密级：公开

连续语音识别中的搜索策略研究

Study on Search Strategy for Continuous 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：计算机应用技术

研 究 生：罗春华

指 导 教 师：郑 方

二零零一年六月

中文摘要

连续语音识别中搜索的目的是根据给定的语音输入，按照某种目标函数，寻找最佳的时间对准序列。本文对时间同步的搜索算法作了比较详细的研究，包括搜索过程中常用的剪枝策略和语言模型 Look-ahead 技术。在此基础上，作者做了以下几方面的工作：

1．声学层错误的分析

在语言模型和声学模型的使用是分离的语音识别系统中，为了考查在该系统中可能发生的各种错误，我们定义了两套判别准则来执行错误分析过程。使用这些判别准则，我们能够有效地发现影响我们的语音识别系统性能的重要因素，并且据此寻找解决问题的方法。为了实现这个目标，我们根据具体的搜索算法实现方式的不同，定义了五类错误类型。识别过程中发生错误的类型可以根据不同的搜索算法给出的搜索路径的得分来进行判断。

2．状态驻留建模在语音识别中的应用

传统的 HMM 模型采用几何分布来对状态驻留建模，它不适合于描述真实的状态驻留分布,而且容易造成在识别过程中出现过长或过短的词驻留的情况，这会在一定程度上降低识别系统的性能。我们在语音识别系统中采用一些指数分布来对状态驻留分布进行建模。这些指数分布包括正态分布，泊松分布和 Gamma 分布。论文中给出了这些分布的参数预测算法以及在识别器中的应用方法。在基本不增加识别器负担的情况下，我们识别系统的性能提高了约 2 个百分点。

3．词图搜索算法的实现和 Trigram 在其中的应用

为了能够在搜索算法中执行更有效地剪枝，必须充分应用语言模型提供的信息，为此我们实现了语言模型和声学模型在一遍搜索过程中都使用的搜索算法。为了算法能够实时运行，我们只能在后续的处理过程中使用 Trigram 语言模型。基于这个考虑，我们实现了能够有效地利用 Trigram 语言模型和更复杂语言模型信息的词图搜索算法，取得了一定的效果。文中对词图的数据结构和词图的生成算法给出了非常详细的论述，基于给定词图的搜索算法也在文中给出。实验表明，词图搜索算法能够充分的利用语言模型提供的信息指导搜索而且速度很快。

关键词：连续语音识别，判别准则，状态驻留建模，词图搜索算法

Abstract

The goal of a search algorithm in continuous speech recognition is to find the best time alignment sequence according to a certain objective function based on the specific speech input. In this paper the time synchronous search algorithm is studied deeply. The commonly used pruning strategies and language model look-ahead techniques are also discussed. Based on these, the following aspects of work are done:

1. Acoustic level error analysis

In order to investigate the errors in our speech recognition system where the usage of language model and acoustic model is separate, we define two sets of judge rules to perform the error analysis task. Using these judge rules, we can efficiently find the most important factors that influence the performance of the system and know how to improve it. To achieve this, we define five types of errors according to different search algorithms. All the errors defined can be distinguished by comparing the search results given by these different search algorithms.

2. State duration modeling and its application to decoder

The geometric distribution is embedded in conventional HMMs. It is not suitable for the describing of the true state duration distribution and may result in either too short or too long words occurring during the speech recognition procedure, hereby decreases the performance of a decoder. So other exponential distributions are introduced to model the state duration distribution. These distributions include the Normal distribution, the Gamma distribution and the Poisson distribution. Their parameter estimation method and application to decoder are discussed. The performance of our speech recognition system can be improved by about 2 percent with little overhead.

3.The implementation of a word graph search algorithm

The language model must be as fully used as in a search algorithm to achieve more efficient pruning. So we implement a one pass search algorithm that uses the acoustic model and the language model in one pass. But the trigram language model can not be easily incorporated into it for the practical consideration. For this, we implement a word graph search algorithm that can use complex language models

and get promising results. The data structure of the word graph and the word graph generation algorithm are given, the word graph search algorithm is also presented. The experimental result indicates that the word graph search algorithm can use the language model efficiently and its time consumption is very small.

Keywords: continuous speech recognition, judge rules, state duration modeling, word graph search algorithm

目　录

图表索引

第一章 语音识别技术综述

语音识别（SR，Speech Recognition）的主要任务是完成语音到文字的转换，其目的是让计算机“听懂”人说话，它是发展人机语音通信和新一代智能计算机的主要组成部分，也是目前业界普遍认为很有前途的一项技术。随着计算机处理能力和存储能力的不断增强，一些非常复杂的语音识别算法能够得以实现，语音识别的性能得以不断提高，这种人与计算机的自然交互方法的研究也日益引起人们的重视。

1.1 语音识别的分类

按不同分类方法，语音识别技术有以下几个不同的方向：

1．按照被识别人的范围，语音识别可以分为特定人（SD, Speaker Dependent）和非特定人（SI, Speaker Independent）语音识别。

2．按照词表的大小，语音识别可以分为小词表、中词表和大词表语音识别。

3．按照说话方式可以分为孤立词、连接词和连续语音语音识别。

上面所列举的几种分类标准，其实都是从识别系统的性能出发的。我们还可以从语音识别系统的实现细节上进行分类，如基于概率统计模型的语音识别，基于人工神经元网络的语音识别等；也可以根据语音识别系统所完成的任务来分，如关键词检测，语音听写等。我们希望实现的语音系统不存在对说话人的限制，不存在对词汇量的限制，不存在对发音方式的限制。但是目前要完全实现上述要求，存在许多困难。

1.2 语音识别中的关键技术

在连续语音识别系统中，所需要用到的重要技术包括如下几个方面：

1．语音流的预处理

- 波形硬件采样率的确定、分帧大小与帧移策略的确定；
- 剔除噪声的带通滤波、高频预加重处理、各种变换策略；
- 波形的自动切分（依赖于识别基元的选择方案）。

2．识别基元与特征参数的选择

- 语音识别基元：包括全音节、声韵母、半音节、音素、子词单元等；
- 语音特征参数：包括时域参数（短时帧能量、过零率、短时自相关等）、频域参数（基音周期、音调曲线、FFT 等）、变换域参数（LPC 系数、LPC 预测误差、LPC CEP、MEL CEP、AR CEP、Delta CEP、LSP 等）；
- 各种特征参数矢量之间的合理距离度量策略的选择方案。

3．特征参数矢量的空间分布假设与描述模型

- 特征参数矢量的空间分布假设：矢量量化（VQ，Vector Quantization）（或统计直方图）、中心距离正态（CDN，Central Distance Normal）分布、混合高斯分布（MGD，Mixed Gaussian Distribution）等；
- 语音模型的确定：隐马尔可夫模型（HMM，Hidden Markov Model）、分段概率模型（SPM，Segmental Probability Model）、中心距离连续概率模型（CDCPM，Central Distance Continuous Probability Model）、人工神经网络（ANN，Artificial Neural Network）等。

4．语音特征参数空间的搜索策略[Ho 1998, Zheng 1998]

- 基于模板的搜索策略：动态时间规整 DTW 等；
- 基于统计模型的搜索策略：HMM 隐含状态的概率转移（Viterbi 解码算法）、帧同步搜索（针对全音节、半音节等识别基元）等；
- 基于神经网络与模糊处理的搜索策略：MLP、Hopfield NN、SOFM、TDNN、RNN、Fuzzy NN、Chaos 等。
- 神经网络（作为后验概率估值器）与 HMM（特征空间搜索）的结合等。

归结起来，语音识别的最关键任务有三个：特征的选择与提取、模式空间的划分、时间对准策略。这些都是决定最终语音识别系统性能的关键因素。本文的研究工作侧重于时间对准策略的研究。

1.3 本文的主要工作和论文安排

在语音识别中，特征参数空间的搜索策略是一个很关键的问题，因为它直接决定了识别系统的性能。搜索策略研究的目的就是在尽量提高识别率的前提下降低系统的开销。由于 HMM 模型在语音识别中得到了广泛应用，因此基于该模型结构的搜索策略的研究是目前的主要方向。为了能够提高识别系统的性能，我们必须首先发现系统中存在的一些问题，据此找到影响系统性能的因素，在此基础上，在识别的过程中充分地使用各种知识来指导搜索。本文对知识指导下的搜索策略做了一定的研究。文中首先对声学层发生的错误进行了分析，发现状态驻留控制策略存在着一些问题，因此提出了状态驻留建模的几种方法，并将其有效地应用到了语音识别过程中。为了能够在搜索过程中使用复杂语言模型，文中还实现了基于词图结构的搜索方法。

本文将按照如下方式组织：在论文的第二部分，将对目前在连续语音识别中使用的搜索策略做一些介绍；声学层错误分析的有关理论分析和实验结果将在第三部分介绍；论文的第四部分和第五部分将主要介绍知识指导下的语音识别，其中第四部分描述状态驻留建模方法及其在语音识别中的应用，而第五部分则介绍了词图搜索算法的具体实现；在论文的最后，给出本论文所引用的参考文献。

第二章 连续语音识别中的搜索策略

语音识别中的搜索问题的理论基础为概率论中的贝叶斯判别准则。显然，如果我们能够以非常高的可靠性来识别音素或者词，那么就没有必要依赖一些延迟判别技术、错误修正技术和统计方法来提高识别率。但是近 30 多年的历史表明：不使用高层次知识的可靠的无错的音素或者词的识别是不可能的，特别是在大词表连续语音识别中。因此，在识别过程中，识别系统必须处理大量的音素或者词的假设，并且考虑由语法、句法和语用等高层次知识所施加的限制。在这种情形下，贝叶斯判别准则就引入到语音识别中的搜索过程上来了。

搜索的目的是为了获得最佳的词序列，即最大化后验概率 $\Pr(w_1 \cdots w_N \mid x_1 \cdots x_T)$ [Bahl 1983]，其中 $w_1 \ldots w_N$ 为词序列，且长度 N 未知，$x_1 \cdots x_T$ 为声学矢量的观察序列。通过应用条件概率下的贝叶斯公式，我们可以把这个后验概率改写为如下的形式：

$$\Pr(w_1 \cdots w_N \mid x_1 \cdots x_T) = \frac{\Pr(w_1 \cdots w_N) \cdot \Pr(x_1 \cdots x_T \mid w_1 \cdots w_N)}{\Pr(x_1 \cdots x_T)} \tag{2.1}$$

$$\hat{w}_1^N = \arg\max_{w_1^N} \Pr(w_1 \cdots w_N \mid x_1 \cdots x_T)$$

这就是所谓的 Bayes 判别准则。它的计算需要用到两种类型的概率分布，通常我们也把这些概率分布称之为伴随着搜索策略的统计知识源。

1）语言模型，即 $\Pr(w_1 \cdots w_N)$。它和声学矢量是独立的，它意味着如何对词典中的词如何形成句子施加一个概率意义上的限制。这些限制来自于句法、语法和语用。它们可以用概率或者非概率的方法来进行建模。在大词表语音识别中，语言模型概率 $\Pr(w_1^N)$ 通常分解表示为一些条件概率 $\Pr(w_n \mid w_1 \cdots w_{n-1})$ 的乘积，而 $\Pr(w_n \mid w_1 \cdots w_{n-1})$ 可以用 Bigram 或者 Trigram 模型来近似，即

$$\Pr(w_n \mid w_1 \cdots w_{n-1}) = p(w_n \mid w_{n-1}) \tag{2.2}$$

$$\Pr(w_n \mid w_1 \cdots w_{n-1}) = p(w_n \mid w_{n-2}, w_{n-1}) \tag{2.3}$$

实际上从语言学的观点来看，存在着一些更强大的语言模型，比如上下文无关语法和树邻接语法等。我们在后面将会看到，这些更精细的语言模型通常仅仅用在语音识别过程的后处理阶段。

2）声学模型，即 $\Pr(x_1 \cdots x_T \mid w_1 \cdots w_N)$。和语言模型概率一样，这些概率是

在识别系统的训练阶段来进行预测的。对于大词表系统，通常它的识别基元是比词小的单位，比如音素（英文中比较常用）、声韵母（汉语识别中用的比较多）、半音节或者音节等。这些子词单元一般用 HMM 来建模，而词的模型则根据发音词典或者字典提供的标注来连接对应的子词模型而成。

我们知道，任何搜索算法的主要目的都是为了减少寻找最佳假设的时间和内存的需要，而同时维持最小的搜索错误。每种搜索算法的具体实现都是不同的[Paul 1992, Aubert 1994]，要想在不同的搜索算法之间做出一个明确的区分是不大可能的。大致说来，搜索算法可以分为两类，一类为时间同步的搜索算法[Lee 1989, Ney 1997, Zheng 1999-1]，另外一类为时间异步的搜索算法。

本章将先介绍两种时间同步的搜索算法：一遍 DP 搜索算法和词图算法，然后对时间异步的搜索算法做一简单介绍。

2.1 时间同步的搜索算法

根据 Bayes 准则，我们可以认为搜索问题是在利用语言模型和声学模型以及发音词典提供的信息下的一个最优化过程，即在这些知识源所定义的状态空间（因为 HMM 是用有限状态自动机来描述的）内的一个求最优解的过程。它可以用如下的公式来表示：

$$\begin{aligned}\hat{w}_1^N &= \arg\max_{w_1^N}\left\{ P(w_1^N)\cdot\sum_{s_1^T} P(x_1^T, s_1^T \mid w_1^N) \right\} \\ &= \arg\max_{w_1^N}\left\{ P(w_1^N)\cdot\max_{s_1^T} P(x_1^T, s_1^T \mid w_1^N) \right\}\end{aligned} \tag{2.4}$$

其中

$$\begin{aligned}p(x_1^T, s_1^T \mid w_1^N) &= \prod_{t=1}^{T} p(x_t, s_t \mid s_{t-1}, w_1^N) \\ &= \prod_{t=1}^{T} [\, p(s_t \mid s_{t-1}, w_1^N)\cdot p(x_t \mid s_t)]\end{aligned} \tag{2.5}$$

请注意我们在上面的求最佳词序列的过程中，使用了最大近似，即所谓的 Viterbi 近似[Jelinek 1976]。在这种近似方法下，搜索空间可以看作是一个巨大的网络，通过它我们来寻找最佳的时间对准序列。搜索将在状态级（s_1^T）和词级（w_1^N）分别进行。显然，由于使用了最大近似方法，我们可以在这两个层次有效地合

并假设，这样可以消除搜索假设数目的组合爆炸，同时以严格的时间同步的方式来产生和评测搜索假设（search hypotheses）。这个特点允许我们采取一种有效的剪枝策略来消除不可能的搜索假设，通常称之为 Beam 搜索。

2.1.1 一遍 DP 搜索

在具体实现某种搜索算法时，我们首先要考虑的就是词典的表示问题。通常词典可以采用线性词典（linear lexicon）和树型词典(tree lexicon)[Haeb 1994,Ney 1992, Ortmanns 1995, Mitchell 1997]两种方式（见图 2.1 和图 2.2）。对于线性词典，在搜索的过程中各个词的模型在搜索的过程中保持严格的分离，词的模型之间没有信息的共享。而对于树型词典，由于它对线性词典中很多具有相同前缀的词进行了合并，因此可以在一定程度上降低搜索空间的规模。但是在树型词典中使用语言模型时(比如 Bigram)，我们遇到了假设词 w 的内容只有在到达树的叶子结点后才能知道的问题。这样，语言模型概率只有在达到 Bigram 中的第二个词的结束状态后才能加入。

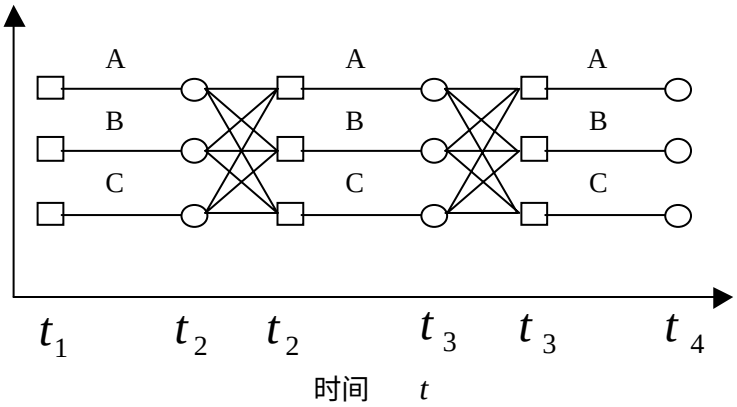
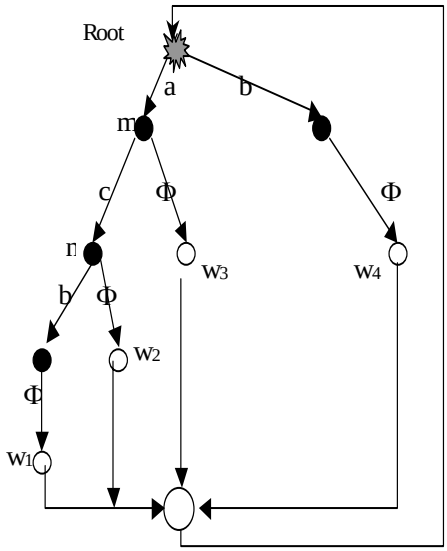


图 2.1 线性词典的例子

为了能够应用动态规划原理，我们可以采用如下的方式来组织搜索空间：对于每个前驱词 v，我们引入词典的一份拷贝，这样在搜索的过程中，当词结束的假设 w 出现时，我们总能够知道它的前驱词为 v。当然如果采用特殊的词典结构，是可以避免搜索过程中树的拷贝的。Schwartz 就曾经提出了一种语法指导下的单树搜索算法[Nguyen 1999]。

搜索的实现需要用到动态规划原理。动态规划概念的主要意图是把一个全

局最优问题的求解分解为小的局部问题并且形成递归联系，更一般的，人们把这个概念称之为 Bellman 最优化原理(Bellman's principle of optimality)[Bellman 1957]。



W_1：acb W_2：ac W_3：a W_4：b

图 2.2 树型词典的例子

2.1.1.1动态规划递归

为了描述动态规划方法，我们引入下面两个量的定义：

$Q_v(t,s)$ 表示时刻 t 到达前驱词为 v 的词法树的状态 s 的最佳部分路径的得分；

$B_v(t,s)$ 表示时刻 t 到达前驱词为 v 的词法树的状态 s 的最佳部分路径的起始时间。

这两个量的计算可以采用如下的公式：

$$Q_v(t,s) = \max\{p(x_t, s \mid s') \cdot Q_v(t-1, s')\} \tag{2.6}$$

$$B_v(t,s) = B_v(t-1, s_v^{\max}(t,s)) \tag{2.7}$$

这里 $s_v^{\max}(t,s)$ 表示前驱词为 v 时假设 (t,s) 的最佳前驱状态。后向指针 $B_v(t,s)$ 只是简单的根据动态规划的决策进行传播。和前驱词 v 不同的是，正在处理的词 w 的索引只有当路径假设到达词法树的结束结点后才有可能知道，因为词法树的每个结束结点标记的是词典中的对应词。

在词的边界，我们需要为每个词 w 找到它的最佳前驱词 v。我们定义

$$H(w;t) := \max\{p(w|v)\cdot Q_v(t,s_w)\} \tag{2.8}$$

这里 S_w 表示词法树中词 w 的结束状态。为了能够传播路径假设，我们需要在处理时刻 t 的数据帧前传递分数和时间索引：

$$Q_v(t-1,s=0) = H(v;t-1) \tag{2.9}$$

$$B_v(t-1,s=0) = t-1 \tag{2.10}$$

算法的流程可以参见图 2.3。

按照时间顺序从左向右处理

声学层：处理 (tree,state) 假设

- 初始化： $Q_v(t-1,s=0) = H(v;t-1)$
  $B_v(t-1,s=0) = t-1$
- 时间对准：使用 DP 计算 $Q_v(t,s)$
- 传播后向指针 $B_v(t,s)$
- 对不可能的假设进行剪枝
- 清除 bookkeeping 列表

词对层：处理词边界假设

"Single best": 对每个词 w

- 计算 $H(w;t) = \arg\max_v\{p(w|v)Q_v(t,S_w)\}$
  $v_0(w;t) = \arg\max_v\{p(w|v)Q_v(t,S_w)\}$
- 存储最佳的前驱词 $v_0 := v_0(w;t)$
- 存储最佳的边界 $\tau_0 := B_{v_0}(t,S_w)$

"Word graph", 对每个词对 (v,w) 存储

- 词边界 $\tau(t;v,w) := B_v(t,S_w)$
- 词的分数 $h(w;\tau,t) := Q_v(t,S_w)/H(v;\tau)$

短语层搜索（可选）

图 2.3 搜索算法的流程

从图中可以看出，DP 递归包括两个层次：声学层，主要是处理词内部一些假设的重新组合；词对层，处理 bigram 语言模型的使用。该搜索过程是一个时间同步的宽度优先的搜索策略。为了降低存储量的需要，可以引入一个回溯数组，它用于记录在每一个时间帧的词边界 (v,w) 和它们的开始时间。在句子的

结束处，通过对回溯数组的一些查找操作可以很轻松地获得识别出来的词序列。

2.1.1.2剪枝操作

对于 LVCSR 中完全的 DP 搜索，在每个时间帧，DP 递归过程需要评测巨大数目的 HMM 状态。比如 20，000 个词的词表，每个词对应一条有 65，000 条弧的树，采用 Bigram 语言模型，每个 HMM 弧有 6 个状态，则可能的搜索空间为 $20k*65k*6=7.8*10^9$，这样大的计算量是无法忍受的，但是如果采取一定的剪枝策略，则可以把计算量降低到每个时间帧只计算 10，000 个或者更少的假设，同时保证识别率基本不下降。

剪枝操作主要有如下 3 个步骤[Steinbiss 1994]：

1．声学剪枝（Acoustic pruning）

保留和最佳的状态假设的分数比较接近的假设，用公式表示为

$$Q_{AC}(t) := \max_{(v,s)}\{Q_v(t,s)\} \tag{2.11}$$

如果

$$Q_v(t,s) < f_{AC} \cdot Q_{AC}(t) \tag{2.12}$$

则剪掉状态假设 $Q_v(t,s)$。

2．语言模型剪枝（Language model pruning）

仅在到达词的边界时，我们才把语言模型的得分加入到累计得分中，每个前驱词的最佳分数用于创建对应的树的假设。用公式表示为：

$$Q_{LM}(t) := \max_{v}\{Q_v(t,s=0)\} \tag{2.13}$$

如果

$$Q_v(t,s=0) < f_{LM} \cdot Q_{LM}(t) \tag{2.14}$$

则剪掉假设 $(t,s=0;v)$。

3．直方图剪枝（Histogram pruning）

这种剪枝方法把存活的状态假设的个数限制在一个固定的范围内。即设置一个上限 M_{Sta}，如果活动状态的个数超过了 M_{Sta}，则仅保留最佳的 M_{Sta} 个假设[Steinbiss 1994]。

以上介绍的是一些基本的剪枝策略。可以看出在搜索的过程中，语言模型的使用和声学模型的使用是严格的分离的。人们在实验中发现，如果能够尽早

的在搜索的过程中使用语言模型，则可以更有效地导剪枝，因此语言模型 Look-ahead 技术[Ortmanns 1997]就在这种环境下产生了。它通过为每个词法树的结点赋予一个语言模型概率而实现，其原理可以参见图 2.4。

如果使用 Bigram 语言模型概率 $p(w|v)$，对于词法树中前驱词为 v 的状态 s 的预测语言模型概率可以表示如下：

$$\pi_v(s) := \max_{w \in W(s)} p(w|v) \tag{2.15}$$

其中 $W(s)$ 是从树的状态 s 可以到达的词的集合。为了能够把预测的语言模型概率应用到剪枝过程中，我们需要对假设 $(t,s;v)$ 的分数做一些修改，我们定义修改过的分数 $\tilde{Q}_v(t,s)$ 为：

$$\tilde{Q}_v(t,s) := \pi_v(s) \cdot Q_v(t,s) \tag{2.16}$$

对于声学剪枝，我们利用修改过的最佳状态假设的分数

$$\tilde{Q}_{AC}(t) = \max_{(v,s)} \{Q_v(t,s)\} \tag{2.17}$$

并且如果有

$$\tilde{Q}_v(t,s) < f_{AC} \cdot \tilde{Q}_{AC}(t) \tag{2.18}$$

则剪掉假设 $(t,s;v)$。

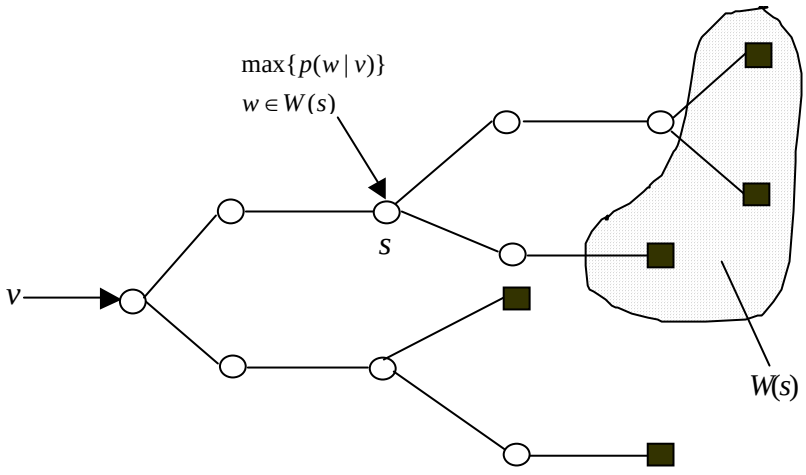


图 2.4 语言模型预测方法

同样的被修改过的分数可以用于直方图剪枝的过程中。实验表明，通过使用这种 Look-ahead 方法，状态假设的数目可以大大的减少。这种 LM look-ahead 方法目前正应用在许多大词表语音识别系统中[Antoniol 1995, Aubert 1994, Austin 1991, Odell 1994, Ortmanns 1997]。

2.1.2 词图搜索算法

词图搜索算法主要是在词的混淆度比较高时用来处理多个词的候选。它的一个显著的优点是声学识别过程可以和复杂语言模型的使用分离开来，而这些复杂语言模型可以在后续的处理过程中被利用上。这些复杂语言模型包括：基于 cache 的语言模型[Kuhn 1990]，基于 trigger 的语言模型[Tillmann 1997]和 long-range 的 trigram 语言模型[Pietra 1994]等。词候选的数目可以根据声学识别过程的混淆程度来进行调整。

和一遍 Beam 搜索过程不同的是，在词图搜索算法中，为了生成词图，我们需要保留多个候选。我们面临的问题是对于一个假设词 w 和它的结束时间 t，我们如何找到它最有可能的一些前驱词。

在词对假设下[Schwartz 1991]（即对于词序列 u_1^m，词 u_{m-1} 和 u_m 之间的边界不依赖于 u_1^{m-2}），可以采用下面的算法来生成词图：

1） 对于每个时间帧 t，我们考虑所有的词对 $u_{m-1}^m = (v, w)$。如果采用 Beam 搜索策略，我们将只考虑最有可能的一些词对。

2） 对于每个三元组，我们记录

a) 词边界 $\tau(t; v, w)$

b) 词的分数 $h(w; \tau(t; v, w), t)$

3） 在语音信号的结尾处，通过回溯来构造词图。

我们可以通过扩展一遍 DP 算法来生成词图。对于词边界，我们可以通过词结束处的后向指针得到

$$\tau(t; v, w) = B_v(t, S_w) \tag{2.19}$$

对于前驱词为 v，词边界为 $\tau = \tau(t; v, w)$，词 w 的分数可以通过下面的公式来计算

$$h(w; \tau, t) := \frac{Q_v(t, S_w)}{H(v; \tau)} \tag{2.20}$$

$$H(w; t) = \max_v \{ p(w \mid v) \cdot Q_v(t, S_w) \} \tag{2.21}$$

因此我们仅仅需要在一遍最佳搜索的过程中多保留一些词序列的假设，就能够在识别结束的时候通过回溯来获得词边界和它们的分数。

2.2 时间异步的搜索算法

对于时间同步的搜索算法，前面已经给出了比较详细的论述。下面将对时间异步的搜索算法做一简单介绍。

时间异步的搜索算法可以通过堆栈解码器（Stack decoder）来实现。堆栈解码器在解码的过程中将使用一些堆栈，这些堆栈包含着一定数目的词的假设。通常，这些假设将通过使用词典得到扩展，扩展出来的新的假设则插入到对应的堆栈中。当所有的堆栈（除了结果堆栈）都变为空时，结果堆栈里将包含最佳假设、N-Best 假设或者网格（lattice），具体的结果形式依赖于搜索的模式。

通常堆栈 stack 指的是后进先出(Last-In-First-Out, LIFO)缓冲区，但是实际上它可以是一个非常简单的按照某种分数排序的的假设的列表（优先队列，Priority queue）。排序所基于的分数可以是：a）部分假设的对数似然度，b）整个完整的句子的对数似然度的预测（A*准则）[Soong 1991]，c）其它一些能够反映部分假设的正确性的分数[Renals 1996]。

所有堆栈解码器都至少包括如下两个层次：a）外层，在堆栈之间循环（词级的搜索）；b）内层，在时间和状态之间循环来搜索词（状态级的搜索）[Robinson 1998]。每当找到一个可能的词边界时，该词的语言模型分数就可以加上去。由于这个动态的语言模型分数可以考虑任何历史词，因此在堆栈解码器中可以很容易地使用任何类型的 *N* 阶 Markov 语言模型或者非 Markov 语言模型。把语言模型的使用从状态空间的 Viterbi 搜索过程分离开来还有其它一些好处。由于词假设的生成完全和词内的搜索独立开来，因此词内的搜索可以采取非常高效的方式来实现，并不需要回溯指针的存储。词的网格可以很轻松地生成。采用类似的过程，N-best 列表也能够很容易地产生。另外 LM lookahead 过程可以作为一个独立的模块结合进来。

第三章 连续语音识别中声学层错误的分析

前面介绍了各种搜索算法，其中一些算法在实际应用中给出了比较好的效果。但是，它们大多是针对具体的应用、具体的系统而提出的，另外某些算法的时间复杂度比较高，因此针对我们的汉语连续语音识别系统，我们有必要根据自己的需要设计搜索算法。

我们知道通常情况下一个语音识别系统由以下几个部分组成：模型训练部分和识别部分。模型训练部分主要包括特征提取，模型结构的定义以及在既定的模型结构下，如何选择一种比较好的方法来预测模型的参数。而模型识别阶段的任务则是采用正确的搜索算法，找到识别基元的边界，并且根据声学模型提供的信息输出合适的声学候选，然后利用语言模型提供的信息得到最终希望的词序列。当然也有一些系统是把语言模型和声学模型的信息一起使用的，这样一遍搜索就能得到词序列。

我们希望这种框架结构下的系统能够工作得尽可能的好。但是实际系统的性能表现往往不尽人意。如果我们独立地用训练集来测试声学模型的性能，通常情况下识别率能够超过 80%。但是如果我们把同样的声学模型应用到我们的汉语连续语音识别系统(CDM, Chinese Dictation Machine)中，识别率却降低到大约只有 70%。系统性能为什么会降低得这么厉害，这是一个值得我们好好考虑的问题。为了提高 CDM 系统的性能，我们必须找出引入识别错误的真正原因，同时对症下药，找出解决的办法。在这种动机的激励下，如何鉴别存在的一些错误类型也是很重要的。

3.1 背景知识介绍

3.1.1 产生识别错误的原因

在一个连续语音识别系统中，存在着多种有可能引入最终识别错误的因素，例如：不够鲁棒的声学模型，端点的不精确检测，不够完善的状态解码算法等等。如下图所示：

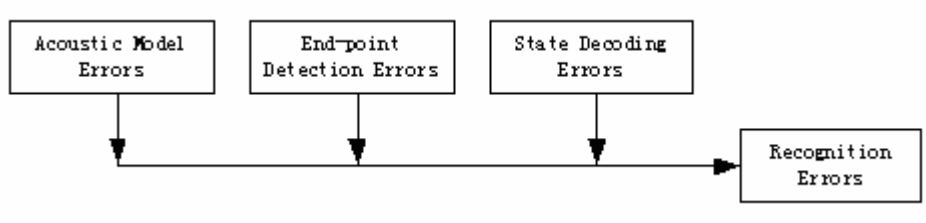


图 3.1 引入错误的 3 个过程

在我们的连续语音识别系统 Easytalk 2000[Zheng 1999-2]中，我们采用的是如下的搜索策略：首先利用时域信息进行切分，这一步操作的目的是为了提高搜索的速度；然后对切分出来的语音段进行状态解码，这是一种修改过的帧同步搜索算法。显然，这些阶段的任何错误都会导致最终的识别错误。基于此，我们从不同的侧面提出了两套判别准则来指明产生识别错误的原因。

3.1.2 利用时域信息进行切分

连续语音流的切分也被称为语音端点检测（Speech End Point Detection），在连续语音识别的预处理中，它是极其重要的环节。其目的是找出语音信号中的各种识别基元（如音素、音节、半音节、声韵母、单词或意群等）的始点和终点的位置，进而将对连续语音的处理变为对各个语音单元的处理，从而大大降低系统的复杂度，提高了系统的性能。

识别基元分点的准确确定，不仅可以使得解码出的状态序列具有很高的准确性，而且对于树搜索方式的 Viterbi 解码或帧同步搜索等算法来说，大大增加了直接剪枝的机会，因而会降低识别系统的时空复杂度，极大地提高系统的总体性能。

在 Easytalk2000 中，我们采用的切分算法是基于归并的音节切分自动机（Merging-Based Syllable Detection Automation， MBSDA）[Zhang 1999]，识别基元选择的是音节。该切分算法充分地利用了短时帧能量(Short-term Frame Energy)，过零率(Zero Crossing Rate)以及基音周期(Pitch Contour)信息来把一个或几个相邻的高度相似的帧合并成一个归并类似段（Merged Similar Segment, MSS），它们被认为属于同一音节的相同状态。这些 MSS 经过一个包含若干状态的“音节切分自动机”后，输出音节的切分点。每个确定的切分段中包含的音节的个数的范围也由 MBSDA 算法给出。该切分算法提供了三种级别的切分

点。

(1) 0 级切分点

该级切分点代表静音的边界。由于它的准确度非常高，因此我们认为这些分点都是正确的，换句话说，通过 0 级切分点不会引入错误。

(2) 一级切分点

它代表音节单元的边界。如果这个音节单元不是静音或噪音，那么这种类型的分点之间的语音段必然包含一个或多个音节。MBSDA 也提供了在每个语音段中包含的音节个数的范围。该范围信息将在我们的搜索算法中得到应用，其主要目的是防止声学搜索空间的迅速膨胀。显然，正确的音节个数可能会超出切分算法提供的音节范围，因此它会引入无法挽回的识别错误。为了和声学搜索产生的切分点相比较，我们有时候把 0 级切分点和一级切分点称之为时域切分点。

(3) 二级切分点

因为 MBSDA 算法是基于自动归并的原理，因此它有能力提供一些二级切分点。我们的状态解码算法要求所有的音节边界都从这些二级分点中选取，也就是说，当在声学搜索的过程中发生音节之间的状态转移时，该转移点必须是一个二级分点。由于这个原因，搜索过程不能保证能够达到最好的状态序列。因此，这些严格的二级分点会带来最终的识别错误。下面的图表是对上述概念的一个解释。

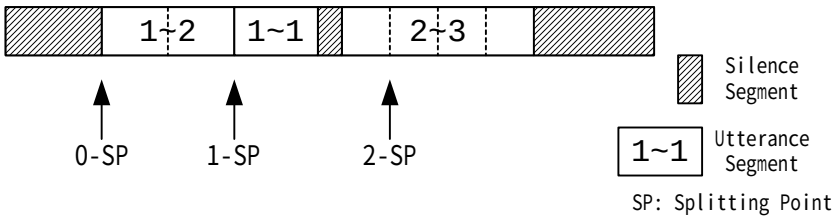


图 3.2 三种级别的切分点

3.1.3 在语音段内进行帧同步的 Viterbi 解码

由于一级切分点的可靠程度很高，因此我们的搜索将局限于在一级分点之

间的语音段内进行。这样做有利于搜索速度的提高，因为每段内搜索的帧数少了，同时也避免了长段语音直接识别时所引起的错误传播。这样既达到了节省空间，又达到了提高效率的目的。

我们在 EasyTalk 2000 中采用的是一种基于统计知识的帧同步搜索算法 SKB-FSS[Zheng 1998](Statistical Knowledge Based Frame Synchronous Speech Algorithm)。它对状态的驻留长度进行了有效的建模，但是需要指出的是可以利用的知识并不局限于状态的驻留长度。实验中我们采用的音节模型是 6 状态的 CDCPM[Zheng 1996]模型。

（1） 利用状态驻留分布（SDD）信息。

状态驻留分布 SDD(State Dwell Distribution)是被广泛使用的用以进行搜索剪枝的一种信息。按照对 32ms 的帧宽、16ms 的帧移对正常语速的汉语普通话数据库进行统计可以得到各个状态内驻留帧数的直方图。这样我们可以得到一个允许的驻留帧数区间[$D_{\min}, D_{\max}$]，该区间确定了在搜索时哪些驻留长度是允许的，同时也避免了帧同步的 Viterbi 算法中驻留帧数的分布概率的计算。很显然，这种做法对语速变化没有很好的鲁棒性，也不能保证有好的效果。因为区间的宽度限制了语速的变化范围，如果发音过长或过短，就不可能得到正确的识别结果，因此提出了下面的 DSDD 方法。

（2） 利用相邻状态间差分状态驻留分布（DSDD）信息

考虑到状态驻留分布 SDD 对语速变化的低鲁棒性，因此我们抛弃了以绝对的驻留区间作为控制状态转移的参考因素，而是采用差分状态驻留分布 DSDD（Differential State Dwell Distribution）作为参考。同样我们可以统计出相邻状态间差分驻留帧数的直方图。

这样，我们可以为每一个状态 s 确定一个允许的差分驻留帧数区间[$d_{\min}^{(s)}, d_{\max}^{s}$]。对于状态 0，我们利用较宽的驻留帧数分布控制搜索时的状态驻留和转移。在实验中我们规定该状态的最小驻留 $m\_fltMinStateDwell$ 和最大驻留 $m\_fltMaxStateDwell$ 分别为：

$$m\_fltMinStateDwell = (1.0 - Beta1) \times nDataLen / (NUMOFSYLSTATES \times m\_nMaxNumOfSyls) \tag{3.1}$$

$$m\_\ fltMaxStateDwell = (1.0 + Beta2) \times nDataLen / (NUMOFSYLSTATES \times m\_nMinNumOfSyls) \quad (3.2)$$

其中 $Beta1$ 和 $Beta2$ 为两个阈值参数，我们分别取的是 0.2 和 1.5；$nDataLen$ 为当前语音段的帧数；$NUMOFSYLSTATES$ 为状态数，这里取 6；$m\_nMaxNumOfSyls$ 和 $m\_nMinNumOfSyls$ 为切分算法提供的音节个数估计的上界和下界。

对于其他状态 $s > 0$，假定已经遍历过的状态的驻留帧数代表值为 $D^{(s-1)}$，那么当前状态的有效驻留帧数就确定为 $[D^{(s-1)} + d_{\min}^{(s)}, D^{(s-1)} + d_{\max}^{(s)}]$。在利用 DSDD 作为控制状态转移的因素之一时，由于第 0 状态允许较宽的驻留帧数分布，因此对语速变化可以有较大的鲁棒性。另一方面，由于当前状态的允许驻留区间是与历史有关的，因此可以避免对一些没有意义的驻留帧数相对应的路径进行存储和搜索，提高搜索的效率和精度。

通过以上的两个步骤，就能够产生声学层的音节候选，送到语言层做进一步的处理，就能够得到 EasyTalk 2000 输出的句子。

3.2 错误分析

让 W_o 表示 CDM(EasyTalk 2000)的输出，W_r 表示输入数据（wav 格式或者特征参数格式文件）所代表的正确句子的标注。如果 W_r 不完全等于 W_o，我们可以认为发生了某些错误。在这种情况下，让 W_e 代表错误的输出。请注意我们只考虑了搜索算法所产生的最佳输出，即拥有最高的匹配似然得分的候选。在下面的叙述中，我们用 $P_{xxx}(W_r)$ 和 $P_{xxx}(W_e)$ 来分别表示 XXX 搜索算法所对应的 W_r 和 W_e 的似然度得分。

在我们的实验中，我们采用了三种搜索算法，它们分别是：

1．（方法 A）根据从特征文件（目前采用的是倒谱）头提取的静音边界信息来执行全局的 Viterbi 搜索。在这种情况下，如果输出不是我们所期望的，我们能够得到两个搜索路径得分 $P_{cep}(W_r)$ 和 $P_{cep}(W_e)$。

2．（方法 B）类似于方法 A，仍然是执行全局的 Viterbi 搜索。不过这个搜索受我们的切分算法所产生的时域分点的限制。同样，如果切分算法不能正确的对 Wave 文件进行分段，也会产生识别错误。因此通过使用这种搜索算法，

我们能够得到 $P_{wav}(W_r)$ 和 $P_{wav}(W_e)$。

3．（方法 C）在各种不同的情形下执行局部搜索。例如：是否使用时域切分点、音节范围或者采用某些状态驻留控制策略等等。在 EasyTalk 2000 中，我们采用修改过的帧同步搜索(FSS)算法来产生声学搜索结果。这样，我们可以用 $P_{fss}(W_r)$ 和 $P_{fss}(W_e)$ 来分别表示正确和错误声学候选的似然度得分。

3.2.1 错误类型定义

当我们用方法 A 来处理输入数据时，我们仅仅使用了静音边界。在这种条件下，如果发生错误，那么它将主要来自于声学模型的错误。我们把这种错误类型定义为声学模型错误(Acoustic Model Error, AME)。

在方法 B 中，由于它不仅使用了时域分点，而且使用了声学模型（用于计算输入数据帧的似然概率），因此会引入 AME 和音节切分错误(Syllable Splitting Error, SSE)。

方法 C 实际上是标准 Viterbi 算法的一个修改版本，它使用了更广泛的信息来指导搜索，这些信息包括时域切分点、音节范围、状态转移控制策略等等。所有这些都会引入一些不同类型的错误。为了区别这些错误，我们根据下述的不同需要对方法 B 做了一定程度的修改。

1．使用音节范围

在搜索过程中，如果某条搜索路径的音节数目范围不满足我们的切分算法所规定的范围要求，那么这条路径将会被抛弃。但是请注意，如果切分算法提供的音节范围不正确，一些潜在的正确路径将会被剪枝，从而最终带来识别的错误。我们把这种错误类型称之为音节范围错误(Syllable Range Error, SRE)。这种情形下的路径匹配得分分别记为 $P_{ran}(W_r)$ 和 $P_{ran}(W_e)$。

2．使用二级分点

在 EasyTalk 2000 的搜索过程中，如果发生了状态转移，那么这个状态转移点必然是一个二级分点。在这样一个受控制的搜索中，搜索空间将会在极大程度上得到减少，同时一些合理的路径可能会被排除在外。这样就引入了二级分点错误(Level-Two Splitting Error, TSE)，同样我们得到了两个分数：$P_{\mathrm{sec}}(W_r)$ 和 $P_{\mathrm{sec}}(W_e)$。

3． 1 和 2 的组合

这意味着声学搜索必须同时满足音节范围和二级分点的限制。从上面的讨论可以看出，对搜索所做的限制越严格，它所引入的错误就会越多。这时我们用声学搜索错误(Acoustic Search Error, ASE)来表示这种情形下的搜索可能会引入的错误。此外，我们得到了另外两个分数：$P_{sdc}(W_r)$和$P_{sdc}(W_e)$。

为了区别引入语音识别错误的各种原因，我们定义了两套不同的判别标准，它们需要对下面的六个条件进行评测：

（1） $P_{cep}(W_r)-P_{cep}(W_e)$

（2） $P_{wav}(W_r)-P_{wav}(W_e)$

（3） $P_{fss}(W_r)-P_{fss}(W_e)$

（4） $P_{sdc}(W_r)-P_{sdc}(W_e)$

（5） $P_{ran}(W_r)-P_{ran}(W_e)$

（6） $P_{\sec}(W_r)-P_{\sec}(W_e)$

3.2.2 判别准则集 A

规则 1：如果（1）< 0，错误类型为 AME。

当我们计算$P_{cep}(W_r)$和$P_{cep}(W_e)$时，我们采用的是同样的 Viterbi 搜索方法，因此如果发生了错误，它必然来自于声学模型。我们用W_o来表示搜索结果。如果$W_o=W_r$，那么必然有$P_{cep}(W_r)>=P_{cep}(W_e)$，在这种情况下，我们可以认为声学模型产生的似然得分是合理的，但是如果$W_o=W_e$，则必然有$P_{cep}(W_r)<P_{cep}(W_e)$，这意味着声学模型并不是象我们想象得那么好，它必然存在着某种程度上的错误。根据这个原则，我们就可以计算出声学模型的可靠程度。

规则 2：如果（1）×（2）< 0，错误类型为 SSE。

$P_{cep}(W_r)$和$P_{wav}(W_r)$之间计算的区别在于：$P_{wav}(W_r)$的计算使用了切分算法所提供的时域分点的额外信息。同样$P_{cep}(W_e)$和$P_{wav}(W_e)$的计算也是这样。如果时域分点是正确的，我们可以确信使用这个知识将会提高我们的语音识别系统的性能，至少不会降低系统的性能。这样当$P_{cep}(W_r)>P_{cep}(W_e)$时，$P_{wav}(W_r)$也

应该比 $P_{wav}(W_e)$ 大，基于这种考虑，我们用（1）×（2）的值来判断音节切分错误 SSE。

规则 3：如果（3）×（4）< 0，则错误类型为 ASE。

和 $P_{sdc}(X)$ 的计算（这里 X 代表 W_r 或者 W_e）不同的是：在计算 $P_{fss}(X)$ 的时候我们使用了状态驻留的控制策略。它是帧同步搜索算法中控制状态驻留和转移的主要方法。因此如果状态驻留控制策略没有问题的话，（3）和（4）应该有同样的增长或下降趋势，这样（3）×（4）就能够反映声学搜索的错误。

规则 4：如果（2）×（5）< 0，则错误类型为 SRE。

在前面提到的修改过的 Viterbi 搜索算法中，我们使用了音节范围信息。和 ASE 的分析类似，使用它会引入音节范围错误(SRE)。这主要是因为我们没法保证音节范围的正确性。

规则 5：如果（2）×（6） < 0，则错误类型为 TSE。

在 Viterbi 算法中使用的二级分点也有可能发生错误，因此我们可以用类似的判别规则来判断这种类型的错误。

3.2.3 判别规则集 B

规则 1：如果（1）< 0，则错误类型为 AME。

和判别规则集 A 中 AME 的定义完全一样。

规则 2：如果（1）>= 0 且（2）< 0，则错误类型为 SSE。

和判别规则集 A 中 SSE 的定义相比较，这个规则更严格一些。它和 AME 的定义联系比较紧密。当（1） >= 0 的时候，我们可以认为错误不会由声学模型引入，那么如果此时（2）< 0，它表明音节切分算法有些问题。

规则 3：如果（1）>= 0 且（2）>= 0 且（4）>= 0 且（5）>= 0 且（6）>= 0 且（3）< 0，则错误类型 ASE。

这个定义和搜索算法 C 有密切的联系。之所以提出这么一个严格的定义，是因为我们只想考察状态驻留控制策略对语音识别系统的性能的影响。只有在其它错误都不发生的情况下，才有可能发生 ASE 错误。

规则 4：如果（1）>= 0 且（2）>= 0 且（5）< 0，则错误类型为 SRE。

如果（1）和（2）都比 0 大，我们可以认为声学模型和时域切分点都是正确的，因此我们可以依据（5）的值来判断音节范围错误 SRE。

规则 5：如果（1）>= 0 且（2）>= 0 且（6）< 0，则错误类型为 TSE。

和判别规则集 A 中 SRE 的定义完全一样。

3.2.4 两套判别规则集之间的差异

在规则集 A 中，一种类型的错误可能在另外一种类型的错误中被重新计算。换句话来说，它们是相关的。例如，对于 SRE 错误，它不但使用了时域切分点，而且使用了音节范围，因此它会包括 SSE 错误，我们的实验结果证明了这一点。但是在规则集 B 中，各种类型的错误是独立的，它们能够从不同侧面反映各种错误的影响，以及具体的影响程度。

这两套规则集都能指示出最有影响的因素，它表明我们能够用判别规则集 A 来做定性的研究。

我们认为规则 B 更可靠，因为它的判决更严格。值得指出的是，集合 B 中各种错误类型的和非常接近于我们以前对系统做总体测试时所得到的识别错误率。

3.3 实验结果

我们采用混合高斯密度模型作为声学模型，在这个模型中每个音节模型采用六个状态来描述，每个状态包含 4 个高斯混合。模型采用的特征参数为 17 维的 MFCC 和 ARCEP，共 34 维。

我们选取 863 数据库中的 M00，M02，M03，M04，M05，M06，M18，M20， M21，M22 作为训练数据来训练声学模型，选取 M24，M25，M26 作为测试模型性能的数据。请注意所有这些数据都属于 A 组。

3.3.1 使用判别规则集 A

结果见表 3.1 和表 3.2。

表 3.1 训练集的测试结果（使用判别规则集 A）

People	AME	SSE	ASE	SRE	TSE
M00	0.182	0.060	0.530	0.198	0.079
M02	0.165	0.046	0.511	0.031	0.050
M03	0.298	0.067	0.536	0.065	0.083
M04	0.088	0.050	0.503	0.058	0.079
M05	0.098	0.054	0.545	0.061	0.065
M06	0.142	0.052	0.522	0.031	0.069
M20	0.226	0.069	0.468	0.274	0.052
M21	0.205	0.031	0.505	0.250	0.038
M22	0.225	0.046	0.511	0.192	0.069

表 3.2 测试集的测试结果（使用判别规则集 A）

People	AME	SSE	ASE	SRE	TSE
M24	0.324	0.094	0.474	0.131	0.046
M25	0.363	0.102	0.457	0.144	0.102
M26	0.257	0.086	0.407	0.232	0.090

在表 3.1 中，AME 表示声学模型错误，它在一定程度上反映了声学模型的可靠程度；SSE 表示音节切分错误，它表示了切分错误对搜索识别结果的影响；ASE 为声学搜索错误，它主要反映的是状态驻留控制策略对搜索结果的影响；SRE 为音节范围错误。由于在搜索的过程中我们利用了切分算法提供的音节范围来限制搜索，因此音节范围的准确性所带来的对识别结果的影响通过 SRE 来反映；TSE 为二级分点错误，它侧重于体现切分算法所提供的二级分点的准确程度。这五种错误的值能够定性地反映各种因素对识别结果的影响程度，通过对这些值的比较，我们能够找到提高算法性能的突破口。

3.3.2 使用判别规则集 B

结果见表 3.3 和表 3.4。

表 3.3 训练集的测试结果（使用判别规则集 B）

People	AME	SSE	ASE	SRE	TSE
M00	0.182	0.033	0.250	0.182	0.046
M02	0.165	0.035	0.167	0.021	0.035
M03	0.298	0.046	0.244	0.027	0.060
M04	0.088	0.044	0.219	0.056	0.060
M05	0.098	0.042	0.223	0.052	0.056
M06	0.142	0.042	0.211	0.027	0.056
M20	0.226	0.044	0.202	0.265	0.023
M21	0.205	0.023	0.246	0.234	0.017
M22	0.225	0.023	0.180	0.179	0.038

表 3.4 测试集的测试结果（使用判别规则集 B）

People	AME	SSE	ASE	SRE	TSE
M24	0.324	0.073	0.276	0.125	0.013
M25	0.363	0.061	0.152	0.111	0.036
M26	0.257	0.056	0.142	0.215	0.061

我们也单独测试了声学模型的性能，该模型的音节识别错误率大约是 24%，和实验数据中的 AME 一栏的数据比较吻合。

3.4 结论

实验结果表明我们的判别准则能够有效地指出在 CDM 系统中发生的各种类型的错误。根据已有的实验数据，我们可以得出如下重要的结论：

- 我们仍然需要做许多工作来提高声学模型的性能，因为仍然有 30%的错误发生了。
- 由切分算法提供的一级分点信息是比较可信的，因为它的准确率高达 95%。
- 由切分算法提供的二级分点也是可信的。
- 由于不同人的说话速度不一样，因此由切分算法提供的音节范围随着说话者的不同变化很大，它必然会严重影响我们的搜索过程，并且最终降低汉语听写机的性能。如果我们使用更宽松的限制，结果会怎么样？这需要实验来验证，另外也可以通过修改切分算法来实现这一点。
- 状态控制策略发生错误的平均概率超过 20%，因此我们有必要寻找更好的状态控制策略来提高搜索的性能。在 SKB-FSS 中，由于采用音节范围的估计来计算状态的驻留，因此这个错误和音节范围的错误估计是有很密切关系的。

从以上的结论可以看出，在本章所介绍的错误分析方法非常有效。它能够找出影响我们的 CSR 系统性能的主要因素和我们需要重点关注的方面。另外需要指出的是我们的错误识别方法能够很容易地扩展到一般情形。

第四章 状态驻留在语音识别中的应用

在语音识别过程中，有很多的知识可以利用。知识利用好了就能够有效地提高系统的性能。从第三章声学层错误的分析过程可以看出，由于采用了切分算法，使得识别过程由句子的范围缩小到有效的语音段的范围，在很大程度上提高了速度；与此同时，由于切分算法提供的音节范围不够准确，状态控制策略不够完善，导致系统的性能在某些情况下出现了下降，因此知识的正确应用是一个需要认真考虑的问题。目前，可以利用的一些知识有状态驻留信息，语言模型（在汉语中，包括 Syllable Ngram 和词的语言模型）等等。下面将分两章来介绍这几个方面的研究，并给出具体的实验结果和分析。本章先介绍状态驻留建模在语音识别中的应用。

4.1 传统的 HMM 模型中的状态驻留建模

我们知道经典的 HMM 模型由三个参数来描述，它们分别是 π, A, B，其中 π 是初始概率分布矢量，A 是各个状态间的转移概率矩阵，而 B 则是输出概率矩阵[Rabiner 1989-1]。在这种框架结构下，它的状态驻留概率为 $a_{ii}^{t-1}(1-a_{ii})$，这是一个几何分布，其中 i 表示状态。a_{ii} 表示状态 i 的自转移概率，t 为状态驻留长度。该几何分步的曲线可以参看图 4.2。通常这种指数分布的描述是不合适的，主要有如下几个方面的原因：

1．不适合于描述真实的状态驻留分布（见图 4.1）

2．没有包含词驻留的概率

3．可能会导致过长或过短的词驻留情形出现

由于上述原因的影响，识别器的性能会在一定程度上下降。我们在实验中发现，声学模型的不够精确所带来的识别错误，很多都和不合适的驻留长度有关。

在图 4.1 中，我们给出了音节/bai/各个状态的驻留长度的统计结果，我们发现图形的形状和几何分布有很大区别，它更充分的说明了 HMM 模型对状态驻留的描述是不精确的。

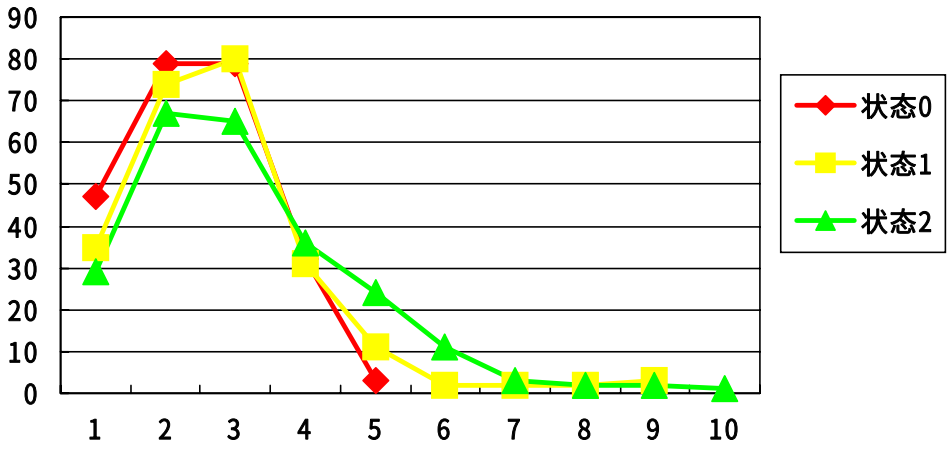


图 4.1 状态驻留长度(/bai/)

4.2 状态驻留建模方法

由于基于 HMM 结构的声学模型本身的内在缺陷，人们尝试用更准确的模型来对状态驻留建模，其主要目的是惩罚不合适的时间对准序列。人们希望采用这些模型的方法在提高识别率的同时，不要带来过多的计算负担。这种方法虽然能够取得一定的效果，但是它也增加了计算的复杂度。

状态驻留建模的基本思想是是对于模型中的每一个状态，赋予一个状态驻留概率 $d_i(t), t = 1,2,...,t^i_{\max}$，$t^i_{\max}$ 是允许的最大驻留。Ferguson 把 $d_i(t)$ 的预测结合到了 Baum Welch 算法中[Ferguson 1980]。但是这种方法有两个缺陷：一个是驻留参数过多，因此参数预测的计算量很大；另外一个是训练这些参数所需要的数据过多。对于每个状态 i，最多可能有 $t^i_{\max}$ 个参数，为了可靠地预测 $d_i(t)$，我们需要对每个 t 做充分的统计，它很有可能带来数据稀疏的问题。为了能够解决训练数据不足的问题，Russell 和 Levinson 提出了半隐 Markov 模型（Hidden Semi-Markov Models, HSMM）[Russell 1985, Levinson 1986]来对状态的时间驻留做精确的建模，该方法能够有效地减少参数的个数。至于状态驻留分布的具体描述方法，可以采用 Gamma 分布，也可以采用正态分布(Normal Distribution)，还可以利用泊松分布(Poisson)（见图 4.2）等。

前面描述的几种分布描述方法都能够解决训练数据不足的问题，但是它们并没有解决计算量过大的问题，为了解决这个问题，Rabiner 提出了一种后处理方法来有效的使用驻留建模[Rabiner 1989-2]。这种方法的一个缺点就是驻留对

Viterbi 搜索算法的贡献是在候选路径都已经产生以后才被加进去，而正确的路径很有可能不在这些候选中，因此它同样有可能带来无法恢复的错误。

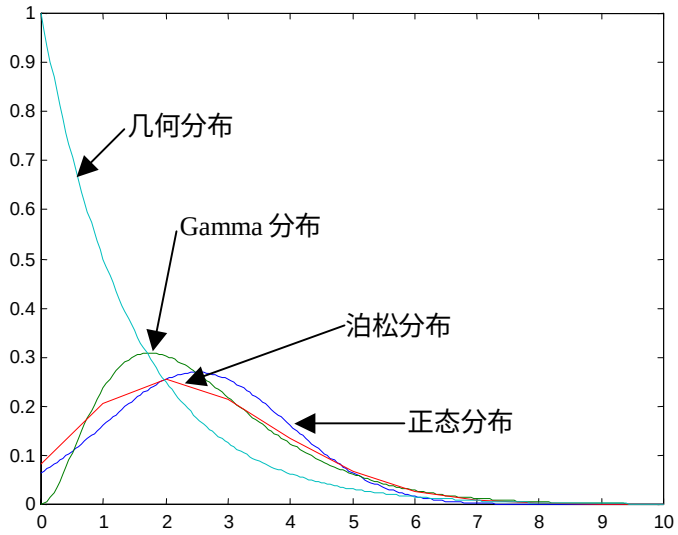


图 4.2 各个分布函数的曲线

本文对采用 Gamma，Normal, Poisson 分布的建模方法都作了尝试。为了解决计算量过大的问题，对搜索算法作了修改，使得驻留在搜索产生合理候选的过程中能够被直接用到，避免了延迟使用所带来的不良后果。由于参数预测算法和驻留使用给搜索算法带来的额外负担都比较小，因此能够有效的提高识别系统的效率。

下面给出对应三种驻留分布的具体的参数预测算法。

对于正态分布，其密度函数为：

$$d_i(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(x-u)^2/2\sigma^2} \tag{4.1}$$

它有两个参数，分别为均值 u 和方差 σ^2，其中

$$u=E(x),\sigma^2=Var(x) \tag{4.2}$$

对于 Gamma 分布，其密度函数为：

$$d_i(x)=\frac{\lambda e^{-\lambda x}(\lambda x)^{n-1}}{(n-1)!} \tag{4.3}$$

它的参数为 λ 和 n，对应的参数预测公式为

$$\lambda = \frac{E(x)}{Var(x)}, n = \frac{E^2(x)}{Var(x)} \tag{4.4}$$

对于泊松分布，它的密度函数为：

$$d_i(x) = e^{-\lambda}\frac{\lambda^x}{x!} \tag{4.5}$$

它只有一个参数 λ，其预测公式为

$$\lambda = E(x) \tag{4.6}$$

从上面的预测公式可以看出，如果我们计算出了状态驻留长度的均值和方差，那么就能够预测出前述三种分布函数的参数（见图 4.3 和图 4.4）。

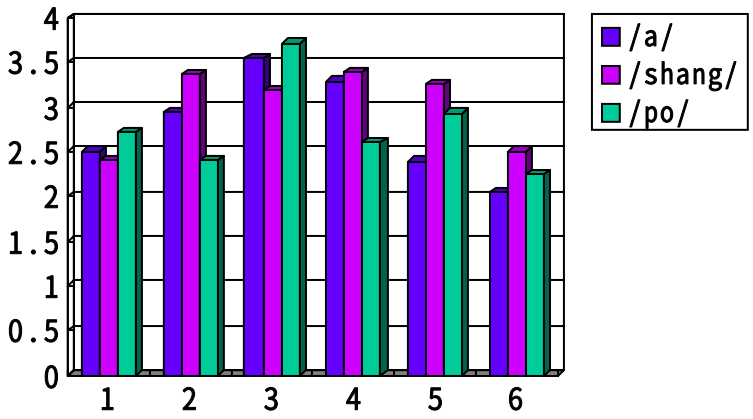


图 4.3 各个状态的均值

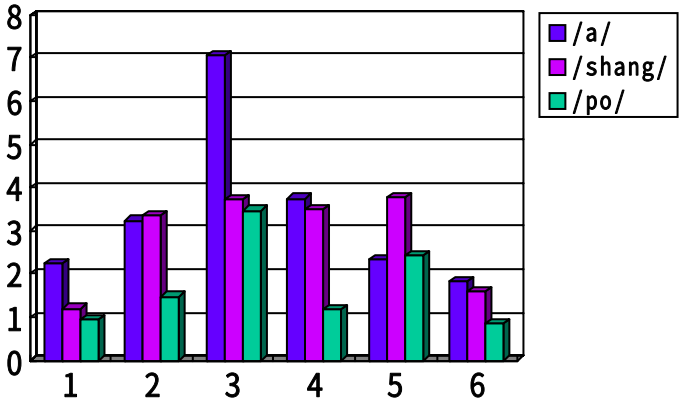


图 4.4 各个状态的方差

由于用 HTK[Young 1995]平台做实验很方便，因此我选择了在 HTK 平台上做状态驻留建模的实验。通过对它的一个很重要的数据结构 Token（Token 在 HTK 的 Token passing[Young 1995]算法中扮演了一个交换搜索信息的角色）做修改，增加驻留域 *dur*，即可记录每个状态的当前驻留长度。HTK 平台上的 HINIT 训练方法的作用是根据观察向量序列来训练初始 HMM 模型的参数。它采用的原理是：重复使用 Viterbi 对准方法来对观察向量序列进行分段，然后使用分段后的向量来重新计算每个状态的参数。对于高斯混合，在每个段内的向量对准到具有最大似然度的混合成分。每一类的向量决定了对应的混合成分的参数。在没有初始模型的情形下，开始时可以对观察向量序列进行均匀分段，每段内的向量采用 K 均值算法进行聚类，这样就可以训练每个混合的参数。在迭代结束的时候，*dur* 里记录了每个状态的驻留长度，因此状态长度的均值和方差可以采用如下方法计算：

$$E(X)=\frac{1}{N}\sum_{i=1}^{N}dur_i \tag{4.7}$$

$$D(X)=E(X^2)-E^2(X)=\frac{1}{N}\sum_{i=1}^{N}dur_i^2-\left(\frac{1}{N}\sum_{i=1}^{N}dur_i\right)^2 \tag{4.8}$$

根据公式（4.2），（4.4），（4.6），可以很容易的得到已知形式的分布函数的参数。

4.3 在识别器中使用驻留分布模型

在驻留分布模型已经得到以后，接下来需要做的工作就是对搜索过程中的部分路径得分加入驻留惩罚分。它根据的是如下的公式：

$$s^{'}=s+\alpha\times\log(d_i(x)) \tag{4.9}$$

其中 s 为原始路径得分，$s^{'}$ 为加入驻留惩罚后的路径得分，$d_i(x)$ 表示状态 i 驻留长度为 x 的概率，α 为一个可调整的因子，其目的是调整驻留惩罚的影响。驻留长度 x 可以根据 Token 中的 *dur* 域的记录来获得。

这里我们采用的惩罚方法（见公式 4.9）比较简单，但是它能够直接影响部分路径的生成，而不是对已经生成的路径作后处理，因此它可以避免驻留惩罚的延迟加入所带来的负面影响。当然也可以采用其它的惩罚方法。比如下面这种方法：

如果我们用 M_q 表示状态 q 的分布取最大值时对应的驻留长度值（以正态分布为例，显然该值为它的均值），假设 t 时刻处于状态 q，$t+1$ 时刻处于 q'，t 时刻的驻留长度为 $D_q(t)$。则

1） 如果 $q=q'$ 且 $D_q(t)<M_q$，则 $t+1$ 时刻的驻留惩罚 $P=0$。

2） 如果 $q=q'$ 且 $D_q(t)>=M_q$，则驻留惩罚

$$P=\log(d_q(D_q(t+1)))-\log(d_q(D_q(t))) \tag{4.10}$$

3） 如果 $q\neq q'$ 且 $D_q(t)<M_q$，则驻留惩罚

$$P=\log(d_q(D_q(t))) \tag{4.11}$$

4） 如果 $q\neq q'$ 且 $D_q(t)\geq M_q$，则驻留惩罚

$$P=\log(d_q(M_q)) \tag{4.12}$$

上面公式表示的意义可以这样来描述：如果处于同样一个状态，当驻留小于 M_q 的时候，我们不加入驻留惩罚，而当驻留大于或等于 M_q 的时候，我们逐渐的加入驻留惩罚。当发生状态转移的时候，则把实际的驻留惩罚加上去。请注意在上面的公式中，我们对发生状态转移的不同情形作了特殊处理。本来只要是发生状态转移，都应该把对应的惩罚加上去。但是我们看到在状态驻留长度小于 M_q 时，我们加的驻留惩罚是真实的驻留惩罚，而在驻留长度大于 M_q 时，我们加的驻留惩罚都是最大值 M_q。这样做的目的是为了对具有相同函数值而驻留长度不同的情形进行不同的处理。以正态分布为例，它是一个对称的函数曲线。对称轴为均值点所在的垂直线上。假设有两个驻留长度，分别为 x_1 和 x_2，且 $d_i(x_1)=d_i(x_2)$，$x_1<x_2$，这样有可能在 x_1 发生转移，也有可能在 x_2 发生转移，但是它们的驻留惩罚是一样的。为了区别这两种情形，而且对长时间的驻留加更大的惩罚，我们对超过 M_q 的 x_2 给出驻留惩罚的最大值，这样可以在一定程度上抑制驻留过长的词的出现。

为了下面叙述的方便，我们将公式(4.9)所表示的方法称之为方法 I，公式(4.10)，(4.11)，(4.12)所表示的方法称之为方法 II。

4.4 实验条件

声学模型采用音节作为语音识别基元，每个音节基元有六个状态，每个状态用 8 个混合高斯来描述。训练数据为 863 数据库中的 10 个男声数据集，测试集为 863 数据库中的 3 个男声数据集。特征参数为 9 维的 MFCC 和 1 维的对数

能量，以及它们的 ARCEP 和 DeltaARCEP 参数，总共 30 维。

4.5 实验结果

(1) $\alpha = 1$，采用方法 I

表 4.1 $\alpha = 1$时的识别结果（方法 I）

People	Baseline		Normal		Gamma		Poisson	
	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc
M24	69.08	67.46	70.40	69.17	70.48	69.35	70.30	69.19
M25	61.44	59.77	62.46	60.48	62.50	60.60	62.39	60.55
M26	69.33	68.93	70.24	69.65	70.53	69.50	70.08	69.80

(2)考察α对识别结果的影响，在这里我们选取 M24 作为例子，而且使用正态分布。

表 4.2 α 对识别结果的影响（方法 I）

0.3		0.5		0.8		1		3	
%Corr	%Acc	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc
68.90	67.31	70.14	69.31	70.34	69.13	70.40	69.17	67.37	67.10

(3) $\alpha = 1$，采用方法 II

表 4.3 $\alpha = 1$时的识别结果（方法 II）

People	Baseline		Normal		Gamma		Poisson	
	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc
M24	69.08	67.46	70.81	69.65	70.97	69.96	70.71	69.68
M25	61.44	59.77	63.18	61.26	63.25	61.30	63.10	61.20
M26	69.33	68.93	70.73	70.15	71.05	70.05	70.50	70.21

4.6 相关结论

- 从表 4.2 可以看出，$\alpha = 1$能够更好地反映状态驻留惩罚的影响。
- 在对状态驻留分布概率进行建模方面，Gamma 分布比其它两种类型的分布（泊松分布和正态分布）更好。
- 可以估计使用词驻留概率也能够提高识别器的性能。
- 方法 II 的效果要高于方法 I。这说明在状态的驻留模型得到后，如何使用驻留还是一个比较重要的问题。
- 通过实验结果可以看出，使用状态驻留建模方法对识别器识别性能的提高幅度不是很大，约 1－2 个百分点，因此如果想大幅度地提高识别器的性能，可能还需要把重点放在更精确地声学建模和语言模型的使用上。

第五章 词图搜索算法及其实现

在前面的叙述中已经提到过，如果我们在识别的过程中利用一些知识来指导搜索过程，一方面可以有效的提高搜索速度，另一方面可以提高识别率。在HTK 中搜索过程使用的是 Token passing 算法。该算法由一个识别网络进行控制，识别网络的生成充分利用了词典和 HMM 模型提供的信息。识别网络由一些结点和连接这些结点的弧组成。每个结点要么是一个 HMM 模型的实例，要么是一个词边界。每个模型实例又由一些状态相连的网络构成。在这种框架下，我们可以认为它的识别网络分为三级：词级、模型级和状态级。在 Token passing 算法中，Token 表示网络中从时刻 0 扩展到时刻 t 的部分路径。时刻 0，在每个可能的开始结点上都有一个 Token，随着时间的推移，它们沿着识别网络的弧进行传播，直到到达一个 HMM 的退出状态。如果一个结点有几个出口，则该 Token 会被拷贝以保证多条路径能够并行的得到扩展。当 Token 沿着弧和结点进行传播时，它的对数似然度会加上相应的转移概率和输出概率。当 Token 沿着网络进行传播时，它必须保留它的历史路由的记录。历史记录保留的详细程度取决于识别输出的需要，通常保留词边界的信息就能够满足大多数应用的需要。

由于 HTK 是一个实验平台，因此它的算法没有考虑到实际应用的需要。为了能够保证该算法的通用性，它的数据结构设计得很复杂。一个典型的例子就是它的识别网络。对于汉语连续语音识别，通常词表的大小为 5－6 万词，按照它的算法构造的网络非常庞大，因此要想在它的上面使用 Trigram 语言模型几乎是不可能的。为了能够建立一个实用的识别器，我们在 HTK 的基础上，借鉴了它的一些好的思路，实现了自己的识别器。通过 Trigram 的加入，取得了比较好的效果。我们算法的基本思路是首先在一遍搜索的过程中使用声学模型和 Bigram 语言模型来产生一些可能的词候选，利用这些词候选来产生词图，然后在后续的过程中利用 Trigram 对词图里的可能路径进行分数重估，最后输出具有最大似然度的路径。

本章将首先介绍词图搜索算法的框架，然后将分小节对该搜索算法的各个主要模块做介绍，重点介绍词图的生成以及基于词图的分数重估算法，最后给出这种搜索算法下的实验结果和相关结论。

5.1 词图搜索算法的框架

采用词图搜索算法的语音识别系统的框架结构如下图所示：

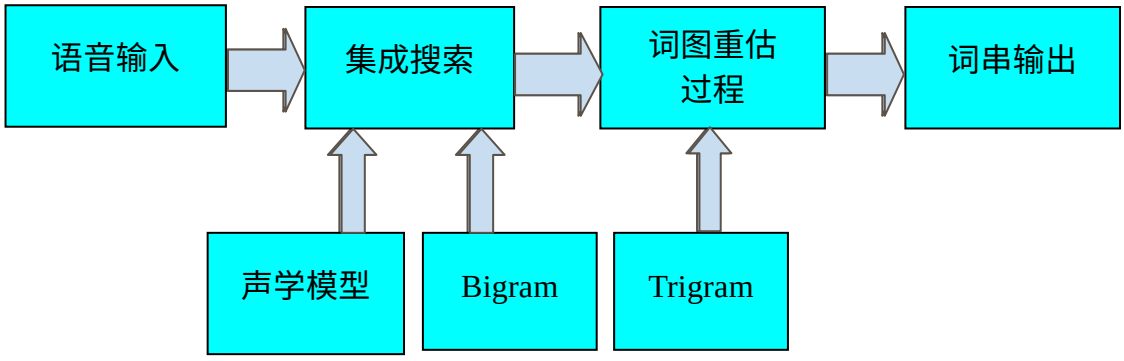


图 5.1 采用词图搜索算法的 LVCSR 系统框架

语音输入后，经过特征提取，首先经过集成搜索（Integrated search）模块，这个过程利用声学模型来产生声学候选，同时利用 Bigram 语言模型来剪枝，该模块产生的词图（Word graph）被词图重估模块（Word graph rescoring）用来进行后处理，在这个过程中可以充分地利用 Trigram 语言模型，最后我们可以得到识别出来的中文词序列。

5.2 集成搜索模块

该模块的主要功能是利用声学模型和 Bigram 语言模型提供的信息来产生词候选，便于词图的生成。为了能够进行上下文相关建模，我们选取了声母和韵母作为语音识别基元，在此基础上，利用 HTK 提供的训练工具 HINIT，HREST 和 HEREST 来训练 TriIF 模型（IF 指的是声母或者韵母）。由于汉语的常用词大约有 50000 多个，因此我们采用树型结构来组织这些词，它可以有效的合并具有相同前缀的词。每个词的模型通过连接对应的 TriIF 模型来获得。如词“饱餐”由“b+ao”，“b-ao+c”，“ao-c+an”，“c-an”组成，当然如果考虑上下文相关（即 Cross-word）[Beulen 1999]，则词的模型要考虑到它的前驱词的具体内容才能最终确定。

在搜索的过程中，充分利用了目前比较常用的一些剪枝技术，如声学剪枝、语言模型剪枝和直方图剪枝，同时还使用了两种 Look-ahead 技术，分别是语言模型 Look-ahead 和 IF Look-ahead。语言模型 Look-ahead 技术在前面已经有过介绍。为了确保算法能够比较高效地运行，我们利用的是 Bigram。至于 IF

Look-ahead 技术，它的引入是为了更好的剪枝，从而提高识别的速度。

IF Look-ahead 的原理是这样的：每次在搜索的过程中要产生一个新的 TriIF 的假设时，首先检查这个 TriIF 的假设是否有可能在剪枝阶段（在下一帧将执行这个操作）存活下来。为了达到这个目的，在 beam search 的过程中，我们将为每一个在特定时间帧可能被激活的 TriIF（对应于网络中的某个结点）预测一个近似的概率。这个概率，我们称之为 look-ahead 分数，将和它的前一 TriIF 的分数结合起来并且在附加的剪枝过程中得到应用。

具体实现如下：

我们用 α 表示在词搜索树中将要开始的一个 TriIF 结点，记住同样的 TriIF 结点 α 可能在不同的词搜索树的拷贝中出现。$\bar{\alpha}$ 代表词搜索树中 α 的特定父结点（parent node）。注意这里 $\bar{\alpha} \to \alpha$ 表明了由 pronunciation lexicon 所加的词法限制。

$\hat{q}(\alpha;t,\Delta t)$ 表示 TriIF α 产生声学特征矢量 $x_{t+1},...,x_{t+\Delta t}$ 的概率。Δt 是平均的 TriIF 驻留时间。

对于 TriIF look-ahead 剪枝，我们把 $\hat{q}(\alpha;t,\Delta t)$ 和 $Q_v(t,s)$ 结合起来，因此对于给定的帧 t，我们为每一个 (α,v) 计算下述分数：

$$\hat{Q}_v(t,\alpha) := \hat{q}(\alpha;t,\Delta t)\cdot Q_v(t,S_{\bar{\alpha}}), \tag{5.1}$$

其中 $S_{\bar{\alpha}}$ 代表 TriIF $\bar{\alpha}$ 的最后状态。

我们用 $Q_{LA}(t)$ 表示 $\hat{Q}_v(t,\alpha)$ 的最大值，如果

$$\hat{Q}_v(t,\alpha) < f_{LA}\cdot Q_{LA}(t), \tag{5.2}$$

则 (α,v) 假设将会从搜索的过程中去掉。这里 f_{LA} 代表 TriIF look-ahead 的剪枝阈值。在系统中我们实际采用的公式为

$$Q_v(t,\alpha) < Q_{LA}(t) - plaThresh \tag{5.3}$$

即设置了一个剪枝的宽度 *plaThresh*。*plaThresh* 可以根据时间来动态的进行调整。通过采用 IF Look-ahead 技术，可以剪掉大约 70%的假设分支。

5.3 词图的结构

词图是一个六元组 $(a,e,w,s_w,t_{a,}t_e)$，其中 a,e 是词 w 的逻辑开始和结束结点，t_a,t_e 为逻辑开始和结束结点对应的时间帧序号，s_w 为词 w 在区间 $[t_a,t_e]$ 上的

HMM 分数的对数值。下图是一个词图的例子：

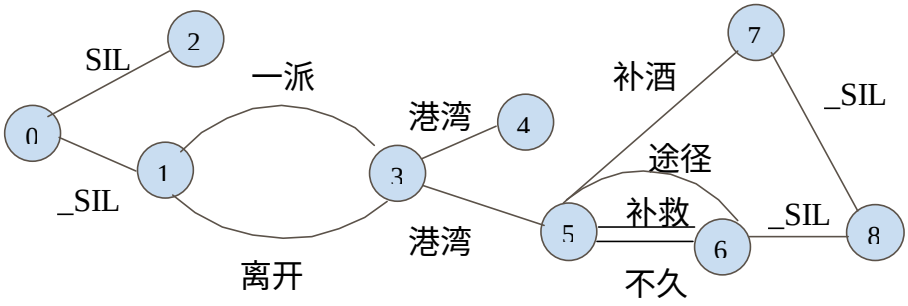


图 5.2 词图的例子

我们可以看出词图具有以下特点：

1．在同样的时间区间内，能够保留多个候选，如 1－3 之间的“ 一派 ”和“ 离开 ”。

2．同一个词可能有不同的结束时刻，如 3－4 之间的“ 港湾 ”和 3－5 之间的“ 港湾 ”。

3．同一个词可能有不同的开始时刻，如 7－8 之间的“ _SIL ”和 6－8 之间的“ _SIL ”。

在对词图做具体描述时，我们采用了如下的数据结构：

词图的数据结构为

- Graphnode node[MAXGRAPHNODE] 记录结点的信息
- Grapharc arc[MAXGRAPHARC] 记录弧的信息
- int link[MAXGRAPHARC] 记录每个结点发出去的弧
- int offset[MAXGRAPHNODE+1]记录每个结点发出去的弧的偏移量
- int nodeNum 记录词图中总的结点数目
- int arcNum 记录词图中总的弧数目

其中 MAXGRAPHARC 和 MAXGRAPHNODE 为两个预先定义好的常量，分别表示词图所能拥有的弧的最大数目和结点的最大数目。

Graphnode 为记录词图中结点信息的数据结构，其具体的表示形式为：

- int nodeIdx 结点的索引
- int timeId 结点的时间戳，和时间帧相对应

- float maxScore 记录通过一个结点的所有路径的最大分数

Grapharc 为记录词图中弧的信息的数据结构，其表示形式为：

- float likeScore 记录词的似然度得分
- int wordIdx 词的索引，该索引是从词典中获得的词的编号
- char wordString[20] 记录表示词的中文字符串的数组

构造词图所需要的所有信息都在集成搜索模块搜索过程中保存的 Path 数据结构里。在 HTK 中，Path 数据结构的定义如下：

- path *prev 记录前一个词的记录
- LogDouble like 记录累计的似然度得分，注意这里不仅包括声学模型得分，也包括语言模型分数
- LogFloat lm 记录累计的词的语言模型得分
- DWORD nodeidx 记录结点在词法树中对应的结点编号
- int frame 记录词结束的时刻（用帧数表示）
- Boolean used 是否被引用的标志位
- int usage 被下一条路径引用的次数
- path *link 链表中的下一条路径
- path *knil 链表中的前一条路径

由该数据结构可以看出，路径信息组成了一个双向链表，这样做的主要目的是为了对链表进行插入、删除和查找等操作的方便。如果我们从那些能够到达最后时间帧的路径开始回溯，则一般能够找出大约 10 条左右的路径。在一遍搜索过程中，通常是从这些路径中挑选出具有最大似然概率的路径作为最终的识别结果。而在采用词图的搜索算法中，搜索的路径范围大大扩大了。我们仍然以图 5.1 为例，在结点 3，这个时候产生的词为“港湾”，它的前驱词可能为“一派”和“离开”。在结点 3 向后扩展之前，将首先执行剪枝操作。这个时候“一派”和“离开”中的某一个词很有可能就被剪掉，失去了继续向后扩展的机会。我们假设“一派”被剪掉，这样将导致“一派”所在的路径无法到达语音输入的结尾帧，从而不可能被识别器输出。而如果采用词图数据结构，我们可以看到，“一派”和“离开”都有可能是“港湾”的前驱，这时的可能路径条数就由原来的一条变成了两条。词图中可能路径的变多，使得我们能够充分的使用 Trigram 来评测各种可能的候选路径。候选路径的多少从某种程度上反映了词图的稀疏程度。词图过密，保留的候选必然很多，这样在词图重估的时候

比较费时，但是它能够提高识别率。词图过疏，重估过程必然加快，但是它可能导致一些期望的候选没有在词图中出现。这个时候无论采用什么重估方法，都将无法获得期望的词串输出，此时出现的错误是无法恢复的。因此需要在识别率和计算复杂度之间作权衡考虑。

5.4 词图重估算法

词图重估算法可以用以下的伪码语言来描述:

1. 对于链表中的每个实例 inst 执行如下操作

{

步骤 1 根据结点 inst->nodeidx 发出去的弧来扩展 inst，同时把 inst 的 token src，dwHistoryId，nodeidx，arcidx 传递给新产生的实例。

- 在实例链表中寻找历史词为 dwHistoryId，弧索引为 arcidx 的实例 inst1
- 如果 inst1 为空，则创建实例 inst1，计算 Trigram 语言模型概率，并且赋给 inst1->nodelm，同时把它加到实例链表的末尾*
- 更新 src 的似然度：src->like += inst1->nodelm + wordgraph.arc[arcidx].likescore
- 如果 src->like > inst1->state->like，则把 token src 传递给 inst1 的状态 0**
- 如果 inst1->like 低于 wordgraph.node[inst1->nodeidx]的剪枝阈值，则把它从实例链表中移走，防止后续的扩展
- 否则如果*为假而**为真，则扩展 inst1

步骤 2：利用路径记录目前正被处理的实例的信息和它的前趋词的记录

}

2．路径数组进行回溯以获得最佳的中文词序列串。

请注意，这个时候路径数组里所保存的路径的似然得分是各个词的纯声学得分以及 Trigram 语言得分的总和，因此，即使是和一遍搜索里产生的路径完全一样（这里的一样指的是词序列一样，而且对应词的时间分点也一样），由于一遍搜索里使用的是 Bigram，它们的似然得分也是不一样的。

5.5 实验结果

实验采用的模型为 TriIF 模型，由 863 数据库中 70 个人的男声数据训练而得，测试集选取了 863 数据库中的 2 个人的男声数据，另外为了做对比，也选取了训练集中的 2 个人来做测试。特征参数为 16 维的 MFCC 和对数能量，以及它们的 ARCEP 参数，共 34 维。每个测试人的数据包括 521 个句子。

1) 一遍搜索和词图搜索的比较

表 5.1 一遍搜索和词图搜索的性能的比较

测试人 / 算法	M00	M08	M11	M16
一遍搜索	68.4%	80.2%	66.0%	50.0%
词图搜索	69.0%	80.7%	67.6%	52.6%

这里 M00 和 M08 来自于训练集的数据，M11 和 M16 来自于测试集的数据。

2) 时间复杂度的比较

选取的测试数据为 M16 的 100 句话，从 m16c1041.mfc 一直到 m16c1140.mfc，采用的方法分别为：直接使用 Trigram 语言模型的一遍搜索（也称集成搜索），在利用 Bigram 的一遍搜索产生的词图基础上的词图搜索。

表 5.2 时间复杂度的比较

算法 / 评测量	词图搜索	一遍搜索I	一遍搜索II
时间	41分钟	62分钟	108分钟
识别率	44.9%	42.3%	43.8%

在使用 Trigram 语言模型的一遍搜索中，为了能够达到比较理想的识别率，剪枝阈值不能设置得太小，否则由于没有保留足够多的历史信息，使用 Trigram 并不一定能够提高识别率。上表中一遍搜索 I 和一遍搜索 II 的区别就在于一遍搜索 II 中的阈值设置得比较大，在搜索的过程中保留了比较多的候选（这样正确的候选就更有可能被保留下来）因此通过使用 Trigram 语言模型分数，正确的候选更有可能被挑选出来，导致识别率有了提高，但是要想超过词图搜索的

识别率，剪枝阈值应该设置得更宽，此时耗费的时间必然更多，这对于一个实时系统而言是无法容忍的，因此在一遍搜索中直接使用 Trigram 是不大可能的。

5.6 实验结论

1．在词图搜索中使用 Trigram 语言模型是可行的，而且耗费的时间很少。通过实验结果表 5.2 可以看出，在没有很好的剪枝策略的前提下，是没有办法在集成搜索中直接使用 Trigram 的。此时使用词图就成为了必然。

2．词图搜索使得复杂语言模型的利用成为可能（比如 Long-span 语言模型）。词图搜索相比集成搜索的优越之处不光在于它能够很轻松的使用 Trigram 语言模型，更重要的是，我们可以在词图的基础上尝试使用各种复杂的语言模型，比如 Long-range 的 Trigram 等。可以这么说，词图能够作为我们测试语言模型性能的一种方法。

3．词图搜索算法在测试集中的性能表现要优于训练集，这主要得益于它从相似候选中选取出正确候选的能力。从表 5.1 可以看出，在训练集（M00、M16）的测试中，M00 和 M08 的识别率没有明显的提高，而在测试集（M11、M16）中，M11 和 M16 的识别率有了一定程度的提高。这种现象出现的主要原因在于：对于训练集，由于声学模型就是用这些数据训练出来的，因此相互之间匹配得比较好，这样在识别的时候，它产生的候选质量比较高，更为重要的是，我们期望的候选（正确的声学候选）所在的路径的分数比较高，这样只通过使用 Bigram，这些正确的候选就能被挑选出来，因此在训练集中的测试看不出明显的效果。Trigram 挑选候选的能力在测试集中体现得比较明显。这充分的说明词与词之间的搭配关系如果在识别的过程中得到正确的应用，是能够提高识别率的，而且不会带来明显的计算负担。

参考文献

[1] Antoniol, G., Brugnara, F., Cettolo, M. & Federico, M. (1995), “Language Model Representations for Beam-search Decoding”, Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, vol.1, Detroit, MI, May 1995, pp.588-591

[2] Aubert, X., Dugast, C., Ney, H. & Steinbiss, V. (1994), “Large Vocabulary Continuous Speech Recognition of Wall Street Journal Data”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Adelaine, Australia, volume II, pp. 129-132

[3] Austin, S., Schwartz, R. & Placeway, P. (1991), “The Forward-backward Search Algorithm”, Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, Toronto, Canada, May 1991, pp.697-700

[4] Bahl, L.R., Jelinek, F. & Mercer, R.L. (1983), “A Maximum Likelihood Approach to Continuous Speech Recognition”, IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-5, pp.179-190, Mar. 1983

[5] Bellman, R.E. (1957), Dynamic Programming，Princeton, NJ:Princeton Univ. Press, 1957

[6] Beulen, K., Ortmanns, S. & Elting, C. (1999), “Dynamic Programming Search Techniques For A cross-Word Modelling In Speech Recognition”, IEEE International Conference On Acoustic, Speech and Signal Processing, vol.2, Phoenix, AZ, Mar. 1999

[7] Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H. & Ures, L. (1994), “Inference and Estimation of a Long-range Trigram Model”, Proc. 2nd Int. Coll. Grammatical Inference and Applications, Alicante, Spain, Sept, 1994, pp.78-92

[8] Ferguson, J.D. (1980), “Variable Duration Models for Speech”, Proc. Symp. on the Application of Hidden Markov Models to Text and Speech, J.D. Ferguson ed., pp. 143-179, Princeton, New-Jersey, 1980

[9] Haeb-Umbach, R. & Ney, H. (1994), “Improvements in Time-synchronous Beam Search for 10000-word Continuous Speech Recognition”, IEEE Trans, Speech Audio Processing, vol.2, pp.353-356, Apr.1994

[10] Ho, T.H., Yang, K.C., Huang, K.H. & Lee, L.S. (1998), “Improved Search Strategy for Large Vocabulary Continuous Mandarin Speech Recognition”, IEEE International Conference On Acoustic, Speech and Signal Processing, part2 (of 6), May 12-15,1998, Seattle

[11] Jeliner, F. (1976), “Continuous Speech Recognition By Statistical Mehtods”, Proc. IEEE, vol. 64, pp.532-556, Apr.1976

[12] Kuhn, R. & De Mori, R. (1990), “ A Cache-based Natural Language Model for Speech Recognition”, IEEE Trans. Pattern Anal. Machine Intell., vol. 12, pp. 570-583, June 1990

[13] Lee, C.H. & Rabiner, L.R. (1989), “A Frame Synchronous network search algorithm for connected word recognition”, IEEE Trans. On ASSP, 37(1):1649-1658, Nov.1989

[14] Levinson, S.E. (1986), “Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition”, Computer, Speech and Language, 1(1):29-45, 1986

[15] Mitchell, C.D. & Setlur, A.R. (1997), “ Improved Spelling Recognition Using A Tree-based Fast Lexical Match”, IEEE International Conference On Acoustic, Speech and Signal Processing

[16] Ney, H., Haeb-Umbach, R., Tran, B.H. & Oerder, M. (1992), “Improvements in Beam Search for 1000-word Continuous Speech Recognition”, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, San Francisco, CA, Mar. 1992, pp.13-16

[17] Ney, H. & Ortmanns, S. (1997), “Progress in Dynamic Programming Search for LVCSR”, Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, Santa Barbara, CA, pp.287-294

[18] Nguyen, L. & Schwartz, R. (1999), “Single-tree Method for Grammar Directed Search”, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, vol.2, Phoenix, AZ, Mar. 1999, pp.613-616

[19] Odell, J.J., Valtchev, V., Woodland, P.C. & Young, S.J. (1999), “A One-pass Decoder Design for Large Vocabulary Recognition”, Proc. ARPA Spoken Language Technology Workshop, Plainsboro, NJ, Mar. 1994, pp.405-410

[20] Ortmanns, S. & Ney, H. (1995), “ Experimental Analysis of the Search Space for 20000-word Speech Recognition”, Proc. 4th Eur. Conf. Speech Communication and Technology, Madrid, Spain, Sept. 1995, pp.901-904

[21] Ortmanns, S., Eiden, A., Ney, H. & Coenen, N. (1997), “ Look-ahead Techniques for Fast Beam Search”, Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol.3, Munich, Germany, Apr. 1997, pp.1783-1786

[22] Paul, D. (1992), “An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with A Stochastic Language Model”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, San Francisco, CA, volume I, pp. I-25-I-28

[23] Rabiner, L.R. (1989-1), “A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition”, Proc. IEEE, vol. 77, pp.257-286, Feb.1989

[24] Rabiner, L.R., Wilpon, J.G. & Soong, F.K. (1989-2), “High Performance Connected Digit Recognition Using Hidden Markov Models”, IEEE Trans. Acoust., Speech and Signal Processing, vol ASSP-37, pp.1214-1225, Aug. 1989

[25] Renals, S. & Hochberg, M. (1996), “Efficient Evaluation of The Search Space Using the NOWAY Decoder”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, GA, volume I, pp.149-153

[26] Robinson, T. & Christie, J. (1998), “Time-first Search for Large Vocabulary Speech Recognition”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, WA, volume II, pp.829-833

[27] Russell, M.J. & Moore, R.K. (1985), “Explicit Modeling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition”, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, 1985, pages 5-8

[28] Schwartz, R. & Austin, S. (1991), “A Comparison of Several Approximate Algorithms for Finding Multiple (N-best) Sentences Hypothesis”, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Toronto, Canada, May 1991, pp.701-704

[29] Soong, F. & Huang, E. (1991), “A Tree-trellis Based Fast Search for Finding the N-best Sentence Hypotheses in Continuous Speech Recognition”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto, Canada, volume I, pp.705-708

[30] Steinbiss, V., Tran, B.H. & Ney, H. (1994), “Improvements in Beam Search”, Int. Conf. Spoken Language Processing, Yokohama, Japan, Sept, 1994, pp. 1355-1358

[31] Tillmana, C. & Ney, H. (1997), “Word Triggers and The EM Algorithm”, ACL Special Interest Group Workshop Computational Natural Language Learning, Madrid, Spain, July 1997, pp. 117-124

[32] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V. & Woodland, P. (1995), “The HTK Book”, Microsoft Corporation, Dec. 1995bbu

[33] 张继勇，郑方，杜术，宋战江，徐明星(Zhang 1999) “连续汉语语音识别中基于归并的音节切分自动机”，软件学报 10(11):1212-1215, 1999 年 11 月

[34] 郑方，徐明星，吴文虎(Zheng 1998)，“ 连续语音识别中的搜索策略 ”，第五届全国人机语音通讯学术会议论文集 (NCMMSC-98), 138-143，1998 年 7 月

[35] Zheng, F. (1999-1), “ A Syllable-Synchronous Network Search Algorithm for Word Decoding in Chinese Speech Recognition”, IEEE International Conference On Acoustic, Speech and Signal Processing,pp.II-601-604, March 15-19,1999,Phoenix

[36] Zheng, F., Song, Z.J., Xu, M.X., Wu, J., Huang, Y.F., Wu, W.H. & Bi, C. (1999-2), “EasyTalk: A Large-Vocabulary Speaker-Independent Chinese Dictation Machine”, EuroSpeech’99, Vol.2,pp.819-822, Budapest, Hungary, Sept.1999

致谢

转眼之间，在清华已经学习了七年。我相信在清华的所学将会对我的一生产生重要的影响。在这里我要感谢我的导师郑方博士，是他带领我进入了语音识别的研究领域。他严谨的科学态度、兢兢业业的工作精神和不断进取的精神是我学习的榜样。感谢吴文虎老师，他为人师表的治学态度将不断激励我前进。感谢方棣棠和李树青老师，他们对待科学的认真态度和钻研精神给我留下了深刻的印象，也使我学到了不少做人的道理。在这里还要感谢徐明星老师，他对我工作的不少建议和讨论使得我的研究工作能够按时完成。

语音组的同仁何磊、宋战江、燕鹏举、张国亮、李净、张继勇、李芳等同学在平时的工作和生活中也给了我不少帮助，在这里一并表示感谢！

最后感谢我的家人和亲戚朋友，是你们对我无微不至的关怀使得我能够圆满的完成在清华七年的学习生活。

个人简历

个人基本情况

姓名：罗春华　　出生日期：1976 年 1 月 6 日

籍贯：湖北省荆门市　获工学学士学位时间：1999 年 7 月

已发表学术论文

- Luo Chunhua, Zheng Fang, Xu Mingxing, **Acoustic Level Error Analysis in Continuous Speech Recognition**, ISCSLP2000，Oct 2000, Beijing

- Chunhua Luo, Fang Zheng, Mingxing Xu, **An Equivalent-Class Based Learning Method for MGCPM**, ICSLP2000, Oct 2000, Beijing