TR-I-0339

SSS-LR 連続音声認識システム

— User's Guide —

永井 明人
Nagai Akito

ハラルド・シンガー
Harald Singer

1993.3

内容梗概

本稿は、ATR 自動翻訳電話実験システム ASURA の音声入力部である、話者適応型 SSS-LR 連続音声認識システムの利用解説書である。本システムは、音声のマイク入力から音声認識結果の出力まで、種々のプログラムから構成されており、それらのインストレーション、システムセットアップ、及び実際のデモの操作方法について説明する。また、SSS-LR を用いて音声認識の評価実験をしたいというユーザーのために、評価実験用の環境についても説明する。

目次

図目次

表目次

第 1 章

まえがき

ATR 自動翻訳電話研究所は、言語の壁をうち破る夢の自動翻訳電話を実現するために、プロジェクトの 7 年間で数多くの様々な研究を積み重ねてきた。そして本プロジェクトの終了にあたり、最終的な研究成果を集大成した日英独三ヶ国間の自動翻訳電話公開実験システム ASURA を構築した。本稿で解説する SSS-LR 連続音声認識システムは、ATR の音声認識研究の最終成果として ASURA に組み込まれた、高い認識性能を持つシステムである。

第 2章では、SSS-LR を構成する様々なプログラムのインストールの方法、コンパイル、及びシステムセットアップの方法について述べる。第 3章では、実際にマイクを利用して音声入力するデモの起動、及び操作方法について述べる。第 4章では、本システムを構成している各プログラムについて述べる。第 5章では、SSS-LR を評価実験システムとして使う場合の実験用シェルスクリプト群の説明を行ない、サンプルデータを用いて、簡単な認識実験ができる環境を提供する。付録として、各プログラムのソースファイルの所在と、評価システムで用いたシェルスクリプトについて簡単な説明を加え、さらに、12 個の DSP を用いる SRTP IOPBOX のインストレーションと、それに関係するプログラムについて簡単に説明する。

第 2 章

インストールの方法

1　テープからのダウンロード

```
$ cd <directory of your choice>
$ tar xv
```

本システムをインストールするためのディレクトリ (例えば、ATREUS) を作成し、そこへ移動して、tar コマンドにより必要なファイルを全てダウンロードする。約 10 〜 15 メガバイトのディスク容量が必要である。

2　セット・アップとコンパイル

```
$ cd ATREUS
$ setup.csh
```

setup.csh は、全てのプログラムをコンパイルし、実行形式ファイルを ATREUS/BINHP にインストールする。この setup.csh は、対話的に動作し、$HOME/.cshrc のサーチパス指定を書き換えても良いかどうかを聞いてくる。また、S-RTP IOPBOX をインストールするかどうか、及び DASBOX-12、DASBOX-16 のどちらが使用されるのかを聞いてくる。

もし、一つのモジュールのみ、例えば、LPC(プログラム WavePara34) のみを再コンパイルしたい場合、以下のようにする。

```
$ cd ATREUS/LPC
$ make
$ make install
```

3　Hardware and Software Constraints

DASBOX-12 (devicename /dev/dm0) または、DASBOX-16 (devicename /dev/das0) と接続された HP9000/7xx シリーズのワークステーション (OS: HP-UX 8.05) を用いる。

このデモには、X11R4 の Motif が使用される。ファイルは、/usr/lib/Motif1.1 及び /usr/lib/X11 (また、ある場合には /usr/lib/X11R4) が用いられる。

ソケット番号は、20000 以上が用いられる (ATREUS/DEMO/Exe/menunew.csh を参照)。

3.1 Fonts

このデモは、以下のフォント (scripts in directory ATREUS/DEMO/Exe) が使用される。著作権上の問題により、今回のリリースでは、大きな漢字フォントをサポートしない。この問題は、X11R5 を用いることにより解決される。

表 2.1: X Window fonts

fontname	script
j16.32x32	NewWordRegistration.csh
jeuc.16x32	NewWordRegistration.csh
-adobe-helvetica-bold-r-normal--34-240-100-100-p-182-iso8859-1	menunew.csh

3.2 Colours

以下の色が使用される。

表 2.2: X Window colours

rgb value	*name*
50 191 193	aquamarine
0 0 0	black
0 0 255	blue
95 158 160	cadetblue
100 149 237	cornflowerblue
0 255 255	cyan
0 255 255	cyan1
105 105 105	dimgray
0 255 0	green
224 255 255	lightcyan
255 182 193	lightpink
255 255 224	LightYellow
176 48 96	maroon
102 205 170	mediumaquamarine
238 232 170	palegoldenrod
175 238 238	paleturquoise
175 238 238	PaleTurquoise
221 160 221	plum
255 0 0	red
70 130 180	steelblue
208 32 144	violetred
255 255 255	white

これらは、/usr/lib/X11/rgb.txt に追加し、rgb プログラム (ATREUS/DEMO/ETC/rgb.txt を参照) により、動作を確認しなければならない。

第 3 章

実際のデモの動かし方

本章では、具体的なデモの操作方法について説明する。

1　起動の方法

以下の手順によりシステムを立ち上げて、デモ画面を表示させる。

1. デモ実行用ディレクトリに移動する。

   ```
   $ cd ATREUS/DEMO
   ```

2. システム立ち上げ用のシェルスクリプトの実行。

   ```
   $ DEMO
   ```

3. 話者選択ウインドウが表示される。ここで、

   (a) 既に話者適応をしているならば、自分の名前を選択する。

   (b) まだ話者適応をしていないならば、9ページを参照して、話者適応を行なう。

4. 図 3.10 のような、システムメニューが表示される。ここで、Recognition_only を選択する。12ページを参照。

5. 図 3.1のようなデモ画面が表示されたら、起動完了。

   ただし、図 3.1は音声が入力されて、認識結果が表示された状態である。音声を入力して、認識処理が終了したときの画面を、図 3.2に示す。

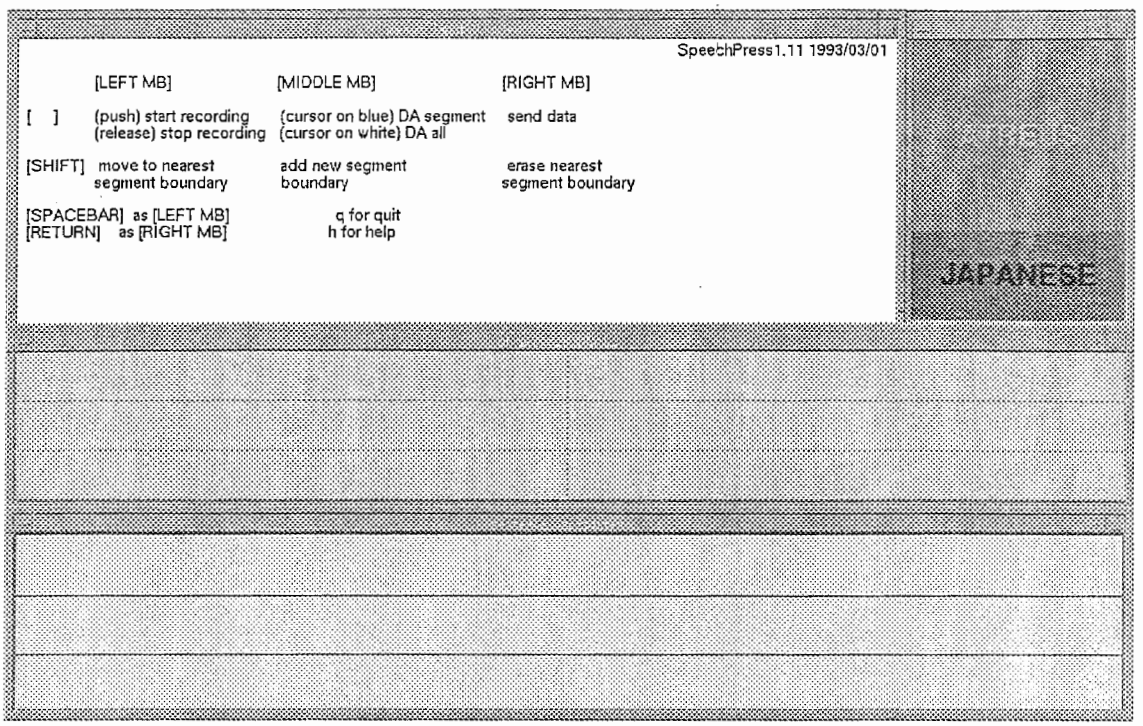

図 3.1: 全体の表示画面 (初期画面)

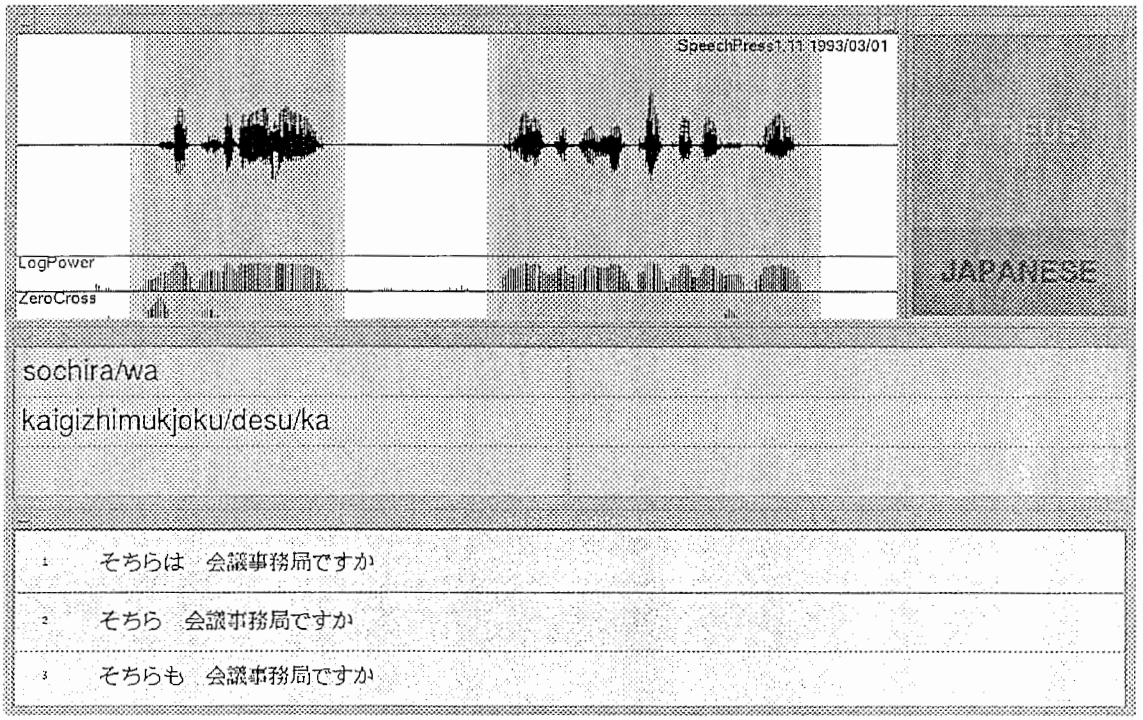

図 3.2: 全体の表示画面 (認識終了後)

2　SSS-LR 音声認識部

システムを立ち上げた後、以下の手順で音声認識のデモを行なう。

2.1　音声入力方法

発話する前に、カーソルが音声波形窓の中にあることを確認してください。マウス左ボタンを押しながら、発話を開始してください。文節間に適当なポーズを挿入しましょう。発話が終了したら、マウス左ボタンをはなしてください。平滑化された対数パワーとゼロ交差数をもとに、プログラムは音声区間境界を計算し、それを画面に表示します (図 3.3参照)。マウス右ボタンを押すと、切り出された音声データが標準出力に書き出され、音声認識処理が開始します。

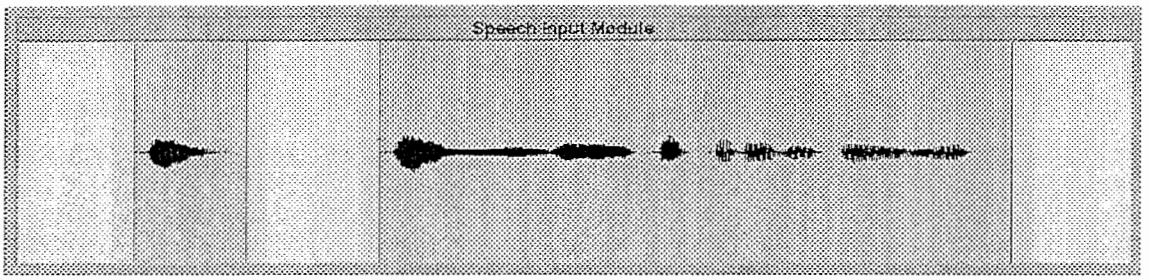

図 3.3: SpeechPress という音声入力窓の動作例。2 文節を発話した状態で、陰影の付いた部分が切り出された音声区間を表している。陰影の付いた部分は X 画面上では淡い青色。

発話している間、音声波形信号は音声波形窓で左から右に表示が進んでいきます。12.5 秒で、音声波形信号は画面の右端に到達し、画面が消去されます。音声波形信号は画面の左端に巻き取られ、再び左端から表示が進んでいきます。全体の音声の最大長は 50 秒に設定されています。

音声波形には、2 段階に陰影が施されています。濃い陰影は抽出アルゴリズムによって検出された音声区間を示しています。薄い陰影は安全のために追加された約 50 ms のマージン[1]を示しています。

誤った音声区間境界 (例えば、発話の最初の弱い /shi/ など) を修正するために、音声区間境界の移動 (シフト＋マウス左ボタン) や、削除 (シフト＋マウス右ボタン) ができます。新しい音声区間境界の追加 (シフト＋マウス中ボタン) もできます。キーボードから h を押すことにより、いつでもヘルプ画面が表示できます (図 3.4参照)。プログラムを終了するには、キーボードから q を入力してください。

[1]認識過程で、予測される文字列の最初と最後に HMM 無音モデルが連結されるため。

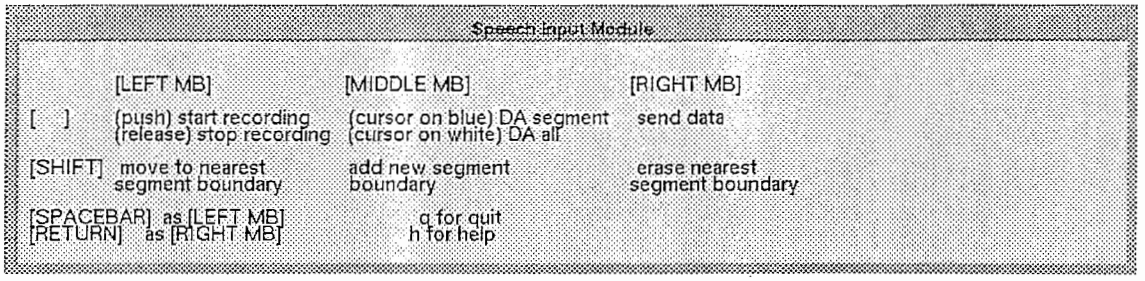

図 3.4: SpeechPress における開始およびヘルプ画面

2.2 文節音声認識窓とその操作方法

SpeechPress の音声入力窓でマウスの右ボタンをクリックすると、音声認識処理が開始され、認識文節表示ウインドウ (図 3.5) に、次々と認識した文節が表示される。ある文節の表示が終了した後、その文節の認識結果をマウス左ボタンでダブルクリックすると、認識候補ウインドウ (図 3.6) が表示される。このウインドウは、認識候補を確からしいものから順に、第五位まで表示する。もう一度、認識候補ウインドウ内でマウス左ボタンをダブルクリックすると、ウインドウが閉じる。

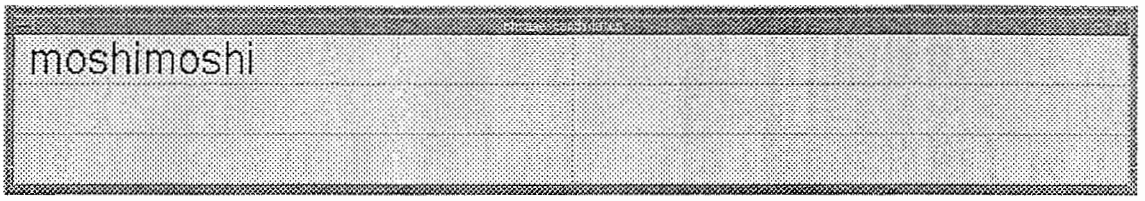

図 3.5: 認識文節表示ウインドウ

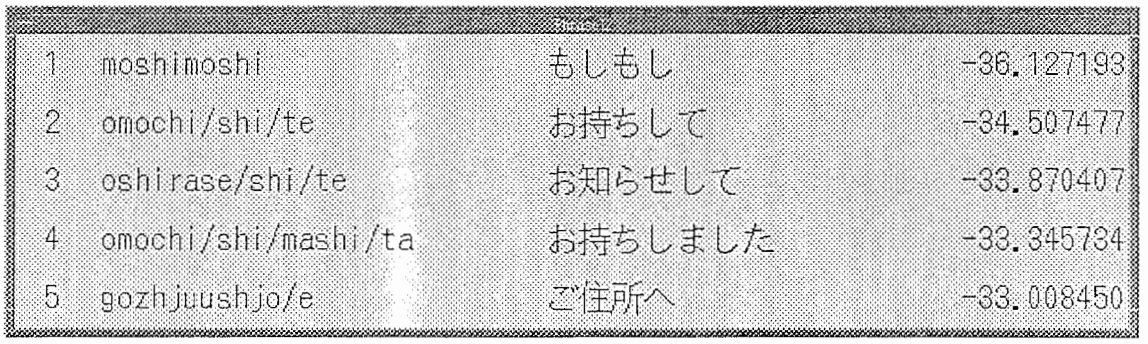

図 3.6: 認識候補ウインドウ

2.3 音声認識結果の文候補表示窓と、そこでの文候補選択方法

認識された文節候補を組み合わせて、文節間の文法にあっている文候補 (最高 10 個) を表示するウインドウが、図 3.7である。1 度に表示出来る文候補は最高 3 候補であり、ウインド

ウ上でマウス右ボタンをクリックすることにより、ページが進んで次の文候補が見られる。マウス操作は、次の通りである。

- マウスクリック方法
  - 左ボタン　文候補の選択 (選択された候補のウインドウの色が変わる)
  - 中ボタン　前ページ
  - 右ボタン　次ページ

ただし、文選択の左ボタンクリックは、選択した文候補を言語翻訳部へ渡す操作であり、本解説書の範囲外なので扱わない。

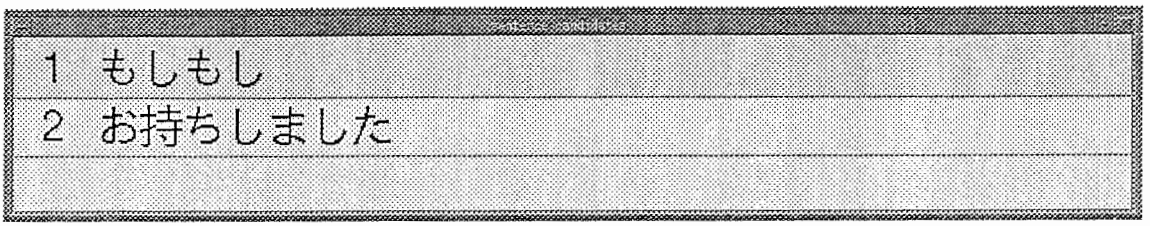

図 3.7: 音声認識結果の文候補表示選択ウインドウ

3 話者適応部

このシステムを始めて使用する際には、使用する方の声の特徴をシステムに登録し、その特徴に基づいて音声認識のためのモデルのパラメータを変更する必要があります。この一連の処理は「話者適応」と呼ばれています。

話者適応を行なうための第一ステップとして、まず使用者があらかじめ決められた内容の言葉(単語、あるいは文節など)を発声し、システムに登録します。

話者適応をするにはまず、次の一連の手順を行なってください。

```
$ cd DEMO/ATREUS
$ DEMO
```

ここまでの操作が終ると、画面上には図3.8のような話者選択画面が表示されます。

図 3.8: 話者選択画面

始めて登録する場合には、ここで :::NEW::: を選択します。すると、図3.9のような話者名入力画面が表示されます。話者適応をする人の名前を入力し、リターンキー を押します。その際、話者選択画面に既に存在する名前と重複しないように注意してください。

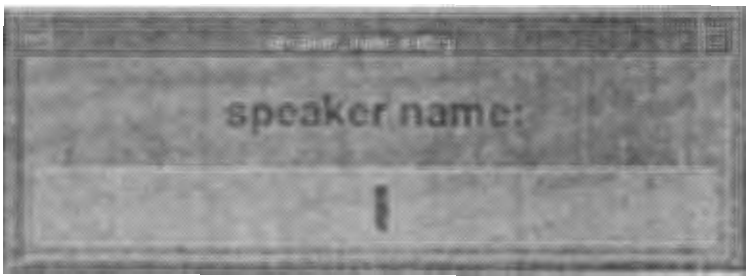

図 3.9: 話者名入力画面

動作環境が変化したりして、話者適応をし直す場合には、図3.8の話者選択画面に自分の名前が存在していますから、それをそのままマウスで選択してください。

ここまでの手順が終了すると、画面上には、図3.10のようなメニューが表示されています。このメニューで Voice_Registration を選択します。これによって、図3.11のようなメニューが表示されます。

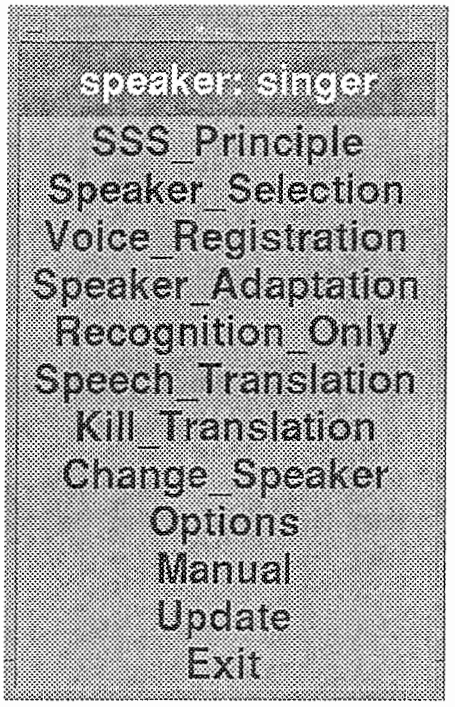

図 3.10: システムメニュー画面

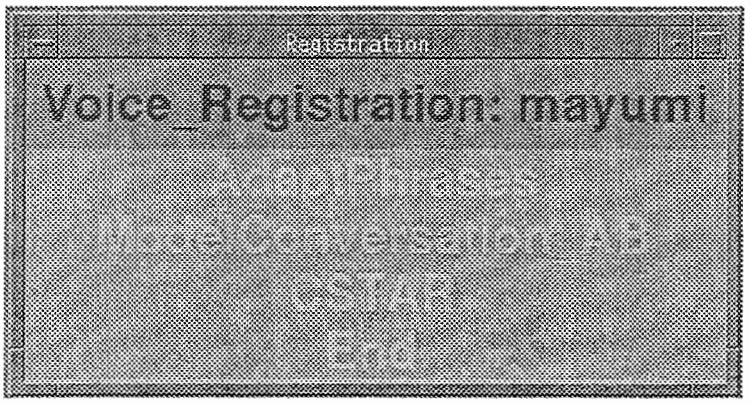

図 3.11: 話者適応メニュー画面

この時、図 3.10や図 3.11のメニューウインドウ上部に表示されている “Speaker:.....” の部分の話者名が自分の名前と一致していることを確認してください[2](他人の名前になっている時に以下の操作を行なうと、既に登録されている音声データが消えてしまいます)。

図 3.11に表示されている “AdaptPhrase”, “ModelConversation_AB”, “CSTAR” は、それぞれ登録の際に使用する発声内容を表しています。通常は最も発声量の少ない “AdaptPhrase” を選択すればよいと思いますが、より高い認識率を要求する場合には、 “ModelConversation_AB”(発声量: 中) や “CSTAR”(発声量: 多) を選択して下さい。一般に多く発声する程高い認識率が期待できます (以降は AdaptPhrase を選択したものとして話を進めます)。

[2]ここでは、 mayumi (鈴木真由美さん) を選択した例を示しています。

図3.12のような画面が表示されたら、いよいよ音声登録の開始です。左上の画面に表示されている言葉や文節を入力します。このとき、それぞれの言葉の間に1秒程度のポーズを入れながら、一つの画面に表示されている全ての言葉を、丁寧に発声して入力してください。

例えば、マウスの左ボタンを押したまま、「いきおい、(1秒あける) いよいよ、(1秒あける) うらやましい、(1秒あける) おもしろい」と入力します。

この入力方法は、6ページに示した通りです。

また、波形の切り出し(6ページ参照)にも十分に注意してください(波形の切り出しが誤っている場合には、うまく話者適応ができないことがあります)。

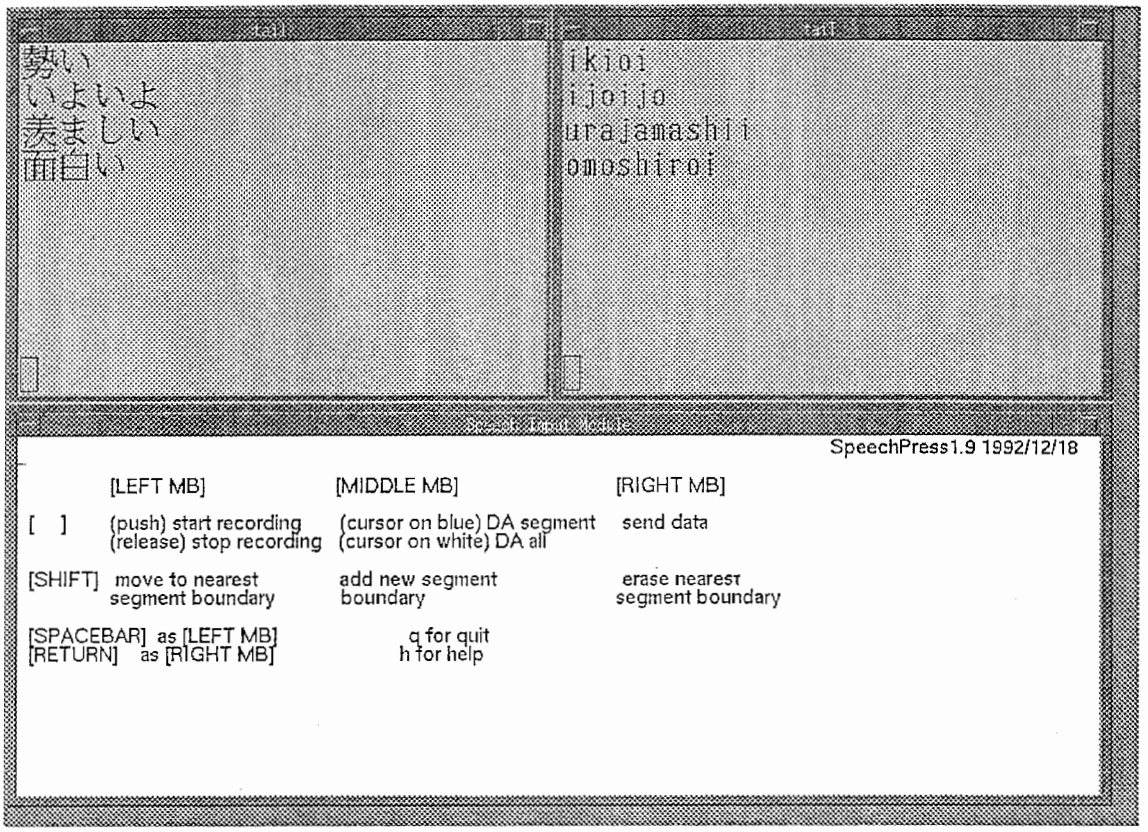

図 3.12: 話者適応入力画面

あらかじめ設定されたすべての言葉の登録が終ると、図3.11のメニューに戻ります。特に問題がなければ、End を選択してください。

これで音声の登録が終了しました。画面には再び図3.10が表示されるはずです。次の処理はモデルの変更です。

このためには、メニューの中から Speaker_Adaptation を選択してください。

メニュー選択後、しばらくは何も表示されません。これは先ほど登録した音声に対する分析処理を行なっているためです。

やがて分析処理が終ると、上部に “Standard Speaker Selection” と表示されたウインドウが表示されます。ここでは、話者適応を行なう際の標準話者の選択が行なわれます。現在のシステムでは、標準話者の候補として、男性3名、女性3名の計6名が登録されています。ここではそれらの中から、使用者の音声の特徴と最も近いと推測される話者のモデルが一つ選択されます。どの話者が標準話者として選択されたかを確認した後、リターンキー を押してください。

この後、続いて上部に “Speaker Adaptation” と表示されたウインドウが表示されます。ここでは先ほど選択された標準話者のモデルパラメータの変更が行なわれます。所要時間は 1 分弱です。

以上で話者適応に関するすべての処理が終了します。これ以降、このシステムを使用する場合には、ここで作られた使用者に専用のモデルが使用されます。

4　システムメニュー

話者選択 (Speaker_Selection)、または話者適応の処理 (Speaker_Adaptation) が終了すると、再びシステムメニュー (図 3.10) に戻る。このメニューには、いままで説明した項目の他に、以下のような機能を持つ項目がある。

4.1　Speaker_Selection

話者適応学習用の音声入力を行なう時間と手間を省くための機能である。「こんにちは」と発声するだけで、標準話者の男女 6 名、および、いままで話者適応が終了している人のモデルの中から、発声者の音声に一番近いモデルを選択する。

1. Speaker_Selection を選択する。

2. 音声入力窓に、「こんにちは」と入力し、右ボタンをクリックする。(図 3.13を参照)

3. 画面下のメッセージ窓で、発声者と各話者のモデルとの近さが計算されるので、終了するまでしばらく待つ。(注：エラーメッセージの表示が表示されるが、無視する)

4. 計算が終了したら、リターンキーを入力する (図 3.14を参照)。以後の音声認識デモには、ここで選択された話者のモデルが用いられる。

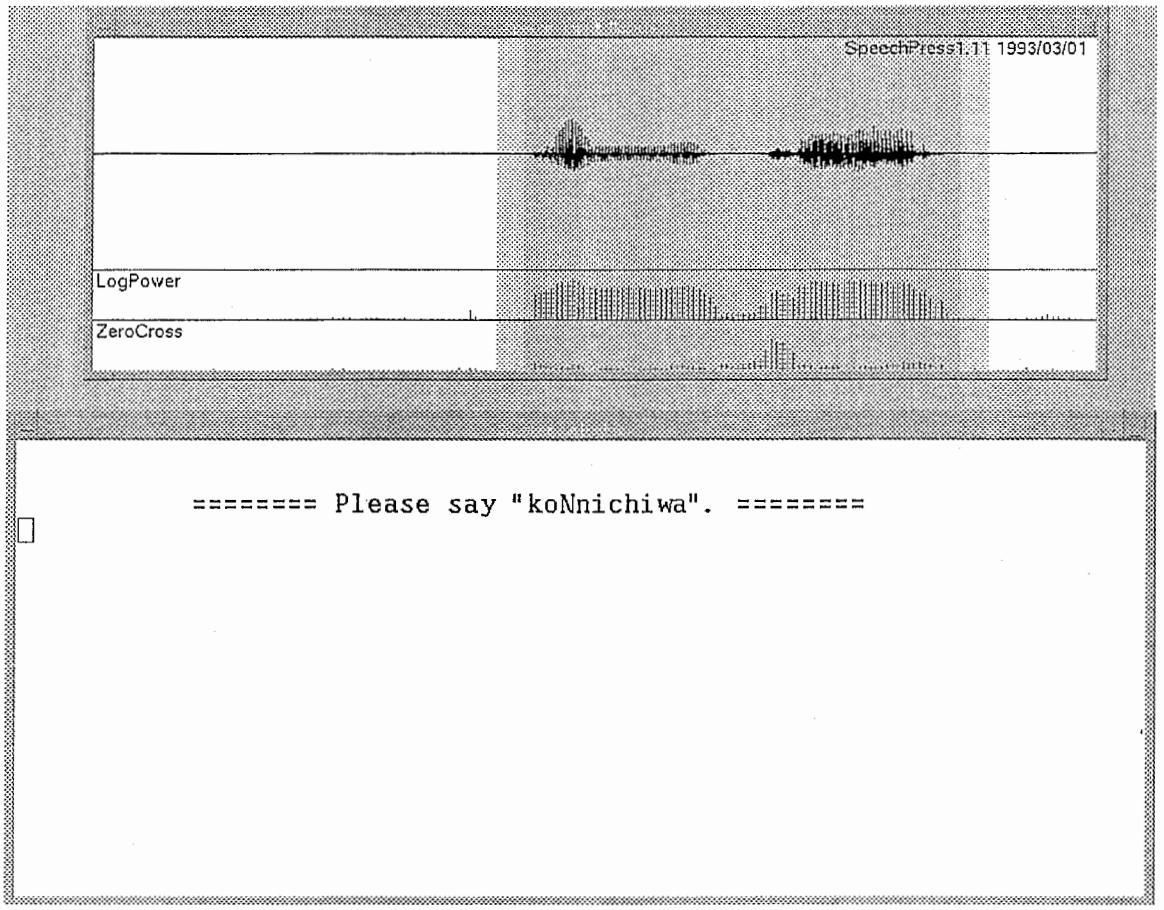

図 3.13: Speaker_Selection 表示画面 (「こんにちは」を入力後)

```
@MESSAGE(WavePara34) inbytes = 21600, outbytes = 11696
@MESSAGE(WavePara34): Fr 85, bytes 11696
ERROR: can't fread 20 byte
@ERROR(WavePara34): fread atrheader from stdin
ERROR: can't fread 20 byte
ERROR (rmheader): fread atrheader from <>
/HMnet/FMS/HMnet.450 : 1.733204e+03
/HMnet/FTK/HMnet.450 : 2.518876e+03
/HMnet/FYM/HMnet.450 : 2.081904e+03
/HMnet/MAU/HMnet.450 : 1.547830e+03
/HMnet/MHT/HMnet.450 : 2.394641e+03
/HMnet/MXM/HMnet.450 : 1.891430e+03
/Data/singer/Adapted_HMnet/HMnet.450 : 3.023836e+03

"/users/Soft/singer/ATREUS/DEMO/Data/singer/Adapted_HMnet/HMnet.450" i
s selected.
Push [RETURN] to return to main demo
```

図 3.14: Speaker_Selection 表示画面 (処理終了)

4.2 Change_Speaker

デモの途中で話者を変更したい時に、この項目を選ぶ。既に登録してある話者を選択するか、あるいは 9ページの手順にしたがって、新しく登録する。

4.3 Options

システムの設定用の項目である (図 3.15参照)。ある項目の設定を変更したい場合は、その行のマウスカーソルを持っていき、どのボタンでもいいからクリックする。すると、設定変更用のウインドウが表示される。数値設定の場合には、emacs のキー操作で修正する。例えば、Ctrl-a で語頭に行き、Ctrl-k で今までの設定を消去し、新しい数値を入力して、リターンする。

- HMnet_Size

  使用する HMnet の状態数の指定である。話者適応版の HMnet は、状態数 450 のものを用いている。その他、不特定話者音声認識用の HMnet は、状態数 200 のものを用いている (付録 46ページを参照)。(default: 48)

- GRAMMAR

  文節内文法のファイル名の選択。(default: `cstar_phrase.sss`)

- BUN_GRA

  文節間文法のファイル名。(default: `cstar_sentence.sss`)

- GLOBAL_BEAM

  ビーム幅の設定。(default: 48)

- MARGIN

  入力音声の前後に付加する無音区間長の設定 (フレーム数)。(default: 5)

- FZK_FILTER

  付属語 ngram の設定 (文献 [8] を参照)。用いない場合は、THROUGH。(default: THROUGH)

- TRANS_WEIGHT 出力確率と遷移確率の重み付けパラメータ。この値が 1 に近いほど、遷移確率に重みがかかる。通常は、0.5。

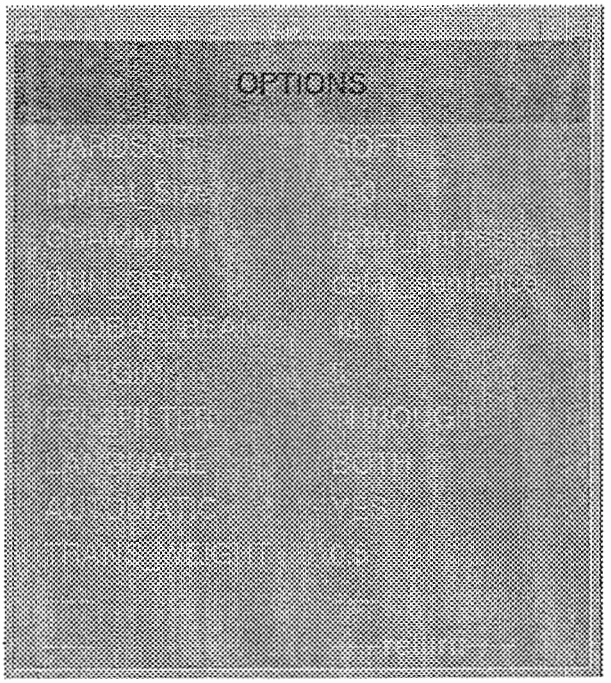

図 3.15: Options 表示画面

5 問題点

1. システムメニューでのいくつかの項目は、今回のリリースではサポートしない。

   - main menu
     - SSS_Principle
     - Speech_Translation
     - Kill_Translation
     - Manual
     - Update
   - options menu
     - HARDSOFT is limited to SOFT (unless IOPBOX is installed as described in Appendix 4)
     - HMnet_Size is limited to 450 (unless models with 200 or 600 states are constructed; see Appendix 5)
     - LANGUAGE doesn't make sense
     - AUTOMATIC doesn't make sense

   もし、これらのオプションが選択されたら、その結果がどうなるかは予測できない。

2. パイプライン構造のため、以下のようなエラーメッセージが頻繁に表示される。

```
ERROR: can't fread 20 byte
@ERROR(WavePara34): fread atrheader from stdin
```

これらのメッセージのほとんどは、UNIX パイプラインの単なる終了合図であり、無視できる。

3. socket プログラムのためのマニュアルインストレーションのエラーメッセージが出力される。

第 4 章

システムを構成する各プログラムの説明

本章では、SSS-LR 連続音声認識システムを構成する、種々のプログラムの概要、及び使用方法について説明する。ソースファイルの所在については、付録を参照のこと。

ATR 連続音声認識システムのデモでは、最近になってフィルターの考え方の枠組を取り入れている。すなわち、音声が入力され、いくつかの直列接続された "filters" を通過して、最終的にフィルタリング処理の結果として、認識結果の文字列が出力される。言い換えれば、音声認識とは、入力される音声信号と、出力される認識文字列との間のフィルタリング問題としてみなされる。フィルター間の接続は、UNIX のパイプとして実現される。

図 4.1に、このデモシステムにおけるデータの流れを示す。それぞれのモジュールの上部に書いてある名前は、実際の実行ファイル名である。第5章で述べる評価システムでは、`speechlabel` プログラムのように、代替的なコンポーネントが用いられる。

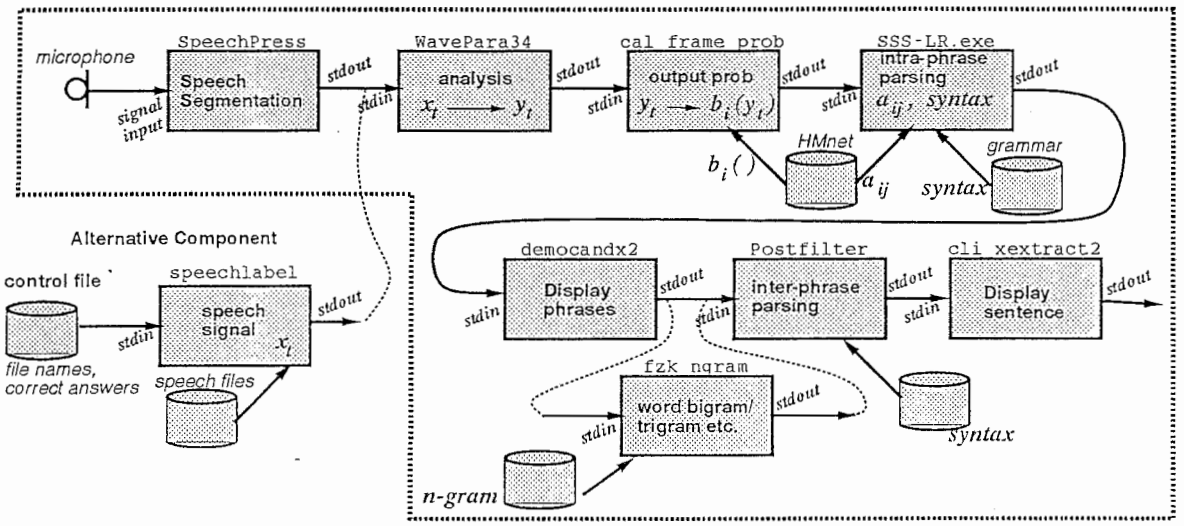

図 4.1: Data flow for ATREUS demo system

1 SPEECHIN

SpeechPress

本プログラム SpeechPress は、インタラクティブな音声の入力を行なう。さらに詳細な説明は、文献 [10] を参照されたい。

1.1 仕様

1. 入力データ

   A/D 変換後のマイク入力音声である。

2. 出力データ

   出力は、標準出力に書く。このために、出力予定のデータのバイト数を指定するヘッダを用いなければならない。ATR で用いているヘッダは、以下のように定義されている。

header.h

```
typedef struct {
  int size;      /* size in bytes (without header) */
  int utt;       /* number of bunsetsu (start with 0) */
  int totalutt; /* total number of bunsetsu in bunsho */
  int time1;     /* time value */
  int time2;     /* time value (for future use) */
} ATRHEADER;
```

   すなわち、20 バイト (5 * 4 int) のヘッダの後に、big-endian フォーマット short で表現されたバイナリデータが続く。utt は、データが何番目の文節で発声されたかを指定し、totalutt は、発声された文中の総文節数を表す。例えば、"kochirawa # kaigizimukyokudesu"(# は文節区切りを表す) という発声において、最初の文節に対するヘッダは utt=0、totalutt=2 であり、二番目の文節に対するヘッダは utt=1、totalutt=2 である。

   size は、生データのバイト数を指定する。

   time1 は、time サブルーチンを用いた値が記入される。このサブルーチンは、00:00:00 GMT, Jan. 1, 1970 からの時刻 (秒) を返す。この値は、リアルタイム実行性能の尺度に使用される。

   time2 は、現在使われていない。

3. 必要なファイル

   ・なにも要らない。

1.2 オプションの説明

実行ファイル SpeechPress を、-@ をオプション指定して実行すると、オプション情報を出力する。いくつかのオプションでは省略時の値が設定されており、すべてのオプションを指定する必要はない。

```
usage: SpeechPress <SpeechPress1.11 1993/03/01>
[-a {fully automatic}]       default: NO
[-A <additional margin>]     default: 0
[-b {bunsho segmentation}]   default: NO
[-c <chmod flag file>]       default: <>
[-d {append date to unseg}] default: NO
[-D <debug level>]           default: 0
[-e {enable manual edit}]    default: YES
[-E <endslots>]              default: 10
[-f {forced exit}]           default: NO
[-F {focus flag}]            default: YES
[-i <input file>]            default: <>
[-I {iconify}]               default: NO
[-H {suppress messages}]     default: NO
[-ht <text height pixel>]    default: 40
[-hw <wave height pixel>]    default: 230
[-hp <power height pixel>]   default: 50
[-hz <zerocross height pix>]default: 50
[-l {show level}]            default: YES
[-L {lock}]                  default: NO
[-m <add. margin in frames>]default: 0
[-M {minimal window}]        default: NO
[-n {no number for outfile}]default: NO
[-N {no subwindow}]          default: YES
[-o {writing to stdout}]     default: NO
[-O <labelfile>]             default: <>
[-s <wait secs for display>]default: 0
[-S <suffix number>]         default: 0
[-u <unseg. outputfile>]     default: <>
[-v <verbose flag>]          default: YES
[-w <outputfile>]            default: xxx
[-x <x position in pixel>]   default: 100
[-y <y position in pixel>]   default: 100
```

使用可能なオプションについて、さらに説明を加える。

-a	flag	exit program *automatically* after first successful segmentation
-A	num	additional margin in data points at beginning of segments
-b	flag	whole utterance segmentation ("sentence"): don't cut into phrases (*bunsetsu*) but use whole sentence. This is also useful for speaker adaptation using words.
-c	file	file name which is used for sync with translation process; `file`'s mode is changed to 222; `file.nph` for number of phrases after successful segmentation
-D	num	display debug information depending on num: the bigger num is, the more debug information is displayed, i.e if num is 0 no debug information is displayed.
-e	flag	enable correction of segmentation boundaries with the mouse
-E	num	minimum number of slots (100ms unit) for judgement of end of utterance
-f	flag	force exit
-F	flag	grab focus
-i	file	read from `file` instead of using data from AD/DA converter
-I	flag	iconify window after segmentation
-ht	num	height of text in pixels
-hw	num	height of wave window in pixels
-hp	num	height of power window in pixels
-hz	num	height of zero-cross window in pixels
-l	flag	display level meter
-L	flag	start in locked mode
-m	num	num additional frames at front and end of each phrase (frameshift is 10 mS); not used for DA!
-M	flag	minimize window size
-n	flag	don't append a number to the output file names for phrase level segmentation
-N	flag	subwindow display (not implemented)
-o	flag	sending data to stdout and not to a file; data is preceded by a header (see 2.3)
-O	file	output (dummy) labels in ATR format to `file`
-s	num	sleep for num seconds after displaying wave and segmentation boundaries
-S	num	start output with `filenum`, e.g `foo28`, `foo29` ... if num was set to 28. This is useful for speaker adaptation using words.
-u	file	output to `file` of unsegmented data, i.e. only rough endpoint segmentation has been performed
-v	flag	show program name and version on wave window
-w	file	wave data segmented at the phrase level (*bunsetsu*) is written to files `file0`, `file1` etc. (unless -n option was chosen)
-x	num	upper left corner x-coordinate of window in pixels
-y	num	upper left corner y-coordinate of window in pixels

1.3 実行シェルの例

以下のオプション設定が標準的である。

```
# program filenames
set SPEECHIN = SpeechPress
$SPEECHIN -o |(出力) &
```

2　LPC

本プログラム WavePara34 は、音声波形信号から音響的特徴量を抽出する、LPC 分析を行なう。さらに詳細な説明は、文献 [11] を参照されたい。

2.1　仕様

1. 入力データ

   SpeechPress からの出力を、標準入力から読みこむ。18ページを参照のこと。

2. 出力データ

   出力を、標準出力に書く。

   ATR ヘッダ (18ページ参照) の後に、34 次元の特徴ベクトル (1 パワー、16 ケプストラム、1 Δパワー、16 Δケプストラム) が、フロートの値 (e.g. 9,384 byte for 69 time-frames) で続く。

3. 必要なファイル

   ・なにも要らない。

2.2　オプションの説明

実行ファイル WavePara34 を、-@ をオプション指定して実行すると、オプション情報を出力する。いくつかのオプションでは省略時の値が設定されており、すべてのオプションを指定する必要はない。

```
usage: WavePara34
[-D <debug option>]                    val: 0
[-em preemphasis coefficient]          val: 0.980000
[-fl frame length in ms]               val: 20.000000
[-fs frame shift in ms]                val: 5.000000
[-p LPC order]                         val: 16
[-q cepstrum order]                    val: 16
[-sf sampling freq in Hz]              val: 12000
```

2.3　実行シェルの例

以下のオプション設定が標準的である。

```
# program filenames
set LPC        = WavePara34
(入力) | $LPC -fs 10 |(出力)
```

3 SSSLIKE

本プログラム cal_frameprob は、HMnet の各状態の出力確率計算を行なう。

3.1 仕様

1. 入力データ

   WavePara34 からの出力を、標準入力から読みこむ。22ページを参照のこと。

2. 出力データ

   出力を、標準出力に書く。

   ATR ヘッダ (18ページ参照) の後に、それぞれのフレーム毎の出力確率値が、フロートの値 (e.g. 124,200 byte for 450 states and 69 timeframes) で続く。

3. 必要なファイル

   - HMnet モデル 文献 [9] 参照
   - log 計算表 (log.tbl)

   HMM の確率計算では、確率値を log 化して扱っている。この log 計算を高速化するために log compression table と呼ばれる表を用いている。

3.2 オプションの説明

実行ファイル cal_frameprob を、-@ をオプション指定して実行すると、オプション情報を出力する。いくつかのオプションでは省略時の値が設定されており、すべてのオプションを指定する必要はない。-l、-m オプションは必ず指定しなければならない。

```
usage: cal_frame_prob
[-D <debug option>]          val: 0
[-l <log table>]             val: <log.tbl>
[-m <model list>]            val: <>
```

使用可能なオプションについて、さらに説明を加える。

-D flag デバッグ情報を出力する。

-l file log 計算表のファイル名を指定する。

-m file HMnet のファイル名を指定する。

3.3 実行シェルの例

以下のオプション設定が標準的である。

```
# program filenames
set SSSLIKE   = cal_frameprob
# filenames
set LOG_TABLE = log.tbl.HP
set HMNET     = HMnet.600.IOP
(入力) | $SSSLIKE -L $LOG_TABLE -m  $HMNET |(出力)
```

4　SSS

本プログラム SSS-LR.exe は、連続音声認識を行なう。

4.1　仕様

1. 入力データ

   cal_frameprob からの出力を、標準入力から読みこむ。23ページを参照のこと。

2. 出力データ

   出力情報の種類に応じて、各出力の先頭に#識別子が付与されている。出力情報は、以下の情報である。

   識別子

   p: 処理時間や処理量に関する情報。
   x: 認識途中の第一位認識候補の出力。
   r: 認識結果の出力。／は、単語の区切りを表す。 数字は認識スコア。
   k: 認識結果の出力 (漢字)。後述の文節間 LR パーザ (Postfilter) への情報。
   l: 付属語 bigram モデルのための情報 (文献 [8] を参照)。
   #: 一文節の認識処理が終了したことを表す。

   また、各識別子の次の数字について説明する。第一番目の数は、一文のなかで何番目に発声された文節かを示す。 0 から始まる。第二番目の数は、一文を構成している文節数である。さらに、識別子の r と l について、第二番目の数は、認識結果の順位を表す。また、識別子 l について、第四番目から認識スコアの前までの数の列は、単語の番号 (.dic のルール番号) を表す。最後の数は、認識スコアが最終的に確定した時点のフレーム数を表す。

   標準出力へのデータサンプル

```
#p 0 1 AllBunsetsuTime= 23
#p 0 1 TimeStamp= 731771503
#x 0 1  m
#x 0 1  mo
#x 0 1  mosh
#x 0 1  moshi
#x 0 1  moshim
#x 0 1  moshimo
#x 0 1  moshimosh
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#x 0 1  moshimoshi
#p 0 1 CPU= 4180 msec Elapsed= 5 sec TotalVerify= 4152 Acc= 57 Depth= 15
#r 0 1 1  moshimoshi                                          (-37.357732)
#k 0 1 もしもし <interj> -37.357732
#l 0 1 1      1  146  -37.357732 69
```

```
#r 0 1 2  moo/shi/mashi/te                                    (-32.708065)
#k 0 1 申しまして <vaux-s> -32.708065
#l 0 1 2      4  528  456  377  122  -32.708065 69
#r 0 1 3  moo/shi                                             (-32.205890)
#k 0 1 申し <vaux-coord> -32.205890
#l 0 1 3      2  528  456  -32.205890 69
#r 0 1 4  moo/shi/masu                                        (-31.257637)
#k 0 1 申します <vaux> -31.257637
#l 0 1 4      3  528  456  378  -31.257637 69
#r 0 1 5  moo/shi/mashi/ta                                    (-31.253937)
#k 0 1 申しました <vaux> -31.253937
#l 0 1 5      4  528  456  377  375  -31.253937 69
#r 0 1 6  mooshiwake                                          (-31.022349)
#k 0 1 申し訳 <np> -31.022349
#l 0 1 6      1  688  -31.022349 69
#r 0 1 7  gozhjuushjo/e                                       (-30.730364)
#k 0 1 ご住所へ <np-e> -30.730364
#l 0 1 7      2  609  297  -30.730364 69
#r 0 1 8  go/seN/eN/o                                         (-30.582171)
#k 0 1 五千円を <np-money-sen-o> -30.582171
#l 0 1 8      4  816  803  798  292  -30.582171 69
#r 0 1 9  moo/shi/masu/ne                                     (-30.465777)
#k 0 1 申しますね <vaux-sfp> -30.465777
#l 0 1 9      4  528  456  378  138  -30.465777 69
#r 0 1 10  jooshi/e                                           (-30.013799)
#k 0 1 用紙へ <np-e> -30.013799
#l 0 1 10      2  608  297  -30.013799 69
## 0 1
```

3. 必要なファイル

   - HMnet モデル 文献 [9] 参照
   - 音素継続時間モデル 文献 [9] 参照
   - log 計算表 (ATREUS/DEMO/ETC/log.tbl.HP)

   HMM の確率計算では、確率値を log 化して扱っている。この log 計算を高速化するために log compression table と呼ばれる表を用いている。

   - 文節内文法 (cstar_phrase.sss.gra) 34ページ参照

   文法は文脈自由文法で記述されている。以下に、簡単な文法のサンプルを示す。

   文節内文法 (.gra) のフォーマット

```
(<start> <--> (<_start>))
(<_start> <--> (q1 <bunsetu> q2))
(<bunsetu> <--> (<np>))
(<np> <--> (<n> <p>))
(<n> <--> (<n-hutu>))
(<n-hutu> <--> (ts u u j a k u d e = w a))
(<n-hutu> <--> (k a i g i))
(<p> <--> (n o))
```

```
(<p> <--> (g a))
(<p> <--> (t o))
```

この文法は、「通訳電話の」「通訳電話が」「通訳電話と」「会議の」「会議が」「会議と」という6つの文節だけを受理する。q1、q2は、それぞれ語頭、語尾の無音モデルを駆動するためのラベルである。非終端記号は、< >で囲まれた記号である。終端記号は音素ラベルである。実際に文法で用いている具体的な音素ラベルは、5母音(/a, i, u, e, o/)、18子音(/b, d, g, p, t, k, ts, ch, n, m, =, s, sh, z, zh, j, h, w/)、促音(/q/)の合計24音素である。ただし、/=/は撥音を表す。

この文法(.gra)は、かな漢字付きの文法(.lex)から、grmakeというプログラムで作成される。従って、ユーザーが自分の好きなタスクで認識実験するために文法を変更する場合には、かな漢字付きの文法(.lex)から作成しなければならない。以下にそのフォーマットを示す。

かな漢字付き文法(.lex)のフォーマット (see ATREUS/DEMO/ETC/test.lex)

```
(<start> <--> (<_start>))
(<_start> <--> (q1 <bunsetu> q2))
(<bunsetu> <--> (<np>))
(<np> <--> (<n> <p>))
(<n> <--> (<n-hutu>))
(<n-hutu> <--> (ts u u j a k u d e = w a) ("通訳電話" "通訳電話" "普名詞"))
(<n-hutu> <--> (k a i g i) ("会議" "会議" "普名詞"))
(<p> <--> (n o) ("の" "の" "格助詞"))
(<p> <--> (g a) ("が" "が" "格助詞"))
(<p> <--> (t o) ("と" "と" "格助詞"))
```

第一番目のかな漢字は、音声認識結果のかな漢字表示に用いられる。第二番目以降のかな漢字と品詞は、翻訳処理のための情報である。動詞などの活用語の場合は、品詞名のあとに活用形の名前が追加される。しかし、音声認識の部分に用いられる情報は、第一番目のかな漢字だけであり、第二番目と第三番目に記述するものはdummyとなる。

このようなフォーマットで書かれた文法であれば、SSS-LRに使用することができる。実際の文法の変更の手順を以下に示す。

(a) 24音素ラベル表記の.lexファイル(test.lex)を作成する。ただし、漢字コードはEUCでなければならない。

(b) 以下のように、grmakeを用いて、test.lexをtest.graとtest.dicに分離する。

```
grmake test
```

(c) test.graに対して、slrを用いて、test.tabを作成する。

```
slr test
```

(注： SSS-LRで用いる24音素ラベル表記の文法には、ファイル名に、sssという識別子が付与されている。)

・文節内LRテーブル(cstar_phrase.sss.tab) 34ページ参照

slrによって作成された.tabファイルがLRテーブルである。

文節内文法のフォーマット

```
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;;
;;      SLR parsing table for HMM-LR
;;      ----------------------------
;;      Created on Thu Mar 11 10:35:20 1993
;;      Constructor version: V1.1
;;
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
(slr-table)
(0 (q1 s1) (<_start> g2))
(1 (ts s3) (k s4) (<n-hutu> g5) (<n> g6) (<np> g7) (<bunsetu> g8))
(2 ($ a))
(3 (u s9))
(4 (a s10))
(5 (n r4) (g r4) (t r4))
(6 (n s11) (g s12) (t s13) (<p> g14))
(7 (q2 r2))
(8 (q2 s15))
(9 (u s16))
(10 (i s17))
(11 (o s18))
(12 (a s19))
(13 (o s20))
(14 (q2 r3))
(15 ($ r1))
(16 (j s21))
(17 (g s22))
(18 (q2 r7))
(19 (q2 r8))
(20 (q2 r9))
(21 (a s23))
(22 (i s24))
(23 (k s25))
(24 (n r6) (g r6) (t r6))
(25 (u s26))
(26 (d s27))
(27 (e s28))
(28 (= s29))
(29 (w s30))
(30 (a s31))
(31 (n r5) (g r5) (t r5))
```

・文節内かな漢字辞書 (cstar_phrase.sss.dic) 34ページ参照

grmake によって、.lex から分離された、かな漢字ファイルが.dic である。先頭の数字は、ルール番号である。

文節内かな漢字辞書のフォーマット

```
5 通訳電話 通訳電話 普名詞
6 会議 会議 普名詞
7 の の 格助詞
```

8 が が 格助詞
9 と と 格助詞

4.2 オプションの説明

実行ファイル SSS-LR.exe を、オプションなしで実行すると、オプション情報を出力する。いくつかのオプションでは省略時の値が設定されており、すべてのオプションを指定する必要はない。また、現在使用されていないものもある。以下に、それらの説明を行なう。

```
Usage: SSS-LR.exe Word-file (with following options)
  -g grammar                                必ず指定
  -B global-beam                            default: 100
  -e HMnet state number                     使用不可
  -K ( 0...use of globalmodel  1...use of multimodel )  必ず 1 を指定
  -b local-beam                             default: 15
  -c total-cells                            default: 512
  -X Threshold                              default: -2000.0
  -x Minimal-threshold                      default: -1500.0
  -Q Threshold_Q                            default: -3000.0
  -d (disable dynamic threshold setting) 使用不可
  -n N-th                                   default: 5
  -m file name of HMnet                     必ず指定
  -s file name of Silence model             使用不可
  -D duration_file name                     必ず指定
  -L log-table                              default: log.tbl
  -N duration-standard-deviation            default: 3.0
  -V (Verbose)                              使用不可
  -T (Trace)                                使用不可
  -t (Trace bun)                            使用不可
  -F (First-trellis)                        使用不可
  -P frame-period (msec)                    default: 10
  -A add-time (frame MAE & USHIRO)          default: 10
  -E end-free (msec)                        使用不可
  -o dump-file                              使用不可
  -O Kanji output result file               使用不可
  -y Sigma_Scale                            default: 1.0
  -a dur conv break point                   必ず指定 (3.0 に固定)
  -R dur conv rate                          必ず指定 (0.6 に固定)
```

使用可能なオプションについて、さらに説明を加える。

-g file 文法のファイル名 (拡張子不要)

-B num ビーム幅。この値が小さいほど、多く枝刈りされる。

-K num 1: 学習されなかった音素環境が出現した場合、その音素環境に近い複数のモデルで多数決をするモード。0: 学習されなかった音素環境が出現した場合、音素環境に非依存のグローバルモデルを駆動するモード。

-b num 局所的なビーム幅 (一つの認識候補から伸びる枝に対する枝刈り)

-c num 総セル数の指定。セルとは認識候補を格納するためのデータ構造。セルの数はビーム幅よりも十分大きくなければならない。

-X num 予測音素の存在確率が低い場合に仮説を捨てるための、いき値。省略時には負の値であり、仮説の棄却を行なわない場合に相当する。

-x num 上述のいき値の最小値。省略時には負の値であり、仮説の棄却を行なわない場合に相当する。

-Q num 予測音素 (無音) の存在確率に対するいき値。省略時には負の値であり、仮説の棄却を行なわない場合に相当する。

-n num 第 n 位までの認識結果を出力する。

-m file HMnet ファイル名

-D file 音素継続時間制御ファイル名

-L file log 計算表のファイル名

-N num 音素継続時間モデルの分散拡大率 (倍)

-P num フレーム周期

-A num 音声データの前後に含ませる無音区間の長さ

-y num HMnet モデルの分散拡大 (倍)

-a num 音素継続時間モデルの補正パラメータ。ここで指定する値以上の継続時間 (フレーム) を持つモデルに対して、-R で指定する比率をかけて補正する。このオプションは、単語で学習した音素継続時間モデルを、連続発声の発声速度に適応させるためのもの。

-R num 音素継続時間モデルの補正パラメータ。

4.3 実行シェルの例

以下のオプション設定が標準的である。ただし、.flag ファイルは現在使われていないので、必要ない。

```
# program filename
set SSSLR = SSS-LR.exe
# filenames
set LOG_TABLE = log.tbl.HP
set GRAMMAR   = cstar_phrase.sss
set HMNET     = HMnet.600.IOP
set DURMODEL  = HMnet.100
(入力) | $SSSLR -L $LOG_TABLE -g $GRAMMAR -B 48 \
                -m $HMNET -D $DURMODEL -N 2.2 -K 1 \
                -P 5 -b 64 -a 3.0 -R 0.6 \
```

```
-c 3000 -A 10 .flag      |(出力)
```

5 POSTFILTER

本プログラムは、SSS-LR の文節認識結果の文節ラティスに、文節間文法を適用して、文の候補を出力する。

5.1 仕様

1. 入力データ

   SSS-LR の出力情報のうち、識別子 r、k、# の情報。

   sample.input

```
#r 0 2 1  sochira/wa                                    (-39.997200)
#k 0 2 そちらは <np> -39.997200
#r 0 2 2  sochira/ga                                    (-39.480426)
#k 0 2 そちらが <np> -39.480426
#r 0 2 3  sochira                                       (-39.176741)
#k 0 2 そちら <np> -39.176741
#r 0 2 4  sochira/ja                                    (-39.014049)
#k 0 2 そちらや <np> -39.014049
#r 0 2 5  sochira/o                                     (-38.256387)
#k 0 2 そちらを <np-o> -38.256387
#r 0 2 6  sochira/mo                                    (-38.244088)
#k 0 2 そちらも <np> -38.244088
#r 0 2 7  sochira/de/wa                                 (-38.115494)
#k 0 2 そちらでは <np> -38.115494
#r 0 2 8  sochira/ni/wa                                 (-37.830109)
#k 0 2 そちらには <np> -37.830109
#r 0 2 9  sochira/made                                  (-37.724014)
#k 0 2 そちらまで <np> -37.724014
#r 0 2 10  sore/ja                                      (-37.696215)
#k 0 2 それや <np> -37.696215
## 0 2
#r 1 2 1  kaigizhimukjoku/desu/ka                       (-37.158442)
#k 1 2 会議事務局ですか <n-vaux-sfp> -37.158442
#r 1 2 2  kaigizhimukjoku/desu/to                       (-36.573171)
#k 1 2 会議事務局ですと <n-vaux-s> -36.573171
#r 1 2 3  kaigizhimukjoku/desu/to                       (-36.573171)
#k 1 2 会議事務局ですと <n-quote> -36.573171
#r 1 2 4  kaigizhimukjoku/desu/to                       (-36.573171)
#k 1 2 会議事務局ですと <n-quote> -36.573171
#r 1 2 5  kaigizhimukjoku/o                             (-36.149193)
#k 1 2 会議事務局を <np-o> -36.149193
#r 1 2 6  kaigizhimukjoku/desu/ga                       (-35.625919)
#k 1 2 会議事務局ですが <n-vaux-s+f> -35.625919
#r 1 2 7  kaigizhimukjoku/deshjo/o                      (-35.353432)
#k 1 2 会議事務局でしょう <n-vaux-cop> -35.353432
#r 1 2 8  kaigizhimukjoku/desu/ne                       (-35.271736)
#k 1 2 会議事務局ですね <n-vaux-sfp> -35.271736
```

```
#r 1 2 9  kaigizhimukjoku/desu                    (-35.263037)
#k 1 2 会議事務局です <n-vaux-cop> -35.263037
#r 1 2 10  kaigizhimukjoku/kara                   (-35.204740)
#k 1 2 会議事務局から <np> -35.204740
## 1 2
```

2. 出力データ

   以下のフォーマットである。識別子は j 。

```
nagai@atrq09<1171> cat sample.input|Postfilter -g cstar_sentence.sss
-> Loading grammar rules from 'cstar_sentence.sss.gra'...[247 rules]
-> Loading LR parsing table from 'cstar_sentence.sss.tab'...[281 states]
#j 0 1 そちらは＃会議事務局ですか          (score = -38.577820)
#j 0 1 そちらが＃会議事務局ですか          (score = -38.319435)
#j 0 1 そちら＃会議事務局ですか            (score = -38.167595)
#j 0 1 そちらや＃会議事務局ですか          (score = -38.086246)
#j 0 1 そちらは＃会議事務局ですが          (score = -37.811562)
#j 0 1 そちらも＃会議事務局ですか          (score = -37.701263)
#j 0 1 そちらは＃会議事務局でしょう        (score = -37.675316)
#j 0 1 そちらでは＃会議事務局ですか        (score = -37.636971)
#j 0 1 そちらは＃会議事務局ですね          (score = -37.634468)
#j 0 1 そちらは＃会議事務局です            (score = -37.630119)
## 0 1
Postfilter terminated.
```

3. 必要なファイル

   - 文節間文法 (cstar_sentence.sss.gra) 34ページ参照
   - 文節間 LR テーブル (cstar_sentence.sss.tab) 34ページ参照

5.2 オプションの説明

Postfilter をオプションなしで実行すると、以下のオプション情報を出力する。

```
Usage: Postfilter -g grammar [-r] [-romaji+kanji] [-k] [-kanji+romaji]
[-v] [-B number] [-b number] [-t]
  -g <grammar>:  inter-phrase grammar[ use (grammar).gra,(grammar).tab]
  -r :  output romaji-sentence only (#k line is neccessary)
  -k :  output japanese-sentence only[Default]
  -romaji+kanji or -kanji+romaji : output japanese and romaji sentence
  -B <globalbeam> : global beam [Default 100]
  -b <localbeam>  : local beam  [Default 18]
  -v   returns  input line to stdout
  -t   returns parsing-realtime to stdout
```

5.3 実行シェルの例

```
# filenames
set BUNSHO_GRA    = cstar_sentence.sss
(入力) | Postfilter -g $BUNSHO_GRA -romaji+kanji -t | (出力)
```

6　その他

表 4.1: その他のプログラム

directory	*executable(s)*	*explanation*
FZK_NGRAM	fzk_ngram	付属語 bigram [8]
SPEECHLABEL	speechlabel	prepare wavedata for UNIX pipeline
DEMOCANDX2	democandx2 srv_democandx2	文節候補の表示 [12]
XEXTRACT2	srv_xextract2 cli_xextract2 srv_trans cli_trans	文候補の表示 [12]

第 5 章

評価実験用システムの説明

バッチ形式の評価用シェル・スクリプト群は、ディレクトリ ATREUS/EVAL にある。このディレクトリ構造は、付録 3に示してある。

1　バッチ形式の評価の例

簡単な例について説明する。まず、話者 FAK の HMnet は、話者 FYM (ATREUS/DEMO/HMnet/FYM/HMnet.450) を用いて、話者 FAK の発声した 25 単語 (勢い、いよいよ、etc.) により話者適応される。次に、話者 FAK の発声した二つの文 (もしもし、そちらは＃会議事務局ですか) が、話者適応された HMnet で認識される。参照のための認識結果は、ディレクトリ ATREUS/EVAL/REF_RESULT に存在する。以下のコマンドは、新しいディレクトリ ATREUS/EVAL/RESULT を作成して、このような話者適応と音声認識を実行する。

```
$ cd EVAL
$ runmain.csh
```

runmain.csh は、 run.csh を呼ぶ。詳細については、ディレクトリ ATREUS/DEMO/EVAL/SCRIPT の csh スクリプトと awk スクリプトを参照されたい。参考として、 runmain.csh と run.csh のソースコードを、付録 3に示す。

典型的な文節認識の結果 (第 5 位までの出力) を、以下に示す。

```
file ATREUS/EVAL/RESULT/250/result.bunsetsu.tmp
(1) 0 0 |moshimoshi|
**************************
Parsing time: CPU-time = 4180 sec, Elapsed-time = 5 sec.
Total-verify= 4152, Accepted= 57, Depth= 15, hypo= 0
  1. moshimoshi           (-37.357732)
  2. moo/shi/mashi/te     (-32.708065)
  3. moo/shi              (-32.205890)
  4. moo/shi/masu         (-31.257637)
  5. moo/shi/mashi/ta     (-31.253937)
**************************
(2) 0 0 |sochirawa|
**************************
Parsing time: CPU-time = 4700 sec, Elapsed-time = 6 sec.
Total-verify= 4364, Accepted= 66, Depth= 16, hypo= 0
  1. sochira/wa           (-39.997200)
  2. sochira/ga           (-39.480426)
  3. sochira              (-39.176741)
  4. sochira/ja           (-39.014049)
  5. sochira/o            (-38.256387)
**************************
(3) 0 0 |kaigizhimukjokudesuka|
**************************
Parsing time: CPU-time = 5240 sec, Elapsed-time = 5 sec.
Total-verify= 4729, Accepted= 176, Depth= 29, hypo= 0
  1. kaigizhimukjoku/desu/ka (-37.158442)
```

```
  2. kaigizhimukjoku/desu/to (-36.573171)
  3. kaigizhimukjoku/desu/to (-36.573171)
  4. kaigizhimukjoku/desu/to (-36.573171)
  5. kaigizhimukjoku/o    (-36.149193)
**************************
```

かっこの中の値は、その認識候補の尤度 (log 値) を表す。

2　音声認識用辞書ファイル

2.1　文法ファイルの説明

1. **文法のバージョン管理**
   オリジナル文法 (日本 IR から納品された文法) のバージョン管理は、ATR 自動翻訳電話研究所 データ処理研究室　谷戸 文廣氏が行なっている。

2. **文法の種類・名前**
   文法の名前は、以下の二種類の名前の組合せになっている。ただし、今回のリリースでは、cstar 用文法のみ扱う。

   (1) タスク別の分類
   - cstar　: CSTAR 用文法
   - eset　: 1500 語彙、機能評価文 600 文
   - mset　: 744 語彙、モデル会話 A、B、1-10

   (2) 文法の適用対象別の分類
   - phrase　: 文節内
   - sentence　: 文節間
   - twolevel　: 2 段 LR パーザ

   拡張子は、以下の種類がある。

   - .lex　: 文法規則 + かな漢字 (オリジナル文法)
   - .gra　: 文法規則
   - .dic　: かな漢字辞書
   - .tab　: LR テーブル
   - .cat　: 2 段 LR 用品詞表

謝辞

SSS-LR デモシステムの構築と実装、及び本マニュアルの作成には、筆者ら以外にも多くの人の努力と貢献がありました。

鷹見淳一、大倉計美、山口耕市、小坂哲夫、藤原紳吾、嵯峨山茂樹、杉山雅英各氏は音声認識システムの構築に寄与されました。谷戸文廣、保坂順子の両氏は音声認識のための文法の開発に寄与されました。林輝昭氏は音声認識と言語翻訳を接続する部分のプログラム開発およびシステムの評価実験に寄与されました。田川博章氏は、デモソフトウエアのインストールチェックに寄与されました。泉昌明氏は画面の開発等に寄与されました。小野佳範、山口毅 両氏はネットワークの整備やマシンの管理に寄与されました。 socket プログラムは、 Juergen Nickelsen (nickel@cs.tu-berlin.de) 氏による PDS です。

記して感謝いたします。

参考文献

[1] 永井 明人, 鷹見 淳一, 嵯峨山 茂樹: "逐次状態分割法 (SSS) と音素コンテキスト依存 LR パーザを統合した SSS-LR 連続音声認識システム," 電子情報通信学会技術研究報告, SP92-33, pp. 69–76 (1992.06).

[2] Akito Nagai, Junichi Takami, Shigeki Sagayama: "The SSS-LR Continuous Speech Recognition System: Integrating SSS-derived Allophone Models and a Phoneme-Context-Dependent LR Parser," Proc. of 1992 International Conference on Spoken Language Processing , Vol. 2, pp. 1511–1514, Canada (1992.10).

[3] Akito Nagai, Shigeki Sagayama, Kenji Kita, Hideaki Kikuchi: "Three Different LR Parsing Algorithms for Phoneme-Context-Dependent HMM-based Continuous Speech Recognition," Trans. of IEICE, (電子情報通信学会論文誌 音声対話処理特集号)(1993.01).

[4] H. Singer and M. Sugiyama, "Manual for Speech Input Programs: `SpeechIn`, `SpeechPress`," Tech. Rep. TR-I-0302, ATR, (1993).

[5] 鷹見, 永井 明人, 嵯峨山 茂樹, "逐次状態分割法 (SSS) と LR パーザを統合した SSS-LR 連続音声認識手法における話者適応の性能評価," 音学講論, 2-5-5 (1992.10).

[6] J.Takami, A.Nagai, S.Sagayama, "Speaker Adaptation of the SSS (Successive State Splitting)-Based Hidden Markov Network for Continuous Speech Recognition," Proc. SST92 (Australia) (1992.12)

[7] 鷹見, 永井, 嵯峨山, "話者適応型 SSS-LR 連続音声認識方式における標準話者予備選択の効果," 音学講論 (1993.3 掲載予定).

[8] 磯谷亮輔, 嵯峨山茂樹, 粟津辰功, "付属語の N-gram、自立語の N-gram を用いた音声認識," 音学講論, (1993.3 掲載予定).

[9] 鷹見, "SSS-ToolKit (ver1.0) ユーザーズ・マニュアル," Tech. Rep. TR-I-0324, ATR, (1993).

[10] H. Singer and M. Sugiyama, "Manual for Speech Input Programs: `SpeechIn`, `SpeechPress`," Tech. Rep. TR-I-0302, ATR, (1993).

[11] 嵯峨山, "音声認識のための音声分析とラベル変換," Tech. Rep. TR-I-0347, ATR, (1993).

[12] 竹澤, その他, "ATR 音声翻訳システム ASURA の実装," Tech. Rep. TR-I-0303, ATR, (1993).

付録 A

ソース・ファイルの所在

以下に、本システムのディレクトリ構造を示し、簡単な説明を加える。

1 Main Directory Structure

```
ATREUS
 |
 +- ADAPT                SSS-VFS speaker adaptation (SOURCE)
 +- ADDHEADER            add ATR header to data      (SOURCE)
 +- ALLOC_DSP            distribute models to DSPs for IOPBOX (SOURCE)
 +- BINHP                executables for HP-UX
 +- DEMO                 the main demo
 |   |
 |   +- Adapt            temporary files for adaptation
 |   |   |
 |   |   +- Seqlist
 |   |
 |   +- BINHP -> ../BINHP
 |   +- BINUTIL          binary utilities without source
 |   +- Data             data of adapted speakers; 1 example
 |   |   |
 |   |   +- singer
 |   |       |
 |   |       +- Adapted_HMnet
 |   |       +- List
 |   |       +- Para
 |   |       +- Wave
 |   |
 |   +- Dur              duration control HMnet
 |   +- ETC              logtable, etc.
 |   +- Exe              scripts for demo
 |   +- Gra              grammars
 |   +- HMnet            standard HMnets for 6 speakers
 |   |   |
 |   |   +- FMS
 |   |   +- FTK
 |   |   +- FYM
 |   |   +- MAU
 |   |   +- MHT
 |   |   +- MXM
 |   |
 |   +- Reco             temporary files for recognition
 |   +- Sp_select        temporary files for speaker selection
 |       |
 |       +- Seqlist
 |
 +- DEMOCANDX2           display bunsetsu candidates (SOURCE)
 +- ENDIAN               little-endian, big-endian conversion   (SOURCE)
 +- EVAL                 main evaluation
 |   |
 |   +- DATA             labelled data for evaluation
 |   |   |
 |   |   +- FAK
 |   |       |
 |   |       +- LBL      labels (only start and end points)
```

```
|   |       |   |
|   |       |   +- B
|   |       |   +- MA2
|   |       |
|   |       +- WAV      wave data (short, big-endian, no header)
|   |           |
|   |           +- B
|   |           +- MA2
|   |
|   +- LIST             lists for labelled data
|   +- REF_RESULT       reference results: HMnet
|   |   |
|   |   +- 250          recognition result for beamwidth 250
|   |
|   +- SCRIPT           scripts for evaluation
|
+- FZK_NGRAM            particle bigrams for reevaluation of bunsetsu scores (SOURCE)
+- GKILL                kill all children processes (SOURCE)
+- GRMAKE               grammar management tool (SOURCE)
+- INCLUDE              common .c and .h files (SOURCE)
+- LPC                  wave to cepstrum conversion (SOURCE)
+- MAKE_SEQLIST         conversion to label format for adaptation (SOURCE)
+- MAKE_WORD_SAMPLE     conversion to data format for adaptation(SOURCE)
+- MULTI_SOCKET         program connection with sockets instead of UNIX pipes; multiple machines (SOURCE)
+- POSTFILTER           bunsetsu to bun LR (SOURCE)
+- REORDER              reorder bunsetsu; multiple machines (SOURCE)
+- RMHEADER             remove ATR header from file (SOURCE)
+- SCORE                score SSS-LR results (SOURCE)
+- SELECT_MODEL         tool for speaker selection (SOURCE)
+- SOCKET               connecting programs with sockets instead of UNIX pipes (SOURCE)
+- SLR                  grammartool: compilation of LR table (SOURCE)
+- SPEECHIN             interactive speech acquisition (SOURCE)
|   |
|   +- BINHPOLD
|   +- SRC
|
+- SPEECHLABEL         labelled speech files to ATR header format (SOURCE)
+- SP_SELECT           speaker selection (SOURCE)
+- SSS                  SSS-LR     (SOURCE)
|   |
|   +- Src_parser
|   +- Src_verify
|
+- SSSLIKE             likelihood calculation (SOURCE)
+- THROUGH             ''do nothing'' filter (SOURCE)
+- VAR_CONV            conversion of variances of HMM (SOURCE)
+- XEXTRACT2           display JAPANESE bun (SOURCE)
    |
    +- DISP
    +- PARENT
    +- TRANS
    +- TRANS_PARENT
+- XINPUT              X Window interactive string input (SOURCE)
+- XMENU               X Window interactive menu (SOURCE)
```

2 Scripts for the Demo

Dependency of scripts for ATREUS demo. All scripts are in directory ATREUS/DEMO/Exe.

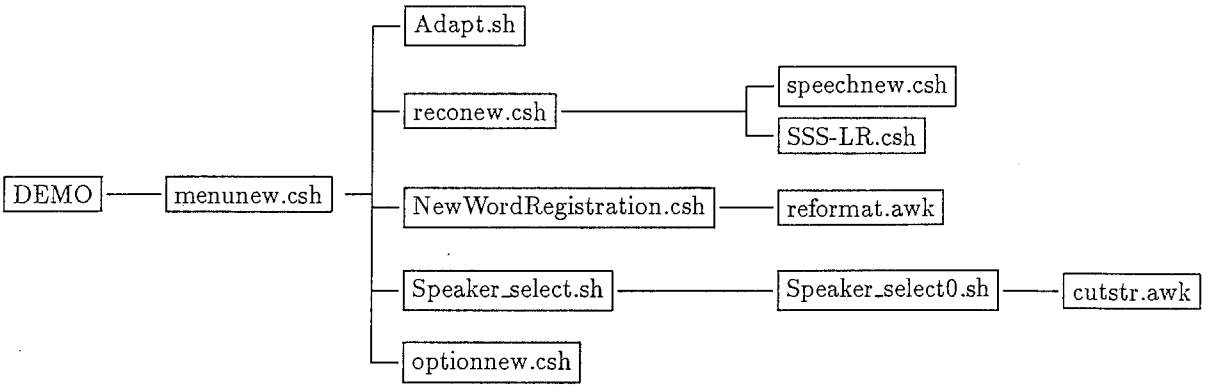

図 A.1: Scripts for demo

3 Directory Structure and Files for Evaluation

```
EVAL
 |- DATA                              all wavefiles and labelfiles (start-end)
 |   |- FAK
 |   |   |- LBL
 |   |   |   |- B                     25 labelfiles (start-end) for adaptation
 |   |   |   |   |- FAK_B_0001.LB
 |   |   |   |   |- FAK_B_0002.LB
 |   |   |   |   |- FAK_B_0003.LB
 |   |   |   |   |- FAK_B_0004.LB
 |   |   |   |   |- FAK_B_0005.LB
 |   |   |   |   |- FAK_B_0006.LB
 |   |   |   |   |- FAK_B_0007.LB
 |   |   |   |   |- FAK_B_0008.LB
 |   |   |   |   |- FAK_B_0009.LB
 |   |   |   |   |- FAK_B_0010.LB
 |   |   |   |   |- FAK_B_0011.LB
 |   |   |   |   |- FAK_B_0012.LB
 |   |   |   |   |- FAK_B_0013.LB
 |   |   |   |   |- FAK_B_0014.LB
 |   |   |   |   |- FAK_B_0015.LB
 |   |   |   |   |- FAK_B_0016.LB
 |   |   |   |   |- FAK_B_0017.LB
 |   |   |   |   |- FAK_B_0018.LB
 |   |   |   |   |- FAK_B_0019.LB
 |   |   |   |   |- FAK_B_0020.LB
 |   |   |   |   |- FAK_B_0021.LB
 |   |   |   |   |- FAK_B_0022.LB
 |   |   |   |   |- FAK_B_0023.LB
 |   |   |   |   |- FAK_B_0024.LB
 |   |   |   |   |- FAK_B_0025.LB
 |   |   |   |- MA2                    2 labelfiles for recognition
 |   |   |   |   |- FAK_MA2_01.LB
 |   |   |   |   |- FAK_MA2_02.LB
 |   |   |- WAV
 |   |   |   |- B                      25 wavefiles for adaptation
 |   |   |   |   |- FAK_B_0001.12K
 |   |   |   |   |- FAK_B_0002.12K
 |   |   |   |   |- FAK_B_0003.12K
 |   |   |   |   |- FAK_B_0004.12K
 |   |   |   |   |- FAK_B_0005.12K
 |   |   |   |   |- FAK_B_0006.12K
 |   |   |   |   |- FAK_B_0007.12K
 |   |   |   |   |- FAK_B_0008.12K
 |   |   |   |   |- FAK_B_0009.12K
 |   |   |   |   |- FAK_B_0010.12K
 |   |   |   |   |- FAK_B_0011.12K
 |   |   |   |   |- FAK_B_0012.12K
 |   |   |   |   |- FAK_B_0013.12K
```

```
|   |   |   |   |- FAK_B_0014.12K
|   |   |   |   |- FAK_B_0015.12K
|   |   |   |   |- FAK_B_0016.12K
|   |   |   |   |- FAK_B_0017.12K
|   |   |   |   |- FAK_B_0018.12K
|   |   |   |   |- FAK_B_0019.12K
|   |   |   |   |- FAK_B_0020.12K
|   |   |   |   |- FAK_B_0021.12K
|   |   |   |   |- FAK_B_0022.12K
|   |   |   |   |- FAK_B_0023.12K
|   |   |   |   |- FAK_B_0024.12K
|   |   |   |   |- FAK_B_0025.12K
|   |   |   |- MA2                          2 wavefiles for recognition
|   |   |   |   |- FAK_MA2_01.12K
|   |   |   |   |- FAK_MA2_02.12K
|- LIST
|   |- AdaptWords25.sss                     example label file for adaptation
|   |- mset2.FAK.sss                        example label file for recognition
|- REF_RESULT                               contains example run
|   |- 250
|   |   |- error.bunsetsu2                  stderr output
|   |   |- result.bunsetsu                  raw bunsetsu results
|   |   |- result.bunsetsu.score            bunsetsu score
|   |   |- result.bunsetsu.tmp
|   |   |- result.bunsho                    raw bun results
|   |   |- result.bunsho.romaji.score       bun (romaji correct) score
|   |   |- result.bunsho.score              bun (kanji correct) score
|   |- HMnet.450
|   |- HMnet.450.IOP
|   |- HMnet.450.log
|- SCRIPT
|   |- bunsetsu.awk                         for bunsetsu scoring
|   |- bunsho.awk                           for bun scoring
|   |- make_filelist.awk                    constructing list of files for speechlabel
|   |- make_seqlist.awk                     for adaptation
|   |- run.csh                              main script
|- runmain.csh                              script calling run.csh
```

Source code for ATREUS/EVAL/runmain.csh

```
#!/bin/csh -f
# runmain.csh
set HERE = `pwd`
set DEMO = $HERE/../DEMO

@ input = 1
@ adapt = 2
set S = (       \
                "( FAK FYM )" \
)

@ i = 1
while ( $i <= $#S)
  set p = $S[$i]
  set SPEAKER = $p[$input]
  set ADAPT_SPEAKER = $p[$adapt]

  foreach ADAPT ( Words25 )
    # adaptation
    SCRIPT/run.csh -g ADAPTATION \
      -sm $DEMO/HMnet/$ADAPT_SPEAKER/HMnet \
      -af $HERE/LIST/Adapt$ADAPT.sss -aw $HERE/DATA/$SPEAKER/WAV/B \
      -al $HERE/DATA/$SPEAKER/LBL/B \
      -si $SPEAKER -t RESULT
   foreach GLOBAL_BEAM ( 250 )
      # SSS-LR recognition
      SCRIPT/run.csh -g RECOGNITION \
        -rf $HERE/LIST/mset2.$SPEAKER.sss -rd $GLOBAL_BEAM \
        -rw $HERE/DATA/$SPEAKER/WAV -rl $HERE/DATA/$SPEAKER/LBL \
        -G $GLOBAL_BEAM \
        -lr POST -bn $DEMO/Gra/cstar_phrase.sss -bk $DEMO/Gra/cstar_sentence \
        -si $SPEAKER -t RESULT
      # scoring
```

```
      SCRIPT/run.csh -g SCORE \
        -rf $HERE/LIST/mset2.$SPEAKER.sss -rd $GLOBAL_BEAM \
        -rw $HERE/DATA/$SPEAKER/WAV -rl $HERE/DATA/$SPEAKER/LBL \
        -G $GLOBAL_BEAM \
        -lr POST -bn $DEMO/Gra/cstar_phrase.sss -bk $DEMO/Gra/cstar_sentence \
        -si $SPEAKER -t RESULT
    end         # GLOBAL_BEAM
  end           # ADAPT
  @ i ++
end # S

### EOF ###
```

Source code for ATREUS/EVAL/SCRIPT/run.csh

```
#!/bin/csh -f
# run.csh
# example:
#       run.csh -g ADAPTATION
#       run.csh -g RECOGNITION
#       run.csh -g SCORE

set com = $0
onintr CLEANUP

# which machine is this?
switch (`/bin/uname`)
case  HP-UX:
        setenv MACH HP; breaksw
case ULTRIX:
        setenv MACH DEC; breaksw
case SunOS:
        setenv MACH SUN; breaksw
case OSF1:
        setenv MACH ALPHA; breaksw
default:
        breaksw
endsw

set ATREUS = `pwd`/..

### directories
set EVALDIR = `pwd`
set PROGDIR = `pwd`/../BINHP
set GRADIR = $EVALDIR/GRA               # cp eset_phrase.sss.* eset_sentence.*
set AWKDIR = $EVALDIR/SCRIPT            # cp bunsetsu.awk bunsho.awk make_filelist.awk make_seqlist.awk
set ADAPTLISTDIR = $EVALDIR/LIST        # cp AdaptWords*
set SEIKAILISTDIR = $EVALDIR/LIST       # cp eset.*.sss mset.*.sss
set MODELDIR = $ATREUS                  # cp -r MHT FYM

### default values              other typical values
set GOTO = RECOGNITION          # ADAPTATION    SCORE
set MODE = MSET                 # ESET
set GLOBAL_BEAM = 100           # 48            250
set DURANGLE = 0.6
set DURBREAK = 3.0
set DURFAT = 2.2
set VAR_FAT = 1.8

set SPEAKER = FAK

set STANDARD_MODEL = $MODELDIR/FYM/mix_1/HMnet          # from Takami

set ADAPTLIST    = $ADAPTLISTDIR/AdaptWords25.sss       # singer 3line format
set ADAPT_WAVDIR =  /NFS/atrfs4/wave51/$SPEAKER/WAV/B
set ADAPT_LBLDIR =  /NFS/atrfs4/wave51/$SPEAKER/LBL/B

if($MODE == "MSET") then

  set RECOGLIST    = $SEIKAILISTDIR/mset.$SPEAKER.sss  # singer 3line format
  set RECOG_WAVDIR = /NFS/atrfs4/wave51/$SPEAKER/WAV   # clean data
```

```
  set RECOG_LBLDIR = /NFS/atrfs4/wave51/$SPEAKER/LBL

else

  set RECOGLIST    = $SEIKAILISTDIR/eset.$SPEAKER.sss   # singer 3line format
  set RECOG_WAVDIR = /NFS/atrfs4/wave53/$SPEAKER/WAV/EP
  set RECOG_LBLDIR = /NFS/atrfs4/wave53/$SPEAKER/LBL/EP

endif

set GRAMMAR =    $GRADIR/eset_phrase.sss
set BUNSHO_GRA = $GRADIR/eset_sentence
set BUNSHO_CONTEXT = $GRADIR/gomi

set RESDIR = "$MODE"_"$GLOBAL_BEAM"

set LRTYPE = POST               # 2DANLR

while($#argv)
switch($argv[1])
        case -af*:
                set ADAPTLIST = $argv[2];shift;breaksw
        case -aw*:
                set ADAPT_WAVDIR = $argv[2];shift;breaksw
        case -al*:
                set ADAPT_LBLDIR = $argv[2];shift;breaksw
        case -bn*:
                set GRAMMAR = $argv[2];shift;breaksw
        case -bk*:
                set BUNSHO_GRA = $argv[2];shift;breaksw
        case -bc*:
                set BUNSHO_CONTEXT = $argv[2];shift;breaksw
        case -da*:
                set DURANGLE = $argv[2];shift;breaksw
        case -db*:
                set DURBREAK = $argv[2];shift;breaksw
        case -df*:
                set DURFAT = $argv[2];shift;breaksw
        case -g*:
                set GOTO = $argv[2];shift;breaksw
        case -G*:
                set GLOBAL_BEAM = $argv[2];shift;breaksw
        case -lr*:
                set LRTYPE = $argv[2];shift;breaksw
        case -m*:
                set MODE = $argv[2];shift;breaksw
        case -rf*:
                set RECOGLIST = $argv[2];shift;breaksw
        case -rw*:
                set RECOG_WAVDIR = $argv[2];shift;breaksw
        case -rl*:
                set RECOG_LBLDIR = $argv[2];shift;breaksw
        case -rd*:
                set RESDIR = $argv[2];shift;breaksw
        case -si*:
                set SPEAKER = $argv[2];shift;breaksw
        case -sm*:
                set STANDARD_MODEL = $argv[2];shift;breaksw
        case -t*:
                set TMPDIR = $argv[2];shift;breaksw
        case -v*:
                set VAR_FAT = $argv[2];shift;breaksw
        default:
                echo "ERROR:$com $argv"
                echo " [-af ADAPTLIST]       VAL: $ADAPTLIST"
                echo " [-aw ADAPT_WAVDIR]    VAL: $ADAPT_WAVDIR"
                echo " [-al ADAPT_LBLDIR]    VAL: $ADAPT_LBLDIR"
                echo " [-bn GRAMMAR]         VAL: $GRAMMAR"
                echo " [-bk BUNSHO_GRA]      VAL: $BUNSHO_GRA"
                echo " [-bc BUNSHO_CONTEXT]  VAL: $BUNSHO_CONTEXT"
                echo " [-da DURANGLE]        VAL: $DURANGLE"
                echo " [-db DURBREAK]        VAL: $DURBREAK"
                echo " [-df DURFAT]          VAL: $DURFAT"
                echo " [-g GOTO]             VAL: $GOTO"
```

```
                echo " [-G GLOBAL_BEAM]      VAL: $GLOBAL_BEAM"
                echo " [-lr LRTYPE]          VAL: $LRTYPE"
                echo " [-m MODE]             VAL: $MODE"
                echo " [-rf RECOGLIST]       VAL: $RECOGLIST"
                echo " [-rw RECOG_WAVDIR]    VAL: $RECOG_WAVDIR"
                echo " [-rl RECOG_LBLDIR]    VAL: $RECOG_LBLDIR"
                echo " [-rd RESDIR]          VAL: $RESDIR"
                echo " [-si SPEAKER]         VAL: $SPEAKER"
                echo " [-sm STANDARD_MODEL]  VAL: $STANDARD_MODEL"
                echo " [-t TMPDIR]           VAL: $TMPDIR"
                echo " [-v VAR_FAT]          VAL: $VAR_FAT"
                exit 1
        endsw
shift
end

if (! $?TMPDIR) then
        set TMPDIR = $SPEAKER
endif

# fixed variables
set FRAME_SHIFT = 10            ##set FRAME_SHIFT = 5
set MARGIN = 5                  ##set MARGIN = 10
set HMNET_SIZE = 450

set DURMODEL   =     $ATREUS/DEMO/Dur/HMnet.100
set LOG_TABLE  =     $ATREUS/DEMO/ETC/log.tbl.HP

# temporary files
set SAMPLELIST = .samplelist                    # takami adaptation format
set FILELIST = .filelist                        # singer format 2
set MODELALL = .tmpmodel
set BUNSETSU_CORRECT = .bunsetsu        # created from RECOGLIST
set BUNSHO_CORRECT = .bunsho            # created from RECOGLIST

# results
set RESULT_2DAN = $RESDIR/result.2dan
set RESULT_POST = $RESDIR/result.bunsho
set RESULT_LR = $RESDIR/result.bunsetsu
set ERROR_LR1 = $RESDIR/error.bunsetsu1
set ERROR_LR2 = $RESDIR/error.bunsetsu2

# programs
set MAKE_SEQLIST = $AWKDIR/make_seqlist.awk
set MAKE_FILELIST = $AWKDIR/make_filelist.awk
set BUNSETSU_AWK = $AWKDIR/bunsetsu.awk
set BUNSHO_AWK = $AWKDIR/bunsho.awk
set SPLIT_AWK = $AWKDIR/split2dan.awk

set SPEECHLABEL = $PROGDIR/speechlabel
set WAVEPARA = $PROGDIR/WavePara34
set RMHEADER = $PROGDIR/rmheader
set MAKEWORD = $PROGDIR/make_word_sample
set MAKE_SEQLIST_CMD = $PROGDIR/make_seqlist
set ADAPT = $PROGDIR/Exe.adapt_HMnet
set VARCONV = $PROGDIR/var_conv
set ALLOCDSP = $PROGDIR/alloc_DSP
set SSSLIKE = $PROGDIR/cal_frame_prob
set SSSLR = $PROGDIR/SSS-LR.exe
set SSS2DANLR = $PROGDIR/SSS-LR.twolevel.exe
set REORDER = $PROGDIR/reorder
set POSTFILTER = $PROGDIR/Postfilter
set SCORE = $PROGDIR/score_SSS

if (! -e $TMPDIR)          mkdir $TMPDIR
cd $TMPDIR

goto $GOTO

ADAPTATION:
  # get list of wavefiles with start and endpoints
  gawk -f $MAKE_FILELIST speaker=$SPEAKER wavdir=$ADAPT_WAVDIR lbldir=$ADAPT_LBLDIR\
       mode="ADAPT" $ADAPTLIST > $FILELIST
```

```
  #construct separate label files for each adaption word
  gawk -f $MAKE_SEQLIST make=$MAKE_SEQLIST_CMD $ADAPTLIST

  # preprocessing of adaptation words
  ADAPT_PREPRO:
  (     cat $FILELIST |\
        $SPEECHLABEL -d |\
        $WAVEPARA -fs $FRAME_SHIFT |\
        $RMHEADER -o lpc )

  # construct samplelist for adaptation
  SAMPLELIST:
  set MAXCOUNT = `wc $FILELIST | nawk '{ print $1 }'`
  @ COUNT = 0
  if ( -e $SAMPLELIST) rm $SAMPLELIST
  while ( $COUNT < $MAXCOUNT )
        cat >> $SAMPLELIST << EOF
        $SPEAKER
        seqlist$COUNT
        1
        lpc$COUNT 34
EOF
        @ COUNT ++
  end


  RUN_ADAPTATION:
  set MAXCOUNT = `wc $FILELIST | nawk '{ print $1 }'`
  echo $STANDARD_MODEL | sed 's/HMnet$/HMM.all/' > $MODELALL
  # smoothing rate is different according to the number of adaptation words
  if ( $MAXCOUNT <= 10 ) then
        set SMOOTHING_RATE = 30
  else if ( $MAXCOUNT <= 25 ) then
        set SMOOTHING_RATE = 20
  else if ( $MAXCOUNT <= 200 ) then
        set SMOOTHING_RATE = 10
  else
        set SMOOTHING_RATE = 1
  endif
  # don't swap bytes on HP
  if( $MACH == DEC || $MACH == ALPHA ) then
        set SWAPBYTE = 1
  else
        set SWAPBYTE = 0
  endif
  $MAKEWORD -sb $SWAPBYTE -sl $SAMPLELIST |\
  $ADAPT -if $STANDARD_MODEL.$HMNET_SIZE -il $MODELALL \
         -of HMnet.$HMNET_SIZE -kn 6 -sr $SMOOTHING_RATE

  # variance fattening and allocation of DSP's for IOPBOX
  cat HMnet.$HMNET_SIZE  | $VARCONV $VAR_FAT | $ALLOCDSP > HMnet.$HMNET_SIZE.IOP

  goto CLEANUP

RECOGNITION:
  if (! -e $RESDIR ) mkdir $RESDIR

  # variance fattening and allocation of DSP's for IOPBOX
  cat HMnet.$HMNET_SIZE  | $VARCONV $VAR_FAT | $ALLOCDSP > HMnet.$HMNET_SIZE.IOP
  set NEWMODEL =  HMnet.$HMNET_SIZE.IOP

  # get list of wavefiles with start and endpoints
  gawk -f $MAKE_FILELIST speaker=$SPEAKER mode=$MODE \
     wavdir=$RECOG_WAVDIR lbldir=$RECOG_LBLDIR $RECOGLIST > $FILELIST

  if($LRTYPE == "2DANLR") then
  (     cat $FILELIST |\
        $SPEECHLABEL -d |\
        $WAVEPARA -fs $FRAME_SHIFT |\
        $SSSLIKE -D 1 -m $NEWMODEL -l $LOG_TABLE |\
        $SSS2DANLR -romaji+kanji -G $BUNSHO_GRA -n 10 \
          -L $LOG_TABLE -g $GRAMMAR -B $GLOBAL_BEAM -m $NEWMODEL \
          -D $DURMODEL -N $DURFAT -K 1 -P $FRAME_SHIFT -b 64 \
          -a $DURBREAK -R $DURANGLE -c 3000 -A $MARGIN .flag |\
```

```
        grep -v "^#x"  > $RESULT_2DAN ) >& $ERROR_LR2
    else
    (   cat $FILELIST |\
        $SPEECHLABEL -d |\
        $WAVEPARA -fs $FRAME_SHIFT |\
        $SSSLIKE -D 1 -m $NEWMODEL -l $LOG_TABLE |\
        $SSSLR -n 10 -L $LOG_TABLE -g $GRAMMAR -B $GLOBAL_BEAM -m $NEWMODEL \
          -D $DURMODEL -N $DURFAT -K 1 -P $FRAME_SHIFT -b 64 \
          -a $DURBREAK -R $DURANGLE -c 3000 -A $MARGIN .flag          |\
        grep -v "^#x" |\
        $REORDER | tee $RESULT_LR |\
        $POSTFILTER -g $BUNSHO_GRA -romaji+kanji -t > $RESULT_POST ) >& $ERROR_LR2
    endif
    goto CLEANUP

SCORE:
    # split bunsetsu and bun results from 2DAN-LR
    if($LRTYPE == "2DANLR") then
      gawk -f $SPLIT_AWK bunsetsu=$RESULT_LR bun=$RESULT_POST $RESULT_2DAN
    endif

    # scoring program for bunsetsu
    gawk '/^#/    { NR--; next } NR%3==0 { for(i=1;i<=NF;i++) print $i}' $RECOGLIST \
        > $BUNSETSU_CORRECT
    # special treatment for (Japanese) iu -> juu
    cat $RESULT_LR  |\
      sed  -e 's/\([ /]\)iu\([ /]\)/\1juu\2/g'\
           -e 's/\([ /]\)dooiu\([ /]\)/\1doojuu\2/g'\
           -e 's/\([ /]\)sooiu\([ /]\)/\1soojuu\2/g' |\
      gawk -f $BUNSETSU_AWK correctfile="$BUNSETSU_CORRECT" |\
      sed '/^[ \t]*[0-9]/s/\([^(]\)-/\1q/g' > $RESULT_LR.tmp
    $SCORE -l $RESULT_LR.tmp > $RESULT_LR.score

    # scoring program for bunsho (kanji)
    gawk '/^#/  { NR--; next } NR%3==2 { for(i=1;i<NF;i++) printf"%s#",$i; printf"%s\n",$NF}' \
        $RECOGLIST > $BUNSHO_CORRECT
    gawk -f $BUNSHO_AWK type="#j" bunsholist=$BUNSHO_CORRECT $RESULT_POST > $RESULT_POST.score

    # scoring program for bunsho (romaji)
    gawk '/^#/    { NR--; next } NR%3==0 { for(i=1;i<NF;i++) printf"%s#",$i; printf"%s\n",$NF}'\
         $RECOGLIST > $BUNSHO_CORRECT
    # special treatment for (Japanese) iu -> juu
    cat $RESULT_POST |\
      sed  -e 's/-/q/g' -e 's/\///g'\
           -e 's/\([ #]\)iu\([ #]\)/\1juu\2/g'\
           -e 's/\([ #]\)dooiu\([ #]\)/\1doojuu\2/g'\
           -e 's/\([ #]\)sooiu\([ #]\)/\1soojuu\2/g' |\
      gawk -f $BUNSHO_AWK type="#o" bunsholist=$BUNSHO_CORRECT > $RESULT_POST.romaji.score

    goto CLEANUP

CLEANUP:
        echo CLEANUP
        rm -f lpc*
        rm -f seqlist*
exit 0
### EOF ###
```

4 IOPBOX

To speed up the recognition process, a DSP implementation was developed which combines the 3 modules SPEECHIN, LPC and SSSLIKE. It was jointly developed with SDS (Systems Design Service Corporation, Tokyo). Especially the low-level parts, like drivers, were only coded by SDS. The following explanation is therefore very superficial and does not cover the standard installation for the IOPBOX like kernel reconfiguration etc..

The IOPBOX at ATR contains 12 TMS320C30 DSPs, which can deliver a maximum

speed of about 400 MFLOP. 1 DSP is responsible for communication with the host and calculation of feature vector (software module LPC), the other 11 DSPs calculate likelihood values for each state in the HMnet (software module SSSLIKE). DSP source codes are in directories ATREUS/IOP_ROOT and ATREUS/IOP_LIKE. To compile these programs, the TI compiler asm30, lnk30, cl30 are needed.

The devicename for the IOPBOX is /dev/das0. The special driver interface was installed as

```
$ mkdir -p /usr/local/srtp
$ cp ATREUS/IOP_LOWLEVEL/iop.x /usr/local/srtp
```

The new module which replaces SPEECHIN, LPC and SSSLIKE is called PARAIN. Its options and user interface are identical to SpeechPress. However, before running ParaIn, DSP programs, log table and HMnet have to be downloaded from the host and the DSP programs have to be started. This is performed by program DSP.

If everything is installed properly, you should be able to run the demo with the IOPBOX by choosing the option HARD in the options menu. For further details please consult the source codes.

5　不特定話者 HMnet の使用法

ここでは不特定話者 HMnet の使用法について説明する。別売の不特定話者混合連続分布型 HMnet 作成ツールに含まれる HMnet を用いることにより、混合連続分布型 HMnet(CM-HMnet) を用いた不特定話者音声認識が可能になる。CM-HMnet はソフトバージョンで動作が確認されているが IOPBOX には対応していない。

以下に使用法について説明する。不特定話者 HMnet は不特定話者混合連続分布型 HMnet 作成ツールをインストールしたときの、以下のディレクトリに存在する。

```
Model/mix12mf/matrix.200        # 男女話者用、状態数 200、混合数 12
Model/mix12mf/matrix.600        # 男女話者用、状態数 600、混合数 12
Model/mix12m/matrix.200         # 男性話者用、状態数 200、混合数 12
Model/mix12m/matrix.600         # 男性話者用、状態数 600、混合数 12
Model/mix12f/matrix.200         # 女性話者用、状態数 200、混合数 12
Model/mix12f/matrix.600         # 女性話者用、状態数 600、混合数 12
```

また不特定話者混合連続分布型 HMnet 作成ツールのサンプルプログラムを動作することにより、混合数 2 の HMnet が以下のディレクトリに生成される。

```
Outmatrix/mix2/matrix.200.NEW    # 2 混合、状態数 200 の HMnet
```

ATREUS/DEMO/Data/independent というディレクトリを作成した後に、いずれかの HMnet を HMnet." 状態数" という名前に変えてコピーする。以下は 200 状態の例。

```
$ mkdir -p ATREUS/DEMO/Data/independent/Adapted_HMnet
$ cp 不特定 HMnet 名 ATREUS/DEMO/Data/independent/Adapted_HMnet/HMnet.200
```

SSS-LR を起動後話者を independent と指定し、さらに Options により以下のオプションを設定する。

- HMnet_Size を 200 に設定
- GLOBAL_BEAM を 128 ～ 256 に設定する（大きいほど認識性能は向上するが認識時間がかかる）。
- —return—

以上を設定後に認識を行なう。