TR-I-0246

HMM－LRユーザーズ・マニュアル
HMM－LR User's Manual

北　研二
Kenji Kita

1992.3

概要

本レポートでは、HMM－LR音声認識システムに関係する様々なプログラムについて説明しています。特に、下記の項目に対して、その使用法を説明しています。

1. 文法開発に関する種々のプログラム。
2. HMM－LR音声認識システム。
3. 2段階LR法に基づくHMM－LR音声認識システム。

ATR自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

目次

第１章

はじめに

このテクニカル・レポートは、HMM - ＬＲ音声認識システムに関係する様々なプログラムの使用法について説明しています。HMM - ＬＲ音声認識システムは、名前の通り、HMM (Hidden Markov Model) とＬＲパーザを統合化した音声認識システムで、文脈自由文法に基づく統語的な制約を用いて、効率よく入力された音声データを認識することができます。

本レポートでは、以下の項目について述べています。

1. 文法開発に関する種々のプログラム。
2. HMM - ＬＲ音声認識システム。
3. ２段階ＬＲ法に基づくHMM - ＬＲ音声認識システム。

以下で説明するプログラムは、すべて atr-dp 上にあります。実行ファイルのあるディレクトリは、

/ifg2/LR/EXE

です。以下では、実行ファイル名、あるいはソース・ファイル名（または、ディレクトリ）をそれぞれ [exe]、[src] で表しています。例えば、

[exe] xyz
[src] /ifg2/LR/XYZ

は、プログラムの実行ファイル名が xyz で、ディレクトリ /ifg2/LR/XYZ にそのソース・ファイルがあることを示しています。

第 2 章

文法の開発

HMM - ＬＲ音声認識システム、あるいは２段階ＬＲ法に基づくHMM - ＬＲ音声認識システム（以下、２段階HMM - ＬＲと略記）は、音声認識の際に、文脈自由文法に基づく統語的な制約を用いています。

この章では、文法の書き方、文法を開発する際に役立つ種々のプログラムについて説明します。

2.1 音韻表記

音声認識に用いる文法では、終端レベルで単語の音韻表記が与えられています。以下に、音韻表記の一覧表を示します。

あ行	a	i	u	e	o
か行	k a	k i	k u	k e	k o
さ行	s a	sh i	s u	s e	s o
た行	t a	ch i	ts u (つ) Q (っ)	t e	t o
な行	n a	n i	n u	n e	n o
は行	h a	h i	h u	h e	h o
ま行	m a	m i	m u	m e	m o
や行	y a		y u		y o
ら行	r a	r i	r u	r e	r o
わ行	w a				o
ん	=				
が行	g a	g i	g u	g e	g o
ざ行	z a	z i	z u	z e	z o
だ行	d a	z i	z u	d e	d o
ば行	b a	b i	b u	b e	b o
ぱ行	p a	p i	p u	p e	p o
きゃ行	ky a		ky u		ky o
ぎゃ行	gy a		gy u		gy o
しゃ行	sh a		sh u		sh o
じゃ行	zy a		zy u		zy o
ちゃ行	ch a		ch u		ch o
にゃ行	ny a		ny u		ny o
ひゃ行	hy a		hy u		hy o
びゃ行	by a		by u		by o
ぴゃ行	py a		py u		py o
みゃ行	my a		my u		my o
りゃ行	ry a		ry u		ry o
長母音	aa	ii	uu	ee ei	oo ou

2.2 文法の記述形式

文法記述の例を以下に示します。

文法の記述例（test.gra）

```
;;-=-=-=-=-=-=-=-=-=-=-=-=-
;;      Sample CFG
;;-=-=-=-=-=-=-=-=-=-=-=-=-
(<start> <--> (<s>))
(<s> <--> (<np> <v>))
(<s> <--> (<v>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

- セミコロン（;）で始まる行はコメントです。セミコロンから行の終りまでがコメントであるとみなされ、プログラムでは無視されます。
- 非終端記号は < と > で囲みます。それ以外の記号は終端記号とみなされます。上の例では、<s>，<np>，<v>，･･･が非終端記号で、k，o，r，･･･は終端記号です。[1]
- 文法の出発記号（start symbol）は <start> で、しかも出発記号を含む規則は一番最初に書かなければいけません。[2]
- 文法を処理するプログラムでは、大文字と小文字を区別していません。従って、<np> と <NP> は同じものとして扱われます。

[1]実際のプログラムでは、ある記号が非終端記号であるかどうかを、記号の最初の文字が < であるかどうかだけで判断しています。従って、必ずしも非終端記号を < と > で囲む必要はありません。しかし、読み易さという点から、< と > で囲んだ方がよいでしょう。

[2]より正確にいうと、文法は拡大文法（augmented grammar）の形式でなければなりません。拡大文法とは、S を出発記号とする文法 G に新しい出発記号 S' および規則 $S' \rightarrow S$ を付け加えた文法のことです。上の例では、<start> が S' に当たります。文法を処理するプログラムは、<start> を含む規則を 0 番目の規則として扱っています。

2.3 ＬＲテーブル作成

[exe] slr
[src] /ifg2/LR/LRconst/SLR

プログラム slr は、文脈自由文法をＬＲテーブルに変換するプログラムです。ＬＲテーブルには色々な種類のものがありますが、slr は単純ＬＲ（simple LR, SLR）と呼ばれるものを出力します。

例えば、test.gra という文法ファイルからＬＲテーブルを作るには、

% slr test

というようにします。この場合、プログラムは自動的に拡張子 .gra を補います。また、新たに作成された LR テーブルには .tab という拡張子が付けられます。

slr の出力例（test.tab）

```
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;;
;;      SLR parsing table for HMM-LR
;;      ----------------------------
;;      Created on Fri Jan 24 14:09:31 1992
;;      Constructor version: V0.7
;;
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
(slr-table)
(0 (k s3) (o s4) (<n> g1) (<np> g2) (<v> g5) (<s> g6))
(1 (o s7) (k r3) (o r3) (<p> g8))
(2 (k s9) (o s4) (<v> g10))
(3 (o s11) (u s12))
(4 (k s13))
(5 ($ r2))
(6 ($ a))
(7 (k r6) (o r6))
(8 (k r4) (o r4))
.....
```

見て分かるように、slr の出力は、

(状態番号 (文法記号１ 動作１) (文法記号２ 動作２) …)

という形の行から構成されています。「動作」項の最初の１文字で、移動（shift）、還元（reduce）、受理（accept）のいずれかであるかを示しています。

2.4 ２段階用ＬＲテーブル作成

[exe] slr2
[src] /ifg2/LR/LRconst/SLR2

プログラム slr2 は、２段階HMM - ＬＲ音声認識システムで用いる文節内ＬＲテーブルを作成します。このプログラムによって作成されたテーブルには、各動作項に到達可能な文節カテゴリ名が与えられています。

slr2 を動かすためには、文法ファイル（拡張子 .gra）以外に、文節カテゴリ名をリストしたファイル（拡張子 .cat）が必要です。これら２つのファイルを用意した後、

```
% slr2 bunsetu
```

のようにしてＬＲテーブルを作成します。この場合、bunsetu.gra, bunsetu.cat という２つのファイルから bunsetu.tab というＬＲテーブル・ファイルが得られます。

文節内文法の例（bunsetu.gra）

```
(<start> <--> (<bunsetu>))
(<bunsetu> <--> (<np>))
(<bunsetu> <--> (<vp>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<vp> <--> (<v>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

文節カテゴリ・リストの例（bunsetu.cat）

```
<np>
<vp>
```

slr2 の出力例 (bunsetu.tab)

```
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;;
;;      SLR parsing table for HMM-LR
;;      ----------------------------
;;      Created on Mon Feb  3 13:46:52 1992
;;      Constructor version: V0.7TL
;;
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
(slr-table)
(0 (k s3 [ <vp> <np> ] ) (o s4 [ <vp> ] )
   (<n> g1) (<np> g2) (<v> g5) (<vp> g6) (<bunsetu> g7))
(1 (o s8 [ <np> ] ) ($ r3) (<p> g9))
(2 ($ r1))
(3 (o s10 [ <np> ] ) (u s11 [ <vp> ] ))
(4 (k s12 [ <vp> ] ))
(5 ($ r5))
(6 ($ r2))
(7 ($ a))
(8 ($ r7))
(9 ($ r4))
(10 (r s13 [ <np> ] ))
(11 (r s14 [ <vp> ] ))
(12 (u s15 [ <vp> ] ))
(13 (e s16 [ <np> ] ))
(14 (e s17 [ <vp> ] ))
(15 (r s18 [ <vp> ] ))
(16 (o r6) ($ r6))
(17 ($ r8))
(18 (e s19 [ <vp> ] ))
(19 ($ r9))
```

上の例では、状態 0 で、音韻 /k/ は文節カテゴリ <vp> または <np> に到達可能であり、音韻 /o/ は <vp> に到達可能であることを示しています。

2.5 音韻列ＬＲパーザ

[exe] lr
[src] /ifg2/LR/LR

プログラム lr は、簡単なＬＲパーザです。与えられた音韻列が、文法によって受理されるかどうかといったことを調べるのに用います。

例えば、test.gra, test.tab という文法とＬＲテーブルを用いて、test.input という音韻列ファイルを解析するには、

% lr -g test test.input

または

% lr test.input -g test

というようにします（-g の後に文法の名前を付けます）。

音韻列ファイルの例（test.input）

```
k o r e o k u r e
o k u r e k u r e
```

lr の出力例

```
INPUT: k o r e o k u r e  -->  Success
INPUT: o k u r e k u r e  -->  Fail
TOTAL = 2, PARSED = 1
```

上の例の場合、音韻列 “k o r e o k u r e” は受理されるが、“o k u r e k u r e” は受理されないことを示しています。

また、

```
% lr -g test test.input -t
```

のように -t オプションを付ければ、音韻列を解析するのに用いられた、文法規則の履歴を出力することができます。

lr の出力例（-t オプション使用時）

```
INPUT: k o r e o k u r e  -->  Success
[1]
  (5) <n> --> k o r e
  (6) <p> --> o
  (4) <np> --> <n> <p>
  (7) <v> --> k u r e
  (1) <s> --> <np> <v>
[2]
  (5) <n> --> k o r e
  (3) <np> --> <n>
  (8) <v> --> o k u r e
  (1) <s> --> <np> <v>
INPUT: o k u r e k u r e  -->  Fail
TOTAL = 2, PARSED = 1
```

音韻列 “k o r e o k u r e” に対しては、2 つの導出（derivation）があることが分かります。

構文解析木を表示するためには、-T オプションを用い、

% lr -g test test.input -t

のようにします。[3]

lr の出力例（-T オプション使用時）

```
INPUT: k o r e o k u r e  -->  Success
[1]
<s>
|--<np>
|   |--<n>
|   |   |--kore
|   |--<p>
|       |--o
|--<v>
    |--kure

[2]
<s>
|--<np>
|   |--<n>
|       |--kore
|--<v>
    |--okure

INPUT: o k u r e k u r e  -->  Fail
TOTAL = 2, PARSED = 1
```

[3]構文解析木の表示プログラムは、データ処理研究室の田代氏が作ってくれました。

2.6 文法中の音韻変換

[exe] convgra
[src] /ifg2/LR/GrammarTool/ConvGra

プログラム convgra は、文法中の音韻名を変換し、HMM - LR音声認識システムで用いられている音韻名に直します。

例えば、sample.gra という文法ファイル中の音韻名を変換して、sample2.gra というファイルを作るには、

```
% convgra sample.gra > sample2.gra
```

というようにします。

変換前の文法 (sample.gra)

```
(<start> <--> (<s>))
(<s> <--> (sh i = g ou))  ;; shiNgou
(<s> <--> (sh o r i))     ;; shori
```

変換後の文法 (sample2.gra)

```
(<start> <--> (<s>))
(<s> <--> (sh i2 = g ou))
(<s> <--> (sy o r i))
```

2.7 文法と語彙情報の分離

[exe] grmake
[src] /ifg2/LR/GrammarTool/Grmake

プログラム grmake は、単語の漢字表記、品詞等の語彙情報を含んだ文法ファイルを読み込み、これを文法と語彙情報に分離します。grmake では、入力ファイルは .lex という拡張子を持つと仮定しており、.gra（文法）と .dic（語彙情報）という拡張子の付いた２つのファイルを作り出します。

使い方は、

```
% grmake test
```

のようにします。この場合、test.lex というファイルを読み込み、test.gra と test.dic という２つのファイルを作成します。

語彙情報を含んだ文法の例（test.lex）

```
(<start> <--> (<s>))
(<s> <--> (<np> <v>))
(<s> <--> (<v>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<n> <--> (k o r e)     ("これ"    "代名詞"))
(<p> <--> (o)           ("を"      "格助詞"))
(<v> <--> (k u r e)     ("くれ"    "動詞"))
(<v> <--> (o k u r e)   ("おくれ"  "動詞"))
```

文法だけを含んだファイル (test.gra)

```
(<start> <--> (<s>))
(<s> <--> (<np> <v>))
(<s> <--> (<v>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

語彙情報だけを含んだファイル (test.dic)

```
5 これ　代名詞
6 を　　格助詞
7 くれ　動詞
8 おくれ 動詞
```

語彙情報だけを含んだファイルの各行の先頭には、何番目の文法規則に対応するものであるかという規則の番号が付けられています。

2.8 文節間文法と文節内文法のマージ

[exe] mergegra
[src] /ifg2/LR/GrammarTool/MergeGra

文節間文法と文節内文法をマージして文の文法を作るには、プログラム mergegra を使います。

例えば、文節間文法 bun.gra と文節内文法 bunsetu.gra をマージして、文の文法 sent.gra を得るためには、

```
% mergegra bun.gra bunsetu.gra > sent.gra
```

というようにします。

文節間文法の例（bun.gra）

```
(<start> <--> (<s>))
(<s> <--> (<np> <vp>))
(<s> <--> (<vp>))
(<np> <--> (np))
(<vp> <--> (vp))
```

文節内文法の例（bunsetu.gra）

```
(<start> <--> (<bunsetu>))
(<bunsetu> <--> (<np>))
(<bunsetu> <--> (<vp>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<vp> <--> (<v>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

文の文法 (sent.gra)

```
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;;
;;   Japanese Sentence Grammar
;;   ~~~~~~~~~~~~~~~~~~~~~~~~
;;     Inter-ph : bun.gra
;;     Intra-ph : bunsetu.gra
;;
;;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

;;-=-=-=-=-=-=-=-=-=-=-=-
;; Inter-phrase rules
;;-=-=-=-=-=-=-=-=-=-=-=-
(<start> <--> (<s>))
(<s> <--> (<np> <vp>))
(<s> <--> (<vp>))
;;-=-=-=-=-=-=-=-=-=-=-=-
;; Intra-phrase rules
;;-=-=-=-=-=-=-=-=-=-=-=-
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<vp> <--> (<v>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

2.9 文法中の左再帰性のチェック

[exe] recursive
[src] /ifg2/LR/GrammarTool/Recursive

プログラム recursive は、文法の中に左再帰性があるかどうかをチェックします。

文法が左再帰的であるとは、ある非終端記号 A に対して、

$$A \stackrel{+}{\Longrightarrow} A\alpha$$

となるような導出が存在することをいいます。

例えば、recursive.gra という文法の左再帰性をチェックするためには、

```
% recursive recursive.gra
```

とします。

左再帰的文法の例（recursive.gra）

```
(<start> <--> (<s>))
(<s> <--> (<a> <b>))
(<a> <--> (<c> <d>))
(<b> <--> (<e> <f>))
(<c> <--> (<s>))
(<d> <--> (d))
(<e> <--> (e))
(<f> <--> (f))
```

recursive の出力例

```
*** Found left-recursive rule:
[RULE 4] (<c> <--> (<s>))
[STK] <s> <a> <c>
```

上の例の場合、<s> が <a> に書き直され、次に <a> が <c> に書き直され、最終的に 4 番目の規則で <c> が再び <s> に書き直されたということを示しています。

2.10 文法中のコメントの除去

[exe] nocmt
[src] /ifg2/LR/GrammarTool/nocmt.c

プログラム nocmt は、文法中からコメントを取り除くプログラムです。

例えば、comment.gra という文法ファイルからコメント部分を取り除き、nocomment.gra というファイルを作るには、

% nocmt comment.gra > nocomment.gra

とします。

コメントの付いた文法の例（comment.gra）

```
;;
;; comment.gra
;;
(<start> <--> (<s>))        ; grammar 0
(<s> <--> (a i u))          ; grammar 1
(<s> <--> (k a k i k u))    ; grammar 2
(<s> <--> (s a sh i s u))   ; grammar 3
;;
;; END.
;;
```

nocmt の出力例（nocomment.gra）

```
(<start> <--> (<s>))
(<s> <--> (a i u))
(<s> <--> (k a k i k u))
(<s> <--> (s a sh i s u))
```

2.11 文法規則数のカウント

[exe] countrules
[src] /ifg2/LR/GrammarTool/countrules.c

プログラム countrules は、文法中の規則数をカウントするのに用います。文法中の規則数が、標準出力に表示されます。

例えば、test.gra という文法中の規則数をカウントするには、

```
% countrules test.gra
```

とします。

2.12 文法中の単語の抽出

[exe] extractwords
[src] /ifg2/LR/GrammarTool/extractwords.c

プログラム extractwords は、文法中から単語だけを抽出するのに用います。

例えば、test.gra という文法ファイル中の単語を抽出するには、

% extractwords test.gra

とします。

extractwords の実行例

```
k o r e
o
k u r e
o k u r e
```

2.13 文法サイズの表示

[exe] grasize
[src] /ifg2/LR/GrammarTool/grasize (shell program)

プログラム grasize は、文法のサイズ（規則数、単語数、異なり単語数）を表示するのに用います。

例えば、test.gra という文法ファイルのサイズを表示するには、

```
% grasize test.gra
```

とします。

grasize の実行例

```
Number of rules: 9
Number of words: 4
Number of different words: 4
```

2.14 逆文法の生成

[exe] revgra
[src] /ifg2/LR/GrammarTool/RevGra

プログラム revgra は、逆文法（文法規則の右辺にある記号列を逆向きにした文法）を生成するのに用います。

例えば、test.gra という文法ファイルから reverse.gra という逆文法を作るには、

```
% revgra test.gra > reverse.gra
```

とします。

revgra への入力例（test.gra）

```
(<start> <--> (<s>))
(<s> <--> (<np> <v>))
(<s> <--> (<v>))
(<np> <--> (<n>))
(<np> <--> (<n> <p>))
(<n> <--> (k o r e))
(<p> <--> (o))
(<v> <--> (k u r e))
(<v> <--> (o k u r e))
```

revgra の出力例（reverse.gra）

```
(<start> <--> (<s>))
(<s> <--> (<v> <np>))
(<s> <--> (<v>))
(<np> <--> (<n>))
(<np> <--> (<p> <n>))
(<n> <--> (e r o k))
(<p> <--> (o))
(<v> <--> (e r u k))
(<v> <--> (e r u k o))
```

第 3 章

HMM - ＬＲ音声認識システム

[exe] Hmmlr
[src] /ifg2/LR/HMM-LR/Hmmlr.V7J

この章では、HMM - ＬＲ音声認識システムについて説明します。

HMM - ＬＲについては参考文献 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] を見てください。

ディレクトリ /ifg2/LR/HMM-LR/Hmmlr.V7J にあるソース・コードには、日本語のコメントが付けられていますので、HMM - ＬＲの詳細なことが知りたければ、ソース・コードを見てください。

3.1 ＨＭＭ-ＬＲの実行手順

HMM-ＬＲ音声認識システムを使って、音声認識を行なうためには、通常、次のような手順が必要です。

ステップ１ §2.1 に示された音韻表記を使って、認識用の文法を作成する。
ステップ２ プログラム convgra （§2.6）を使って、文法中の音韻名の変換を行なう。
ステップ３ プログラム slr （§2.3）を使って、ＬＲテーブル・ファイルを作成する。
ステップ４ HMM-ＬＲを実行する。

3.2 オプション指定

HMM-ＬＲ音声認識システムには、非常に多くのオプション指定を与えることができます。いくつかのオプションでは、省略時にデフォルトのファイル名または値が用いられます。デフォルト値を持つものに対しては、その値を示してあります。

例えば、次のようにしてHMM-ＬＲを起動します。

HMM-ＬＲの使用例

```
% Hmmlr test.speech  \    …   音声データ・ファイル名
        -g test     \    …   文法名
        -m hmmlist  \    …   音韻モデル・ファイル名
        -B 100      \    …   ビーム幅
        -b 18       \    …   １つのセルからの最大分岐数
        -c 1000     \    …   総セル数
        -P 9        \    …   フレーム長
        -M 3             …   音声データに対する間引き
```

音声データの指定

HMM-LRを使って音声認識を行なうためには、音声データがどこにあるかを指定しなければなりません。音声データの場所を示すために、VQインデックス・ファイルと呼ばれるファイルを、コマンド・ライン上にオプション指定子なしで与えます。

VQインデックス・ファイルの例

```
0001 MAU_MA2_01 3 moshimoshi                          295.0 875.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_01.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_01.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_01.FZPOW
0002 MAU_MA2_02 3 sochirawa                          250.0 895.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_02.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_02.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_02.FZPOW
0003 MAU_MA2_02 3 kaigijimukyokudesuka               2800.0 4035.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_02.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_02.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_02.FZPOW
```

SB3およびモデル会話に対するVQインデックス・ファイル（話者MAU）は、それぞれ以下の場所にあります。

SB3 ··· /ifg2/LR/VQLBL/SB3.LBL
モデル会話 ··· /ifg2/LR/VQLBL/MODELCONV.LBL

文法の指定（-g オプション）

-g オプションの後で、認識時に使用する文法名を指定します。例えば、

```
-g test
```

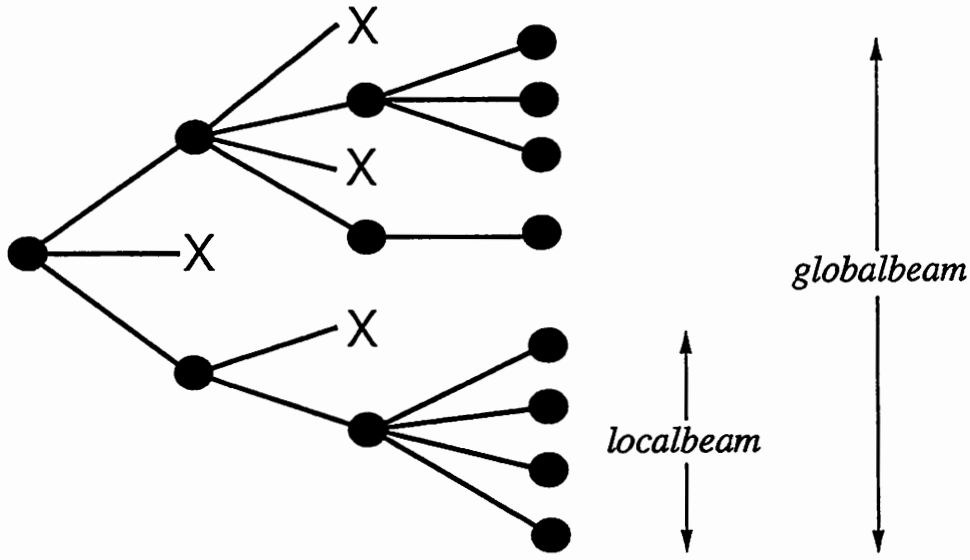
とすると、test.gra という文法ファイルと test.tab というLRテーブルファイルを用いて、音声認識を行ないます。

ビーム幅の指定 (-B, -b オプション)

[Default] -B 100, -b 15

-B および -b の後で、それぞれビーム幅 (globalbeam) および局所的なビーム幅 (localbeam) の大きさを指定します。局所的なビームとは、探索木の各節点における最大分岐数を意味しています。

次の図は、それぞれのビームがどのようなものであるのかを模式的に示したものです。図で、● または × で示されているのは節点であり、各節点にはその時点で認識された音韻が対応づけられています。また、× 節点は枝刈りされたものであることを意味しています。



総セル数の指定 (-c オプション)

[Default] -c 512

-c オプションの後で、システムで用いるセルの個数を指定します。セルとは、簡単にいえば、認識された仮説を格納しておくためのデータ構造です。セルの個数は、ビーム幅よりも十分に大きくなければなりません。

出力する認識候補数の指定 (-n オプション)

[Default] -n 5

HMM - LR音声認識システムは、1つの入力音声データに対して、複数の認識候補を出力することができます。-n オプションの後で、いくつの認識候補を出力するかを指定します。

HMM音韻モデルの指定 (-m オプション)

[Default] -m /ifg1/MAU-TH/HMM/list

-m オプションの後で、認識で用いるHMMの音韻モデルおよび継続時間長制御パラメータの入ったファイルを指定します。

例として、話者MAUに対する、HMM音韻モデル指定ファイル
/ifg1/MAU-TH/HMM/LIST-DEVOC1/List_9mQ81Ddceps_nosmthU3
の一部を、下に示します。

音韻モデルの指定

```
/ifg1/MAU-TH/HMM/hmm_9m_dceps_nosmth/mau_S_3wdsP9m.mhmm s 1
/ifg1/MAU-TH/HMM/dur_9m_dceps_nosmth/mau_S_3wdsP9m_mhmm.cdsb.dur
/ifg1/MAU-TH/HMM/hmm_9m_dceps_nosmth/mau_SH_3wdsP9m.mhmm sh 1
/ifg1/MAU-TH/HMM/dur_9m_dceps_nosmth/mau_SH_3wdsP9m_mhmm.cdsb.dur
......
/ifg1/MAU-TH/HMM/hmm_9m_dceps_nosmth/mau_a_3wdsP9m.mhmm a 2
/ifg1/MAU-TH/HMM/dur_9m_dceps_nosmth/mau_a12_3wdsP9m_mhmm.cdsb.dur
/ifg1/MAU-TH/HMM/dur_9m_dceps_nosmth/mau_a12_3wdsP9m_mhmm.cdsb.dur2
......
```

上の例では、ファイル
/ifg1/MAU-TH/HMM/···/mau_S_3wdsP9m.mhmm
に音韻 /s/ に対する HMM モデルのパラメータが書かれていることを示しています。また、行の最後の数字1は、音韻 /s/ の継続時間長を制御するために用いるパラメータ・ファイルの個数です。音韻 /s/ の場合は1ですので、次の行に書かれているファイル
/ifg1/MAU-TH/HMM/···/mau_S_3wdsP9m_mhmm.cdsb.dur
を用いますが、音韻 /a/ の場合は2ですので、2つのファイル
/ifg1/MAU-TH/HMM/···/mau_a12_3wdsP9m_mhmm.cdsb.dur
/ifg1/MAU-TH/HMM/···/mau_a12_3wdsP9m_mhmm.cdsb.dur2
を用いることになります。

logテーブルの指定 (-L オプション)

[Default] -L /ifg1/logtbl/log.tbl

音韻照合の際のHMMの確率計算では、確率値をlog化して扱っています。また、log計算は、高速化のために、表引きによって行なっていますが、このためには log compression table と呼ばれるものが必要になります。-L オプションの後で、log compression table の入ったファイル名を指定します。

冗長なモードの指定 (-V オプション)

-V オプションを指定すると、冗長 (verbose) なモードになります。このモードでは、認識の各サイクルごとに (新たな音韻が認識されるごとに)、尤度順に複数の認識候補が表示されます。

音韻照合用の threshold の指定 (-X, -x, -Q, -d オプション)

[Default] -X 20.0, -x 15.0, -Q 30.0

音韻照合により得られる尤度が、決められた threshold よりも悪ければ、音韻照合が失敗したとして、ビームサーチを待たずに即座に枝刈りが行なわれます。-X オプションの後で、このための threshold を設定します。-Q オプションの後では、最初の無音区間に対する threshold を設定します。threshold は、認識の各サイクルごとに適当な値に再設定されますが、-x オプションの後で、再設定される threshold の下限を指定します。

また、-d オプションを指定すると、認識の各サイクルごとに、threshold を自動的に再設定する機能を解除します。

照合範囲の縮小モードの指定 (-F オプション)

音韻照合区間は、認識が進むにつれ (認識候補の音韻数が増えるにつれ)、次第に大きくなり、音韻照合計算に費やされる時間がそれに従い増えてきます。-F オプションを指定すると、各フレームの正規化確率と threshold を用いて、照合区間を自動的に縮小します。このオプションを使うことにより、認識時間は速くなりますが、多少の認識率の低下を招きます。

統計的言語モデルに関する指定 (-l, @S, @G オプション)

-l オプションを指定すると、言語の統計的な情報を考慮して、認識仮説の尤度を決定します。現在、使用可能な統計情報は、音節の連鎖統計情報と確率文法の2種類です。@S, @G オプションの後で、それぞれの統計情報に対する重みを指定します。

話者適応用のヒストグラムの指定 (-H オプション)

-H オプションの後で、話者適応用のヒストグラムが入ったファイル名を指定します。

継続時間長制御に関する指定 (-W, -s, -N オプション)

[Default] -W 7.0, -s 3.0, -N 1.0

-W オプションの後では、継続時間長制御の重みを指定します。

-s オプションの後では、HMMの状態ごとの継続時間長制御用のペナルティの付与範囲を指定します。また、-N オプションの後で、-s オプションにより与えられた値に対する係数（実際には、何倍して用いるかという値）を指定します。

音声データに関する種々の指定 (-A, -E, -P, -M オプション)

[Default] -A 54, -E 27, -P 3, -M 1

HMM - L R音声認識システムでは、音声データを読み込む際に、データの前後に無音区間を含めて読み込みます。-A オプションの後では、前後に何 msec. の無音区間を含めて読み込むのかを指定します。

-E オプションの後では、最終的な認識候補の尤度を決定する際に、音声データの終端からどのくらいの範囲までを考慮するかという値（msec.）を指定します。例えば、音声データの終端を X、-E オプションで与えられた値を Y とすれば、認識候補の尤度は、X - Y から X までの正規化尤度のうちで最も良い値のものとなります。

-P オプションの後では、フレーム長を指定します。また、-M オプションの後では、音声データに対する間引きの割合を指定します。

3.3 認識結果

ＨＭＭ-ＬＲは認識結果を標準出力に出しますので、ファイルに結果を保存するためには、例えば UNIX のリダイレクションを使いましょう。

実行結果の例を次に示します。

認識結果の例

```
(002) MAU_MA2_02    196.0   949.0   |sochirawa|  84 frames
************************************************************
Recognition time: CPU-time = 3549 msec, Elapsed-time = 3 sec.
Total-verify = 278, Depth = 17
  1: sochira-wa                (prob = 11.74841)
  2: sochira-e-wa              (prob = 12.16559)
  3: sochira-ni-wa             (prob = 12.36498)
  4: sochira-ga                (prob = 12.37688)
  5: sochira-ni                (prob = 12.37688)
************************************************************
```

- 最初の (002) は、文節番号です。
- MAU_MA2_02 は、音声データを識別する名前です。
- 196.0 と 949.0 は、音声の始終端（msec.）を示しています。
- sochirawa は、実際の発声内容です。
- 84 frames は、音声の総フレーム数です。
- Recognition time: で始まる行に、認識にかかったＣＰＵ時間と経過時間が示されています。
- Total_verify は、駆動された音韻照合の回数を示しています。
- Depth は、認識木の深さを示しています。
- 最後に、認識候補が１位から５位まで示されています。また、各候補には尤度が付いています。

3.4 認識率の計算

[exe] score
[src] /ifg2/LR/Score/score.c

HMM - ＬＲからの出力結果から認識率を計算するためには、プログラム score を用います。

例えば、test.output という出力結果のファイルから認識率を計算するには、

% score test.output

というようにします。

score の実行例

```
***** HMM-LR Recognition Rate *****

Total:   279
Correct: 277

 1: 258 ( 92.5%)  /  258 ( 92.5%)
 2:  15 (  5.4%)  /  273 ( 97.8%)
 3:   4 (  1.4%)  /  277 ( 99.3%)
 4:   0 (  0.0%)  /  277 ( 99.3%)
 5:   0 (  0.0%)  /  277 ( 99.3%)
```

第４章

２段階ＨＭＭ－ＬＲ音声認識システム

[exe] Hmmlr2
[src] /ifg2/LR/HMM-LR/TwoLevel/SRC

２段階ＨＭＭ－ＬＲ音声認識システムは、文節内文法と文節間文法を用いた文認識システムです。

２段階ＨＭＭ－ＬＲについては参考文献 [21, 22, 23, 24] を見てください。

4.1　２段階ＨＭＭ－ＬＲの実行手順

ＨＭＭ－ＬＲ音声認識システムを使って、音声認識を行なうためには、通常、次のような手順が必要です。

ステップ１ §2.1 に示された音韻表記を使って、文節内文法を作成する。
ステップ２ 文節カテゴリのリストを作成する。
ステップ３ プログラム convgra （§2.6）を使って、文法中の音韻名の変換を行なう。
ステップ４ プログラム slr2 （§2.4）を使って、文節内ＬＲテーブルを作成する。
ステップ５ 文節間文法を作成する。
ステップ６ プログラム slr （§2.3）を使って、文節間ＬＲテーブルを作成する。
ステップ７ ２段階ＨＭＭ－ＬＲを実行する。

4.2 オプション指定

基本的なオプション指定は、通常のHMM-LR音声認識システムと同様です。ここでは、2段階HMM-LR音声認識システムに特有のオプション指定についてのみ述べます。

音声データの指定

2段階HMM-LRは、文を認識するシステムですので、VQインデックス・ファイルの中に文の終りが明示されている必要があります。このために、通常のVQインデックス・ファイルに文の終りを示す記号（#）を挿入したものを用います。

2段階用のVQインデックス・ファイル

```
0001 MAU_MA2_01 3 moshimoshi                    295.0 875.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_01.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_01.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_01.FZPOW
#
0002 MAU_MA2_02 3 sochirawa                    250.0 895.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_02.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_02.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_02.FZPOW
0003 MAU_MA2_02 3 kaigijimukyokudesuka          2800.0 4035.0
/data9/IFG/MAU/FZWLR/MA2/MAU_MA2_02.FZW
/data9/IFG/MAU/FZSCEP/MA2/MAU_MA2_02.FZSCEP
/data9/IFG/MAU/FZPOW/MA2/MAU_MA2_02.FZPOW
#
```

文法の指定 (-g, -G オプション)

-g オプションの後で、認識時に使用する文節内文法名を指定します。また、-G オプションの後では、文節間文法名を指定します。

4.3 認識率の計算

[exe] score-bun
[src] /ifg2/LR/Score/score-bun.c

2段階HMM - ＬＲからの出力結果から認識率を計算するためには、プログラム score-bun を用います。

例えば、test2.output という出力結果のファイルから認識率を計算するには、

% score-bun test2.output

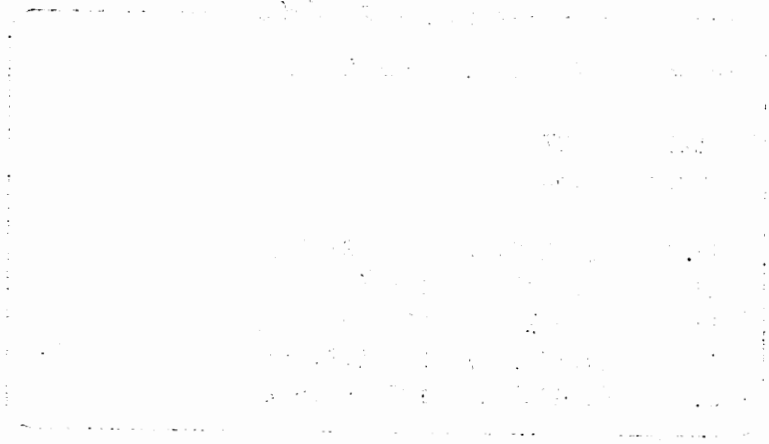
というようにします。

score-bun の実行例

```
/* Score of Sentence HMM-LR */

Total   = 137
Correct = 135

 1: 125 (91.24%) /  125 (91.24%)
 2:   3 (2.19%)  /  128 (93.43%)
 3:   4 (2.92%)  /  132 (96.35%)
 4:   1 (0.73%)  /  133 (97.08%)
 5:   2 (1.46%)  /  135 (98.54%)
```



参考文献

[1] 北 研二, 川端 豪, 斎藤 博昭.
"HMM 音韻認識と予測 LR パーザを用いた文節認識".
日本音響学会秋季研究発表会, pp. 259-260, 1988 年 10 月.

[2] 北 研二, 川端 豪, 斎藤 博昭.
"HMM 音韻認識と予測 LR パーザを用いた文節認識".
電子情報通信学会音声研究会, pp. 63-69, 1988 年 10 月.

[3] 北 研二, 川端 豪, 斎藤 博昭.
"HMM Continuous Speech Recognition Using Predictive LR Parsing".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 703-706, 1989 年 5 月.

[4] 北 研二, 川端 豪, 斎藤 博昭.
"HMM 音韻認識と拡張 LR 構文解析法を用いた連続音声認識".
ATR テクニカル・レポート TR-I-0082, 1989 年 6 月.

[5] 北 研二, 川端 豪, 斎藤 博昭.
"Parsing Continuous Speech by HMM-LR Method".
Proceedings of the International Workshop on Parsing Technologies, pp. 126-131, 1989 年 8 月.

[6] 北 研二, 坂野 俊哉, 保坂 順子, 川端 豪.
"SL-TRANS における文節音声認識 — HMM 音韻認識と LR 構文解析法による文節音声認識 —".
情報処理学会第 39 回全国大会, pp. 718-719, 1989 年 10 月.

[7] 花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
"HMM-LR 音声認識システムの性能評価".
電子情報通信学会音声研究会, pp. 63-70, 1989 年 12 月.

[8] 北 研二, 川端 豪, 斎藤 博昭.
"HMM 音韻認識と拡張 LR 構文解析法を用いた連続音声認識".
情報処理学会論文誌, Vol. 31, No. 3, pp. 472-480, 1990 年 3 月.

[9] 北 研二.
"Generalized LR Parsing in Hidden Markov Model".
ATR テクニカル・レポート TR-I-0161, 1990 年 4 月.

[10] 花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
"ATR HMM-LR Continuous Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 53-56, 1990 年 5 月.

[11] 花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
"ATR HMM-LR Continuous Speech Recognition System".
Readings in Speech Recognition, Waibel, A. and Lee, K. F. Eds., Morgan Kaufmann Publishers, pp. 611-614, 1990 年.

[12] 花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
"HMM-LR 音声認識システムの性能評価".
日本音響学会誌, Vol. 46, No. 10, pp. 817-823, 1990 年 10 月.

[13] 北 研二, 江原 暉将, 森元 逞.
"HMM-LR 音声認識の大語彙への適用".
情報処理学会第 42 回全国大会, pp. 2.110-2.111, 1991 年 3 月.

[14] 北 研二, 川端 豪, 斎藤 博昭.
"GLR Parsing in Hidden Markov Model".
Generalized LR Parsing, Tomita, M. Ed., Kluwer Academic Publishers, pp. 153-164, 1991 年.

[15] 北 研二, 川端 豪, 花沢 利行.
"HMM-LR 音声認識システムにおける統計的言語情報の利用".
電子情報通信学会音声研究会, pp. 1-6, 1990 年 1 月.

[16] 北 研二, 川端 豪, 花沢 利行.
"HMM-LR 音声認識システムにおける統計的言語情報の利用".
日本音響学会春季研究発表会, pp. 85-86, 1990 年 3 月.

[17] 重盛 勝, 北 研二, 森元 逞.
"テキスト・データベースを用いた文脈自由文法の適用確率推定".
ATR テクニカル・レポート TR-I-0141, 1990 年 3 月.

[18] 北 研二, 川端 豪, 花沢 利行.
"HMM Continuous Speech Recognition Using Stochastic Language Models".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 581-584, 1990 年 4 月.

[19] 北 研二, 森元 逞.
"音声認識システムにおける確率文法の有効性".
情報処理学会第 41 回全国大会, pp. 2.237-2.238, 1990 年 9 月.

[20] 北 研二, 川端 豪, 花沢 利行.
"HMM Speech Recognition Using Stochastic Language Models".
The Journal of the Acoustical Society of Japan, Vol. 12, No. 3, pp. 99-105, 1991 年 5 月.

[21] 北 研二, 竹澤 寿幸, 保坂 順子, 江原 暉将, 森元 逞.
“2 段階 LR 構文解析法を用いた文認識”.
日本音響学会秋季研究発表会, pp. 127-128, 1990 年 9 月.

[22] 北 研二, 竹澤 寿幸, 保坂 順子, 江原 暉将, 森元 逞.
“Continuous Speech Recognition Using Two-Level LR Parsing”.
Proceedings of the International Conference on Spoken Language Processing, pp. 905-908, 1990 年 11 月.

[23] 北 研二, 竹澤 寿幸, 森元 逞.
“Continuous Speech Recognition Using Two-Level LR Parsing”.
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E 74, No. 7, pp. 1806-1810, 1991 年 7 月.

[24] 北 研二, 森元 逞, 嵯峨山 茂樹.
“上位文法カテゴリへの到達可能性照合機構を備えた LR 解析法とその音声認識への応用”.
日本音響学会秋季研究発表会, pp. 171-172, 1991 年 10 月.