanbul, Turkey, 2000, 3: 1583~1586.

[7] 中华人民共和国国家标准 GB2312-80. 信息交换用汉字编码字符集基本集. 1981.

[8] 郑方，牟晓隆，徐明星，等. 汉语语音听写机技术的研究与实现. 软件学报，1999, 10(4): 436~444.

[9] Jelinek F, Mercer R L. Interpolated Estimation of Markov Source Parameters from Sparse Data. Pattern Recognition in Practice. E. S. Gelsema and L. N. Kanal, Eds. Amsterdam: North-Holland, 1980.

[10] Bahl L R, et al. A Tree-Based Statistical Language Model for Natural Language Speech Recognition. IEEE Transaction on Acoustics, Speech, and Signal Processing, 1989, 37(7): 1001~1008.

[11] Rosenfeld R. Adaptive Statistical Language Modeling: A Maximum Entropy Approach. Ph.D. Thesis in School of Computer Science. Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[12] Cerf-Danon H, El-Beze M. Three Different Probabilistic Language Models: Comparison and Combination. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Toronto, Canada, 1991, 297~300.

[13] Church K. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In:

Proceedings of 2nd Conference on Applied Natural Language Processing. Austin, Texas, 1988, 136~143.

[14] El-Beze M, Derouault A M. A Morphological Model for Large Vocabulary Speech Recognition. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Albuquerque, NM, USA, 1990, 577~580.

[15] Maltese G, Mancini F. An Automatic Technique to Include Grammatical and Morphological Information in a Trigram-based Statistical Language Model. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). San Francisco, CA, USA, 1992, 157~160.

[16] Steinbiss V, et al. A 10,000-word Continuous Speech Recognition System. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Albuquerque, NM, USA, 1990, 57~60.

[17] Niesler T R, Woodland P C. Variable-length Category-based N-grams for Language Modeling. In: Technical Report, Cambridge University, UK, 1995.

[18] Shannon C E. Prediction and Entropy of Printed English. Bell System Technical Journal, 1951, 50~62.

[19] Lafferty J D, Sleator D, Temperley D. Grammatical Trigrams: a Probabilistic Model of Link Grammar. In: Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language. Cambridge, MA, 1992, 89~97.

[20] Ye Z X, Berger T. Infromation Measures for Discrete Random Fields. Science Press, Beijing and New York. 1998.

[21] 徐通锵. 语言论--语义型语言的结构原理和研究方法. 长春市：东北师范大学出版社, 1997.

[22] Rosenfeld R, et al. Error Analysis and Disfluency Modeling in the Switchboard Domain. In: Proceedings of the 4th International Conference on Speech and Language Processing(ICSLP). Philadelphia, PA, USA, 1996.

[23] Good I J. The Population Frequencies of Species and the Estimation of Population Parameters. Biometrika, 1953, 40(3and): 237~264.

[24] Church K W, Gale W A. A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams. Computer Speech and Language, 1991, (5): 19~54.

[25] Katz S M. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. IEEE Transaction on Acoustic, Speech and Signal Processing, 1987, 35(3): 400~401.

[26] Lidstone G J. Note on the General Case of the Bayes-Laplace Formula for Inductive or a Posteriori Probabilities. Transactions of the Faculty of Actuaries, 1920, 8: 182~192.

[27] Johnson W E. Probability: deductive and inductive problems. Mind, 1932, 41: 421~423.

[28] Jeffreys H. Theory of Probability. Oxford: Clarendon Press, second edition, 1948.

[29] Brown, Peter F, Stephen A, et al. An Estimate of an Upper Bound for the Entropy of English. Computational Linguistics, 1992, 18(1):31~40.

[30] Baum L E. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. Inequalities, 1972, 3: 1~8.

[31] Kneser R, Ney H. Improved Backing-off for M-gram Language Modeling. In Proceedings of the IEEE Intenational Conference on Acoustics, Speech and Signal Processing. Detroit, MI, USA, 1995, 181~84.

[32] Chen S, Goodman J. An Empirical Study of Smoothing Techniques for Language Modeling. Technical report TR-10-98, Harvard University, 1998.

[33] Rosenfeld R. Two Decades of Statistical Language Modeling: Where Do We Go from Here? Proceedings of the IEEE, 2000, 88(8): 1270~1278.

[34] Ney H, Essen U, Kneser R. On Structuring Probabilistic Dependences in Stochastic Language Modeling. Computer, Speech & Language, 1994, 8: 1~38.

[35] WU J, Zheng F. On Enhancing Katz-smoothing Based on Back-off Language Model. In: Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP). Beijing China, 2000, 1: 198~201.

[36] 刘挺. 语言模型：[网上讲义,http://ir.hit.edu.cn/download/NLP_3.pdf], 哈尔滨：工业大学信息检索实验室. 2004.

[37] Wu G Q, Zheng F, Wu W H, et al. Improved Katz Smoothing for Language Modeling in Speech Recognition. In: Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP). Colorado, USA, 2002, 925~928.

[38] Witten I. H., Bell T. C. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transaction on Information Theory, 1991, 37: 1085~1094.

[39] Piantanida J. P., Estienne C. F. Maximum Entropy Good-Turing Estimator for Language Modeling. In: Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech). Geneva, Switzerland, 2003, 2277~2280.

[40] 林元烈. 应用随机过程. 北京：清华大学出版社, 2002.

[41] 武健. 汉语语音识别中统计语言模型的构建及其应用: [硕士学位论文], 北京: 清华大学. 2000.

[42] Kuhn R, Mori R D. A Cache-Based Natural Language Model for Speech Recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence, 1990, g:570~582.

[43] Jelinek F, et al. A Dynamic Language Model for Speech Recognition. In: Proceedings of the DARPA Speech and Natural Language Workshop. Asilomar, CA, USA, 1991.

[44] Bellegarda J. A Latent Semantic Analysis Framework for Large-Span Language Modeling. In: Proceedings of the 4th European Conference on Speech Communication and Technology (EuroSpeech). Rhodes, Greece, 1997, 1451~1454.

[45] Iyer R, Ostendorf M, Rohlicek J R. Language Modeling with Sentence-Level Mixtures. In: Proceedings of the ARPA Human Language Technology Workshop. Plainsboro, NJ, USA, 1994, 82~86.

[46] Mahajan M D, Beeferman, Huang X D. Improved Topic-Dependent Language Modeling Using Information Retrieval Techniques. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Phoenix, USA, 1999, 541~544.

[47] Chen S F, Seymore K, Rosenfeld R. Topic Adaptation for Language Modeling Using Unnormalized Exponential Models. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Phoenix, USA, 1998, 2: 681~684.

[48] Khudanpur S, Wu J. A Maximum Entropy Language Model Integrating n-grams and Topic Dependencies for Conversational Speech Recognition. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. Phoenix, AZ, USA, 1999.

[49] Galescu L, Ringger E K, Allen J F. Rapid Language Model Development for New Task Domains. In: Proceedings of the ELRA First International Conference. on Language Resources and Evaluation (LREC). Granada, Spain, 1998.

[50] Salton G, McGill M J. Introduction to Modern Information Retrieval. New York:

McGraw-Hill. 1983.

[51] Rosenfeld R. A Maximum Entropy Approach to Adaptive Statistical Language Model. Computer Speech & Language, 1996, 10: 187~228.

[52] Darroch J N, Ratcliff D. Generalized Iterative Scaling for Log-Linear Models. The Annals of Mathematical Statistics, 1972, 43(5): 1470~1480.

[53] Berger A, Della Pietra S, Della Pietra V. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 1996, 22(1): 39~71.

[54] Jelinek F. Statistical Methods for Speech Recognition. Cambridge, MA: MIT Press, 1998.

[55] Lafferty J D, Suhm B. Cluster Expansions and Iterative Scaling for Maximum Entropy Language Models. Maximum Entropy and Bayesian Methods. K. Hanson and R. Silver eds, Kluwer Academic Publishers, 1995.

[56] Lau R, Rosenfeld R, Roukos S. Trigger-Based Language Models: A Maximum Entropy Approach. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP). Minneapolis, MN, USA, 1993, 108~113.

[57] Pietra S A D, et al. Adaptive Language Model Estimation Using Minimum Discrimination Estimation. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). San Francisco, CA, 1992, 633~636.

[58] Ratnaparkhi A, Roukos S, Ward R T. A Maximum Entropy Model for Parsing. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Yokohama, Japan, 1994, 803~806.

[59] Wu G Q, Zheng F, Jin L, et al. An Online Inremental Language Model Adaptation Method. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EuroSpeech). Aalborg, Denmark, 2001, 2139~2142.

[60] Federico M. Bayesian Estimation Methods for N-gram Language Model Adaptation. In: Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP). Philadelphia, PA, USA, 1996, 1: 180~183.

[61] Reichl W. Language Model Adaptation Using Minimum Discrimination Information. In: Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech), 1999, 4: 1791~1794.

[62] Rao P S, Dharaniprada S, Roukos S. MDI Adaptation of Language Models across

Corpora. In: Proceedings of the 4th European Conference on Speech Communication and Technology (EuroSpeech), 1997, 1979~1982.

[63] Federico M. Efficient Language Model Adaptation through MDI Estimation. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EuroSpeech). Aalborg, Denmark, 2001, 4:1583~1586.

[64] Masataki H, Sagisaka Y, Kawahara T. Task Adaptation Using MAP Estimation in N-gram Language Modeling. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Phoenix, USA, 1997, 783~786.

[65] Santner T J, Duffy D E. The Statistical Analysis of Discrete Data. New York: Springer Verlag. 1989.

[66] Berger J. Statistical Decision Theory and Bayesian Analysis, New York: Springer-Verlag. 1985.

[67] http://www.oursci.org/magazine/200204/020410.htm.

[68] Nie J Y, Hannan M L, Jin W Y. Unknown Word Detection and Segmentation of Chinese Using Statistical and Heuristic Knowledge. Communications of COLIPS, Singapore, 1995, 5(1 &2): 47~57.

[69] Sproat R, Shih C L. A Statistical Method for Finding Word Boundaries in Chinese Text. Computer Processing of Chinese and Oriental Languages, 4(4): 336~349.

[70] 黄萱菁, 吴立德, 等. 基于机器学习的无需人工编制词典的切词系统. 模式识别与人工智能, 1996, 4: 297~303.

[71] http://www.iteer.net/modules/newschina/article.php?storyid=162.

[72] http://www.qualcomm.com/brew/images/developer/resources/ms/pdf/developer_market_research_americas.pdf.

[73] Di S, Zhang L, Chen Z, et al. N-gram Language Model Compression Using Scalar Quantization and Incremental Coding. In: Proceedings of the 2nd International Symposium on Chinese Spoken Language Processing. Beijing, China, 2000.

[74] Jelinek F. Self-organized Language Modeling for Speech Recognition. Readings in Speech Recognition. San Mateo, CA: Morgan Kaufmann, 1991, 450~506.

[75] Stolcke A. Entropy-based Pruning of Backoff Language Models. In: Proceedings of DARPA News Transcription and Understanding Workshop. Lansdowne, VA, USA, 1998, 270~274.

[76] Goodman J. Language Model Size Reduction by Pruning and Clustering. In: Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP). Beijing China, 2000, 3: 110~113.

[77] Zheng F, Wu J, Song Z J. Improving the Syllable-synchronous Network Search Algorithm for Word Decoding in Continuous Chinese Speech Recognition. Journal of Computer Science & Technology, 2000, 15(5): 461~471.

[78] Seymore K, Rosenfeld R. Scalable Back-off Language Models. In: Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP). Philadelphia, 1996, 1: 232~235.

[79] Yan P J, Zheng F, Xu M X, et al. Word-class Stochastic Model in a Spoken Language Dialogue System. In: Proceedings of the 3th International Symposium on Chinese Spoken Language Processing, Beijing, 2000, 141~144.

[80] Zipf G K. Selective Studies and the Principle of Relative Frequency in Language. Cambridge, MA: Harvard University Press, 1932.

[81] Wu G Q, Zheng F. A Method to Build a Super Small but Practically Accurate Language Model for Handheld Devices. J. Computer Science & Technology, 2003, 18(6): 747~755.

[82] 中华人民共和国国家标准 GB13000.1. 信息技术多八位编码字符（UCS). 1995.

[83] 中华人民共和国国际标准 GB18030. 信息交换用汉字编码字符集基本集的扩充. 2000.

[84] 王轩，王晓龙. 大规模文本计算机音字转换技术的研究. 计算机研究与发展，1998, 35(9): 417~421.

[85] 王晓龙，王开铸. 声音语句输入的研究. 计算机学报, 1994, 17(2):96~103.

[86] Jin L, Wu G Q, Zheng F, et al. Improved Strategies for Intelligent Sentence Input Method Engine System. In: Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP). Beijing China, 2000, 247~250.

[87] 万建成. 语音代码－汉字智能转换研究. 中文信息学报, 1994, 8(2): 61~71.

[88] 潘凌云，杨长生. 拼音、汉字计算机自动转换系统. 计算机学报，1990，13(4): 271~275.

[89] Zheng F, Wu J, Wu W H. Input Chinese Sentences Using Digits. In: Proceedings of the 2nd International Symposium of Chinese Spoken Language Processing (ISCSLP), Beijing, China, 2000, 3: 127~130.

[90] 张瑞强, 王作英, 张建平. 带拼音纠错的汉语音字转换技术. 清华大学学报(自然科学版), 1997, 37: 9~12.

[91] 关毅, 王晓龙. 基于转移的音字转换纠错规则获取技术. 计算机研究与发展, 1999, 36(3): 268~273.

[92] 倪小东, 李人厚, 余克艰. 适用于信息设备的汉字输入法研究. 中文信息学报, 2001, 15(5): 58~64.

[93] Zheng F. A Syllable-synchronous Network Search Algorithm for Word Decoding in Chinese Speech Recognition. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Phoenix, USA, 1999, 2: 601~604.

[94] 中华人民共和国国家语委规范 GF3001. GB13000.1 字符集汉字笔顺规范. 上海: 上海教育出版社, 1999.

[95] Zhang G L, Zheng F, Wu W H. A Two-Layer Lexical Tree Based Beam Search in Continuous Chinese Speech Recognition. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EuroSpeech). Aalborg, Denmark, 2001, 3: 1801~1804.

[96] 张国亮, 徐明星, 李净, 等. 语音识别中基于双层词法树的跨词搜索算法. 清华大学学报(自然科学版), 2003, 43(7): 981~984.

[97] Ortmanns S, Ney H, Eiden A. Language-Model Look-ahead for Large Vocabulary Speech Recognition. In: Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP). Philadelphia, USA, 1996. 2095-2098, 1995.

[98] Goodman J. Parsing Inside-Out. PhD Thesis in Computer Science. Cambridge: Harvard University, 1998.

[99] Magerman D M, Marcus M P. Pearl: A Probabilistic Chart Parser. In: Proceedings of the 4th DARPA Speech and Natural Language Workshop. Pacific Grove, California, 1991.

[100] Black E, et al. Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio, USA, 1993, 31~37.

[101] Hindle D, Rooth M. Structural Ambiguity and Lexical Relations. In: Proceedings of 3rd DARPA Speech and Natural Language Workshop, Hidden Valley, PA, 1990.

[102] Fellbaum C (editor). WordNet: An Electronic Lexical Database. Cambridge, MA: MIT

Press, 1998.

[103] http://www.cogsci.princeton.edu/~wn/.

[104] Chen S F, Rosenfeld R. Efficient Sampling and Feature Selection in Whole Sentence Maximum Entropy Language Models. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. Phoenix, AZ, USA, 1999.

[105] Zhu X J, Chen S F, Rosenfeld R. Linguistic Features for Whole Sentence Maximum Entropy Language Models. In: Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech), Budapest, Hungary, 1999.

[106] Rosenfeld R, Wasserman L, Cai C, et al. Interactive Feature Induction and Logistic Regression for Whole Sentence Exponential Language Models. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, Keystone, CO, USA, 1999.

致　谢

衷心感谢导师吴文虎教授和郑方副教授对本人的精心指导。两位导师不仅专业上给我很好的指导，而且其严谨求实的治学作风、平易近人的待人原则和忘我的工作精神都将是我长期学习的榜样。在此，谨向两位恩师致以最诚挚的谢意！

感谢语音技术中心的其它老师，包括方棣棠教授、李树青教授、徐明星老师以及实验室的全体同窗对我论文工作的帮助和支持。

感谢我的家人，他们无私的爱和默默的关怀，一直伴随着我的奋斗过程。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____日　期：_____

附录　博士就读期间完成的其它研究任务

1 嵌入式语言模型开发包的设计

基于本文第四章所述的语言模型压缩技术和嵌入式设备上的语言模型应用框架，本文实现了一个嵌入式语言模型 SDK 开发包，用于快速开发嵌入式设备上的语言模型应用。该开发包主要面向输入法开发，可以支持各种编码方式。其语言模型概率层的 API 可以用于其他应用如 OCR 和语音识别的开发。该开发包是国内第一个嵌入式系统上的语言模型应用开发包。

按功能划分，开发包分为预处理模块和运行模块两大部分。如图 F-1 所示，预处理模块用于语言模型的训练和压缩、音节树和词法树的建立、单词映射关系的建立等。运行模块可以分成三个层次，分别为词表层、概率模型层和解码层。

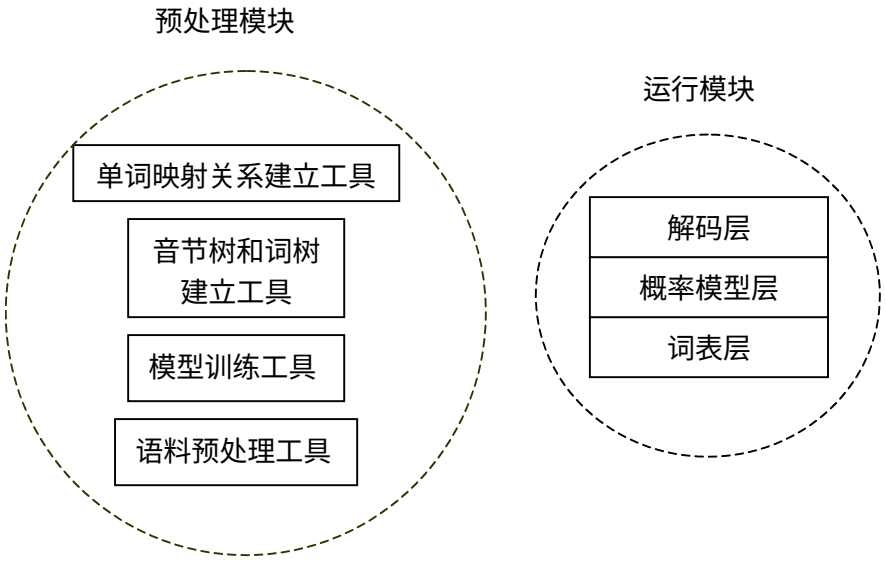

图 F-1 语言模型应用开发包结构图

词表含有 51,157 个汉语词汇，包括单字词、二字词、三字词和四字词。

词表的单字词部分为 GB2312-80 中规定的全部 6,763 个汉字，即国家一级和二级字库的所有汉字。词表中不同长度的词的分布如图 F-2 所示意。

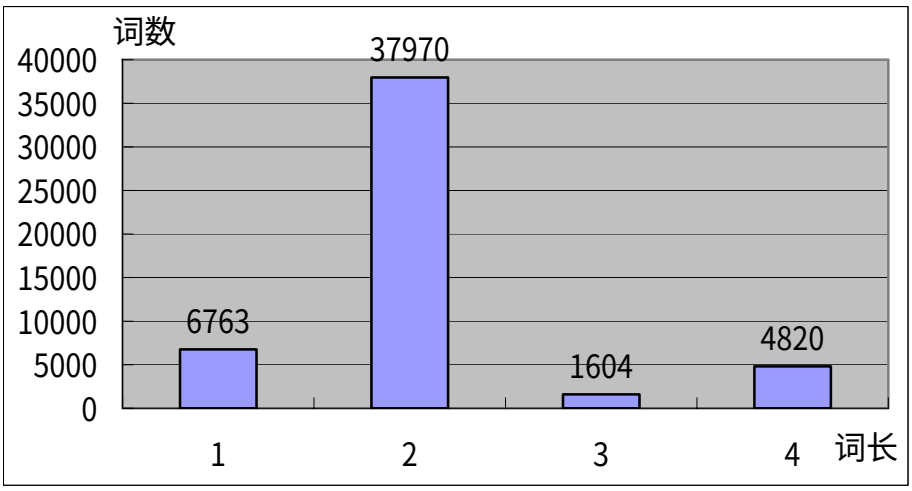

图 F-2　词长的分布

语言模型概率层函数

语言模型概率层函数非常简单，主要包括初始化函数和关闭函数，以及获得 n-gram 模型概率分值的函数。

解码层函数

解码层函数面向输入法开发。可以支持汉字的各种编码方式(比如拼音、笔划等)。在确定输入法的编码方式后，可以先使用预处理模块建立搜索结构，包括音节树和词树等，然后就可以方便的调用解码层函数用于整句输入法的开发。

解码层函数主要包括以下几个方面：

1. 打开或关闭解码器；
2. 进行一次递增式解码；
3. 获得解码的整句候选结果；
4. 获得音节切分结果；

5. 获得单个汉字的候选；

6. 音节局部匹配功能，比如输入 sh，给出“ 沙..上..设.. ”等候选；

7. 预测功能，比如最后一个字是“ 中 ” ，预测下一字可能为“ 国、间、标、意… ”等；

8. 限制解码功能，比如将路径中某位置限制为特定字，然后解码；

对于拼音编码，特别是 10 键键盘输入时，还有一些用于 10 键拼音和 26 键拼音之间转换的函数。

2 多个嵌入式平台上实现首个中文整句输入法

使用嵌入式语言模型 SDK 开发包，确定了输入法的编码方式后，可以非常方便的在嵌入式平台上开发出好用的整句输入法。本文实现的输入法主要基于标准拼音的整句输入法和基于汉字前 4 笔笔划(不足 4 笔的加结束码)的整句输入法。下面先介绍一下这种笔划编码方式。

2.1 适用于整句输入的笔划编码设计

汉字的结构非常复杂，字典中列出的汉字部首达 190 多种，可见直接用部首编码并不合适；另外，根据《信息处理 GB13000.1 字符汉字部件规范》(即 GF3001)的统计，对 GB13000.1 规定的 20902 个汉字进行拆分，发现基础部件(也称末级部件，是最小的不可拆分的部件)多达 560 个，可见使用基础部件编码输入汉字显然也很不现实。

本文定义如下的笔划编码规范：

1. 基本笔划(码元)构成。采用国家语委颁布的《GB13000.1 字符集汉字笔顺规范》(GF3002)，即汉字的基本笔形有五种，其排列顺序为一(横)、丨(竖)、丿(撇)、丶(点)、乛(折)，分别用编码 1、2、3、4、5 表示。

2. 每个汉字的笔顺采用 GF3002 中定义的汉字规范笔顺；

3. 为了简化输入，提高输入效率，每个汉字的编码采用其规范笔顺的

前 4 个笔划。比如“ 我 ”字，规范笔顺为 3121534，则本文采用的编码为 3121，即“ 丿一丨一 ”；

4. 对于不足 4 划的汉字，增加一个结束笔划 0。比如“ 二 ”字，其规范笔顺为 11，不足 4 划，因此本文采用的编码为 110。这种做法的目的是降低边界混淆。

实际上，对 GB2312 中定义的 6763 个汉字，如果考虑每个字的所有笔划，则每个“ 音节 ”的长度参差不齐，最短的是“ 一 ”字，只有一划，较长的如“ 藏 ”，其笔顺为 12213513125125534，达 17 划。这样导致构成的“ 音节 ”数非常多。而且，在给定一个包含多个音节的编码串时，相对拼音编码来说，该“ 笔划 ”编码的音节边界将可能发生非常多的混淆。因为对于拼音编码来说，每个音节由声母和韵母构成(零声母的音节非常少)，这种“ 声母－韵母 ”结构形成了音节间的天然边界；而对于笔划编码来说，每个笔划都可能是字的起始笔划。例如，编码串“ 一一一 ”可以对应汉字串“ 一一一 ”，也可以对应“ 一二 ”、“ 二一 ”、“ 三 ”等汉字串。

这种笔划输入法规范限制了每个字的编码长度最长为 4 划，而且对不足 4 划的汉字编码结尾补零，这样既避免了“ 音节 ”过多过长的问题，也避免了“ 音节 ”边界混淆问题，适用于整句输入。

这种全新的编码方式共构成的笔划“ 音节 ” 数目和拼音音节数目比较如表 F-1 所示。

表 F-1 4 笔笔划编码和拼音编码“ 音节 ”情况比较

	音节最大长度	音节平均长度	音节个数	音节平均重码率(%)
笔划	4	3.97	436	15．5
拼音	6	3.23	402	16．8

从表中可以看出，该编码方法和拼音编码方法有一定的类似性，如音节平均长度都是 3～4，音节个数都是 400 多个，音节平均重码率都是 16 左右。测试表明采用该编码方式的输入法和基于拼音编码输入法性能相近。

2.2 应用举例

输入法开发包是通用的，可以在平台上编译。当选定开发平台时，可以快速的使用这些模块开发出易学易用的输入法。本文选用了 Palm OS、Symbian(包括支持 26 键键盘和手机 10 键键盘两种版本)、PPC(Pocket PC)等操作系统平台，开发出了嵌入式设备上第一个(也是目前为止唯一一个)整句输入法。

PPC 平台和 Palm OS 平台

Palm OS 平台和 PPC 平台的界面相似。它们都没有键盘，输入法需要设计软键盘。本文设计整句输入法相对于现有输入法来说在界面布局上有些不同，主要是整句输入法需要一个额外的整句候选界面。以 PPC 平台的输入法设计为例说明，图 F-3 是界面基本布局。

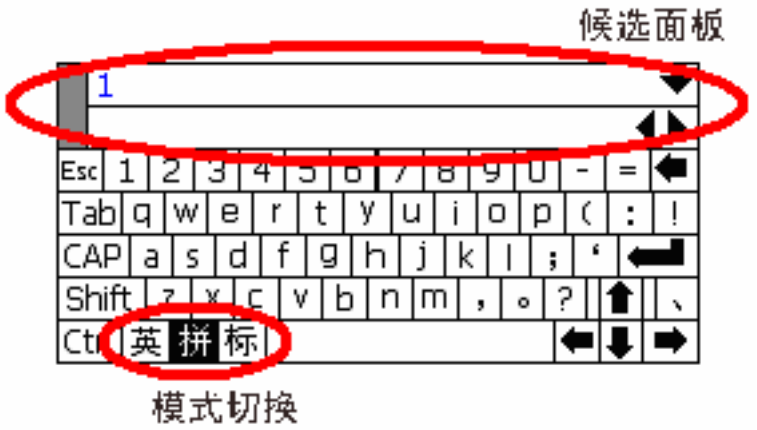

图 F-3 PPC 上的输入法界面

该界面提供了三个用于输入模式切换的键，“ 英 ”键用于切换到英文输入状态，“ 拼/双/写/笔 ”用于在不同的中文输入模式之间切换，“ 标/符 ”用于输入标点和特殊符号，总切换流程如图 F-4。

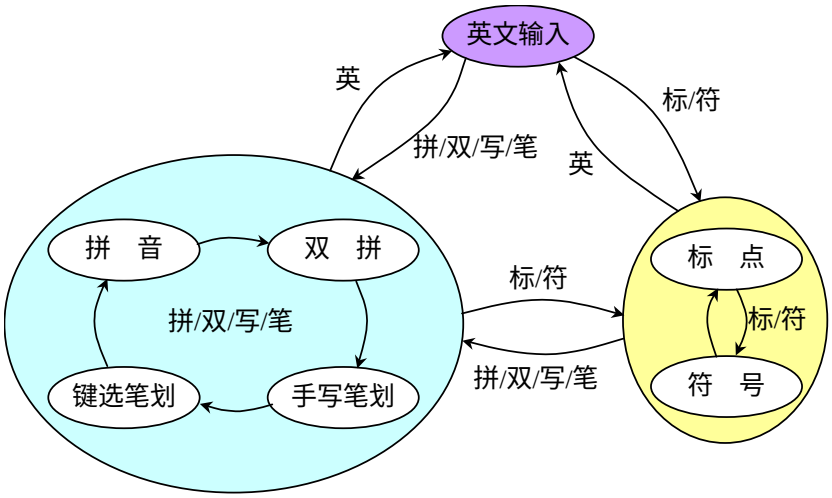

图 F-4 PPC 上的输入法切换流程

中文输入模式包括四种，拼音输入指的是逐字母输入拼音的方式，双拼则是按声韵方式输入(输入声母后自动给出合法的韵母键盘)，手写笔划指的是用手写板识别出“一丨丿丶乛”五个基本笔划代替按键输入笔划。而键选笔划则是用按键方式输入五个基本笔划，然后解码。每个输入方式都对应一个软键盘，这里没有一一列出。

该整句输入法提供了现有输入法的单字输入模式，即输入单字时，不会比现有的输入法复杂。同时，该输入法支持联想输入方式和整句输入方式的无缝自由切换。也就是说，用户在输入过程中，如果选定了最后一个字，则输入法预测出下一个字的候选列表(列表按照预测概率大小排序)；如果用户想输入的下一个字刚好在预测列表中，用户可以直接选定，输入法将继续进行后续的预测；如果用户发现想输入的下一字不在预测列表时，则可以直接输入下一个字的拼音(或笔划)，这时输入法就自动转入整句输入状态。这种自由切换方式在用户输入句子包含较多常用词时或者用户对下一字的拼音不太确定时非常有用。

本文为该输入法设计了整句候选功能(图 F-5)，也提供灵活的单字修正方式。另外，单字修改能自动对整句重新解码。当初始解码结果有多个错误时，用户选择一个错误字进行修正后，输入法将自动对整句进行重新解

码，从而实现将其它错误自动纠正的功能。当然，重新解码有时会在一遍解码正确的地方造成新的错误。测试表明，由于语言模型有较强的指导功能，这个重新码过程“捎带”纠正其他错误的概率要远远高于引入新错误的概率，从而从整体上提高了输入的效率。

图 F-5 PPC 上的输入法的整句候选模式

S90 平台 (诺基亚 9210)

Symbian 是智能手机上使用最为广泛的操作系统，S90 平台是 Symbian 操作系统的一个版本。S90 支持 26 键键盘和大屏幕，典型产品是诺基亚的 9210c 手机，因此本文选择诺基亚 9210c 为 26 键手机中文整句输入法的开发平台。

图 F-6 展示了整句输入和整句候选的情形。

图 F-6 诺基亚 9210c 上的输入法整句候选

同样，该平台上的输入法也支持联想输入方式和整句输入方式的无缝自由切换，同时带预测功能。

S60 平台(Nokia 6610/**7610)**

S60 平台是 Symbian 操作系统的另一版本。该版本支持数字键盘，即 10 键手机键盘，典型产品有诺基亚的 6610/7610 手机。本文选择 6610 为数字键盘手机中文整句输入法的开发。

S60 系统的键盘只有 10 个数字键可用于输入汉字，这样在输入拼音编码时无法实现一键输入一个拼音编码。为解决这个问题，需要在音节树层和语言模型映射层中间加入一个 10 键音节/26 键音节映射层，如图 F-7 所示。

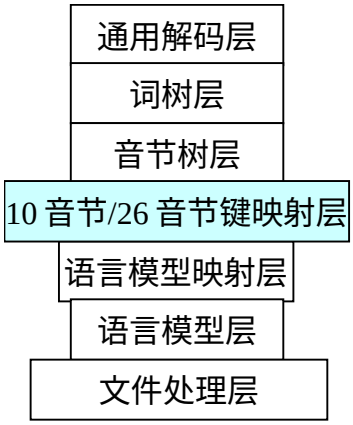

图 F-7 适用于 10 键手机平台的输入法分层结构设计

对于 26 键键盘而言，pan 是一个音节，而对于 10 键键盘来说，726 是一个 10 键音节，但是它对应了 pan/pao/ran/rao/san/sao 共六个 26 键音节。通过这个映射层，实现只需输入 10 键音节就完成整句输入的功能。

因为一个 10 键音节可能对应多个 26 键音节，而且 26 个字母在 10 键键盘上是简单的顺序分配，造成一些声母字母和韵母字母分配在同一数字键上，比如数字键 2 对应了 a/b/c 三个字母，a 是韵母字母，而 b/c 是声母字母，这样就破坏了汉语拼音音节中由“声母－韵母”构成的天然音节边界。比如“汽车”一词的拼音对应的 10 键编码为 74243，当用户输入该编

码串时，可以切分出 74/243、 742/43、7424/3 三种情况，其中 74/243 对应了 pi/che、qi/che、ri/che、si/che、pi/bie、qi/bie、ri/bie、si/bie 八种拼音串，742/43 对应了 qia/ge、sha/ge、qia/he、sha/he 四种拼音串，7424/3 则对应了 shai/e 一种拼音串。如果输入的 10 键编码串更长，则对应的合法拼音串可能更多。

为了解决该问题，输入法要求每输完一个音节后强制输入 0 键表示当前音节结束。比如，欲输入句子“我在清华西门等你”，应输入的编码串为“960924074640482094063603364064 0”。

手机上的输入法主要用于短信输入，因此使用短信语料训练语言模型有利于提高系统性能。本文用到的训练语料为 7.4M 的实际短信文本，并使用语言模型压缩算法做压缩处理。测试文本为诺基亚中国研究中心提供的 500 句短信文本，进行批量音字转换测试。该 500 句短信文本涉及学习、工作、交通、体育活动等各个方面，且未参与训练。所得结果如表 F-2。

表 F-2 Nokia 6600 上输入法性能

	首句单字正确率(%)	首句整句正确率(%)	前两句整句正确率(%)	前三句整句正确率(%)	前五句整句正确率(%)
26 键编码	95.5	80.0	88.8	92.4	93.8
10 键编码	91.1	65.8	75.8	81.6	85.0

表中的 26 键编码解码结果仅供参考。对于带 26 键键盘的手机和 PDA 来说，采用该输入方法可以获得前五句 93.8 的整句正确率，输入效果远远高于普通输入。

对于数字键盘手机而言，实际使用时采用的是 10 键编码。从表中可以看出，前 5 句的整句正确率达到了 85.0%，也就是说用户输入短信时基本不需要修改就可以整句输入，非常方便。

输入法提供了方便的修改接口以获得更多整句候选或者修改单字。而且，输入法也提供了能和整句输入模式无缝切换的联想功能。输入法界面和过程如图 F-8 所示。

图 F-8 诺基亚 6600 上的整句输入法应用

由于在输完每个音节后要求输入音节结束符 0，本文设计的这种输入方式表面看起来每个音节多输入了一个编码，增加了按键次数，但考虑当前手机上使用广泛的的 T9 输入法可以发现，用户在用 T9 输完每个拼音对应的数字串后需要选择正确的拼音，比如用户欲输入“然”字，需要先输入数字串 726，然后从所有的拼音候选(pan/pao/ran/rao/san/sao)选择 ran，然后才能选字，因此单从按键次数来看 T9 的输入效率并不高。

为了测试 10 键整句输入法的性能，随机选择了 12 个短信句子，采用手工输入方式，统计使用 T9 输入法和本文的整句输入法输入这些句子时按键次数。结果如表 F-3 所示。

其中 A1 和 A2 是使用 T9 输入法的结果。A1 是使用 T9 输入法逐字输入的情形，也就是说不使用其联想功能；A2 是使用 T9 输入法，而且在输入过程中如果发现后一字在联想字列表的第一屏中(因为如果不在联想字第一屏中，则使用联想字需要翻屏，反而会多按键)，则使用联想功能的情形。因此，可以认为 A2 的结果是使用 T9 输入法输入时需要按键次数的下限，即 T9 输入法输入性能的极限。B1 和 B2 是使用整句输入法的结果。B1 是使用整句输入模式按照每个字对应的 10 键编码逐字输入，输入完毕

后修改错误(如果有错误)并送入目标编辑区所需按键次数； B2 的结果是整句输入法在整句输入过程中结合联想功能时的按键次数(因此 B2 称为混合模式)。

表 F-3 使用 T9 和 10 键整句输入法按键次数比较

句子	A1 T9(逐字)	A2 T9(联想)	B1 整句模式	B2 混合模式
你睡了吗	20	20	15	15
我说你命好	26	26	24	24
你感觉到哪去了	33	30	30	29
好吧	10	10	8	8
这么晚了你睡吧	35	34	30	30
不然我该心疼了	35	35	30	25
放心	12	8	10	9
吻你	10	10	8	8
五一你回家吗	29	26	21	20
冰月	12	12	17	13
这就是你小妹的名字	46	35	36	35
明白了吗	21	17	16	15
(总共按键数)	289	263	245	231

表 F-4 给出了按键减少的百分比。

A1 和 B1 是各自没有使用联想功能时的按键次数，A2 和 B2 则是各自使用了联想功能时的按键次数。因此 B1 相对 A1、B2 相对 A2 的比较是最有意义的。从表 F-4 可以看到，B1 相对 A1、B2 相对 A2 的按键减少率都很明显。甚至整句输入法在没有使用联想功能时的按键次数也比 T9 输入法在使用联想功能时的按键次数减少了近 7%。

表 F-4 整句输入法相对 T9 输入法的按键减少率

	B1 按键减少率	B2 按键减少率
A1	15.2%	20.1%
A2	6.8%	12.2%

个人简历、在学期间的研究成果及发表的论文

个人简历

吴根清，1976 年 10 月 24 日出生于浙江庆元县，95 年 9 月考入清华大学计算机科学与技术系学习，1999 年 9 月免试保送清华大学计算机科学与技术系攻读计算机科学与技术学科工学博士学位至今。

在国际和国内学术刊物上发表的论文(第一、二作者)

[1] Wu G Q, Zheng F. A Method to Build A Super Small but Pracatically Accurate Language Model for Handheld Devices. Journal of Computer Science and Technology. 2003, 18(6): 747-755.

[2] Wu G Q, Zheng F. Reducing Language Model Size by Importance-based Pruning and rank-based Quantization. In: Proceedings of Oriental-COCOSDA. Sentosa, Singapore. 2003, 156-159.

[3] Wu G Q, Zheng F, Wu W H, et al., Improved Katz Smoothing for Language Modeling In Speech Recognition. In: Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP). Denver, Colorado, USA, 2002. 2:925-928.

[4] Wu G Q, Zheng F, Wu W H. A Compression Method Used in Language Modeling for Handheld Devices. In: Proceedings of the 3th International Symposium on Chinese Spoken Language Processing (ISCSLP). Taipei, 2002. 339~342.

[5] 吴根清，郑方，金凌，吴文虎. 一种在线递增式语言模型自适应方法. 中文信息学报, 2002，16 (1): 60-65. (NCMMSC6 会议获奖论文，被期刊选中发表).

[6] 吴根清，郑方，金凌，吴文虎. 一种在线递增式语言模型自适应方法. 第六届全国人机语音通讯学术会议(NCMMSC6)，深圳，2001, 51-54 (获得优

秀论文三等奖)

[7] Wu G Q, Zheng F, Jin L, et al. An Online Inremental Language Model Adaptation Method. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EuroSpeech). Aalborg, Denmark, 2001. 2139~2142.

[8] Jin L, Wu G Q, Zheng F, et al. Improved Strategies for Intelligent Sentence Input Engine System. In: Proceedings of the 3th International Symposium on Chinese Spoken Language Processing (ISCSLP). Beijing, 2000. 247~250.