VoiceMaker-1.1
— HMM 音声合成用音響モデルの構築 —

音声対話技術コンソーシアム 音声合成グループ / 酒向 慎司
sako@mmsp.nitech.ac.jp

1 はじめに

Galatea Toolkit の音声合成モジュール (GalateaTalk) で提供されている話者モデルは，実際に発声された大量の音声データを元に作られたものです．これは，音声認識における音響モデルと同じように，様々な音の特徴を音素を基本単位としてモデル化したもので，さらにアクセントや時間長など，実際に発声された音声に現れる様々な変動を考慮して学習されています．このような音声合成の枠組みは東工大・名工大の研究グループが研究を行っており，その開発ツールも HTS という形で公開されています．

このテキストでは，それらのツールを利用し，音声データの収録からモデル学習までの一連の作業を行います[*1]．さらに，学習によって得られた音響モデルを GalateaTalk へ組み込み，新たな話者モデルとして動作させるところまでを目標にしています．このツールは HTS とともに配布されている日本語用の学習キットをベースに作られており，主に音声認識・音声対話技術講習会で行われてきた演習内容をテキストにしたものです．

1.1 準備

このテキストでは，実際に音を録音したりモデル学習のプログラムを実行するための計算機環境が必要になるため，その準備として，モデル学習用の音声データやプログラムについて説明します．

- 計算機
  Linux がインストールされた一般的な PC で動作します．一般的に，音響モデルの学習には多くの時間がかかるため，CPU が高速なほど有利ですが，メモリの容量さえ十分であれば，CPU の速度はそれほど重要ではありません．必要なメモリ容量は学習データの量や学習方法に依存しますが，例えば 500 文章の音声データを扱う場合では，512MB もあれば十分です．
- データベース
  音声データは，しっかりとした発声で，できるだけ録音状態が良いものを用います．一般的にデータは多ければ多いほど，品質のよい音響モデルを学習する上で有利になりますが，現状では言語情報等のラベルデータの整備には手作業を伴うため時間がかかります．なお，学習用のプログラムと一緒に，サンプルデータベースとして自由に利用できる 503 文章の音声データを公開しており，これを用いてモデル学習を行うことも可能です．また，自分で音声を収録する場合には，このサンプルデータベースが参考になるでしょう．
- ソフトウエア
  データベースの整備や学習に必要なプログラムは，すべてフリーソフトとして公開されているものを利用しています．個別のツールの概要と導入方法については，**??**節を参照してください．
  - HTK ··· HMM の学習ツールキット
  - HTS ··· HMM 音声合成のための HTK のパッチ
  - SPTK ··· 音声信号処理ツールキット
  - Julius ··· 音声認識ソフトウエア
  - get_f0s ··· F_0 抽出プログラム

図 1 は，合成用 HMM を学習する工程の概略です．主なものとしては，音声収録，データの整備，モデル学習があります．音声データの収録は，データ量に応じて時間がかかりますが，その後のデータの整備はまだ自動化が十分でないため，人手による作業が入ります．このように，学習を行うまでの準備には一定の時間と手間が必要になります．

このテキストでは，発話されている内容やアクセント・ポーズの位置などが，想定されている情報と一致しているという仮定で，音声データベースの整備をスキップして，モデル学習を行います．本来であれば，収録音声のチェック（読み誤り，ポーズの位置，アクセントの位置）を行い，発声にあった言語情報や韻律情報の修正作業を行う必要

[*1] 自動化できない部分の作業を簡略化してあります

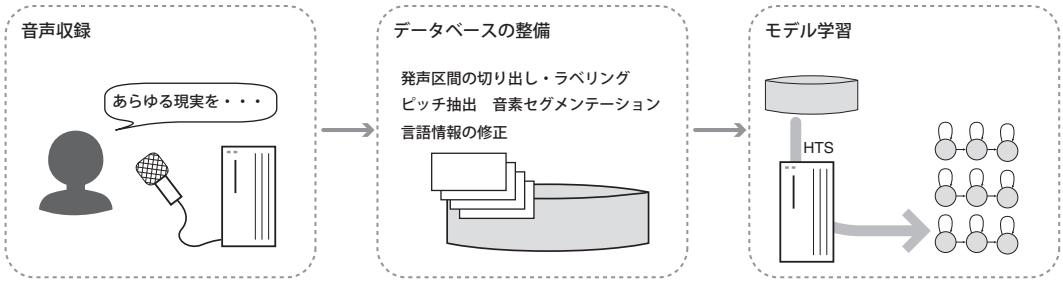

図 1　音声合成モデルの学習過程

がありますが，この作業は自動化が難しく，人手による修正も時間がかかります．

仮に，これらの言語情報・韻律情報に誤りが存在した場合，大きな誤りであれば学習時にエラーとなる可能性があり，軽微なものは学習そのものは完了します．ただ，学習が完了したとしても結果として学習データに存在した誤りが合成時に再現されてしまう場合があり，不自然なところが生じやすくなります．このような学習に悪影響を及ぼすデータベース内の要因は，適切に修正する必要がありますが，このテキストでは取り扱いません．

1.2　データベース

ATR 日本語音声データベースは，音声研究の分野で広く用いられており，その中の音韻バランス文の 503 文章は HMM 音声合成の研究でも標準的なデータベースとして用いられています．このテキストでは，文章セットはこの文章セットと同じものを利用しますが，ATR データベースは商用のため，発声サンプルやラベルデータは名古屋工業大学で収録・整備された音声データベースを利用しています[*2]．ATR データベースを所有している場合は，話者 MHT などの音声データや F_0 データなどと差し替えて使うことも可能です．

1.3　学習プログラム

東工大・名工大による HMM に基づいた音声合成の研究成果をもとに，音声合成用の音響モデルを学習するためのプログラムとして，HTS というツールキットが公開されています[*3]．HTS は，音声認識を目的として HMM のモデル学習や様々な処理を行うためのツールキットである HTK (Hidden Markov models Tool Kit) [*4] のモデル構造や学習プログラムの拡張であり，実質は HTK に対するパッチとして配布されています．HTK のソースにこれらのパッチを適用してコンパイルすることで，HMM 音声合成用のモデル学習が可能になります．なお，HTS のパッチを適用したプログラムでも，通常の HTK としても全く同様に使うことができます．

また，音声信号処理ツールキットである SPTK (Signal Processing Tool Kit) [*5] は，学習データの作成に利用されます．図 2 に，各種プログラムの関係と役割の概要を示します．

1.4　その他のプログラム

このテキストでは，音素の時間境界を利用して初期モデルを学習します．音素境界を人手で付けるのは時間のかかる作業ですが，それを自動化するツールとして Julius [*6] を使用します．Julius の持つ汎用的な音響モデルによって，比較的安定した音素境界を自動的に得ることができます．これはキットに含まれる `segment.pl` によって，音声データとその発話された音素列から，各音素の境界を求めています．

また，HMM 音声合成では，本人の特徴をモデルに取り込むため，スペクトルと韻律の特徴を同時にモデル化して

[*2] 下記の HTS の www ページからダウンロードすることができます
[*3] `http://hts.ics.nitech.ac.jp/`
[*4] `http://www.htk.ac.uk/`
[*5] `http://kt-lab.ics.nitech.ac.jp/~tokuda/SPTK/`
[*6] `http://julius.sourceforge.jp/`

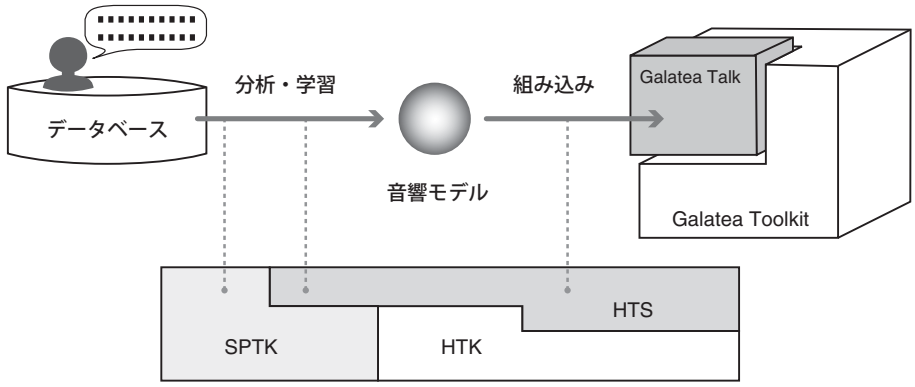

図 2　学習プログラムの構成

./ （パッケージを展開したディレクトリ）

README	簡単な導入
config_ini.pl	初期設定ファイル雛形
mkdata.pl	学習データ作成スクリプト
configure	configureスクリプト
Makefile.in	Makefileの雛形
tools/detect_f0range.sh	F0検出の範囲推定スクリプト
tools/record.pl	録音スクリプト
tools/segment.pl	Juliusによるセグメンテーション用スクリプト
data/Makefile.in	Makefileの雛形
data/scripts	学習データ作成時のスクリプト郡
data/questions	コンテキストクラスタリングの質問ファイル
data/win	デルタパラメータ計算用の回帰係数

図 3　展開された学習キットのファイルツリー

います．そのため，音声データから F_0 抽出を行い，対数基本周波数を学習データとして用います．F_0 自動抽出プログラムは様々なものがありますが，本実習では，get_f0s というコマンドを利用します．これは Snack[*7]や Wavesurfer[*8] で実装されている関数を単独で利用できるようにしたもので，本来は，ESPS 社の get_f0 という F_0 抽出プログラム（すでに販売されていません）がベースになっています．

2　初期設定

VoiceMaker のパッケージを展開し，初期設定を行います．まず，適当な場所に VoiceMaker-1.1.tar.gz を展開します．展開されたディレクトリ内に，学習データやモデル等のファイルを作成します．必要な容量は文章数に依存しますが，500 文章の場合は，中間的なデータも含めると 1GB 程度は必要になります．

図3のようなディレクトリ階層が作成されます．

```
% tar zxf VoiceMaker-1.1.tar.gz
% cd VoiceMaker
```

最初に各種の基本設定となる `config.pl` を編集します．このファイルでは，他の Perl スクリプトで参照されます．作業ディレクトリ，ファイルの命名規則やプログラム類のインストールされた場所修正する主な変数は以下のようなものがあります．

[*7] http://www.speech.kth.se/snack/
[*8] http://www.speech.kth.se/wavesurfer/index.html

- base ··· config.pl ファイルのあるディレクトリを指定
- speaker ··· 話者名を指定（任意の文字列）
- dataset ··· データセットの指定．任意の文字列ですが，現状では文章セットに応じて分けてあります．atr_503（ATR バランス音韻文 503 文章）と istc_b50（ISTC 技術講習会で使われている 50 文章）
- recdir ··· 録音データの保存ディレクトリ
- datadir ··· 学習データの保存ディレクトリ
- sptkbindir ··· SPTK コマンドをインストールしたパス（README に従って導入を行っている場合は変更不要）
- htsbindir ··· HTS コマンドをインストールしたパス（README に従って導入を行っている場合は変更不要）
- julius_dickit ··· Julius の dictation kit をインストールしたディレクトリ
- getf0_base ··· F_0 抽出プログラムをインストールし た場所
- mceporder ··· メルケプストラム分析次数[*9]

3 音声データの収録

まず，学習に必要な音声データの収録を行います．音声の録音では，本来であれば機材や録音環境などの配慮が必要ですが，ここでは比較的入手しやすい機材を例にとって説明を進めます．

また，データセットとして atr_503 を選択した場合は，最大で 503 文章まで音声データを録音しますが，録音されただけのデータで学習を進めることも可能ですので，内容を把握するだけなら 100 文章程度など，適当な文書数で留めて先に進むことも可能です．

3.1 収録の準備

録音は (サウンドカードを備えた) 一般的な PC とマイクロフォンを用います．16kHz サンプリング・モノラル録音が可能な環境であることが前提になります．最近では，マイクアンプを備えた USB オーディオインタフェースが市販されていますので，それらを利用して比較的安価に録音環境を整えることもできるようになってきました．

3.2 音声収録

学習に必要な文章の下限は定めていませんが，50 文程度を想定しています．録音を効率よく行うためのスクリプト (record.pl) を利用します．この Perl スクリプトでは，サンプル音声の再生，録音のサイクルで録音を続ける設計になっています．以下のようにスクリプトを実行すると，まず各文章について読みと漢字が表示され，手本の音声が再生されます．リターンキー，あるいは録音コマンドである “r” を入力すると録音モードに入るので，「please speak」という行が表示されたところで発声を開始します．録音中は “.” が発声の長さに応じて表示され，発話の終端を検出して録音を終了します．録音された音声は，ヘッダなし・16kHz・モノラル・16bit short・Big エンディアン形式の音声データとして，./speech/ の中に保存されます．

```
% ./record.pl
[a01] 未収録
      あらゆるげんじつを、すべてじぶんのほうへねじまげたのだ。
      あらゆる現実を、すべて自分のほうへねじ曲げたのだ。
                    * ここでサンプル音声が再生される
fragment size = 1024 bytes (32msec)
AD-in thread created
<<< please speak >>..... ←ここで発話を開始する
%
```

[*9] 現状の gtalk では可変次元の混在したモデルを扱えません

なお，スクリプト中では，以下のコマンドが使えます．

- l ··· 録音した音声を聞く
- r ··· 録音やり直し
- t ··· お手本を聞き直す
- b ··· 前の文に戻る
- n ··· 次の文へ進む
- m X ··· 文章 X へ移動 (例：m b01)
- q ··· 終了

録音の際は，発話区間の自動検出を行っているため，発声中に極端に長いポーズでは文末の手前で切れてしまう恐れがあります．また，発声する音量や，マイクの録音レベルが低い場合には，切り出しに失敗する場合もあるため，ミキサーの設定を確認して適切な値に調節します．

なお，録音された音声は，play コマンドでオプションを適切に指定するか，sox コマンドなどでヘッダを付与することで再生可能です．

```
% play -x -r 16000 -f s -s w -c 1 〜.ad

% sox -r 16000 -c 1 -s -w -x input.ad output.wav （Microsoft WAV 形式に変換）
% play 〜.wav
```

3.3 録音に関する注意点

録音時には，マイクロフォンを正しく使うことが大切です．自分の声以外の背景雑音ができるだけ混入しないように，マイクロホンのレベル調整を適切に行ってください．ヘッドセットを利用すると，マイクロホンと口の距離を一定に保つことができ，均一な録音レベルのデータが得られます．マイクの指向性に注意することや，ブレスノイズの混入を軽減するためにポップガードやウィンドスクリーンなどの併用が有効です．

キットの録音スクリプトでは，録音ミスを防ぐために，録音音声の確認をその都度繰り返すことから，1 文章の録音でも 30 秒程度を要します．503 文章の録音では，調子よく進めても数時間かかるため，適当な間隔で休憩を取りながら，咽の調子を維持して行うと良いでしょう．また，発話を開始してしばらくの間は，発話のスピードやボリュームが安定しないため,，冒頭の 50 文章程度を練習して発話が安定してから行うか，最初の 50 文章程度を捨てて学習を行うと良いでしょう．

4 学習データの作成

HMM 音声合成では，スペクトルパラメータや基本周波数，およびそれらの動的特徴（Δ パラメータ）を複数のストリームとして扱うため，標準の HTK では定められていないユーザー定義型の構造を持った学習データを SPTK を利用して作成します．HTK では，音声信号から MFCC や LPC ケプストラムなど計算し，学習データを作成するコマンド (HCopy) がありますが，このような独自の構造の学習データを定義することも可能です．

4.1 モデル学習作業の準備

以下のスクリプトを実行して，波形データなどの必要なファイルのコピーと，F_0 抽出，Julius を用いた音素セグメンテーションを実行し，その結果も ./data/ にコピーします．

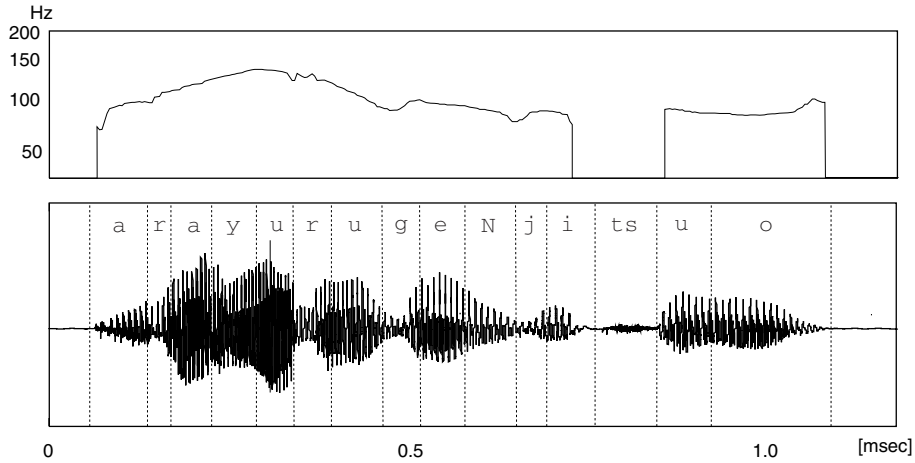

図 4　ピッチ抽出とセグメンテーションの結果（文章: あらゆる現実を...）

```
% ./mkdata.pl
```

これによって，音素セグメンテーションや無音区間の切り出し，基本周波数抽出などを行い，指定した学習データ用のディレクトリに必要なファイルが作成されます．

4.2　音素セグメンテーション

初期モデルの学習では，音声データの各フレームがどの音素であるかという情報をもとに，音素モデルを構築します．そのため，ある音素がどこからどこまで継続しているかというラベル情報が必要になります．音声データに対して，発声された音声のテキストは既知なので，音素列を与えてその時間境界を求める音素セグメンテーションを行います．この処理は大語彙音声認識ソフトウエアの Julius を利用しています．図 4に，その結果の一例を示します．ここで求めた音素境界が大きく異なる場合，正しい音素モデルが学習できなくなる恐れがあるため，修正する必要があります．音素の境界情報のはテキスト形式で書かれているため，直接編集することもできますが，波形データの表示や再生などを備えたラベル修正ツールとして，HTK 付属のコマンド (HSLab) や wavesurfer などがあります．

自動推定された音素セグメンテーションの結果は，ディレクトリ ./data/labels/mono/ 以下に発話データごとにテキスト形式で保存されており，以下のように発話開始，終了，音素名という HTK のラベル形式で記述されています．時間は 100ns 単位になっています．

```
0       2500000 sil
2500000 3000000 a
3000000 3700000 r
3700000 4500000 a
4500000 4800000 y
4800000 5200000 u
5200000 5800000 r
5800000 6400000 u
6400000 6700000 g
6700000 8000000 e
   :
   :
```

wavesurfer では，音声データと共に音素境界データを読み込んで表示し，さらにその境界情報の編集・保存が行えます．wavesurfer を起動して，“Choose Configuration” のフレームで，“HTK transcription” を選択します．波形が表示されている領域の上にある空白部分で右クリックして “Load Transcription..” を選択すると，ラベルデータを読み込

むダイアログが表示されます．そこで，./data/labels/mono/ にあるラベルデータを選択すると，音素境界が表示されます．

4.3 ピッチ抽出

音声データからピッチ抽出を行い，基本周波数のデータを作成します．ピッチ抽出を行うプログラムにはいくつか候補があり，最終的にフレーム単位の F_0 値 (有声区間は Hz，無声区間は 0) がバイナリ形式で保存された形になればどのような方法をとっても問題ありません．

このツールキットでは，ピッチ抽出を行うプログラムを用意しています．以下のように入力とする音声データを引数に与えて実行しています．オプションの-U/-L によって探索範囲の上限と下限を指定できますが，声の性質に合わせて（例えば男性は低め，女性は高めに）適切に調整すると推定精度が向上する可能性があります．

```
% /opt/get_f0s/get_f0s -z -Z 3 -U 400 -L 60 -r 16000 -b raw/istc_balance_b01.raw \
 > f0/istc_balance_b01.f0
```

この処理は，学習データ作成のスクリプト内で，すべての音声ファイルについて実行されます．なお，実際の学習データは，これらの対数をとったものを使用しています．基本周波数の自動抽出機能は完全ではないため，場合によっては倍ピッチなどの抽出誤りが発生します．本来であれば，そのような抽出誤りを修正して学習する必要がありますが，ここではその過程をスキップして進めます[*10]．

推定された F_0 はディレクトリ ./data/f0/ 以下に，発話データごとに基本周波数 (単位は Hz) の列が浮動小数点 (float) のバイナリ形式で保存されています．以下のように SPTK コマンドを使って F_0 の軌跡を表示できます．

```
% PATH=$PATH:〜/opt/SPTK-3.0/bin （SPTK コマンドへのパスを設定）
% gwave -i 1 +s f0/〜.f0 | xgr
```

4.4 学習データ作成

収録した音声データから，HMM の学習データを作成します．これらの学習データは，SPTK の複数のプログラムを組み合わせて作成しますが，それらを自動的に作成するようになっています．以下のようにして，make を実行します[*11]

```
% make data
```

これによって収録した音声データから，メルケプストラム分析を行い，HTK 用学習データの作成，その他，学習に必要となるファイルが作成されます．学習データは，./data/cmp/ ディレクトリ以下に作成されています．収録した音声データの量によって変わりますが，ここの処理は 1 文章あたり数秒程度で終わります．以下は，このスクリプトで実行される学習データ作成の各過程について説明します．

4.5 メルケプストラム分析

音声データに対してメルケプストラム分析を行い，各フレームの係数ベクトルをスペクトルパラメータとして HMM を学習します．この部分は SPTK を用いて計算します．学習データの作成スクリプトでは，各音声ファイルについて以下のようなコマンドを組み合わせた分析が行われています．この処理の結果は，./data/mcep/ 以下に作

[*10] 例えば，Wavesurfer では，ピッチ抽出とフレーム単位での修正が行えます
[*11] mkdata.pl を実行した際に，標準的な設定の Makefile が作成されています

成されます．

- x2x ··· バイナリデータの型変換．ここでは，short 型から float 型への変換を行う．
- frame ··· フレームの切り出し．入力された音声信号から，特定の長さとシフト幅に従ってオーバーラップした区間ごとに切り出して出力する
- window ··· 入力されたフレームごとに窓関数をかける
- mcep ··· 入力されたフレームごとにメルケプストラム分析を行う

4.6 HTS 用学習データの作成

以上で計算した 18 次のメルケプストラム係数，対数基本周波数，およびそれぞれについて，前後のフレームから計算される動的特徴量 Δ，Δ^2 を計算し，フレーム単位で連結します．これに HTK 固有のヘッダ情報（サンプルあたりのバイト数，フレーム数など）を付加することで HMM の学習データを作成します[*12]．

作成した学習データは，HTK の HList コマンドによってその内容を確認することができます．以下のように実行すると，在る音声データについて，メルケプストラム係数とその Δ，Δ^2，対数基本周波数とその Δ，Δ^2 による全 60 次元の特徴ベクトルが，フレーム毎に表示されます[*13]．

```
% HList -C work/?/data/hlist.conf work/?/data/cmp/istc_?_b01.cmp
0:   -0.684   0.923   0.467   0.469   0.201   0.347   0.198   0.020   0.040   0.154
      0.165   0.167   0.265  -0.026   0.001   0.043   0.007   0.179   0.182  -0.144
     -0.073   0.000  -0.084  -0.098   0.002   0.007  -0.131  -0.033   0.067   0.098
      0.067   0.198   0.125   0.095   0.018  -0.050   0.047   0.091   0.174  -0.029
     -0.027   0.057   0.092   0.013   0.021   0.085   0.030  -0.018   0.021   0.007
     -0.090  -0.020  -0.028   0.005   0.056  -0.024  -0.025 -1.0e10 -1.0e10 -1.0e10
1:   -0.480   0.792   0.413   0.498   0.288   0.376   0.246   0.059   0.067   0.186
      0.305   0.248   0.283   0.059   0.041   0.070   0.068   0.179   0.224   0.119
     -0.109  -0.052  -0.014   0.060   0.008   0.036   0.059   0.076   0.079   0.115
      0.082   0.016   0.080   0.040   0.051   0.039  -0.030   0.022  -0.043   0.011
      0.001  -0.022  -0.014  -0.010  -0.006   0.010   0.025   0.023  -0.013   0.001
     -0.001  -0.002   0.000   0.012  -0.011  -0.015  -0.010 -1.0e10 -1.0e10 -1.0e10
2:   -0.447   0.706   0.363   0.440   0.321   0.363   0.270   0.138   0.193   0.312
      0.394   0.331   0.298   0.134   0.081   0.144   0.085   0.120   0.226   0.050
     -0.014  -0.014  -0.005   0.054  -0.022  -0.005   0.062   0.096   0.099  -0.014
     -0.003  -0.064  -0.013   0.007   0.040  -0.011  -0.079  -0.038   0.008   0.037
      0.018   0.026   0.011  -0.005  -0.015  -0.009  -0.015  -0.014  -0.051  -0.043
     -0.039  -0.044  -0.017  -0.017  -0.014  -0.010  -0.020 -1.0e10 -1.0e10 -1.0e10
  :
  :
```

4.7 その他の処理

その他，make を実行することによって，以下のファイルが作成されます．これらは HTS のモデル学習の過程で使用されます．

- 学習データに含まれる音素のリスト lists/mono_?.listg
- 学習データに含まれるコンテキスト依存モデルのリスト lists/full_?.list
- 学習データ (./cmp/*.cmp) のリスト lists/data_istc_b50_?.scp
- マスタラベルファイル label/mono_?.mlf, label/full_?.mlf

[*12] HTK では，ファイルの先頭の 12 バイトに保存されているヘッダ情報によって学習データの種別を識別しています
[*13] 対数基本周波数の-1.0e10 は無声区間をあらわす

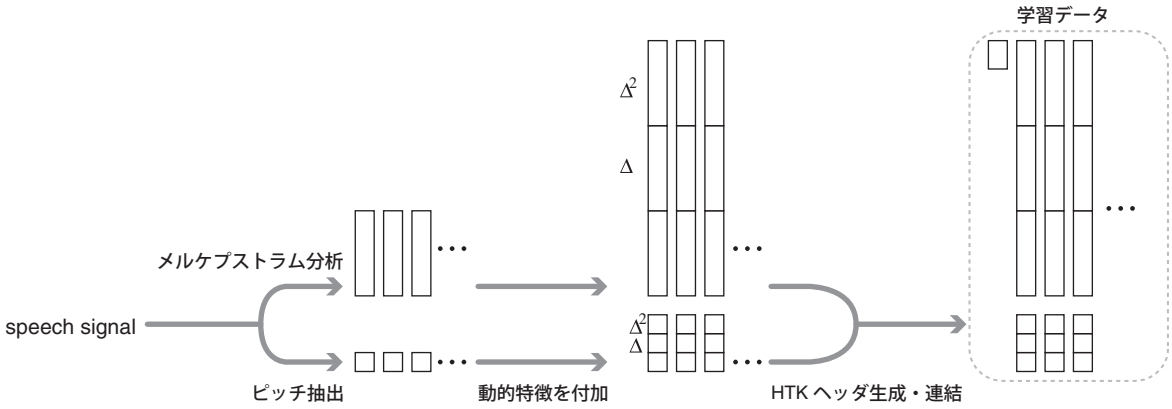

図 5　特徴ベクトルの構成

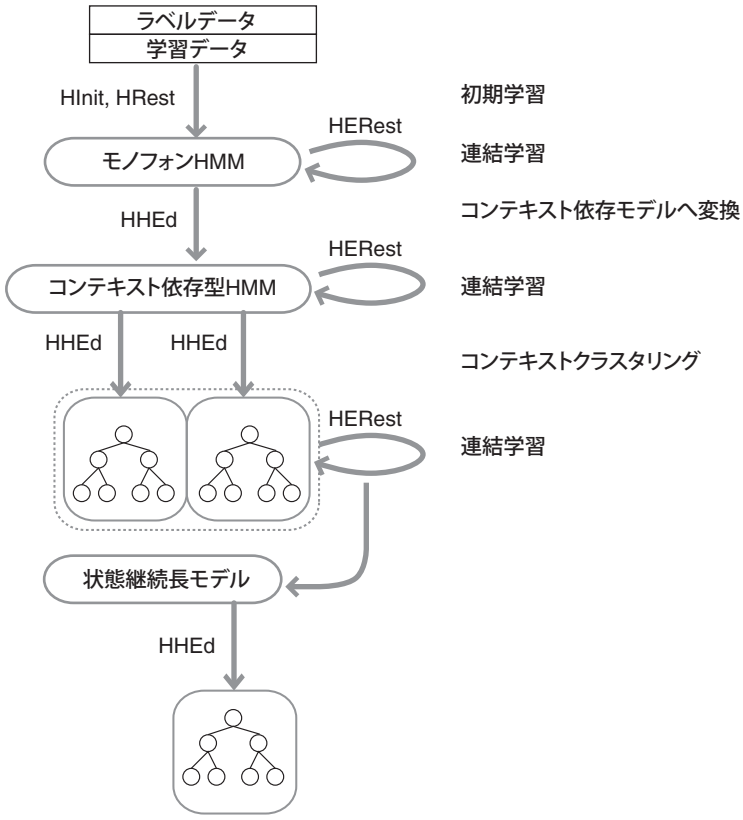

図 6　モデル学習の過程

5　モデル学習

次に，作成された学習データを用いて，合成用 HMM の学習を行います．ここでも，学習用のスクリプトを実行することによって，初期モデルの作成から，逐次的に処理されていきます．図6 にその学習過程の概要を示します．

次から，その学習過程を順に追って，どのようなコマンドが実行されて，どのような処理を行っているかを簡単に説明します．

5.1 準備

まず，モデル学習用のディレクトリに移動します．ここで，make を実行するとモデル学習用のスクリプト scripts/Training_*.pl が作成されます．このスクリプトでは，モデルの初期化から，gtalk 用のモデル作成まで一連の処理が実行されます．ここでの処理の時間は，学習データの量に依存しますが，450 文章で数時間はかかります (Pentium4 3GHz クラスの PC の場合)．また，必要となるメモリ容量も文章に依存しますが，このスクリプトの学習法では，450 文章なら 500MB 程度でも問題ありません．

```
% cd ..
% perl ./scripts/Training.pl scripts/Config.pl
```

なお，Makefile では，以下の修正を行います．

- SPTKBINDIR ··· SPTK コマンドのパス
- HTSBIN ··· HTS コマンドのパス
- DATADIR ··· 学習データのパス
- NAME ··· 話者名 (任意の文字列)
- DATASET ··· データセット (任意の文字列)
- MCEPORDER ··· 分析時の次数に合わせる（この例では 18）

5.2 モノフォン HMM の学習

まず，収録した音声データから，各音素の初期モデルを作成します．HTK の HInit コマンドによって，音素セグメンテーションの結果をもとに，学習データ中のある音素に該当するフレームから，HMM の各状態のモデルパラメータを計算します．続いて，HRest コマンドによって，音素モデルのパラメータを最推定します．このモデルから，HERest コマンドによって境界なし学習を繰り返すことにより，各モデルのパラメータを更新します．

音声データの時間境界が分からない場合には，全ての学習データからモデルパラメータの平均を計算し，繰り返し学習によって収束させていく手法によっても初期モデルを得ることができます（フラットスタート法)．

5.3 コンテキスト依存型 HMM への変換と最推定

次に，モノフォン HMM から，コンテキスト依存型 HMM へ変換します．

HMM 音声合成用のモデルは，基本的には音素レベルによって分類されていますが，前後環境や，言語的な構造に依存した精度の高いモデル化を行うために，「コンテキスト」というさまざまな変動要因を考えることで，より詳細にモデルを分類します．ここで取り扱っているコンテキストの要因としては以下のようなものがあります．

- 文の長さ
- 当該呼気段落の位置
- { 先行，当該，後続 } 呼気段落の長さ[*14]
- 当該アクセント句の位置，前後のポーズの有無
- { 先行，当該，後続 } アクセント句の長さ，アクセント型
- { 先行，当該，後続 } の品詞，活用形，活用型
- 当該音素のアクセント句内でのモーラ位置
- { 先行，当該，後続 } 音素

[*14] ここでは，句読点で区切られる句とする

表 1　コンテキストの分類

p_L	先行音素
p_C	当該音素
p_R	後続音素
a_{C1}	アクセント句内モーラ位置 (単位: モーラ)
a_{C2}	アクセント型とモーラ位置との差 (単位: モーラ)
b_{L1}	先行品詞 ID
b_{L2}	先行品詞の活用形 ID
b_{L3}	先行品詞の活用型 ID
b_{C1}	当該品詞 ID
b_{C2}	当該品詞の活用形 ID
b_{C3}	当該品詞の活用型 ID
b_{R1}	後続品詞 ID
b_{R2}	後続品詞の活用形 ID
b_{R3}	後続品詞の活用型 ID
c_{L1}	先行アクセント句の長さ (単位: モーラ)
c_{L2}	先行アクセント句のアクセント型
c_{L3}	先行アクセント句と当該アクセント句の接続強度
c_{L4}	先行アクセント句と当該アクセント句間のポーズの有無
c_{C1}	当該アクセント句の長さ (単位: モーラ)
c_{C2}	当該アクセント句のアクセント型
c_{C3}	先行アクセント句と後続アクセント句の接続強度
c_{C4}	当該呼気段落でのアクセント句の位置
c_{C5}	疑問文かそうでないか
c_{R1}	後続アクセント句の長さ (単位: モーラ)
c_{R2}	後続アクセント句のアクセント型
c_{R3}	後続アクセント句と当該アクセント句の接続強度
c_{R4}	後続アクセント句と当該アクセント句間のポーズの有無
d_{L1}	先行呼気段落の長さ (単位: モーラ)
d_{C1}	当該呼気段落の長さ (単位: モーラ)
d_{C2}	文中での当該呼気段落の位置
d_{R1}	後続呼気段落の長さ (単位: モーラ)
e	文の長さ (単位: モーラ)

ある音素についてみたとき，これらのコンテキスト要因の違いに応じて区別することで，様々な状況に適したモデルを学習することができます．モデルの表記としては，a のような音素表記から，表 1に従って，以下のようなフォーマットで表現されます．実際のデータでは，`./labels/full/` にあるラベルファイルを見ると，音素ごとに以下のようなラベル列が並んでいることが分かります．

$$p_L\text{-}p_C\text{+}p_R\texttt{/A:}a_{C1}\_a_{C2}\texttt{/B:}b_{L1}\_b_{L2}\_b_{L3}\text{-}b_{C1}\_b_{C2}\_b_{C3}\text{+}b_{R1}\_b_{R2}\_b_{R3}$$
$$\texttt{/C:}c_{L1}\_c_{L2}\_c_{L3}\_c_{L4}\text{-}c_{C1}\_c_{C2}\_c_{C3}\_c_{C4}\_c_{C5}\text{+}c_{R1}\_c_{R2}\_c_{R3}\_c_{R4}\texttt{/D:}d_{L1}\text{-}d_{C1}\_d_{C2}\text{+}d_{R1}\texttt{/E:}e^{*15}$$

このような音素表記のモデルから，コンテキスト依存モデルへの変換は，`HHEd` コマンドによって行います．ただし，ここでの処理では，初期学習によって得られたモノフォン HMM から，新たに作りたいコンテキスト依存型を複

製するだけなので，再び HERest を実行してモデルパラメータを最推定する必要があります．

5.4 コンテキストクラスタリング

先に説明したコンテキストのすべての出現を考慮した組み合わせは膨大な数になり，今回の実習の場合に限らず，限られた音声データではそれらのすべてをカバーすることは不可能です．様々なコンテキストの組み合わせに対して，なんらかのモデルを用意しないことには，音声を合成するためのパラメータを得ることはできませんが，他のモデルの中から，意図したコンテキストに近いモデルで代用することを考えます．ここでは，二分木を用いたクラスタリングを行うことによって，類似したモデル間のパラメータを共有させて，少ないデータ量でも様々なコンテキストの組み合わせに対応できるモデルを構築しています．

コンテキストクラスタリングは，HTK の HHEd コマンドによって行います．このプログラムでは，コマンドベースの命令によってその動作をテキストファイルに記述して実行させるため，そのスクリプトファイルにコンテキストを分類するための質問を複数並べて記述します．

学習キットの中では，./questions/question.hed というファイルがその雛形に該当し，その質問の一部を以下に示します．これらの質問は，音素の発声方式の分類に関する質問を表しており，直後にある音素が，有声摩擦音，無声摩擦音，有声破裂音，無声破裂音のそれぞれに属するか，について分類するためのものです．同様にして，それらの当該音素や直前の音素に関しても質問が用意されています．括弧の中は，コンテキストのラベルにマッチする正規表現で記述されており，他の質問も同様にして，ある質問に対してマッチするラベルの組がすべてのコンテキスト要因について記述されています．

```
        :
QS 'R_yusei_masatuon' { *+j/A*,*+z/A* }
QS 'R_musei_masatuon' { *+fy/A*,*+hy/A*,*+sh/A*,*+s/A*,*+f/A*,*+h/A* }
QS 'R_yusei_haretuon' { *+dy/A*,*+gy/A*,*+vy/A*,*+by/A*,*+g/A*,*+d/A*,*+v/A*,*+b/A* }
QS 'R_musei_haretuon' { *+ky/A*,*+py/A*,*+k/A*,*+t/A*,*+p/A* }
        :
```

コンテキストクラスタリングは，多数の質問を適用しながらモデルの分割を繰り返す処理を行うため，計算時間が長くなります．学習データが多い場合では，それだけ対象となるモデルパラメータの空間は大きくなるため，より多くの時間が必要となります．

クラスタリングの結果，どのような木構造が構築されているかを確認することができます（図7 の例を参照）．大量の学習データを使用すると木が大きくなり，全体を見渡すのは困難ですが[*16]，本実習ではデータ量が少ないため，スペクトル，基本周波数，状態継続長の各モデルにおいて，どのような質問が適用されて木が構築されているか，その全体像を見て取ることができます．もちろん音質は充分ではないかもしれませんが，これだけのパラメータ数で音声が合成可能である，という点も注目できます．

6 音声合成

HTS には合成用のプログラムが付属していますが，これはラベルデータからパラメータ生成を行うもので，いわゆる TTS (Text-to-speech) システムという完結した音声合成を行うプログラムではありません．

自由なテキストから音声を合成するためには，言語情報などを含んだラベルデータを作成するためのテキスト解析器と，音声パラメータから波形を生成する処理が同時に必要となります．GalateaTalk では，そのようなテキスト解析と音声合成エンジンが一つにまとまっているため，ここで作成した音響モデルを組み込むことで，オリジナルの話者モデルを持った音声合成システムを作ることができます．

[*16] 話者にも依存しますが，例えば 450 文章の場合はリーフ数が 1000 を超える大きさの木ができます

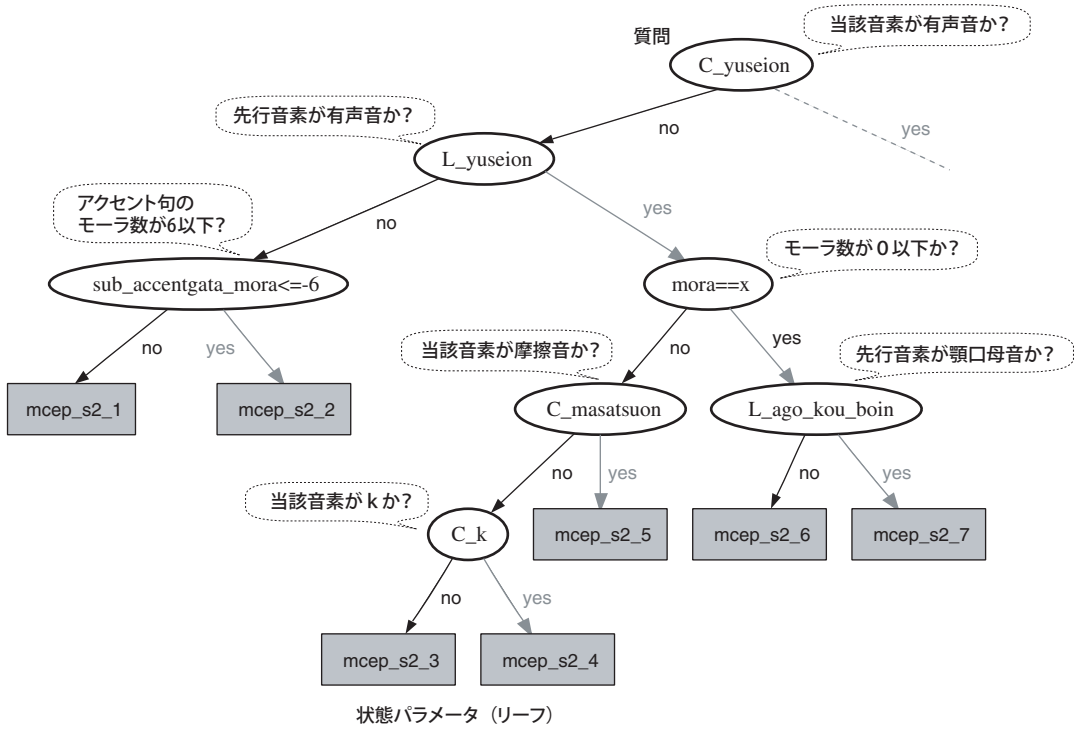

図７　モデル学習によって作成された木構造の一部

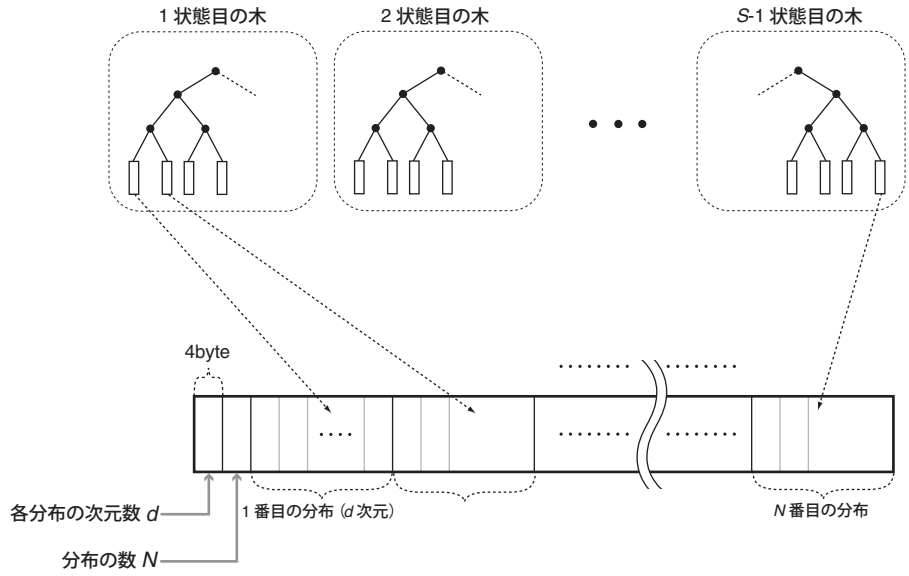

図８　GalateaTalk 用音響モデルの構造

6.1　GalateaTalk 用音響モデルへの変換

GalateaTalk では，クラスタリングされたメルケプストラム，基本周波数，状態継続長のそれぞれのモデルを，別々の木構造の形で読み込みますが，ファイルの形式は HTK で用いられているマクロモデルファイルとは異なります．GalateaTalk 用のモデル形式は，非常にシンプルなもので，必要な情報だけを単純に並べたものになっています．HTS によって新たに追加された機能として `HHEd` を用いて変換することができます．

学習スクリプトによって，それらの変換の処理が実行されており，`./voices/` 以下のディレクトリにある〜`.pdf`，

〜.inf というファイルが，メルケプストラム，基本周波数，状態継続長のモデルパラメータとそのそれらの木構造を表したものに相当します．モデルパラメータは，木のリーフにある分布（float 値の多次元ベクトル）がバイナリ形式で保存され，木構造のファイルには，質問木を構成するための情報がテキスト形式で記述されています（図8 参照）．変換には，HHEd で./edfiles/?/?/cmp/convert_?_qst001_〜.hed というファイルを用いて行っています．

6.2 GalateaTalk への組み込み

変換された音響モデルは./voices/?/?/qst001/ の下に保存されている 6 つのファイルからなります．これらのファイルを，GalateaTalk の設定ファイルへ追加することで，新たな話者モデルを組み込むことができます．

gtalk の設定ファイル ssm.conf には既存の話者と同様に，以下のようにして追加します．〜-SPEAKER-ID には適当な文字でも構いません．TREE-FILE はモデルの木構造データ，〜-MODEL-FILE は木のリーフにあたるモデルパラメータに相当し，それぞれに対応したファイルが指定されています．

```
SPEAKER-ID: test
GENDER: male
DUR-TREE-FILE:   /〜/tree-dur.inf
PIT-TREE-FILE:   /〜/tree-lf0.inf
MCEP-TREE-FILE:  /〜/tree-mcep.inf
DUR-MODEL-FILE:  /〜/duration.pdf
PIT-MODEL-FILE:  /〜/lf0.pdf
MCEP-MODEL-FILE: /〜/mcep.pdf
```

この設定ファイルを読み込んで gtalk を起動すると，新たな話者モデルが利用可能になります．gtalk のプロンプトから，「set Speaker = test」とコマンドを実行することで，追加した話者モデルを選択することができます．

7 まとめ

本テキストでは，GalateaTalk に追加する話者モデルを構築するため，音声収録から学習データの作成，HMM の学習，GalateaTalk への組み込みという一連の作業を行いました．あくまで「簡易版」ということで，十分な音質が得られるわけではありませんが，音質を良くするための余地はまだ大きく残されています．録音環境，実際の発声音声と，学習に用いた言語情報ラベルとの不一致，自動抽出した音素セグメンテーションや基本周波数の精度の問題も考えられます．それらを正しく修正した上で学習を行うことで，品質の改善が見込まれます．

8 付録: ツール類の導入

config.pl に記述されている標準の設定にしたがって，/opt/ 以下にコマンド類をインストールする例をまとめておきます．インストール先のディレクトリの違いは，設定ファイル内で吸収できるため，/opt 以外の場所でも問題はありません．これらの動作確認は，Vine Linux 4.1 で行いましたが，固有の環境に依存する問題は少ないと思われます．

8.1 SPTK

このキットでは，学習データの作成や，そのほかデータの加工等で，SPTK と呼ばれる音声信号処理関連のプログラム郡を利用しています．SPTK は下記のページからダウンロードできます．

ソースの入手

SPTK の配布サイト http://kt-lab.ics.nitech.ac.jp/~tokuda/SPTK/ から SPTK-3.0.tar.gz をダウンロードします．必要ならリファレンスマニュアル (PDF 形式) もダウンロードしておくとよいでしょう．

コンパイル

一例として，/opt/ 以下にインストールする手順は以下のようになります．

```
% cd /opt
% tar zxf SPTK-3.0.tar.gz （ファイルの場所は適宜読みかえる）
% cd SPTK-3.0/src
% make PREFIX=/opt/SPTK-3.0 （インストールパスを/usr/local/SPTKにしない場合）
% make install
```

以上の作業で，/opt/SPTK-3.0/bin/ 以下に SPTK コマンドがインストールされます．

8.2 get_f0s

収録された波形データから，基本周波数を推定するプログラムです．基本周波数推定にはさまざまな手法があり，同様の形式のものが得られれば，どのような方法でも構いません．ここでは，ESPS の get_f0 に相当するコマンドを利用します．

パッケージの入手

get_f0s は，http://galateatalk.sourceforge.jpにて配布されています．

コンパイル方法

```
% cd /opt
% tar zxf get_f0s-0.1.tar.gz
% cd get_f0s-0.1
% make
```

8.3 Julius

音素セグメンテーションを自動的に得るために，大語彙連続音声認識ソフトウエアの Julius を利用しています．Linux の場合は，実行形式を含めたキットが提供されているため，コンパイル等の必要は無い．

パッケージの入手

Julius の配布サイト http://julius.sourceforge.jp/ から，dictation-kit-v3.1.tar.gz をダウンロードする．このパッケージにエンジンと音響モデルが含まれている．ソースが必要な場合は別途ダウンロードする．

パッケージの展開

```
% cd /opt
% tar zxf dictation-kit-v3.1.tar.gz （ファイルの場所は適宜読みかえる）
```

8.4 HTS

HTS は HTK(HMM Tool Kit) を拡張する差分のパッチとして提供されています．HTK そのものはケンブリッジから無償提供されてい，入手するには HTK の配布サイトよりユーザ登録を行い，ダウンロードのためのアカウントを取得する必要がある．HTS は HTK の個別のバージョンごとにリリースされている．この時点では HTK-3.4 に対応

した HTS-2.0 を利用する．/opt/htk/ 以下にインストールする手順です．

準備

HTK の配布サイトhttp://htk.eng.cam.ac.uk/ にて，ユーザー登録を行い，HTK-3.4.tar.gz，HDecode.tar.gz をそれぞれダウンロードする．また，HTS のパッチは

http://hts.ics.nitech.ac.jp/release/HTS-2.0_for_HTK-3.4.tar.gz からダウンロードできる．

コンパイル

HTK，HTS のパッケージをそれぞれ展開する．

```
% cd /opt
% tar zxf HTK-3.4.tar.gz （ファイルの場所は適宜読みかえる）
% tar zxf HDecode-3.4.tar.gz （ファイルの場所は適宜読みかえる）
% cd htk
% tar zxf HTS-2.0_for_HTK-3.4.tar.gz （ファイルの場所は適宜読みかえる）
```

HTS のパッケージを展開して得られる HTS-2.0_for_HTK-3.4.patch が HTK-3.4 に対するパッチになる．

```
% patch -p0 < HTS/HTS-2.0_for_HTK-3.4.patch
```

あとは通常の HTK のコンパイルを行うと HTS 対応の HTK コマンドが作成される．まず，コマンドをインストールする先のディレクトリを作成しておき，ライブラリのディレクトリに移動して，以下のように make コマンドを実行します．

```
% ./configure --prefix=/opt/htk
% make
% make install
```

以上の作業で，/opt/bin/ に HTS のコマンドがインストールされます[*17]．

[*17] これらのツールは，HTK としても利用可能です