VõĐình Phong
Khoa học máy tính
MSSV: 0312060

Huấn luyện HTK 3.2.1

Nhận dạng tự động tiếng nói (ASR)

Lý do

Anh em học môn ASR đã khổ sở hơn một tháng nay vào việc huấn luyện HTK. Vấn đề không khó nhưng cần sự tỉ mỉ, kiên trì và hướng dẫn từ thầy Hạ. Sau khi hoàn thành việc huấn luyện ở trường, mình nghĩ nên làm một cái tool, trước hết là phục vụ cho mình, và cho những ai chưa huấn luyện xong HTK. Với quĩ thời gian một ngày rưỡi ngắn ngủi tranh thủ "xuất bản" để tất cả chúng ta có kịp thời gian nộp bài vào thứ 7 tới, mình đã vọc lại mớ dòng lệnh và tìm hiểu thêm ý nghĩa của nó. Những gì trình bày dưới đây là từ hướng dẫn của thầy Hạ và những kinh nghiệm làm sai của mình lẫn bạn bè. Ngoài ra mình có giải thích thêm một số ý nghĩa thông qua việc đọc HTK Book. Nhiều vấn đề còn chưa hiểu thấu đáo cũng như vốn từ dịch thuật củ chuối chắc chắn để lại nhiều thắc mắc các bạn khi đọc bài này. Nhưng rất mong các bạn đóng góp cho sự hoàn thiện của tutorial để anh em ta có thể đỡ "trâu bò" hơn cũng như các bạn năm sau đỡ khổ. Mong anh em ủng hộ. Xin cảm ơn.

Thông tin chung

1.Cấu trúc thư mục

Tạo thư mục ngoài cùng với tên tùy ý (trong bài này giả định tất cả để trong thư mục gốc C:\, tuy nhiên điều này không được khuyến khích).

Các thư mục con gồm có

- hmm0 – hmm15: các thư mục chứa file MMF.
- cfg (config): chứa các file config cho một số lệnh.
- ins (instruction): chứa các file .hed và .led.
- mlf (master label file): chứa các file .mlf.
- ph (phones): chứa các file phones: mono, tri.
- pl (Perl script): chứa các file script viết bằng Perl.
- txt (other files): chứa các file linh tinh như từ điển, danh sách file, wdnet, gram, train…
- wave: chứa các file Wave và mfcc.

2.Một vài lưu ý

Nhớ thiết lập biến môi trường cho HTKTools và Perl.

Các hướng dẫn trong bài này đều giả sử các tập tin âm thanh đã được thu trước đó.

Các tiến trình là bán tự động, yêu cầu người dùng gõ các lệnh thi hành các file Perl.

Thư mục yourproject chứa một số tập tin khởi đầu tối thiểu để người dùng có thể bắt đầu thực hiện theo chỉ dẫn.

Thư mục sample chứa các tập tin đã hoàn chỉnh các bước trong bài này.

Bạn có thể dùng allin1.bat (all in one) để chạy tự động các lệnh.

Người dùng nếu tìm thấy lỗi làm ơn thông báo đến mọi người khác, và nếu có thể, xin gởi mail về địa chỉ phongkhtn@yahoo.com. Xin cảm ơn nhiều.

Chuẩn bị dữ liệu

1.Tạo cấu trúc văn phạm

Cấu trúc văn phạm là một đồ thị có hướng tổng quát. Nó chứa các cấu trúc câu có thể có trong ngữ cảnh của ứng dụng mà ta muốn dùng ASR. Ví dụ nếu ta muốn áp dụng ASR trong môi trường chứng khoán, cấu trúc văn phạm phải có những từ (word) và câu (sentence, cấu thành từ word) được thiết kế sao cho nó có thể trình bày tất cả các mẫu câu mà một người trong ngữ cảnh đó có thể nói về.

Tập tin cấu trúc văn phạm

```
//gram.txt

$digit = moojt | hai | ba | boosn | nawm | sasu | bary | tasm
| chisn | khoong;

(<$digit>)
```

```
C:\>HParse txt/gram.txt txt/wdnet.txt
```

wordnet tạo ra sẽ có định dạng giống file lattice:

```
//wdnet.txt

VERSION=1.0
```

```
N=13   L=31

I=0    W=khoong

I=1    W=!NULL

I=2    W=chisn

I=3    W=tasm

I=4    W=bary

I=5    W=sasu

I=6    W=nawm

I=7    W=boosn

I=8    W=ba

I=9    W=hai

I=10   W=moojt

I=11   W=!NULL

I=12   W=!NULL

J=0     S=1    E=0

J=1     S=12   E=0

J=2     S=0    E=1

J=3     S=2    E=1

J=4     S=3    E=1

J=5     S=4    E=1

J=6     S=5    E=1

J=7     S=6    E=1

J=8     S=7    E=1

J=9     S=8    E=1

J=10    S=9    E=1

J=11    S=10   E=1
```

```
J=12    S=1    E=2

J=13    S=12   E=2

J=14    S=1    E=3

J=15    S=12   E=3

J=16    S=1    E=4

J=17    S=12   E=4

J=18    S=1    E=5

J=19    S=12   E=5

J=20    S=1    E=6

J=21    S=12   E=6

J=22    S=1    E=7

J=23    S=12   E=7

J=24    S=1    E=8

J=25    S=12   E=8

J=26    S=1    E=9

J=27    S=12   E=9

J=28    S=1    E=10

J=29    S=12   E=10

J=30    S=1    E=11
```

Để rõ hơn, xem cấu trúc file lattice SLF trong chương 20 phần phụ lục HTK Book.

2.Tạo từ điển

Muốn xây dựng từ điển thì bước đầu tiên là tập hợp tất cả các từ được dùng trong ngữ cảnh. Các từ này được xếp thứ tự alphabet trong tập tin và phải được phiên âm tương ứng. Qui cách phiên âm rất quan trọng, có thể hạ thấp chất lượng nhận dạng nếu không cẩn thận.

```
#dict.dct

ba          b     a     sp
bary   b    ar    y     sp
boosn  b    oos   n     sp
chisn  ch   is    n     sp
hai         h     a     i     sp
khoong      kh    oo    ng    sp
moojt  m    ooj   t     sp
nawm   n    aw    m     sp
sasu   s    as    u     sp
tasm   t    as    m     sp
silence     sil
```

Lưu ý có phone sp (short pause) trong từ điển.

Đối với bộ từ vựng nhỏ thì tốt nhất là gõ tay hoặc copy từ bộ từ điển có sẵn.
Sau khi có từ điển thì chúng ta có thể tạo ra bộ phiên âm, gọi là monophones0

```
HDMan -m -w txt/wlist -n ph/monophones -l dlog txt/dict
txt/dict.dct
```

Giải thích

wlist: đầu vào, đơn giản là danh sách các từ được sử dụng trong wordnet, mỗi từ một dòng. wlist vẫn chưa có cho đến khi ta đến bước này. Để tạo wlist, ta cần có prompts. Prompts là gì ? Đây là một tập tin chứa các đoạn text yêu cầu utterance đọc vào. Utterance phải đọc từng câu trong prompts và lưu vào file .wav có tên tương ứng.

Tạo Prompts

```
C:\>HSGen.exe   -l   -n   10   txt/wdnet.txt   txt/dict.dct   >>
txt/prompts
```

Ở đây ta tạo tập tin prompt có tất cả 10 câu.

```
#prompts

001 bary chisn chisn hai boosn moojt khoong
002 khoong boosn tasm bary bary nawm khoong
003 ba nawm moojt chisn sasu bary khoong
004 bary boosn bary tasm nawm sasu nawm
005 bary nawm ba chisn boosn chisn nawm
006 bary boosn boosn khoong moojt tasm nawm
```

```
007 moojt hai bary moojt bary tasm khoong
008 khoong boosn hai bary sasu bary nawm
009 moojt ba ba hai chisn nawm nawm
010 tasm nawm sasu tasm bary bary nawm
```

Sau khi đã tạo prompts thì ta sẽ tạo được wlist bằng Perl script như sau:

```
C:\> perl pl/prompts2wlist.pl txt/prompts txt/wlist
```

Như vậy ta đã có wlist, có cấu trúc như sau:

```
#wlist

ba
bary
boosn
chisn
hai
khoong
khoong
moojt
nawm
sasu
tasm
```

monophones1: đầu ra, danh sách các phones được dùng để phiên âm trong từ điển.

```
#monophones1

sil
b
a
sp
ar
y
oos
n
ch
is
h
i
kh
oo
ng
m
ooj
t
aw
```

```
s
as
u
```

beep: Đầu vào, từ điển phiên âm.

names: Đầu vào, từ điển tên riêng (tùy chọn).
dict: Đầu ra, từ điển mới được tạo, tổng hợp từ beep. names và wlist.

Lưu ý: *Nếu beep có silence thì HDMan sẽ bỏ silence đi, làm cho dict mới bị thiếu silence. Vì sao ? Bởi vì HDMan sẽ tìm trong tất cả các từ trong beep mà nó có trong wlist. Mà wlist lại được tạo từ prompts. Prompts không chứa silence. Do đó, wlist, và hầu quả là, dict mới tạo cũng không chứa silence. Tốt nhất là ta cứ tạo dict cho có hình thức, nhưng không dùng. Ta chỉ dùng monophones0 mà nó tạo ra. Ở các bước sau ta chỉ dùng beep thôi. Nhưng nếu beep thực sự quá lớn trong khi wlist lại nhỏ, ta hiệu chỉnh lại dict để dùng. Việc hiệu chỉnh rất đơn giản, chỉ cần thêm silence – sil là được. Việc này được tự động hóa bằng Perl script.*

Thêm nữa, việc đặt tên monophones0 hay monophones1 trong dòng tham số lệnh là không quan trọng. Dù thế nào thì sau khi tạo, monophones# sẽ luôn có âm sp và thiếu sil. Thêm sil và giữ nguyên sp thì có được monophones1.

Thêm sil và xóa sp thì có được monophones0.

```
C:\>perl pl/mkMonophones.pl ph/monophones ph/monophones0
ph/monophones1
```

Giải thích

createMonophones.pl: Đầu vào, perl script.

monophones: Đầu vào, tập tin monophones sau được tạo ở HDMan.

monophones0: Đầu ra, tập tin monophones0 sau khi hiệu chỉnh.

monophones1: Đầu ra, tập tin monophones0 sau khi hiệu chỉnh.

3.Thu dữ liệu

Bước này bao gồm các công việc sau:

1) Thu âm giọng đọc lưu trữ thành các tập tin .wav.

2) Viết tập tin transcript tương ứng với mỗi file.

hoặc:

1) Tạo tập tin transcript (đã được tạo tự động bằng HSGen trong bước trước, chính là tập tin prompts).

2) Thu âm giọng đọc lưu trữ thành các tập tin .wav bằng cách đọc theo transcription trong tập tin prompts.

Như vậy ta có tất cả 2 cách thu dữ liệu:

- Tự động tạo prompts và ghi âm theo.
- Ghi âm và tạo prompts sau.

4.Tạo tập tin transcription

HTK không sử dụng file prompts cho xử lý sau này. Ta cần tạo ra một số file khác, cụ thể là dạng file MLF (Master Label File, tham khảo thêm trong HTK Book).

Có hai loại tập tin MLF cần tạo ra:

MLF ở mức từ (word), tức là tập tin words.mlf định dạng tập tin prompts theo chuẩn MLF:

```
//words.mlf

#!MLF!#
"*/001.lab"
bary
chisn
chisn
hai
boosn
moojt
khoong
.
"*/002.lab"
khoong
boosn
tasm
bary
bary
nawm
khoong
.................
```

Làm sao tạo ?

```
C:\> Perl pl/prompts2mlf.pl mlf/words.mlf txt/prompts
```

MLF ở mức âm (phones), tức là định dạng tập tin phones0.mlf và phones1.mlf theo MLF:

```
//phones1.mlf

"*/002.lab"
sil
kh
oo
ng
b
oos
n
t
as
m
b
ar
y
b
ar
y
n
aw
m
kh
oo
ng
sil
.
```

Thực ra, phones#.mlf là dạng khai triển của words.mlf ở mức âm.

Làm sao tạo?

```
C:\> HLEd -l * -d txt/dict.dct -i mlf/phones0.mlf
ins/mkphones0.led mlf/words.mlf
```

Giải thích

-d dict: đầu vào, từ điển ta đã có từ trước.
words.mlf: đầu vào, vừa được tạo ở trên.

-i phones0.mlf: đầu ra.
mkphones0.led: chứa các lệnh script để chuyển words.mlf thành phones0.mlf

```
#mkphones0.led
EX
IS sil sil
DE sp
```

Giải thích

EX: Thay thế mỗi từ trong words.mlf bằng phiên âm tương ứng trong từ điển dict.
IS: Chèn mô hình lặng (silence - sil) vào đầu và cuối của một từ.
DE: Xóa tất cả các short pause (sp) được thêm vào sau lệnh EX.

Tạo phones1.mlf

Vì sao lại cần phones0.mlf và phones1.mlf ? Ai mà biết HTK muốn gì ! Nhưng đại khái là phones0.mlf không chứa âm sp, còn phones1.mlf thì có. Thêm sp vào chắc với mục đích tăng tính hiệu quả cho quá trình nhận dạng sau này.

Cách tạo phones1.mlf ?

```
C:\> HLEd -l * -d txt/dict.dct -i mlf/phones1.mlf
ins/mkphones1.led mlf/words.mlf
```

Giải thích

-d dict: đầu vào, từ điển ta đã có từ trước.
words.mlf: đầu vào, vừa được tạo ở trên.
-i phones1.mlf: đầu ra.
mkphones1.led: chứa các lệnh script để chuyển words.mlf thành phones1.mlf

```
//mkphones1.led
EX
IS sil sil
```

Nhận xét: *phones1.mlf được tạo đơn giản bằng cách bỏ lệnh xóa sp trong mkphones1.led*.

5.Mã hóa dữ liệu

Tại bước này, các file âm thanh mà ta đã thu ở bước 3 sẽ được rút đặc trưng. HTK hỗ trợ 2 dạng đặc trưng MFCC và LPC. MFCC nên được sử dụng vì nó tốt hơn (không tin thì thử LPC xem). Các thông tin cấu hình khác được lưu trong tập tin cấu hình config_HCopy.txt:

```
#config_HCopy.txt
#coding parameters - HCopy
SOURCEKIND = WAVEFORM
SOURCEFORMAT = WAV
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
```

```
C:\>HCopy -T 1 -C cfg/HCopy.cfg -S txt/listwavmfc
```

Giải thích

-S listwavmfc: Đầu vào, chứa danh sách file wave - file mfc tương ứng.

Làm sao tạo ?

```
C:>perl pl/listwavmfc.pl wave txt/listwavmfc
```

Giải thích

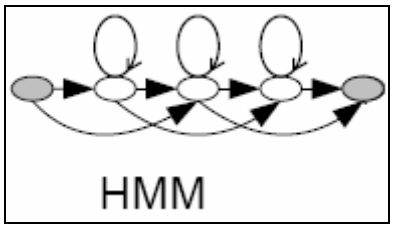
listwavmfc.pl: Tập tin script.
Wave: Đầu vào, thư mục chứa các tập tin .wav, ở đây giả sử là nằm ở thư mục C:\.
listwavmfc: Đầu ra, tập tin text chứa danh sách file wave.

Tạo monophone HMMs

Ở giai đoạn này, chúng ta sẽ tạo và huấn luyện cho monophones HMM xấp xỉ bằng một hàm Gauss. Đầu tiên, tất cả các xấp xỉ của các âm đều có hàm Gauss như nhau (kỳ vọng và phương sai). Sau khi được huấn luyện, âm sp và silence lần lượt được thêm vào. Cuối cùng, chúng được huấn luyện lại.

1.Tạo Flat Start Monophones

Tại bước này, chúng ta sẽ định nghĩa ra một "đề cương" cho HMM (chữ đề cương "dịch " từ chữ prototype). Việc gán thông tin nào cho prototype là không quan trọng, chủ yếu là xây dựng một cái khung. Một mô hình tốt mà HTK Book đề xuất là mô hình 3 trạng thái trái – giữa – phải tuần tự.




```
#proto

~o <VecSize> 39 <MFCC_0_D_A>

~h "proto"

<BeginHMM>

<NumStates> 5

<State> 2

 <Mean> 39

 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 <Variance> 39

 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

<State> 3

 <Mean> 39
```

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

 <Variance> 39

 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1

<State> 4

 <Mean> 39

 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

 <Variance> 39

 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1

<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>
```

Giải thích thêm

Số chiều Vector đặc trưng MFCC_0_D_A là 39 = 13 tĩnh (MFCC_0) + 13 hệ số delta + 13 hệ số acceleration.

Tuy là có 5 trạng thái nhưng 2 trạng thái đầu và cuối không xét.

Các vector kỳ vọng (mean) và phương sai (variance) đều bằng nhau.

Ma trận xác suất chuyển được khởi tạo theo kinh nghiệm.

1.1.Làm sao tạo proto?

```
C:\>HCompV -C cfg/HCompV.cfg -f 0.01 -m -S txt/train.scp -M
hmm0 hmm0/proto
```

Giải thích

-C HCompV.cfg: Đầu vào, tập tin cấu hình để HCompV tham khảo, có nội dung như sau:

```
#HCompV.cfg

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
```

-f 0.01: Tham số đầu vào, yêu cầu xuất file vFloor chứa vector floor có giá trị bằng 0.01 vector variance.

-S train.scp: Đầu vào, chứa danh sách các tập tin đặc trưng mfc. Cách đơn giản để tạo file train.scp là dùng lệnh dir của hệ thống (có thể dùng Perl).

Làm sao?

```
C:\Wave>dir /B *.mfc > ../train.scp
```

Hoặc

```
C:\>dir /B Wave\*.mfc > train.scp
```

Tuy nhiên ta phải điều chỉnh tập tin train.scp bằng tay để thêm đường dẫn tuyệt đối cho từng tập tin .mfc được liệt kê trong đó. Ta có thể tránh điều này bằng cách sử dụng Perl script:

```
C:\>perl pl/mkTrainFile.pl wave txt/train.scp
```

Giải thích

CreateTrainFile.pl: Đầu vào, file script Perl.

Wave: Đầu vào, thư mục chứa các tập tin .mfc.

train.scp: Đầu ra, tên tập tin chứa danh sách file .mfc.

-M hmm0: Đầu vào, thư mục mà HCompV sẽ dùng để chứa proto (phải được tạo trước).

Hmm0/proto.txt: Đầu vào, tập tin chứa cấu trúc proto như phần trên đã trình bày (nhớ là lưu trong thư mục hmm0).

1.2.

Sau khi chạy HCompV, hai tập tin proto và vFloors được tạo ra trong thư mục hmm0. Thông thường, tiết mục tiếp theo là "cắt may" thủ công tập tin hmmdefs từ proto và monophones0. Tuy nhiên, chúng ta có script để tự động hóa chuyện này.

Tạo macros tự động

```
C:\>perl pl/mkMacrosFile.pl hmm0/vFloors hmm0/macros
```

Giải thích

createMacrosFile.pl: Đầu vào, perl script.

hmm0/vFloors: Đầu vào, file vFloors được tạo từ lệnh HCompV ở trên.

hmm0/macros: Đầu ra, file macros cần tạo.

Tạo hmmdefs tự động

```
C:\>perl pl/mkHmmdefsFile.pl hmm0/proto ph/monophones0
hmm0/hmmdefs
```

Giải thích

createHmmdefsFile.pl: Đầu vào, perl script;

hmm0/proto: Đầu vào, tập tin proto có được từ bước trước.

monophones0: Đầu vào, tập tin monophones0 có từ bước .

hmm0/hmmdefs: Đầu ra, tên tập tin hmm.

Sau khi đã có được hmmdefs và macros, chúng ta sử dụng HERest để tái ước lượng các tham số trong hmmdefs. Vì sao phải ước lượng lại ? Cần nhớ là khi ta tạo tập tin hmmdefs ở trên, mô hình xấp xỉ Gauss là như nhau cho mọi phones (mean và variance đều giống nhau và giống mô hình trong file proto). Với HERest, nó sẽ sử dụng thông tin trong các tập tin đặc trưng .mfc để ước lượng lại thông số hàm xấp xỉ.

```
C:\> HERest -C cfg/HERest.cfg -I mlf/phones0.mlf -t 250.0
150.0 1000.0  -S txt/train.scp -H hmm0/macros -H hmm0/hmmdefs
-M hmm1 ph/monophones0
```

Giải thích

-C HERest.cfg: Đầu vào, tập tin cấu hình

```
#HERest.cfg

# Coding parameters

TARGETKIND = MFCC_D_A_0

TARGETRATE = 100000.0

SAVECOMPRESSED = T

SAVEWITHCRC = T

WINDOWSIZE = 250000.0

USEHAMMING = T

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

ENORMALISE = F
```

-I mlf/phones0.mlf: Đầu vào, tập tin MLF được tạo từ trước.

-t 250.0 150.0 1000.0: Đầu vào, tham số prunning.

-S txt/train.scp: Đầu vào, danh sách các file .mfc.

-H hmm0/macros: Đầu vào, vừa tạo.

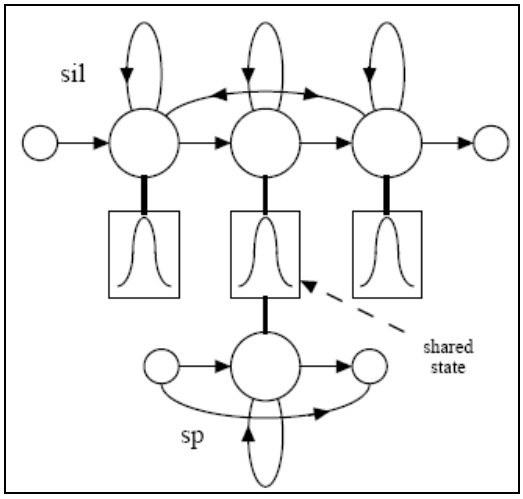
-H hmm0/hmmdefs: Đầu vào, vừa tạo.

-M hmm1: Đầu ra, chứa tập tin hmmdefs và macros mới.

ph/monophones0: Đầu vào, danh sách các phones (ngoại trừ sp).

Sau khi đã có hmm1, ta huấn luyện tiếp hmm2 và hmm3 bằng HERest. Lưu ý là khi huấn luyện hmm2 thì hmmdefs và macros là của hmm1, tương tự như vậy với hmm3, là các file của hmm2.

2.Fixing the Silence Models



Bước này thêm vào mô hình silence hai bước chuyển từ trạng thái 2 đến trạng thái 4 và ngược lại (như hình). Mô hình short pause (sp) cũng được thêm vào trạng thái trung tâm của mô hình sil. Vì sao làm vậy và tác dụng ra sao thì chưa được rõ.

Có hai bước nhỏ phải thực hiện:

Thêm mô hình sp vào hmmdefs4

```
C:\>perl pl/makesp.pl hmm3/hmmdefs hmm4/hmmdefs hmm3/macros
hmm4/macros
```

Chạy HHEd để thực hiện việc "trói buộc" mô hình sil và sp với nhau, đồng thời thêm các xác suất chuyển cho mô hình sil.

```
C:\> HHEd -H hmm4/macros -H hmm4/hmmdefs -M hmm5 ins/sil.hed
ph/monophones1
```

Giải thích

-H hmm4/macros: Đầu vào, tập tin macros trong hmm4.

-H hmm4/hmmdefs: Đầu vào, tập tin hmmdefs trong hmm4.

-M hmm5: Đầu ra hmm5.

ph/monophones1: Đầu vào, tập tin sanh sách âm, có chứa sil và sp.

ins/sil.hed: Đầu vào, tập tin chứa lệnh điều chỉnh.

```
#sil.hed

AT 2 4 0.2 {sil.transP}

AT 4 2 0.2 {sil.transP}

AT 1 3 0.3 {sp.transP}

TI silst {sil.state[3],sp.state[2]}
```

Giải thích lệnh

AT: thêm các xác suất dịch chuyển cho các dịch chuyển 2 – 4, 4 – 2 trong ma trận transition của mô hình sil (mở file hmm5/hmmdefs để kiểm chứng).

```
#hmm5/hmmdefs

~h "sil"

<BEGINHMM>

<NUMSTATES> 5

<STATE> 2

<MEAN> 39

…
```

```
<VARIANCE> 39
```

```
…
```

```
<GCONST> 5.238852e+001

<STATE> 3

~s "silst"

<STATE> 4

<MEAN> 39

…

<VARIANCE> 39

…

<GCONST> 1.008743e+002

<TRANSP> 5

 0.000000e+000   1.000000e+000   0.000000e+000   0.000000e+000
0.000000e+000

 0.000000e+000   7.391776e-001   6.082239e-002   2.000000e-001
0.000000e+000

 0.000000e+000   0.000000e+000   3.656323e-001   6.343677e-001
0.000000e+000

 0.000000e+000   2.000000e-001   0.000000e+000   4.123393e-001
3.876607e-001

 0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000
0.000000e+000

<ENDHMM>
```

AT 1 3 0.3 {sp.transP}: thêm xác suất dịch chuyển 1 -3 cho mô hình sp.

```
#hmm5/hmmdefs

~h "sp"

<BEGINHMM>

<NUMSTATES> 3
```

```
<STATE> 2

~s "silst"

<TRANSP> 3

 0.000000e+000 7.000000e-001 3.000000e-001

 0.000000e+000 5.000000e-001 5.000000e-001

 0.000000e+000 0.000000e+000 0.000000e+000

<ENDHMM>
```

Dễ thấy trạng thái 2 của sp cũng bị buộc với mô hình silst vừa mới tạo.

TI: thực hiện trói buộc sp và sil bằng silst, được lưu ở đầu file hmmdefs. Quan sát mô hình sil, có thể thấy trạng thái 3 liên kết với mô hình silst mới này.

```
#hmm5/hmmdefs

~s "silst"

<MEAN> 39

 -1.233204e+001   9.116629e-001   3.020478e-001   2.393242e+000
3.001888e+000   3.456290e+000   3.706096e+000   2.861026e+000
6.466328e-001   9.644089e-001   7.715054e-001   6.577221e-001
7.104364e+001  -5.122911e-002  -2.084613e-002  2.598116e-001  -
2.869415e-001  1.950579e-001  -3.822781e-002  -8.443902e-002  -
1.072719e-001   -4.904599e-002   8.692242e-002   8.164209e-003
1.979581e-002   2.325216e-001   1.303717e-001   -4.080342e-002
4.128299e-002  -2.011553e-001  -7.077198e-002  -2.321420e-001  -
9.913503e-002  -9.830537e-002  -1.752981e-001  -1.535826e-001  -
1.363883e-001 -1.232475e-001 2.405691e-001

<VARIANCE> 39

 1.300436e+001   1.042869e+001   1.641352e+001   1.574066e+001
1.265830e+001   1.167596e+001   1.477261e+001   1.271613e+001
1.887716e+001   1.765516e+001   1.965426e+001   1.868261e+001
6.421104e+000   9.842667e-001   1.060887e+000   1.250746e+000
1.715912e+000   1.131693e+000   1.550215e+000   1.346661e+000
1.496958e+000   1.858089e+000   2.148503e+000   1.762000e+000
1.668207e+000   8.345646e-001   1.546411e-001   1.906256e-001
2.265411e-001   2.813516e-001   2.009885e-001   2.763181e-001
2.499908e-001   2.784255e-001   2.954797e-001   3.488080e-001
3.192161e-001 2.714097e-001 1.329405e-001
```

```
<GCONST> 9.173016e+001
```

Ta nhận ra rằng silst đơn thuần là một hàm Gauss với mean và variance như trên.

Thực hiện HERest thêm 2 lần nữa để tạo ra hmm6 và hmm7.

3.Canh chỉnh lại dữ liệu huấn luyện

Trong từ điển phát âm có một số từ có nhiều kiểu phát âm khác nhau. Ở bước trước, HLEd chọn tùy ý một trong các kiểu phát âm. Ở bước này, chúng ta sẽ canh chỉnh lại tập tin transcription words.mlf. Nó sẽ chọn cách phiên âm nào khớp nhất so với dữ liệu ngữ âm. Đồng thời, nó sẽ thêm mô hình silence vào đầu và cuối mỗi utterance. Nên nhớ ta phải có entry silence sil trong từ điển (Điều này đã được giải quyết từ bước trước, nếu không nhớ xem lại bước tạo từ điển).

```
C:\> HVite -l * -o SWT -b silence -a -H hmm7/macros -H
hmm7/hmmdefs -i mlf/aligned.mlf -m -t 250.0 -y lab  -I
mlf/words.mlf -S txt/train.scp txt/dict.dct ph/monophones1
```

Giải thích

-b silence: Đầu vào, chèn thêm sil vào đầu và cuối utterance.

HERest được thực hiện thêm 2 lần nữa để tạo hmm8 và hmm9.

Lưu ý là ta không dùng phones1.mlf nữa mà chuyển sang aligned.mlf.

```
C:\> HERest -B -C cfg/HERest.cfg -I mlf/aligned.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm7/macros -H
hmm7/hmmdefs -M hmm8 ph/monophones1

C:\> HERest -B -C cfg/HERest.cfg -I mlf/aligned.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm8/macros -H
hmm8/hmmdefs -M hmm9 ph/monophones1
```

Tạo triphones

Giai đoạn cuối cùng trong việc xây dựng mô hình HMM là tạo các triphones phụ thuộc vào ngữ cảnh. Có hai bước nhỏ. Thứ nhất, monophones transcription (transcription đơn âm) được chuyển thành triphones transcription (transcription 3-âm). Các mô hình

triphones được ước lượng lại từ mô hình monophones. Thứ hai, các trạng thái triphones được "tied" với nhau để quá trình ước lượng tốt hơn. Tied là gì ? Nhớ lại bước trước, khi mà ta thêm mô hình sp cho hmmdefs, tied (tạm dịch là trói buộc) nghĩa là 2 ay nhiều mô hình HMM sẽ dùng chung một bộ các tham số (mean hay variance chẳng hạn).

1.Tạo triphones từ monophones

1.1

```
C:\> HLEd -n ph/triphones1 -l * -i mlf/wintri.mlf
ins/mktri.led mlf/aligned.mlf
```

Giải thích

-n ph/triphones1: Đầu ra, danh sách các triphones.

-i mlf/wintri.mlf: Đầu ra, triphones transcription.

mlf/aligned.mlf: Đầu vào, monophones transcription đã được ước lượng lại.

ins/mktri.led: Đầu vào, chứa lệnh tạo triphones từ monophones.

```
#mktri.led

WB sp

WB sil

TC
```

Giải thích lệnh

WB: coi như sp và sil là những từ ở biên (word boundary symbol), và do đó, không chuyển chúng thành triphones.

TC: chuyển tất cả các monophones thành triphones trừ các WB. Một điều đáng lưu ý là cũng có các biphones được tạo ra trong quá trình này, bởi vì chúng có một bên nằm sát biên. Xem xét ví dụ sau để hiểu rõ hơn:

```
sil th ih s sp m ae n sp

sil th+ih th-ih+s ih-s sp m+ae m-ae+n ae-n sp

biphones

triphones
```

```
word boundary symbol
```

Mô hình trên đây gọi là Word internal. Còn có hai mô hình nữa, chúng ta sẽ đề cập trong một dịp khác.

1.2.

Tiếp theo, chúng ta sẽ "nhái" mô hình monophones trong hmm9 thành triphones trong hmm10

```
C:\> HHEd -B -H hmm9/macros -H hmm9/hmmdefs -M hmm10
ins/mktri.hed ph/monophones1
```

Giải thích

-H hmm9/macros -H hmm9/hmmdefs: Đầu vào, hmm của monophones.

-M hmm10: Đầu ra, hmm10 được huấn luyện thành triphones.

Ins/mktri.hed: Đầu vào, tập tin chứa lệnh thực hiện "trói buộc" các ma trận chuyển của mỗi triphone trong tập tin triphones1.

-B: Đầu vào, lưu trữ hmmdefs ở dạng nhị phân thay vì text (giảm không gian chiếm dụng).

Làm sao tạo mktri.hed?

```
C:\>perl pl/mkTriHed.pl ph/monophones1 ph/triphones1
ins/mktri.hed
```

Lý giải cách làm việc của makeTriHed.pl (chưa có).

Chúng ta sẽ nhận được một số WARNING về T_sil và T_sp. Không sao cả, chuyền này rất bình thường.

1.3.

Sau khi đã "nhái" xong, việc tiếp theo là tái ước lượng mô hình triphones này. Chúng ta cũng vẫn sử dụng HERest.

```
C:\> HERest -B -C cfg/HERest.cfg -I mlf/wintri.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm10/macros -H
hmm10/hmmdefs -M hmm11 ph/triphones1
```

Thực hiện thêm một lần nữa để có hmm12.

```
C:\> HERest -B -C cfg/HERest.cfg -I mlf/wintri.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm11/macros -H
hmm11/hmmdefs -M hmm12 ph/triphones1
```

2.Tạo trạng thái ràng buộc cho triphones (tied state triphones)

Công việc cuối cùng trong việc xây dựng mô hình là ràng buộc các các trạng thái trong các tập triphones, từ đó chia sẻ dữ liệu trong từng tập và kết quả nhận dạng sẽ tốt hơn. Mặc dù bước trước đã ràng buộc, quá trình này đòi hỏi phải tinh tế hơn một chút vì nó sẽ ảnh hưởng rất lớn đến hiệu suất nhận dạng. Ở bước này chúng ta sẽ sử dụng HHEd để gom nhóm (clusterring) các trạng thái và sau đó trói buộc các trạng thái trong cùng một nhóm với nhau. HHEd đưa ra hai tùy chọn : i) Dùng độ đo tính tương đồng giữa các trạng thái để gom nhóm (cách này gọi là hướng dữ liệu); ii) Dùng cây quyết định, dựa trên việc đưa ra các câu hỏi về trạng thái trái (left) và phải (right) của ngữ cảnh (context) của từng triphones. Cây quyết định sẽ thử tìm những ngữ cảnh (context) nào tạo ra sự khác biệt lớn nhất đối với ngữ âm và sử dụng nó để gom nhóm.

```
C:\> HHEd -B -H hmm12/macros -H hmm12/hmmdefs -M hmm13
ins/tree.hed ph/triphones1 > log
```

Khoan thực hiện lệnh này, nó sẽ được thực hiện sau khi đọc toàn bộ.

Giải thích

tree.hed: là tập hợp các chỉ thị tìm kiếm các ngữ cảnh phù hợp cho việc gom nhóm.

Hiểu & dùng tập tin tree.hed

Trong tree.hed có một số lệnh như: RO, TR, QS, TB, AU, CO, ST.

RO 100.0: Thiết lập ngưỡng ngoài là 100 (không hiểu: The outlier threshold determines the minimum occupancy of any cluster and prevents a single outlier state forming a singleton cluster just because it is acoustically very different to all the other states.) và nạp file thống kế stats đã tạo ở bước trước.

TR 0: Thiết lập trace về 0.

QS *: Nạp câu hỏi. QS (question) là do người dùng tự định nghĩa. Và định nghĩa thế nào cho hiệu quả là cả một vấn đề lớn. Perl script của ta chỉ thực hiện những việc

thiết yếu đối với QS: QS cho ngữ cảnh Left và Right. Một hình dung dễ hiểu về QS là gì có thể thông qua ví dụ sau:

```
QS "L_Class-Stop" {p-*,b-*,t-*,d-*,k-*,g-*}
```

Mỗi QS (câu hỏi) được định nghĩa bằng một tập các ngữ cảnh theo sau nó, đặt trong hai dấu ngoặc nhọn. Câu hỏi "L_Class_Stop" sẽ TRUE nếu ngữ cảnh bên trái là p,b,t,d,k hoặc g.

Các câu hỏi cụ thể hơn về consonant, vowel, nasal, diphthong,… có thể không cần nhưng nếu có sẽ tốt hơn.

TR: .

TB: Hoạt động như sau: tất cả các triphones được cho chung vào một pool (đơn giản nghĩa là một chỗ chứa). Tất cả các QS lần lượt được nạp và được dùng để phân đôi pool này làm hai pool con. QS nào làm cực đại logarit likelihood của dữ liệu huấn luyện sẽ được chọn làm nhánh đầu tiên trong cây quyết định. Quá trình này được lặp lại cho đến khi với bất cứ QS nào, mức tăng log likelihood nhỏ hơn ngưỡng mà chúng ta qui định. Giá trị ngưỡng là con số đi theo sau TB

```
(VD: TB 40 "st_y_4_" {("y","*-y+*","y+*","*-y").state[4]}).
```

AU "fulllist": tập tin chứa danh sách đầy đủ các phones: mono,bi và tri.

CO "tiedlist": có một số mô hình sẽ giống nhau cả ba trạng thái và ma trận chuyển. Lệnh này tìm kiếm các mô hình giống nhau và nén lại bằng cách trói buộc chúng với nhau (khái niệm tie đã trình bày trước đây), tạo ra một danh sách mới các mô hình, lưu trữ trong tiedlist.

Như vậy, chúng ta phải tạo fulllist.

```
C:\>perl pl/mkFullList.pl ph/monophones0
```

Tạo tập tin tree.hed

```
C:\>perl pl/mkTree.pl 40 ph/monophones0 ins/tree.hed
```

Giải thích

40: ngưỡng qui định đối với TB, có thể thay đổi tùy ý.

ins/tree.hed: Đầu ra, tập tin tree.hed cần tạo.

Đến đây thì ta có thể yên tâm thực hiện lệnh HHEd.

Ta tiếp tục chạy HERest thêm 2 lần nữa cho hmm14 và hmm15.

```
C:\>HERest -B -C cfg/HERest.cfg -I mlf/wintri.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm13/macros -H
hmm13/hmmdefs -M hmm14 ph/triphones1

C:\>HERest -B -C cfg/HERest.cfg -I mlf/wintri.mlf -t 250.0
150.0 1000.0 -s stats  -S txt/train.scp -H hmm14/macros -H
hmm14/hmmdefs -M hmm15 ph/triphones1
```

Đánh giá nhận dạng

Bộ nhận dạng bây giờ đã hoàn hoàn thành và có thể được sử dụng. Bao nhiêu công sức bỏ ra và bây giờ là giai đoạn đánh giá [^.^].

1.Nhận dạng dữ liệu test

Giả sử chúng ta muốn nhận dạng thử bộ dữ liệu gồm 10 utterances.

```
C:\> HVite –C cfg/Hvite.cfg -H hmm15/macros -H hmm15/hmmdefs
-S test/test.scp -i test/recout.mlf -w txt/wdnet.txt
txt/dict.dct tiedlist
```

Giải thích

–C cfg/Hvite.cfg: Đầu vào, tập tin cấu hình.

-H hmm15/macros -H hmm15/hmmdefs: Đầu vào

-S test/test.scp: Đầu vào, tập tin chứa danh sách các file .mfc cần nhận dạng.

-i test/recout.mlf: Đầu ra, transcription nhận dạng được.

-w txt/wdnet.txt: Đầu vào, wordnet được tạo từ những bước đầu.

txt/dict.dct: Đầu vào, từ điển phiên âm.

tiedlist: Đầu vào, danh sách phones tạo được từ lệnh CO “tiedlist” trong tree.hed.

Lưu ý

Với việc cấu tạo triphones theo kiểu word internal như đã nói phần trước, trong tập tin cấu hình Hvite.cfg cần có thêm 2 tham số FORCECXTEXP = T và ALLOWXWRDEXP=F. Muốn hiểu tại sao, xem chương 12 HTK Book.

Có thêm một vài tham số của Hvite như p, s, tùy người dùng điều chỉnh.

2.Xuất kết quả

```
C:\>HResults –f –t –I mlf/words.mlf tiedlist test/recout.mlf
> test/result
```

Kết quả nhận dạng 20 utterances đầu trong train.scp

```
---------------------- Overall Results -------------------
SENT: %Correct=40.00 [H=8, S=12, N=20]

WORD: %Corr=100.00, Acc=89.29 [H=140, D=0, S=0, I=15, N=140]

============================================================
```

Kết quả không khả quan mấy, nhưng chạy được là tốt rồi.[^_^].

Công việc tiếp theo

Bước tiếp theo là tăng số hàm Gauss dùng để xấp xỉ mô hình HMM từ 1 lên 3, 5, 7,. Đọc Mixture Incrementing trong slide môn học.

Xa hơn là Adapting HMMs.

Hy vọng hai chủ đề này sẽ sớm được cập nhật.