Internal Use Only

非公開

TR-HIS-0001

0013

音声データベースの自動音素セグメンタの評価

早川　徹 (ATR-HIS/早稲田大学), 加藤　宏明

2001.11.30

国際電気通信基礎技術研究所　人間情報科学研究所

〒619-0288　京都府相楽郡精華町光台二丁目2番地2

Tel: 0774-95-2641 Fax: 0774-95-2647

Advanced Telecommunications Research Institute International (ATR)
Human Information Science Laboratories

2-2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

Tel : +81-774-95-2641 Fax : +81-774-95-2647

音声データベースの自動音素セグメンタの評価

早川　徹　　加藤　宏明

（早稲田大学　情報科学研究科）

平成 13 年　11 月　12 日

abstract

本実習では、自動音素セグメンタの精度評価を目的とし、ATRSPREC を用いた自動セグメンテーション法の獲得、ATR 連続音声データベース B セットを用いた実験、結果の分析を行った。B セットに付与された視察によるラベリングの時間データを正解として、誤差を評価する。実験を通して、自動セグメンテーションを行う際の問題点を洗い出し、そのいくつかには対処した。また、ATRSPREC を用いた音響モデルの話者適応法も習得し、実験に取り入れた。結果分析では、話者適応が自動セグメンテーション精度向上に有効であること、音素境界部が無声区間側に張り出す傾向にあること、同系統音素連続の境界部では、境界誤差が大きくなる傾向があることなどがわかった。以上の傾向をキャンセルするような処理を施すことにより、自動セグメンテーションの精度向上が望まれる。

目次

付録目次

1.　はじめに

話し言葉、特に韻律の研究とその応用技術には、対象データの音素列時間構造の情報が重要である。この情報は熟練した作業者による視察（ハンドラベリング）に頼ることで、作業者によらず安定した時間情報ラベルが獲得できることが分かっており[1]、これまで多方面で利用されてきた。しかしながら熟練作業者の人手に頼ることによる時間と費用の両面におけるコスト高が大きな問題であった。

近年、このコストの問題を解消する目的で、音声認識エンジンを用い、この作業を自動化する方法が提案されており、実際に自動セグメンテーション機能を持った認識エンジンもいくつか存在している[2、3、4、10、11]。ハンドラベリング精度には及ばないにしろ、この機能によって得られる音素列時間構造情報がその精度にある程度近いものであれば、自動セグメンテーションがハンドラベリングの代わりとして使用できる場合も出てくるだろう。しかし、この技術に関する検討は十分なされていないのが現状であり、自動セグメンテーションの結果の特性、性能向上の余地など、明らかになっていない点は多い。

そこで本研究では、自動セグメンテーションの現状把握を行う。さらに自動セグメンテーションの結果に対する多角的な分析を通じ、自動セグメンテーションの能力向上への指針獲得をねらう。

本報告は２００１年９月１日から９月３０日にかけて、ATR 先端情報科学研究部において行われた実習内容及び結果の要約である。

2.　自動セグメンテーションの方法

本研究では、音声認識エンジンとしてATRSPREC[5、6]、分析資料としてATR「研究用日本語連続音声データベース」[1、7、8]（男性６名、女性４名、各話者５０３文、合計５０３０文の朗読文）を用いて自動セグメンテーションを行った。ATRSPRECを用いた自動セグメンテーションには、音声データのみを認識エンジンに与え、認識された音素列を元にセグメンテーションを行う方法、音声データと共に正解音素列を与え、正解音素列に基づいたセグメンテーションを行う方法がある。前者の場合は、セグメンテーション結果のエラーの原因が音声認識部、セグメンテーション部のどちらにも存在し得る。一方後者では、正解音素列をあらかじめ与えるため、セグメンテーション結果のエラーがセグメンテーション部のみに起因することになる。本研究の目的はセグメンテーション部の評価であるため、後者の方法を取ることにする。

図１にATRSPRECを用いた自動音素セグメンテーションの一連の流れを示す。また，付録１（p. 22）に各処理にて用いるコマンド，スクリプト，入出力ファイルの一覧を添付する。以下，これらの処理について順を追って説明する。

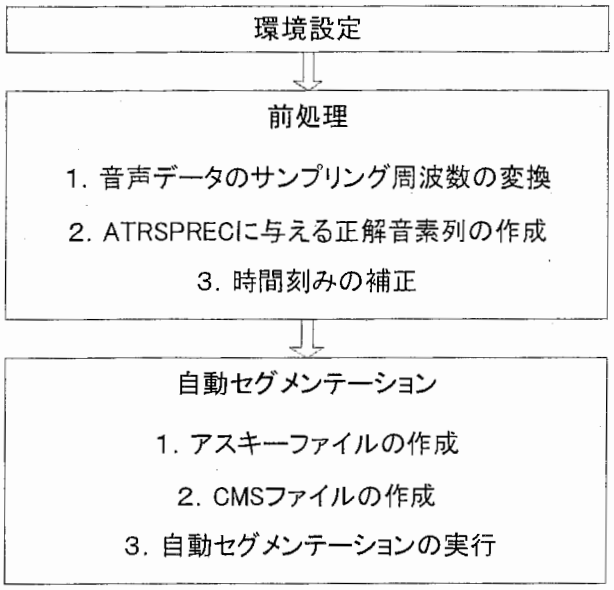

図 1　自動音素セグメンテーションの一連の流れ

2．1　環境設定

環境変数、パスの設定を行う。ATRSPRECで提供される設定ファイルを使用者の環境に合わせて変更し，sourceコマンドで読み込むことで必要な設定が行われる。付録2（p.26）に設定ファイルの例を添付する。なお，sourceコマンド実行に先立ち，環境変数"LD_LIBRARY_PATH"の設定（変数の値はNULLでよい）が必要である。リスト1に環境設定の実施例を示す。

リスト 1　環境設定の実施例

```
setenv LD_LIBRARY_PATH ""
source SPREC.setup
```

2．2　前処理

分析対象データと自動セグメンタとの間の諸仕様の不整合を吸収するための前処理を行う。まず，自動セグメンテーションに用いる認識エンジンの音響モデルが想定する音響分析の諸条件が分析対象の音声データのそれと異なる場合，事前に両者を整合させる必要がある。また，正解音素列を与える場合は，音響モデルのラベリング規則と正解ラベルの規則間でのラベルの翻訳が必要である。さらに，ラベルの時間情報を比較する場合は，比較対間での時間分解能の相違に注意する必要がある。今回用いたATRSPRECの音響モデルを

ATR音声データベースBセットで評価するにあたって，以下の３つの処理を考慮する必要があった。

２．２．１　音声データのサンプリング周波数の変換

今回用いた ATRSPREC の音響モデルが対応する音声データのサンプリング周波数は、16kHz であった。今回用いたデータベースの音声データは 20kHz でサンプリングされていたため、ダウンサンプリングを行う必要があった。ダウンサンプリングには ESPS というツールを用いた。ESPS によるダウンサンプリングは、データの劣化が比較的少ない。このツールを用い複数の音声データにダウンサンプリングを行うのに使用したスクリプトを付録３（p.29）として添付する。

２．２．２　ATRSPREC に与える正解音素列の作成

ATRSPREC に与える正解音素列を TRS （TRanScription）ファイルと呼ぶ。TRS ファイルは発話開始、終了時刻と音素列からなるファイルで、独特のファイルフォーマットを持っている（TRS フォーマット）。よって、正解音素列を ATRSPREC に与えるには、対象音声データのテキスト情報を TRS フォーマットの音素列に翻訳する必要がある。今回用いたデータベースにはハンドラベリング結果が付随しており、その音素列を TRS フォーマットに翻訳し正解音素列を得た。ハンドラベリングと TRS フォーマットで表記法が異なるものの全対応表を付録４（p.30）、この翻訳のために使用したスクリプトを付録５（p.32）として添付する。また、出力される TRS ファイルの例をリスト２に示す。

リスト 2　TRS ファイルの例

```
255.0 a,r,a,j,u,r,u,g,e,ng,zh,i,ts,u,o,-,s,u,b,e,t,e,zh,i,b,u,ng,n,
o,h,o,u,e,n,e,zhi,m,a,g,e,t,a,n,o,d,a 3985.0
```

２．２．３　時間刻みの補正

本研究では、自動セグメンテーション結果とハンドラベリング結果との比較を行う。データベース付随のハンドラベリングの時間刻みは 2.5 ms であるが、自動セグメンテーションの時間刻みは音響モデルの学習時フレームシフトの時間幅に等しい値になる。今回自動セグメンテーションに用いた音響モデルは 10 ms 刻みで学習されたものであったため、厳密な比較を行うためには 10 ms と 2.5 ms の時間刻みの相違を吸収する工夫が必要であった。2.5 ms 刻みで学習した音響モデルを作成し自動セグメンテーションに利用する方法、自動セグメンテーションを行うまでの中間データ（デルタケプストラム値）に補正をかけ、強制的に 2.5 ms 刻みでのセグメンテーション結果を出力する方法等が考えられるが、研修期間中には達成できなかった。そのため、本研究での自動セグメンテーションの結果は 10 ms

刻みである。

２．３　自動セグメンテーション

次の手順を踏み、自動セグメンテーションを実行する。

1　アスキーファイルの作成
2　CMS（Cepstrum Mean Subtruction）ファイルの作成
3　自動セグメンテーションの実行

以下、各手順について詳しく述べる。

２．３．１　アスキーファイルの作成

アスキーファイルとは、音声データを収録したときの情報を書いたものであり、次節に述べる CMS（Cepstrum Mean Subtruction）ファイル作成時に必要になる。今回分析対象とした音声データベースにはアスキーファイルは含まれていなかったため、ダミーのアスキーファイルを用いることで対処した。ダミーのアスキーファイルを作成するために使用したスクリプトを付録６（p.37）として添付する。引数として与えるのは、分析対象の音声データが格納されているディレクトリ名と出力されるアスキーファイルのファイル名（拡張子除く）の２つだけである。また、出力されるアスキーファイルの例をリスト３に示す

リスト ３　アスキーファイルの例

```
ATR   JTEXT          X     JMOR      WAV      TRS          X   MHNIL
2001-9-4  503       AD         CUSTOMER
```

２．３．２　CMS ファイルの作成

CMS（Cepstrum Mean Subtruction）ファイルとは、自動セグメンテーションを行う対象となる音声ファイルから抽出された、特徴ベクトルの平均値情報をもったファイルであり、入力波形をケプストラムに変換する際の、データの正規化に用いられる。このファイルは、次節に述べる自動セグメンテーションの実行時に必要なファイルである。CMS ファイル作成には ATRSPREC に含まれるパイソンスクリプト atrCM.py を使用する。CMS ファイルの例をリスト４に示す。

リスト４　CMS ファイルの例

```
MeanCep   2530375   12   -2.924169e+00   -3.433570e-01   3.399912e-01
-4.380901e-01      -4.692597e-01      4.814745e-02      -5.789096e-01
-1.748646e-01      -3.484284e-01      4.346725e-03      -1.776003e-01
-1.453464e-01
MeanLogPow 2530375 1 7.633901e+00
```

このスクリプトと共に atrCM.py 用のコンフィギュアファイルが必要になる。コンフィギュアファイルの例を、付録７（p.39）として添付する。I/Ocontrol は入出力ファイルについてのコンフィギュレーション、ATRwave2cep、ATRwavecut、ATRepd はそれぞれ atrCM.py に含まれる同名のモジュールに対応したコンフィギュレーションである。ATRwave2cep は入力波形をケプストラムに変換するための、また ATRwavecut、ATRepd は音声ファイルから音声区間を取り出すためのモジュールである。各コンフィギュレーションの設定は CMS ファイル作成環境によって変える必要があるが、それぞれの意味する内容、変更方法は ATRSPREC リファレンスマニュアル[6]に記載されている（リファレンスマニュアル 15.1 I/Ocontrol、15.3　ATRwave2cep、15.10　ATRwavecut、15.2　ATRepd）。

なお、atrCM.py を用い CMS ファイルを作成するために使用したシェルスクリプトを、付録８（p.41）として添付する。

２．３．３　自動セグメンテーションの実行

自動セグメンテーションの実行については，さらに以下の３つのステップに分けて説明する。

1　実行ファイルの作成
2　コンフィギュアファイルの作成
3　自動セグメンテーションの実行

２．３．３．１　実行ファイルの作成

ATRSPREC は音声処理に関する様々なモジュールから成り、目的に応じて、必要なモジュールを選択、コンパイルし、ひとつのツールとして使用することが可能である。今回の方法で自動セグメンテーションを行う際に必要なモジュールは、入力音声ファイルの制御を行う ATRinput、入力波形をケプストラムに変換する ATRwave2cep、ケプストラムを特徴パラメーターに変換する ATRcep2para、音素の viterbi 整列を行う ATRviterbi、音素列を取り込む ATRsendans、結果を出力する ATRresult であり、以上をコンパイルしてひとつの実行ファイルとする。

コンパイルの際、それぞれのモジュールファイル（.mod）を読み込むが、ATRinput のモジュールファイル ATRinput.mod は、統合するモジュールの種類に応じて設定を変える必要がある。リスト５は今回モジュールを統合する際に用いた、ATRinput.mod ファイルの内容である。

リスト ５　ATRinput.mod ファイル

```
Header=ATRinput.h
Receiver=ATRD_MARKED_WAVE
InitializeFunction=ATRinput_init
ExecuteFunction=ATRinput_exec
TerminateFunction=ATRinput_term
```

上記のうち、統合するモジュールに応じて変更が必要になるのは Receiver の部分である。変更が必要な場合、方法については ATRSPREC のマニュアルに記載されている（リファレンスマニュアル 15.15　ATRinput)。このファイルは、コンパイルを行うフォルダに置くことで自動的に読み込まれる。

リスト６はコンパイル、実行ファイルの作成コマンドである。

リスト ６　自動セグメンテーション実行ファイルの作成コマンド

```
make -f $ATRSPREC/src/ATRMAINGEN/makefile.template¥
MODULES=ATRinput,ATRwave2cep,ATRcep2para,ATRviterbi,ATRsendans,¥
          ATRresult¥
NAME=ATRexec
```

$ATRSPREC は、２．１節の環境設定の際に、設定ファイルの環境変数で指定した ATRSPREC のホームディレクトリである。このコマンドにより、"NAME="で指定した名前の実行ファイル(上の例では ATRexec)ができる。ここからはこの実行ファイル名を仮に、ATRexec と呼ぶ。

２．３．３．２　コンフィギュアファイルの作成

前節で作成したファイルを実行するには、さらに、統合されたそれぞれのモジュールに応じたコンフィギュアファイルが必要になる。コンフィギュアファイルの例を、付録９（p.43）として添付する。

I/Ocontrol は入出力ファイルについてのコンフィギュレーション、ATRwave2cep、

ATRcep2para、ATRviterbi、ATRresult はそれぞれ同名のモジュールに対応したコンフィギュレーションである。モジュール ATRsendans に対応するコンフィギュレーションは、ATRexec の実行時に引数として渡す。コンフィギュレーションの中身は自動セグメンテーションを行う環境に応じて変更する必要がある。各設定の意味する内容、変更方法は ATRSPREC リファレンスマニュアル[6]に記載されている（リファレンスマニュアル 15.1 I/Ocontrol、15.3 ATRwave2cep、15.4 ATRcep2para、15.7 ATRviterbi、15.6 ATRresult、15.8 ATRsendans）。なお、前節で述べた CMS ファイルはモジュール ATRwave2cep で用いることになり、コンフィギュレーション中の ATRwave2cep:MeanInFile で、この CMS ファイルの場所を指定する。

２．３．３．３　自動セグメンテーションの実行

以上の実行ファイル、コンフィギュアファイルを用い、自動セグメンテーションを行う。実行コマンドをリスト７に示す。

リスト ７　自動セグメンテーションの実行コマンド

```
./ATRexec  -config=コンフィギュアファイル¥
-ATRinput:file_list=wave ファイルリスト¥
-ATRsendans:answer=TRS ファイル  >  結果ファイル
```

config には、前節で作成したコンフィギュアファイル名を指定する。ATRinput:file_list には、自動セグメンテーションを行う wave ファイルの絶対パスを１行に１つ格納した wave ファイルリストのファイル名を指定する。ATRsendans:answer には、個々の wave ファイルに対応した TRS ファイルを順番に cat でつないで、一つの長い TRS ファイルを作成し、そのファイル名を指定する。">"のあとには結果ファイル名を指定する。ATRinput:file_list で指定する wave ファイルリストの例をリスト８、ATRsendans:answer で指定する TRS ファイルの例をリスト９に示す。

リスト ８　wave ファイルリストの例

```
/prj/slap2/project/ATRSpeechDB/B.16k/MHT/A/MHTSDA01.AD.16k
/prj/slap2/project/ATRSpeechDB/B.16k/MHT/A/MHTSDA02.AD.16k
/prj/slap2/project/ATRSpeechDB/B.16k/MHT/A/MHTSDA03.AD.16k
```

リスト 9　catでつないだTRSファイルの例

```
255.0 a,r,a,j,u,r,u,g,e,ng,zh,i,ts,u,o,-,s,u,b,e,t,e,zh,i,b,u,ng,n,
o,h,o,u,e,n,e,zhi,m,a,g,e,t,a,n,o,d,a 3985.0
255.0 i,q,sh,j,u,u,k,a,ng,b,a,k,a,r,i,-,n,j,u,u,j,o,o,k,u,o,sh,j,u,
z,a,i,sh,i,t,a 3100.0
250.0 t,e,r,e,b,i,g,e,e,m,u,j,a,p,a,s,o,k,o,ng,d,e,-,g,e,e,m,u,o,sh,
i,t,e,a,s,o,b,u 3110.0
```

なお、自動セグメンテーションを複数の音声ファイル対して行う際に使用したスクリプトを付録１０（p.45)、セグメンテーション結果ファイルの例を付録１１（p.47）として添付する。

３．　話者適応

本研究では、音声認識性能を向上させるための話者適応が、自動セグメンテーションにどの程度の影響を及ぼすのかについても検討している。ここでは、ATRSPRECを使った音響モデルの話者適応法について述べる。なお話者適応には、移動ベクトル場平滑化法(Vector Field Smoothing: VFS）を用いる。

３．１　sssdataの作成

３．１．１　実行ファイルの作成

音響モデルの話者適応を行うには、適応に使用する wave データを sssdata（sss：Successive State Splitting 逐次状態分割法)に変換する必要がある。sssdataとは発話を構成する音素と時間情報および特徴ベクトルを結合したファイルである。sssdataを作成する際に必要なモジュールは、TRS ファイルの情報を元に音声ファイルから音声区間を取り出すためのATRwavecut、入力波形をケプストラムに変換するATRwave2cep、ケプストラムを特徴パラメーターに変換するATRcep2para、ATRwavecutで生成した正解音素列をイベントループに送るATRanssync、sssdataフォーマットのファイルを作成するATRsssdataであり、以上をコンパイルしてひとつの実行ファイルとする。実行ファイル作成コマンドをリスト１０に示す。

リスト １０　sssdata 作成用実行ファイルの作成コマンド

```
make -f $ATRSPREC/src/ATRMAINGEN/makefile.template¥
MODULES=ATRwavecut,ATRwave2cep,ATRcep2para,ATRanssync,ATRsssdata¥
NAME=ATRexec clobber all
```

$ATRSPREC は、２．１節の環境設定の際に、設定ファイルの環境変数で指定した ATRSPREC のホームディレクトリである。このコマンドにより、"NAME="で指定した名前の sssdata 作成用実行ファイル(上の例では ATRexec)ができる。

３．１．２　コンフィギュアファイルの作成

自動セグメンテーションと同様、sssdata 作成ファイルを実行するには、統合されたそれぞれのモジュールに応じたコンフィギュアファイルが必要になる。コンフィギュアファイルの例を付録１２（p.49）として添付する。

I/Ocontrol は入出力ファイルについてのコンフィギュレーション、ATRwavecut、ATRwave2cep、ATRcep2para、ATRsssdata はそれぞれ同名のモジュールに対応したコンフィギュレーションである。コンフィギュレーションの中身は sssdata を作成する環境に応じて変更する必要がある。それぞれの設定の意味する内容、変更方法は ATRSPREC リファレンスマニュアル[6]に記載されている（リファレンスマニュアル 15.1　I/Ocontrol、15.10 ATRwavecut、15.3　ATRwave2cep、15.4　ATRcep2para、15.12　ATRsssdata）。なお sssdata 作成時も、２．３．２節でで述べた CMS ファイルが必要になり、コンフィギュレーション中の ATRwave2cep:MeanInFile で、この CMS ファイルの位置を相対もしくは絶対パスで指定する。

３．１．３　sssdata 作成

前節までの実行ファイル、コンフィギュアファイルを用い、sssdata を作成する。実行コマンドをリスト１１に示す。

リスト １１　sssdata 作成コマンド

```
./ATRexec -config=コンフィギュアファイル名¥
-I/Ocontrol:inputFd=入力音声ファイル名¥
-ATRwavecut:TRS=TRS ファイル名¥
-ATRsssdata:outputFd=結果ファイル名
```

config には、前節で作成したコンフィギュアファイル名を指定する。作成された sssdata ファイルはバイナリ形式である。なお、複数の音声ファイル対して sssdata を作成する際に使用したスクリプトを付録１３（p.51）として添付する。

３．２　話者適応の実行

sssdata を用い音響モデルに話者適応をかける。リスト１２はコマンドの一例である。

リスト １２　話者適応コマンドの例

```
cat sssdata ファイル名 | $ATRSPREC/bin/Exe.adapt_HMnet -di 26¥
-if 音響モデル名 -kn 6 -tm 9 -st 0 -em 1 -it 10 -tt 0 -mt 2 -of 話者適応後音響モデル名
```

与える sssdata ファイルは、話者適応に使用する sssdata をすべて cat でつないだものである。コマンド中にある Exe.adapt_Hmnet の各オプションについては ATRSPREC リファレンスマニュアル[6]に記載されている（17.7　HMnet の話者適応 Exe.adapt_HMnet）。

今回の分析では、移動ベクトルの平滑化近傍数を６、平滑化の方法をガウス法、パラメーター推定の方法を MAP 推定法、最尤推定の繰り返し回数を１０、MAP 推定における hyper-parameter(γ)値を２に固定して話者適応を行った。また学習モードとしては、平均値ベクトルの移動ベクトル場平滑化法による適応と、状態遷移確率の Baum-Welch 法による適応を選択した。

4　評価方法

データベース付随のハンドラベリング値を正解とし、自動セグメンテーションの結果と比較することで評価を行った。ATR 音声データベースに付与されたハンドラベリングのセグメント境界時刻の信頼性は極めて高く，ラベラーの違いによる誤差の平均値は一部のラベル（母音の出わたり，半母音，拗音）を除いて 5 ms 以内と報告されている[1、9]。各推定境界ごとに、ハンドラベリング値よりも時刻が前寄りにずれたら負、後ろ寄りにずれたら正として時間差を計算し、評価に用いた。

以下に，評価実施に先立ち必要となった２通りの処理について説明する。

４．１　評価情報の取り出し

付録１１（p.47）に示すように、ATRSPRECから得られる自動セグメンテーションの結果ファイルには様々な情報が含まれている。前処理として、結果ファイルから評価に必要な音素列と音素境界部の時間情報のみを取り出した。この処理に使用したスクリプトを付録１４（p.52)、取り出した音素列と時間情報の例を付録１５（p.54）として添付する。

４．２　評価対象外情報の取り扱い

ハンドラベリングとTRSとのフォーマットの相違により、両者で対応が取れないセグメント境界は評価対象外とし、以下のように対処した。以下の対処法を用い、自動セグメンテーション結果とハンドラベリング情報に対し、比較対象となる音素境界部の同定を行うのに用いたスクリプトを付録１６（p.56)、同定後の音素列、時間情報を付録１７（p.59）に示す。

４．２．１　促音

ハンドラベリングでは、促音部分は後続の子音部に含められて記述されている。例えば「バット」は「b/a/tt/o」と表現されるが、TRS フォーマットでは「b/a/q/t/o」と表現される。評価の際は、"q"の終了部分は評価対象からはずした。

４．２．２　拗音

拗音も促音と同様､ハンドラベリングでの表記法とTRSフォーマットで異なる。例えば、「きゅ」は「ky/u」と表現されるが、TRS フォーマットでは「k/j/u」と表現される。評価の際は、"j"の開始部分は評価対象からはずした。

４．２．３　その他

ハンドラベリングでは、境界の決定が困難である場合、分割は行わずラベル記号をカンマで区切って列記されている。たとえば「き」が「k,i」と表記される場合（母音の無声化）や、「あー」が「a,a」と表記される場合（長母音）などである。それぞれカンマ部分には時間情報が存在しないことから、このような音素境界を評価対象からはずした。

５．結果

５．１　全音素境界に対する評価

まず、今回使用したデータベース中の音素境界全体に対する評価結果について述べる。

評価に利用した統計値は、境界時刻差の絶対値の平均を取った絶対平均誤差、誤差の標準偏差、誤差の絶対値が 30 ms、50 ms 以内に含まれる境界の比率である。この統計値を用い以下の 3 つの観点から比較を行った。なお、付録１７（p.59）の比較対象同定後のファイルを用い、以上の統計値を得るのに用いたスクリプトを付録１８（p.61）、出力結果を付録１９（p.64）として添付する。

５．１．１　音響モデル、音声認識エンジン間の比較

自動セグメンテーションを行うときに使用する音響モデルの種類に応じて、セグメンテーション精度がどのように変わるのかを調べた。比較に利用した音響モデル 3 種類を、表１、自動セグメンテーションの結果と正解との比較結果を表２に示す。

表 1　評価に用いた音響モデルの種類

	学習資料	使用用途	Hmnet の状態数、混合数
モデル A	音素バランス文(男性 165 話者、女性 235 話者)	朗読	音声 1400 状態 5 混合、無音 3 状態 10 混合
モデル B	旅行対話(男性 167 話者、女性 240 話者)	自然発話	音声 1400 状態 5 混合、無音 3 状態 10 混合
モデル C	旅行対話(男性 166 話者、女性 235 話者)	自然発話	音声 1500 状態 3 混合、無音 3 状態 10 混合

表 2　音響モデル毎の評価結果

	絶対平均誤差	標準偏差	±30 ms 以内	±50 ms 以内	平均誤差
モデル A	14.68 ms	20.84 ms	90.86 %	97.36 %	2.85 ms
モデル B	14.53 ms	20.39 ms	91.24 %	97.50 %	2.58 ms
モデル C	14.34 ms	20.02 ms	91.32 %	97.68 %	2.11 ms

使用用途のみが異なる音響モデル A、B では、各比較要素値でそれほど大きな違いがみられない。今回使用した音声データは朗読文であるが、この音声データの種類と音響モデルの用途には必ずしも相関があるわけではないことが分かった。

また、音響モデル B、C 間にもそれほど大きな違いがみられないことから、 HMM の状態、混合数[12]の違いも、必ずしもセグメンテーション精度に影響を与えるわけではないことが分かった。

次に、自動セグメンテーションに利用する音声認識エンジン間の比較を行う。ATRSPREC 以外の音声認識エンジン(HTK、Julius)を用いた自動セグメンテーション結果を表３に示す。この結果は、同一データベースを用いて自動セグメンテーション実験を行ったことを報告している過去の論文[3]から抜粋したものである。

表 3　HTK,Julius を用いた自動セグメンテーションの評価結果

	絶対平均誤差	標準偏差	±30 ms 以内	±50 ms 以内	平均誤差
HTK		19.3 ms	89.9 %	97.3 %	4.0 ms
Julius		19.3 ms	89.6 %	97.8 %	4.3 ms

絶対平均誤差の情報は記載されていなかったため比較できないが、その他の統計値と表２の結果を比較すると、音声認識エンジン間で自動セグメンテーション精度の相違はほとんどないことが分かった。

５．１．２　話者適応前後の比較

３章で述べた方法で音響モデルＡ、Ｂに話者適応をかけた。この音響モデルを用いた場合の正解との比較結果を表４に示す。

表 4　話者適応後の評価結果

	絶対平均誤差	標準偏差	±30 ms 以内	±50 ms 以内	平均誤差
モデル A	14.19 ms	19.74 ms	91.48 %	97.66 %	2.87 ms
モデル B	14.08 ms	19.49 ms	91.80 %	97.85 %	2.82 ms

音響モデルＡ、Ｂ共に、絶対平均誤差約 0.5 ｍｓ、標準偏差約１ ｍｓの改善がみられた。しかし話者ごとで見ると、両方のモデルで１話者ずつについては、話者適応後の方が正解との比較結果が若干悪くなった（表５）。

表 5　話者適応による改善が見られなかった場合の評価結果

	絶対平均誤差	標準偏差	±30 ms 以内	±50 ms 以内	平均誤差
モデル A 話者適応前(話者ID:FKN)	13.22 ms	17.44 ms	94.15 %	99.12 %	2.66 ms
モデル A 話者適応後(話者ID:FKN)	13.53 ms	18.39 ms	93.45 %	98.80 %	2.67 ms
モデル B 話者適応前(話者ID:MHT)	13.17 ms	17.09 ms	95.07 %	99.06 %	2.21 ms
モデル B 話者適応後(話者ID:MHT)	13.57 ms	17.38 ms	94.99 %	99.02 %	2.25 ms

話者適応は、平均的にセグメンテーション精度改善にプラスに働くようだが、話者によっては改善の見られない場合もあることが分かった。

５．１．３　話者間の比較

全話者についてセグメンテーション性能を比較した。自動セグメンテーションに使用する音響モデル（モデルＡ話者適応前、後，モデルＢ話者適応前、後，モデルＣ）に応じて、話者間で各統計値に若干のばらつきが見られ、その値の最大差は、絶対平均誤差で４ms、標準偏差で６ms、±30 ms 以内の比率で７％、±50 ms 以内の比率で５％であった。

しかしどの音響モデルを使用した場合も、精度の良さ、悪さが目立つ話者はいないことから、話者によるセグメンテーション精度の差はその話者の音質と音響モデルの相性の影響であると考えられる。

５．２　音素特性による評価

次に、音素特性の違いによるセグメンテーション精度への影響を調べた。音素境界前後の音素の種類（母音、有声子音、無声子音、無声区間）に応じて、正解との誤差がどのように分布しているかを見る。ここでは音響モデル、話者を限定（モデル B 話者適応後、話者 ID : MHT）して得られた結果について述べるが、他の音響モデル、話者についても同様の結果が得られている。なお、音素それぞれについて、母音子音、有声無声など様々な情報を出力するためのスクリプトを付録２０（p.65）、出力結果の例を付録２１（p.83）として添付する。

５．２．１　母音、子音間

母音と有声子音あるいは無声子音との間の各音素境界について、正解との誤差分布を図２～図５、それぞれの境界数、誤差平均値、標準偏差を表６に示す

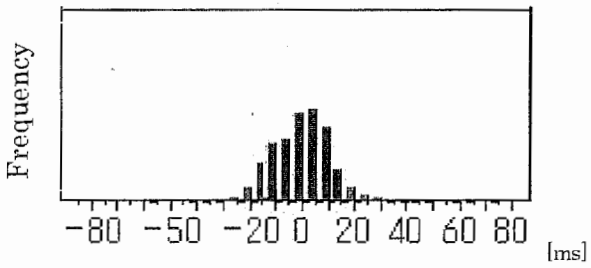

図 2　母音から有声子音への境界の誤差分布

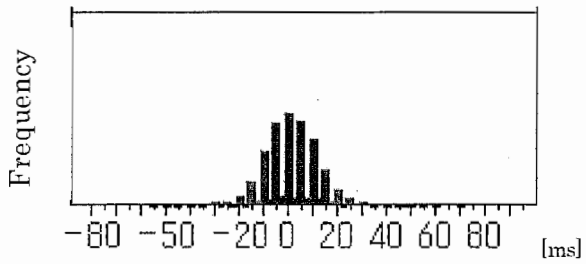

図 3　母音から無声子音への境界の誤差分布

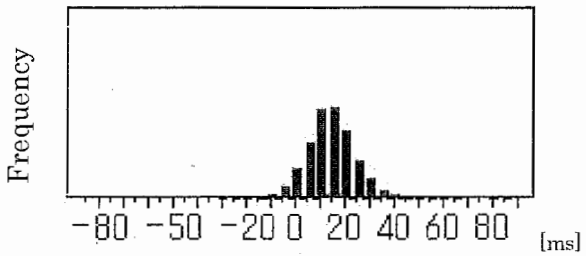

図 4　有声子音から母音への境界の誤差分布

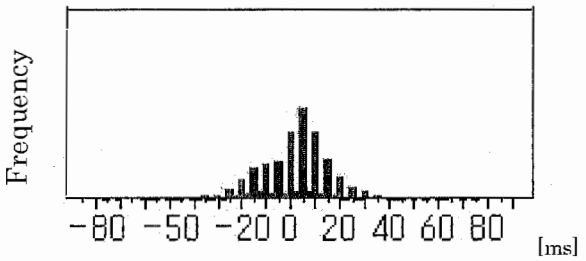

図 5　無声子音から母音への境界の誤差分布

表 6　母音子音間の境界数、誤差平均値、標準偏差

	境界数	平均誤差(ms)	標準偏差(ms)
母音ー有声子音	4912	0.92	11.41
母音ー無声子音	3222	1.46	11.33
有声子音ー母音	5181	13.81	10.04
無声子音ー母音	4199	1.26	14.70

有声子音から母音への間では、推定境界が後方にずれて分布しており、平均誤差も正に大きい値になっている。他の母音、子音間境界では、ほぼ 0 ms を中心に誤差が分布している。

5．2．2　母音、母音間

母音と母音との間の音素境界について、正解との誤差分布を図６、境界数、誤差平均値、標準偏差を表７に示す。

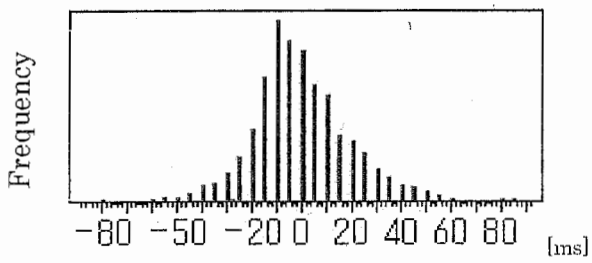

図 6　母音と母音との境界の誤差分布

表 7　母音間の境界数、誤差平均値、標準偏差

	境界数	平均誤差(ms)	標準偏差(ms)
母音ー母音	4912	-0.62	21.26

ほぼ０ms を中心に誤差が分布しているが、標準偏差値が他の音素間誤差分布に比べて大きい。つまり推定境界誤差が大きくなる傾向にあることが分かる。

5．2．3　子音、子音間

子音が連続するのは、子音にはさまれた母音が無声化する場合に限られる。子音と子音との間の音素境界について、正解との誤差分布を図７、境界数、誤差平均値、標準偏差を表８に示す。

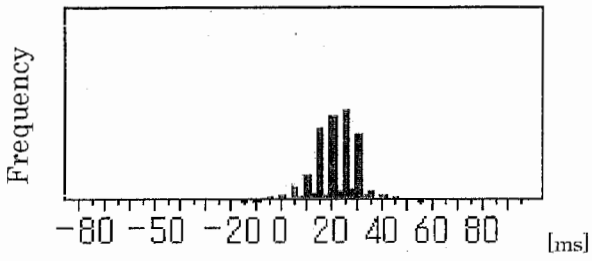

図 7　子音と子音との境界の誤差分布

表 8　子音間の境界数、誤差平均値、標準偏差

	境界数	平均誤差(ms)	標準偏差(ms)
子音－子音	477	21.21	8.72

子音と子音との間では、推定境界が後方にずれて分布しており、誤差平均値も正に大きい値になっている。

5．2．4　母音、無音区間間

無音区間とは、文中のポーズ区間と発話前後のポーズ区間を指す。母音と無音区間との間の音素境界について、正解との誤差分布を図８、図９、それぞれの境界数、平均誤差、標準偏差を表９に示す。

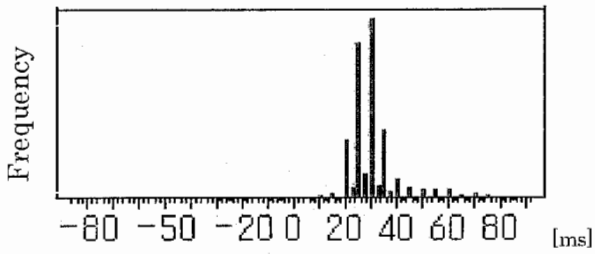

図 8　母音から無音区間への境界の誤差分布

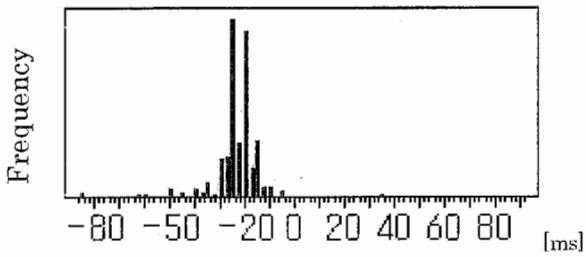

図 9　無音区間から母音への境界の誤差分布

表 9　母音と無音区間との間の境界数、誤差平均値、標準偏差

	境界数	平均誤差(ms)	標準偏差(ms)
母音－無声区間	1432	30.95	11.84
無声区間－母音	301	-24.10	11.60

母音から無音区間への間では、推定境界が後方に分布しており、平均誤差も正に大きい値になっている。また、無音区間から母音への間では、推定境界が前方に分布しており、平均誤差も負に大きい値になっている。つまり、推定境界が無音区間側へずれ込むという傾向がある。

５．２．５　子音、無音区間間

無音区間から有声子音、無声区間から無声子音への音素境界について、正解との誤差分布を図１０、図１１、それぞれの境界数、平均誤差、標準偏差を表１０に示す。

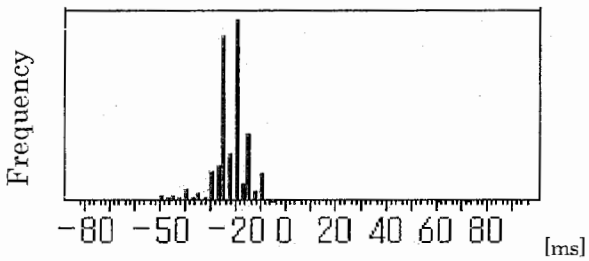

図 2　無音区間から有声子音への境界の誤差分布

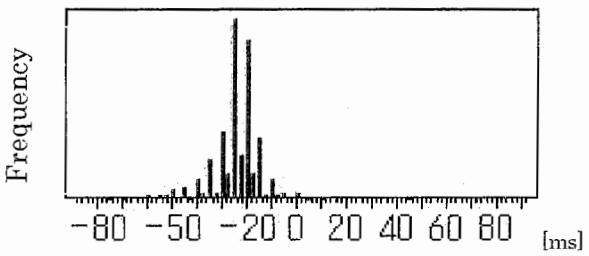

図 3　無音区間から無声子音への境界の誤差分布

表 10　子音と無音区間との間の境界数、誤差平均値、標準偏差

	境界数	誤差平均値(ms)	標準偏差(ms)
無音区間－有声子音	547	-22.32	9.54
無音区間－無声子音	659	-24.50	10.43

両方とも、推定境界が前方に分布しており、誤差平均値も負に大きい値になっている。

また、子音から無音区間への場合、連鎖するパターンとして考えられるのは無音区間前の母音が無声化する場合である。全データ中５例しかなかったが、全例共に推定境界が後方に分布していた。

つまり、子音と無音区間との間では推定境界が無音区間側へずれ込む傾向にあることがわかる。

6．考察

今回の実験で得られた自動セグメンテーションの結果は、ハンドラベリングの精度には及ばないものの、応用目的によっては利用価値のある精度であると言える。今回は実施できなかったが、セグメンテーション単位の補正を加えることで、比較的小さいセグメンテーション単位を必要とする応用目的にも対処しうるだろう。しかもその精度は、今回比較に用いた音響モデル、音声認識エンジンの範囲内において、それほど大差ないことが分かった。また、話者によるセグメンテーション精度の違いも、音響モデルとの相性によって生じる比較的小さな差の範囲内におさまることが分かった。

話者適応はセグメンテーション性能向上に有効であったが、適応の際のパラメーター変更により、更に向上の幅が広がる可能性も考えられる。しかし最良の話者適応効果を得るためには、様々なパターンを試して探す他ないであろう。

音素特性を考慮に入れた分析の結果からは、以下の２点が明らかになった。

● 無声、有声部の境界では、境界部が無声区間側に張り出す。

音声区への境界の開始部、終了部において無音の判断が厳しくなる傾向が見られた。原因としては、ハンドラベリングでは波形の視察によりある時間点によって音素境界部を判断できるのに対し、自動セグメンテーションでは、フレームごとに音素境界部を判断するためと考えられる。フレーム内に少しでも音声波形が見られる時点では無声と判断しないため、境界が無声区間に入り込むことにつながっている可能性がある。

● 同系統連続音素の境界部では、境界誤差が大きくなりやすい。

まず２母音間境界部では、誤差平均値は０ms に近いものの、標準偏差が大きく、誤差の大きくなる傾向が確認された。また２子音間では、境界誤差が後方に大きくずれる傾向があった。さらに有声無声のカテゴリで考えた場合も、有声子音から母音への境界（有声音素の連続）で推定値が後方に大きくずれる傾向が見られた。しかし母音から有声子音への境界にはこのパターンが当てはまらず、今後の分析課題である。

これらの知見は、自動セグメンテーションに付随する誤差特性として、セグメンテーション結果の応用時に有効な情報になるだろう。

今後は、更に細かい音素カテゴリによる音素境界時間、音韻長、推定音韻長と正解音韻長のオーバーラップ率などについて分析を行いたい。また、セグメンテーション結果と各音素の尤度との関連性についても調べたい。

7．あとがき

本実習では、ATRSPREC を用いた自動セグメンテーション法の獲得、ATR データベース B セットを用いた自動セグメンテーション実験を行った。実験を通じ、音素自動セグメンテーションの現状把握を行い、セグメンテーション結果特性を調べた。

セグメンテーション結果の応用分野次第では、自動セグメンテーションは十分利用可能であることが分かった。また自動セグメンテーションでは誤差修正の余地部分のみをハンドラベリングするなどの方法で、ハンドラベリングにより近い精度を出すことも可能であろう。

今後、更に細かい音素カテゴリによる分析、セグメンテーション精度と音素尤度の関連性分析を行っていきたい。また、他の朗読、自然発話データを用いた実験も行い、データによる自動セグメンテーション特性の不変性も確かめたい。

8．謝辞

今回の実習の実施、並びに最終報告書作成にあたっては多くの方々のお世話になった。特に以下の方々の助力がなければ今回の様な成果は得られなかっただろう。ここに記して感謝したい。

中東さん、段原さん（ATR 音声言語通信研究所 ラベラー）：ハンドラベリングについてのアドバイス
山本さん、林さん、伴さん、奥田さん（ATR 音声言語通信研究所）：SPREC の技術的アドバイス
大槻さん（ATR-ISD/SLAP）：ATR 音声データベースの使い方，最終報告書の改善についてのアドバイス
久保さん（ATR-ISD/SLAP）：ハンドラベリングについてのアドバイス
兵頭さん（ATR-ISD/SLAP）：計算機環境全般についてのアドバイス
高田さん、山下さん、中川みゆきさん、中川佳代子さん、西田さん、足立さん、上田さん、駒木さん、岸本さん、ハリーさん、アマンダさん（ATR-ISD/SLAP）：ATR-ISD/SLAP での生活全般のお世話

最後に、本実習の結果に対し貴重なアドバイスをしていただいた山田プロジェクトリーダー、一学生に ATR での実習という貴重な機会を与えてくださった匂坂先生に感謝の意を表し、謝辞とする。

9．参考文献

[1] 武田、匂坂、片桐、桑原（1988)「研究用日本語音声データベースの構築」日本音響学会誌 44、pp. 747-754.

[2] 岩橋、藤原、小森、杉山、匂坂（1991)「自動セグメンテーションによる音声合成単位の作成」日本音響学会秋季研究発表会講演論文集 I, pp. 231-232.

[3] 大須賀、堀内、市川（2001)「音素境界認識の自動化の検討」日本音響学会春季研究発表会講演論文集 I, pp. 107-108.

[4] 米良、李、猿渡、鹿野（2001）「Julius を用いた自由発話の自動ラベリングにおけるラベル尤度の統計的分析」日本音響学会秋季研究発表会講演論文集 I, pp. 57-58.

[5] 内藤、山本、シンガー、中嶋、中村、匂坂（2001)「対話音声を対象とした連続音声認識システムの試作と評価」電子情報通信学会論文誌, Vol.J84-D-II, No.1, pp. 31-40.

[6] ATR 音声翻訳通信研究所（2000)「ATRSPREC（Ver06r06）リファレンスマニュアル」

[7] 匂坂、浦谷（1992)「ＡＴＲ音声・言語データベース」日本音響学会誌 48, pp. 878-882.

[8] 阿部、匂坂、梅田、桑原（1990)「研究用日本語音声データベース利用解説書（連続音声データ編）」ATR テクニカルレポート, TR-I-0166.

[9] 武田、匂坂、片桐、桑原（1988)「音声データベース構築のための視察に基づく音韻ラベリング」ATR テクニカルレポート, TR-A-0019.

[10] 深田、シュスター、匂坂（1998)「双方向リカレントニューラルネットワークを用いた音素境界推定とその応用」電子情報通信学会論文誌, Vol.J81-D-II, No.7, pp. 1481-1490.

[11] 古市、相澤、井上（1999)「音素セグメンテーションに基づく統計的音素セグメントモデルによる音声認識」電子情報通信学会論文誌, Vol.J82-D-II, No.7, pp. 1111-1119.

[12] 鹿野、伊藤、河原、武田、山本（2001)「音声認識システム」オーム社.

付録１　スクリプト、コマンド、入出力ファイル一覧

２．自動セグメンテーションの方法

２．２　前処理

２．２．１　音声データのサンプリング周波数の調整（20 kHz→16 kHz）

スクリプト	付録３＜20kto16k.csh＞（p. 29）
入力	ATR 音声 DB の音声ファイル(*.AD)
出力	ダウンサンプリング後の音声ファイル(*.AD.16k)

２．２．２　ATRSPREC に与える正解音素列の作成

スクリプト	付録５＜LBtoTRS.pl＞（p. 32）
入力	ATR 音声 DB のハンドラベリング結果ファイル(*.LB)
出力	TRS フォーマットファイル(例：リスト２　*.trs（p. 3）)

２．２．３　セグメンテーション単位の補正（10 ms→2.5 ms）

今回は実施せず

２．３　自動セグメンテーション

２．３．１　アスキーファイルの作成

スクリプト	付録６＜newascii.csh＞（p. 37）
入力	ATR 音声 DB の音声ファイル(*.AD.16k)
出力	アスキーファイル(例：リスト３　話者 ID.ascii（p. 4）)

２．３．２　CMS ファイルの作成

スクリプト	・付録８＜sampledo.CMS.csh＞（p. 41） ・atrCM.py(sampledo.CMS.csh の内部で使用、ATRSPREC に含まれる python スクリプト)
入力	・ATR 音声 DB の音声ファイル(*.AD.16k) ・TRS フォーマットファイル(*.TRS) ・アスキーファイル(話者 ID.ascii) ・atrCM.py 用コンフィギュアファイル(例：付録７＜CMS.config＞（p. 39）)
出力	CMS ファイル(例：リスト４　*.mean（p. 5）)

２．３．３．　自動セグメンテーションの実行

２．３．３．１　実行ファイルの作成

コマンド	リスト６　自動セグメンテーション実行ファイルの作成コマンド（p. 6）
入力	リスト５　ATRinput.mod ファイル（p. 6）
出力	実行ファイル（コマンド中の、"NAME="で指定した名前）

２．３．３．３　自動セグメンテーションの実行

コマンド	リスト７　自動セグメンテーションの実行コマンド（p. 7）
入力	・ATR 音声 DB の音声ファイル(*.AD.16k) ・wave ファイルの絶対パスを１行に１つ格納した wave ファイルリスト（例：リスト８（p. 7）） ・個々の wave ファイルに対応した TRS ファイルを順番に cat でつないだ、一つの長い TRS ファイル（例：リスト９（p. 8）） ・CMS ファイル（*.mean） ・自動セグメンテーション用コンフィギュアファイル（例：付録９＜SEG.config＞（p. 43））
出力	結果ファイル（例：付録１１＜sample_result.res＞（p. 47））

スクリプト（付録１０＜sample_viterbi.csh＞（p. 45））を使用して自動セグメンテーションを行うことも可能である。

３　話者適応

３．１　sssdata の作成

３．１．１　実行ファイルの作成

コマンド	リスト１０（p. 9）
出力	実行ファイル（コマンド中の、"NAME="で指定した名前）

３．１．３　sssdata 作成

コマンド	リスト１１（p. 9）
入力	・ATR 音声 DB の音声ファイル(*.AD.16k) ・TRS フォーマットファイル(*.TRS) ・sssdata 作成用コンフィギュアファイル（例：付録１２＜sssdara.config＞（p. 49））
出力	sssdata ファイル（コマンド中の、"outputFd="で指定した名前）

スクリプト（付録１３＜mk_sssdata.csh＞（p. 51））を使用して sssdata を作成することも可能である。

3．2　話者適応の実行

コマンド	リスト１２（p. 10）
入力	・sssdata ファイル ・話者適応をかける音響モデル
出力	話者適応後の音響モデル

4　評価方法

4．1　評価情報の取り出し

スクリプト	付録１４＜resultprs1.pl＞（p. 52）
入力	ATRSPREC から得られる自動セグメンテーションの結果ファイル（例：付録１１＜sample_result.res＞（p. 47））
出力	音素列と時間情報ファイル（例：付録１５＜res1out＞（p. 54））

4．2　評価対象外情報の取り扱い（比較対象となる音素境界部の同定）

スクリプト	付録１６＜resultprs2.pl＞（p. 56）
入力	・自動セグメンテーションの結果から取り出した音素列と時間情報（例：付録１５＜res1out＞（p. 54）） ・ATR 音声 DB のハンドラベリング結果ファイル(*.LB)
出力	同定後の音素列、時間情報ファイル（例：付録１７＜res2out＞（p.59））

5　結果

5．1　全音素境界に対する評価（評価に用いる統計値の計算）

スクリプト	付録１８＜resultprs3.pl＞（p.61）
入力	比較対象となる音素境界部の同定を行った音素列、時間情報ファイル（例：付録１７＜res2out＞（p. 59））
出力	各統計値情報を含むファイル（付録１９＜res3out＞（p.64））

5．2　音素特性による評価（音素それぞれについて、母音子音、有声無声など様々な情報を出力）

スクリプト	付録２０＜resultprs4.pl＞（p. 65）
入力	比較対象となる音素境界部の同定を行った音素列、時間情報ファイル（例：付録１７＜res2out＞（p. 59））
出力	音素それぞれについて、母音子音、有声無声など様々な情報を

	持たせたファイル（例：付録２１<res4out>（p. 83））

付録２　ATRSPREC 用環境変数、パスの設定(SPREC.setup)

```
# the following variables are completely machine/OS dependent and should
# be set via .bashrc or .cshrc for this particular machine
# (example for Linux/HP-UX/SunOS/OSF1 at ATR ITL)
#
# SPREC_INPUTBYTEORDER byteorder from audio input
# SPREC_ADIN audio input command
# SPREC_OUTPUTBYTEORDER byteorder to audio output
# SPREC_DAOUT  audio output command
# SPREC_MACHINEBYTEORDER byteorder for eventloop input

set samplingfrequency=16000
set bitpersample=16

   setenv INSTALL_DIR       /usr/local/SPREC
   setenv ATRSPREC          $INSTALL_DIR/SPRECr07r03
   setenv PYTHON_TOOL_PREFIX  $INSTALL_DIR/bin

   setenv LD_LIBRARY_PATH    $ATRSPREC/shlib:"$LD_LIBRARY_PATH"
   #setenv SHLIB_PATH       $ATRSPREC/shlib:"$SHLIB_PATH"
   setenv OS Linux
   set path  = ($ATRSPREC/bin $path)

# for SLT environment
   setenv PYTHONHOME $PYTHON_TOOL_PREFIX/python-1.5.2
   setenv ATR_PYTHON $PYTHON_TOOL_PREFIX/python-1.5.2/bin/python
   setenv ATR_SWIG $PYTHON_TOOL_PREFIX/swig1.1p5a/bin/swig
   setenv ATR_LIBSWIG $PYTHON_TOOL_PREFIX/swig1.1p5a/lib/swig_lib
   setenv ATR_SWIGVERSION 1.5
   setenv ATR_INCPY
"-I$PYTHON_TOOL_PREFIX/python-1.5.2/include/python1.5
-I$PYTHON_TOOL_PREFIX/lang/$OS/python-1.5.2/lib/python1.5/config"
   setenv ATR_LIBPY
"-L$PYTHON_TOOL_PREFIX/python-1.5.2/lib/python1.5/config
-lpython1.5"
   setenv ATR_LIBX11 "-L/usr/local/X11/lib -lX11"
```

```
    setenv ATR_LIBTCLPATH $PYTHON_TOOL_PREFIX/tcl-8.0.5jp/lib
    setenv ATR_LIBTKPATH $PYTHON_TOOL_PREFIX/tk-8.0.5jp/lib
    setenv ATR_LIBTCL -ltcl8.0jp
    setenv ATR_LIBTK -ltk8.0jp
    setenv TK_LIBRARY $PYTHON_TOOL_PREFIX/tk-8.0.5jp/lib/tk8.0jp
#

switch($OS)
  case "HP-UX":
    setenv SPREC_INPUTBYTEORDER BigEndian;
    setenv SPREC_ADIN
"/homes/singer/audio/bin/srecorder -srate
$samplingfrequency -l$bitpersample";
    setenv SPREC_OUTPUTBYTEORDER BigEndian;
    setenv SPREC_DAOUT
"/homes/singer/audio/bin/splayer -srate
$samplingfrequency -l$bitpersample";
    setenv SPREC_MACHINEBYTEORDER BigEndian;
    breaksw
  case "OSF1"
    setenv SPREC_INPUTBYTEORDER BigEndian;
    setenv SPREC_ADIN
"/usr/local/datlink/bin/narecord -o left -p $bitpersample -e linear -f
raw -s $samplingfrequency";
    setenv SPREC_OUTPUTBYTEORDER LittleEndian;
    setenv SPREC_DAOUT
"/usr/bin/mme/audioplay -rate $samplingfrequency
-bitspersample $bitpersample -encoding pcm -filename -";
    setenv SPREC_MACHINEBYTEORDER LittleEndian;
    breaksw
  case "SunOS"
    setenv SPREC_INPUTBYTEORDER BigEndian;
    setenv SPREC_ADIN
"/home/atrh34/hayashi/MIKOSHI/AUDIO.SunOS/eaw16rec -g 10 -v -m -r
$samplingfrequency";
    setenv SPREC_OUTPUTBYTEORDER BigEndian;
```

```
    setenv SPREC_DAOUT
"/usr/demo/SOUND/raw2audio -s $samplingfrequency -p $bitpersample
 -e LINEAR | /usr/demo/SOUND/play";
    setenv SPREC_MACHINEBYTEORDER BigEndian;
    breaksw
  case "Linux"
    setenv SPREC_INPUTBYTEORDER LittleEndian;
    #setenv SPREC_ADIN "adda $samplingfrequency $bitpersample w";
    setenv SPREC_ADIN "arecord -s $samplingfrequency -f $bitpersample
-r";
    setenv SPREC_OUTPUTBYTEORDER LittleEndian;
    #setenv SPREC_DAOUT "adda $samplingfrequency $bitpersample r";
    setenv SPREC_DAOUT "aplay -s $samplingfrequency -f $bitpersample
-r";
    setenv SPREC_MACHINEBYTEORDER LittleEndian;
    breaksw
  default:

endsw
unset samplingfrequency
unset bitpersample
```

付録3　ESPSを用いたダウンサンプリング用スクリプト（20kto16k.csh）

```
#!/bin/csh -f
#ESPSを用い20kHzから16kHzに変換後、
#ddを用い、上位バイトと下位バイトを入れ換えるためのスクリプト

#usage----- 第一引数、20kHzの音声ファイル名リスト

set fl = $argv[1]
set list = `cat $fl`
# loop
foreach file ($list)

   set dest = "$file:r".temp
      /usr/local/esps531.linux/bin/btosps -E -f 20000 -c nocomment
      $file - | /usr/local/esps531.linux/bin/sfconvert -s 16000 - -
      | /usr/local/esps531.linux/bin/bhd - $dest

   set out = "$dest:r".AD.16k
   dd if=$dest of=$out conv=swab
end
```

付録４　ハンドラベリングとTRSフォーマットで表記法が異なるものの全対応表

	ハンドラベリングフォーマット	TRSフォーマット
ふ	f,u	h,u
や	y,a	j,a
ゆ	y,u	j,u
よ	y,o	j,o
を	O	w,o
ん	N	ng
じ	j,I	zh,i
きゃ	ky,a	kj,a
きゅ	ky,u	kj,u
きょ	ky,o	kj,o
しゃ	sy,a	sj,a
しゅ	sy,u	sj,u
しょ	sy,o	sj,o
ちゃ	ch,a	chj,a
ちゅ	ch,u	chj,u
ちょ	ch,o	chj,o
にゃ	ny,a	nj,a
にゅ	ny,u	nj,u
にょ	ny,o	nj,o
ひゃ	hy,a	hj,a
ひゅ	hy,u	hj,u
ひょ	hy,o	hj,o
みゃ	my,a	mj,a
みゅ	my,u	mj,u
みょ	my,o	mj,o
りゃ	ry,a	rj,a
りゅ	ry,u	rj,u
りょ	ry,o	rj,o
ぎゃ	gy,a	gj,a
ぎゅ	gy,u	gj,u
ぎょ	gy,o	gj,o
じゃ	j,a	zhj,a

じゅ	j,u	zh,j,u
じょ	j,o	zh,j,o
でゃ	dy,a	d,j,a
でゅ	dy,u	d,j,u
でょ	dy,o	d,j,o
びゃ	by,a	b,j,a
びゅ	by,u	b,j,u
びょ	by,o	b,j,o
ぴゃ	py,a	p,j,a
ぴゅ	py,u	p,j,u
ぴょ	py,o	p,j,o
～は	h,a	w,a
促音	子音の連続 （2文字で表される子音は 先行文字の連続） 例　みっつ　m/i/tts/u	q + 子音 例　みっつ　m/i/q/ts/u

付録５　ハンドラベリングフォーマットから TRS フォーマットへの翻訳スクリプト (LBtoTRS.pl)

```
#!/usr/bin/perl
#ラベルファイル（.LB）から第一層を選択-TRS ファイル化

#usage----- 第一引数 ラベルファイルリスト

$flist = $ARGV[0];
open INL,"$flist";
@files = <INL>;

foreach $fname1 (@files){

    $temp = $fname1;
    $temp =~ s/LB/TRS/g;
    $fname2 = $temp;

    open IN,"$fname1";
    open OUT,">$fname2";

    @array = <IN>;
#---音素列の初期化
    @Oarray = ();

#---start 時間を取り出す
    @temp0 = split(/ /,$array[0]);
    $starttime = $temp0[0];

#---一層め終了行
    $layer1end = 0;

#---ha 用カウンター
    $hcount = 0;

#---それぞれの音素が配列に入っている状態にする
    foreach $temp1(@array){
```

```
        if($temp1 == "#"){
            $layer1end++;
            last;
        }

        @temp2 = ();
        @temp3 = ();

        @temp2 = split(/ /,$temp1);
        @temp3 = split(/,/,$temp2[1]);

        #---ha の第 2 層参照用
        for($s=0;$s<=$#temp3;$s++){
            if($temp3[$s] eq h){
                @Hendtimearray = (@Hendtimearray,$temp2[2]);
            }
        }

        @Oarray = (@Oarray,@temp3);
        $layer1end++;
    }

#---end 時間を取り出す
    $endtime = $temp2[2];

#---TRS ファイル書き出し
    print OUT $starttime;
    print OUT " ";

    for($i=0 ; $i<=$#Oarray ; $i++){

#--ポーズ
        $Oarray[$i] =~ s/pau/-/;

#--促音
```

```
        $Oarray[$i] =~ s/(.)¥1/q,¥1/;

#--j
        if($Oarray[$i] =~ /j/){
            $nextonso = $Oarray[$i+1];
            if($nextonso =~ /[auo]/){
                $Oarray[$i] =~ s/j/zh,j/;
            }else{
                $Oarray[$i] =~ s/j/zh/;
            }
        }

#--sh
        if($Oarray[$i] =~ /sh/){
            $nextonso = $Oarray[$i+1];
            if($nextonso =~ /[auo]/){
                $Oarray[$i] =~ s/sh/sh,j/;
            }
        }

#--ch
        if($Oarray[$i] =~ /ch/){
            $nextonso = $Oarray[$i+1];
            if($nextonso =~ /[auo]/){
                $Oarray[$i] =~ s/ch/ch,j/;
            }
        }

#--ha   ch,sh,等は除外
        if($Oarray[$i] eq "h"){
            $nextonso = $Oarray[$i+1];
            if($nextonso eq "a"){
                for($j=$layer1end+1 ;$j<=$#array; $j++){
                    @layer2input = split(/ /,$array[$j]);
                    if($layer2input[2] == $Hendtimearray[$hcount]){
                        if($layer2input[1] =~ /w/){
```

```
                    $Oarray[$i] =~ s/h/w/;
                }
                last;                    #2 層目で抜ける
            }
        }
    }
    $hcount++;
}


#--f
    if($Oarray[$i] =~ /f/){
        $nextonso = $Oarray[$i+1];
        if($nextonso =~ /u/){
            $Oarray[$i] =~ s/f/h/;
        }
    }

#--その他
    $Oarray[$i] =~ s/wo/o/;
    $Oarray[$i] =~ s/N/ng/;
    $Oarray[$i] =~ s/ky/k,j/;
    $Oarray[$i] =~ s/hy/h,j/;
    $Oarray[$i] =~ s/ry/r,j/;
    $Oarray[$i] =~ s/gy/g,j/;
    $Oarray[$i] =~ s/py/p,j/;
    $Oarray[$i] =~ s/my/m,j/;
    $Oarray[$i] =~ s/ny/n,j/;
    $Oarray[$i] =~ s/dy/d,j/;
    $Oarray[$i] =~ s/by/b,j/;
    $Oarray[$i] =~ s/y/j/;
    $Oarray[$i] =~ s/he/e/;

    print OUT $Oarray[$i];
    if($i == $#Oarray){
        print OUT " ";
```

```
        }else{
            print OUT ",";
        }

    }

    print OUT $endtime;

    close OUT;
    close IN;
}

close INL;
```

付録６　アスキーファイル作成用スクリプト(newascii.csh)

```
#!/bin/csh -x
#アスキーファイル作成用スクリプト

#usage----- 第一引数 音声ファイルの存在するディレクトリの相対 or 絶対パス
#usage----- 第二引数 話者 ID(出力アスキーファイル名、拡張子として.ascii が付く)

#アスキーファイルは CMS ファイル作成単位に合わせて作る。CMS ファイルを話者毎に作る
場合は、アスキーファイルも話者毎に作る。CMS ファイルを全話者共通に一つ作る場合は、
全話者の全音声ファイルを一つのディレクトリにまとめ、第一引数にそのディレクトリま
でのパスを与えることでアスキーファイルを作成する。

set tmp = ($*)
set cnt = $#tmp
if ( $cnt != 2 ) then
        goto Usage
endif

set WAVEDIR = $1

set  ASCIIFILE = ${2}.ascii
touch $ASCIIFILE

set speaker = $2
set fno = `¥ls $WAVEDIR/*.AD.16k|wc -1`

#  注意！ ： アスキーファイル作成には拡張子が二つ付いた音声ファイルを使用します。
その拡張子に合わせて上の.AD.16k を変更してください。

if ($fno != 0) then
        set role = CUSTOMER

#  注意！  ：  下の行の"AD"これも、対象により変更のこと。一番目の拡張子を指定。

        echo $speaker $speaker $fno AD $role| ¥
        awk
```

```
'{printf("%14s %4s %5s %5s %5s %5s %5s %-11s %9s %15s %-6d %-6s %5s¥n",
$1, "JTEXT", "X", "JMOR", "WAV", "TRS", "X", $2, "NIL", "2001-9-4", $3,
$4, $5);}' > $ASCIIFILE

endif

goto Exit

Usage:
        echo    usage : command WaveDir AsciiFileName
        exit

Exit:
        echo ""
        echo "newascii.csh" $* "...done"
        echo ""
        exit
```

付録７　CMS ファイル作成用コンフィギュアファイルの例(CMS.config)

```
#CMSファイル作成用コンフィギュアファイル
#I/Ocontrol
        I/Ocontrol:inputFormat=NoHeader
        I/Ocontrol:inputParamSize=160
        I/Ocontrol:inputParamType=short
        I/Ocontrol:inputFd=stdin
        I/Ocontrol:inputByteorder=BigEndian
        I/Ocontrol:outputFormat=NoHeader
        I/Ocontrol:outputParamSize=13
        I/Ocontrol:outputParamType=float
        I/Ocontrol:outputFd=/dev/null
        I/Ocontrol:outputByteorder=BigEndian
        I/Ocontrol:rpcNumber=30

#ATRwave2cep
        ATRwave2cep:Preemphasis=0.98
        ATRwave2cep:FrameLength=20
        ATRwave2cep:FrameShift=10
        ATRwave2cep:SamplingFrequency=16000
        ATRwave2cep:TimeWindow=hamming
        ATRwave2cep:LagWindowFactor=0.01
        ATRwave2cep:LpcOrder=16
        ATRwave2cep:CepstrumOrder=12
        ATRwave2cep:FrequencyWarping=mel
        ATRwave2cep:FilterBankOrder=20
        ATRwave2cep:CutoffLowFrequency=0.0
        ATRwave2cep:CutoffHighFrequency=8000
        ATRwave2cep:AnalysisType=fft
        ATRwave2cep:DebuggingLevel=0
        ATRwave2cep:CentroidFreqOrder=0
        ATRwave2cep:CentroidFreqGamma=0.5
        ATRwave2cep:CentroidFreqType=linear
        ATRwave2cep:MeanFrame=0

#ATRwavecut
```

```
        ATRwavecut:pause_symbol=SIL
        ATRwavecut:PausePeriod=NOT
        ATRwavecut:SamplingFrequency=16000.0

#ATRepd
        ATRepd:SamplingFrequency=16000
        ATRepd:energyThreshold=70
        ATRepd:upperDispersionThreshold=30
        ATRepd:lowerDispersionThreshold=8
        ATRepd:orderInMs=300
        ATRepd:alpha=1.04
        ATRepd:skewInMs=300
        ATRepd:slTimerLimitInMs=200
        ATRepd:epdFramesInMs=2000
        ATRepd:framePointsInMs=10
        ATRepd:sendNonspeech=1
        ATRepd:track=0
```

付録８　CMS ファイル作成用スクリプト(sampledo.CMS.csh)

```
#! /bin/csh
#CMS ファイル作成用スクリプト

#usage-------- 引数なし

#以下で ATRSPREC のディレクトリを指定する。
Setenv ATRSPREC /usr/local/SPREC/SPRECr07r03

#以下でアスキーファイルの存在するディレクトリを指定する。
Set ascii = ../ASCIImk

#DirN には、使用するアスキーファイルの拡張子.ascii を取った名前を指定する。
Set DirN = ( MHT )

set endBcount = ${#DirN}
set bCount = 1
while ( $bCount <= $endBcount )


#以下のコマンド行では"-db"の指定，ディレクトリ・拡張子に気をつける。
#CMS ファイル作成には、'wavDir':で指定したディレクトリに存在する、DirN で指定した名前のディレクトリ内の音声ファイルを使用する。同様に、'trsDir':で指定したディレクトリに存在する、DirN で指定した名前のディレクトリ内の TRS ファイルを使用する。
また、それぞれのファイル拡張子を、'wavExt'、'trsExt'で指定する。
#CMS ファイルはこのスクリプトを使用した同一ディレクトリ内にできる、".mean"という拡張子を持つファイルである。
#"config"で、使用するコンフィギュアファイルを相対 or 絶対パスで指定する。


$ATRSPREC/bin/ATRpython $ATRSPREC/script/atrCM.py ¥
        -calcCMS=CalcMeanWav ¥
        -AsciiFile=${ascii}/${DirN[$bCount]}.ascii ¥
        -db="{'cmsDir':'../CM','cmsExt':'CMS','wavDir':'/prj/slap2/p
roject/ATRSpeechDB/B.16k','wavExt':'16k','trsDir':'../TRS','trsExt'
:'TRS'}" ¥
```

```
        -config=config.cm.v9

$ATRSPREC/bin/ATRpython $ATRSPREC/script/atrCM.py ¥
        -calcCMS=None ¥
        -MeanCep=./CM.${DirN[$bCount]}.mean ¥
        -AsciiFile=${ascii}/${DirN[$bCount]}.ascii ¥
        -db="{'cmsDir':'../CM','cmsExt':'CMS','wavDir':'/prj/slap2/p
roject/ATRSpeechDB/B.16k','wavExt':'16k','trsDir':'../TRS','trsExt'
:'TRS'}" ¥
        -config=config.cm.v9

        @ bCount ++
end
```

付録９　自動セグメンテーション用コンフィギュアファイルの例(SEG.config)

```
#自動セグメンテーション用コンフィギュアファイル
#ATRI/Ocontrol config :
        I/Ocontrol:inputFormat=NoHeader
        I/Ocontrol:inputParamSize=160
        I/Ocontrol:inputParamType=short
        I/Ocontrol:inputFd=NULL
        I/Ocontrol:inputEOFexit=OFF
        I/Ocontrol:inputByteorder=BigEndian
        I/Ocontrol:outputFormat=NULL
        I/Ocontrol:outputFd=stdout
        I/Ocontrol:outputByteorder=BigEndian
        I/Ocontrol:rpcNumber=5

#ATRwave2cep config :
        ATRwave2cep:CalcMean=off
        ATRwave2cep:CentroidFreqType=linear
        ATRwave2cep:CentroidFreqGamma=0.5
        ATRwave2cep:CentroidFreqOrder=0
        ATRwave2cep:AnalysisType=fft
        ATRwave2cep:CutoffHighFrequency=8000
        ATRwave2cep:CutoffLowFrequency=0.0
        ATRwave2cep:FilterBankOrder=20
        ATRwave2cep:FrequencyWarping=mel
        ATRwave2cep:CepstrumOrder=12
        ATRwave2cep:LpcOrder=16
        ATRwave2cep:LagWindowFactor=0.01
        ATRwave2cep:TimeWindow=hamming
        ATRwave2cep:SamplingFrequency=16000
        ATRwave2cep:FrameShift=10
        ATRwave2cep:FrameLength=20
        ATRwave2cep:Preemphasis=0.98
        ATRwave2cep:DebuggingLevel=0
        ATRwave2cep:MeanInFile=/home/slap/xhayaka/SPREC/CM/ATRMEAN/C
M.ATR.mean
        ATRwave2cep:Subtract=logpow+cep
```

```
#ATRcep2para config :
        ATRcep2para:LDA=
        ATRcep2para:OutputParameter=cep(12)+dpow+dcep(12)
        ATRcep2para:rho=1.0
        ATRcep2para:DDCepstrumPadding=zero
        ATRcep2para:DDCepstrumWindow=5
        ATRcep2para:deltaCepstrumPadding=zero
        ATRcep2para:DeltaCepstrumWindow=5
        ATRcep2para:CepstrumOrder=12
        ATRcep2para:DebuggingLevel=10
        ATRcep2para:WindowType=rectangular

#ATRviterbi config :
        ATRviterbi:amname=/home/slap/xhayaka/SPREC/ResearchJV9/train
SLFfromTRS.BLA_adapt/AM.M.5.1400.MHTadapt
        ATRviterbi:active_model=all
        ATRviterbi:beam=0
        ATRviterbi:frame_shift=10
        ATRviterbi:boundary=PHONEME
        ATRviterbi:dimension=25

#ATRresult config :
        ATRresult:N_best=1
        ATRresult:N_best_out=stdout
        ATRresult:Lattice_out=/dev/null
```

付録１０　自動セグメンテーション用スクリプト(sample_viterbi.csh)

```
#! /bin/tcsh -x
#自動セグメンテーション用スクリプト

#usage----- 第一引数、話者 ID

#  [name] 話者 ID
#  [waveTopDir]  "waveTopDir/話者 ID 名ディレクトリ/"以下の音声ファイルを用いて自動セグメンテーションを行う
#  [ResultTopDir]  自動セグメンテーション結果はこのディレクトリ内に書き出される
#  [waveName] 自動セグメンテーションを"{話者 ID}_{A-J セット}"の単位で行う
#  [WaveFileListDir] 音声ファイルへの絶対パスが書かれた、"{話者 ID}_{A-J セット}"と言う名前のファイルの存在する場所(拡張子.Flist)
#  [ansTopDir] "{話者 ID}_{A-J セット}"毎の TRS ファイルが存在する場所(拡張子.TRS)

set name = $argv[1];
set waveTopDir = /prj/slap2/project/ATRSpeechDB/B.16k/{$name}

set ResultTopDir =
~xhayaka/SPREC/result/Viterbi_ResearchJV9read_adapt
set waveName = ({$name}_A {$name}_B {$name}_C
{$name}_D {$name}_E {$name}_F {$name}_G {$name}_H {$name}_I {$name}_J)
set WaveFileListDir = ~xhayaka/SPREC/Flist
set ansTopDir = ~xhayaka/SPREC/TRS/{$name}forV

setenv ATRSPREC /usr/local/SPREC/SPRECr07r03

set originCurrent = `pwd`

set SPREC_BASE = ${originCurrent}/ATRsprec
make -f
$ATRSPREC/src/ATRMAINGEN/makefile.template
MODULES=ATRinput,ATRwave2cep,ATRcep2para,ATRviterbi,ATRsendans,ATRr
esult NAME=$SPREC_BASE
```

```
unlimit
limit coredumpsize 0

set ResultDir = $ResultTopDir/
mkdir -p $ResultDir

set endAcount = ${#waveName}
set Acount = 1
while ( $Acount <= $endAcount )

set FlistName = ${WaveFileListDir}/${waveName[$Acount]}.Flist
set ansFile = ${ansTopDir}/${waveName[$Acount]}.TRS

       ($SPREC_BASE
-config=${originCurrent}/{$name}config.jv9read_adapt.viterbi¥
                    -ATRinput:file_list=$FlistName ¥
                    -ATRsendans:answer=$ansFile ¥
                    ${ResultDir}/${waveName[$Acount]}.res ) >&!
                    ${ResultDir}/${waveName[$Acount]}.err


       @ Acount ++
end
```

付録１１　自動セグメンテーション結果の例(sample_result.res)

```
VERSION=ReleaseATRVITERBI 07r03 Linux
base=1.000100


comment=START-OF-FILE(/prj/slap2/project/ATRSpeechDB/B.16k/MHT/A/MH
TSDA01.AD.16k)

UTTERANCE=1
amname=[/home/slap/xhayaka/SPREC/MatrixJV5/AM.M.1000.3.bin]
utime=4.230000
abstime=0.000000
cputime=0.500000
NBEST=1

ORDER=1

#これ以降にセグメンテーションの結果が記述される。
#自動セグメンテーションを行う場合、WORDS と wordids は同じ音素列となる。
#divs には、"音素名"、"音韻区間長"、"該当区間の尤度"が出力されている。
#times には、正解音素列の音素境界時間が羅列されている。
#いずれの出力でも各セグメントは"/"で区切られている

WORDS=-/a/r/a/j/u/r/u/g/e/ng/zh/i/ts/u/o/-/s/u/b/e/t/e/zh/i/b/u/ng/
n/o/h/o/u/e/n/e/zh/i/m/a/g/e/t/a/n/o/d/a/-
wordids=-/a/r/a/j/u/r/u/g/e/ng/zh/i/ts/u/o/-/s/u/b/e/t/e/zh/i/b/u/n
g/n/o/h/o/u/e/n/e/zh/i/m/a/g/e/t/a/n/o/d/a/-
divs=-,0.230000,-777671.000000/a,0.100000,-2566682.000000/r,0.04000
0,-1160439.000000/a,0.050000,-1987505.000000/j,0.090000,-2708928.00
0000/u,0.040000,-1645106.000000/r,0.040000,-2129001.000000/u,0.0900
00,-2455701.000000/g,0.040000,-718960.000000/e,0.070000,-1003008.00
0000/ng,0.100000,-1453105.000000/zh,0.060000,-874834.000000/i,0.080
000,-503605.000000/ts,0.100000,-1144631.000000/u,0.120000,-1845532.
000000/o,0.200000,-2926127.000000/-,0.400000,-828162.000000/s,0.120
000,-1792853.000000/u,0.050000,-1555314.000000/b,0.040000,-1771786.
000000/e,0.080000,-2010629.000000/t,0.050000,-915654.000000/e,0.090
```

000,-1265049.000000/zh,0.080000,-765270.000000/i,0.060000,-1112991.
000000/b,0.040000,-324420.000000/u,0.060000,-1668699.000000/ng,0.10
0000,-1069671.000000/n,0.050000,-1257488.000000/o,0.060000,-1998722
.000000/h,0.080000,-2027567.000000/o,0.130000,-3801556.000000/u,0.0
50000,-3723498.000000/e,0.100000,-990246.000000/n,0.080000,-1136809
.000000/e,0.080000,-585326.000000/zh,0.060000,-244473.000000/i,0.05
0000,-586917.000000/m,0.060000,-1804237.000000/a,0.060000,-1353528.
000000/g,0.060000,-1306549.000000/e,0.070000,-1972733.000000/t,0.05
0000,-918060.000000/a,0.060000,-1438723.000000/n,0.050000,-932554.0
00000/o,0.070000,-2526427.000000/d,0.040000,-1763899.000000/a,0.110
000,-2107582.000000/-,0.240000,-305867.000000
times=0.000000/0.230000/0.330000/0.370000/0.420000/0.510000/0.55000
0/0.590000/0.680000/0.720000/0.790000/0.890000/0.950000/1.030000/1.
130000/1.250000/1.450000/1.850000/1.970000/2.020000/2.060000/2.1400
00/2.190000/2.280000/2.360000/2.420000/2.460000/2.520000/2.620000/2
.670000/2.730000/2.810000/2.940000/2.990000/3.090000/3.170000/3.250
000/3.310000/3.360000/3.420000/3.480000/3.540000/3.610000/3.660000/
3.720000/3.770000/3.840000/3.880000/3.990000 score=-73764094.000000
acoustic=-73764094.000000

comment=END-OF-FILE(/prj/slap2/project/ATRSpeechDB/B.16k/MHT/A/MHTS
DA01.AD.16k)

付録１２　sssdata 作成用コンフィギュアファイルの例(sssdata.config)

```
# sssdata 作成用コンフィギュアファイル
#I/Ocontrol config :
        I/Ocontrol:inputFormat=NoHeader
        I/Ocontrol:inputByteorder=BigEndian
        I/Ocontrol:inputFd=
        I/Ocontrol:inputParamType=short
        I/Ocontrol:inputParamSize=160
        I/Ocontrol:outputFormat=NULL
        I/Ocontrol:rpcNumber=2

#ATRwavecut config :
        ATRwavecut:TRS=
        ATRwavecut:SamplingFrequency=16000
        ATRwavecut:SpeakerName=MHT
        ATRwavecut:PausePeriod=NOT
        ATRwavecut:pause_symbol=-

#ATRwave2cep config :
        ATRwave2cep:Preemphasis=0.98
        ATRwave2cep:FrameLength=20
        ATRwave2cep:FrameShift=10
        ATRwave2cep:SamplingFrequency=16000.0
        ATRwave2cep:TimeWindow=hamming
        ATRwave2cep:LagWindowFactor=0.01
        ATRwave2cep:LpcOrder=16
        ATRwave2cep:CepstrumOrder=12
        ATRwave2cep:FrequencyWarping=mel
        ATRwave2cep:FilterBankOrder=20
        ATRwave2cep:CutoffLowFrequency=0
        ATRwave2cep:CutoffHighFrequency=8000.0
        ATRwave2cep:AnalysisType=fft

ATRwave2cep:MeanInFile=/home/slap/xhayaka/SPREC/CM/ATRMEAN/CM.MHT.m
ean
        ATRwave2cep:Subtract=cep
```

```
        ATRwave2cep:DebuggingLevel=0

#ATRcep2para config :
        ATRcep2para:CepstrumOrder=12
        ATRcep2para:LDA=
        ATRcep2para:DeltaCepstrumWindow=5
        ATRcep2para:deltaCepstrumPadding=zero
        ATRcep2para:rho=1.0
        ATRcep2para:OutputParameter=cep(12)+dpow+dcep(12)
        ATRcep2para:WindowType=rectangular

        ATRsssdata:outputFd=
        ATRsssdata:outputParamSize=25
        ATRsssdata:FrameShift=10
        ATRsssdata:pause_symbol=-
        ATRsssdata:speakerId=MHT
        ATRsssdata:sssHeaderSource=TRS
```

付録１３　sssdata 作成用スクリプト(mk_sssdata.csh)

```
#! /bin/csh -f
#sssdata を出すためのスクリプト

#usage--- 第一引数 TRS ファイル名リスト,  第二引数 話者 ID

#ATRexec は sssdata 作成用実行ファイル
#config で sssdata 作成用コンフィギュレーションファイルを指定
#拡張子(.AD.16k)を除いた音声ファイル名と、拡張子(.TRS)を除いた TRS ファイル名が
同名のとき、音声と TRS ファイルの中身が対応していることを前提としたスクリプトであ
る
#

set fl = $argv[1]
set list = `cat $fl`
set name = $argv[2]

# loop
foreach file ($list)
        set input = "$file:r".AD.16k
        set trs = "$file"
        set output = "$file:r".sss

        ./ATRexec
        -config=JV9{$name}config_emb_sss
        -I/Ocontrol:inputFd=/prj/slap2/project/ATRSpeechDB/B.16k/$na
        me/$input
        -ATRwavecut:TRS=/home/slap/xhayaka/SPREC/TRS/$name/$trs
        -ATRsssdata:outputFd=$output

end
```

付録１４　自動セグメンテーション結果から音素列と音素境界部の時間情報を取り出すためのスクリプト(resultprs1.pl)

```
#!/usr/bin/perl
#自動セグメンテーション結果ファイルから音素、時間情報抽出

#usage--- 第一引数 自動セグメンテーション結果ファイルのリストファイル名

$flist = $ARGV[0];
open IN1,"$flist";
@files = <IN1>;

foreach $infile (@files){

    open IN2,"$infile";
    $i = 1;

    while(!eof IN2){

        $temp = $infile;
        if($i<=9){
            $temp =~ s/¥.res/_0$i¥.res/;
        }else{
            $temp =~ s/¥.res/_$i¥.res/;
        }

        $outfile = $temp;
        open OUT,">$outfile";

        while ($input = <IN2>){
            if($input =~ /^ORDER/){
                @RESULTS = split(/ /,$input);

                @onso = split(/¥//,$RESULTS[1]);

                @times = split(/¥//,$RESULTS[4]);
```

```
            for($j=1;$j<=$#onso;$j++){
                $anstime = $times[$j]*1000;
                print OUT $onso[$j]." ".$anstime;
                print OUT "¥n";
            }
            last;
        }
    }
    close OUT;
    $input = <IN2>;
    $input = <IN2>;
    $input = <IN2>;
    $input = <IN2>;
    $input = <IN2>;

    $i++;
  }

  close IN2;
}

close IN1;
```

付録１５　取り出した音素列と時間情報の例(res1out)

```
a 230
r 330
a 370
j 420
u 510
r 550
u 590
g 680
e 720
ng 790
zh 890
i 950
ts 1030
u 1130
o 1250
- 1450
s 1850
u 1970
b 2020
e 2060
t 2140
e 2190
zh 2280
i 2360
b 2420
u 2460
ng 2520
n 2620
o 2670
h 2730
o 2810
u 2940
e 2990
n 3090
e 3170
```

zh 3250

i 3310

m 3360

a 3420

g 3480

e 3540

t 3610

a 3660

n 3720

o 3770

d 3840

a 3880

- 3990

付録１６　音素境界部の同定用スクリプト（resultprs2.pl）

```
#!/usr/bin/perl
#ラベルファイルから第一層を選択
#resultprs1.pl 出力とラベルファイルとの比較結果ファイル作成

#usage--- 第一引数 resultprs1.pl 出力結果ファイルのリストファイル名
#usage--- 第二引数 ラベルファイルのリストファイル名
#usage--- 各リスト内で出力結果とラベルが順番に対応している必要がある

#---出力されるファイルの拡張子は.end

$flist1 = $ARGV[0];
$flist2 = $ARGV[1];
open IN1,"$flist1";
open IN2,"$flist2";
@resfiles = <IN1>;

foreach $inresfile (@resfiles){

   open IN3,"$inresfile";

   $inLBfile = <IN2>;
   open IN4,"$inLBfile";

   $output = $inresfile;
   $output =~ s/res/end/;
   open OUT,">$output";

   print OUT "  res ans¥n";

   @array1 = <IN3>;
   @array2 = <IN4>;

   $array1counter = 0;

   for($i=0;$i<=$#array1;$i++){
```

```
        substr($array1[$i],-1,1) = " ";
    }

    for($i=0 ;$i<=$#array2 ;$i++){

        $nextarray1cnt = 0;

        if($array2[$i] == "#"){
            @lasttime = split(/ /,$array2[$i-1]);
            $array1[$array1counter]=
$array1[$array1counter].$lasttime[2];
            last;
        }

        @onso2 = split(/ /,$array2[$i]);

        if($onso2[1] =~ /(.)¥1/){
            print "A";
            $nextarray1cnt++;
        }

        while($onso2[1] =~ /,/g ){
            print "B";
            $nextarray1cnt++;
        }

        while($onso2[1] =~
/sh,u|ch,u|zh,u|ky,u|ny,u|hy,u|my,u|ry,u|gy,u|dy,u|by,u|py,u|j,u/g)
{
            print "C";
            $nextarray1cnt++;
        }

        @onso1 = split(/ /,$array1[$array1counter]);
        @preonso1 = split(/ /,$array1[$array1counter-1]);
```

```
        if($onso1[0] eq "j"){
            if($preonso1[0] =~ /sh|ch|zh|k|n|h|m|r|g|d|b|p/){
                if($preonso1[0] ne "ng"){
                    $array1counter++;
                }
            }
        }

        @Aarray = split(/ /,$array2[$i]);
        $array1[$array1counter]= $array1[$array1counter].$Aarray[0]."
".$Aarray[1];
        $array1counter++;
        $array1counter += $nextarray1cnt;
    }

    for($i=0;$i<=$#array1;$i++){
        print OUT $array1[$i];
        print OUT "¥n";
    }

    close OUT;
    close IN3;
    close IN4;

}

close IN1;
close IN2;
```

付録１７　音素境界部同定後の例(前二列が自動セグメンテーションの結果、後２列がハンドラベリング結果、res2out)

```
 res ans
a 230 255.0 a
r 330 340.0 r
a 370 350.0 a
j 420 445.0 y
u 510 530.0 u
r 550 550.0 r
u 590 570.0 u
g 680 665.0 g
e 720 720.0 e
ng 790 805.0 N,j
zh 890
i 950 950.0 i
ts 1030 1025.0 ts
u 1130 1150.0 u
o 1250 1240.0 wo
- 1450 1417.5 pau
s 1850 1865.0 s
u 1970 1982.5 u
b 2020 2025.0 b
e 2060 2035.0 e
t 2140 2135.0 t
e 2190 2190.0 e
zh 2280 2265.0 j
i 2360 2375.0 i
b 2420 2415.0 b
u 2460 2470.0 u
ng 2520 2530.0 N,n
n 2620
o 2670 2650.0 o
h 2730 2725.0 h
o 2810 2815.0 o,u
u 2940
e 2990 2965.0 he
```

```
n 3090 3100.0 n
e 3170 3160.0 e
zh 3250 3270.0 j
i 3310 3325.0 i
m 3360 3355.0 m
a 3420 3415.0 a
g 3480 3480.0 g
e 3540 3515.0 e
t 3610 3605.0 t
a 3660 3655.0 a
n 3720 3725.0 n
o 3770 3750.0 o
d 3840 3830.0 d
a 3880 3880.0 a
- 3990 3985.0
```

付録１８　絶対平均値、標準偏差、誤差が±30 ms、±50 ms 以内に含まれる境界の比率を得るためのスクリプト(resultprs3.pl)

```
#!/usr/bin/perl
#resultprs2.pl の出力結果から統計値を出すためのスクリプト

#usage--- 第 1 引数 resultprs2.pl 出力結果のリストファイル名
#usage--- 第 2 引数 結果ファイル名 (任意の名前)

$flist = $ARGV[0];
open IN,"$flist";
@files = <IN>;

$alldiff = 0;
$count = 0;
$Tcounter = 0;
$Fcounter = 0;

foreach $file (@files){
    open INF1,"$file";

    $input = <INF1>;

    while ($input = <INF1>){
        @results = split (/ /,$input);
        if($results[2] =~ /[0-9]/){
            $alldiff += $results[1] - $results[2];
            $absdiff += abs($results[1] - $results[2]);
            $count++;
            if(abs($results[1] - $results[2]) <= 30){
                $Tcounter ++;
            }
            if(abs($results[1] - $results[2]) <= 50){
                $Fcounter ++;
            }
        }
    }
```

```
    close INF1;
}

close IN;

$file2 = $ARGV[1];
open OUT,">$file2";

#-絶対平均誤差

$absavg = $absdiff / $count;
$avg = $alldiff / $count;
print OUT "absaverage  ";
print OUT $absavg;
print OUT "¥n";

#-平均誤差

print OUT "average  ";
print OUT $avg;
print OUT "¥n";

#- +-30ms,+-50ms 精度
$Trate = ($Tcounter / $count)*100;
$Frate = ($Fcounter / $count)*100;
print OUT "30m  ";
print OUT $Trate;
print OUT "¥n";
print OUT "50m  ";
print OUT $Frate;
print OUT "¥n";

#-標準偏差
$temp = 0;
foreach $file (@files){
```

```
    open INF2,"$file";

    $input = <INF2>;

    while ($input = <INF2>){
        @results = split (/ /,$input);
        if($results[2] =~ /[0-9]/){
            $temp += ($results[1] - $results[2] - $avg)**2;
        }
    }

    close INF2;
}
```

付録１９　出力される絶対平均値、標準偏差、誤差が±30 ms、±50 ms 以内に含まれる境界の比率の例(res3out)

```
absaverage  12.600506338752
average  2.68483267978833
30m  94.6663113412266
50m  99.2271671679293
standard deviation  16.53651352310
```

付録２０　ハンドラベリング側の音素それぞれについて多様な情報を出力するためのスクリプト(resultprs４.pl)

```
#!/usr/bin/perl
#resultprs2.pl の出力を使用し、ハンドラベリング側の音素それぞれについて
#当該音素
#先行音素
#後続音素
#開始時刻
#終了時刻
#呼気段落長(音素数モーラ数)
#先頭からの呼気段落内位置(音素数モーラ数)
#末尾からの呼気段落内位置(音素数モーラ数)
#当該音素属性(母音子音、有声無声、調音様式、調音位置)
#先行音素属性(母音子音、有声無声、調音様式、調音位置)
#後続音素属性(母音子音、有声無声、調音様式、調音位置)
#開始推定時刻
#終了推定時刻
#開始時間誤差
#終了時間誤差
#音韻長誤差
#オーバーラップ率(推定時間長に対する)
#を"/"区切りで出力するスクリプト

#usage--- 第一引数 resultprs2.pl 出力ファイルのリストファイル名
#usage--- 第二引数 結果ファイル名(任意)

$flist = $ARGV[0];
open IN,"$flist";
@files = <IN>;

$OUTPUT = $ARGV[1];
open OUT,">$OUTPUT";

#end ファイルから正解音素列のみ取り出す
foreach $file ( @files ){
    open INF,"$file";
```

```
    @array = <INF>;
    for($i=1; $i<=$#array-1; $i++){
        @results = split (/ /,$array[$i]);
        if($#results == 3){
            @Ansarray = (@Ansarray,$array[$i]);
        }
        if($i == $#array-1){
            @Ansarray = (@Ansarray,$array[$i]);
        }
    }
    close INF;
    @Ansarray = (@Ansarray,"#¥n");
}

#音素カウンター
$OCT = 1;
#モーラカウンター
$MCT = 0;
$MCTBACK = 0;

#呼気用
$kokiCNT = 0;
$moraCNT = 0;

#統計用データを取り出す
for($i=0; $i<=$#Ansarray; $i++){

#次の文へ
    if($Ansarray[$i+1] eq "#¥n"){
        $i = $i + 1;
        $OCT = 1;
        $MCT = 0;
        $kokiCNT = 0;
        $moraCNT = 0;
        next;
    }
```

#pau 音素位置

```
    if($i == 0){
        @kokiiti = (0);
        $k = 0;
        while(1){
            @kokiyou = split(/ /,$Ansarray[$k]);
            if($kokiyou[3] eq "pau¥n"){
                @kokiiti = (@kokiiti,$k+1);
            }elsif($kokiyou[0] eq "#¥n"){
                @kokiiti = (@kokiiti,$k);
                $k++;
                $temp = $k;
                last;
            }
            $k++;
        }
    }elsif($Ansarray[$i-1] eq "#¥n"){
        $k = $temp;
        @kokiiti = ($k);
        while(1){
            @kokiyou = split(/ /,$Ansarray[$k]);
            if($kokiyou[3] eq "pau¥n"){
                @kokiiti = (@kokiiti,$k+1);
            }elsif($kokiyou[0] eq "#¥n"){
                @kokiiti = (@kokiiti,$k);
                $k++;
                $temp = $k;
                last;
            }
            $k++;
        }
    }
```

#pau モーラ位置

```
    if($i == 0){
```

```
    $k = 0;
    @moraiti = (0);
    $moracount = 0;
    while(1){
        @morayou = split(/ /,$Ansarray[$k]);
        @tenkiri = split(/,/,$morayou[3]);
        if($morayou[3] ne "pau¥n"){
            for($s=0;$s<=$#tenkiri;$s++){
                if($tenkiri[$s] =~ /[aiueo]/){
                    $moracount++;
                }elsif($tenkiri[$s] =~ /(.)¥1/){
                    $moracount++;
                }elsif($tenkiri[$s] =~ "N"){
                    $moracount++;
                }
            }
        }

        if($morayou[3] eq "pau¥n"){
            $moracount++;
            @moraiti = (@moraiti,$moracount);
        }elsif($morayou[0] eq "#¥n"){
            $moracount++;
            @moraiti = (@moraiti,$moracount);
            $k++;
            $temp2 = $k;
            last;
        }
        $k++;
    }
}elsif($Ansarray[$i-1] eq "#¥n"){
    $k = $temp2;
    @moraiti = (0);
    $moracount = 0;
    while(1){
        @morayou = split(/ /,$Ansarray[$k]);
```

```
            @tenkiri = split(/,/,$morayou[3]);
            if($morayou[3] ne "pau¥n"){
                for($s=0;$s<=$#tenkiri;$s++){
                    if($tenkiri[$s] =~ /[aiueo]/){
                        $moracount++;
                    }elsif($tenkiri[$s] =~ /(.)¥1/){
                        $moracount++;
                    }elsif($tenkiri[$s] =~ /N/){
                        $moracount++;
                    }
                }
            }

            if($morayou[3] eq "pau¥n"){
                $moracount++;
                @moraiti = (@moraiti,$moracount);
            }elsif($morayou[0] eq "#¥n"){
                $moracount++;
                @moraiti = (@moraiti,$moracount);
                $k++;
                $temp2 = $k;
                last;
            }
            $k++;
        }
    }

#前後結果取り出し
    @preresults = split (/ /,$Ansarray[$i-1]);
    @results = split (/ /,$Ansarray[$i]);
    @postresults = split (/ /,$Ansarray[$i+1]);
    if($Ansarray[$i+2] == "#¥n"){
        @ppostresults = "-";
    }elsif($Ansarray[$i+3] == "#¥n"){
        @ppostresults = "-";
    }else{
```

```
        @postresults = split (/ /,$input[$i+2]);
    }
    substr($preresults[3],-1,1) = "";
    substr($results[3],-1,1) = "";
    substr($postresults[3],-1,1) = "";
    substr($ppostresults[3],-1,1) = "";

#モーラカウンター
    @MCTtenkiri = split(/,/,$results[3]);
    if($results[3] ne "pau¥n"){
        $MCTBACK = $MCT;
        for($s=0;$s<=$#MCTtenkiri;$s++){
            if($MCTtenkiri[$s] =~ /[aiueo]/){
                $MCTBACK++;
            }elsif($MCTtenkiri[$s] =~ /(.)¥1/){
                $MCTBACK++;
            }elsif($MCTtenkiri[$s] =~ /N/){
                $MCTBACK++;
            }
        }
    }

#当該音
    $onso = $results[3];

#先行音素
    if($i == 0){
        $preonso = "-";
    }elsif($Ansarray[$i-1] eq "#¥n"){
        $preonso = "-";
    }else{
        $preonso = $preresults[3];
    }

#後続音素
    if($Ansarray[$i+2] eq "#¥n"){
```

```
        $postonso = "-";
    }else{
        $postonso = $postresults[3];
    }

#開始時刻
    $rstarttime = $results[2];

#終了時刻
    if($Ansarray[$i+2] eq "#¥n"){
        substr($postresults[2],-1,1) = "";
    }
    $rendtime = $postresults[2];

#ポーズが来たら次の呼気段落へ
    if( $results[3] eq "pau"){
        $kokiCNT++;
        $moraCNT++;
        $OCT = 0;
        $MCT = 0;
        $MCTBACK = 0;
    }

#呼気段落長
#音素数
    if( $results[3] eq "pau"){
        $KOKIONSOSU = "-";
    }else{
        $KOKIONSOSU = $kokiiti[$kokiCNT+1] - $kokiiti[$kokiCNT] - 1;
    }
#モーラ数
    if( $results[3] eq "pau"){
        $KOKIMORASU = "-";
    }else{
        $KOKIMORASU = $moraiti[$moraCNT+1] - $moraiti[$moraCNT] - 1;
    }
```

```
#呼気段落内位置(先頭から)
#音素数
    if( $results[3] eq "pau"){
        $FRONTONSO = "-";
    }else{
        $FRONTONSO = $OCT;
    }

#モーラ数
    if( $results[3] eq "pau"){
        $FRONTMORA = "-";
    }else{
        $FRONTMORA = $MCT + 1;
    }

#呼気段落内位置(末尾から)
#音素数
    if( $results[3] eq "pau"){
        $BACKONSO = "-";
    }else{
        $BACKONSO = $kokiiti[$kokiCNT+1] - $kokiiti[$kokiCNT] - $OCT;
    }

#モーラ数
    if( $results[3] eq "pau"){
        $BACKMORA = "-";
    }elsif($MCT == $MCTBACK){
        $BACKMORA = $moraiti[$moraCNT+1] - $moraiti[$moraCNT] -
$MCTBACK - 1;
    }else{
        $BACKMORA = $moraiti[$moraCNT+1] - $moraiti[$moraCNT] -
$MCTBACK;
    }

#当該音素属性
#母音子音
```

```
    $bosi = &bosicheck($results[3],$postresults[3]);
#有声無声
    $umu = &umucheck($results[3],$results[0]);
#調音様式
    $yoshiki = &yoshikicheck($results[3],$results[0]);
#調音位置
    $ichi = &ichicheck($results[3],$postresults[3],$results[0]);

#先行音素属性
#母音子音
    if($i == 0){
        $prebosi = "-";
    }elsif($Ansarray[$i-1] eq "#¥n"){
        $prebosi = "-";
    }else{
        $prebosi = &bosicheck($preresults[3],$results[3]);
    }
#有声無声
    $preumu = &umucheck($preresults[3],$preresults[0]);
#調様式
    $preyoshiki = &yoshikicheck($preresults[3],$preresults[0]);
#調音位置
    $preichi =
&ichicheck($preresults[3],$results[3],$preresults[0]);

#後続音素属性
#母音子音
    if($i == $#Ansarray){
        $postbosi = "-";
    }elsif($Ansarray[$i+2] eq "#¥n"){
        $postbosi = "-";
    }else{
        $postbosi = &bosicheck($postresults[3],$ppostresults[3]);
    }
#有声無声
    $postumu = &umucheck($postresults[3],$postresults[0]);
```

```
#調音様式
    $postyoshiki = &yoshikicheck($postresults[3],$postresults[0]);
#調音位
    $postichi =
&ichicheck($postresults[3],$ppostresults[3],$postresults[0]);

#開始時刻推定
    $sstarttime = $results[1];

#終了時刻推定値
    $sendtime = $postresults[1];

#開始時刻誤差
    $startdiff = $sstarttime - $rstarttime;

#終了時刻誤差
    $enddiff = $sendtime - $rendtime;

#音素長誤差
    $lengthdiff = ($sendtime - $sstarttime) - ($rendtime - $rstarttime);

#オーバーラップ率
    if($sendtime <= $rstarttime){
        $overlap = 0;
    }elsif(($rendtime >= $sendtime) && ($sendtime >= $rstarttime)){
        if($sstarttime <= $rstarttime){
            $overlap = ($sendtime - $rstarttime)/($sendtime -
                        $sstarttime);
        }else{
            $overlap = 1;
        }
    }else{
        if($sstarttime <= $rstarttime){
            $overlap = ($rendtime - $rstarttime)/($sendtime -
                        $sstarttime);
        }elsif(($rendtime >= $sstarttime) && ($sstarttime >=
```

```
    $rstarttime)){
        $overlap = ($rendtime - $sstarttime)/($sendtime -
                    $sstarttime);
    }else{
        $overlap = 0;
    }
}

print OUT $onso."/";
print OUT $preonso."/";
print OUT $postonso."/";
print OUT $rstarttime."/";
print OUT $rendtime."/";
print OUT $KOKIONSOSU."/";
print OUT $KOKIMORASU."/";
print OUT $FRONTONSO."/";
print OUT $FRONTMORA."/";
print OUT $BACKONSO."/";
print OUT $BACKMORA."/";
print OUT $bosi."/";
print OUT $umu."/";
print OUT $yoshiki."/";
print OUT $ichi."/";
print OUT $prebosi."/";
print OUT $preumu."/";
print OUT $preyoshiki."/";
print OUT $preichi."/";
print OUT $postbosi."/";
print OUT $postumu."/";
print OUT $postyoshiki."/";
print OUT $postichi."/";
print OUT $sstarttime."/";
print OUT $sendtime."/";
print OUT $startdiff."/";
print OUT $enddiff."/";
print OUT $lengthdiff."/";
```

```
    print OUT $overlap."/";
    print OUT "¥n";

#モーラカウンター増
    $MCT = $MCTBACK;
#音素カウンター
    $OCT++;

    close INF;
}

sub bosicheck{
    @subbositenkiri = split(/,/,$_[0]);

    if($#subbositenkiri == 1){
        if($_[0] =~ /N¥,m|N¥,n/){
            $subbosi = "mora_nasal+";
        }elsif($_[0] =~
/ch¥,i|sh¥,i|k¥,i|h¥,i|f¥,u|ts¥,u|s¥,u|k¥,u/){
            if($_[0] =~ /(.)¥1/){
                $subbosi = "geminate_consonant+devoiced_vowel";
            }else{
                $subbosi = "devoiced_vowel";
            }
        }elsif($_[0] =~ /([aiueo])¥,¥1/ || $_[0] =~ /o¥,u|e¥,i/){
            $subbosi = "long_vowel";
        }elsif($_[0] =~ /sh¥,u/ ){
            if($_[0] =~ /(.)¥1/){
                $subbosi = "geminate_consonant+devoiced_vowel";
            }else{
                $subbosi = "platalized_consonant+devoiced_vowel";
            }
        }else{
            $subbosi = "cluster";
        }
    }elsif($#subbositenkiri == 0){
```

```
        if($_[0] eq "pau"){
            $subbosi = "-";
        }elsif($_[0] =~ /[aiueo]/){
            $subbosi = "vowel";
        }elsif($_[0] =~ /N/){
            $subbosi = "mora_nasal";
        }elsif($_[0] =~ /(.)¥1/){
            if($_[0] =~ /(.)¥1y/){
                $subbosi = "geminate_consonant+platalized_consonant";
            }elsif($_[0] =~ /sh|ch/){
                if($_[1] =~ /[auo]/){
                    $subbosi =
                    "geminate_consonant+platalized_consonant";
                }else{
                    $subbosi = "geminate_consonant";
                }
            }else{
                    $subbosi = "geminate_consonant";
            }
        }elsif($_[0] =~ /ky|hy|ry|gy|py|my|ny|dy|by/){
            $subbosi = "platalized_consonant";
        }elsif($_[0] =~ /sh|ch|zh|j/){
            if($_[1] =~ /[auo]/){
                $subbosi = "platalized_consonant";
            }
        }else{
            $subbosi = "consonant";
        }
    }else{
        $subbosi = "cluster";
    }

    return $subbosi;
}

sub umucheck {
```

```
    @subumutenkiri = split(/,/,$_[0]);

    if($#subumutenkiri == 1){
        if($_[0] =~ /N¥,m|N¥,n/ || $_[0] =~ /([aiueo])¥,¥1/ || $_[0] =~
/o¥,u|e¥,i/){
            $subumu = "voiced";
        }elsif($_[0] =~
/ch¥,i|sh¥,i|k¥,i|h¥,i|f¥,u|ts¥,u|s¥,u|k¥,u|sh¥,u/){
            $subumu = "voiceless";
        }else{
            $subumu = "-";
        }
    }elsif($#subumutenkiri == 0){
        if($_[0] =~ /pau/){
            $subumu = "-";
        }elsif($_[0] =~ /(.)¥1/ || $_[0] =~ /[ptkfs]|sh|ch|ts/){
            $subumu = "voiceless";
        }elsif($_[0] =~ /h/){
            if($_[1] =~ /w/){
                $subumu = "voiced";
            }else{
                $subumu = "voiceless";
            }
        }elsif($_[0] =~ /[aiueobdgzmnrwjyN]/){
            $subumu = "voiced";
        }
    }else{
        $subumu = "-";
    }
    return $subumu;
}

sub yoshikicheck{
    @subyoshikitenkiri = split(/,/,$_[0]);
    if($#subyoshikitenkiri == 1){
        if($_[0] =~ /N¥,m|N¥,n/){
```

```
            $subyoshiki = "nasal";
        }elsif($_[0] =~
/ch¥,i|sh¥,i|k¥,i|h¥,i|f¥,u|ts¥,u|s¥,u|k¥,u|sh¥,u/){
            if($_[0] =~ /ch|ts/){
                $subyoshiki = "affricate";
            }elsif($_[0] =~ /sh|f|h/){
                $subyoshiki = "fricative";
            }elsif($_[0] =~ /k/){
                $subyoshiki = "plosive";
            }
        }elsif($_[0] =~ /([aiueo])¥,¥1/ || $_[0] =~ /o¥,u|e¥,i/){
            $subyoshiki = "verbal";
        }else{
            $subyoshiki = "-";
        }
    }elsif($#subyoshikitenkiri == 0){
        if($_[0] =~ /pau/){
            $subyoshiki = "-";
        }elsif($_[0] =~ /ch|ts|zh|z|j/){
            $subyoshiki = "affricate";
        }elsif($_[0] =~ /[pbtdkg]/){
            $subyoshiki = "plosive";
        }elsif($_[0] =~ /ch|ts|zh|z/){
            $subyoshiki = "affricate";
        }elsif($_[0] =~ /[fs]|sh/){
            $subyoshiki = "fricative";
        }elsif($_[0] =~ /h/ && $_[0] ne "he" ){
            if($_[1] =~ /w/){
                $subyoshiki = "approximant";
            }else{
                $subyoshiki = "fricative";
            }
        }elsif($_[0] =~ /[mnN]/){
            $subyoshiki = "nasal";
        }elsif($_[0] =~ /r/){
            $subyoshiki = "flap";
```

```
        }elsif($_[0] =~ /[wy]/ && $_[0] ne "wo" ){
            $subyoshiki = "approximant";
        }elsif($_[0] =~ /[aiueo]/){
            $subyoshiki = "verbal";
        }
    }else{
        $subyoshiki = "-";
    }

   return $subyoshiki;

}

sub ichicheck{
    @subichitenkiri = split(/,/,$_[0]);

    if($#subichitenkiri == 1){
        if($_[0] =~ /N,m/){
            $subichi = "labial";
        }elsif($_[0] =~ /N,n/){
            $subichi = "alveolar";
        }elsif($_[0] =~
                /ch¥,i|sh¥,i|k¥,i|h¥,i|f¥,u|ts¥,u|s¥,u|k¥,u|sh,u/){
            $subichi = &ichicheck($subichitenkiri[0]);
        }elsif($_[0] =~ /([aiueo])¥,¥1/ || $_[0] =~ /o¥,u|e¥,i/){
            $subichi = &ichicheck($subichitenkiri[0]);
        }else{
            $subichi = "-";
        }
    }elsif($#subichitenkiri == 0){
        if($_[0] =~ /pau/){
            $subichi = "-";
        }elsif($_[0] =~ /[pbfmw]/ && $_[0] ne "wo"){
            $subichi = "labial";
        }elsif($_[0] =~ /ch|zh|sh|j/){
            $subichi = "palatal";
```

```
    }elsif($_[0] =~ /[kg]/){
        $subichi = "velar";
    }elsif($_[0] =~ /h/ && $_[0] ne "he"){
        if($_[2] =~ /w/){
            $subichi = "labial";
        }else{
            $subichi = "glottal";
        }
    }elsif($_[0] =~ /N/){
        if($_[1] =~ /[pbm]/){
            $subichi = "labial";
        }elsif($_[1] =~ /[tdn]|ch|ts/){
            $subichi = "alveolar";
        }else{
            $subichi = "uvular";
        }
    }elsif($_[0] =~ /[tdnrzs]|ts/){
        $subichi = "alveolar";
    }elsif($_[0] =~ /y/){
        $subichi = "palatal";
    }elsif($_[0] =~ /a/){
        $subichi = "mid_low";
    }elsif($_[0] =~ /i/){
        $subichi = "front_high";
    }elsif($_[0] =~ /u/){
        $subichi = "back_high";
    }elsif($_[0] =~ /e/){
        $subichi = "front_mid";
    }elsif($_[0] =~ /o/){
        $subichi = "back_mid";
    }else{
        $subichi = "-";
    }
}else{
    $subichi = "-";
}
```

```
    return $subichi;
}

close IN;
close OUT;
```

付録２１　出力される音素情報の例（res4out）

#順に、
#当該音素
#先行音素
#後続音素
#開始時刻
#終了時刻
#呼気段落長(音素数モーラ数)
#先頭からの呼気段落内位置(音素数モーラ数)
#末尾からの呼気段落内位置(音素数モーラ数)
#当該音素属性(母音子音、有声無声、調音様式、調音位置)
#先行音素属性(母音子音、有声無声、調音様式、調音位置)
#後続音素属性(母音子音、有声無声、調音様式、調音位置)
#開始推定時刻
#終了推定時刻
#開始時間誤差
#終了時間誤差
#音韻長誤差
#オーバーラップ率(推定時間長に対する)
#を表す。
#各情報は"/"で区切られており，当該音素毎に改行される。

```
a/-/r/255.0/340.0/14/9/1/1/14/9/vowel/voiced/verbal/mid_low/-/-/-/-
/consonant/voiced/flap/alveolar/230/330/-25/-10/15/0.75/
r/a/a/340.0/350.0/14/9/2/2/13/8/consonant/voiced/flap/alveolar/vowe
l/voiced/verbal/mid_low/vowel/voiced/verbal/mid_low/330/370/-10/20/
30/0.25/
a/r/y/350.0/445.0/14/9/3/2/12/8/vowel/voiced/verbal/mid_low/consona
nt/voiced/flap/alveolar/consonant/voiced/approximant/palatal/370/42
0/20/-25/-45/1/
y/a/u/445.0/530.0/14/9/4/3/11/7/consonant/voiced/approximant/palata
l/vowel/voiced/verbal/mid_low/vowel/voiced/verbal/back_high/420/510
/-25/-20/5/0.722222222222222/
u/y/r/530.0/550.0/14/9/5/3/10/7/vowel/voiced/verbal/back_high/conso
nant/voiced/approximant/palatal/consonant/voiced/flap/alveolar/510/
550/-20/0/20/0.5/
```

r/u/u/550.0/570.0/14/9/6/4/9/6/consonant/voiced/flap/alveolar/vowel
/voiced/verbal/back_high/vowel/voiced/verbal/back_high/550/590/0/20
/20/0.5/