应用 HTK 搭建语音拨号系统

苏统华
哈尔滨工业大学人工智能研究室
2006 年 10 月 30 日

该系统能够识别连续说出的数字串和若干组姓名。建模是针对子词(sub-word, eg. 音素)，具有一定的可扩充性。当加入一个新名字时，只需修改发音词典和任务语法即可。模型为连续混合高斯输出，运用语音决策树聚类形成的绑定状态式三音素。

1. 数据准备

需要录制训练数据和测试数据。为进行校准，还需要数据的标注文本。这里用任务语法(task grammar)产生真值文本(ground truth)。为了处理训练数据，需要定义一个语音集合和一个字典用以涵盖训练和测试数据中涉及的单词。

[step 1]任务语法定义

任务语法以包含变量的正则表达式形式定义，存储在文件 gram 里：

文件名：gram

```
$digit = ONE | TWO | THREE | FOUR | FIVE |
      SIX | SEVEN | EIGHT | NINE | OH | ZERO;
$name = [ SUE ] LAW |
      [ JULIAN ] TYLER |
      [ DAVE ] WOOD |
      [ PHIL ] LEE |
      [ STEVE ] YOUNG;
( SENT-START ( DIAL <$digit> | (PHONE|CALL) $name) SENT-END )
```

上面的语法是高层表示，必须通过 HParse 转成 HTK 可用的底层表示。底层表示存于文件 wnet 中：

HParse gram wdnet

文件名：wdnet

```
VERSION=1.0
N=31     L=62
I=0      W=SENT-END
I=1      W=YOUNG
……
J=0      S=2      E=0
……
J=61     S=0      E=29
```

[step 2]字典定义

此例中利用 BEEP 语音字典，除去其中的重音符，并在每个发音后加入 sp（short pause, 小停顿）。如果有哑音标志，就用 MP 命令把 sil 和 sp 合并成 sil。这些处理命令放在 global.ded 编辑脚本中：

文件名：global.ded
AS sp RS cmu MP sil sil sp

执行 HDMan 生成与任务相关的发音词典 dict1：

HDMan -m -w .\lists\wlist -n .\lists\monophones1 -l dlog .\dict\dict1 .\dict\beep .\dict\names

上面的 names 文件是手工制作的专有人名的发音（包括 SEND-START，SEND-END），文件 wlist 是出现在任务语法中的所有单词的有序列表，文件 monophones1 是用到的音素的列表，dlog 为参数文件。

注 1：实际上，这里应该手工修改 dict1，为 SENT-END 和 SENT-START 加上无输出标志，为了展示队后面步骤地影响，这里故意不添加。

文件名：.\dict\dict1	
CALL	k ao l sp
DAVE	d ey v sp
DIAL	d ay ax l sp
EIGHT	ey t sp
FIVE	f ay v sp
FOUR	f ao sp
FOUR	f ao r sp
JULIAN	jh uw l ia n sp
JULIAN	jh uw l y ax n sp
LAW	l ao sp
LEE	l iy sp
NINE	n ay n sp
OH	ow sp
ONE	w ah n sp
PHIL	f ih l sp
PHONE	f ow n sp
SENT-END	sil
SENT-START	sil
SEVEN	s eh v n sp
SIX	s ih k s sp
STEVE	s t iy v sp
SUE	s uw sp
SUE	s y uw sp
THREE	th r iy sp

TWO	t uw sp
TYLER	t ay l ax sp
WOOD	w uh d sp
YOUNG	y ah ng sp
ZERO	z ia r ow sp

文件名：.\dict\names	
DAVE	d ey v
JULIAN	jh uw l y ax n
JULIAN	jh uw l ia n
LAW	l ao
LEE	l iy
PHIL	f ih l
SENT-END	[] sil
SENT-START	[] sil
STEVE	s t iy v
SUE	s uw
SUE	s y uw
TYLER	t ay l ax
WOOD	w uh d
YOUNG	y ah ng

为了避免在 dlog 里出现烦人的 wanning，那就在 names 和 beep 同一目录下分别建立同名的编辑脚本，内容为空即可。

文件名：names.ded

文件名：beep.ded

[step 3]录制语音数据

HSGen 工具可以生成符合 task grammar 的句子，用来指导录音(HTK book 里叫 sentence prompts)：

HSGen -l -n 140 wdnet .\dict\dict1>.\labels\trainprompts

HSGen -l -n 15 wdnet .\dict\dict1>.\labels\testprompts

根据上述指令文件，录制相应的 140 个训练用语音数据文件和 15 个测试用语音数据文件。一个录制的例子如下：

HSLab .\data\Train\speech\S0001

注 2：随本文的压缩包没有包括这些数据文件，但包括了它们对应的特征文件，所以对整个实验没有什么影响。

[step 4]标注数据，得到真值文件

Perl 脚本 prompts2mlf 可以把录音文本截成单词级真值文件。例如：

perl .\scripts\prompts2mlf .\labels\trainwords.mlf .\labels\trainprompts

perl .\scripts\prompts2mlf .\labels\testwords.mlf .\labels\testprompts

标注编辑器 HLEd 可把单词级真值文本（word level MLF）转成音素级真值文本（phone level MLF）：

HLEd -l * -d .\dict\dict1 -i .\labels\phones0.mlf mkphones0.led .\labels\trainwords.mlf

编辑脚本 mkphones0.led 的内容如下：

文件名：mkphones0.led
EX IS sil sil DE sp

EX 命令表示按照字典 dict1 进行展开，IS 表示在每个话语的前后插入标志，DE 一行表示 phones0.mlf 中单词间不用 sp 隔开。

[step 5]数据的特征提取

这里所用特征为 MFCC。工具 HCopy 可以实现提取特征的工作。

HCopy -T 1 -C .\config\config1 -S codetr.scp

其中，配置文件 config1 要设置转换参数，另一脚本文件(script file)codetr.scp 指定训练及输入和输出文件列表。执行结果，HCopy 对 codetr.scp 文件左侧的语音数据按 config 的配置提取特征并存入 codetr.scp 文件右侧特征文件中。

文件名：.\config\config1
Coding parameters TARGETKIND = MFCC_0_D_A TARGETRATE = 100000.0 SAVECOMPRESSED = T SAVEWITHCRC = T WINDOWSIZE = 250000.0 USEHAMMING = T PREEMCOEF = 0.97 NUMCHANS = 26 CEPLIFTER = 22 NUMCEPS = 12 ENORMALISE = F

对于测试数据如法炮制：

HCopy -T 1 -C .\config\config1 -S codete.scp

注 3：由于附带的压缩包已经包括了特征文件（在.\data*\feature 下），本步骤不用执行。

2. 创建单音素 HMM 模型

涉及创建一系列单高斯单音素 HMM 的过程。

[step 6]一致初始法创建单音素模型

定义一个原始模型 proto：

文件名：proto

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
   <NumStates> 5
   <State> 2
      <Mean> 39
         0.0 (x39)
      <Variance> 39
         1.0    (x39)
 <State> 3
      <Mean> 39
         0.0    (x39)
      <Variance> 39
         1.0    (x39)
 <State> 4
      <Mean> 39
         0.0    (x39)
      <Variance> 39
         1.0    (x39)
 <TransP> 5
   0.0 1.0 0.0 0.0 0.0
   0.0 0.6 0.4 0.0 0.0
   0.0 0.0 0.6 0.4 0.0
   0.0 0.0 0.0 0.7 0.3
   0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

训练文件存于 train.scp 中，用全局均值和方差来初始化 HMM 模型的高斯参数：

HCompV -C .\config\config1 -f 0.01 -m -S train.scp -M .\hmms\hmm0 proto

注 4：也可以省掉-C 参数，只要 train.scp 里是特征文件列表，并且特征是 MFCC_0_D_A)

在目录 hmm0 下生成了更新后的 proto 和一个截至宏 vFloors。基于.\hmms\hmm0\下的两个文件，手工制作主宏文件（Master Macro File）hmmdefs 和与 vFloors 相关的宏 macro，具体制作过程见 HTK book，压缩包中有实例。

由于暂时不用 sp 模型，删去 monophones1 中的 sp，构成 monophones0 文件，重估参数：

HERest -C .\config\config1 -I .\labels\phones0.mlf -t 250.0 150.0 1000.0 -S train.scp

-H .\hmms\hmm0\macros -H .\hmms\hmm0\hmmdefs -M .\hmms\hmm1 .\lists\monophones0

同上，重复估计两次：

HERest -C .\config\config1 -I .\labels\phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm1\macros -H .\hmms\\hmm1\hmmdefs -M .\hmms\hmm2 .\lists\monophones0

HERest -C .\config\config1 -I .\labels\phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm2\macros -H .\hmms\hmm2\hmmdefs -M .\hmms\hmm3 .\lists\monophones0

注 5：为节省空间，也因为上面三步很简单（想出错都难），hmm1 和 hmm2 下的模型文件并没有包含在压缩包里。但 hmm3 下的模型文件包含在压缩包里了（因为下一步要用到）。

[step 7]修补哑音素模型

对 sil 模型加入回溯链，对 sp 绑定到 sil 的中间状态上。具体的，哑音素模型按下面两步执行。首先，修改 hmm3\hmmdef，复制 sil 的中间状态为 sp 模型的唯一状态,另存到\hmms\hmm4 目录下。然后，指明 sp 绑定到 sil 中间状态，利用 HHEd 加入回溯转移概率：

HHEd -H .\hmms\hmm4\macros -H .\hmms\hmm4\hmmdefs -M .\hmms\hmm5 sil.hed .\lists\monophones1

修改 mkphones0.led,去掉最后一行，存为 mkphones1.led，利用 HLEd 工具得到包含 sp 的音素级真值文本：

HLEd -l * -d .\dict\dict1 -i .\labels\phones1.mlf mkphones1.led .\labels\trainwords.mlf

重估两次：

HERest -C .\config\config1 -I .\labels\phones1.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm5\macros -H .\hmms\hmm5\hmmdefs -M .\hmms\hmm6 .\lists\monophones1

HERest -C .\config\config1 -I .\labels\phones1.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm6\macros -H .\hmms\hmm6\hmmdefs -M .\hmms\hmm7 .\lists\monophones1

注 6：hmm5 和 hmm6 下的模型文件并没有包含在压缩包里。为了进行下面的阶段性测试，hmm7 下的模型文件包含在压缩包里了。

这么没头没尾的干下去，太枯燥了！我们先看看这时的识别率如何吧，也能增加一点成就感。执行如下命令：

HVite -H .\hmms\hmm7\macros -H .\hmms\hmm7\hmmdefs -S test.scp -l * -i .\results\recout_step7.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict1 .\lists\monophones1

HResults -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step7.mlf

输出结果如下：

```
====================== HTK Results Analysis =======================
  Date: Mon Oct 30 20:20:52 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step7.mlf
```

```
---------------------- Overall Results -------------------------
SENT: %Correct=0.00 [H=0, S=15, N=15]
WORD: %Corr=94.12, Acc=41.18 [H=64, D=0, S=4, I=36, N=68]
================================================================
```

看着上面的结果，你不感到奇怪么？话语级识别正确率怎么是零！通过分析识别输出文件 recout_step7.mlf，发现在每一句上都加上了 SENT-START 和 SENT-END。这是与标注真值文本无法完全吻合的黑手呀！一个解决办法是在运行 HResults 时加入-e 选项来忽略掉 SENT-START 和 SENT-END，如下所示：

```
HResults -e ??? SENT-START -e ??? SENT-END -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step7.mlf
```

这时的输出结果：

```
====================== HTK Results Analysis =======================
  Date: Mon Oct 30 20:33:27 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step7.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=66.67 [H=10, S=5, N=15]
WORD: %Corr=94.12, Acc=85.29 [H=64, D=0, S=4, I=6, N=68]
================================================================
```

上面的方案可以解决问题，但不是很完美。应该在获得的识别结果中（recount_step7.mlf）不产生 SENT-START 和 SENT-END 才好。分析 dict1 字典发现，对应 SENT-START 和 SENT-END 的相关信息如下：

```
SENT-END        sil
SENT-START      sil
```

应该加入[]以表示他们并不输出任何东西。把 dict1 字典另存为 dict2 并替换上面两行成如下形式：

```
SENT-END        []  sil
SENT-START      []  sil
```

重新运行问题解决。相应的，HResults 的参数可以省掉-e 了：

```
HVite -H .\hmms\hmm7\macros -H .\hmms\hmm7\hmmdefs -S test.scp -l * -i .\results\recout_step7_2.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict2 .\lists\monophones1

HResults -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step7_2.mlf
```

[step 8]重校准训练数据

确认 trainwords.mlf 中的路径为”*/S0???.lab”，修改 dict2 加入 silence sil 一项，另存为 dict3，执行 HVite 进行 Viterbi 校准：

```
HVite -l * -o SWT -b silence -C .\config\config1 -a -H .\hmms\hmm7\macros -H .\hmms\hmm7\hmmdefs -i .\labels\aligned.mlf -m -t 350.0 -y lab -I .\labels\trainwords.mlf -S train.scp .\dict\dict3 .\lists\monophones1
```

重估两次：

```
HERest -C .\config\config1 -I .\labels\aligned.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm7\macros -H .\hmms\hmm7\hmmdefs -M .\hmms\hmm8 .\lists\monophones1

HERest -C .\config\config1 -I .\labels\aligned.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm8\macros -H .\hmms\hmm8\hmmdefs -M .\hmms\hmm9 .\lists\monophones1
```

我们再来看看这时的识别率怎么样。

```
HVite -H .\hmms\hmm9\macros -H .\hmms\hmm9\hmmdefs -S test.scp -l * -i .\results\recout_step8.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict2 .\lists\monophones1

HResults -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step8.mlf
```

这时的输出结果：

```
====================== HTK Results Analysis =======================
  Date: Mon Oct 30 21:06:51 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step8.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=73.33 [H=11, S=4, N=15]
WORD: %Corr=97.06, Acc=88.24 [H=66, D=0, S=2, I=6, N=68]
============================================================
```

可以看出，识别结果比校准前有不小的提高。我们继续重估两次测试一下结果，看看会出现什么情况：

```
HERest -C .\config\config1 -I .\labels\aligned.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm9\macros -H .\hmms\hmm9\hmmdefs -M .\hmms\hmm9_1 .\lists\monophones1

HERest -C .\config\config1 -I .\labels\aligned.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm9_1\macros -H .\hmms\hmm9_1\hmmdefs -M .\hmms\hmm9_2 .\lists\monophones1

HVite -H .\hmms\hmm9\macros -H .\hmms\hmm9\hmmdefs -S test.scp -l * -i .\results\recout_step8_2.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict2 .\lists\monophones1

HResults -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step8_2.mlf
```

识别结果如下：

```
====================== HTK Results Analysis =======================
  Date: Mon Oct 30 21:18:34 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step8_2.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=73.33 [H=11, S=4, N=15]
```

```
WORD: %Corr=97.06, Acc=88.24 [H=66, D=0, S=2, I=6, N=68]
============================================================
```

发现识别结果不再提高!下面通过绑定状态的三音素模型来进一步提高识别效果。

注 7：hmm8 和 hmm9_1 下的模型文件并没有包含在压缩包里。

3. 创建绑定状态的三音素 HMM 模型

目的是加入上下文依赖（context-dependent）三音素模型并得到稳健的训练。包括两步，先由单音素得到三音素并重估参数，第二步就是绑定三音素的状态以使输出分布更加稳健。

[step 9]得到三音素 HMM

上下文依赖三音素模型可以用单音素作为初始，再进行重估。由于重估时要三音素级标注文本，就先生成标注文本。

HLEd -n .\lists\triphones1 -l * -i .\labels\wintri.mlf mktri.led .\labels\aligned.mlf

编辑脚本 mktri.led 包括如下命令：

文件名：mktri.led
WB sp WB sil TC

其中文件 wintri.mlf 是由单音素标注文本文件 aligned.mlf 转换成的等价三音素标注文本。一个三音素的列表保存到了 triphones1。

下面利用 HMM 编辑器初始化三音素模型。所用命令为：

HHEd -H .\hmms\hmm9\macros -H .\hmms\hmm9\hmmdefs -M .\hmms\hmm10 mktri.hed .\lists\monophones1

其中的 mktri.hed 文件由 perl 脚本生成：

perl .\scripts\maketrihed .\lists\monophones1 .\lists\triphones1

修改 mktri.hed 文件，把第一行的.\lists\triphones1 改成./lists/triphones1。

运行 HHEd 得到如下警告，没有大碍的：

```
 WARNING [-2631]  ApplyTie: Macro T_sp has nothing to tie of type t in HHEd
 WARNING [-2631]  ApplyTie: Macro T_sil has nothing to tie of type t in HHEd
```

重估两次：

HERest -C .\config\config1 -I .\labels\wintri.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm10\macros -H .\hmms\hmm10\hmmdefs -M .\hmms\hmm11 .\lists\triphones1

HERest -C .\config\config1 -I .\labels\wintri.mlf -t 250.0 150.0 1000.0 -s stats -S train.scp -H .\hmms\hmm11\macros -H .\hmms\hmm11\hmmdefs -M .\hmms\hmm12 .\lists\triphones1

两次均会有如下警告，不用理会：

```
Pruning-On[250.0 150.0 1000.0]
 WARNING [-2331]  UpdateModels: ax-n[7] copied: only 1 egs in HERest
```

```
WARNING [-2331]    UpdateModels: l-y+ax[26] copied: only 1 egs in HERest
WARNING [-2331]    UpdateModels: y-ax+n[39] copied: only 1 egs in HERest
WARNING [-2331]    UpdateModels: uw-l+y[43] copied: only 1 egs in HERest
```

试着执行一下识别任务，看看怎么样：

```
HVite -H .\hmms\hmm12\macros -H .\hmms\hmm12\hmmdefs -S test.scp -l * -i recout_step9.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict2 .\lists\triphones1
```

暂时未通过，报错：

```
  ERROR [+8231]    GetHCIModel: Cannot find hmm [y-]uw[+???]
 FATAL ERROR - Terminating program HVite
```

令 config2 在 config1 基础上加入 FORCECXTEXP=T，ALLOWXWRDEXP=F，仍没有改善。通过分析 dict2，发现 y-uw+?出现在单词 SUE 的第二种发音上。注释掉这一行，仍然报错，但跟上次的错误有了可喜的差别：

```
  ERROR [+8231]    GetHCIModel: Cannot find hmm [ao-]r[+???]
 FATAL ERROR - Terminating program HVite
```

看来路子是对的。ao-r+?出现在单词 FOUR 的第二种发音上，再次注释掉，把进行了如上两行注释的字典存为 dict4。重新运行 HVite，不再出错：

```
HVite -H .\hmms\hmm12\macros -H .\hmms\hmm12\hmmdefs -S test.scp -l * -i .\results\recout_step9.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict4 .\lists\triphones1
```

dict4 字典如下所示：

文件名：.\dict\dict4

```
CALL            k ao l sp
DAVE            d ey v sp
DIAL            d ay ax l sp
EIGHT           ey t sp
FIVE            f ay v sp
FOUR            f ao sp
#FOUR           f ao r sp
JULIAN          jh uw l ia n sp
JULIAN          jh uw l y ax n sp
LAW             l ao sp
LEE             l iy sp
NINE            n ay n sp
OH              ow sp
ONE             w ah n sp
PHIL            f ih l sp
PHONE           f ow n sp
SENT-END    []  sil
SENT-START  []  sil
SEVEN           s eh v n sp
SIX             s ih k s sp
STEVE           s t iy v sp
```

```
SUE             s uw sp
#SUE             s y uw sp
THREE           th r iy sp
TWO              t uw sp
TYLER           t ay l ax sp
WOOD             w uh d sp
YOUNG            y ah ng sp
ZERO            z ia r ow sp
```

进行识别验证：

HResults -I .\labels\testwords.mlf .\lists\monophones1 .\results\recout_step9.mlf

所得结果如下：

```
====================== HTK Results Analysis =======================
  Date: Mon Oct 30 22:45:31 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step9.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=93.33 [H=14, S=1, N=15]
WORD: %Corr=100.00, Acc=97.06 [H=68, D=0, S=0, I=2, N=68]
============================================================
```

可以看出，基于三音素的 HMM 比基于单音素的 HMM 有较大的性能提升。

我们进一步进行讨论 dict4 的由来。由于在 step 8 中对标记文本和语音数据进行了校准，对于象 FOUR，SUE 这样有多个发音的单词，在 Vitebi 算法的作用下，会选择最大化似然率的单词，而不是象 step 4 那样选择第一种读音。结果使 dict2 中的发音在扩展成三音素时在音素级真值文本中没有实例，当然也就找不到该三音素的 HMM 模型。这时会报错。根据这个原理，可以编一个 Perl 脚本自动对字典中没用到的单词注释掉。我把这个脚本命名为 makedict.pl，包含在 scripts 文件夹里。它可以代替上面的手工注释工作，使用下面的命令可以得到 dict4：

perl .\scripts\makedict.pl .\dict\dict2 .\dict\dict4 .\lists\triphones1

注 8：hmm11 下的模型文件并没有包含在压缩包里。

[step 10]绑定三音素

在上一步估计模型时，因数据不足导致很多分布的方差只好用截至方差 vFloors。这一步就是通过绑定状态来共享数据，使输出分布更加的稳健。HHEd 提供两种聚类状态的机制，这里采用的是决策树：

HHEd -H .\hmms\hmm12\macros -H .\hmms\hmm12\hmmdefs -M .\hmms\hmm13 tree.hed .\lists\triphones1 > log

上面的 tree.hed 用 perl 脚本 mkclscript.prl (.\scripts\目录下)生成：

perl mkclscript.prl TB 350.0 .\lists\monophones1>tree.hed

这时候的 tree.hed 的内容：

文件名：中间 tree.hed
TB 350 "k_s2" {("k","*-k+*","k+*","*-k").state[2]} TB 350 "ao_s2" {("ao","*-ao+*","ao+*","*-ao").state[2]} …… TB 350 "z_s4" {("z","*-z+*","z+*","*-z").state[4]}

还要往 tree.hed 中插入问题集和 trace 等信息。最终形式如下：

文件名：最终 tree.hed
RO 100.0 stats TR 0 QS "R_NonBoundary"{ *+* } …… QS "L_z" { z-* } TR 2 TB 350 "k_s2" {("k","*-k+*","k+*","*-k").state[2]} TB 350 "ao_s2" {("ao","*-ao+*","ao+*","*-ao").state[2]} …… TB 350 "z_s4" {("z","*-z+*","z+*","*-z").state[4]} TR 1 ST trees CO ./lists/tiedlist AU ./lists/fulllist

在 tree.hed 中，tiedlist 是绑定后的不同三音素列表。fulllist 是输入参数，代表绑定前的所有三音素列表，所以要事先制作。HTK book 的制作方法：

HDMan -b sp -n .\lists\fulllist -g global2.ded -l flog .\dict\beep-tri .\dict\beep

其中，文件 global2.ded 是在 global.ded 基础上加入了 TC 命令：

文件名：global2.ded
AS sp RS cmu MP sil sil sp TC

HDMan 这里所做的工作包括 1）按照 global.ded 的配置把 beep 字典中的发音扩展成（词内）三音素形式，保存到 beep-tri 字典；2）提取 beep-tri 中出现的所有不同三音素保存到 fulllist 列表中。但是你会发现，按照上述方式制作的 fulllist 在执行 HHEd 时会引发错误，下面是我得到的一个例子：

ERROR [+2662] FindProtoModel: no proto for k-b+l in hSet

FATAL ERROR - Terminating program HHEd

这是什么原因呢？其实上面错误提示中的 k-b+l 并没在我们的演示任务中出现。正因为没出

现，所以 HMM 模型文件里没它的位置。要解决这个错误，有两种解决途径。第一个路子，既然提示模型文件里没有三音素的模型，那就给它制作一个。可以把单音素*-b+*的模型复制一份给 k-b+l，对于其他没模型的三音素也这样处理。我们这里不用这个路子，因为工作量太大了一点点。我们走第二条路，这条路在 htk 的 maillist 中有人提到过，详见 2006 年 4 月 24 日的一封信，作者是 Moustafa Nabil。这条路不用 beep 字典，而使用我们制作的任务字典。我们对 dict2 进行修改，另存为 dict5，其中去掉了下述两项：

SENT-END [] sil

SENT-START [] sil

执行 HDMan 生成 fulllist：

HDMan -b sp -n .\lists\fulllist -g global3.ded -l flog .\dict\dict5-tri .\dict\dict5

上面的 global3.ded 是在 global2.ded 基础上去掉了 AS sp 一行，因为 dict5 中的每一单词发音后已经存在 sp 了。文件 global3.ded 的内容如下：

文件名：global3.ded
RS cmu MP sil sil sp TC

重新执行 HHEd，不会再出错了。我们重估两次：

HERest -C .\config\config1 -I .\labels\wintri.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm13\macros -H .\hmms\hmm13\hmmdefs -M .\hmms\hmm14 .\lists\tiedlist

出现如下错误：

ERROR [+5010] InitSource: Cannot open source file sil

ERROR [+7010] LoadHMMSet: Can't find file

ERROR [+2321] Initialise: LoadHMMSet failed

FATAL ERROR - Terminating program HERest

在 fulllist 中加入 sil。重新运行 HHEd 一次，这时的 HERest 只有一个警告：

Pruning-On[250.0 150.0 1000.0]

WARNING [-2331] UpdateModels: ax-n[7] copied: only 1 egs in HERest

再重估一次：

HERest -C .\config\config1 -I .\labels\wintri.mlf -t 250.0 150.0 1000.0 -S train.scp -H .\hmms\hmm14\macros -H .\hmms\hmm14\hmmdefs -M .\hmms\hmm15 .\lists\tiedlist

这次的警告与上次相同，没有关系的。

4. 识别器评估

[step 11]验证测试结果

用 Viterbi 算法进行识别已经在前文又多处涉及，这里轻松了，执行如下命令：

HVite -C .\config\config2 -H .\hmms\hmm15\macros -H .\hmms\hmm15\hmmdefs -S test.scp -l * -i .\results\recout_step11.mlf -w wdnet -p 0.0 -s 5.0 .\dict\dict4 .\lists\tiedlist

其中的 config2（前文已经提到过）在 config 基础上加入 FORCECXTEXP=T，

ALLOWXWRDEXP=F。进行识别验证：
HResults -I .\labels\testwords.mlf .\lists\tiedlist .\results\recout_step11.mlf
所得结果如下：

```
====================== HTK Results Analysis =======================
  Date: Tue Oct 31 10:06:53 2006
  Ref : .\labels\testwords.mlf
  Rec : .\results\recout_step11.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=93.33 [H=14, S=1, N=15]
WORD: %Corr=100.00, Acc=98.53 [H=68, D=0, S=0, I=1, N=68]
============================================================
```

识别结果还不错吧。

下面的现场识别和话者适应应该没太大难度的，我就不赘言惹人烦了。Good Luck!

上面的实验执行过多次，在我这里都能通过，但这并不能保证所有过程都是正确的。我只是一个门外汉！如果您发现了其中的任何问题，请不吝赐教。我的 email 和 blog 分别是：tonghuasu@gmail.com 和 http://maotong.blog.hexun.com。

本文所用实验材料的下载地址在 http://su.tonghua.googlepages.com/VoiceDialSystem.zip 。