密级：秘密

汉语语音识别中统计语言模型的构建及其应用

Implementation and Application of Statistical Language Model in Mandarin Speech Recognition

(申请清华大学工学硕士学位论文)

院（系、所）：计算机科学与技术系

专　　业：计算机应用

研　究　生：武健

指 导 教 师：郑方　　（职称）副教授

2000 年 6 月

摘 要

语音识别的目的之一是将人的语流自动转化为相应的文本信息，在自动转化的过程中，语言模型起着重要的作用。本文对汉语连续语音识别涉及到的一些语言模型构建和使用的关键技术进行了较深入的研究，主要讨论了统计语言模型 *n-gram* 的理论基础及其实现技术，以及在语音识别和相关系统中的应用。具体包括：

1、提出了复合的机器词典构造流程，该流程应用了基础词条的分级选择策略，基于合并的新词发现算法，以及最大似然的同步分词算法；构成了汉语语料库的自动处理平台；生成了一个 5 万词的语音识别用汉语机器词典 *EasyDic99*。

2、提出了针对图灵估计退化模型的修正方案，从防止折扣系数计算出现溢出，减少模型存储，以及降低概率估计失真等三个方面解决了基于图灵估计的退化模型在实际应用中的问题；实现了一个基于两亿字语料库的 *trigram* 语言模型。

3、在音节同步算法的基础上，提出了基于多活动栈的音节同步算法，用于完成汉语音字的转换。该算法更合理地利用了同步算法的优势，减少了预测的工作量；与单栈的音节同步算法相比，它在一定程度上结合了广度优先算法和深度优先算法的特点，能够更加准确地实现搜索目标。

4、在基于词的同步搜索算法基础之上，提出了综合同步搜索算法，用于连续语音识别中的路径搜索。该算法应用了分阶段的联合预测技术，搜索时有机地结合声学模型和语言模型的预测技术。此外，还提出了基于语言模型的分级剪枝策略，并在此基础之上构造了多层的剪枝算法。

5、根据基于多活动栈的音节同步算法，实现了汉语智能整句输入法的原型 *EasyConv99*。

6、根据综合同步搜索算法，实现了汉语语音听写机的原型系统 *EasyTalk2000*。

关键词： N-gram 语言模型，机器词典，修正退化平滑算法，基于多活动栈的音节同步算法，综合同步搜索算法

Abstract

The language model plays an important role in the process of speech recognition, which is used to transform human speech into corresponding text information. In this paper, some key techniques for language model in Mandarin continuous speech recognition have been studied. Two main aspects are focused on: the theory and technology involved in the implementation of the statistical language model; the utilization of statistical language model in speech recognition and related system. In detail, the followings are included:

1. A complex strategy for constructing machine dictionary, including a hierarchical-selecting strategy for basic word items, a merging based new word discovery algorithm and a maximizing likelihood based synchronous algorithm for word boundary detecting, is proposed. And then an automatic processing system for Mandarin corpus and a machine dictionary, namely *EasyDic99*, are established.
2. An enhanced version of Katz-smoothing based back-off model is proposed to amend the computation of the discounting coefficients, decrease the storage space of n-gram, and reduce the estimating distortion of the back-off probabilities, so that an applicable *trigram* can be achieved.
3. A Multi-Active-Stack based Syllable Synchronous Search (MS-SSS) algorithm for Mandarin pinyin-to-character conversion is presented, in which some characteristics of the width-first algorithm and the depth-first algorithm are combined to improve the efficiency of the search.
4. An Integrated Synchronous Search (ISS) algorithm based on the word-conditioned beam search is proposed and applied in the continuous speech recognition. A Stage based Look Ahead (SLA) technique and a Language Model Rank based Pruning (LMRP) strategy is utilized in this algorithm.
5. A prototype of Mandarin intelligent keyboard input method based on MS-SSS, namely *EasyConv99*, is implemented.
6. A prototype of Chinese Dictation Machine (CDM) based on ISS, namely *EasyTalk2000*, is implemented.

Keywords: N-gram Language Model, Machine Dictionary, Modified Katz-smoothing Algorithm, Multi-Active-Stack based Syllable

Synchronous Search Algorithm, Integrated Synchronous Search Algorithm

目录

图表索引

第一章　前言

语言是人们用来进行信息传递和相互交流的一种信息载体，是人所特有的一种高级智能。上古时代，各部族的人们用简单的符号记忆他们的生活经历，用一些相互可理解的发音进行交流。随着时代的变迁，这些记忆符号逐渐发展，形成了文字，并与人们之间最密切的交流方式－语音结合在一起，组成了语言。

1.1 语言处理

语言处理实际上就是对语言中孕育着的信息进行提取和加工处理的过程，它是一门与人工智能、概率计算、语音学、信息论、认知心理学相关的多学科交叉的综合研究课题。分析自然语言的构成可以发现，语言是感知知识(*Perception Knowledge*)和认知知识(*Cognition Knowledge*)的综合体，认知是一种能够形式化的知识，能用语言描述的；而感知的并不一定都能用语言表达。而目前的计算机只能处理形式化了的知识，那么语言处理的任务就是寻找和发现这些感性知识到认知知识的转换或映射机制，并用形式化的语言模型加以表述。

语言处理是语音识别过程的一个重要组成部分。实际上，当我们听到一个具体话题的时候，可能会漏掉对方所说的几个词，甚至是整个句子都没有听清，但我们往往总能通过各方面的知识，包括上下文信息和具体语境等，弥补上我们漏听的词，从而正确地理解对方所说的内容。这就告诉我们，在人进行语音识别的时候，不仅利用了声学信息，而且还使用了许多非声学的知识，如词法、语法、语义等，在很大程度上，人们正是利用这些高层次的知识来获得对语音的正确识别和理解。

1.2 语言模型在语音识别系统中的作用

一个大字表连续语音识别系统（即听写机）应该由两部分组成：声学模型和语言模型。二者之间的关系可以用图 1.1 来描述：假设声学模型的输入（即语音信号）为 A，它的输出（也是语言模型的输入）为 w，则整个语音识别系统的任务就是找到一个 w^*，使它满足：

$$w^* = \arg\max_w P(w|A) = \arg\max_w \frac{P(A|w)P(w)}{P(A)} \tag{1.2.1}$$

其中第二步是根据 Bayes 法则得到的。在这个式子中，$P(A|w)$是由声学模型计算得出的匹配概率，$P(A)$是一个常数值，而 $P(w)$是由语言模型给出的。容易看到，没有语言模型的识别系统，实质上是认为，对每个可能的句子 $P(w)$都是一样的，这种假设显然不尽合理。

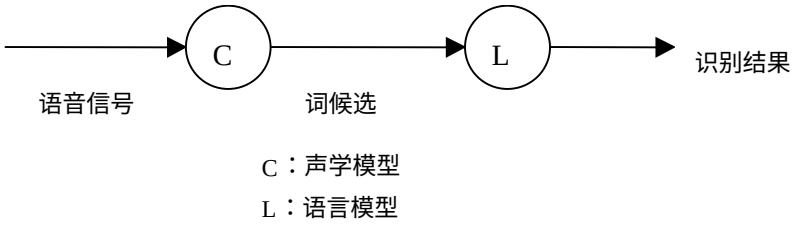

图 1.1　语音识别器的组成

1.3 确定型语言模型和统计语言模型

根据对语言分析处理的方法不同，用于语言处理尤其是用于语音识别中的语言模型主要分为两种：**确定性语言模型**（*Deterministic Language Model*），也称**句法语言模型**（*Syntactic Language Model*）或**形式语言模型**（*Formal Language Model*），和**统计语言模型**（*Statistical Language Model* 或 *Stochastic Language Model*）。确定性语言模型是建立在形式语言理论的基础上的，它用一个先验的形式语法来描述语言的内在结构，以判别句子的下一个语言单位（*Linguistic Unit*）。而统计语言模型是用概率统计的方法来揭示语言单位内在的统计规律。

1.3.1 确定型语言模型

语言中有成千上万的词汇，但词汇的任意组合并不是语言，它们只有依照语法规则组合起来的时候，才能构成语言。语法是语言的构架，是描述语言的语法结构的形式规则（即语法规则）。这些规则必须准确，而且应有相当强的描述能力，足以描述各种不同结构。语法具有抽象性，它的各种规则是从无数的

具体句子中抽象出来的。具体的句子是无限的，而支配句子的组合规则是有限的。

1.3.1.1 确定性语言模型简介

形式语言（*Formal Language*）和自动机理论是 *Chomsky* 提出的，它认为一个文法 G 是由四部分组成的：一组终结符号，一组非终结符号，一个开始符号，以及，一组产生式。

所谓终结符号乃是组成语言的基本符号，从语法分析的角度来看，终结符号是一个语言的不可再分的基本符号。

非终结符号（也称语法变量）用来代表语法范畴。一个非终结符号代表一个确定的语法概念。因此，一个非终结符是一个类（或集合）记号，而不是一个个体记号。例如，“ 算术表达式 ” 这个非终结符乃代表一定算术式组成的类。因而，也可以说，每个非终结符号乃表示一定符号串的集合（由终结符号和非终结符号组成的符号串）。

开始符号是一个特殊的非终结符号，它代表语言中我们最终感兴趣的语法范畴。这个语法范畴通常称为 “ 句子 ”。

产生式（也称产生规则或简称规则）是定义语法范畴的一种重写规则。一个产生式的形式是：

$$A \to \alpha$$

其中，箭头（有时也用::=）左边的 A 是一个非终结符，称为产生式的左部符号；箭头右边的 α 是由终结符号或/与非终结符号组成的一符号串，称为产生式的右部。

形式上说，一个文法 G 是一个四元式（V_T，V_N，S，P），其中

V_T 是一个非空有限集，它的元素称为终结符号；

V_N 是一个非空有限集，它的元素称为非终结符号，$V_T \cap V_N = \phi$ ；

S 称为开始符号，是一个非终结符号；

P 是一个产生式集合（有限），每个产生式的形式是 $P \to \alpha$ ，其中，$P \in V_N$ ，$\alpha \in (V_T \cup V_N)^*$ 。开始符号 S 至少必须在某个产生式的左部出现一次。

假定 G 是一个文法，S 是它的开始符号。如果 $S \overset{*}{\Rightarrow} \alpha$，则称 α 是一个句型。仅含终结符号的句型是一个句子。文法 G 所产生的句子的全体是一个语言，将它记为 $L(G)$。

$$L(G) = \{\alpha \mid S \overset{+}{\Rightarrow} \alpha \,\&\, \alpha \in V_T^*\}$$

按照对产生式限制条件的不同，文法分为四种类型：0 型、1 型、2 型和 3 型，它们描述语言的能力依次降低。

我们说 $G=（V_T，V_N，S，P）$ 是一个 0 型文法，如果它的每个产生式

$$\alpha \to \beta$$

是这样一种结构：$\alpha \in (V_T \cup V_N)^*$ 且至少含有一个非终结符，而 $\beta \in (V_T \cup V_N)^*$。

0 型文法也称短语文法。0 型文法的能力相当于图灵机（*Turing Machine*）。或者说，任何 0 型语言都是递归可枚举的；反之，递归可枚举集必定是一个 0 型语言。

如果对 0 型文法分别加上以下的第 i 条限制，我们就得到 i 型文法：

1. G 的任何产生式 $\alpha \to \beta$ 均满足 $|\alpha| < |\beta|$（$|\alpha|$ 指符号串 α 的长度）；仅仅 $S \to \varepsilon$ 例外，但 S 不得出现在任何产生式的右部。
2. G 的任何产生式为 $A \to \beta$，其中 $A \in V_N$，$\beta \in (V_T \cup V_N)^*$。
3. G 的任何产生式为 $A \to \alpha B$ 或 $A \to \alpha$，其中 $\alpha \in V_T^*$，$A, B \in V_N$。

1 型文法也称上下文有关文法。这种文法意味着，对非终结符进行替换时务必考虑上下文，并且，一般不允许替换成空串 ε。

如果非终结符的替换可以不必考虑上下文这就是 2 型文法。2 型文法也称上下文无关文法。

3 型文法也称右线性文法。由于这类文法等价于正规式，所以也称正规文法。

根据形式语言的这些定义所建立起来的语言模型就称作确定性语言模型。显然，这样的语言模型能自然地描述语言的结构及语法制约，和人类自身进行语音识别时有一定程度上的相似，是很理想的。

目前，研究得最多的是上下文无关文法和正规文法。虽然，这两种文法定义的语法范畴（或语言单位）是完全独立于这种范畴可能出现的环境，而自然语言的一个句子、一个词及至一个字，它们的语法性质和所处的上下文往往密切相关，因此，在理论上，上下文无关文法和正规文法都不宜于描述任何自然语言。但是，在有限的领域内，它们还是能提供相当多的语法信息，因为它们毕竟抓住了语言中部分的本质特征和语法制约。我们并不要求所用的语法规则多么完美，能穷尽所有的语法现象（当然，也不可能穷尽），只要它能提供足够的信息，帮助我们区分开在声学模型中无能为力分开的词，那么，就达到了使用语言模型的目的。

1.3.1.2 确定性语言模型所面临的问题

通常情况下，确定性语言模型在处理实际问题时受到很大的限制，与统计性语言模型相比，它有以下不足：

- 确定性语言模型只能判断候选语句在语法上的对和错，由此决定取舍，因此对实际语音识别系统中的语言模型应用并不十分适合，而且由于语言表达的灵活性，寻找语言表述的统计语句模式的模糊匹配往往比追求严格的语法结构的完全匹配更容易；
- 语法规则的获取并不直接和有效，往往需要一批具有较高专业素养的专家对大量例句进行分析才能给出合理的形式化描述，这带来了很大的人员、资源及时间的耗费；
- 语法规则的准确性检验缺乏简单的方法。同样是由于自然语言现象的复杂性，对于任意一条语法规则，都必然存在着例外，如何看待例外情况对规则性的破坏，这一问题在确定性语法规则的应用中尚未得到解决；
- 确定性语言模型对待不同的语言环境缺乏较好的鲁棒性，与基于统计的语言模型相比，蕴含在不同领域不同习惯的语言现象中规则很难保持一致，无法覆盖广泛的语言现象；
- 确定性语言模型缺乏良好的可计算性和可集成性，利用规则的判定往往需要进行对每条规则的穷举，计算复杂度过高。

1.3.2 统计语言模型

统计语言模型是根据统计理论，通过对大量语料进行统计，揭示出语言内部固有的统计特性的一种语言模型。最常用的统计语言模型是 *n-gram* 模型。

1.3.2.1 *n-gram* 模型

n-gram 模型基于这样一种假设：第 n 个词的出现只与前面 n-1 个词相关，而与其它任何词都不相关。我们用 $w_1,\ldots,w_n$ 来表示这 n 个词，那么词 w_n 出现的概率就可以写为 $p(w_n \mid w_1^{n-1})$，这里 w_1^{n-1} 表示词串 $w_1,\ldots,w_{n-1}$。在有大量训练语料保证的前提下，根据最大似然准则，可以得到：

$$p(w_n \mid w_1^{n-1}) = \frac{c(w_1^n)}{c(w_1^{n-1})} \tag{1.3.1}$$

$c(w_1^n)$ 和 $c(w_1^{n-1})$ 分别表示词串 $(w_1,\ldots,w_n)$ 和 $(w_1,\ldots,w_{n-1})$ 在训练语料中出现的次数。又假设这 n 个词组成一个句子 W，那么这句话的先验概率就是：

$$p(W) = \prod_{i=1}^{n} p(w_i \mid w_{i-n+1}^{i-1}) \tag{1.3.2}$$

之所以把这种模型称为 *n-gram* 模型，就在于它只反映了连续 n 个词之间的相关信息。

在 n 比较小（$n \le 3$）的情况下，这个模型还是比较可行的。

当 n=1 时，称为 *unigram* 模型，它实际上计算的是各个词在训练语料中出现的频度。在这个模型下，所有词都被认为是相互独立的，彼此之间没有相关信息，只使用了词频的统计特性。

当 n=2 时，称为 *bigram* 模型，它实际上计算的是各个词对在训练语料中出现的频度。在这个模型下，每个词只与它前面出现的那个词相关。从（1.3.1）式可得：

$$p(w_2 \mid w_1) = \frac{c(w_1^2)}{c(w_1)} \tag{1.3.3}$$

当 n=3 时，称为 *trigram* 模型，它实际上计算的是词的三元组在训练语料中出现的频度。在这个模型下，每个词只与它前面出现的那个词对相关。从（1.3.1）

式可得：

$$p(w_3 \mid w_1^2) = \frac{c(w_1^3)}{c(w_1^2)} \tag{1.3.4}$$

1.3.2.2 *n-gram* 模型的缺陷

作为统计模型中最常用的 *n-gram* 模型，在包括语音识别在内的自然语言处理中得到了广泛的应用，但是它也有很大的缺陷，如不加以改进，必然无法满足未来应用中对更加精确的语言模型的需求。其弱点体现在：

- 统计模型并不能理解文本或语句的实际意义，因此，一些存在严重语义错误的语句可能被求解为似乎合理的结果；
- 统计模型需要大量的训练语料，同时需要存储大量的模型参数。在现实情况下，这种要求并不总能得到满足。存储空间的制约，使得训练过程中，语序长度不可能太长，只能得到近序的语言联接关系。同时训练数据的稀疏往往导致求解精度的下降，从而影响统计模型的实际求解能力；
- 目前，*n-gram* 只能实现语言时序上的联接概率求解，因此并不能俘获全局远邻的语词关联信息，同时也不能显式地利用一些语言或领域知识，而这些特殊的知识对求解可能是非常有用的。

两种语言模型各自的特点非常鲜明，对于每种语言模型，都可以对语音识别系统起指导作用，但考虑到在实际系统中算法的实用性和成熟程度，本文中主要涉及的将是统计语言模型及其改进和应用。

本文的组织如下：第二章讨论统计语言模型的构建，包括汉语语料库的处理流程，统计语言模型（*n-gram*）的训练、平滑算法及其改进。第三章是统计语言模型在汉语智能整句输入法中的应用，该部分将主要论及一个高效的语言解码算法——基于多栈的音节同步算法。第四章的内容是关于统计语言模型在听写机中应用的问题和解决方案，此章中首次提出了基于预测技术的综合同步搜索算法。

第二章　统计语言模型的构建

2.1 汉语语料库的处理

2.1.1 汉语与西方语言的特点

从信息处理和语言知识的角度分析，汉语与西方语言相比具有显著的不同点，主要体现在：

- 西方语言的语言单元即词之间具有明显的空格分界，词的定义十分明确。而汉语的语言单元之间没有明确的分界。
- 西方语言的同音词较少，而汉语的发音只有约 400 个无调音节或 1200 个有调音节，所以同音字、同音词比较多。
- 西方语言明确的语法变化标志较多，如单复数、时态等等。汉语没有这些明确的标志。
- 西方语言具有较为固定的语法、语义表示形式。而汉语只有一小部分书面语言能够借用西方语法规则对其进行形式化表示。绝大部分语言模式目前还难以找到结构化的表示方式。
- 汉语一词多意的现象更加严重。
- 英语文本是小字符集上的词串，汉语文本是大字符集上的字串。因此汉语处理和英语处理也就不同，额外需要大字符集处理和字串到词串处理这两大块任务。

以上的特点决定了汉语语言模型在研究和实现上有自己的难点：

- 汉语语言模型中分词的特殊问题。汉语词切分成为汉语理解与处理首要的问题之一。
- 大量的同音字、同音词给语言模型的处理增加了负担。
- 十分灵活和自由的语言表述使得寻找形式化的语言规则来描述汉语语法、语义限制比较困难。

但是，词是最小的能独立活动的有意义的语言成分[Zhu 82]，汉语处理应用系

统只要涉及语法语义，就需要以词为单位，因此，汉语处理的首要问题就是确立适合的词典。

2.1.2 机器词典

随着语音识别和计算机语言学的发展，自然语言处理的目标已经转向对大规模真实文本进行处理。这里人们面临的一个主要障碍就是知识空缺。如何获取足够的词汇知识以建立处理大规模真实文本所必需的机器词典，也成为了语音识别和自然语言处理研究的一个重要课题。

一般说来，目前构造机器词典大致有以下三种方法[Zhang 94]：

（1）人工编写机器词典。这种方法需要投入大量人力。如日本的电子词典开发计划（*EDR*）。

（2）从机器可读词典（*MRD, Machine Readable Dictionary*）中获取各种词汇知识来建造机器可循字典（*MTD，Machine Tractable Dictionary*）。

（3）从语料库中获取词汇信息构造机器词典。这种方法是伴随语料库语言学的发展应运而生的。其出发点是：从“大规模”和“真实”这两个角度来考察，最理想的语言知识不是语言学著作，而是经过多层次加工的大规模语料库，因此，用这种办法生成的机器词典应该更适合大规模真实文本的处理。

2.1.2.1 语音识别用机器词典的选择

一般情况下，人工编写机器词典不太现实也不易保证一致性，因此我们采用的复合处理流程结合了上节后两种构造方法，详见图 2.1。首先我们选用了一个考虑了多种切分原则和经过人工校对的标注语料库，作为确定机器词典的初始语料。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。经统计，切分之后的语料中含大约 26 万个不同词条，基本上囊括了人们日常中使用的全部词汇，构成图中所示的**真实世界词典**。

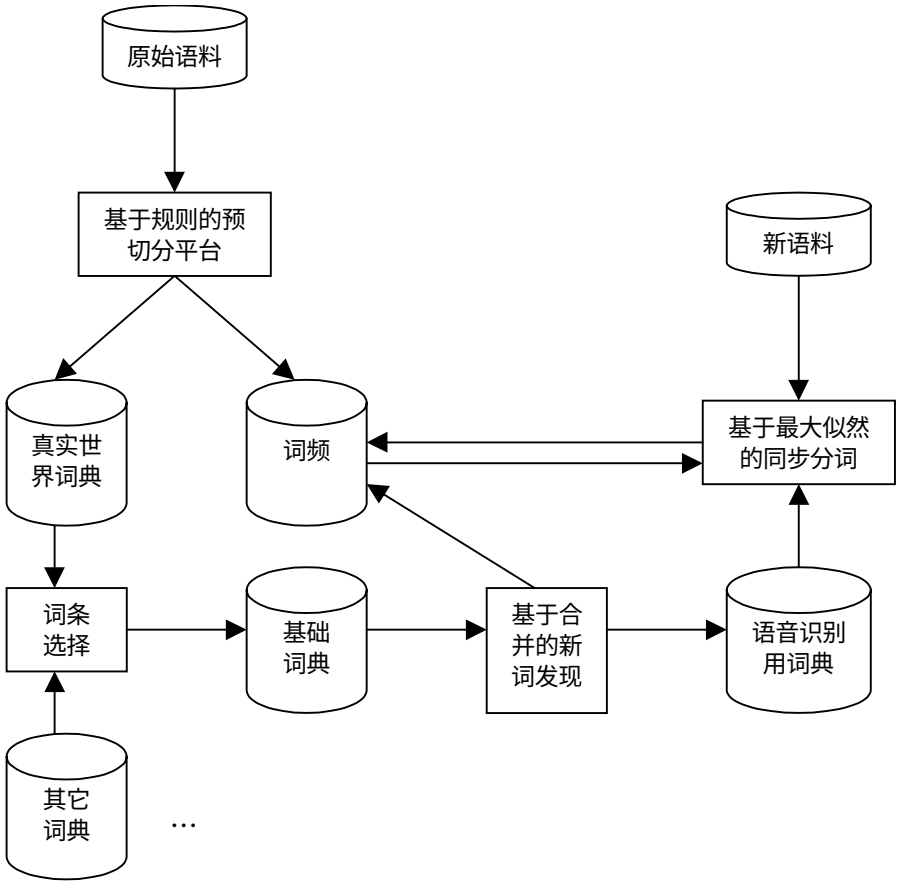

图 2.1　语音识别用机器词典的构造流程

由于实际语音识别中无法处理如此规模的词典，我们利用以下分级原则进行词条的选择和精简。

（1）真实世界词典中的高频词部分，由于来源于大规模的语料统计，我们认为可靠性较高，全部收入基础词典；

（2）真实世界词典中的中频词部分，须与其它版本的机器词典相比较，保留重合部分，这部分是包括语言学家在内的大部分汉语使用者所认同的部分。本系统中采用的**其它机器词典**主要是北京大学计算语言学研究所研制的《现代汉语语法信息词典》[Yu 98]和东北大学计算机系的《机器翻译用电子词典》。

（3）真实世界词典中的低频词部分，虽然有一部分也是人们认同的词，但

由统计结果知道，在日常中使用不多，经人工审查确认后，不作为基础词汇收入词典。

（4）所有的汉字须作为单字词加入词典。

2.1.2.2 基于合并的新词发现

新词发现是一个中文处理和西文处理都需要解决的问题。由于在建造词典时无法兼顾到所有可能，会漏掉很多常用的词汇，或将常用词汇错分为更小的单元（如将“匍匐”分为“匍”和“　匐”两个词），还有可能将若干常用词汇错合成不常用的词组，并当作了一个孤立词汇（如“北京”和“市”错合成“北京市”）。但是，由于汉语词定义的不明确，很多“词”介于词和词组之间，无法准确界定，因此我们很有必要确定一个准则来定义出适合语音识别的词典。

在实际系统中，由于我们获得的真实世界词典分词标准比较严格，此外根据上述原则（4），所有汉字均收入基础词典，所以我们假定新词只能由基础词典中的词条合并而成，并提出了一个**基于合并的以最小困惑度为目标函数的新词发现**策略。

困惑度（*Perplexity*）是一个衡量基于 *n-gram* 的统计语言模型的基本参数，定义为：

$$H = -\frac{1}{N-n+1}\sum_{i=n}^{N}\log(p(w_i \mid w_{i-n+1}^{i-1})) \tag{2.1.1}$$

式中 n 为 2 或 3 分别对应 *bigram* 或 *trigram*；N 为测试文本的总词数。H 值越小，表示模型的不确定度越低，性能越好。我们应该注意到，困惑度同时也是文本（语料）的评价函数。从物理意义上，困惑度可以被解释为每步马尔科夫求解的平均分枝路径数，也即每个语言单元平均可能的后接单元数，因此，它也可以是比较不同的语词切分和语料处理方法所得到的基础信息词典和词联接信息结果的评价策略。

具体算法流程如图 2.2，其中，W_i 为第 i 次迭代生成的词表，$N(k)$表示词条 k 的词频，为了计算方便，在实际系统中计算困惑度时 n 取为 1，即模型为 *unigram*。

通过合并，我们可以发现一些不太容易被注意到，但在日常生活中常常用

到的词汇，如“车匪路霸”、“株式会社”、“尊师重教”等。

(1) $i=0$，计算语料库的困惑度 H_0；

(2) $i=i+1$，选取阈值 M_i；

(3) 定义 $W_i^{(1)}=\{k|k=k_1k_2,\text{且}N(k_1k_2)>M_i*(N(k_1)+N(k_2)),\text{且}k_1,\ k_2\in W_i\}$；

(4) 定义 $W_i^{(2)}=\{k_1|N(k_1k_2)=N(k_1)\text{或}N(k_2k_1)=N(k_1),\text{且}k_1,\ k_2\in W_i\}$；

(5) $W_{i+1}=W_i+W_i^{(1)}-W_i^{(2)}$；

(6) 计算语料库的困惑度 H_i，若 $H_i<H_{i-1}$ 则转到(2)，否则停止。

图 2.2　基于合并的新词发现算法

2.1.2.3 语音识别用机器词典 *EasyDic99*

经过上述过程，最后我们获得了一个含 50,624 个词（包括所有全角标点符号）的语音识别用机器词典 *EasyDic99*，词典中所有词条的长度从 1 到 4 不等。图 2.3 和图 2.4 分别记录了该机器词典按照长度和词频的分布情况。

从图 2.3 可以看出机器词典 *EasyDic99* 中二字词占绝对多数，这与大部分机器词典的统计结果相同；从按词频分布的图 2.4 中可以看出，仅仅 22,000 多词就可占所有词条在语料库中出现次数的 99%以上，这也与语言学家认定的常用词个数基本相符。这两个方面说明，*EasyDic99* 是一个基本符合语法规范同时又充分考虑了语音识别需要的机器词典。

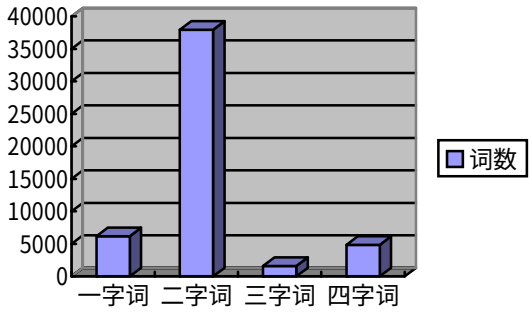

图 2.3 机器词典按词长的分布

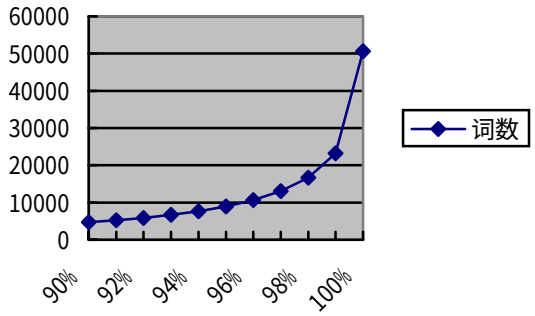

图 2.4 机器词典按词频的分布(占总词频的比例)

2.1.2.4 基于最大似然的同步分词算法

由于训练语言模型需要较大规模的语料，此外自然语言的时间渐变性也需要语言模型随之不断地更新，所以我们需要经常性地扩充或更新训练语料，以及修改语言模型的参数。图 2.2 中的同步分词环节即是为了这一目的而设计的。基于最大似然的求解方法在很多分词系统中都有所使用，其本质是首先依据分词规范和构词原则寻找汉字字符串 $C = C_1C_2 \ldots C_n$ 的所有可能的切分，然后对每一种可能的语词切分组合 $W = W_1W_2 \ldots W_m$，计算并寻找最大似然概率所对应的语词联接形式作为该字符串的最终切分结果。需要指出的是，传统的基于最大似然的分词算法由于需要预先确定所有可能的切分方式，重复的工作量较大，因而一般只以 *unigram* 作为计算似然概率的依据，导致切分不准确。针对这一不足，我们提出了一种同步的分词算法，该算法的特点是不需预先分析出所有的切分方式，而是在切分的过程中，利用动态规划算法或束搜索（*Beam Search*），对已经切分出的若干候选结果进行同步评价，并进而得出一个全局最优的切分结果，这样我们可以利用更精确的语言模型 *trigram*。

首先我们定义对路径的评价函数为

$$W^* = \arg\max_w P(W|C) = \arg\max_w P(W)P(C|W) = \arg\max_w P(W)\prod_{i=1}^{m} P(C_{k(i)}^{l(i)}|W_i) \tag{2.1.2}$$

其中

$$P(C_k^l|w) = P(w|C_k^l) = \begin{cases} 1 & w = C_k \ldots C_l \\ 0 & \text{其它情况} \end{cases} \tag{2.1.3}$$

根据评价函数我们可以构造如下的基于 *trigram* 的同步算法：

用 $Q_{uvw}(\Delta i, i)$ 表示当前处理字为 $C_{i+\Delta i}$ 且前导词对为 (u,v) 的搜索路径的得分，该路径中词 v 的末字为第 i 个字，下一个可能决定（但未到达边界）的词为 w。它是评价当前分词方案，并决定该方案能否继续进行下去的依据。**前导词(对)**指的是搜索路径已确定出的最后一(两)个词。

用 $Q_{uv}(i)$ 表示前导词对为 (u,v) 且当前处理字 C_i 为词 v 的末字的搜索路径的得分。它是动态规划算法中计算 $Q_{uvw}(\Delta i, i)$ 的基础。

用 $H(i,v,w)$ 表示在当前处理字为 C_i 的情况下，前导词对为 (v,w) 且 C_i 恰好

为 w 的末字的最优路径得分。它一方面可以用来衡量在 i 时刻到达词边界的路径，另一方面可以用来作为下一个词开始进入搜索的起始得分。

根据上述定义，相应的计算公式如下：

$$Q_{uvw}(\Delta i,i)=Q_{uv}(i)\cdot P(w|C_{i+1}^{i+\Delta i}) \tag{2.1.4}$$

$$H(i,v,w)=\max_{u}\{Q_{uvw}(L_w,i-L_w)\cdot p(w|u,v)\} \tag{2.1.5}$$

$$Q_{uv}(i)=H(i,u,v) \tag{2.1.6}$$

其中 L_w 表示词 w 的长度，$p(w|u,v)$ 为已知前导词对为 (u,v) 时 w 出现的条件概率。

在实际搜索中，需要对路径进行一定的剪枝，剪枝的依据是 $Q_{uvw}(\Delta i,i)$。首先，对每个可能的 $(\Delta i,i)$ 求出该情况的最佳路径得分

$$Q^{*}(\Delta i,i)=\max_{u,v,w}Q_{uvw}(\Delta i,i) \tag{2.1.7}$$

然后将所有满足

$$Q_{uvw}(\Delta i,i)<\alpha\cdot Q^{*}(\Delta i,i),\quad \alpha<1 \tag{2.1.8}$$

的路径从待扩展路径堆栈中去掉不予继续处理。这样可以大大减少对一些不必要的分词方式的进一步处理。

2.2 *n-gram* 模型的训练及其平滑

n-gram 语言模型作为一个简单、易用的模型已经在许多语音识别系统中得到广泛的应用。但 *n-gram* 模型存在着严重的训练数据稀疏问题，我们下面主要讨论这个问题出现的原因及其解决办法。

在一个听写机中，词表的规模往往是两三万，甚至会达到五万左右。在这样的规模下，可以想见，即使 *n=3*，也要估计 50000^3 个参数。如此庞大的数据量对现有的计算机来说，既难于存储，也难于计算。此外，要估计这么多参数，就需要一个非常庞大的训练语料以保证统计数据的可靠性，而在目前，这么庞大的一个训练语料是不可能实现的。因此，*n-gram* 模型存在着训练数据严重稀疏的问题。从（1.3.1）式中可以看出，如果有任何 *n* 元词对没有出现在训练语

料中，那么根据最大似然准则，它的概率估计就是 *0*。也就是说，如果这个 *n* 元词对出现在识别文本中的话，那么它就被认为是不可能的组合。这样一来，一些本来有意义的 *n* 元词对就有可能因为没有出现在训练语料中，而被认为是没有意义的。这样的估计会直接导致识别率的下降，从而降低整个系统的性能。下面介绍的几种方法就是针对解决 *n-gram* 模型中零概率估计问题的。

2.2.1 几种常用的 *n-gram* 平滑方法

2.2.1.1 基于词类的 *n-gram* 模型

在基于词类的 *n-gram* 模型中，每个词属于唯一的一个类。如果某个词的 *n-gram* 没有在训练语料库中出现，该词的 *n-gram* 仍然可以通过所属词类的 *n-gram* 而得到估计。更一般的是，由于词类的数目小于词的数目，模型参数的数目减少了，从而参数能够更可靠地被估计。然而另外一方面，减少模型参数的数目使得模型变得不够精细而导致临近词的预测精确度下降，因此要从这两个极端中进行一定的折衷。

一般情况下，词类依据语法和语义概念而且由语言学专家来定义，这些语法语义概念，经常包括词性（*POS, Part of Speech*）和语义类。此外，我们还可以通过统计的标准来定义词类，该统计标准在大多数情况下（但不是必需的）是最大似然或最小困惑度[Jelinek,91; Brown,92; Kneser,93; Ney., 94]。在按照统计标准的定义方法中，词类通过使用一种基于聚类的算法来定义，此类算法都是基于类在训练语料库上的 *n-gram* 语言模型的困惑度得到最小，因此我们简称为 *n-gram* 聚类算法。

以 *bigram* 为例，基于词类的 *bigram* 模型实际上就是将条件概率表述为

$$p(w \mid v) = p_0(w \mid g_w) \cdot p_1(g_w \mid g_v) \qquad (2.2.1)$$

其中

$p_1(g_w \mid g_v)$ 为传递概率函数，它表示前一个词类是 g_v，当前词类为 g_w 的一阶 *Markov* 传递概率，

$p_0(w \mid g)$ 为成员概率函数，用来判断词 w 属于词类 g 的概率。由于一个词只属于唯一的一个词类，我们有

$$p_0(w \mid g) = \begin{cases} > 0 \ \text{if g} = \text{g}_\text{w}, \\ = 0 \ \text{if g} \neq g_{w.} \end{cases} \tag{2.2.2}$$

因此，在某种程度上我们可以粗略地表达为 $p_0(w \mid g_w)$。

2.2.1.2 基于最大熵的 *n-gram* 模型

在统计语言模型中，我们需要通过条件概率 $P(w|\phi(h))$来估计所需的语言模型条件概率 $P(w|h)$，其中 $\phi(h)$代表历史词串所属的等价类。为达到这个目的，我们需要满足以下三个相关条件：

（1）不须估计太多参数，

（2）有足够多的数据用于估计，

（3）概率值可以在有限的时间中，利用有限的存储空间，根据存储的参数依时序计算得出。

最大熵的方案[Jelinek 97]的提出直接解决了前两个问题，从而自动解决第三个问题。它的思路是在以下两个假设的基础之上构造联合概率 $P(w,h)$

（1） $P(w,h)$应该满足一些线性的限制，

（2）对未受限制的方面的情况我们完全不知道，从而也无法对 $P(w,h)$的构建起任何提示作用。

假设在估计语言模型概率 $P(x,y,z)$时我们知道它的一部分边界概率，比如说 $P(x,y)$和 $P(x,z)$，即 $P(x,y,z)$需要满足以下限制

$$\sum_{x,y,z} P(x,y,z)k(x,y,z \mid x',y') = P(x',y') \tag{2.2.3}$$

和

$$\sum_{x,y,z} P(x,y,z)k(x,y,z \mid x',z') = P(x',z') \tag{2.2.4}$$

其中

$$k(x,y,z \mid x',y') = \begin{cases} 1 & if \ \ x = x' \ and \ \ y = y' \\ 0 & \text{其它情况} \end{cases} \tag{2.2.5}$$

和

$$k(x,y,z \mid x',z') = \begin{cases} 1 & if\ \ x = x'\ and\ \ z = z' \\ 0 & 其它情况 \end{cases} \qquad (2.2.6)$$

如果我们知道的有关 $P(x,y,z)$的情况只限于（2.2.3）（2.2.4）两式，那么为了与我们预先假设的对其他方面一无所知的限制保持一致，我们只能认为 $P(x,y,z)$是满足（2.2.3）（2.2.4）两式的所有概率分布中墒最大的一个。在实际应用中，为了简化计算起见，我们可以不去最大化墒值，而是选择 ***P*** 使其与一些其他已知的概率分布 ***Q*** 之间的分歧最小，也就是说，去最小化分歧函数（*Divergence Function*）

$$D(P\|Q) = \sum_{x,y,z} P(x,y,z)\log\frac{P(x,y,z)}{Q(x,y,z)} \qquad (2.2.7)$$

为了使叙述更加清楚，我们定义

$x = x_1, x_2, ..., x_n$ 为一个 *n* 维随机变量，

$k(x|i)$是第 *i* 个限制函数。

我们要求解的问题是：决定 $P(x)$使

（1） 它对每个给定的目标限制 *d(i), i=1,2,…,m* 都满足 $\sum_x P(x)k(x \mid i) = d(i)$

（2） 分歧函数 $D(P\|Q)$对给定的分布 $Q(x)$值最小。

我们可以用拉格朗日乘数法（*Undetermined Lagrangian Multipliers*）来确定解的形式。对所有可能的 x 的值，将（2.2.7）式对 $P(x)$求偏导并使之等于 0

$$D(P\|Q) - \sum_i \lambda_i \left[\sum_{x'} P(x')k(x'|i) - d(i) \right] = 0 \qquad (2.2.8)$$

结果为

$$\log\left[\frac{P(x)}{Q(x)}\right] + 1 = \sum_{i=1}^{m} \lambda_i k(x|i) + \lambda_0 \qquad for\ all\ x \qquad (2.2.9)$$

于是

$$P(x) = Q(x)\cdot[\exp\lambda_o]\cdot\left[\exp\{\lambda_i k(x|i)\}\right] \qquad (2.2.10)$$

其中 λ_i 必须满足限制，它们的值可以通过一些梯度下降的迭代算法来进行估计。

2.2.1.3 基于图灵估计的退化 *n-gram* 模型

图灵（*Turing*）概率估计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，把调整出来的概率和按照一定的规则分布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

$$N = \sum_r r n_r \tag{2.2.11}$$

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：

$$P_T = \frac{r^*}{N} \tag{2.2.12}$$

其中

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{2.2.13}$$

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”，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，最大似然概率估计为

$$P_{ML} = \frac{c(w_1^n)}{N} \tag{2.2.14}$$

而图灵概率估计为

$$P_T = \frac{c^*(w_1^n)}{N} \tag{2.2.15}$$

其中

$$c^*(x) = (c(x)+1)\frac{n_{c(x)+1}}{n_{c(x)}} \tag{2.2.16}$$

根据(2.2.1)、(2.2.2)、(2.2.3)和(2.2.5)，对出现在训练语料中的词串，它们的图灵概率估计和为

$$\sum_{w_1^n:c(w_1^n)>0} P_T(w_1^n) = 1 - \frac{n_1}{N} \tag{2.2.17}$$

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

$$\sum_{w_1^n:c(w_1^n)=0} P_T(w_1^n) = \frac{n_1}{N} \tag{2.2.18}$$

另一方面，

$$\sum_{w_1^n:c(w_1^n)>0} [P_{ML}(w_1^n) - P_T(w_1^n)] = \sum_{w_1^n:c(w_1^n)>0} \delta_{c(w_1^n)} = \sum_{r>0} n_r (1-d_r)\frac{r}{N} = \frac{n_1}{N} \tag{2.2.19}$$

其中

$$\delta_c = \frac{c}{N} - \frac{c^*}{N} = (1-d_c)\frac{c}{N} \tag{2.2.20}$$

因此，

$$\sum_{w_1^n:c(w_1^n)>0} \delta_{c(w_1^n)} = \sum_{w_1^n:c(w_1^n)=0} P_T(w_1^n) \tag{2.2.21}$$

Katz[1987] 基于此“ 残余 ”提出了一种退化（*Back off*）的思想，即：把(2.2.21)中的 δ_c 解释为一个具有计数值 $c(w_1^n)$ 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，我们将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(2.2.20)中给出的 δ_c ，我们引入 $\delta_c^{(cond)}$ ：

$$\delta_{c(w_1^n)}^{(cond)} = (1-d_{c(w_1^n)})\frac{c(w_1^n)}{c(w_1^{n-1})} \tag{2.2.22}$$

我们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，

$$P_s(w_n \mid w_1^{n-1}) = \tilde{P}(w_n \mid w_1^{n-1}) = d_{c(w_1^n)}\frac{c(w_1^n)}{c(w_1^{n-1})} \tag{2.2.23}$$

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 $\tilde{\beta}$ ：

$$\tilde{\beta}(w_1^{n-1}) = \sum_{w_n:c(w_1^n)>0} \delta_{c(w_1^n)}^{(cond)} = 1 - \sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_1^{n-1}) \qquad (2.2.24)$$

这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(2.2.24)定义的条件“块” $\tilde{\beta}$ 分配到 $c(w_1^n)=0$ 的那些 w_n 上：

$$P_s(w_n \mid w_1^{n-1}) = \alpha P_s(w_n \mid w_2^{n-1}) \qquad (2.2.25)$$

其中

$$\alpha = \alpha(w_1^{n-1}) = \frac{\tilde{\beta}(w_1^{n-1})}{\sum_{w_n:c(w_1^n)=0} P_s(w_n \mid w_2^{n-1})} = \frac{1 - \sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_1^{n-1})}{1 - \sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_2^{n-1})} \qquad (2.2.26)$$

是一个归一化常数。当 $c(w_1^{n-1})=0$ 的时候，我们定义

$$P_s(w_n \mid w_1^{n-1}) = P_s(w_n \mid w_2^{n-1}) \qquad (2.2.27)$$

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

最后，我们把(2.13)、(2.15)和(2.17)结合起来，用如下的递归表达式表示条件概率分布：

$$P_s(w_n \mid w_1^{n-1}) = \tilde{P}(w_n \mid w_1^{n-1}) + \theta(\tilde{P}(w_n \mid w_1^{n-1})) \cdot \alpha(w_1^{n-1}) P_s(w_n \mid w_2^{n-1}) \qquad (2.2.28)$$

其中

$$\theta(x) = \begin{cases} 1, & \text{如果} x = 0 \\ 0, & \text{其它情况} \end{cases} \qquad (2.2.29)$$

现在，我们就有了一个修正过的概率估计式，它由(2.2.28)给出。但对于计数值 $c>k$ 的那些 n 元词串来说，我们认为它们的概率估计是可靠的，而不想对它们的概率估计进行折扣，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，我们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，我们重新定义

$$d_r = 1, \qquad \text{对} r > k \qquad (2.2.30)$$

而且我们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：

$$\sum_{w_n:c(w_1^n)>0}\delta_{c(w_1^n)}=\sum_{1\le r\le k}n_r(1-d_r)\frac{r}{N}=\frac{n_1}{N} \tag{2.2.31}$$

式(2.2.31)类似于式(2.2.19)。我们按照如下形式求解(2.2.31)中的d_r：

$$(1-d_r)=\mu(1-\frac{r^*}{r}) \tag{2.2.32}$$

其中μ是一个常数。这样，(2.2.31)的唯一解就是：

$$d_r=\frac{\frac{r^*}{r}-\frac{(k+1)n_{r+1}}{n_1}}{1-\frac{(k+1)n_{r+1}}{n_1}},\qquad 对1\le r\le k \tag{2.2.33}$$

式(2.2.32)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

$$P_s(v|u)=\begin{cases}d_r(u)f(v|u) & for\ \ r>0\\ \alpha(u)P_s(v) & for\ \ r=0\end{cases} \tag{2.2.34}$$

其中

$$f(v|u)=\frac{C(u,v)}{C(u)}=\frac{r}{c(u)}\ , \tag{2.2.35}$$

$$d_r=\begin{cases}1 & for\ \ r>k\\ \dfrac{\frac{r^*}{r}-\frac{(k+1)n_{k+1}}{n_1}}{1-\frac{(k+1)n_{k+1}}{n_1}} & for\ \ k\ge r>0\end{cases}, \tag{2.2.36}$$

$$\alpha(u)=\frac{1-\sum_{v:c(u,v)>0}P_s(v|u)}{1-\sum_{v:c(u,v)>0}P_s(v)}. \tag{2.2.37}$$

基于图灵估计的退化算法除了能够合理有效地估计未出现过的词串w_1^n的

概率之外，还考虑到了在同一 w_1^{n-1} 词前缀下，出现过不同次数的不同词串 w_1^n 之间概率的平滑，避免了联合概率 $P_s(w_1^n)$ 在随 w_1^n 同现次数变化方向上的过分“ 波动 ”。图 2.5 示意了基于图灵估计的退化算法中平滑前后的概率，其中横轴是词串出现的次数，纵轴是出现相应次数的所有词串估计出的概率和。可以看出，经过图灵折扣的退化模型计算出的概率更合理。因此，我们的 *n-gram* 语言模型的平滑算法采用基于图灵估计的退化算法。

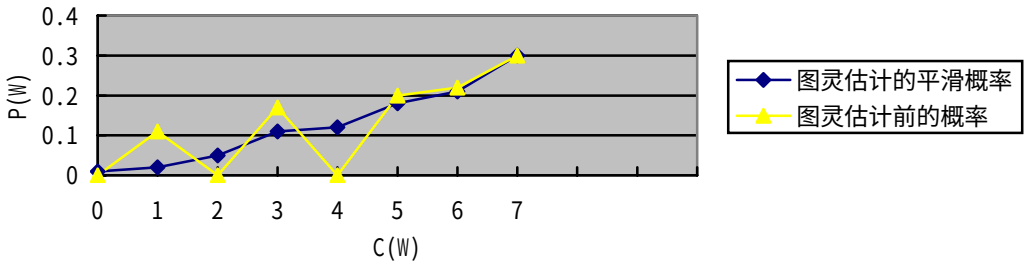

图 2.5　图灵估计前后的退化模型比较

2.2.2 基于图灵估计的退化算法在实际应用时的若干改进

虽然基于图灵估计的退化算法能够将未出现的 *n* 元词串的概率近似地用相应的（*n-1*）元词串的概率有效地估计出来，但是在实际应用中，公式中的退化折扣系数的计算以及估计的结果还是会受到很多因素的制约。这些制约体现在：

（1）由于训练数据的极度稀疏和畸形，导致部分系数计算出现溢出现象；

（2）在利用高阶 *n-gram*（*n>2*）的系统中，用于存储 *n* 元词串出现次数的空间需求过大；

（3）由于退化算法在估计未出现词串的概率时考虑了低阶模型的影响，而在对出现过的词串进行估计时完全不考虑低阶模型，导致部分概率估计结果与直观感觉不一致，即**概率估计失真**。

本节将对以上问题进行具体分析并给出合理的解决方案。

2.2.2.1 折扣系数计算的改进 [Mou 98]

由于在实际的语料统计中，数据的稀疏会经常使得 $n_r = 0 \quad (1 \le r \le k+1)$，无法直接计算由（2.2.13）式定义的 r^*，因而无法应用（2.2.33）式对 d_r 进行修正。因此在在估计 d_r 之前我们先对 n_r 进行了平滑，平滑公式为

$$\tilde{n}_r = \sum_{w_1^{n-1}:C(w_1^{n-1})>0} n_r(w_1^{n-1}) \Big/ \sum_{w_1^{n-1}:C(w_1^{n-1})>0} 1 \qquad (1 \le r \le k+1) \qquad (2.2.38)$$

其中 $n_r(w_1^{n-1})$ 表示在 w_1^{n-1} 词前缀下恰好出现了 r 次的不同词串 w_1^n 的计数。

此外，在根据 (2.2.26) 估计零概率分配系数 α 时，会遇到因 $\sum_{w_n:c(w_1^n)>0} \tilde{P}(w_n \mid w_2^{n-1}) = 1$ 而分母为零的情况。因此，我们对 α 的计算也进行了一定的修正：

$$\tilde{\alpha}\left(w_1^{n-1}\right) = \tilde{\beta}\left(w_1^{n-1}\right) \Big/ \left(1 - \sum_{w_m:C(w_1^m)>0,\text{且}C(w_2^m)>\tilde{\theta}(0)} \tilde{P}(w_m \mid w_2^{m-1})\right) \qquad (2.2.39)$$

其中

$$\tilde{\theta}(r) = r + \min(\theta \mid \theta \ge 0 \text{且} n_{r+\theta} > 0)$$

2.2.2.2 对模型存储空间的改进

首先，在高阶 *n-gram* 语言模型的训练过程中，需要记录 *n* 元词串的出现次数。我们从对 *trigram* 的统计中，发现对于 *EasyDic99* 这样有 5 万多词的词典而言，存储的数据量过于庞大。此外，由于训练数据的极度稀疏，当 *n=3* 时，会有大量的三元词组在训练语料中不出现，其概率需要通过退化算法来估计，存储大量的折扣系数。而存储量太大，一方面会提高系统的存储需求，另一方面在系统运行时也需要花费大量时间去进行磁盘读操作和查找操作。

针对这个问题，许多系统在实际使用 *n-gram* 时，都根据一定的准则抛弃若干在语料库中出现过的词串，如 CMU[Seymore 96]曾提出了一个描述退化模型估计概率与真实概率之间差别的加权因子，并以此进行 *n* 元词串的选取。

我们根据实时系统的要求，从减少运算量的角度，也提出了相应的简化解决方案。

考虑实际连续语音识别系统中，由于搜索过程中的候选里有大量的不正确

候选语句，这些不正确的候选语句通常含有许多“**无意义**”（训练中不出现的）词串。一般来说，在“无意义”词串的前提下进行的搜索，需要语言模型计算概率的新词串大多也是“无意义”的，并不会出现在训练语料库中，它们的概率估计同样要通过（2.2.25）式给出。

事实上，我们的统计结果给出，实际计算时大约 90%以上的 *trigram* 语言模型概率估计都是通过（2.2.25）式给出，只有不到 10%的概率估计由（2.2.23）式给出。因此，对于训练数据稀疏的模型统计而言，折扣系数 d_r 对系统的贡献不大，可以酌情精减。针对这一情况，我们对 *trigram* 部分的概率计算进行如下修正：

$$P_s\left(w_i\middle|w_{i-2}^{i-1}\right)=\begin{cases} f\left(w_i\middle|w_{i-2}^{i-1}\right) & for\ \ r>k \\ \alpha\left(w_{i-2}^{i-1}\right)P_s\left(w_i\middle|w_{i-1}\right) & for\ \ r\le k \end{cases} \qquad (2.2.40)$$

这样我们就在系统性能基本不下降的同时，减少了对 d_r 的存储，也减少了同现次数为 *0* 到 *k* 次的三元词串的存储。

2.2.2.3 对概率估计失真的改进

概率估计的失真

当前对 *n-gram* 统计语言模型的平滑估计有两种算法，一种算法是前述的退化算法，另一种是删除插入算法（*DI，Delete Interpolation Algorithm*），该算法的概率计算公式一般可以表述为

$$P_s(w_n\mid w_1^{n-1})=\lambda P_{ML}(w_n\mid w_1^{n-1})+(1-\lambda)\cdot P_s(w_n\mid w_2^{n-1}) \qquad (2.2.41)$$

退化算法和删除插入算法都是用低阶的概率估计对高阶的概率估计进行一定的修正，但是它们在修正的时机上有所差别：退化算法只在待估计词对的同现次数为零时才引入低阶模型，而删除插入算法在任意时刻都由不同阶的语言模型计算出所需的概率得分。

但是由于基于图灵估计的退化算法在估计训练集中出现过的词串 w_1^n 的出现概率时完全不考虑词串 w_2^{n-1} 的情况，因此在某些情况下会出现异常现象。下面的例子可以充分地说明这个问题。

假设在某个 *n-gram* 语言模型中有下列语言现象存在，

$C(w)=5$，$C(ww_1)=0$，$C(ww_2)=1$，

$P_s(w_1)=0.2$，$P_s(w_2)=0.3$，

$d_1(w)=0.5$，$\alpha(w)=0.6$。

此例中，$C(ww_2)>C(ww_1)$且$C(w_2)>C(w_1)$，根据我们的直观感觉，联合概率$P_s(ww_2)$应该大于$P_s(ww_1)$，然而根据（2.2.34），我们可以计算出相应的平滑概率：

$$P_s(w_1|w)=\alpha(w)\cdot P_s(w_1)=0.6*0.2=0.12$$

$$P_s(w_2|w)=d_1(w)\cdot\frac{c(ww_2)}{c(w)}=0.4*1/5=0.08$$。

这时我们会发现$P_s(w_2|w)<P_s(w_1|w)$，从而$P_s(ww_2)<P_s(ww_1)$，计算结果和我们对模型的预期结果截然相反，我们称之为**概率估计失真**。

算法的修正

上面所述问题出现的原因是在平滑“ 生词串 ”（语料库中出现频度为零的词串）和平滑“ 熟词串 ”（语料库中出现频度不为零的词串）时考虑了不同的影响因素。为了消除这一分歧，我们构造了一个修正的退化平滑算法，该算法在基于图灵估计的退化算法基础之上，在平滑熟词串时结合了删除插入算法的特性，对每个词串给出更合理的概率估计。

事实上，基于图灵估计的退化算法可以表述成删除插入算法的一个特例。以 *bigram* 为例，（2.2.34）～（2.2.37）可以改写为

$$P_s(v|u)=\lambda(u,v)P_T(v|u)+(1-\lambda(u,v))P_s(v) \qquad (2.2.42)$$

其中

$$\lambda(u,v)=\begin{cases}1 & for\ r>0\\ \dfrac{\alpha(u)-1}{\beta(u,v)-1} & for\ r=0\end{cases} \qquad (2.2.43)$$

$$\beta(u,v)=\frac{1-\sum_{v:c(u,v)>0}P_s(v|u)}{n_0\cdot P_s(v)} \qquad (2.2.44)$$

显然，与（2.2.41）相比，两者的主要差别就在于权重函数$\lambda(u,v)$的选取。

为了在$r>0$时也能够考虑到$P_s(v)$的作用，我们对（2.2.43）进行扩展，得到

$$\lambda(u,v)=\begin{cases}\mu_r(u,v) & for\ r>0\\ \dfrac{\alpha(u)-1}{\beta(u,v)-1} & for\ r=0\end{cases} \tag{2.2.45}$$

为了使修改之后不破坏原来基于图灵估计的退化算法的平滑性，我们要求

$$\begin{aligned}&\sum_{v:c(u,v)=r}\left(\mu_r(u,v)\cdot P_T(v|u)+(1-\mu_r(u,v))\cdot P_s(v)\right)\\&=\sum_{v:c(u,v)=r}P_T(v|u)\qquad\forall r>0\end{aligned} \tag{2.2.46}$$

即

$$\begin{aligned}&\sum_{v:c(u,v)=r}\left(\mu_r(u,v)\cdot\left(P_T(v|u)-P_s(v)\right)\right)\\&=\sum_{v:c(u,v)=r}\left(P_T(v|u)-P_s(v)\right)\qquad\forall r>0\end{aligned} \tag{2.2.47}$$

由于（2.2.47）中只有$N\cdot R$个限制函数，而有$N^2\cdot R$个自由参数（N为词表大小，R为r的范围），因此一般情况下会存在多解。为了得到一个最优解，我们定义目标函数为最小化监测集（*held-out* corpus）的困惑度或最大化监测集的似然度，即最大化

$$\sum_{r>0}\sum_{u,v:c(u,v)=r}N(u,v)\cdot\log\left(\mu_r(u,v)P_T(v|u)+(1-\mu_r(u,v))P_s(v)\right) \tag{2.2.48}$$

其中$N(u,v)$为词对(u,v)在监测集中出现的次数。于是这一问题归结为条件极值问题，但是由于$N^2\cdot R$个自由参数和$N\cdot R$个限制函数对于任何一种条件极值求解方法而言都是十分复杂的，因此我们需要进行进一步简化。

（1）简化方案 A

首先我们可以考虑的是简化权重函数$\lambda(u,v)$的表达式，从而减少待估计参数。假定

$$\mu_r(u,v)=\lambda_1(r)+\lambda_2(r)\cdot C(v) \tag{2.2.49}$$

于是需求解的自由参数减少为$2\cdot R$个，为了使限制函数数目也能够相应地减少，并保证不出现无解的情况（即限制函数数少于自由参数数目），我们将（2.2.49）代入（2.2.47）之后，还将涉及不同的词u的限制函数进行求和，得

到

$$\lambda_1(r)\cdot\sum_{u,v:c(u,v)=r}\left(P_T(v|u)-P_s(v)\right)+\lambda_2(r)\cdot\sum_{u,v:c(u,v)=r}\left(C(v)\cdot\left(P_T(v|u)-P_s(v)\right)\right)$$
$$=\sum_{u,v:c(u,v)=r}\left(P_T(v|u)-P_s(v)\right)\qquad\forall r>0$$

（2.2.50）

共 R 个限制函数。

采用简化方案 A 的算法流程如图 2.6

(1)准备一个不同于训练集的监测集；
(2)根据基于图灵估计的退化算法在训练集上训练模型，得到相应概率的原始估计；
(3)在监测集中统计不同词对 (u,v) 的同现次数 $N(u,v)$ ；
(4)根据(2.2.48) 、(2.2.49) 和(2.2.50) 用迭代算法求解条件极值以及相应的自由参数值。

图 2.6　修正退化算法简化方案 A

（2）简化方案 B

根据前面的分析，$C(u,v)$越大，则 $P_T(v|u)$的估计越精确，为了体现这一原则，我们采用了一个分段的线性函数来代替（2.3.12）中复杂的权重函数，即

$$\mu_r(u,v)=\begin{cases}1 & r>k\\ \dfrac{r}{k+1} & r\le k\end{cases}\qquad(2.2.51)$$

于是

$$P_s(v|u)=\begin{cases}f(v|u) & for\ \ r>k\\ \dfrac{r\left(d_r(u)f(v|u)-P_s(v)\right)}{k+1}+P_s(v) & for\ \ k\ge r>0\\ \alpha(u)P_s(v) & for\ \ r=0\end{cases}\qquad(2.2.52)$$

（3）简化方案 C

采用简化方案 A 之后，修正退化模型训练的复杂度大大下降，但是仍需要一个迭代过程来决定最佳的参数估计。简化方案 B 的权重选择方案计算方便，而且也体现了我们修正退化模型的原则：在估计熟词串(u,v)的条件概率 $P_s(v|u)$时考虑了低阶模型估计的概率 $P_s(u)$的影响。但是它还是没有充分体现频度不同

的词 u 之间的差别。基于以上两点分析，我们提出了简化方案 C，即定义

$$\mu_r(u,v)=\begin{cases}1 & for\ r>k\\ \dfrac{P_T(v|u)}{P_T(v|u)+P_s(v)} & for\ k\geq r>0\end{cases} \tag{2.2.53}$$

于是

$$P_s(v|u)=\begin{cases}f(v|u) & for\ \ r>k\\ \dfrac{P_T(v|u)\cdot P_T(v|u)+P_s(v)\cdot P_s(v)}{P_T(v|u)+P_s(v)} & for\ \ k\geq r>0\\ \alpha(u)P_s(v) & for\ \ r=0\end{cases} \tag{2.2.54}$$

由于采用上述三种简化方案时，（2.2.47）式都没有得到满足，因此为了保证 $\sum_v P_s(v|u)=1$，在计算完 $r>0$ 时的 $P_s(v|u)$ 后需要利用（2.2.37）式对 $\alpha(u)$ 进行重估。

2.3 实验结果及分析

2.3.1 基于合并的策略发现的新词

在我们的实验中，基于合并的新词发现策略在语料中共发现新词 36 个，其中一部分是由于词典选择时的疏忽而漏掉的，还有一部分是在语料库中经常出现而合并得来的。

表 2.1　基于合并策略发现的新词

车匪+路霸	踌+躇	缔+约国	傀+儡	傈+僳
辽源+得亨	咆+哮	霹+雳	鸳+鸯	囫+囵
尊师+重教	鲋+骟	匍+匐	坩+埚	揶+揄
株式+会社	囵+圄	彷+徨	猞+猁	沆+瀣
枇+杷	桎+梏	觊+觎	镨+毪	氍+毹
腼+腆	忐+忑	盱+眙	褴+褛	襁+褓
颟+顸	蜉+蝣	蝙+蝠	蟋+蟀	踟+蹰
蹂+躏				

2.3.2 基于最大似然分词策略的测试结果

用于测试分词算法的语料库为《人民日报》1999 年部分新闻语料，共有 324，806 字。实验表明基于最大似然的同步分词算法分词正确率为 99.43%，正确率的计算是将同步分词结果与北京工业大学人工智能实验室研制的分词系统“工智 98”的分词结果相比较得出。

下面给出几个比较典型的结果：

例 1 风景\旅游\景点\是\人们\轻松\娱乐\的\好\去处\，\应该\清爽\亮丽\，\赏心悦目\。\遗憾\的\是\，\一些\景点\脏\、\乱\、\差\现象\严重\，\瓜\皮\果\壳\、\纸屑\烟蒂\随处\可见\。\这\固然\是\因为\一些\游客\不\文明\所致\，\景点\管理\不善\也\是\一个\重要\原因\。\推广\清洁\袋\的\做法\，\既\能\给\游客\提供\方便\，\也\有\助\于\提高\其\环保\意识\，\进一步\搞好\景点\的\环卫\工作\。\

例 2 美\国会\召开\会议\（此句的难点是“美国”和“国会”之间的切分歧义）

例 3 当\原子\结合\成\分子\时\（此句的难点是多种组词方式，“结合”、“合成”、“成分”、“分子”、“子时”等，无论是简单从左向右最大匹配还是从右向左最大匹配，都无法给出满意结果）

以上测试结果说明，利用在准确切分的语料库上统计出的语言模型和基于最大似然的同步算法，可以合理且高效地进行汉语分词工作。利用该系统进行分词不仅可以回避一般歧义校正过程中复杂规则的判断和应用，还可以部分解决如例 2 例 3 所示的连续型交集字段的问题。

2.3.3 修正的退化模型实验结果

（1）困惑度测试

如前所述，困惑度是检查语言模型有效程度的一项重要测度。我们的测试语料与分词测试相同，测试结果如表 2.2。

表 2.2　不同平滑算法下语言模型的困惑度比较

语言模型种类	Unigram	Bigram	Trigram
基于图灵估计的标准退化模型	1817	442	274
修正退化模型简化方案 A	-	397	244
修正退化模型简化方案 B	-	429	267
修正退化模型简化方案 C	-	402	250

（2）音字转换测试

音字转换实验可以看作是汉语连续语音识别系统的一个子模块测试平台，它用来测试语言模型在实际系统中的有效性。实验的输入为有意义的无调音节串，输出结果为根据语言模型决定的对应汉字串，衡量效果的指标为字错误率。该系统所采用的搜索算法为基于双活动栈的音节同步算法（*BAS－SSS，Bi-Active-Stack based Syllable Synchronous Search*），我们将在第三章中详述。测试集来源于 863 提供的语音库语料，共有 1559 句，22083 个音节。测试结果如表 2.3。

表 2.3　不同平滑算法下音字转换字错误率比较

语言模型种类	标准退化	简化方案 A	简化方案 B	简化方案 C
字错误率（%）	6.5%	4.5%	5.0%	4.7%

从以上结果我们可以看出，无论是基于困惑度的测试，还是实际系统中音转字的测试，几种修正的退化模型与标准的基于图灵估计的退化模型相比，都有较大程度的改进。相对而言，简化方案 A 由于提供了更多的可估计的自由参数而更加有效。简化方案 B 和简化方案 C 虽然效果略有下降，但是其训练计算量和模型所需存储空间都较小，在实际系统中是更好的选择。

2.3.4 对模型存储空间的改进实验结果

表 2.4 是在标准退化模型基础上，对模型存储空间进行降低之后的音字转换结果。此外本实验还在基本配置为 *PII-450MHz，128MB Ram* 的 *PC* 上进行了运算速度的比较，评价指标为平均每秒输出汉字数。

表 2.4　模型存储空间修正前后音字转换字错误率和系统效率比较

	字错误率（%）	运算速度（字/秒）	存储空间（MB）
基于图灵估计的标准退化模型	6.5	12	427
减小存储空间后的退化模型	6.1	75	111

2.4 综合结论

本章主要论及的是常用的统计语言模型 *n-gram* 训练中的问题及其解决方案。

首先，我们提出了一个完整高效的语料库处理流程，其中应用了分级的词表选择思路，基于合并的最小化困惑度的新词发现策略以及同步分词算法，通过反复修正的过程，实现了一个适于语音识别的 5 万词规模的机器词典 *EasyDic99*。

其次，对 *n-gram* 语言模型训练中的稀疏问题，进行了参数的平滑，并且针对基于图灵估计的退化平滑算法中的不合理因素，提出了一个修正的退化平滑算法，实验结果表明该修正算法与基于图灵估计的标准退化平滑算法相比，性能有明显的提高。

第三，我们对语言模型 *n-gram* 训练中出现的其它问题，如参数溢出和存储复杂度过高等，给出了合理的解决方案。

第三章　统计语言模型在汉语音字转换中的应用

在汉语中，无调音节共有 400 多个，汉字有数万之多，常用的也有 4000 多，因此汉语中的同音字词问题特别严重。在 GB2312-80 中一个音节最多可以对应 116 个汉字，比如说音节“ *yi* ”就对应了 110 个汉字，而在“ 中文之星 ”输入法中，音节串“ *shi shi* ”对应了“ 事实 ”“ 试试 ”“ 实施 ”“ 时时 ”等 21 个不同的词汇。普遍存在的重码问题使得汉语的键盘输入不得不另辟蹊径，但是那些字形码，如五笔输入等，并不是人们最自然的汉语输入方式。因此，如何通过一定的语言知识，将有实际意义的短语或句子从大量同音字词中搭配出来，成为了一个理论研究和实际应用的重要课题，计算机汉语音字转换技术在语音识别及智能汉字键盘输入等领域有着重要的理论价值和广阔的应用背景。

汉语音字转换可以有很多实现方案和技术途径，可以从简单词法的角度进行，也可以与语法规则相结合，甚至还可以加上语义和语用的信息。本章将主要从统计语言模型的角度给出一个完整的解决方案，提出了音节同步算法及其改进版本——基于多栈的音节同步算法。

3.1 音节同步算法[Zheng 99a]

3.1.1 词网格

汉语音字转换技术就是利用语言模型，将一串汉语无调音节串 $S = S_1 S_2 \ldots S_n$ 通过一定的解码算法，转为对应的有意义的词串 $W = W_1 W_2 \ldots W_m$ 或字串 $C = C_1 C_2 \ldots C_n$，其中 $W_i = C_{i_1}^{i_l}$。为了达到这一目的，我们实际上需要寻找最优词串 W^* 使得

$$W^* = \arg\max_w P(W|S) = \arg\max_w P(W)P(S|W) = \arg\max_w P(W)\prod_{i=1}^{m} P(S_{k(i)}^{l(i)}|W_i) \tag{3.1.1}$$

其中

$$P(S_k^l|w) = P(w|S_k^l) = \begin{cases} 1 & w = C_k \ldots C_l, \text{且} S_j \in Syll(C_j),\ j = k \ldots l \\ 0 & \text{其它情况} \end{cases} \qquad (3.1.2)$$

$Syll(c)$ 表示汉字 c 的可能读音集。

实际上，由于汉语存在大量的同音字词，（3.1.1）式的求解需要在一个复杂的词网格中进行的，因此不可能通过简单枚举来得出最优路径。图 3.1 是由“ *yu yin shi bie* ”四个音节形成的词网格示意图。

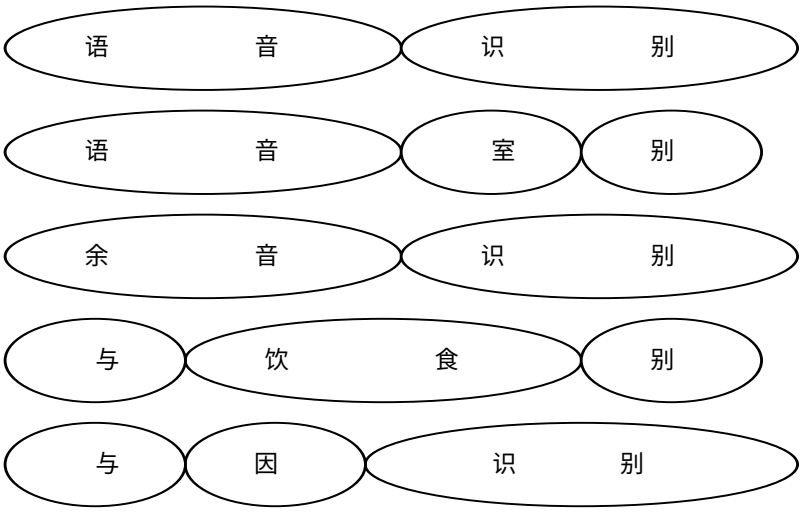

图 3.1　词网格示意

与上一章中描述的同步分词算法类似，我们提出了音节同步算法用于（3.1.1）式的求解。该算法可以很容易地扩展为连续语音识别多遍搜索中的语言处理模块算法，并已在清华大学计算机系研制的 *CDM98* 中实现，取得了较好的使用效果[Zheng 99a]。

3.1.2 问题的求解

为了方便求解算法的描述，我们引入以下几个定义：

用 $Q_{uvw}(i)$ 表示当前处理音节为 S_i 且前导词对为 (u,v) 的搜索路径的得分，该路径下一个可能决定（但未到达边界）的词为 w。

用 $head(w,l)$ 表示词 w 的前 l 个音节组成的音节串。

根据（3.1.1）的要求，$Q_{uvw}(i)$ 的计算可以通过（3.1.3）式以同步处理的方式完成。

$$Q_{uvw}(i)=\begin{cases} H(i-1,u,v) & S_1^{i-1}=W_1^{h-1}\text{且}S_i=head(w,1) \\ Q_{uvw}(i-1) & S_1^{k}=W_1^{h-1}\text{且}S_{k+1}^{i}=head(w,i-k) \\ Q_{uvw}(i-1)\cdot p(w|u,v) & S_1^{k}=W_1^{h-1}\text{且}S_{k+1}^{i}=w \end{cases} \quad (3.1.3)$$

其中

$$H(i,v,w)=\max_u\{Q_{uvw}(i)\} \quad (3.1.4)$$

3.1.3 词法树的引入

对于一般的大词汇表语音识别器而言，受词法约束的词法树（*Lexical Tree*）在进行词内声学搜索约束是必要的。对于汉语音字转换而言，词法树同样可以用来减少冗余信息的存储和对搜索的约束。在我们构造的词法树[Zheng 99b]中，有一个公共的虚根节点（搜索入口），和 N_W 个叶子节点（搜索出口），每个叶子节点代表词表中的一个词。此外每条树枝都代表了相应的音节。所有的搜索路径都从根节点开始，当连续音节串的搜索进入叶子节点的时候，一个新的搜索树将复制给当前路径（视实际情况决定需要复制的内容，可能不需复制，只将相应指针赋值），并重新进入根节点。图 3.2 是一个简单词法树的示意图。由于部分词可以共享词的前缀（*prefix*），因此在利用这样的结构进行搜索时可以减少重复的存储。此外，基于词法树的搜索算法也能在到达叶子节点时加入相应的语言模型得分，满足音节同步算法的需要。

词法树的引入还可以提高判断（3.1.3）式中条件等式的效率。由于在词法树中的每个结点都能够表征出相应的音节串，这样可以很方便地判断当前路径是否可能继续扩展下去，回避了利用线性词表进行判断时所需的大量查找和比较工作。

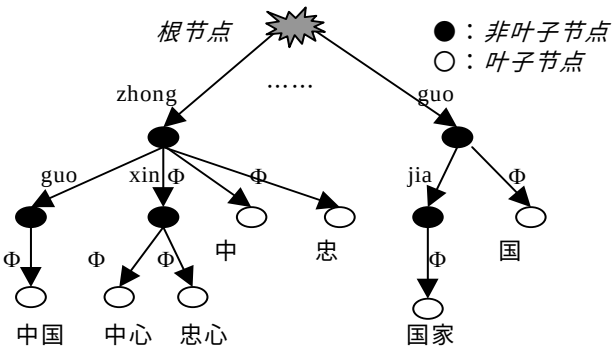

图 3.2　词法树示意

3.1.4 正向匹配中的预测

在引入词法树之后，虽然在同步扩展路径时条件判断的效率大大提高，但无法完全回避产生“ 无效 ”路径的现象：有时可能某条路径在词法树中扩展了好几层结点之后，发现即将处理的下一个音节在当前路径下无法继续扩展。解决的方法就是在每一层扩展时都进行一定深度的预测，检查后续音节能否使当前路径到达叶子结点。如果发现此扩展会导致“ 无效 ”，则提前放弃这个可以扩展的候选路径。

3.1.5 路径剪枝策略

由于汉语同音字词的普遍存在，实用系统必须对候选路径加以控制，通过将搜索范围限制在部分搜索网络而非全部网络中，来保证只有那些最可能的候选路径被考虑。必须注意的是，这个搜索范围的限制对搜索的效率和效果都会产生很大的影响。一般来说，我们在同步算法中采用的剪枝策略通常包括以下两种：

（1） 设定得分阈值

选定一个常数$\alpha(0<\alpha<1)$，在当前时刻 t，在所有可以扩展的路径中选择得分值最大的一条，设其得分为 S_{Max}，只保留得分在 S_{Max} 和 αS_{Max} 之间的路径，其余路径被放弃。

（2） 设定保留路径数阈值

选定一个保留路径数的上限值 β。当扩展出的路径数超过此阈值时，按路径得分的大小顺序仅保留 β 条路径。

3.1.6 算法的描述

在具体实现时，为了提高搜索算法的效率，我们在候选路径 *Path* 中记录了一些必要的信息：

- *CurWord*　　记录当前可能的词（当前音节尚未到达词尾）；
- *CurSyllPos*　　记录此路径当前音节在 *CurWord* 中的位置；
- *WordString*　　记录已解码得到的历史词串；
- *WordStringLength*　　记录历史词串中词的个数；

1. **初始化：** 路径堆栈中仅放入一条历史词串为空，累计语言模型得分为 1 的路径，并置 $i=1$；
2. **扩展路径:** 对堆栈中的每一条路径 *Path[j]*，考虑当前的待处理音节 S_i

   如果预测 *Path[j]*可能继续扩展，则

   2.0　根据 S_i 修改 *Path[j].CurNode*

   2.1　如果 *Path[j].CurNode* 为叶子结点，则

   将 *Path[j].CurNode 对应的词*记录入 *Path[j]. WordString,*

   Path[j].CurNode 重置为根结点,

   增加 *Path[j]. WordStringLength,*

   根据 n-gram 修正 *Path[j].Score；*

   2.2　将修改之后的 *Path[j]*放入新的路径堆栈，等待下一次扩展;

   2.3　将原来的路径从路径堆栈中取出；
3. 交换新老路径堆栈中的路径并排序；
4. 如果 $i=n$，算法停止，堆栈中栈顶 *CurWord* 为空的路径即为最优路径，

   否则 $i=i+1$，转到 2，继续进行。

图 3.3　音节同步算法流程描述

- *Score*　　记录历史词串的累计语言模型得分。

这个数据结构的定义将有利于我们理解如图 3.3 所示的音节同步算法的描述。

3.2 修正的音节同步算法

3.2.1 音节同步算法的不足

音节同步算法在很大程度上避免了大量非最优的候选路径对搜索过程的干扰，但是在实际使用中仍然存在很多问题。如由于音节同步算法中，候选路径的累计得分只在搜索到达一个新的词边界时才做修改，而不同路径到达词边界的时刻不同，因此在搜索对它们进行同步评价并剪枝时会引入偏差。下面这个例子就能充分说明这个问题。

假设我们的输入音节串为“ wai dian ping lun shuo zhong guo ren min zu yi shi qiang ”，其对应的正确词串为 “ 外电\评论\说\中国人\民族\意识\强 ”。

根据音节同步算法，在搜索的过程中将会出现以下两条候选路径：

路径 1：wai-dian-（外电）-ping-lun-（评论）-shuo-（说）-zhong-guo-ren-min

➔外电\评论\说\ (zhong guo ren min)

路径 2：wai-dian-（外电）-ping-lun-（评论）-shuo-（说）-zhong-guo-ren-（中国人）- min　➔外电\评论\说\中国人\+(min)

其中路径 2 为正确候选，然而两条路径的累计语言得分分别为

$$P_1 = p("外电") \cdot p("评论" | "外电") \cdot p("说" | "外电","评论")$$

$$P_2 = p("外电") \cdot p("评论" | "外电") \cdot p("说" | "外电","评论") \cdot p("中国人" | "评论","说")$$

显然 $P_1 > P_2$，如果同时由于路径数目限制，路径 1 和路径 2 只能保留一个的话，正确候选将被剪枝。

3.2.2 多栈解码算法

解决上述问题的一个合理途径是避免这样的路径直接互相比较，我们注意到许多语音识别系统中采用的多栈解码算法（*MSDA, Multi-Stack Decoding Algorithm*）[Ho 98, Renals 99]。

在多栈的同步解码算法中，原有的一个路径堆栈扩充成了 N 个堆栈 $Stack(i)\quad i=1,\cdots,N$，相同堆栈中存放的路径处理过的音节都相同，而在不同的堆栈中则存储了不同时刻到达词边界的候选路径。此外，在搜索过程中，也必须保证不同堆栈中的路径之间不会因为互相比较而剪枝。图 3.4 是多栈解码算法在音字转换中应用的一个简单示例。

多栈解码算法与单栈的音节同步算法相比，虽然所有路径还是每次只同步处理一个音节，但由于扩展之后的路径依情况分栈存储，实际上相当于一次处理若干个音节。

与单栈的解码算法相同，多栈解码算法也是一次仅处理一个活动堆栈。但它可以根据具体情况将新扩展出的路径分别推入不同堆栈中，而且可以根据不同堆栈所处理路径的不同情况在剪枝策略上予以区别对待，如调整得分阈值α或保留路径数阈值β，从而使得算法的自由度更大。

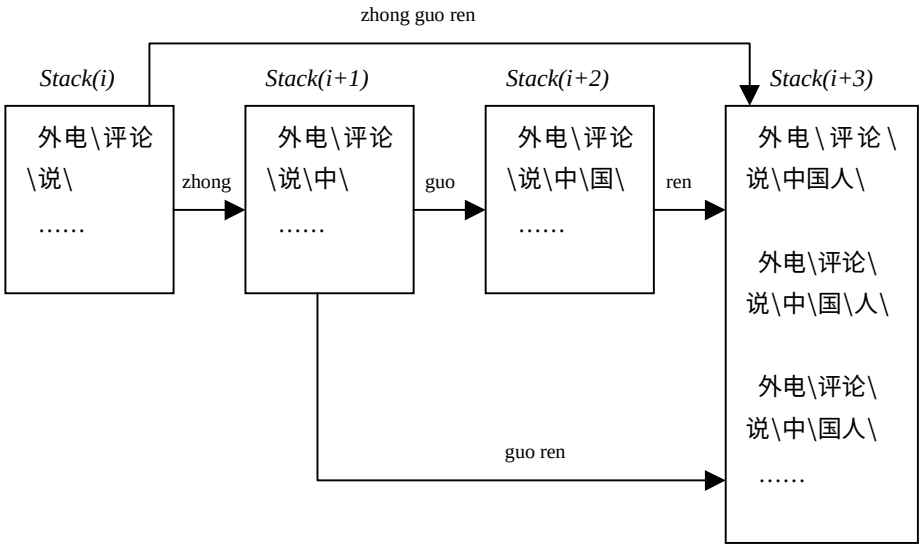

图 3.4　多栈解码算法示例

1. **初始化：**$i=0$，$k=K-1$把当前结点为词法树根结点，且累计语言得分为 *1* 的初始路径放入 *stack(0)*。
2. **扩展路径:** 从当前活动堆栈 *stack(k)*中依次取出每条路径 *Path[j]，考虑后续的音节*S_i

   如果预测 *Path[j]*可能继续扩展，则

   2.0　根据S_i修改 *Path[j].CurNode*

   2.1　如果 *Path[j].CurNode* 为叶子结点，

   则

   将 *Path[j].CurNode 对应的词*记录入 *Path[j]. WordString,*

   Path[j].CurNode 重置为根结点，

   增加 *Path[j]. WordStringLength,*

   根据 n-gram 修正 *Path[j].Score；*

   将修改后的 *Path[j]放入 stack(K)*

   否则

   将修改后的 *Path[j]放入 stack(k+1)*

   2.2　将原来的路径从路径堆栈 *stack(k)*中取出；

   2.3　如果$k=0$，转到 3，否则$k=k-1$，转到 2;

   3.　将 *stack(K)*中所有路径取出放入 *stack(0)*中，并将每个堆栈中的路径依累计得分排序;

   3.　如果$i=n$，算法停止，堆栈 *stack(0)*中栈顶的路径即为最优路径，

   否则$i=i+1$，$k=K-1$转到 2，继续进行。

图 3.5 基于多活动栈的音节同步算法

3.2.3 基于多活动栈的音节同步算法

多栈解码算法也有许多不足。首先，由于多栈解码算法在同步处理时需要判断当前待处理音节串中后续的若干音节能否成词,如图 3.4 中的音节串‘ *zhong guo ren* ”和“ *guo ren* ”等，在处理时查找出所有可能的后续词需要巨大的工作量。另外，由于理论上对每个可能的后续词而言，都会产生出一条候选路径存

储在相应堆栈中，从存储空间上来说比较冗余，并没有利用到词法树的全部优点。

因此，我们针对音节同步算法和多栈的解码算法各自的不足和优点，提出了一个基于多活动栈的音节同步算法(*MAS-SSS, Multi-Active-Stack based Syllable Synchronous Search*)。

在修正的音节同步算法中，我们考虑到词表中的词长是有一定限制的（假设词表的词长范围为从 *1* 到 *K*），因此可以只保留少数堆栈 $Stack(k)\quad (k=0,\cdots,K)$。其中 $Stack(k)\quad (k=0,\cdots,K-1)$ 中存放的路径当前都停留在词法树的第 k 层结点上，而 $Stack(K)$ 用来作为临时缓存。修正之后的算法流程见图 3.5，从描述上来看，与图 3.3 所示单栈音节同步算法的主要差别在于对扩展路径部分所进行的修改。表 3.1 是单栈音节同步算法、多栈的解码算法以及多活动栈的音节同步算法三者特点之间的具体比较。

表 3.1　音节同步算法、多栈解码算法以及多活动栈音节同步算法的特点比较

	需要的 路径堆栈	每步处理时 的活动堆栈	每步处理时 考虑的音节数	路径剪枝时 有无偏差*
单栈的 音节同步算法	1	1	1	有
多栈的 解码算法	N	1	多个	无
多活动栈的 音节同步算法	K+1	K	1	无

*此表中所指的偏差仅限于本节第一部分描述的错误

从表中可以看出，多活动栈的音节同步算法在一定程度上结合了另外两个算法的长处。但在实际应用中，每次需要同时处理 K 个活动堆栈对于堆栈的维护工作还是非常麻烦。因此，我们更倾向于使用其进一步简化版本—双活动堆栈的音节同步算法(*BAS-SSS, Bi-Active-Stack based Syllable Synchronous Search*)。该简化算法将所有非停留在根结点的路径，即 $Stack(k)\quad (k=1,\cdots,K-1)$ 中的所有路径，合并放入一个新的堆栈，该堆栈被称作完全路径堆栈（*CWPS, Complete Word Path Stack*）；而将所有当前停留在根结点的路径，或者说 $Stack(0)$ 中的所有路径，放入另一个堆栈，该堆栈我们称作部分路径堆栈（*PWPS, Partial Word*

Path Stack）；并使图 3.5 所示的搜索流程在这两个堆栈之间进行。

3.3 实验结果

搜索算法比较实验的实验环境和测试数据都同上章音字转换测试，比较测试是在音节同步算法和双活动堆栈的音节同步算法之间进行的。为了准确反映算法之间的性能差别，两个搜索算法的音字转换系统使用的语言模型完全一致。而且，两个系统的剪枝都只根据上述的第二种策略，即限定路径总数的上限。在本实验中，应用音节同步算法的系统中，唯一的活动堆栈容量上限为 20 条，应用双活动堆栈的音节同步算法的系统中，每个活动堆栈的容量上限都设为 10 条，这样两个系统在侯选路径的存储空间上也基本一致。比较结果见表 3.2。从表中结果我们看到，修正过后的音节同步算法的错误率下降了大约 25%。

表 3.2　单栈的音节同步算法与双活动栈的音节同步算法性能比较

	单栈的音节同步算法(SSS)	双活动栈的音节同步算法(BAS-SSS)
字错误率	6.1%	4.6%

对于双活动栈的音节同步算法而言，两个堆栈的容量上限对实验结果也有一定的影响。在总路径数一定的情况下（本实验中为 20），我们可以通过调整两个堆栈之间的比例，来寻求更优的实验性能。表 3.3 就是这个比较实验的结果。从中我们可以看到，两个堆栈所需的空间基本相同时系统的性能最好。

表 3.3　双活动栈的音节同步算法中堆栈容量对性能影响的比较

堆栈容量上限（总数为 20）		字错误率
完全路径堆栈	部分路径堆栈	
10	10	4.5%
15	5	4.6%
5	15	4.8%

3.4 综合结论

本章提出的基于多栈的音节同步算法与同类算法相比，具有计算效率高，存储冗余度小，同时准确率高的优点。

首先，它引入了词法树的概念，使搜索过程中的词法约束方便地应用到路径的评价中。

其次，同步搜索的方法实际上是一种广度优先的算法，它可以使系统避免大量的预测计算。

第三，它是一个基于多个活动堆栈的算法，结合了栈解码算法的部分特点，在广度优先的同时也具有一定的深度优先的功能。部分避免了纯粹广度优先算法的盲目性。

最后，从实验结果来看，双活动堆栈的同步搜索算法可以通过调整部分参数来寻求最佳实验效果。

第四章　连续语音识别中搜索算法对统计语言模型的利用

4.1 连续语音识别中的搜索策略

4.1.1 连续语音识别的问题描述

汉语语音听写机（*CDM，Chinese Dictation Machine*）是非特定人、大词汇量的连续或连接词识别系统，目的是由计算机将人的语流转化为相应的文本信息。在当今人与计算机交互日益频繁的条件下，探索高效而自然的交互方式是人们不断努力的目标。汉语语音听写机正是这样一种十分有潜力的人机交互系统，它可望把人从不自然的信息输入方式中解放出来，从而推进计算机的应用和发展。

对于连续语音识别来说，语言模型是一个十分重要的问题。由于声学信号的动态改变，瞬时和随机性，单靠声学模式的匹配与判断，是很难完成语音到语言文字的正确转换的。较高层次的语言知识的利用可以大幅度地减小声学模式匹配的模糊性，从而提高听写机的识别正确率。

假设输入的连续语音信号是$S = S_1 S_2 S_3 \cdots S_T$，语音识别器或听写机的目的就是寻找到最佳的词串$\hat{W} = W_1 W_2 \cdots W_n$，使得联合概率$P(S,\hat{W}) = P(S|\hat{W}) \cdot P(\hat{W})$达到最大。

4.1.2 连续语音识别中的搜索策略

在大词汇量连续语音识别系统中，实现一个快速有效的搜索算法对整个系统的最终性能起着至关重要的作用。一个好的算法需要在搜索的过程中尽量使用准确的联合概率来衡量候选路径，至少要保证联合概率较大的路径尽可能地保留下来，直至搜索任务的全部结束。近年来，国内外研究者们在此方面作了大量的研究工作，提出了不少实用的搜索算法。这些搜索算法，可以根据不同的研究角度，分为不同的类别。

根据搜索的空间，它们可以分为基于词法树的搜索和基于线性词表的搜索，这两种搜索算法的差别在于前者利用了词汇的发音具有大量的共享前缀的特性，从而减少了声学得分的重复计算并被大量采用。

从搜索时各种信息使用时机的角度，可分为一遍搜索（*One-pass*）[Ney 92]和多遍搜索（*Multi-pass*）[Li 96]，一遍搜索是将所有可利用的信息按照某种准则集成为一个合一的路径评价标准，通过一遍扫描语音信号便给出侯选结果的方法，而多遍搜索则是将这些信息根据不同的需要，在搜索的不同阶段，单独作为或部分集成为路径的评价标准，进行多遍扫描给出侯选的搜索算法。

从历史路径保存的存储结构来看，可分为 *N-Best* 和词图（*Word Graph*）算法[Ortmanns 97]，后者可以看作是前者在历史路径存储方式上做的一个较大改进，它以一个较为紧凑的结构存储了更多的路径。

从搜索时词法树的使用方式，可以分为基于词（*Word-conditioned*）[Ney 92]和基于时间（*Time-conditioned*）[Ortmanns 96a]两种方法，在基于词的搜索算法中，每条假设的搜索路径都取决于前面已识别出的词，且词的边界已经在前面的搜索过程中得到确认，而基于时间的搜索算法中，每条假设的搜索路径取决于前一个词的结束时间，也就是说，词和词之间的分界在搜索的后期才完成。

根据搜索时考虑的语音信号的范围，可以分为 *A** 算法[Paul 91]和时间同步算法（*Time-synchronous*）[Lee 89]，前者利用了一个对后续信号的预测启发函数来完成对部分路径的衡量，并寻求最优解；而后者评价当前部分路径的依据只是当前时间前的语音信号，希图以较小的代价获得较好的次优解。

总而言之，这些算法都是为了在保证系统的实时性和限制系统的时空开销的前提下，对搜索的过程进行了一定的简化，使搜索的复杂度和最后的系统性能之间达到平衡。综合比较上述算法，按照从左向右拓扑结构组织的基于词的时间同步一遍搜索算法在系统开销限制较多的情况下仍能够取得较好的效果而被许多系统所采用。本章将具体介绍在此基础之上发展的综合同步搜索算法（*ISS，Integrated Synchronous Search*），以及一个应用了综合同步搜索算法的听写机雏形 *EasyTalk2000* 的实现。

4.2 语言模型在同步搜索算法中的使用

如前所述，利用词法树的时间同步的束搜索算法有两种不同的实现版本：基于词的同步算法和基于时间的同步算法。它们都是从左向右扫描一遍的搜索算法，而且都在搜索中合理地利用了统计语言模型。两者的不同只在于它们的搜索空间组织方式上的不同。在基于词的方法中，我们通过使用基于词的词法树来决定搜索空间，也就是说，对于语言模型所需要的每个历史词对，我们复制一个单独的词法树拷贝用于搜索。而在基于时间的方法中，对每个可能的词开始的时间都会有不同的词法树拷贝。

4.2.1 基于词的搜索算法

如果搜索时采用了词法树，我们将面临的一个问题就是，在搜索中词只能在到达词法树的叶子结点时才能确定。为了解决这个问题，，我们必须对每个可能的 *n-1* 元词串保留一个词法树的单独拷贝。对于 *trigram* 而言，我们需要保证词法树的单独拷贝与路径的前两个词相关。

为了方便表述，我们引入几个定义：

$Q_{uv}(t,s)$ 表示前导词对为*(u,v)*的候选路径在词法树的约束下，在 t 时刻到达状态 s 的最佳得分；

$B_{uv}(t,s)$ 表示前导词对为*(u,v)*，且在词法树的约束下，在 t 时刻到达状态 s 的最佳路径的起始时间。

在词法树内部搜索时的动态规划迭代计算公式为

$$Q_{uv}(t,s)=\max_{\sigma}\{q(x_t,s|\sigma)\cdot Q_{uv}(t-1,\sigma)\} \tag{4.2.1}$$

$$B_{uv}(t,s)=B_{uv}(t-1,\sigma_{uv}^{\max}(t,s)) \tag{4.2.2}$$

其中 $q(x_t,s|\sigma)$ 是 HMM 声学模型的输出概率或转移概率，$\sigma_{uv}^{\max}(t,s)$ 是在 t 时刻到达状态 s 及前导词对为*(u,v)*的情况下的前一时刻最优状态。

在词的边界，路径需要进行根据前导的词进行归并

$$H(v,w:t)=\max_{u}\{p(w|u,v)\cdot Q_{uv}(t,S_w)\} \tag{4.2.3}$$

其中 S_w 是词 w 在词法树中的最后一个状态。

在开始进入新的词法树搜索，必须进行初始化 $Q_{uv}(t,s)$ 和 $B_{uv}(t,s)$

$$Q_{uv}(t-1,0)=H(u,v:t-1) \tag{4.2.4}$$

$$B_{uv}(t-1,0)=t-1 \tag{4.2.5}$$

4.2.2 基于时间的搜索算法

基于时间的搜索算法是同步算法的另一个变形。和基于词的搜索不同，基于时间的搜索的侯选路径取决于词的开始时间。

下面以 *bigram* 语言模型的应用为例来说明这个算法。同样地，我们引入如下定义：

$h(w;\tau,t)$ 表示词 w 生成声学特征序列 $x_{\tau+1}\cdots x_t$ 的概率；

$H(v:\tau)$ 表示由以 v 结尾的词串生成声学特征序列 $x_1\cdots x_\tau$，且 τ 为词边界的概率；

$Q_\tau(t,s)$ 表示 τ 时刻开始进入词法树且尚未到达叶子结点的路径的得分。

首先进行初始化

$$Q_{t-1}(t-1,s)=\begin{cases}\max\limits_u H(u;t-1) & \text{如果} s=0\\ 0 & \text{如果} s>0\end{cases} \tag{4.2.6}$$

相应的词内搜索公式为

$$Q_\tau(t,s)=\max_\sigma\{q(x_t,s|\sigma)\cdot Q_\tau(t-1,\sigma)\} \tag{4.2.7}$$

注意到在基于时间的搜索算法中，侯选路径的得分 $Q_\tau(t,s)$ 是根据最优的前导词计算得来的。因此，基于词的搜索中所需要的 $B_{uv}(t,s)$ 在这里并不需要。

然而，在词边界时语言模型概率的加入变得更为复杂。为了保证语言模型加入的合理性，在搜索到达词边界的时候，还需要进行一些附加的计算。为了方便计算，我们引入了一个新的定义：

$$\hat{H}(v,w:t)=\max_\tau\{\frac{H(v;\tau)}{\max\limits_u H(u;\tau)}\cdot h(w;\tau,t)\} \tag{4.2.8}$$

对每个可能的词对*(v,w)*找到最佳的时间分点和得分。公式中归一的原因是每个搜索树都是由最优的前导词得来，对时间取最大值 τ 是因为路径得分依赖于搜索进入词法树的开始时刻。

在每个词对的最佳时间分点和得分找到后，对前导词 v 取最大值可以求出最佳前导词

$$H(w:t)=\max_{v}\{p(w|v)\cdot\hat{H}(v,w:t)\} \qquad (4.2.9)$$

基于时间的搜索算法已经被用于连接数字识别和词网格的生成中。

下面提出的综合同步搜索算法是在基于词的搜索算法基础之上所作出的改进。

4.3 语言模型概率的折入

前节所述的两个同步算法中，语言模型的加入时机都是在搜索达到词边界时，但是这将意味着，路径在从一棵搜索树的根结点至叶子结点的阶段中，整个搜索过程将只由声学模型和简单的词法约束来控制，即如（4.2.1）或（4.2.7）式所示。它们在词内搜索时的路径评价函数实际上退化成为了

$$P(S_1S_2S_3\cdots S_T|W_1W_2\cdots W_n)\cdot P(W_1W_2\cdots W_{n-1})=\eta(S,W)\cdot P(S,W) \qquad (4.3.1)$$

比设想的 $P(S,W)$ 多了一个不定的系数 $\eta(S,W)$。而且由于一般一个词法树从根结点到叶子结点所跨越的声学时间段相对都比较长，在这之间足以发生很大的偏差。

为了在搜索中保证评价函数的合理性，**语言模型折入概率**被引入到连续语音的搜索算法中，定义如下：

$$h_{uv}(n|m)=\sum_{w\in\Pi(n)}p(w|u,v)\Big/\sum_{w\in\Pi(m)}p(w|u,v) \qquad (4.3.2)$$

其中 *m,n* 为词法树中的一对结点，结点 *n* 是结点 *m* 的子结点；*Π（n）*表示由结点 *n* 可以到达的叶子结点（词）的集合，称为**叶子集**。这个概率也可以近似解释为语言模型在词法树内部搜索时对结点跳转的惩罚。这一定义被称为概率的**折入**[Ortmanns 96b]（*Factorization*）。

为了简化折入概率的计算，（4.3.2）式中的求和计算可以用取最大值来代替，即定义

$$h_{uv}(n|m) = \max_{w\in\Pi(n)} p(w|u,v) \Big/ \max_{w\in\Pi(m)} p(w|u,v) \qquad (4.3.3)$$

就向 *Viterbi* 算法中用最大值来代替求和一样，性能不会下降。而且这个简化带来的另一个好处是树中的许多结点的折入概率值都为 1（实际上只会有 N_w 个结点的值不为 1），从而可以简化存储的结构。

由于在树状结构中，从任一结点可以到达的叶子结点的集合相当于从该结点的所有子结点出发可到达的叶子结点集合的总集，即 $\Pi(m) = \bigcup_{n\in S(m)} \Pi(n)$，其中 *S(m)*表示结点 *m* 的子结点集。因此，在实际计算时，还可以用一个简单递归公式来进一步减少求和或取最大值的运算量，计算过程如（4.3.4）～（4.3.6）所示,

$$g_{uv}(m) = \begin{cases} \sum_{n\in S(m)} g_{uv}(n) & \text{如果}m\text{为非叶子结点} \\ p(w|u,v) & \text{如果}m\text{为叶子结点} \end{cases} \quad \text{（求和）} \qquad (4.3.4)$$

$$g_{uv}(m) = \begin{cases} \max_{n\in S(m)} g_{uv}(n) & \text{如果}m\text{为非叶子结点} \\ p(w|u,v) & \text{如果}m\text{为叶子结点} \end{cases} \quad \text{（最大值）} \qquad (4.3.5)$$

$$h_{uv}(n|m) = g_{uv}(n)/g_{uv}(m)。 \qquad (4.3.6)$$

4.4 语言模型和声学模型的预测

4.4.1 声学模型的预测

声学模型预测（*AMLA，Acoustic Model Look-Ahead*）指的是每次在搜索即将从一个结点跳转入下一个结点时，即开始扩展新的音节前，进行的声学上的预测。声学模型预测时暂时不直接扩展路径，而是先对扩展之后的路径进行是否会被剪枝的检查。如果检查通过，则该跳转被允许进行，否则的话，则被禁止，从而减少了大量无效的计算。

应该指出，这种预测检查并不能保证十分准确。一般来说，我们会首先计算出一个尽可能准确的近似声学得分，称作**声学预测得分**。然后在预测的过程

中，声学预测得分与当前路径的历史得分相结合，通过一定的比较准则来完成对路径扩展可能性的检查。具体算法描述如下：

设 $\tilde{q}(n,\gamma;t,\Delta t)$ 为声学预测跨越时间从 t 到 $t+\Delta t$，结束状态为 γ 的路径的预测得分，其中 n 是 γ 所属的结点。

然后对给定的时间间隔 τ 和每个可能的三元组 $(u,\ v,\ \gamma)$，计算出一个与 $Q_{uv}(t,s)$ 代表意义不同的评价函数值

$$\tilde{Q}_{uv,n}(t+\tau)=\max_{\gamma}\tilde{q}(n,\gamma;t,\tau)\cdot H_{uv}(t,m) \qquad (4.4.1)$$

通过（4.4.4）式可以将声学预测得分与当前路径的得分组合在一起。

求出 $\tilde{Q}^{*}=\max_{u,v,r,n}\tilde{Q}_{uv,n}(t+\tau)$ 并确定阈值α，检查所有的 $\tilde{Q}_{uv,n}(t+\tau)$，并拒绝掉满足

$$\tilde{Q}_{uv,n}(t+\tau)<\alpha\cdot\tilde{Q}^{*}_{uv} \qquad (4.4.2)$$

的结点的扩展可能。

4.4.2 语言模型的预测

由于语言模型对搜索中可能的后续音节可以起到很大指导和预测作用，因此一般来说，在搜索中我们希望精确的语言模型对搜索过程的影响越早越好，这就是**语言模型的预测**[Ortmanns 96b]（*LMLA，Language Model Look-Ahead*）。在引入语言模型折入概率之后，不需到达词法树的叶子结点，我们就可以很方便地根据折入概率实现语言模型对搜索的预测。

在词法树非叶子结点的任意状态和结点跳转时，搜索的评价函数从（4.2.1）式改为（4.4.3）式

$$Q_{uv}(t,s)=h_{uv}(n(s)\,|\,n(\sigma))\cdot\max_{\sigma}\{q(x_t,s\,|\,\sigma)\cdot Q_{uv}(t-1,\sigma)\} \qquad (4.4.3)$$

其中，$n(s)$ 表示状态 s 所在的结点。

由于每个结点包括叶子结点的语言模型概率都被语言模型的折入概率考虑

在内，因此，在到达叶子结点时不需要对 *n-gram* 加以单独考虑，（4.2.3）式可以简化为（4.4.4）式

$$H(v,w:t)=\max_{u}\{Q_{uv}(t,S_w)\} \tag{4.4.4}$$

4.4.3 语言预测与声学预测的结合

在语言模型的预测过程中，经过的词法树每个结点都需要计算语言模型折入概率，而从（4.3.3）式我们知道，该概率的计算复杂度与结点的叶子集规模成正比。因此，伴随着语言模型预测技术的除了搜索精度的提高之外，还有大量的求和计算或比较，以及对 *trigram* 概率值的频繁读取。虽然可以只根据具体情况计算出需要的折入概率，而无需事先计算出所有的折入概率，但是对于一个实时系统而言，仍然是很大的负担。

声学模型的预测不需要附加的大运算量问题。虽然声学的预测也需要计算一个预测得分，但是由于与路径实际扩展过程中使用的精细声学模型无关，可以采用近似的快速算法或简单的声学模型来完成，而且即使两者采用相同的声学模型和计算方法，计算结果也可以缓存下来以供以后的使用。

声学模型的预测在一定程度上可以缓解语言预测计算上的膨胀问题，因为声学预测之后可以减少许多可扩展的结点。综合同步搜索算法中就同时结合了两种不同的预测技术，我们称作**分阶段的联合预测技术**（*SLA，Stage-based Look-Ahead*）。

首先，为了避免声学预测得分的计算花销被浪费，在预测和路径扩展时综合搜索算法采用了同样的声学模型和计算方法，使得 $\widetilde{Q}_{uv,n}(t)=Q_{uv}(t)$。在声学预测的过程中，式（4.2.1）所示的同步算法和式（4.4.2）所示的剪枝策略被联合使用。当声学预测结束时，得到的 $\widetilde{Q}_{uv,n}(t)$ 又作为语言预测的初始得分。

在 *EasyTalk2000* 中，我们规定搜索路径的当前声学状态到达某一阈值 M 时，声学预测结束，因此，分阶段的联合预测技术可以表达为（4.4.5）~（4.4.12）

初始化：
$$Q_{uv,n}(t-1,0)=\begin{cases}H(u,v:t-1) & n\text{为根结点}\\ Q_{uv,m}(t-1,S_m) & \text{否则}\end{cases} \quad (4.4.5)$$

$$B_{uv,n}(t-1,0)=t-1 \quad (4.4.6)$$

声学预测：
$$Q_{uv,n}(t,s)=\max_{\sigma}\{q(x_t,s|\sigma)\cdot Q_{uv,n}(t-1,\sigma)\} \qquad s<M \quad (4.4.7)$$

$$B_{uv}(t,s)=B_{uv}(t-1,\sigma_{uv}^{\max}(t,s)) \qquad s<M \quad (4.4.8)$$

语言预测：
$$Q_{uv,n}(t+1,M)=h_{uv}(n|m)\cdot q(x_{t+1},M|M-1)\cdot Q_{uv,n}(t,M-1) \quad (4.4.9)$$

$$Q_{uv,n}(t,s)=\max_{\sigma}\{q(x_t,s|\sigma)\cdot Q_{uv,n}(t-1,\sigma)\} \qquad \sigma>=M \quad (4.4.10)$$

$$B_{uv}(t,s)=B_{uv}(t-1,\sigma_{uv}^{\max}(t,s)) \quad (4.4.11)$$

词结束：
$$H(v,w:t)=\underset{u}{Max}\{Q_{uv,n}(t,S_n)\} \quad (4.4.12)$$

其中 n 为词 w 最后一个音节相对应的结点，S_n 为结点 n 的末状态。

与上章所述多栈的音节同步算法相似，综合搜索算法中任意时刻处于不同预测阶段的路径之间不能互相比较，因此它可以用一个多栈的算法来方便实现。

4.5 搜索中的剪枝策略

剪枝在语音识别的搜索中必不可少，一般来说，系统使用最多的是第三章中提到的 *Beam Search* 中两种常用的阈值剪枝。*EasyTalk2000* 中，还针对不同的情况采用了一些其它合理的剪枝方法。这些剪枝算法按照一定的原则，在搜索过程的不同阶段应用，联合起来相当于一个多层的网络对候选路径进行过滤，被称为**多层剪枝策略**（*MTP，Multi-tier Pruning*）。总的来说，采用的剪枝方法有：

1、路径得分阈值限制。

确定阈值α，并求得该时刻所有路径得分中的最大值$Q^*(t)=\underset{u,v,n,s}{Max}Q_{uv,n}(t,s)$，将不满足$Q_{uv,n}(t,s)>\alpha\cdot Q^*(t)$的路径剪枝掉。实际上，由于采用了两种不同的预测方案，在系统中也相应地选取了不同的阈值α_a（针对声学预测）和α_l（针对

语言预测）对处于不同预测阶段的路径分别求最大值和剪枝。

2、路径数目限制。

确定阈值β，对任意时刻的路径都只保留不超过β条。同样道理，对处于不同预测阶段的路径选取了不同的 β_a 和 β_l。

3、声学帧得分阈值。

确定阈值 α_f，考虑当前帧的声学得分 $q(x_t, s|\sigma)$，求最大值 $q^*(t) = \max_s q(x_t, s|\sigma)$，只保留满足 $q(x_t, s|\sigma) > \alpha_f \cdot q^*(t)$ 的若干条路径。

4、基于语言模型的分级剪枝策略（*LMRP, Language-Model-Rank based Pruning*）。

虽然在语言预测之前采用了声学模型的预测，但在词法树第一层结点处，由于每个结点的叶子集规模都很大，因此语言模型折入概率的计算还是很大。针对这一情况，我们提出了分级剪枝策略对该层结点作了特殊处理。

首先，到达词边界回到根结点的待扩展路径可以根据得分分为若干个集合 P_i，每个集合都指定一个阈值 α_i。然后，对集合 P_i 中的某条路径而言，考虑第一层结点中的每一个结点 m，如果 $g_{uv}(m) < \alpha_i$，则禁止这条路径扩展至该结点。

在实际计算中，由于 $g_{uv}(m)$ 只是用来判断以结点 m 为起始的词中是否有在历史环境下的常用词汇，因此也可以用方便易算的 *unigram* 概率折入值 $g(m)$ 来作为 $g_{uv}(m)$ 的近似估计。

4.6 实验结果及结论

4.6.1 性能比较

实验使用的基本平台 *EasyTalk2000* 所采用的声学特征包含 16 维 *MFCC* 和能量，以及它们各自的自回归系数，共 34 维。声学模型采用的是分段的高斯混合分布模型（*GMSM, Gausian Mixture Segmentation Model*）[mou 98]，该模型分段数（相当于 *CHMM* 中的状态数）为 6，每个状态内的混合数为 16。训练数据是 863 提供的 80 名男声数据；测试集共有三个，都为男声朗读发音，其中测试集 A 有 519 个句子，一人发音；测试集 B 有 240 个句子，三人发音，每人

80 句；测试集 C 有 520 个句子，一人发音。

参照系统 *EasyTalk99*[Zheng 99c]采用的是多遍的搜索算法，它首先利用声学模型和帧同步算法生成一个带匹配概率的音节网络，然后利用语言模型和多栈的音节同步算法在音节网络中完成候选的选择工作。

两个系统在除搜索算法和结构框架外，其它各方面条件完全相同。

下列所有表中的测试结果是听写机最后输出句子的字错误率，包括插入、删除和替换错误。如不加注明，所有的综合搜索算法不采用基于语言模型的分级剪枝策略，实验中的其它参数为 $\beta_a = \beta_l = \beta/2$，$\beta = 200$，预测分界状态 $M = 3$。

1、综合搜索算法与多遍搜索算法比较

表 4.1　综合搜索算法与多遍搜索算法性能比较

	EasyTalk2000	EasyTalk99
测试集 A	10.8%	11.9%
测试集 B	23.3%	32.1%
测试集 C	17.8%	26.3%

结论：在所有的测试集上综合搜索算法都优于多遍搜索算法，平均错误率减低约 23%。

2、分阶段的联合预测技术对综合搜索算法性能的影响

表 4.2　调整联合预测算法中的参数 *M* 对 *EasyTalk2000* 系统性能的影响

	测试集 A	测试集 B
M=1	11.0%	23.1%
M=3	10.8%	23.3%
M=5	10.7%	23.2%

结论：联合预测技术中引入语言模型的时机早晚对系统性能影响不大，原因是语言模型折入概率只是对相应结点的语言模型平均估计，并不能准确描述同一结点不同状态之间的差别。

3、剪枝算法参数对综合搜索算法性能的影响

表 4.3　调整剪枝算法中的参数β 对 *EasyTalk2000* 系统性能的影响

	测试集 A	测试集 B
β=100	12.1%	27.3%
β=200	10.8%	23.3%
β=400	8.3%	19.6%

表 4.4　分别调整剪枝算法中的参数 β_a 和 β_l 对 *EasyTalk2000* 系统性能的影响

	测试集 A	测试集 B
β_a =50， β_l =150	11.9%	27.2%
β_a =100， β_l =100	10.8%	23.3%
β_a =150， β_l =50	9.7%	21.8%

4、采用基于语言模型的分级剪枝策略对综合搜索算法性能的影响

表 4.5　采用基于语言模型的分级剪枝策略对 *EasyTalk2000* 系统性能的影响

	测试集 A	测试集 B
不采用 *LMRP*	10.8%	23.3%
采用 *LMRP*	10.8%	23.4%

结论：采用语言模型的分级剪枝策略虽然抛弃了大量的侯选路径，但是从识别率来看，对系统并无太大影响。

4.6.2 搜索复杂度的评价指标

为了客观地评价搜索算法的搜索复杂度，针对综合搜索算法，我们提出了下列三个评价指标。

每帧活动声学状态数：记录搜索过程中每时刻所有待扩展路径中不同的声学状态（包括结点和结点内的 *HMM* 状态）数目，此指标代表搜索过程中每帧的声学模型概率计算量；

每帧语言模型折入概率计算次数：此指标代表搜索过程中每帧的语言模型折入概率的计算量；

每帧活动词法树数：记录搜索过程中每时刻所有待扩展路径中不同的前导词对数目，此指标代表搜索过程中每帧的语言模型折入概率的存储复杂度；

上述三个指标的值越大，说明搜索所需的计算量和存储空间越大，搜索算法效率越低。此外，第一个指标——每帧的活动状态数——可以分别记录剪枝

前后的结果，以反映剪枝算法的效率。

4.6.3 搜索复杂度的比较

复杂度的比较实验的测试数据于搜索性能测试相同，下列所有表中的值都是帧平均数。

1、分阶段的联合预测技术对搜索算法复杂度的影响

表 4.6　分阶段联合预测算法对搜索复杂度的影响

	活动声学状态数	折入概率计算次数	活动词法树数
LMLA	250.1	801.7	3.9
SLA，M=1	252.9	759.4	3.2
SLA，M=3	253.6	681.9	2.7
SLA，M=5	260.8	623.5	2.4

结论：采用了联合预测技术之后，系统复杂度的后两项指标比仅采用语言模型预测要小了许多，与后三行实验数据结合起来，可以发现，语言模型预测开始得越早，声学模型计算量越小，但语言模型的计算量越大。实际系统中，两者之间需要有一个平衡。

2、采用基于语言模型的分级剪枝策略对搜索复杂度的影响

表 4.7　基于语言模型的分级剪枝策略对搜索复杂度的影响

	活动声学状态数	折入概率计算次数	活动词法树数
不采用 *LMRP*	253.6	681.9	2.7
采用 *LMRP*	252.4	672.0	2.7

结论：采用基于语言模型的分级剪枝策略可以降低搜索复杂度，与表 4.5 联合起来，说明该剪枝策略具有使用价值。

3、剪枝算法对搜索复杂度的影响

图 4.1 是 *EasyTalk2000* 中使用了所有的剪枝算法后的效果。图中虚线是如不剪枝，每帧需计算的声学概率数，实线为剪枝后实际需要计算的声学概率数。

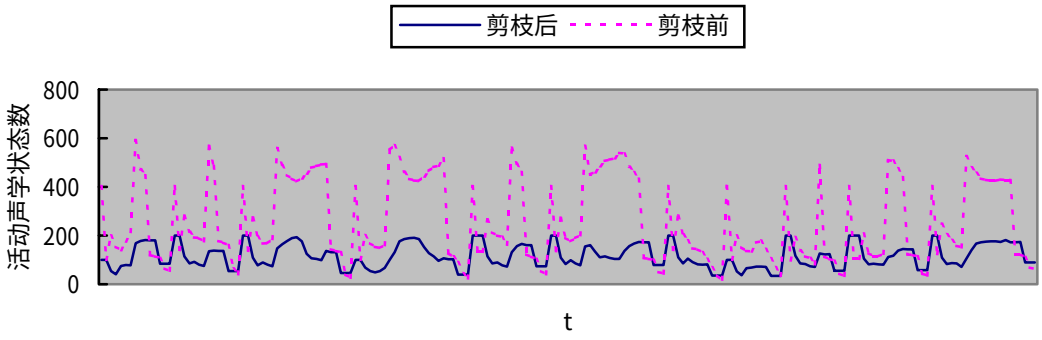

图 4.1　剪枝算法对搜索复杂度的影响

结论：多层剪枝策略可以使系统的搜索复杂度大大下降。

4.6.4 综合结论

本章提出的综合同步搜索算法实际上是一个一遍的多阶段同步搜索算法，其基本特点如下：

1、基于从左向右拓扑结构同步进行；

2、语言模型概率已折入到词法树的每个结点，而不必在词结束时才得到确认和处理；

3、在语言模型的预测基础之上加入声学模型的预测，使语言模型和声学模型的影响分阶段地在搜索过程中起主导作用，同时减少语言模型折入概率的计算量和存储空间；

4、根据语言模型的近似概率，进行路径的分级剪枝；

5、在不同的时机，针对不同的情况，应用不同的剪枝算法，形成了多层的剪枝策略。

其中 3~5 条是本算法的特色所在。此外，本章还针对搜索算法的复杂度提出了三项评价指标，用于评价搜索算法的效率。

参考文献

1. **Brown, P.F., Della Pietra, V.J., de Souza, P.V., etc., (Brown 1992)** "Class based n-gram models of natural language", *Computational Linguistics*, v18, n4, pp.467-479, 1992.
2. **Federico, M., Cettolo, M., Brugnara, F., etc., (Federico 1995)** "Language modeling for efficient beam-search", *Computer Speech and Language*, v9, pp.353-379, 1995.
3. **Ho, T-H, Yang, K-C, Huang, K-H, etc., (Ho, 98)** "Improved search strategy for large vocabulary continuous mandarin speech recognition", Proceedings of *IEEE International Conference on Acoustic, Speech and Signal Processing*, ICASSP98, 1998.
4. **Jelinek,F., (Jelinek 1991)** "Self-organized language modeling for speech recognition", *Readings in Speech Recognition*, Morgan Kaufmann, San Mateo, San Mateo, CA, pp.450-506, 1991。
5. **Jelinek, F., (Jelinek 1997) "**Statistical Methods for Speech Recognition" The MIT Press, 1997.
6. **Katz, S.M.,（Katz 1987）**"Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", *IEEE transactions on acoustics, speech and signal processing*, v35, n3, March 1987.
7. **Kneser, R., Ney, H., (Kneser 1993)** "Improved clustering techniques for class based statistical language modeling", Proceedings of *3rd European Conference on Speech Communication and Technology*, Berlin, pp.973-976.
8. **Lee,C.H., Rabiner,R., (Lee1989)** "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition", *IEEE Transactions on Acoustics, Speech and Signal Processing*, v37, n11, Nov., 1989. pp.1649-1658.
9. **Li,Z., Boulianne,G., etc., (Li 1996)** "Bi-directional Graph Search Strategies for Speech Recognition", *Computer Speech and Language*, v10, n4, Oct., 1996, pp.295-321.
10. **李志敏, (Li 1995)** "大字表语音识别系统中的语言模型"，[硕士学位论文]，北京：清华大学计算机科学与技术系，1995。
11. **牟晓隆，（Mou 1998）**"汉语听写机的研究与实现"， [硕士学位论文]，北京：清华大学计算机科学与技术系，1998。
12. **Nadas, A., (Nadas 1985)** "On Turing's formula for word probabilities," *IEEE Trans.*

Acoust., Speech, Signal processing, vol. ASSP-33, pp. 1414-1416, Dec. 1985.

13. **Ney,H., Haeb-Umbach,R., Tran,B.-H. and Oerder,M., （Ney 1992）** "Improvements in Beam Search for 10,000-word Continuous Speech Recognition", Proceedings of *IEEE International Conference on Acoustic, Speech and Signal Processing*, pp.13-16, 1992, San Francisco, CA,
14. **Ney, H., Essen, U., Kneser, R., (Ney 1994)** "On structuring probabilistic dependences in stochastic language modeling", *Computer Speech and Language*, v8, pp.1-38, 1994.
15. **Ortmanns,S., Ney,H., Seide,F. and Lindam,I., （Ortmanns 1996a）** "A Comparison of Time Conditioned and Word Conditioned Search Techniques for Large Vocabulary Speech Recognition", Proceedings of *International Conference on Spoken Language Processing*, ICSLP'96 v4, Oct., 1996, pp.2091-2094.
16. **Ortmanns,S., Ney,H. and Eiden,A., (Ortmanns 1996b)** "Language-Model Look-ahead for Large Vocabulary Speech Recognition", Proceedings of *International Conference on Spoken Language Processing*, ICSLP'96 v4, Oct., 1996, pp.2095-2098.
17. **Ortmanns,S., Ney,H., Aubert X., (Ortmanns 1997)** "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition", *Computer Speech and Language*, v11, pp.43-72, 1997.
18. **Paul D B., (Paul 1991)** "Algorithms for an optimal A* search and linearizing the search in the stack decoder", Proceedings of *IEEE International Conference on Acoustic, Speech and Signal Processing*, Toronto, Canada, 1991, v1, pp.693-696.
19. **Renals, S., and Hochberg, M.M., (Renals, 99)** "Start-synchronous search for large vocabulary continuous speech recognition", *IEEE Transactions on Speech and Audio Processing*, v7, n5, Sept. 1999
20. **Seymore, K., Rosenfeld, R., (Seymore 1996)** "Scalable backoff language models", Proceedings of *International Conference on Spoken Language Processing*, (ICSLP96), 1996.
21. **Viterbi, A.J., (Viterbi 1967)** "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. on IT*, 13(2), Apr. 1967.
22. **俞士汶，朱学峰，王惠 等，（Yu 1998）** "现代汉语语法信息词典详解"， 北京：清华大学出版社 1998。
23. **张津，黄昌宁，杨尔弘， （Zhang 1994）** "根据《现代汉语通用字典》建造机器词

典”， *中文与东方语言信息处理学会通讯*（COLIPS），v4，n2，Dec 1994。

24. **张树武, (Zhang 1997)** “汉语语言处理及语言模型研究”，[博士学位论文]，北京：中国科学院自动化研究所，1997。
25. **Zheng, F., (Zheng 1999a)** “A syllable-synchronous network search algorithm for word decoding in chinese speech recognition”, *International Conference on Acoustic, Speech and Signal Processing* (ICASSP’99), pp. II-601~604, March 15~19, 1999, Phoenix, US.
26. **郑方，牟晓隆，徐明星，武健，宋战江，(Zheng 1999b)** “汉语语音听写机的研究与实现”，*软件学报*，v10, n4, pp.436-444, 1999 年 4 月。
27. **Zheng,F., Song,Z.J., Xu,M.X. etc. (Zheng 1999c)** “EasyTalk: A large-vocabulary speaker-independent Chinese dictation machine”, Proceedings of 6^{th} *European Conference on Speech Communication and Techniques*, EUROSPEECH’99. Budapest, Hungary, 1999, v2, pp.819-822.
28. **朱得熙, (Zhu 1982)** “语法讲义”, 北京：商务印书馆，1982。

致　谢

感谢我的导师郑方博士对我论文工作的悉心指导。郑方老师兢兢业业的工作作风，严谨的科学态度和不断创新的精神是我在今后学习和工作中的榜样。同时感谢他带领我进入语音识别这个领域，多年来，郑方老师为人师表的治学作风，对我的谆谆教诲和严格要求是我成长的动力。在我的研究工作中，方棣棠教授、吴文虎教授和李树青教授也给予了我很多详细的指导，帮助我把握了工作的方向。他们渊博的知识和对待科学问题严肃认真的态度给我留下了深刻的印象，这些都是我的宝贵财富。

感谢一同工作或曾经一同工作过的徐明星、宋战江、郭庆、黄寅飞、何磊、翁武斌、张国亮、罗春华、张继勇、吴根清等同学，他们的帮助和友谊是我工作不断进步的条件。特别感谢宋战江同学在系统集成方面对我的帮助，与何磊同学进行的学术讨论也使我受益匪浅。

感谢所有关心、支持和帮助我的老师，同学和朋友们。

个人简历

武健，出生于 1975 年 4 月，1993 年从湖北省十堰市第一中学考入清华大学计算机科学与技术系，1998 年 6 月毕业，获工学学士学位和管理学学士学位，并于同年继续攻读清华大学计算机科学与技术系计算机应用专业工学硕士 学位。

发表（已接受）论文

1. “**基于音调的特征提取在非特定人语音识别中的运用**”，武健，郑方，吴文虎，方棣棠，*第三届计算机智能接口与应用会议*，张家界，8 月 11－16 日，1997 年。

2. **“基于最小分类错误的声学模型间距离度量”**，郑方，武健，吴文虎，方棣棠， *第三届计算机智能接口与应用会议*，张家界，8 月 11－16 日，1997 年。

3. **“基于模型等价类的快速识别算法”**，武健，郑方，吴文虎，方棣棠，*第五届全国人机语音通讯会议 (NCMMSC)*，哈尔滨，7 月 26－31 日，1998 年。

4. **“Non-Linear Probability Estimation Method Used in HMM for Modeling Frame Correlation”**, Qing GUO, Fang ZHENG, Jian WU and Wen-hu WU, *International Conference on Spoken Language Processing (ICSLP’98)*, SST: 1-6, Nov. 30, 1998, Sydney.

5. “**The Similarity Measure among Acoustic Models and its Two Applications**”, Jian WU, Fang ZHENG, Wen-hu WU and Di-tang FANG, *International Symposium on Chinese Spoken Language Processing (ISCSLP’98)*, ASR-B3, pp.164-168, Dec.7-9, 1998, Singapore.

6. **“A Vocabulary-Independent Keyword Spotter for Spontaneous Chinese Speech”**, Fang ZHENG, Ming-xing XU, Xiao-long MOU, Jian WU, Wen-hu WU and Di-tang FANG, *ISCSLP’98*, RSR-3, pp.202-206, Dec.7-9, 1998, Singapore.

7. **“A New Method Used in HMM for Modeling Frame Correlation”**, Qing GUO, Fang ZHENG, Jian WU and Wen-hu WU, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP’99)*, pp. I-169~172, March 15-19, 1999, Phoenix.

8. **“汉语语音听写机的研究与实现”**, 郑方，牟晓隆，徐明星，武健，宋战江， *软件学报*, 10(4): 436-444, 1999 年 4 月。

9. **“A Novel Discriminative Method for HMM in Automatic Speech Recognition”**, Jian WU and Qing GUO, *6th European Conference on Speech Communication and Technology (EuroSpeech’99)*, Sept.5~10, 1999, Budapest.

10. **“Improving The Syllable-Synchronous Network Search Algorithm for Word Decoding in Continuous Chinese Speech Recognition”**, Fang ZHENG, Jian WU and Zhan-jiang SONG, accepted by *J.Computer Science and Technology.*

11. **“Speaker Adaption based on Combination MAP Estimation and Weighted Neighbor Regression”**, Lei HE, Jian WU, Di-tang FANG and Wen-hu WU, accepted by *IEEE International Conference on Acoustic Speech and Signal Processing,* （*ICASSP2000*）.