TR-I-0290

HMMによる言語モデルの自動獲得の検討
Radial Basis Function-Based Fuzzy Partition Models Aiming at Speech Recognition

山本寛樹† 村上仁一 嵯峨山茂樹
Hiroki Yamamoto, Jin'ichi Murakami, Shigeki Sagayama

1992.12

概要

従来のHMMによる言語のモデル化の研究では、単語を品詞などのカテゴリーに分類した言語データをHMMの学習に用いていた。これに対して本研究では、カテゴリー分類を与えずに単語列のみをergodic HMMに学習させて、確率つきネットワーク文法と同時に単語のカテゴリーの自動獲得も試みた。その結果得られたergodic HMMは、従来の確率つきネットワーク文法と類似した形態をしており、単語を従来の品詞より、文中での機能に従って、さらに細かいカテゴリーに分類していた。また、openデータとclosedデータの文生成尤度に大差がなく、HMMの獲得した文法は一般性があるとことも示された。これにより、ergodic HMMを用いて確率つきネットワーク文法を自動生成し、単語を分類できる可能性が示された。更に、得られたHMMに対して連続文音声認識実験を試みた。

†早稲田大学

目次

第 1 章

序論

1.1 本研究の背景・目的

音声を機械で認識する過程では、大きく分けて音声処理および言語処理の二つの処理が行なわれる。

音声処理は、発声された音声波形をスペクトル分析などを行ない音響的特徴を抽出し、あらかじめ作成された音響モデルと照合し、音韻などの音声単位の言語列に変換する処理である。

言語処理は、音響処理に加え、音声認識率を向上するために言語の様々な情報（言語情報）を取り込む処理である。言語処理で用いる情報とは、例えば、構文の規則、単語（または品詞）間の接続に関する制約などが挙げられる。音声認識では、これらの言語情報の集合を言語モデル（Language Model）と呼んでいる。

言語モデルには、ネットワーク文法（有限オートマトン）や文脈自由文法に代表される構文情報を記述した構文モデルや bigram、trigram に代表される統計情報を記述した統計（確率）モデルなどがある。また、構文モデルに単語の出現確率などの確率値（統計情報）を加えると、文法の複雑さが低減され音声認識システムの言語モデルとして用いると、認識率はさらに向上することもわかっている [1]。一般に、構文情報は人間が獲得した言語に関する知識に基づいて人間が記述しているため、大規模な言語情報のモデル化には、多大な労力を要する。また、bigram や trigram など統計情報のみを用いたモデルでは言語のもつ構文情報が容易に表現されない。

そこで、構文、統計両方の情報を記述した文法を、自動獲得すること を本研究の目的とした。

構文情報に確率値を加えた文法の一つに確率つきネットワーク文法（確率有限状態オートマトン）がある。この確率つきネットワーク文法の記述形式と全状態間の遷移の許された離散型 Ergodic HMM の構造とは非常に類似している。離散型 Ergodic HMM は状態遷移

表 1.1: 本研究と従来の研究の比較

	学習データ	出力シンボル	評価方法
村瀬等	文	ワードクラス (268 クラス)	モデルのエントロピーを bigram,trigram の場合と比較
田本等	文節	品詞 (25 品詞)	学習後の HMM の内部解析
本研究	文	単語 (6418 単語)	学習後の HMM の内部解析 open data と closed data の生成確率を比較 連続音声認識へ適用

確率、シンボル出力確率、初期状態確率で特徴づけられ、確率つきネットワーク文法は、状態遷移確率と単語（品詞）出力確率を用いて記述した言語モデルである。Ergodic HMM の出力シンボルを単語（品詞）とすれば、両者は等価となる。また、HMM には Baum-Welch algorithm という学習アルゴリズムがあり、入力されたデータの生成確率が最大になるように各パラメータを推定する。そこで、ergodic HMM に単語列や品詞列などの言語データを入力することにより、確率つきネットワーク文法を自動的に獲得できる可能性がある。

従来、同様な考え方でいくつかの研究が村瀬 [2]、田本 [3] 等によって報告されている。村瀬等は学習後のモデルのエントロピーを調べ、bigram や trigram でモデル化した場合と比較し、ergodic HMM による言語のモデル化の可能性を調査している。田本等は学習後の HMM を解析し、その形態が従来使われているネットワーク文法と類似していることを報告している。しかし、これらの研究では、HMM の学習に単語を品詞などのカテゴリーに分類した言語データを用いていた。これに対して、本研究 [4] [5] では、カテゴリー分類を与えずに単語列のみを ergodic HMM に学習させることを試みた。

これにより、

文法だけでなく、単語のカテゴリーも状態遷移出力の偏りとして同時に学習される

ことが期待できる。

本論文では、ergodic HMM に品詞情報を持たない単語列を学習させ、学習後の ergodic HMM について、パラメータの解析、文の生成確率の算出、連続音声認識への適用などを行なった。これらの結果を示し、本手法の有効性を明らかにする。

1.2 概要

本論文の構成・概要を述べる。

本研究の目的および背景については 1.1節で述べた。

第2章では、準備として、本研究で言語のモデル化の手段として用いた隠れマルコフモデル (Hidden Markov Model) に関して説明する。2.1節で基本的な考え方を述べ、2.2節で HMM に関する基本的な問題を説明し、2.3節で基本問題の解決法である学習アルゴリズムなどをについて簡単な例を用いて紹介する。また、計算機で扱う上での問題点と、よく用いられているその対策について 2.3.4節で述べる。

第3章では、本研究の目的である HMM を用いた言語のモデル化に関して述べる。状態数の異なる ergodic HMM について、言語データを学習させる実験について、3.1節で HMM の学習に用いた言語データベースに関する資料を示し、3.2節で学習に用いた HMM の構造など、実験条件を述べる。3.3節では、実験結果の評価を行なう。学習後のパラメータの解析方法を 3.3.1節で説明し、学習後の HMM を解析した結果を状態数ごとに 3.3.2節〜3.3.5節で示し、結果から得られる特徴を述べる。3.3.6では、言語データ中によく見られる表現が学習後の HMM のどの部分で表されるかを示している。さらに、言語モデルとしての能力を調べるため、HMM の文生成確率やエントロピーを計算した結果を 3.3.7節で示す。3.3.8節では、学習データ量を変化させた場合のモデル化の違いについて述べる。3.4節では HMM が獲得した文法を連続音声認識に適用する。その実験条件を 3.4.1 節で、実験結果を 3.4.2節で示す。また、HMM を用いる際に問題となる初期パラメータによるモデル化の違いについて 3.5節で述べる。

4章では第3章で行なった実験の結果に対する考察、および実験の問題点をあげる。

最後に 5章で本研究の結論を述べる。

巻末に本研究で用いたプログラムのリストを付録として掲載する。

第 2 章

HMM

2.1 HMM(Hidden Markov Model, 隠れマルコフモデル)

状態遷移確率、シンボル生成確率、初期状態の分布確率（初期状態確率）で構成される HMM は、不確定な時系列のデータをモデル化するための有効な統計的手法である [6]。HMM は、出力シンボルによって一意に状態遷移先が決まらないという意味での非決定性有限オートマトンとして定義される。出力シンボル系列が与えられても状態遷移系列は唯一に決まらない。観測できるのはシンボル系列だけであることから hidden(隠れ) マルコフモデルと呼ばれる [7]。

T	: 観測系列の長さ
$o_1, o_2, ..., o_T$	: 観測系列
N	: 状態数
L	: 観測シンボルの数
$S = \{s\}$	: 状態集合
s_t	: 時刻 t の時の状態（番号）
i, j	: 状態番号
$v = \{v_1.v_2, ..., v_L\}$	: 出力可能なシンボル集合

とすると、このオートマトンは, 状態遷移確率分布 A, シンボル生成確率分布 B, 初期状態確率分布 π、

$$A = \{a_{ij} \mid a_{ij} = P(s_{t+1} = j \mid s_t = i)\} \qquad (1 \leq i, j \leq N) \tag{2.1}$$

$$B = \{b_{ij}(o_t) \mid b_{ij}(o_t) = P(o_t \mid s_t = j)\} \qquad (1 \leq i, j \leq N, 1 \leq t \leq T) \tag{2.2}$$

$$\pi = \{\pi_i \mid \pi_i = P(s_0 = i)\} \qquad (1 \leq i \leq N) \tag{2.3}$$

で構成される。これらのパラメーターを用いて、HMM を次のように略記する。

$$\lambda = (A, B, \pi) \tag{2.4}$$

HMM は、シンボルの生成確率を状態について定義した Moore マシンと状態遷移について定義した Mealy マシンに分類できる。以下では、本報告で実験に用いた Mealy タイプの HMM について述べる [7]。

HMM が、

$$o_1, o_2, ..., o_T \qquad (o_t = v_k, 1 \leq k \leq L, 1 \leq t \leq T)$$

という系列を生成する過程は次のようになる。

1. 初期状態確率を π にしたがって決定する。
2. 次に遷移する状態（$s_{t+1} = j$）を現在の状態（$s_t = i$）と状態遷移確率 a_{ij} にしたがって決定する。
3. 状態遷移する際に出力するシンボルを出力シンボル確率 $b_{ij}(o_t)$ にしたがって決定する。
4. 2. に戻る

HMM には、ある状態から全ての状態に遷移できる全遷移型（ergodic）モデルや、状態遷移が一定方向に進む Left to Right モデルがある。図 2.1 に簡単な HMM （Left to Right モデル）の例を示す。

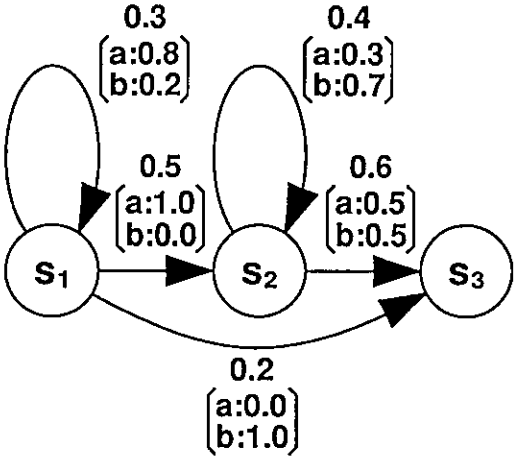


図 2.1: 3 状態 Left-to-Right HMM

この HMM は三つの状態で構成され、2 種類のラベル a と b のみからなるラベル系列を出力する。初期状態確率は $\pi_1 = 1.0, \pi_2 = 0, \pi_3 = 0$、最終状態を S_3 とし、図のような遷

移のみ行なうものとする。図において、0.3 などアークに添えられている数字は状態遷移確率を表し、[ ] 内の数字の上段はラベル a の出力確率、下段はラベル b の出力確率を表す。状態 s_1 を例にとれば、s_1 から状態 s_1 自身に 0.3 の確率で遷移し、遷移の際に 0.8 の確率で a を出力し、0.2 の確率で b を出力する。他の状態、遷移についても同様である。ここで、ラベル系列が aab を出力する確率を考える。この HMM で許される状態系列は長さの異なるものも含めて無数にあるが、aab を出力する可能性のあるものは、$s_1 - s_1 - s_2 - s_3$ と $s_1 - s_2 - s_2 - s_3$ と $s_1 - s_1 - s_1 - s_3$ の 3 種類だけであり、それぞれの確率は、

$$
\begin{aligned}
0.3 \times 0.8 \times 0.5 \times 1.0 \times 0.6 \times 0.5 &= 0.036 \\
0.5 \times 1.0 \times 0.4 \times 0.3 \times 0.6 \times 0.5 &= 0.018 \\
0.3 \times 0.8 \times 0.3 \times 0.8 \times 0.2 \times 1.0 &= 0.01152
\end{aligned}
$$

である。状態系列は観測されず、いずれの可能性もあるので、三つの合計、

$$0.036 + 0.018 + 0.01152 = 0.06552$$

の確率でこの HMM は aab を出力する。HMM の状態系列は観測されないが推定することはできる。この例では、aab を出力する可能性がもっとも高い状態系列は、前記の計算から容易に $s_1 - s_1 - s_2 - s_3$ とわかる。HMM がラベル系列を出力する確率を子のような最適状態系列上の確率評価（この場合は 0.036）だけで近似することもよく行なわれる。

2.2 HMMの基本問題

シンボルの出力形式（Mealy またはMoore）以外にHMMに関して重要な基本問題として次の五つが挙げられる[7][6]。

1. モデルの尤度評価
   観測系列 $O = o_1, o_2, ..., o_T$ とHMM,$\lambda(\pi, A, B)$ が与えられている時、モデル λ が O を出力する尤度 $P(O \mid \lambda)$ を求めること。

2. モデルの推定
   訓練用シンボル O を与えて尤度 $P(O \mid \lambda)$ が最大になるようにモデル λ のパラメータ π, A, B を推定すること。

3. 最適状態系列の推定
   モデル λ がシンボル系列 O を出力する時の最も可能性の高い状態遷移系列を推定し、その系列に対する尤度を求めること。

4. モデルの設計
   状態数や遷移先の種類など、どのような構造のHMMを用いるか決定すること。

5. 訓練用データの基準
   良いモデルを得るための訓練用データの量や質を決定すること。

1. の解法を2.3.1節で、2. についての解法は2.3.2節で、3. については2.3.3節で方法を紹介する。4. 5. については、現在のところ具体的な方法はわかっておらず、パラメータなどの初期条件を変えながら訓練を繰り返し、経験的に得られたものから最適になるものを用いる。

2.3 HMMの基本アルゴリズム

2.2節で挙げた問題に対する基本的アルゴリズムを以下で紹介する [6] [7]。

2.3.1 Forward-Backward algorithm

HMM $\lambda(\pi, A, B)$ が観測系列 $O = o_1, o_2, ..., o_T$ を生成する尤度 $P(O \mid \lambda)$ を求めるには、まず長さ T の全ての状態系列に対して、確率の計算を行なうことが考えられる。可能な状態系列 $S = s_0, s_1, ..., s_T$ が O を生成する確率は、次のように書ける。

$$P(O \mid S, \lambda) = \prod_{t=1}^{T} P(O_t \mid s_t, \lambda) \tag{2.5}$$

各観測は、確率的に独立とみなせるので、

$$P(O \mid S, \lambda) = b_{s_0 s_1}(o_1) \cdot b_{s_1 s_2}(o_2) \cdot ... \cdot b_{s_{T-1} s_T}(o_T) \tag{2.6}$$

一方、状態系列 S の生成確率は次のようになる。

$$\begin{aligned} P(S \mid \lambda) &= \pi_{s_1} a_{s_1 s_2} a_{s_2 s_3} ... a_{s_{T-1} s_T} \\ &= a_{s_0 s_1} a_{s_1 s_2} ... a_{s_{T-1} s_T} \end{aligned} \tag{2.7}$$

したがって、観測系列 O のモデル λ における生成確率（尤度）は、

$$\begin{aligned} P(O \mid \lambda) &= \sum_{allS} P(O \mid \lambda) P(O \mid S, \lambda) \\ &= \sum_{all s_0 ... s_T} \pi_{s_0} a_{s_0 s_1} b_{s_0 s_1}(o_1) \cdot a_{s_1 s_2} b_{s_1 s_2}(o_2) \cdot ... \cdot a_{s_{T-1} s_T} b_{s_{T-1} s_T}(o_T) \end{aligned} \tag{2.8}$$

この方法による計算量は $O(2TN^T)$ になり、実質的に計算不可能である。計算量を削減した実用的なアルゴリズムとして Forward-Backward algorithm がある。

Forward algorithm

時刻 t の時に $o_1, o_2, ..., o_t$ という観測系列を出力して、状態 j にいる確率を次のように定義する。

$$\alpha_t(j) = P(o_1, o_2, ..., o_t, s_t = j \mid \lambda) \tag{2.9}$$

$P(O \mid \lambda)$ は $\alpha_t(j)$ の漸化式を次のように計算することによって求めることができる。

Forward algorithm

1. 初期化

   全ての状態 $j(1 \leq j \leq N)$ に対して

$$\alpha_0(j) = \pi_j \tag{2.10}$$

   とする。

2. 導出過程

   時間軸 $(t = 1, ..., T)$ に沿って、全ての状態 $j(1 \leq j \leq N)$ に対し、$\alpha_t(j)$ を次のように計算する。

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \tag{2.11}$$

3. 結果

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.12}$$

このアルゴリズムでは、直前の確率 $\alpha_{t-1}(i)$ $\alpha_t(j)$ を求めるが解っていれば $\alpha_t(j)$ $1 \leq j \leq N)$ を求めることができる。これは図 2.2 のようなトレリス（出力ラベル系列が対応する時間経過を横軸にして、各状態を縦に並べて状態遷移を示した図）の上で考えると理解しやすい。前記の図 2.1 の HMM がラベル系列 aab を出力する例に適応すると、$\alpha_t(j)$ は図 2.2 のようにトレリス上の左上（初期状態）から右下（最終状態）に向かって順次求まり、

$$P(a, a, b \mid \lambda) = \alpha_3(3) = 0.06552$$

となる。また、この方法での計算量は $O(N^2T)$ で計算可能である。

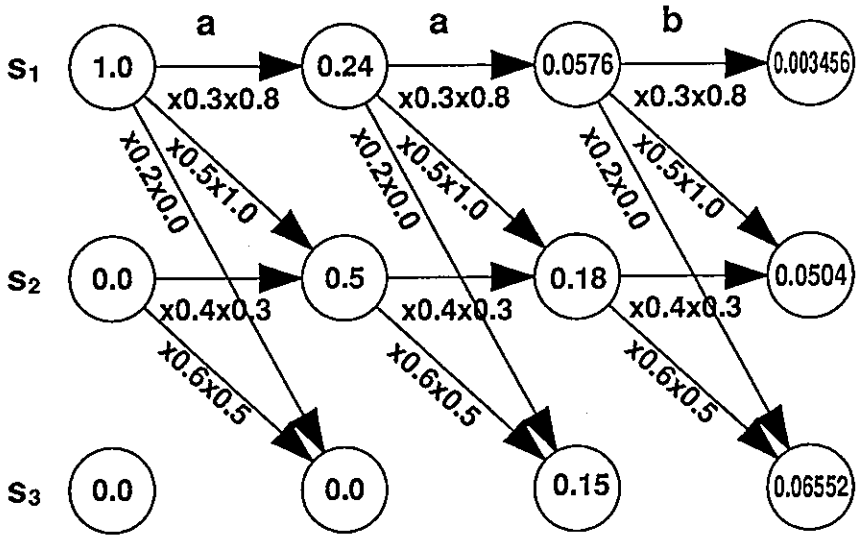


図 2.2: トレリス上の $\alpha_t(i)$ の計算

Backward algorithm

Forward algorithm が初期状態から前向きに計算するのに対して、Backward Algrithm は後向きに計算していく。モデル $\lambda(\pi, A, B)$ において、時刻 t に状態 i にいて、以後 $o_{t+1}, o_{t+2}, ..., o_T$ を出力する確率を次のように表す。

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T, s_t = i \mid \lambda) \tag{2.13}$$

この $\beta_t(i)$ は以下の手順で求まる。

Backward algorithm

1. 初期化

   全ての状態 $i(1 \leq i \leq N)$ に対して、

   $$\beta_T(i) = 1 \tag{2.14}$$

   とする。

2. 導出過程

   時間軸 $(t = T-1, ..., 0)$ に沿って、全ての状態 $i(1 \leq i \leq N)$ に対し、β_t を次のように計算する。

   $$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_{ij}(o_t) \beta_{t+1}(j) \tag{2.15}$$

3. 結果

   $$P(O \mid \lambda) = \sum_{i=1}^{N} \pi \beta_0(i) \tag{2.16}$$

観測系列 O のモデル λ における生成確率を求めるには、Forward algorithm, Backward algorithm のいずれかを用いれば良く、問題 1. の解決に必ずしも Backward algorithm は必要ではない。Backward algorithm は問題 2. 問題 3. の解決に用いられる。

2.3.2 Baum-Welch Re-estimation algorithm

問題2. で述べた観測系列の生成確率を最大にするモデル λ のパラメータの局所的最適値を求める方法として、Baum-Welch Re-estimation algorithm （パラメータ再推定法）がある。

モデル λ が観測系列 $O = o_1, o_2, ..., o_T$ を生成する場合において、時刻 t で状態 i から状態 j に遷移する確率 $\xi_t(i,j)$ を次のように定義する。

$$\begin{aligned} \xi_t(i,j) &= P(s_{t-1} = i, s_t = j \mid O, \lambda) && (2.17) \\ &= \frac{\alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}{P(O \mid \lambda)} \qquad (1 \leq t \leq T) && (2.18) \end{aligned}$$

ここで、シンボルの生成過程で、時刻 t で状態 j にいる確率 $\gamma_t(j)$ を定義する。

$$\begin{aligned} \gamma_t(j) &= P(s_t = j \mid O, \lambda) && (2.19) \\ &= \sum_{i=1}^{N} \xi_t(i,j) \qquad (1 \leq t \leq T) && (2.20) \end{aligned}$$

この $\gamma_t(i)$ と $\xi_t(i,j)$ とからモデル λ の再推定 （$\lambda \rightarrow \bar{\lambda}$ ） を次のように行なう。

Baum-Welch algorithm

1. 初期状態確率

$$\bar{\pi}_i = \gamma_0(i) = \frac{\alpha_0(i)\beta_0(i)}{P(O \mid \lambda)} \qquad (1 \leq i \leq N) \tag{2.21}$$

2. 状態遷移確率

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_{t-1}(i)} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}{\sum_{t=1}^{T} \alpha_{t-1}(i) \beta_{t-1}(i)} \tag{2.22}$$

3. シンボル出力確率

$$\bar{b}_{ij}(o_t) = \frac{\sum_{t=1}^{T}{}_{s.t.\, o_t = v_k} \xi_t(i,j)}{\sum_{t=1}^{T} \xi_t(i,j)} = \frac{\sum_{t=1}^{T}{}_{s.t.\, o_t = v_k} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}{\sum_{t=1}^{T} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)} \tag{2.23}$$

再推定された $\bar{\lambda}$ の評価は次のようになる。

1. $\bar{\lambda} = \lambda$ → （局所的な）収束状態

2. $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$ → シンボル系列 O を出力するより最適なモデル λ を推定

2.3.3 Viterbi algorithm

最適な状態系列（最適パス） $S = s_1, s_2, ..., s_T$ とこのパス上での確率を求めるアルゴリズムは最初に Viterbi によって提案されたことから Viterbi algorithm と呼ばれている。

モデル λ において観測系列 $O = o_1, o_2, ..., o_T$ に対する最適な状態系列 $s = s_1, s_2, ..., s_T$ を求めるために、時刻 t で状態 i に至るまでの最適状態確率 $\delta_t(i)$ を定義する。

$$\delta_t(i) = \max_{s_1, s_2, ..., s_{t-1}} p(s_1, s_2, ..., s_t = i, o_1, o_2, ..., o_t \mid \lambda) \tag{2.24}$$

時刻 $t+1$ における最適状態の確率は次のように導出できる。

$$\delta_t(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_{ij}(o_{t+1}) \tag{2.25}$$

時刻 t 状態 i において生成確率を最大にするパス（状態遷移）を $\psi_t(j)$ 、最適パスの生成確率を P^* 、最適パス上の最終状態を s_T^* とすると最適パス、およびその生成確率は以下の手順で求まる。

Viterbi algorithm

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T, s_t = i \mid \lambda) \tag{2.26}$$

1. 初期化

$$\delta_0(i) = \pi_i \tag{2.27}$$

$$\psi_0(i) = 0 \qquad (1 \le i \le N) \tag{2.28}$$

2. 導出過程

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij} b_{ij}(o_t)] \tag{2.29}$$

$$\psi_t(j) = \underset{1 \le i \le N}{argmax} [\delta_{t-1}(i) a_{ij} b_{ij}(o_t)] \qquad (1 \le t \le T, 1 \le j \le N) \tag{2.30}$$

3. 結果

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \tag{2.31}$$

$$s_T^* = \underset{1 \le i \le N}{argmax} [\delta_T(i)] \tag{2.32}$$

4. 状態系列のバックトラック

$$s_t^* = \psi_{t+1}(s_{t+1}^*)(0 \le t \le T-1) \tag{2.33}$$

4. で求めた $s_0^*, s_1^*, ..., s_T^*$ が最適パスとなる。前出の aab を出力するモデルに Viterbi algorithm を用いた簡単な例を図 2.3 に示す。

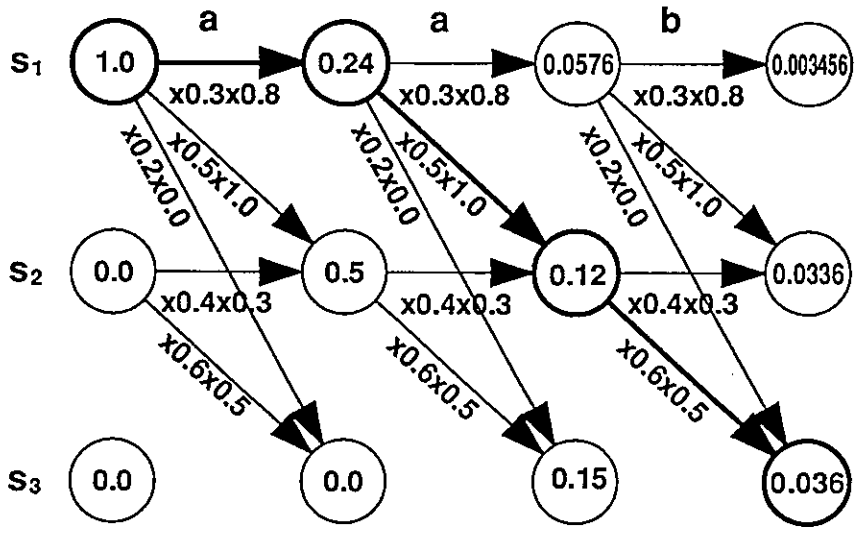


図 2.3: Viterbi algorithm の適用例

2.3.4 スケーリング

Forward-Backward algorithm によって、$\alpha_t(i)$ や $\beta_t(i)$ を求める際、入力系列が長いために、計算が進むにつれてこれらの値が小さくなり、計算機で扱える最小値よりも小さくなることがある（アンダーフロー）。このアンダーフローを回避するために、$\alpha_t(i)$ や $\beta_t(i)$ に適当な係数をを掛けた $\alpha'_t(i)$ や $\beta'_t(i)$ を用いるスケーリングという方法が知られている [7] [6]。スケーリングを導入した Forward-Backward algorithm は以下のようになる。

スケーリングを適用した Forward algorithm

1. 初期化

   全ての状態 $i(1 \leq i \leq N)$ に対して

$$\alpha^*_0(i) = \alpha_0(i) = \pi_i \tag{2.34}$$

$$\alpha'_0(i) = C_0\alpha^*_0(i) = C_0\alpha_0(i) \tag{2.35}$$

   とする。

2. 導出過程

   時間軸 $(t = 1, ..., T)$ に沿って、全ての状態 $i(1 \leq i \leq N)$ に対し、α'_t を次のように計算する。

$$\alpha^*_t(i) = \sum_{j=1}^{N} \alpha'_{t-1}(j)a_{ji}b_{ji}(o_t) \tag{2.36}$$

$$\alpha'_t(i) = C_t\alpha^*_t(i) = C_0C_1...C_t\alpha_t(i) \tag{2.37}$$

   とする。

3. 結果

$$P(O \mid \lambda) = \prod_{t=0}^{T} C_t \tag{2.38}$$

   この式は積の形であるので、実際に計算する時はアンダーフローを回避するため対数で $P(O \mid \lambda)$ を算出する。

$$\log P(O \mid \lambda) = \log - \sum_{t=0}^{T} C_t \tag{2.39}$$

但し、時刻 t におけるスケーリング係数 C_t は以下の式で求める。

$$C_t = \left[\sum_{i=1}^{N} \alpha_t^*(i)\right]^{-1} \tag{2.40}$$

これにより、全時刻において

$$\sum_{i=1}^{N} \alpha_t'(i) = 1 \tag{2.41}$$

となり、アンダーフローを回避できる。

スケーリングを適用した Backward algorithm

Backward algorithm では Forward algorithm で算出したスケーリング定数を用いる。

1. 全ての状態 $i(1 \leq i \leq N)$ に対して

$$\beta_T^*(i) = \beta_T(i) = 1 \tag{2.42}$$

$$\beta_T'(i) = C_T \beta_T^*(i) = C_T \beta_T(i) \tag{2.43}$$

とする。

2. 時間軸 $(t = T-1, ..., 0)$ に沿って、全ての状態 $i(1 \leq i \leq N)$ に対し、β_t' を次のように計算する。

$$\beta_t^*(i) = \sum_{j=1}^{N} a_{ij} b_{ij}(o_{t+1}) \beta_{t+1}'(j) \tag{2.44}$$

$$\beta_t'(i) = C_t \beta_t^*(i) = C_t C_{t-1} ... C_t \beta_t(i) \tag{2.45}$$

スケーリング法を用いて算出した $\alpha_t'(i)$、$\beta_t'(i)$ とスケーリングを用いない $\alpha_t(i)$、$\beta_t(i)$ の間には次式のような関係がある。

$$\alpha_t'(i) = \prod_{\tau=1}^{t} C_\tau \alpha_t(i) \tag{2.46}$$

$$\beta_t'(i) = \prod_{\tau=t}^{t} C_\tau \beta_t(i) \tag{2.47}$$

スケーリング適用時のパラメータの再推定

スケーリングを行なう場合の各パラメータの更新式は次式となる。

1. 初期状態確率

$$\bar{\pi}_i = \alpha'_0(i)\beta'_0(i) \qquad (1 \leq i \leq N) \tag{2.48}$$

2. 状態遷移確率

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha'_{t-1}(i) a_{ij} b_{ij}(o_t) \beta'_t(j)}{\sum_{t=1}^{T} \alpha'_{t-1}(i) \beta'_{t-1}(i) / C_{t-1}} \tag{2.49}$$

3. シンボル出力確率

$$\bar{b}_{ij}(o_t) = \frac{\sum_{t=1 \ s.t. o_t = v_k}^{T} \alpha'_{t-1}(i) a_{ij} b_{ij}(o_t) \beta'_t(j)}{\sum_{t=1}^{T} \alpha'_{t-1}(i) a_{ij} b_{ij}(o_t) \beta'_t(j)} \tag{2.50}$$

第 3 章

HMM による言語のモデル化

確率つきネットワーク文法と全状態間の遷移の許された離散型 ergodic HMM の構造とは非常に類似している。確率つきネットワーク文法は、状態遷移確率と単語（品詞）出力確率を用いて記述した言語モデルであり、ergodic HMM の出力シンボルを単語（品詞）とすれば、両者は等価となる。また、Baum-Welch algorithm によって、入力データから HMM の各パラメータを推定することができる。そこで、実際の会話から作成した文ごとの品詞情報を持たない単語列を ergodic HMM に学習させて、確率つきネットワーク文法を自動的に抽出することを試みた。品詞情報を持たない単語列を学習させることにより、文法だけでなく、単語のカテゴリーも出力の偏りとして同時に学習されることが期待できる。

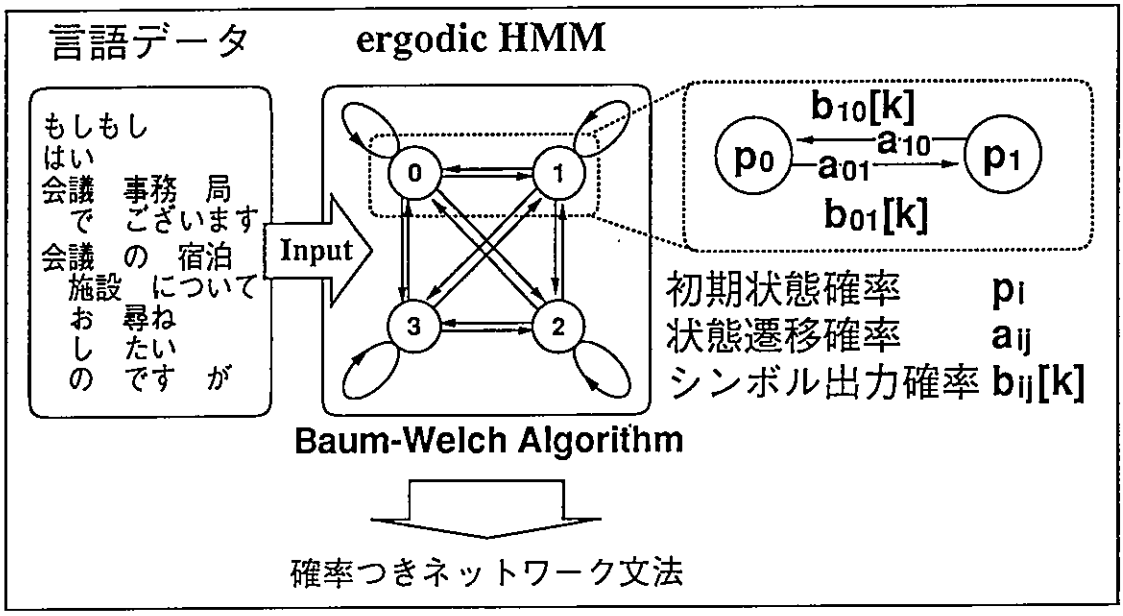


図 3.1: HMM を用いた文法の自動獲得

3.1 言語データ

HMM の学習に用いる言語データベースとして、ADD(ATR 対話データベース)[8][9] を用いた。ADD は情報の伝達や行為の要求などの明確な目的を持った「目的指向型」の対話を対象に、模擬会話実験によってデータベースを収集したものであり、対話メディアは電話が主体である。本研究では、2種ある対話内容から“国際会議に関する問い合わせ”の 8000 文を用いた。文の例を表 3.1 に示す。

表 3.1 のように、このデータベースはあらかじめ単語（形態素）に区切られており、同じ表記でも読み方の異なる場合や、品詞、活用形、活用型の異なる場合は、別単語として扱っている（表 3.2 参照）。また、日本語において、単語の概念はあいまいであるが、単語の単位はデータベースの形態素解析 [8] に依存した。

表 3.1: 文の例

はい もしもし
えーっと そちら 第 1 回 の 通訳 電話 国際 会議 の 事務 局で しょ う か
はい そう です
えーっと ちょっと その 会議 の こと でね
はい どうぞ
えーっと 今 手元 に あの 登録 用紙 が ある ん です けれども
えーっと その 中 で ちょっと あの クレジットカード を ね
あのー クレジットカード の 名前 と なん か ナンバー を 書く ところ が ある ん です が
はい そう です
えーっと それ を ちょっと クレジットカード を 持っ て い ない 者 が いる ん です けれども
その 場合 は どう な ん でしょ う か

表 3.2: 同一表記の単語の扱い

車を持って ない 人もいる。	助動詞連体形
がまんするほか ない。	形容詞終止形
これしか ない のですか。	形容詞連体形

実験に使用したデータベース中の単語の種類は全部で 6418 種類である。品詞は 25 種類に分類され、活用を持つものはさらに活用形、活用型の違いで分類している（表 3.3 表 3.4 表 3.5）。

表 3.3: 品詞分類

形容詞	副詞	副助詞	接頭語	間投詞
普通名詞	連体詞	接続助詞	補助動詞	準体助詞
サ変名詞	接続詞	格助詞	固有名詞	並立助詞
代名詞	感動詞	終助詞	形容名詞	係助詞
数詞	助動詞	接尾語	本動詞	慣用句

表 3.4: 活用形分類

変則型	五段	上一	下一	サ変	カ変	特殊
文語四段	文語上二	文語下二	文語ラ変	文語ナ変	形容詞ク変	

表 3.5: 活用型分類

未然	連用	終止	連体	仮定	命令	語幹

また、これらの単語には数字のラベルが付いており、このデータでは 0 から 6417 までの数字が付けられている（表 3.6 ）。

表 3.6: 文 (単語番号) の例

11	71														
119	65	265	206	315	7	119	296	402	38	7	94	93	33	30	16
11	31	8													
119	75	191	38	7	29	21	25								
11	371														
119	112	690	10	23	67	129	19	143	20	8	36				
119	47	199	21	75	23	757	17	25							
24	757	7	132	84	794	69	1457	17	3104	177	19	143	20	8	32
11	31	8													
119	55	17	75	757	17	256	9	55	889	310	19	1260	20	8	36
47	168	12	101	41	20	33	30	16							

表 3.7: 構成単語数

set	odd1000	odd2000	odd4000	even1000	even2000	even40000
文数	1000	2000	4000	1000	2000	4000
単語数	13299	20730	57354	13824	21114	56826
文平均単語数	13.30	10.37	14.34	13.82	10.56	14.21
最大単語数	99	99	128	81	81	118

実験ではデータベースの 8000 文を奇数番目の文の set と偶数番目の文の set とに分け、さらにそれぞれ先頭から 1000 文の set、先頭から 2000 文の set、4000 文の set に分けている。奇数番目の set3 種類をそれぞれ odd1000, odd2000, odd4000、偶数番目の set を even1000, even2000, even4000 と呼ぶことにする。それぞれの set での品詞の出現頻度などを表 3.7 ～表 3.15 に示す。

表 3.8 ～表 3.13 から、このデータベースの文は主に、普通名詞、格助詞、本動詞、助動詞で構成されており、また、文が電話対話であるため、間投詞（あのー、えーなど）や感動詞（もしもし、はい など）が多く含まれていることがわかる。また、一文章の平均単語数が 13 前後になっっている（表 3.7 ）が、実際の分布は単語数の少ない方に偏っている。これは、「はい。」「もしもし。」の様な感動詞 1 単語のみからなる文や「わかりました。」「そうですか。」などの受け答えの会話が多く存在するためであり、電話対話という特殊な環境を反映した言語データであることがわかる。

表 3.8: 品詞別出現頻度 (odd1000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	普通名詞	1915	14.40	1.915
2	助動詞	1900	14.29	1.900
3	格助詞	1651	12.41	1.651
4	本動詞	1028	7.73	1.028
5	間投詞	912	6.86	0.912
6	接続助詞	780	5.87	0.780
7	補助動詞	596	4.48	0.596
8	感動詞	552	4.15	0.552
9	終助詞	514	3.86	0.514
10	副詞	512	3.85	0.512

表 3.9: 品詞別出現頻度 (odd2000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	普通名詞	2937	14.17	1.468
2	助動詞	2878	13.88	1.439
3	格助詞	2586	12.47	1.293
4	本動詞	1672	8.07	0.836
5	間投詞	1473	7.11	0.737
6	感動詞	1307	6.30	0.653
7	接続助詞	1208	5.83	0.604
8	補助動詞	858	4.14	0.429
9	副詞	775	3.74	0.388
10	終助詞	712	3.43	0.356

表 3.10: 品詞別出現頻度 (odd4000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	普通名詞	8590	14.98	2.147
2	格助詞	7832	13.66	1.958
3	助動詞	7778	13.56	1.944
4	本動詞	4886	8.52	1.222
5	間投詞	4404	7.68	1.101
6	接続助詞	3813	6.65	0.953
7	補助動詞	2849	4.97	0.712
8	副詞	2065	3.60	0.516
9	感動詞	1997	3.48	0.499
10	終助詞	1586	2.77	0.397

表 3.11: 品詞別出現頻度 (even1000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	普通名詞	2089	15.11	2.089
2	助動詞	1989	14.39	1.989
3	格助詞	1819	13.16	1.819
4	本動詞	1105	7.99	1.105
5	間投詞	938	6.79	0.938
6	接続助詞	816	5.90	0.816
7	補助動詞	646	4.67	0.646
8	感動詞	509	3.68	0.509
9	副詞	503	3.64	0.503
10	終助詞	463	3.35	0.463

表 3.12: 品詞別出現頻度 (even2000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	助動詞	3008	14.25	1.504
2	普通名詞	3004	14.23	1.502
3	格助詞	2662	12.61	1.331
4	本動詞	1756	8.32	0.878
5	間投詞	1473	6.98	0.737
6	接続助詞	1258	5.96	0.629
7	感動詞	1237	5.86	0.619
8	補助動詞	913	4.32	0.457
9	副詞	780	3.69	0.390
10	終助詞	664	3.14	0.332

表 3.13: 品詞別出現頻度 (even4000)

順位	品詞名	出現数	割合 (%)	一文当たり出現頻度
1	普通名詞	8424	14.82	2.106
2	助動詞	7948	13.99	1.987
3	格助詞	7744	13.63	1.936
4	本動詞	4932	8.68	1.233
5	間投詞	4281	7.53	1.070
6	接続助詞	3723	6.55	0.931
7	補助動詞	2835	4.99	0.709
8	副詞	2072	3.65	0.518
9	感動詞	1973	3.47	0.493
10	終助詞	1547	2.72	0.387

表 3.14: 文構成単語数 1

	odd1000		odd2000		odd4000	
文の単語数	頻度	(%)	頻度	(%)	頻度	(%)
1	127	12.70	399	19.95	610	15.25
2	18	1.80	86	4.30	140	3.50
3	77	7.70	125	6.25	206	5.15
4	152	15.20	222	11.10	356	8.90
5	52	5.20	92	4.60	190	4.75
1 ～ 10	570	57.00	1273	63.65	2136	53.40
11 ～ 20	201	20.10	453	22.65	901	22.53
21 ～ 30	113	11.30	153	7.65	449	11.22
31 ～ 40	62	6.20	66	3.30	252	6.30
41 ～ 50	30	3.00	30	1.50	111	2.77
51 ～ 130	24	2.40	25	1.25	148	3.70

表 3.15: 文構成単語数 2

	even1000		even2000		even4000	
文の単語数	頻度	(%)	頻度	(%)	頻度	(%)
1	107	10.70	375	18.75	547	13.68
2	15	1.50	79	3.95	131	3.27
3	75	7.50	134	6.70	259	6.47
4	139	13.90	210	10.50	362	9.05
5	51	5.10	87	4.35	172	4.30
1 ～ 10	526	52.60	1239	61.95	2107	52.68
11 ～ 20	232	23.20	482	24.10	943	23.57
21 ～ 30	128	12.80	160	8.00	454	11.35
31 ～ 40	64	6.40	66	3.30	250	6.25
41 ～ 50	30	3.00	32	1.60	116	2.90
51 ～ 130	20	2.00	21	1.05	130	4.25

3.2 実験条件

前節で作成した対話データ set odd1000, odd2000, odd4000 を学習データ (closed data) として ergodic HMM の状態数を変えてモデル化の実験を行なった。また、実験に用いた ergodic HMM には開始状態、終了状態を与えず、したがって任意の状態から状態遷移を開始し、任意の状態で終了できるモデルになっている。学習を始める際のモデルのパラメータ A, B, π の初期値は以下の通りである。

$$\begin{aligned} \pi: &\quad \pi_i = 1/N \\ A: &\quad a_{ij} = \text{ランダム} \\ B: &\quad b_{ij}(v_k) = \text{ランダム} \end{aligned}$$

ただし、 N : 状態数, L : 語彙数 (単語の種類) , $1 \leq i, j \leq N$, $1 \leq k \leq L$

$$\sum_{j=1}^{N} a_{ij} = 1.0$$

$$\sum_{k=1}^{L} b_{ij}(v_k) = 1.0$$

である。

HMM の学習には Baum-Welch algorithm を用いた。一般に、モデル $\lambda(A, B, \pi)$ の繰り返し行なうと、学習に対するモデルの尤度 $P(O \mid \lambda)$ は徐々にある一定値に近づいて増加する。そこで、学習 (パラメータ再推定) 終了条件として、モデルがシンボル系列を生成する確率の上昇率

$$\frac{\text{学習後の生成確率} - \text{前回の学習後の生成確率}}{\text{学習後の生成確率}}$$

がある一定値以下になるまで学習を繰り返す方法をとった。

実験は、状態数の異なる 4 種類の ergodic HMM(2 状態、 4 状態、 8 状態、 16 状態) の場合について行なった。学習データは odd1000, odd2000, odd4000 を用いた (16 状態は odd4000 のみ)。また、初期パラメータを変えた場合のモデル化の変化を調べるため、初期状態の異なる 8 種の 8 状態の ergodic HMM について、再推定回数 20 回を学習終了条件として odd4000 を学習させた。その他の実験の条件を表 3.16 に示す。

表 3.16: 言語モデル生成実験の条件

HMM の構造	状態遷移出力型 ergodic HMM
HMM の状態数	2 状態,4 状態,8 状態,16 状態
HMM の出力シンボル	単語
開始・終了状態	任意
初期状態遷移確率	ランダム
初期シンボル出力確率	ランダム
初期状態確率	均等
語彙数	6418
学習データ set	odd4000,odd2000,odd1000
学習データ数	4000 文,2000 文,1000 文
単語総数	57354 単語,20730 単語,13299 単語
学習終了条件	生成確率上昇率 1% 未満

3.3 実験結果

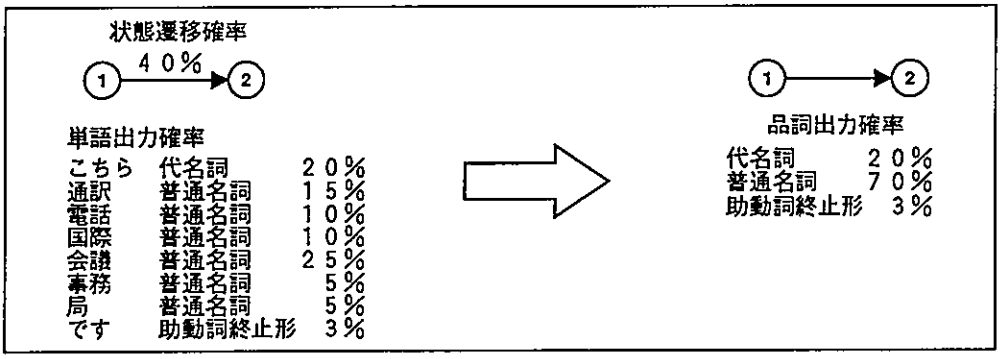
学習されたergodic HMMを評価するために、HMMを解析して獲得されたモデルの特徴を調べ、closed dataとopen dataの文の生成確率を求めて比較した。

3.3.1 ergodic HMMの解析手順

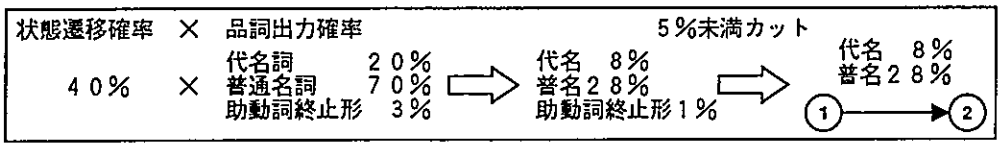
1. 単語に品詞(活用するものは活用形の区別をしている)のラベルをつける。

こちら	通訳	電話	国際	会議	事務	局	です
代名詞	普通名詞	普通名詞	普通名詞	普通名詞	普通名詞	普通名詞	助動詞終止形

2. 各遷移について、同一品詞の単語のシンボル出力確率の和を品詞ごとに求める。



3. (状態遷移確率 × 出力確率)が5%未満の品詞はカットしてネットワークを表示する。



ネットワークの表示に用いた略号は表3.17の通りである。

以後示すネットワークの図において、遷移の太細は遷移確率の大小を示し、品詞名、活用形の略号の右の数字は(状態遷移確率 × 出力確率)の値を示している。また、初期確率が最大になっている状態(イニシャルノード)を太丸で示した。

表 3.17: 品詞・活用形の略号

品詞名	略号	品詞名	略号	品詞名	略号	品詞名	略号	品詞名	略号
普通名詞	普名	助動詞	助動	格助詞	格助	準体助詞	準助	未然形	未
代名詞	代名	間投詞	間投	係助詞	係助	接頭辞	接頭	終始形	終
本動詞	本動	感動詞	感動	接続助詞	接助	接尾辞	接尾	連体形	体
補助動詞	補動	副詞	副詞	終助詞	終助	連用形	用		

3.3.2 2状態 ergodic HMM の解析結果

2状態の ergodic HMM について解析した結果を図 3.2 に、初期状態確率、状態遷移確率を表 3.18 に示す。

2状態の ergodic HMM では、主として、①⇒ ⓪の遷移で普通名詞を出力し、これに続く状態⓪からの遷移 (⓪ ⇒ ①, ⓪ ⇒ ⓪) で普通名詞に接続する格助詞を出力している。また、実験に用いた言語データは前述した通り電話での対話であるため、間投詞 (あのー、えー、など) や感動詞 (もしもし、はい、など) が文頭や文の切れ目で用いられた文が多く含まれている。これらが ergodic HMM では、太丸で示したイニシャルノード (= 遷移を開始する状態 = 文頭) ①のループで出力されており、学習データの特徴を示している。しかし、状態数が少ないために、全体的に表現力が乏しく、文法を表現する特徴は抽出するのは困難である。

表 3.18: 2状態 ergodic HMM のパラメータ

初期状態確率	$\pi_0 = 0.002$	$\pi_1 = 0.998$
状態遷移確率	$a_{00} = 0.36$	$a_{01} = 0.64$
	$a_{10} = 0.54$	$a_{11} = 0.46$

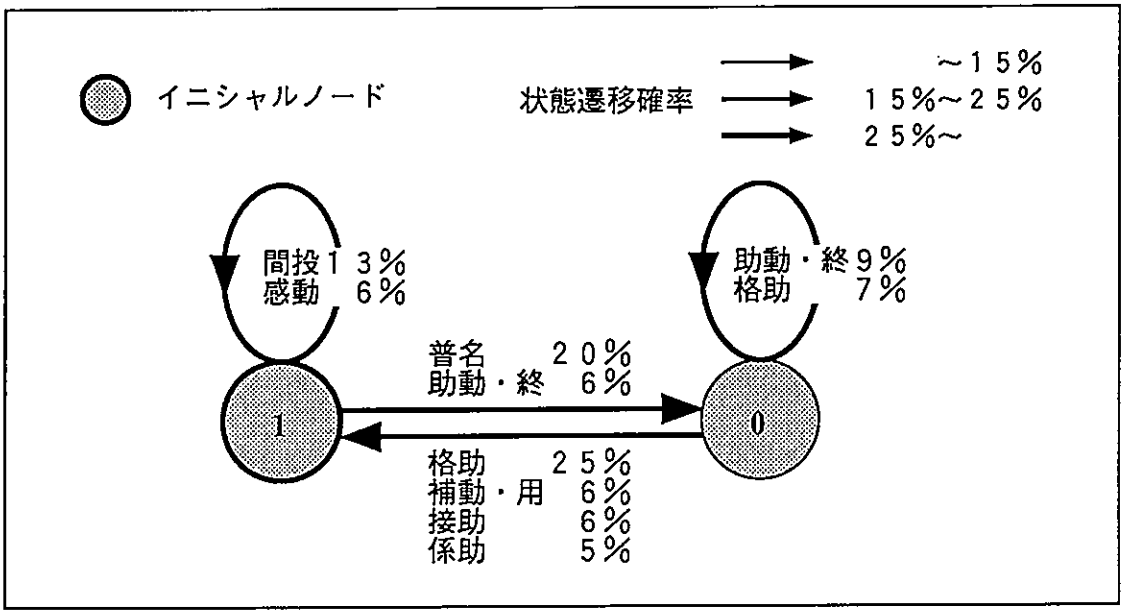


図 3.2: 2状態 ergodic HMM の解析結果

3.3.3 4 状態 ergodic HMM の解析結果

4 状態の ergodic HMM について解析した結果を図 3.3 に、初期状態確率、状態遷移確率を表 3.19 に示す。

2 状態の ergodic HMM に比べ、遷移の数が増えて表現力が増し、文法的な特徴も見られる。また、単語の分離が生じ、ネットワーク上では品詞ごとに固まって出力されているのがわかる。

体言・活用する品詞の分離

体言は主として、状態⓪に集まる遷移 (②⓪⇒⓪) で出力され、活用する品詞（本動詞、補助動詞、助動詞、形容詞）は状態②③に集まる遷移 (②③⇒②, ⓪①②⇒③) で出力されており、体言と活用する品詞が分離されているのが見られる。活用する品詞の中で、連体形のものは状態②に集まる遷移 (②③⇒②) で出力され、連用形のものは主に状態③に集まる遷移 (①②③⇒③) で出力されている。連体形の集まる状態②からは体言を出力する遷移がのびており、また、連用形の集まる状態③からは助動詞を出力する遷移が出ており、文法にかなった特徴を示している。

2 状態の ergodic HMM で見られた、文頭の間投詞、感動詞は 4 状態の ergodic HMM でもイニシャルノード①のループで出力されている。また、体言を出力した遷移の集まる状態⓪からの遷移 (⓪⇒⓪①) で格助詞、係助詞が出力され、体言から格助詞への接続を表現している。

表 3.19: 4 状態 ergodic HMM のパラメータ

初期状態確率	$\pi_0 = 0.00$	$\pi_1 = 1.00$	$\pi_2 = 0.00$	$\pi_3 = 0.00$
状態遷移確率	$a_{00} = 0.38$	$a_{01} = 0.41$	$a_{02} = 0.07$	$a_{03} = 0.14$
	$a_{10} = 0.27$	$a_{11} = 0.43$	$a_{12} = 0.17$	$a_{13} = 0.13$
	$a_{20} = 0.35$	$a_{21} = 0.11$	$a_{22} = 0.22$	$a_{23} = 0.32$
	$a_{30} = 0.04$	$a_{31} = 0.48$	$a_{32} = 0.23$	$a_{33} = 0.25$

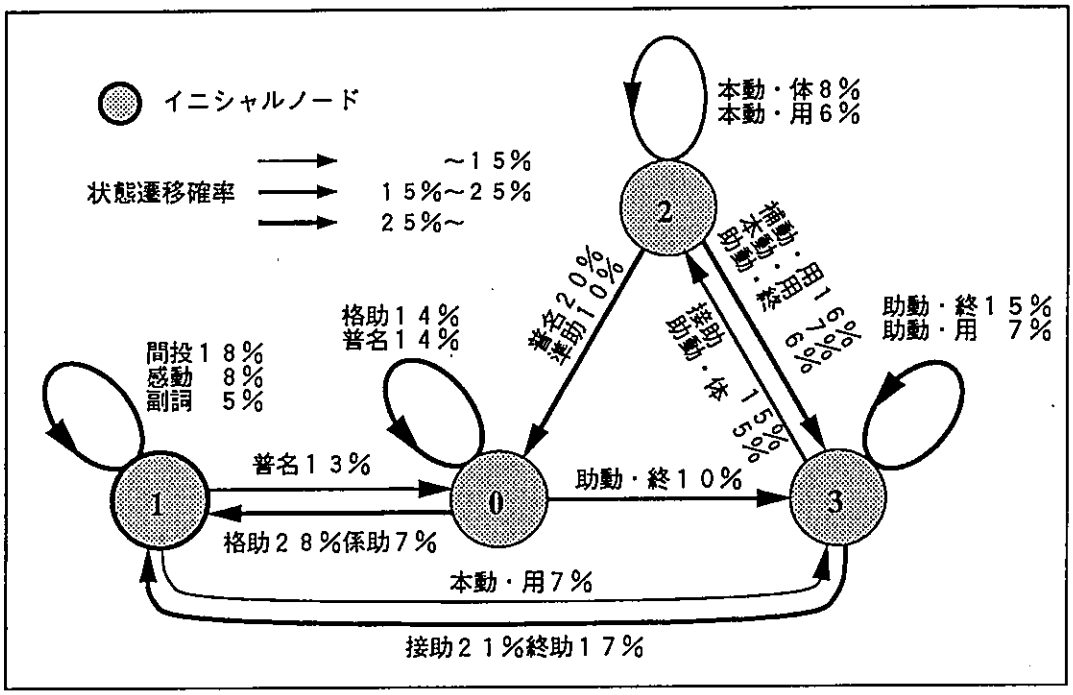


図 3.3: 4 状態 ergodic HMM の解析結果

3.3.4 8 状態 ergodic HMM の解析結果

8 状態の ergodic HMM について解析した結果を図 3.4 に、初期状態確率、状態遷移確率を表 3.20 に示す。

表 3.20: 8 状態 ergodic HMM のパラメータ

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
初期状態確率 π_j	0.999	0.000	0.001	0.000	0.000	0.000	0.000	0.000
状態遷移確率 a_{ij}								
i=0	0.20	0.15	0.18	0.05	0.05	0.03	0.02	0.33
i=1	0.02	0.04	0.05	0.05	0.04	0.35	0.28	0.17
i=2	0.29	0.05	0.12	0.03	0.15	0.24	0.02	0.10
i=3	0.02	0.57	0.23	0.03	0.01	0.05	0.02	0.06
i=4	0.02	0.01	0.08	0.19	0.18	0.26	0.24	0.03
i=5	0.04	0.04	0.10	0.10	0.31	0.06	0.17	0.19
i=6	0.27	0.02	0.19	0.03	0.16	0.04	0.01	0.28
i=7	0.05	0.14	0.24	0.09	0.05	0.04	0.09	0.30

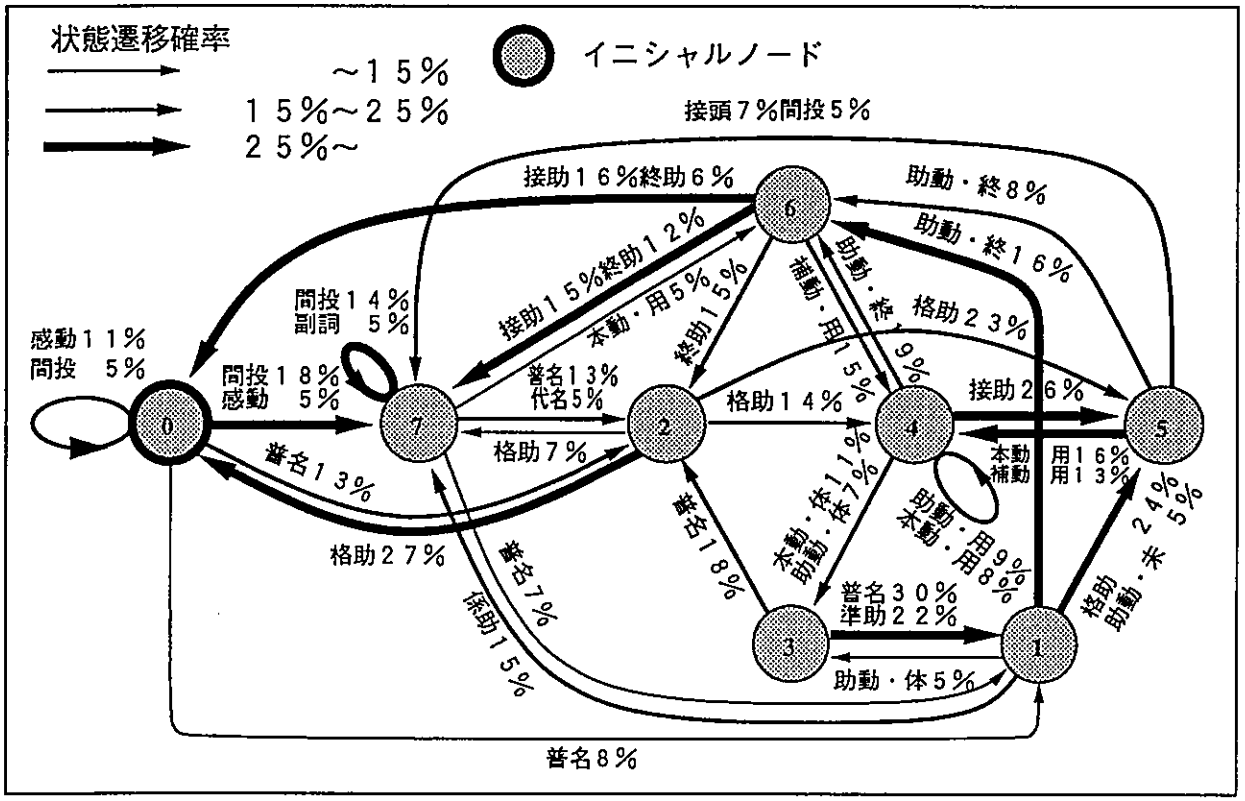


図 3.4: 8 状態 ergodic HMM の解析結果

図 3.4 から、文中で同じような役割を果たす (品詞、活用形が等しい) 単語が、4 状態の場合よりもさらに偏って出力されていることがわかる。体言や活用する品詞など同一品詞が複数の遷移で出力されているが、出力品詞が同じ遷移は遷移の起点または終点が特定の状態に偏っている。また、2 状態、4 状態の場合にに比べ表現力が増すことにより、より細かな文法的特徴が見られる。

以下で 8 状態 ergodic HMM から抽出される細かい特徴を述べる。

<u>間投詞、感動詞について</u>

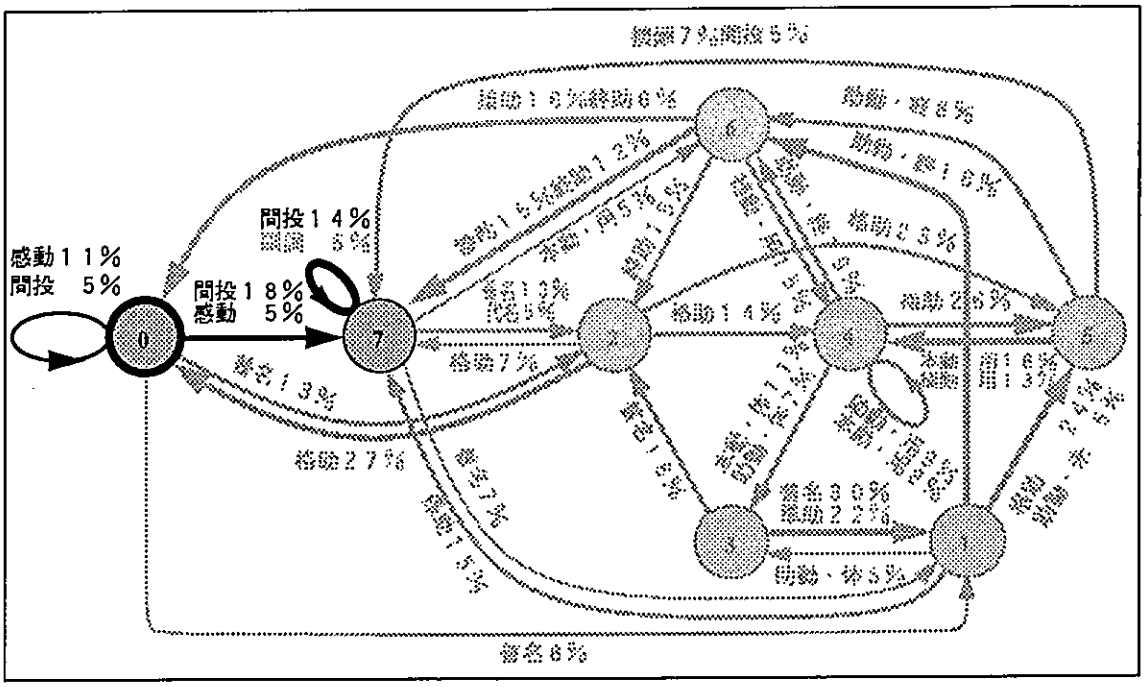


図 3.5: 間投詞・感動詞の出力

2 状態、4 状態と同様に 8 状態の ergodic HMM でも間投詞、感動詞がイニシャルノード⓪からの遷移 (⓪⇒⓪, ⓪⑦⇒⑦) で出力されており、学習データの特徴を表現している。出力される主な単語は、以下の通りである。括弧内の数字は、シンボル出力確率である。

- 状態⓪⇒ 状態⓪ 「はい」 (50%) 「あ」 (10%)
- 状態⓪⇒ 状態⑦ 「え」 (19%) 「あのー」 (10%) 「はい」 (8%) 「え」 (7%)
- 状態⑦⇒ 状態⑦ 「あの」 (16%) 「えー」 (10%) 「あのー」 (9%) 「はい」 (6%)

体言について

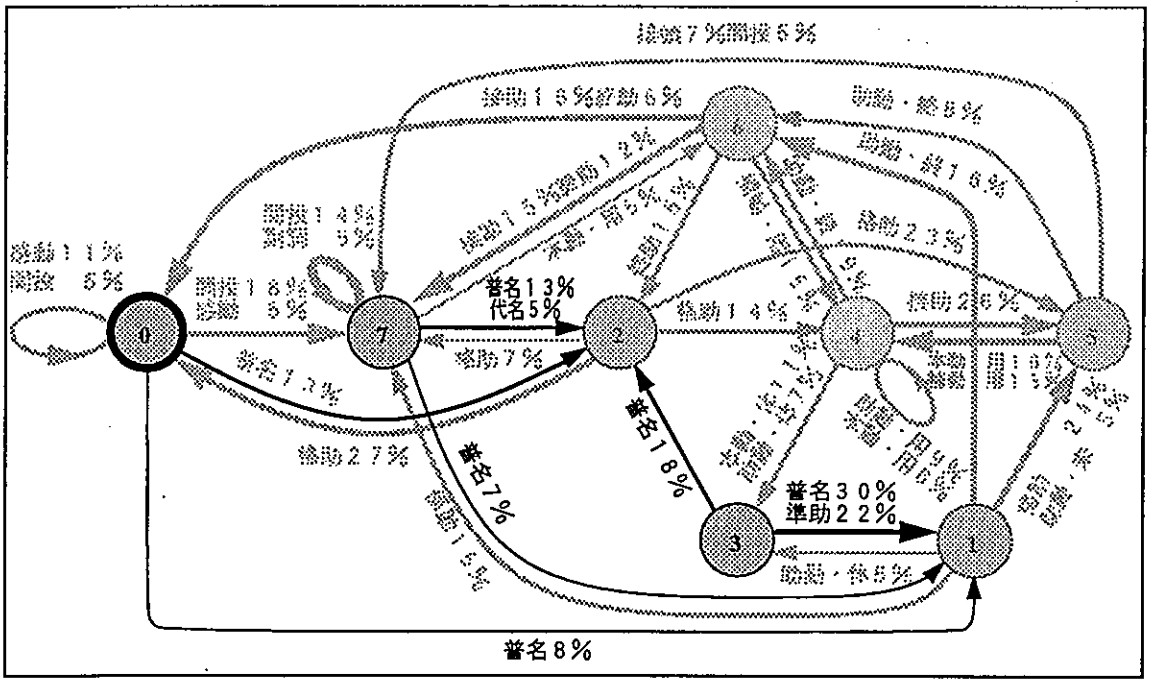


図 3.6: 体言の出力

図 3.4 では、複数の品詞を出力する遷移が見られる。しかし、ergodic HMM は活用する品詞、体言を分離し、別々の遷移で出力している。体言は主として状態⓪③⑦から状態①②に遷移する際に出力されている (図 3.6 参照)。出力されている単語は以下の通りである。

- 状態⓪⇒ 状態①
  「方（ほう）」(11%) 「方（かた）」(5%) 以下 「それ」「私」「これ」

- 状態⓪⇒ 状態②
  「方（ほう）」(23%) 「会議」(7%) 以下 「事務」「方（かた）」「中」

- 状態③⇒ 状態①
  「ん」(準体助詞, 31%) 「こと」(18%) 以下「の」「もの」「わけ」

- 状態③⇒ 状態②
  「こと」(13%) 「時」(4%) 以下 「ふう」「用紙」

- 状態⑦⇒ 状態①
  「私」(9%) 「それ」(5%) 以下 「結構」「これ」「名前」

- 状態⑦⇒ 状態②
  「こちら」(7%) 以下 「会議」「そちら」「先生」「私」

上記の単語以外でも、体言の単語が多数出力されているのがどの遷移でも確認された。

状態③からの遷移では、形式名詞（ADDでは普通名詞として扱っている）が他の名詞類よりもやや高い出力確率になっている。また、③⇒①の遷移では準体助詞「ん」と同様に用いられる準体助詞「の」が出力されている。

状態⑦からの遷移では、出力確率の合計では普通名詞が最も高いが、個々の単語出力の上位は代名詞が多い。人に対して用いられる名詞類が多く出力されていることもわかる。

状態⓪からの遷移では、主として形式名詞「方」が出力されている。また、体言以外の単語では副詞の出力が多く見られた(⓪⇒①の遷移では、副詞「そう」の出力確率25%である。)。これは、状態①からの遷移で出力される「です」や「でしょ」をともなって学習データに多く見られた「そうです」「そうですか」などを表現しているため、文頭に当たる状態からの遷移で出力されていると考えられる。

遷移によって出力される単語や品詞が異なり、複数ある体言を出力する遷移でも、個々の表現する内容は異なっていることがわかる。

活用する品詞について

8 状態の ergodic HMM では活用する品詞が、品詞ごとではなく、活用形ごとに集まって出力されている。連体形のものは状態③への遷移 (④,①⇒③) で (図 3.7 参照)、連用形のものは状態④への遷移 (④, ⑤,⑥⇒④) で (図 3.8 参照)、助動詞の終止形は状態⑥への遷移 (①,④,⑤⇒ ⑥) で (図 3.9 参照)、それぞれ出力されている。以下に、活用形ごとに見られる特徴を述べる。

まず、連体形の単語が出力される遷移について述べる。図 3.7 に連体形の出力される遷移を示し、以下にその遷移で出力される主な単語を示す。

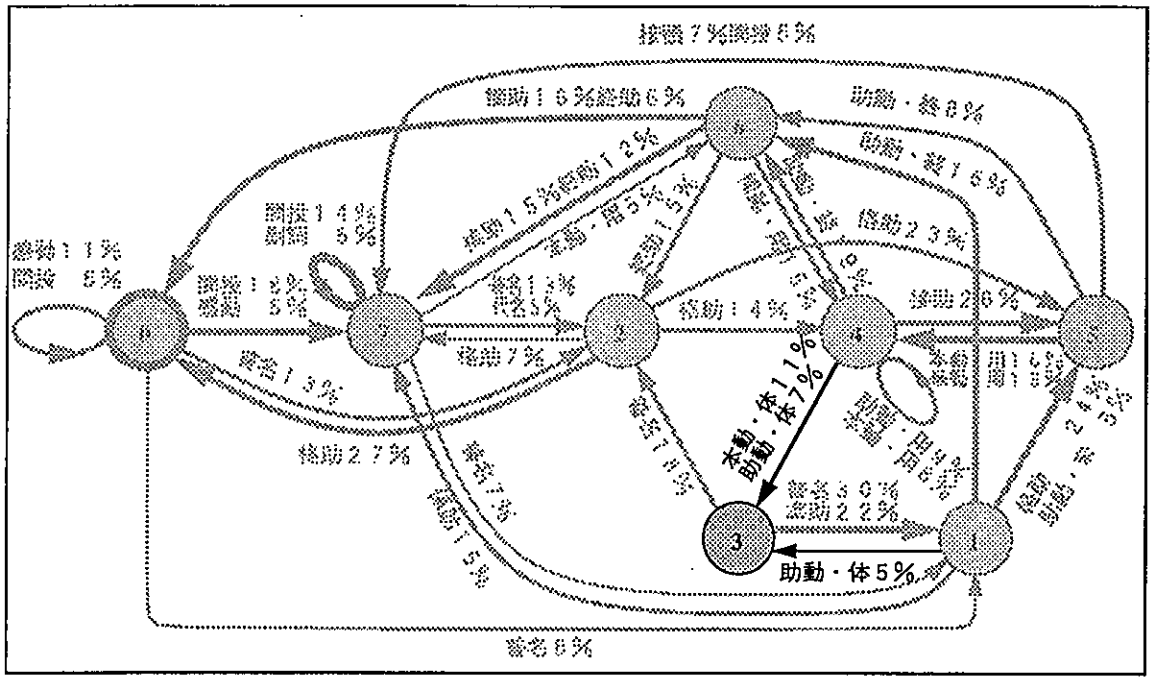


図 3.7: 連体形の出力

- 状態④⇒ 状態③

  「いう」 (50%) 「た」 (26%) 「たい」 (10%) 以下「思う」「ます」

図 3.7 から連体形を出力して遷移する先の状態③は、連用形を出力する遷移と体言を出力する遷移の節点になっていることがわかる。状態③からの遷移では、体言 (形式名詞、準体助詞) が多く出力されていた。ここで、「〜(と) いうこと」や「〜(し) たこと」、「〜(し) たいん (です)」などの「連体形+体言」の接続が表現されていると思われる。

連用形の単語が出力される遷移について述べる。図 3.8 に連用形の出力される遷移を示し、以下にその遷移で出力される主な単語を示す。

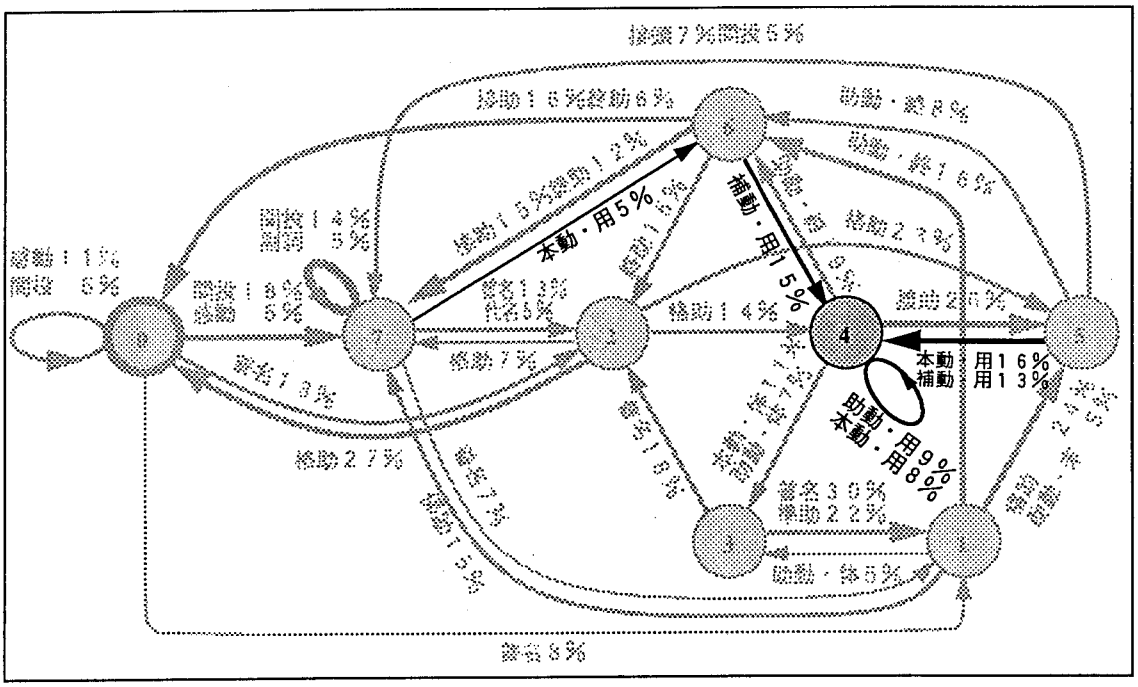


図 3.8: 連用形の出力

- 状態④⇒ 状態④
  「まし」 (50%) 「思い」 (23%) 以下 「申し」「思っ」「いい」

- 状態⑤⇒ 状態④
  「おり」 (15%) 「頂き」 (9%) 以下 「し」「ございまし」「頂い」

- 状態⑥⇒ 状態④
  「し」 (64%) 「いたし」 (21%) 以下「申し上げ」「でき」

- 状態⑦⇒ 状態⑥
  「願い」 (16%) 「送り」 (8%) 以下「待ち」「伺い」

状態④への遷移は連用形を出力するものが多く、状態④からの遷移は活用する品詞、接続助詞 (④⇒⑤) を出力している。状態④は連用形の単語と助動詞・補助動詞との節点と考えられる。④⇒④⑥の遷移で、助動詞の「ます」「まし」が多く出力され、「〜思います」「〜おります」のような接続が考えられる。④⇒⑤の遷移では 97% の確率で「て」が出力され、「〜頂いて」「〜して」「思って」など活用する品詞の「連用形 + 接続助詞 て」を表現している。

⑦⇒⑥の遷移で出力される単語は、本動詞がほとんどで、⑤⇒⑦で出力される (出力確率 34%) 接頭辞「お」と接続して、「お待ち (して) 〜」「お願い (いたし) 〜」などの謙譲表現を形成している。

図 3.9 に終止形の出力される遷移を示す。終止形の出力はほとんどが助動詞である。学習データの内容が会議に申し込みに関する電話対話であるため、本動詞の言いきりの表現が少なく、「〜です」「〜ます」などの助動詞をともなった丁寧な表現が多いためと考えられる。

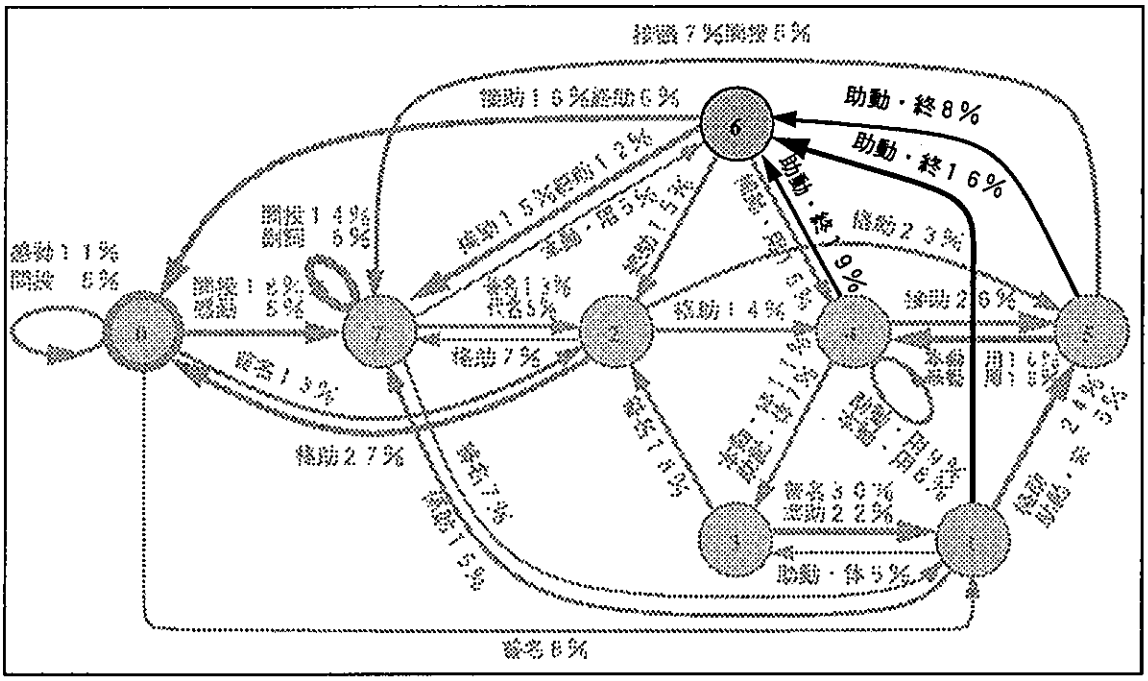


図 3.9: 終止形の出力

以下に終止形を出力する遷移で出力される主な単語を示す。

- 状態①⇒ 状態⑥
  「です」 (94%)

- 状態④⇒ 状態⑥
  「ます」 (69%) 「ます」 (助動詞連体形, 17%) 以下 「た」

- 状態⑤⇒ 状態⑥
  「う」 (37%) 「ございます」 (14%) 「です」 (7%) 以下 「ない」

単語の出力をみると、「です」「ます」が分離して出力されているのがわかる。体言から接続する「です」を出力する遷移が状態①から遷移しているおり、連用形の単語から接続する「ます」が状態④から遷移の遷移で出力されている。状態①は体言を出力した遷移の集まる状態であり、連体形の単語を出力した遷移は状態④に集まっていることから、これらはいずれも文法に合致した接続を表現している。状態⑤から遷移する「う」は推量や意志の意

味を持つ助動詞で、未然形の単語から接続する。状態①⇒ 状態⑤の遷移で助動詞の未然形「でしょ」が 13% 出力されており、「〜でしょう (か)」を表現していると考えられる。

格助詞の分離について

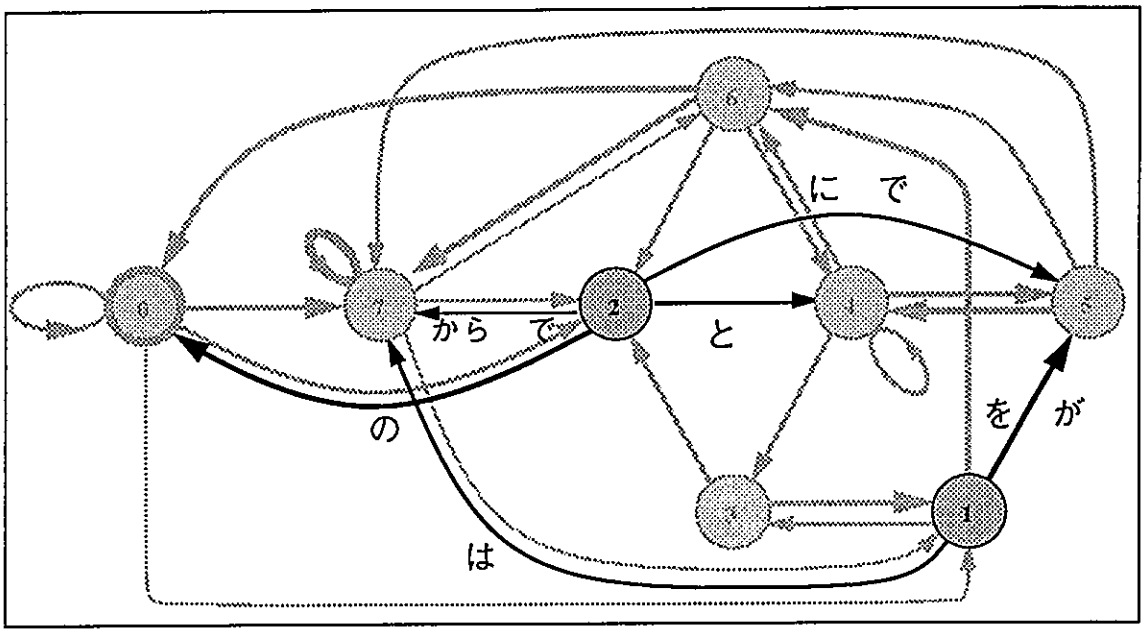


図 3.10: 格助詞の出力

体言を出力する遷移の集まる状態②からの全ての遷移は、格助詞を出力している。(②⇒⓪,④,⑤,⑦, 図 3.10 参照)。体言を出力する遷移の集まるもう一つの状態①からの遷移も、格助詞や係助詞を出力している (①⇒⑤,⑦)。この連鎖は「私は」「会議の」など 名詞 + 格助詞・係助詞 を表現している。複数の遷移で格助詞が出力されているが、後接する品詞の違い (文中での機能の違い) によって各遷移は別々の格助詞を出力していることが、各遷移の単語出力確率を調べた結果分かった。

- 状態②⇒ 状態⓪ 「の」 (94%)
- 状態②⇒ 状態⑦ 「から」 (28%) 「で」 (25%) 以下 「が」「まで」
- 状態②⇒ 状態④ 「と」 (94%)
- 状態②⇒ 状態⑤ 「に」 (73%) 「で」 (18%)
- 状態①⇒ 状態⑤ 「を」 (39%) 「が」 (27%)
- 状態①⇒ 状態⑦ 「は」 (80%) 「も」 (7%)

3.3.5 16 状態 ergodic HMM の解析結果

16 状態の ergodic HMM について解析した結果を図 3.11 に示す。ネットワークが複雑なため、より簡略化した略号を用い (表 ?? 参照)、出力確率値の % は省略して表示した。また、初期状態確率を表 3.22 に示す。

表 3.21: 品詞・活用形の略号 2

品詞名	略号	品詞名	略号	品詞名	略号	品詞名	略号	品詞名	略号
普通名詞	普	助動詞	助	格助詞	格	準体助詞	準	未然形	未
代名詞	代	間投詞	間	係助詞	係	接頭辞	頭	終始形	終
本動詞	本	感動詞	感	接続助詞	接助	接尾辞	尾	連体形	体
補助動詞	補	副詞	副	終助詞	終	連用形	用		

表 3.22: 16 状態 ergodic HMM の初期状態確率

π_0	0.00	π_4	0.00	π_8	0.00	π_{12}	0.00
π_1	0.00	π_5	0.00	π_9	0.00	π_{13}	0.00
π_2	0.89	π_6	0.00	π_{10}	0.00	π_{14}	0.00
π_3	0.11	π_7	0.00	π_{11}	0.00	π_{15}	0.00

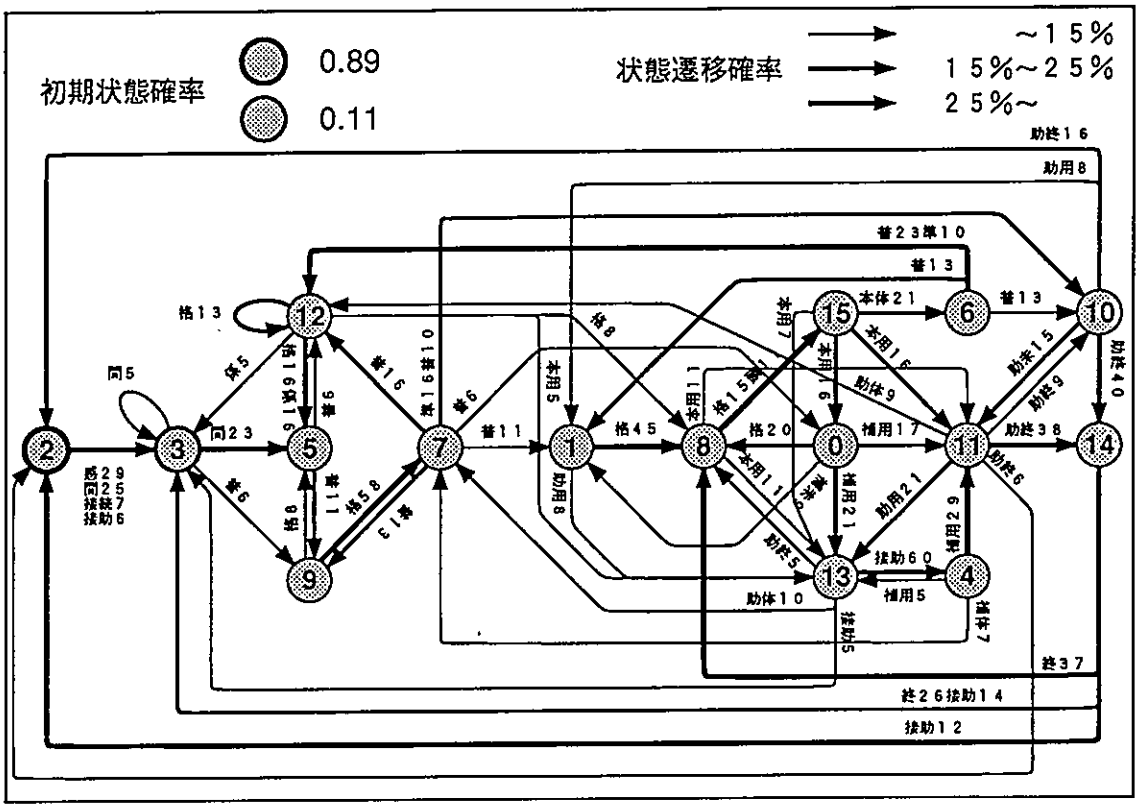


図 3.11: 16 状態 ergodic HMM の解析結果

16 状態の ergodic HMM では、状態数の少ないモデルに比べ、ほとんど全ての遷移で一品詞のみが出力され、単語の出力の偏りがさらに顕著になっている。8 状態に比べ、活用する品詞の記述がより細かくなり、また、2 つの状態 (②, ③) で初期状態確率の値を持つなど、より複雑なネットワークを形成している。8 状態の場合と同様に同一品詞が複数の遷移で出力される場合は、遷移の起点か、または終点が同じ状態であることが多い。主に、ネットワークの左の部分で「名詞＋格助詞」などの主部が記述され、右の部分で「動詞＋助動詞」などの述部が記述されている。以下では、16 状態 ergodic HMM から抽出される特徴を品詞ごとに述べる。

間投詞、感動詞について

図 3.12 に間投詞・感動詞を出力する遷移を示す。間投詞、感動詞は 2、4、8 状態の場合と同様で、初期状態確率の高い状態 (②, ③) からの遷移で出力される。

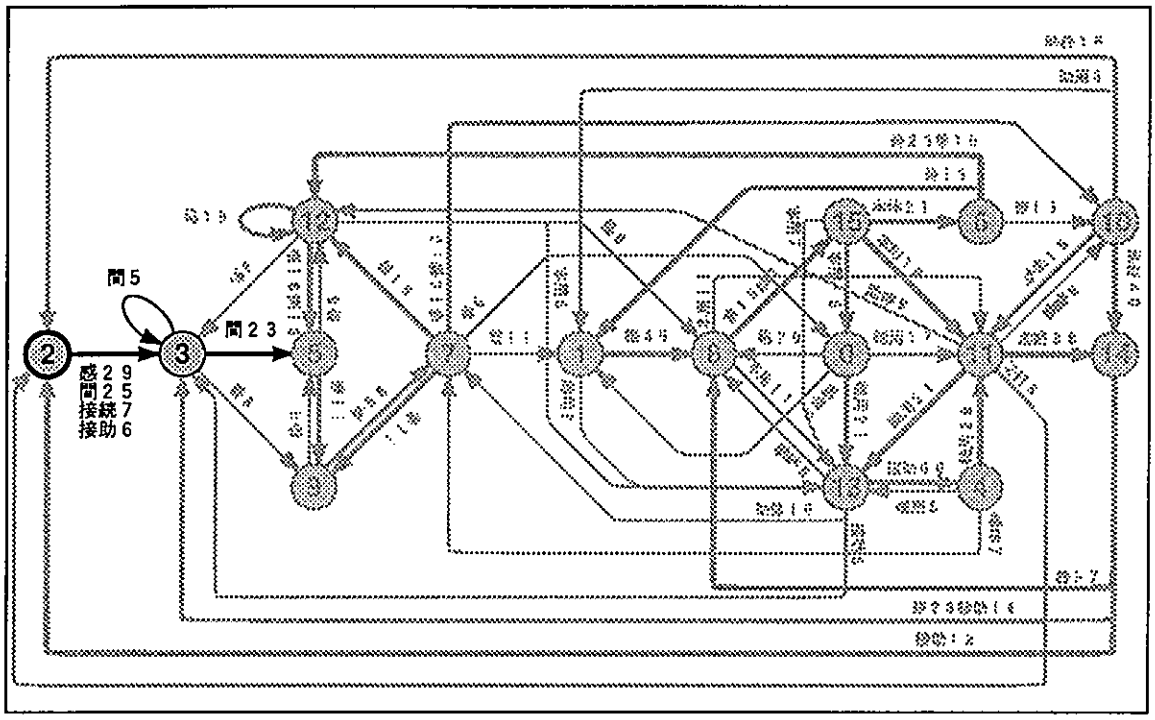


図 3.12: 間投詞・感動詞の出力

- 状態②⇒ 状態③
  「はい」 (33%) 「えー」 (16%) 以下 接続詞「けれども」「え」「えーと」
- 状態③⇒ 状態③
  「え」 (8%) 「ま」 (8%) 以下 「まあ」 「あの」
- 状態③⇒ 状態⑤
  「あの」 (32%) 「あのー」 (24%) 「えー」 (19%)

状態数の少ない場合と異なり、初期状態確率が最も高い状態②での自己ループが見られない。初期状態確率が状態②③の 2 状態が値を持つことと考え合わせると、8 状態の場合にイニシャルノードからの遷移で表現していたものを、16 状態では 2 つの状態からの遷移で表現していると考えられる。

体言について

図 3.13 に体言が出力される遷移を示す。

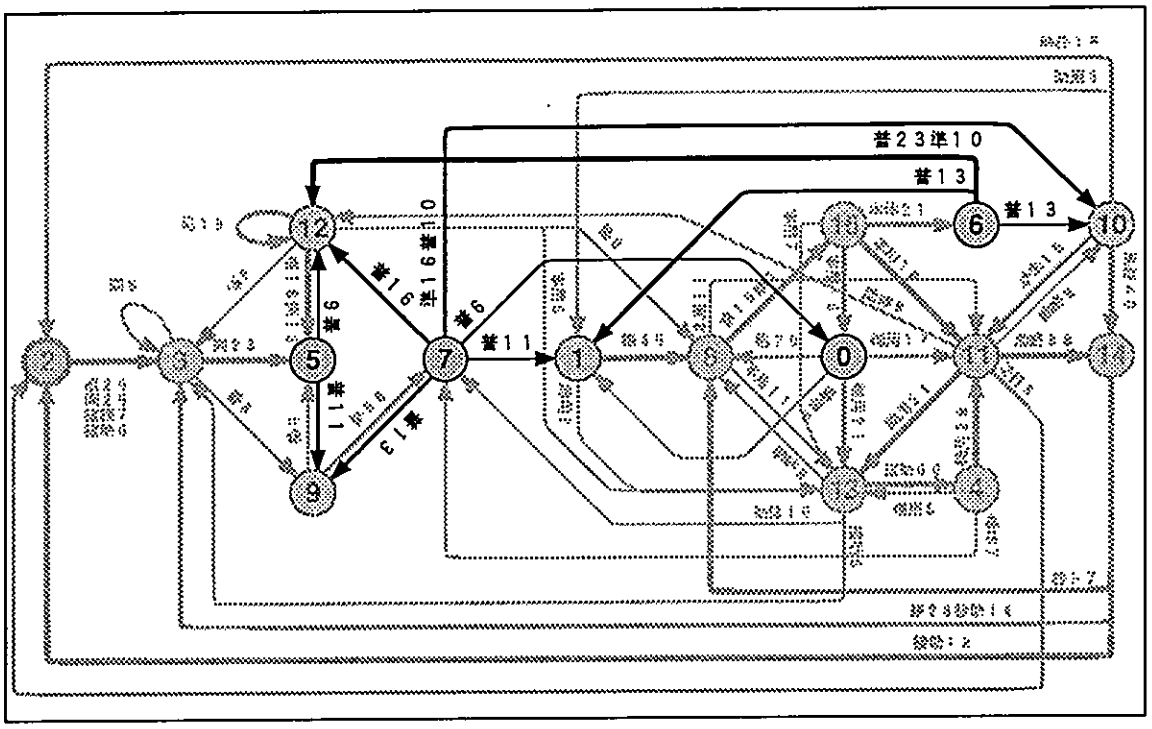


図 3.13: 体言の出力

体言は主として、⑤, ⑥, ⑦から①, ⑨, ⑫への遷移で出力されている。主に出力されているのは以下の単語である。

- 状態⑤⇒ 状態⑨
  「会議」 (10%) 「こちら」 (8%) 以下「先生」「そちら」

- 状態⑤⇒ 状態⑫
  「会議」「１００」「語」 (接尾辞) 「それ」「私」

- 状態⑦⇒ 状態⓪
  「方 (ほう)」 (31%) 「ありがとう」 (感動詞, 18%) 以下「中」「方 (かた)」

- 状態⑦⇒ 状態①
  「方 (ほう)」 (20%) 「事務」 (9%) 以下「中」「方 (かた)」

- 状態⑦⇒ 状態⑨
  「方 (ほう)」 (15%) 「共」 (接尾辞, 10%) 以下「会議」「方 (かた)」

- 状態⑦⇒ 状態⑩

  「ん」(準体助詞, 67%) 以下「わけ」「の」「方(かた)」「もの」

- 状態⑦⇒ 状態⑫

  「方(ほう)」(19%) 「方(かた)」(10%) 以下「時間」「こと」「もの」

- 状態⑥⇒ 状態①

  「こと」(31%) 「ふう」(14%) 以下 「一」

- 状態⑥⇒ 状態⑩

  「こと」(73%) 以下「ん」(準体助詞) 「もの」「ところ」

- 状態⑥⇒ 状態⑫

  「こと」(38%) 「の」(準体助詞, 29%) 以下「もの」「必要」「ところ」

遷移の起点となる状態によって出力される単語が異なり、状態⑥⑦からの遷移では主として形式名詞が出力され、状態⑤からの遷移では、形式名詞以外の名詞が出力される。形式名詞を出力する遷移でも、普通名詞は出力されるが、8 状態の場合と比べ出力確率の差が大きく、普通名詞と形式名詞の分離が著しい。

状態⑤からの遷移では、出力確率が特に高い単語はなく、多種類の普通名詞の単語がこの遷移で出力され、全体の和が他の品詞よりも高くなっている。これは ergodic HMM が学習によって、単語のカテゴリーを獲得していることを示している。

活用する品詞について

図 3.14 に活用する品詞が出力される遷移を示す。4 状態の ergodic HMM では、活用するものが品詞・活用形に関係なく同じ遷移で出力され、8 状態の ergodic HMM では、活用形が同じ単語が品詞に関わりなく同じ遷移に出力されていた。これに対し、図 3.14 ～図 3.14 を見ると 16 状態ではさらに細かく分類して品詞の異なるものを別々の遷移で出力している。本動詞は状態⑧⑤からの遷移で、補助動詞は状態⓪④からの遷移で、助動詞は状態⓪①⑬からの遷移でそれぞれ出力されている。これは、状態数が増えるにしたがい、単語の分類がより詳細になっていくことを示している。以下では各品詞ごとに出力される単語を示し、考えられる単語連鎖や特徴を述べる。

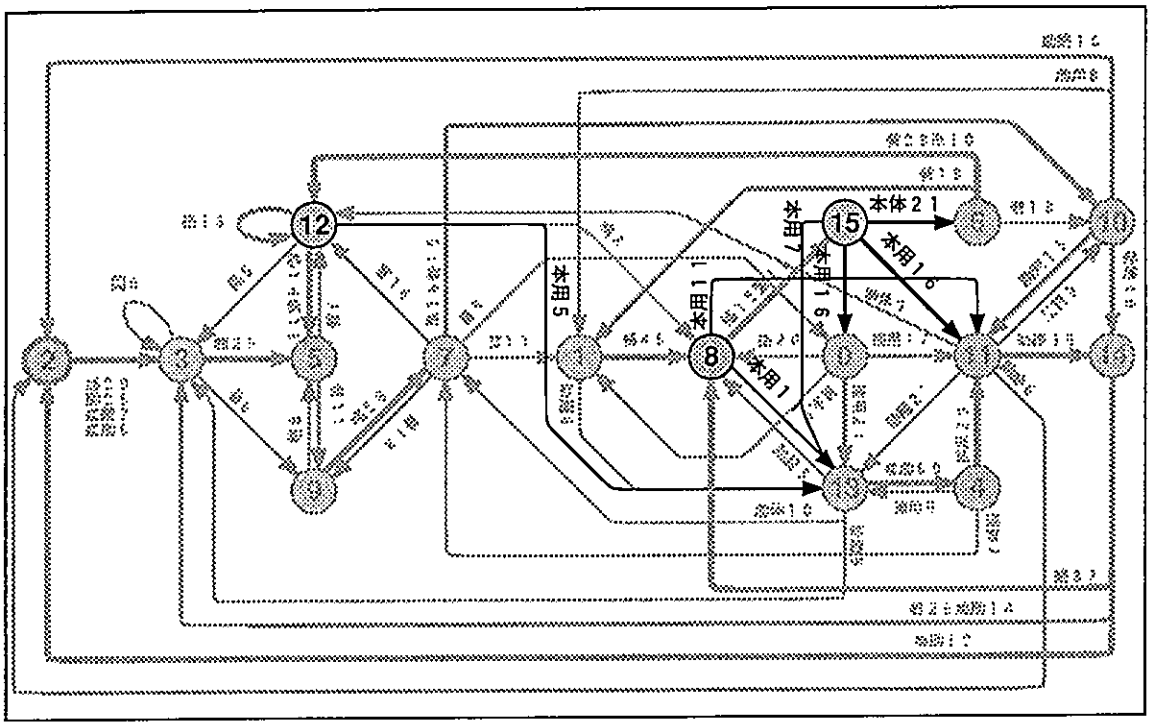


図 3.14: 本動詞の出力

本動詞は、主に⑧,⑮からの遷移で出力される (図 3.14 参照)。複数の遷移で本動詞が出力されているが、各遷移によって出力される単語に違いが見られ、次のような単語が出力されている。

- 状態⑧⇒ 状態①
  「なり」 (21%) 「つき」 (10%) 以下「関し」「致し」

- 状態⑧⇒ 状態⑬
  「なっ」 (21%) 「し」 (14%) 以下「関し」「送っ」「書い」

- 状態⑫⇒ 状態⑬

  「し」(21%) 「持っ」(10%) 以下「でし」「送っ」

- 状態⑮⇒ 状態⓪

  「願い」(28%) 「送り」(13%) 「待ち」(11%) 「伺い」(8%)

- 状態⑮⇒ 状態⑥

  「いう」(94%)

- 状態⑮⇒ 状態⑪

  「思い」(60%) 「申し」(14%) 「し」(12%) 以下「いい」

- 状態⑮⇒ 状態⑬

  「いっ」(55%) 「思っ」(24%) 以下「し」「なっ」「考え」

本動詞は、連用形の単語の出力が他の活用形に比べ非常に高く、言い切り（終止形）や体言への接続よりも、助動詞、補助動詞をともなうことが多いことがわかる。

意志を伝える本動詞「いう」「思う」などが状態⑮から出力され、状態⑧からは「～になる」など格助詞「に」に続く本動詞が多く出力されている。

また、8状態で見られた「お願い(いたし)～」などの接頭辞「お」をともなう謙譲表現が16状態でも見られ、状態⑧⇒⑮の遷移で出力される接頭辞「お」に状態⑮⇒ 状態⓪で出力される本動詞が接続すると考えられる。

図 3.15 に補助動詞の出力される遷移を示す。

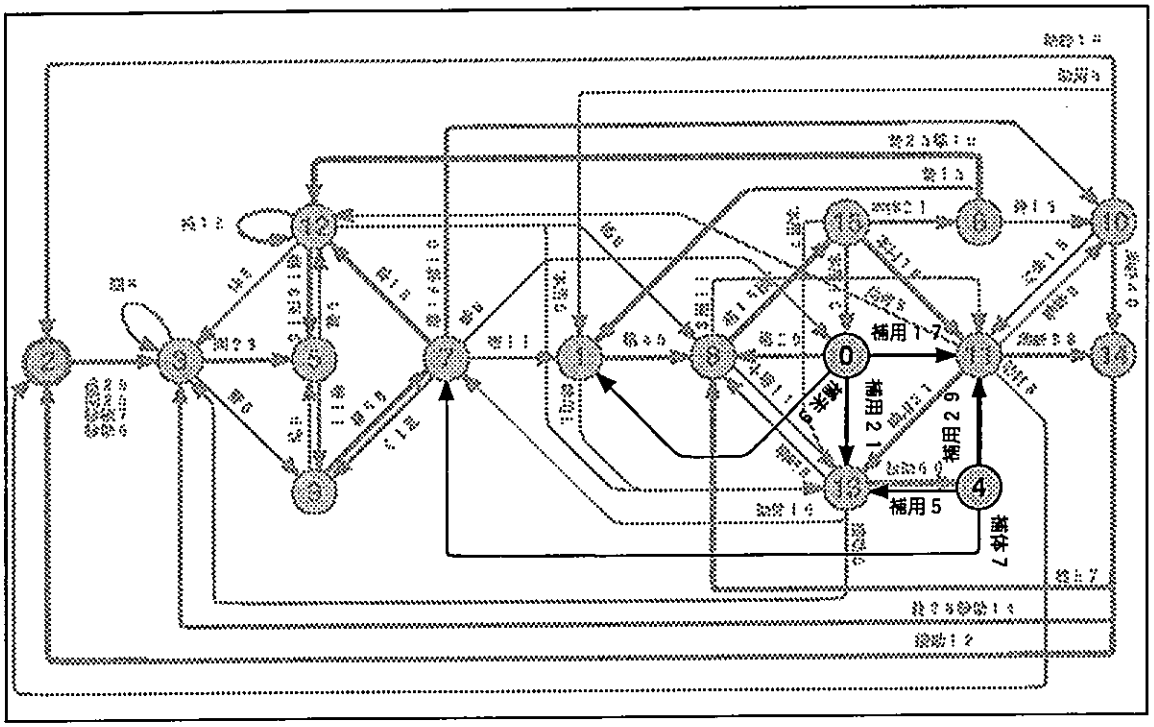


図 3.15: 補助動詞の出力

補助動詞は、⓪,④からの遷移で出力され、出力されている単語は以下の通りである。

- 状態⓪⇒ 状態①
  「さ」(66%) 「ください」(11%) 以下「し」

- 状態⓪⇒ 状態⑪
  「いたし」(44%) 「し」(38%) 以下「でき」「申し上げ」

- 状態⓪⇒ 状態⑬
  「し」(77%) 「ございまし」(19%)

- 状態④⇒ 状態⑦
  「いる」(34%) 「る」(14%) 「いただける」(11%) 以下「ない」

- 状態④⇒ 状態⑪
  「おり」(51%) 「いただき」(28%) 以下「いただけ」

- 状態④⇒ 状態⑬
  「いただい」(62%) 「しまっ」(28%) 以下「き」「行」

本動詞と同様に連用形の出力が多く、助動詞をともなうことが多いのがわかる。状態⓪から出力される単語は表記が異なるものの意味的には「する」と同じものが多い。

図 3.16 に助動詞を主力する遷移を示す。

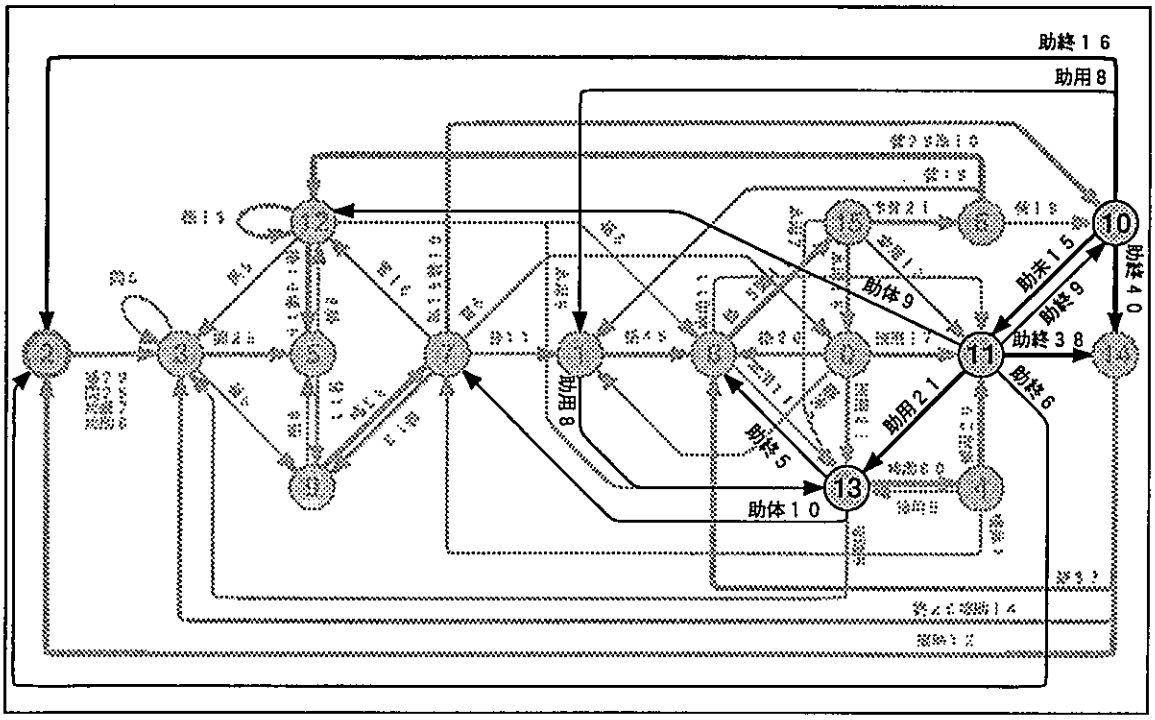


図 3.16: 助動詞の出力

助動詞は、主に⑩,⑪,⑬からの遷移で出力される。各遷移での単語の出力は以下の通りである。

- 状態①⇒ 状態⑬ 「せ」 (36%) 「し」 (23%) 「れ」 (22%)
- 状態⑩⇒ 状態① 「で」 (93%)
- 状態⑩⇒ 状態② 「です」 (97%)
- 状態⑩⇒ 状態⑪ 「でしょ」 (96%)
- 状態⑩⇒ 状態⑭ 「です」 (100%)
- 状態①⇒ 状態② 「ます」 (99%)
- 状態⑪⇒ 状態⑩ 「ます」 (98%)
- 状態⑪⇒ 状態⑫ 「ます」 (97%)

- 状態①⇒ 状態⑬ 「まし」 (99%)
- 状態①⇒ 状態⑭ 「ます」 (58%) 「う」 (41%)
- 状態⑬⇒ 状態⑦ 「た」 (87%) 「たい」 (13%)
- 状態⑬⇒ 状態⑧ 「た」 (67%) 「たい」 (13%)

遷移の起点となる状態によって全く異なる助動詞を出力しており、8状態と同様に「です」「ます」が分離され、さらに16状態では「た」「たい」が他の単語から分離して出力されている (8状態では本動詞「いう」と同じ遷移で出力されていた)。また、8状態の場合には連体形と連用形の「ます」が同一遷移で出力されていたが、上記の出力確率を見てわかる通り、「です」も「ます」も活用形ごとに明確に分離されている。これらのことから、HMMの状態数が増えることによって品詞分類が細分化されることがわかる。

活用する単語の活用形にごとの出力を以下に示す。

図 3.17 に連体形の単語が出力される遷移を示す。

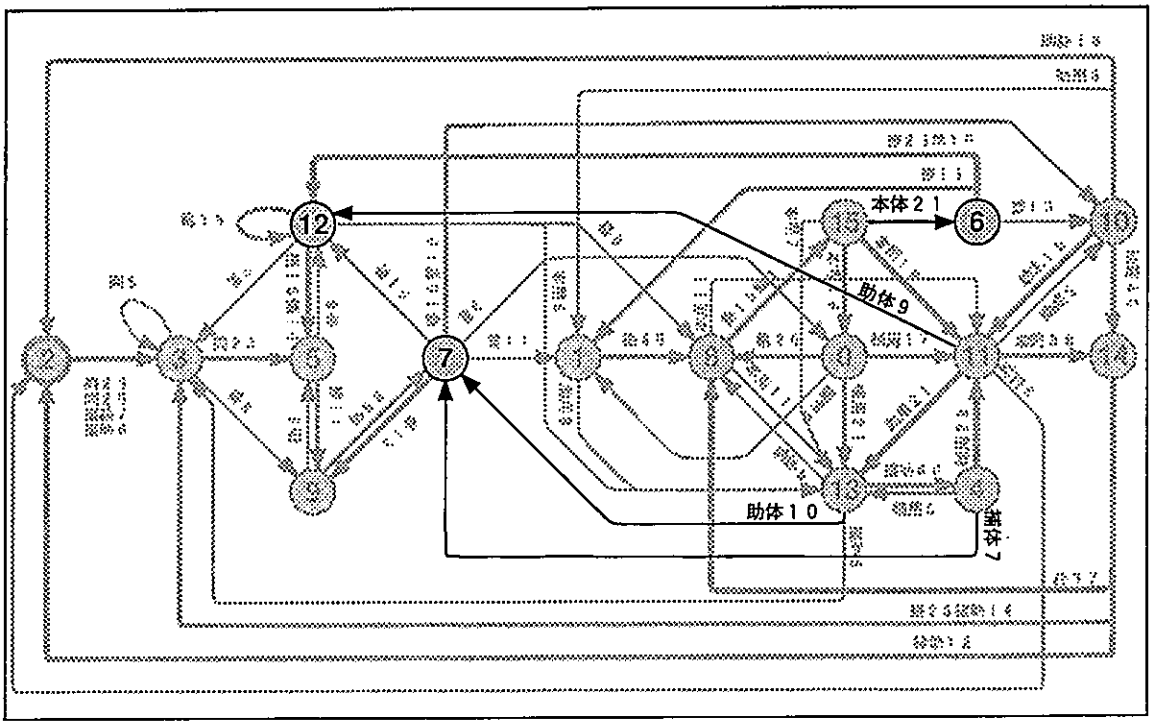


図 3.17: 連体形の出力

8状態の場合と同じく、主として体言を出力する状態⑥⑦への遷移で出力される。助動詞連体形の「ます」は（体言をほとんど出力していない）状態⑫への遷移で出力されてい

る。ネットワーク上には現れないが、状態⑫⇒ 状態②③で接続助詞「ので」が出力されており、これに接続するものと思われる。

連用形が出力される遷移を図 3.18 に示す。

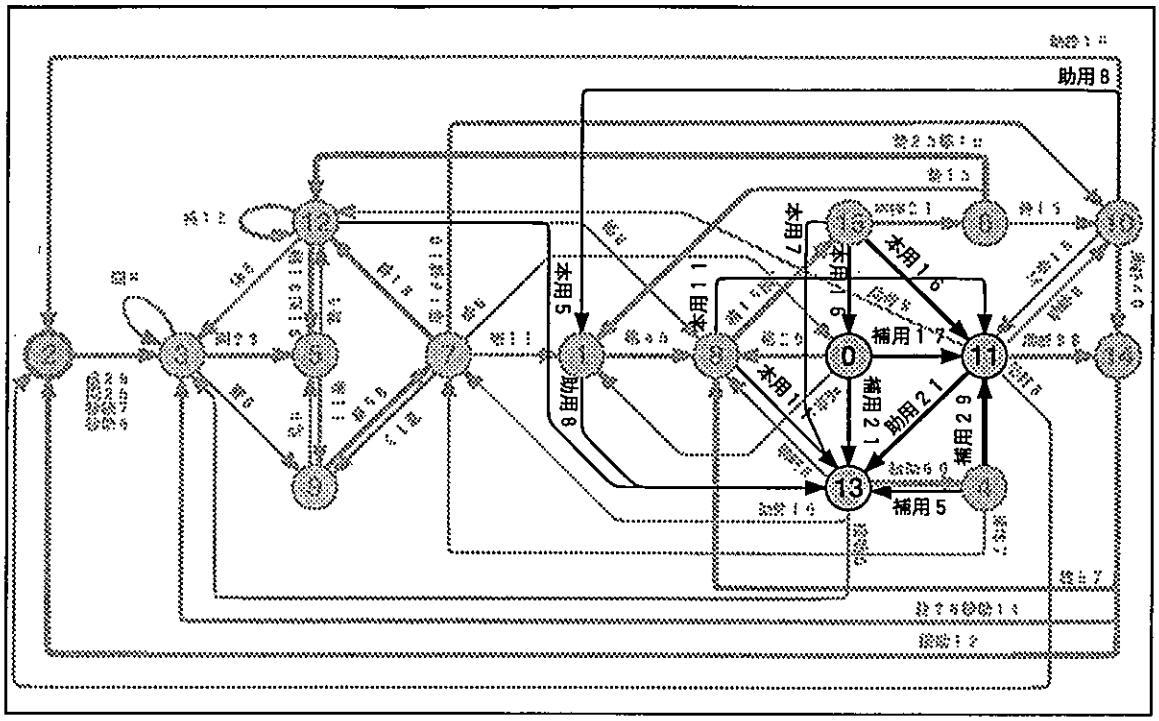


図 3.18: 連用形の出力

図 3.18 から、活用する品詞間の接続の様子がわかる。「(お) 願いいたします」のような「本動詞＋補助動詞＋助動詞」は状態⑮⇒⓪で本動詞を出力し、さらに状態①⑬の助動詞を出力する状態へと遷移すると考えられる。また補助動詞をともなわない「本動詞＋助動詞」のような場合は、状態⑧⑮ ⇒ 状態①⑬のように遷移すると思われる。

また、8 状態で見られた、「〜いただいて」「思って」などの「連用形＋接続助詞 て」の連鎖が、状態④⇒ 状態⑬⇒ 状態④で見られる。

終止形を出力する遷移を図 3.19 に示す。

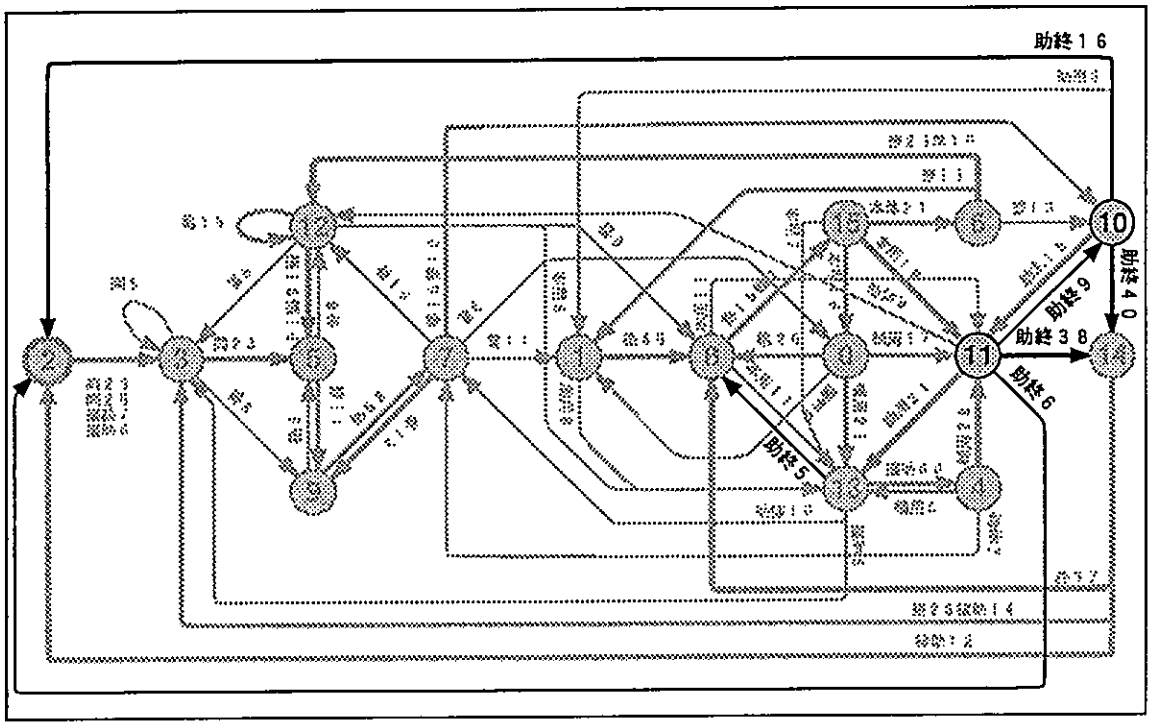


図 3.19: 終止形の出力

８状態と同じく終止形は助動詞がほとんどであり、終助詞や接続助詞、間投詞を出力し文を終了あるいは文節頭に接続する状態⑩⑭②への遷移で出力される。

格助詞について

格助詞の出力は、8状態ergodic HMMと同様に、主に体言を出力した遷移の集まる状態⑧⑨⑫からの遷移で出力される。

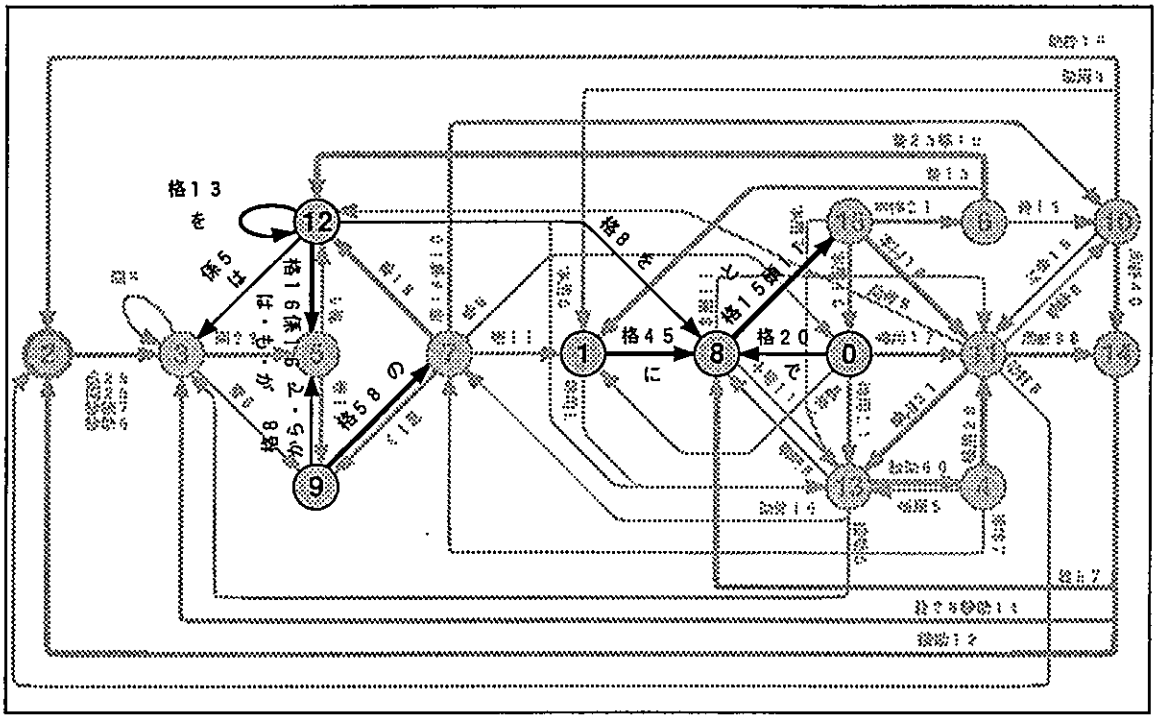


図 3.20: 格助詞の出力

- 状態⓪⇒ 状態⑧ 「で」 (86%)
- 状態①⇒ 状態⑧ 「に」 (87%)
- 状態⑧⇒ 状態⑮ 「と」 (63%) 「お」 (接頭辞,28%)
- 状態⑨⇒ 状態⑤ 「から」 (57%) 「で」 (23%) 以下「が」「として」
- 状態⑨⇒ 状態⑦ 「の」 (93%)
- 状態⑫⇒ 状態③ 「は」 (46%) 「ので」 (接続助詞,35%) 以下「を」「と」
- 状態⑫⇒ 状態⑤ 「が」 (55%) 「は」 (24%) 以下「は」「も」
- 状態⑫⇒ 状態⑧ 「を」 (75%) 以下「について」
- 状態⑫⇒ 状態⑫ 「を」 (74%) 「について」 (24%)

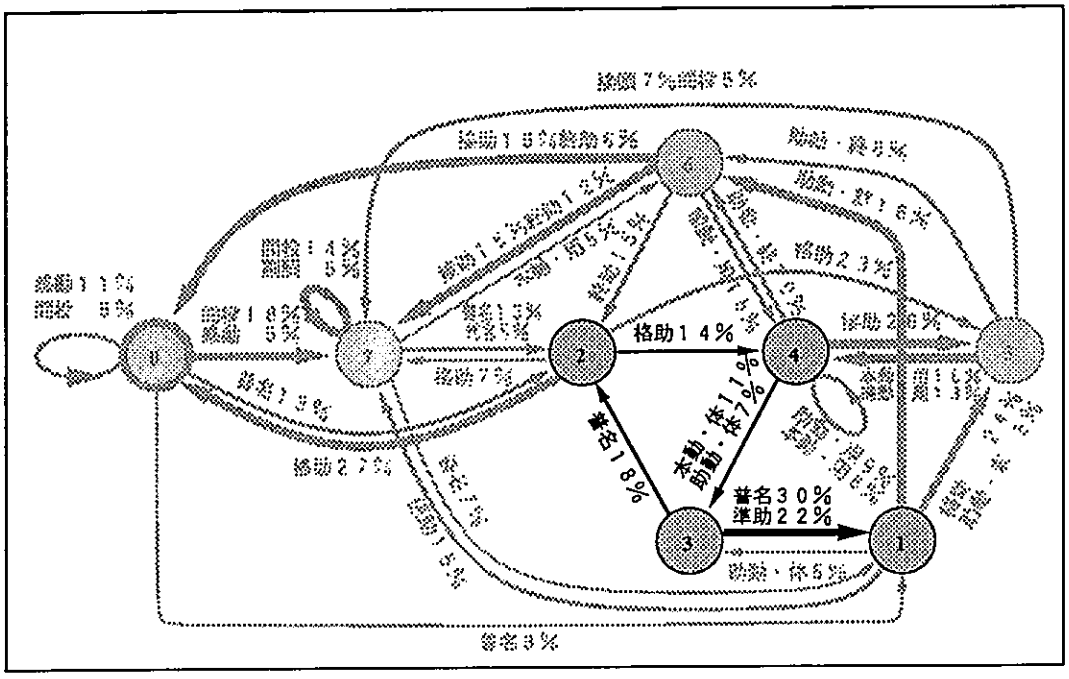
8状態では同じ遷移で出力されていた「で」と「に」、「が」と「を」が別々の遷移に分かれて出力され、ここでも、状態数が増えることによって単語が細かく分類されていることがわかる。

3.3.6 文を表現する遷移

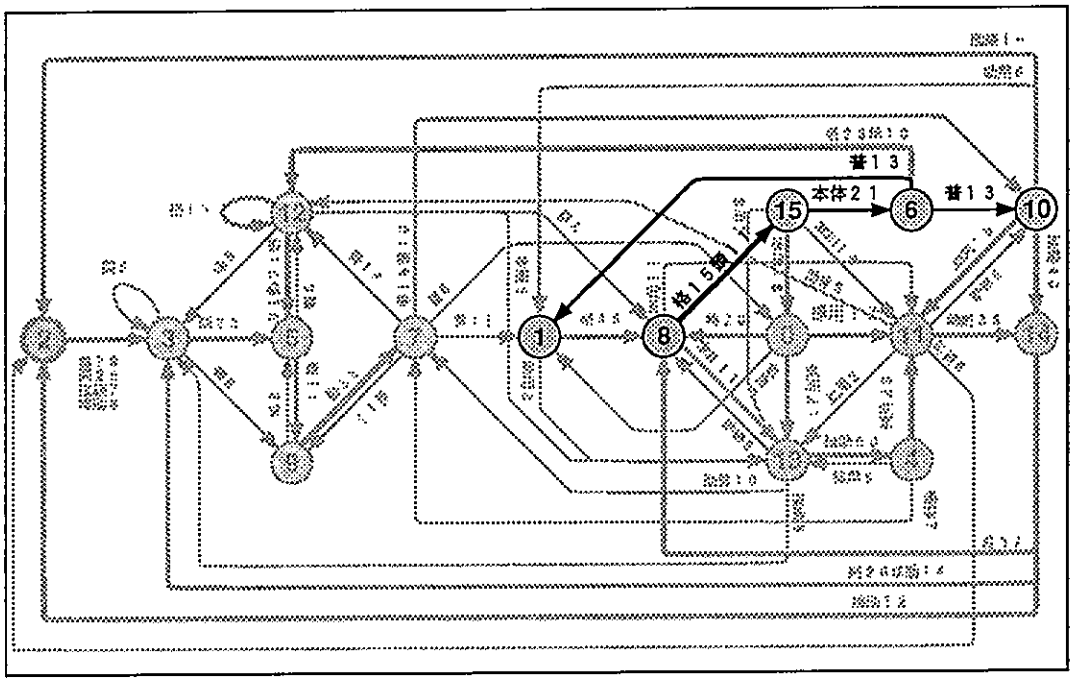
8 状態および、16 状態の ergodic HMM の解析結果で、文を表現すると考えられる遷移について述べた。これらを確かめるため、Ergodic HMM に実際に文を入力した時に、文を生成する確率が最大になる遷移を Viterbi search によって調べた。学習データの文でよく用いられる表現である「〜ということ」「〜と思います」「〜おります」「〜したいんです」「お願いいたします」「〜でしょうか」「〜いただいて」「〜して」「「そうですか」「お送り〜」について遷移系列を調べた結果を以下に示す。この結果は、解析結果で推測した遷移と同じものであった。

「～ということ」を表す遷移

後続の単語によって最後の遷移が変わり「～ということ(に)」は②⇒④⇒③⇒② で、「～ということ(です)」は②⇒④⇒③⇒①で出力される。

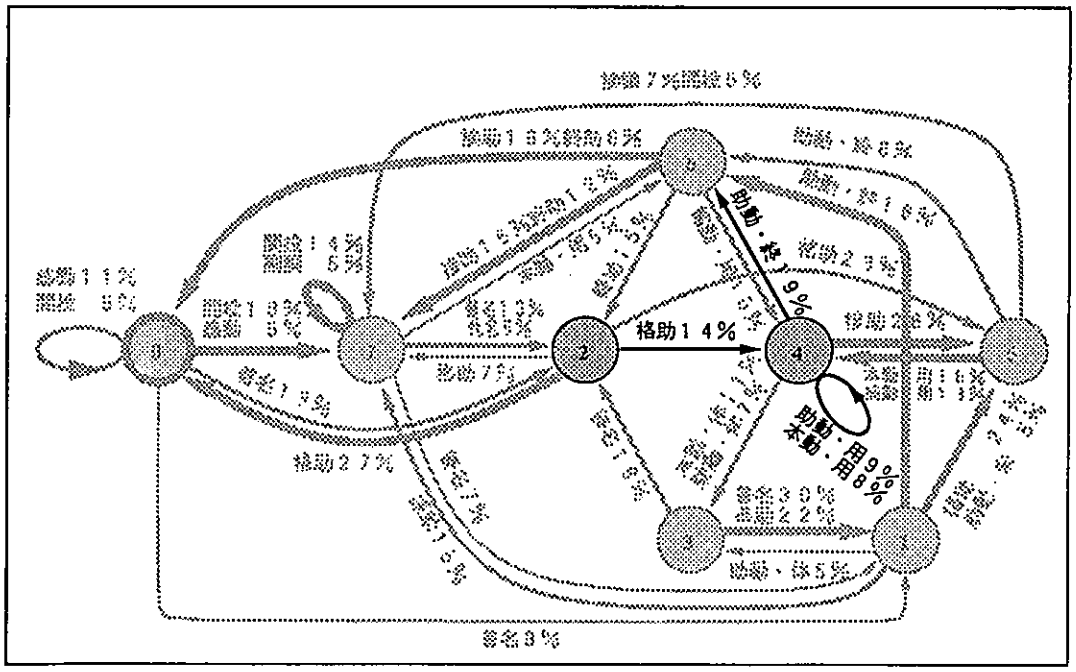


「～ということ(に)」：⑧⇒⑮⇒⑥⇒①、「～ということ(です)」：⑧⇒⑮⇒⑥⇒⑩

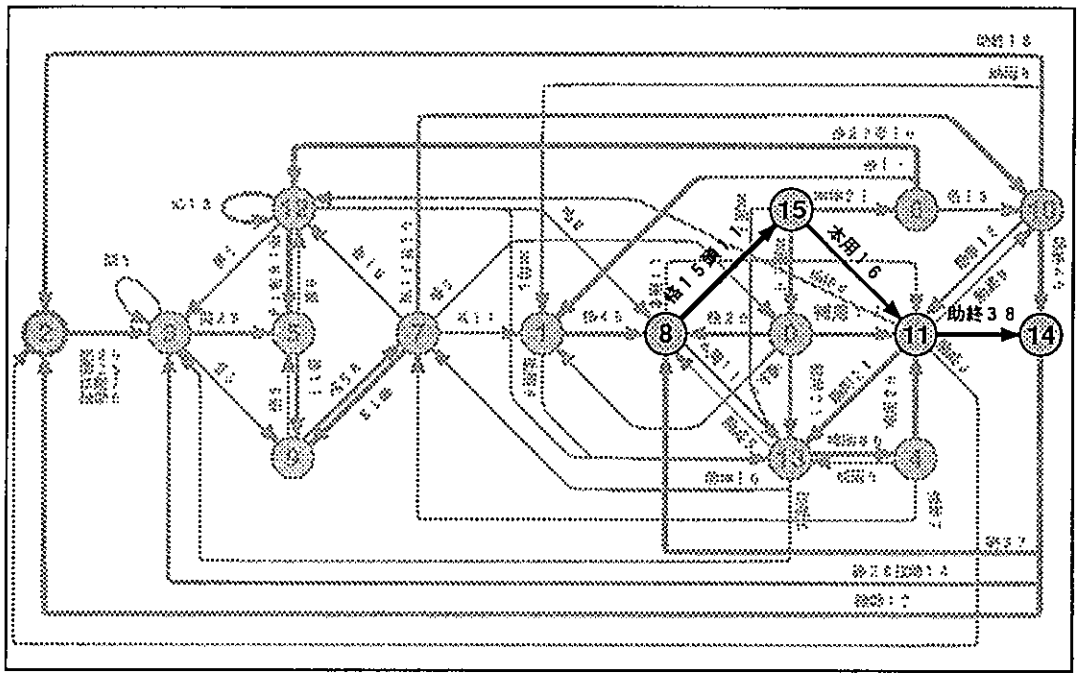


「～と思います」を表す遷移

状態遷移 ：②⇒④⇒④⇒⑥

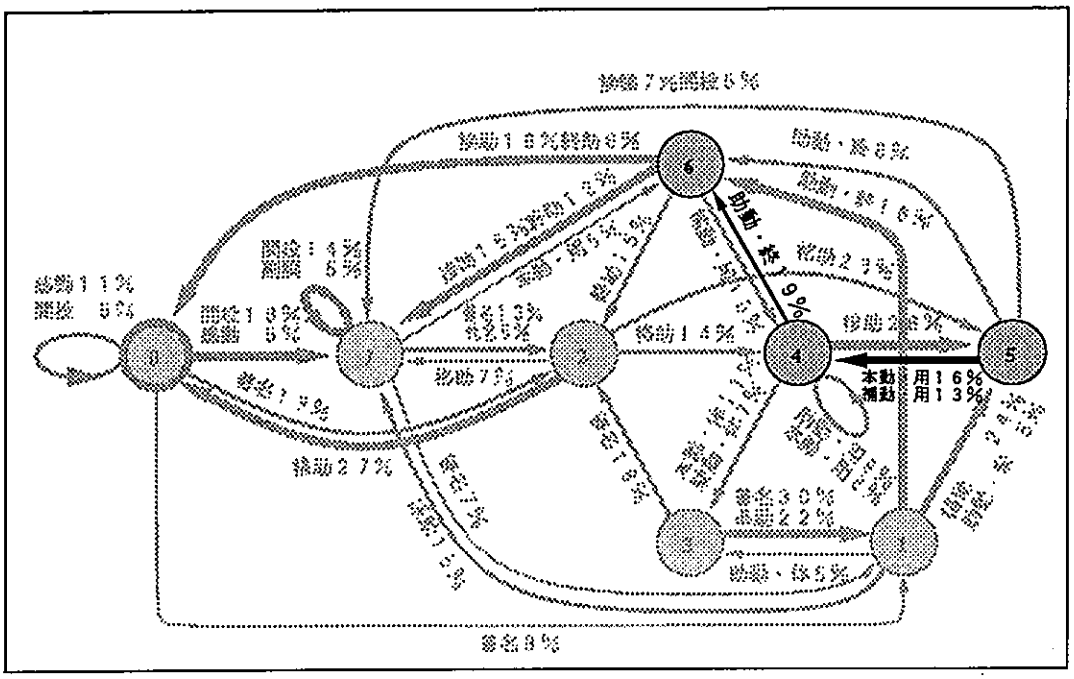


状態遷移 ：⑧⇒⑮⇒⑪⇒⑭

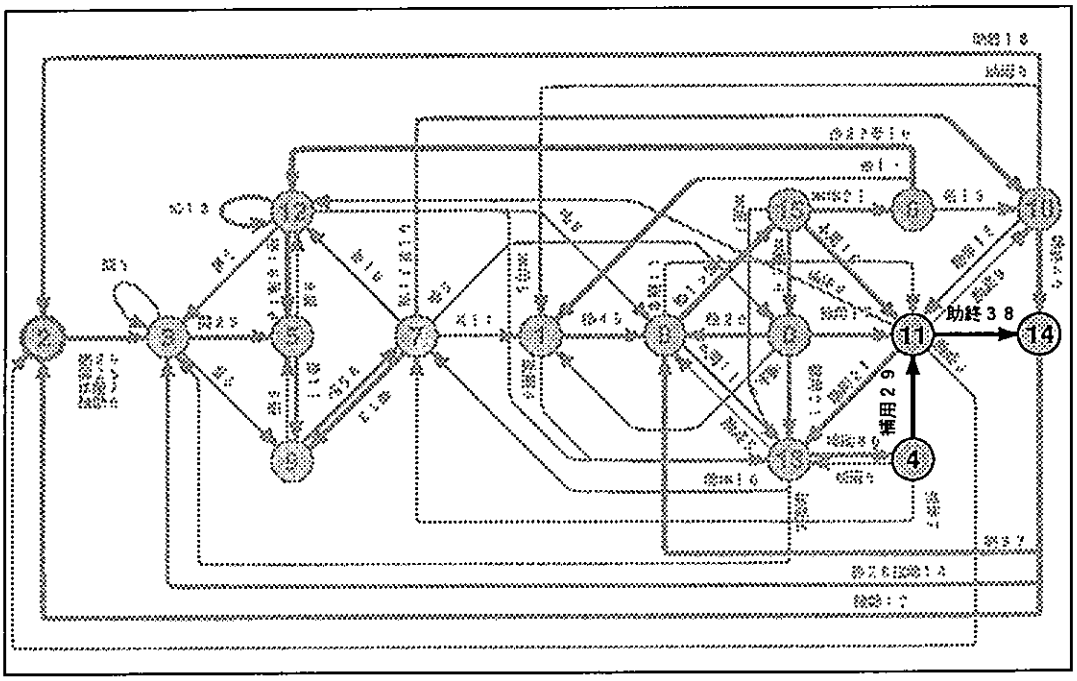


「～おります」を表す遷移

状態遷移 ：⑤⇒④⇒⑥

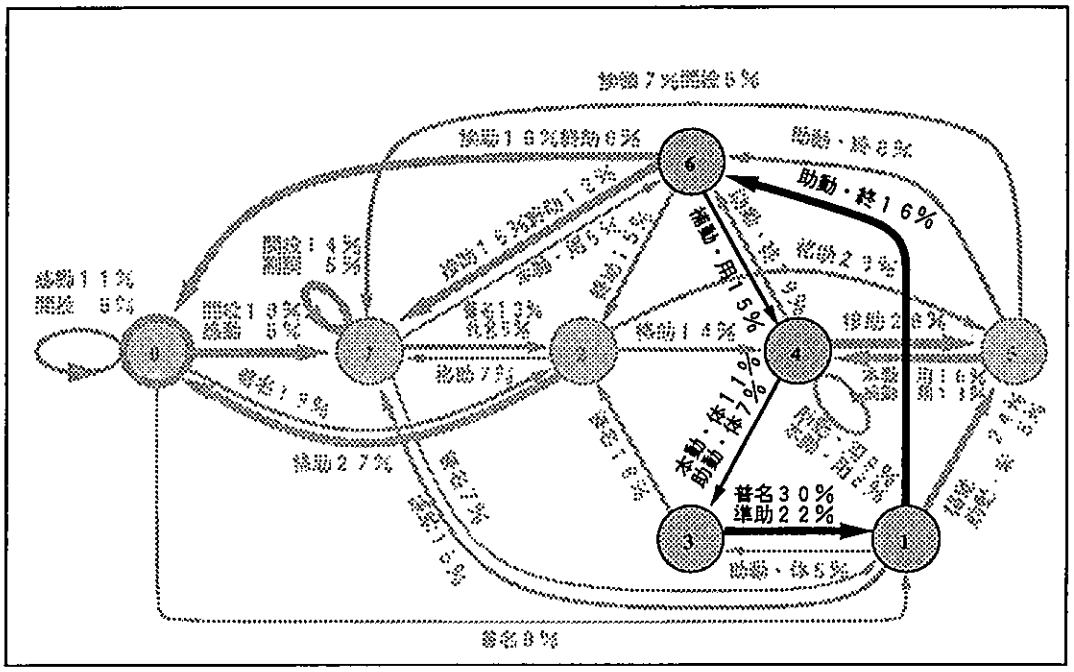


状態遷移 ：④⇒⑪⇒⑭

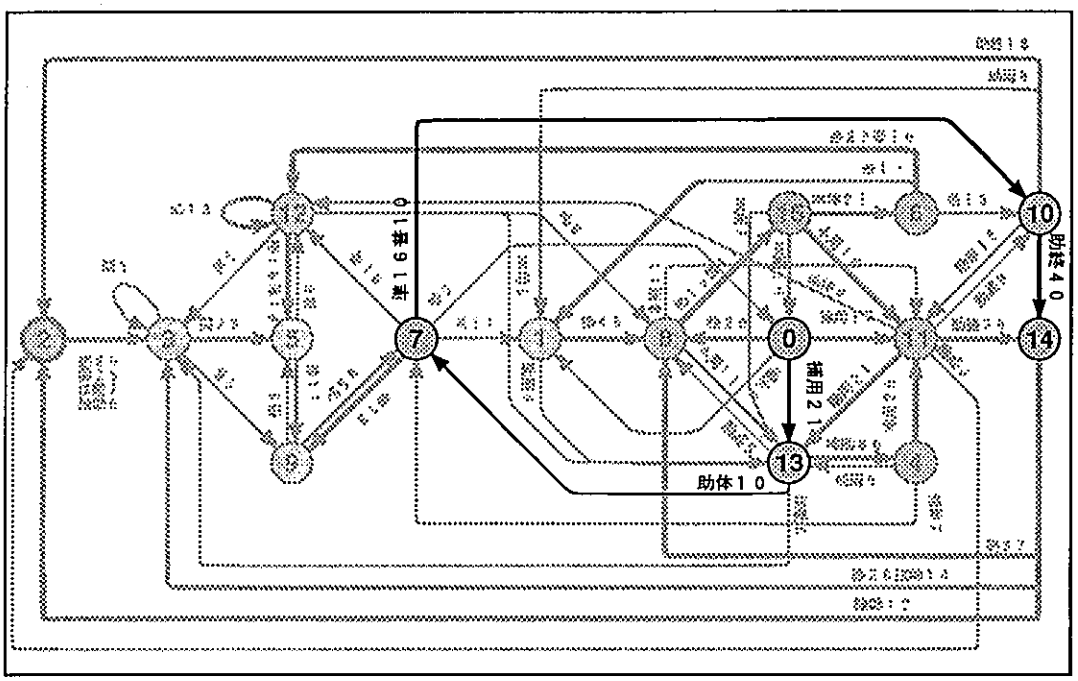


「～したいんです」を表す遷移

状態遷移 ：⑥⇒④⇒③⇒①⇒⑥

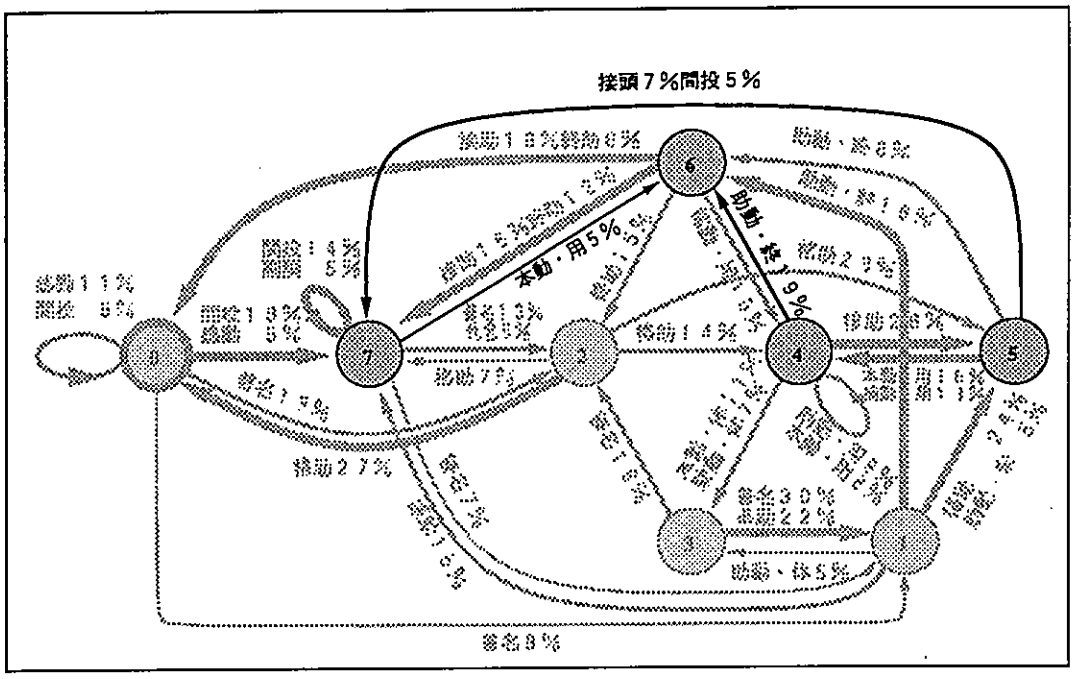


状態遷移 ：⓪⇒⑬⇒⑦⇒⑩⇒④

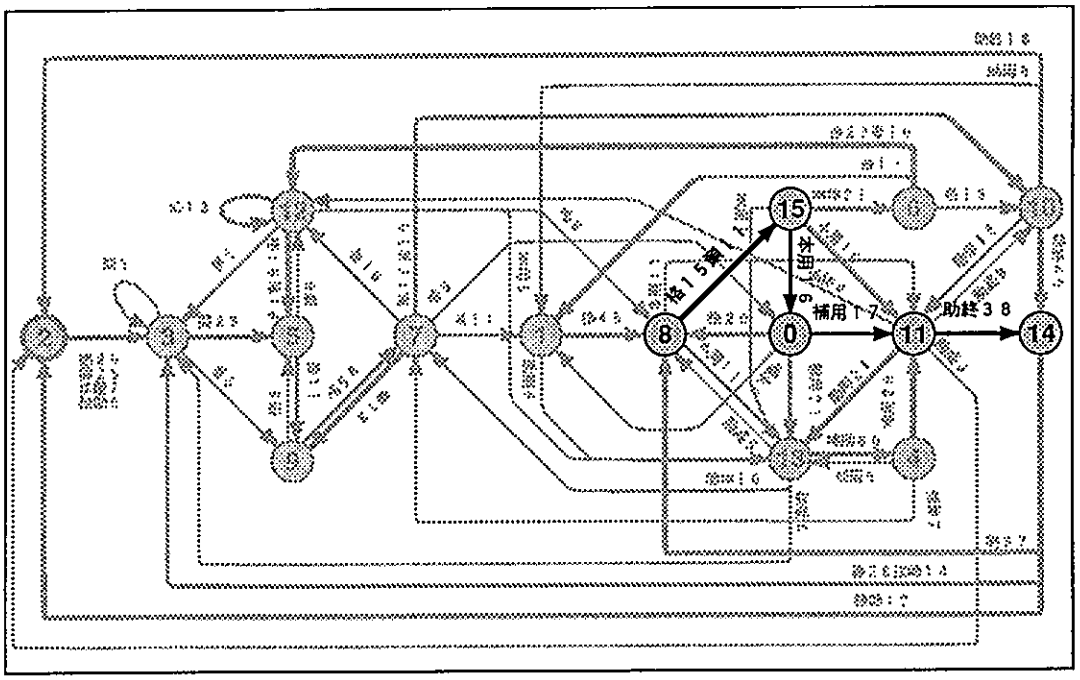


「お願いいたします」を表す遷移

状態遷移 ：⑤⇒⑦⇒⑥⇒④⇒⑥

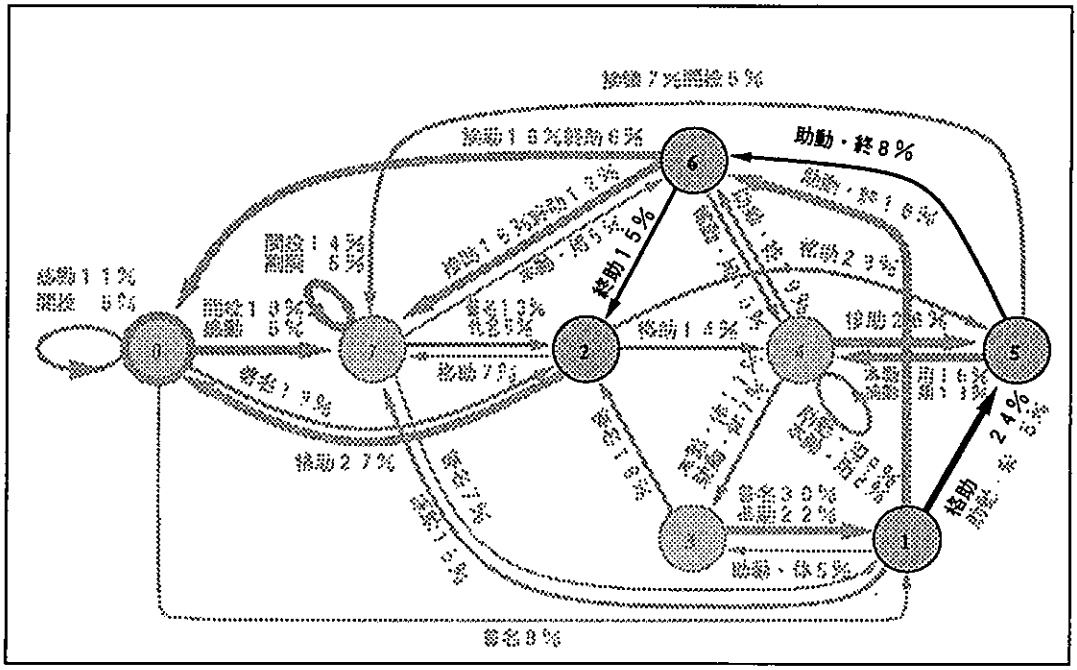


状態遷移 ：⑧⇒⑮⇒⓪⇒⑪⇒⑭

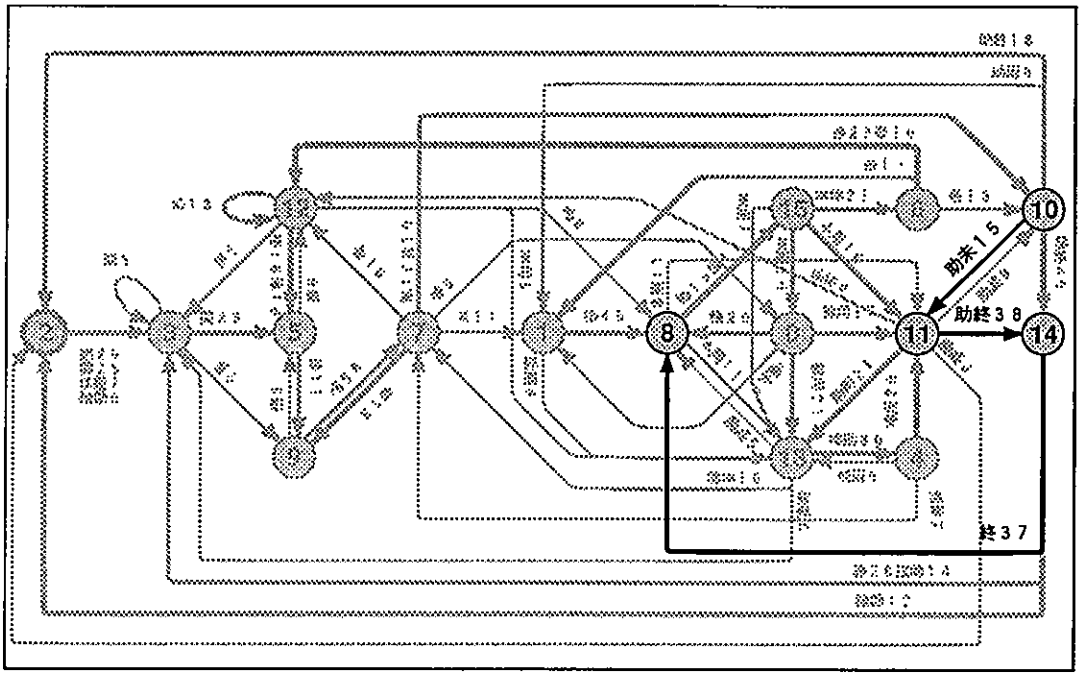


「～でしょうか」を表す遷移

状態遷移 ：①⇒⑤⇒⑥⇒②

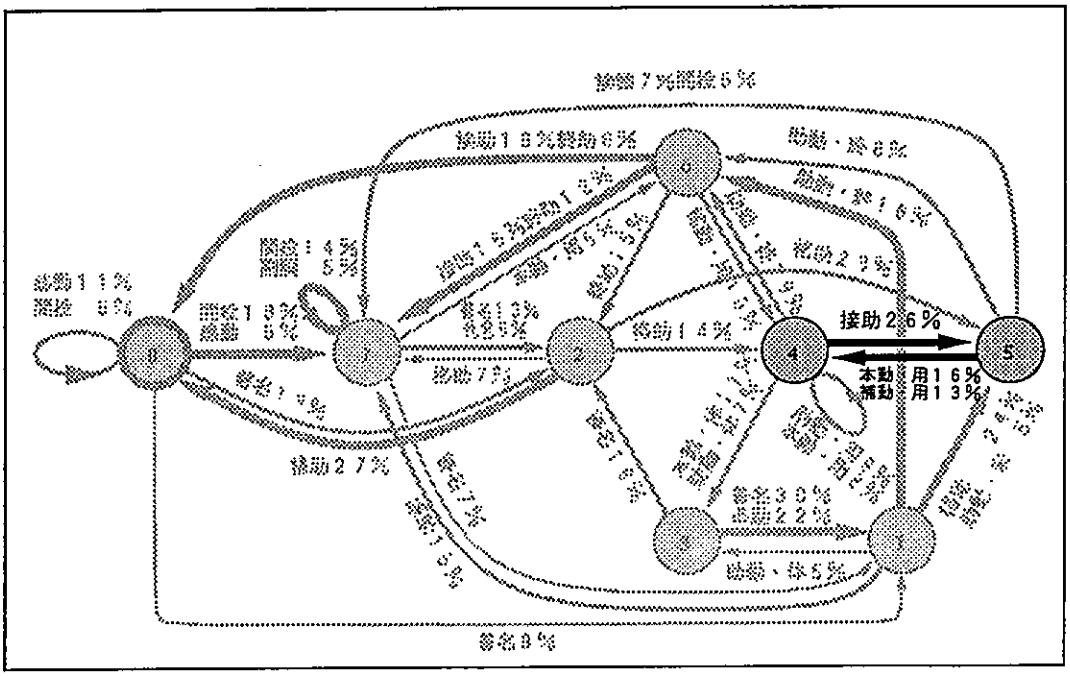


状態遷移 ：⑩⇒⑪⇒⑭⇒⑧

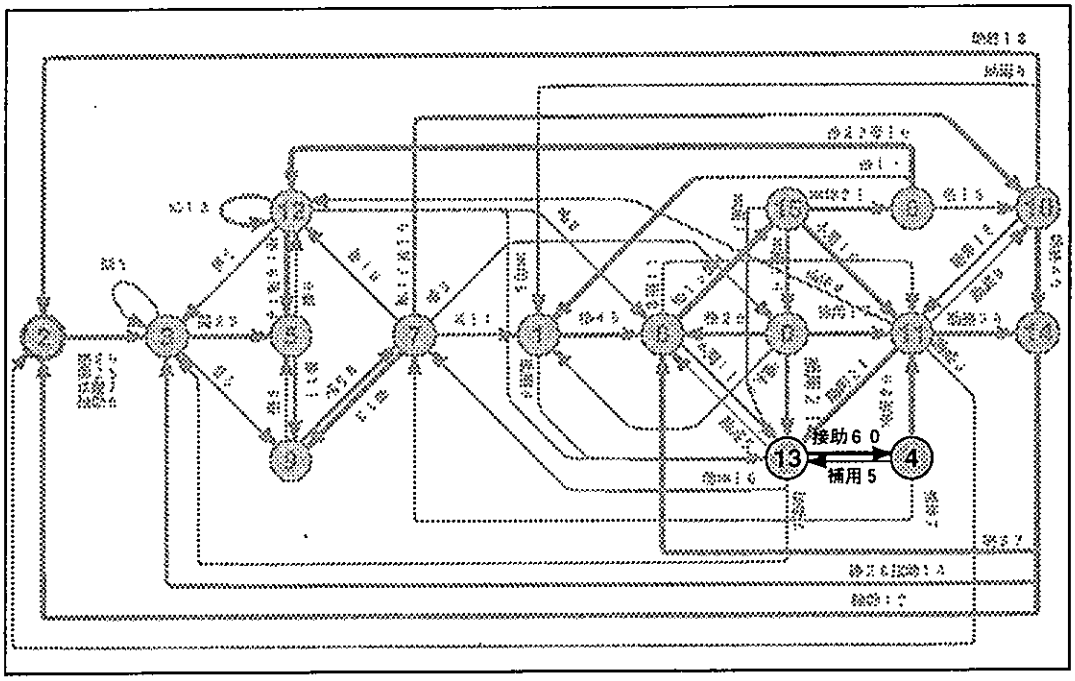


「～いただいて」を表す遷移

状態遷移 ：⑤⇒④⇒⑤

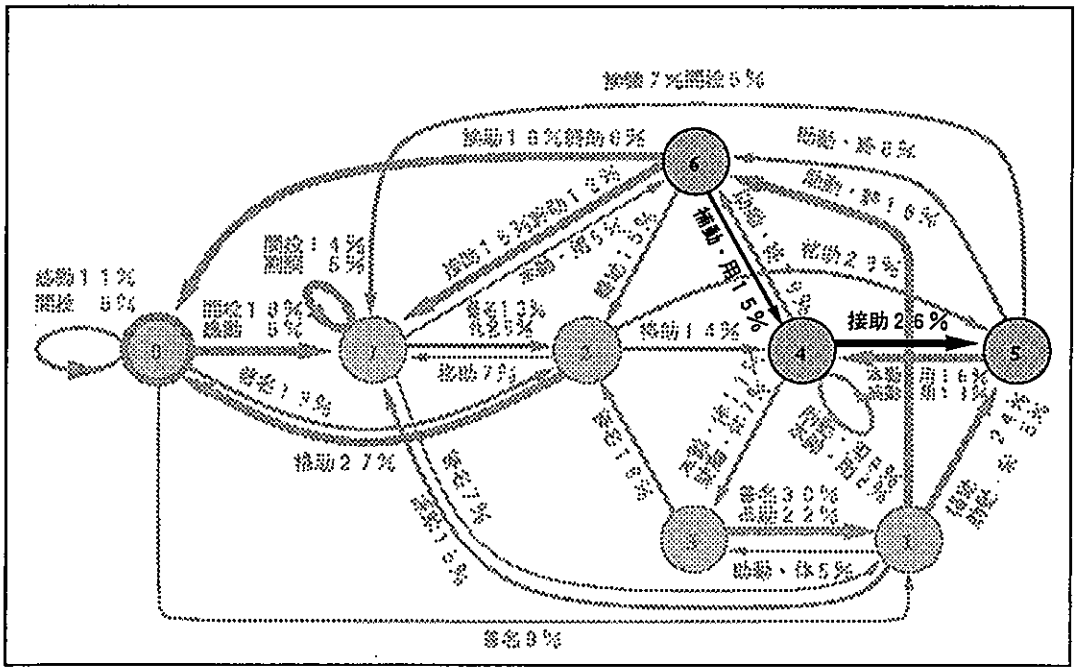


状態遷移 ：④⇒⑬⇒④

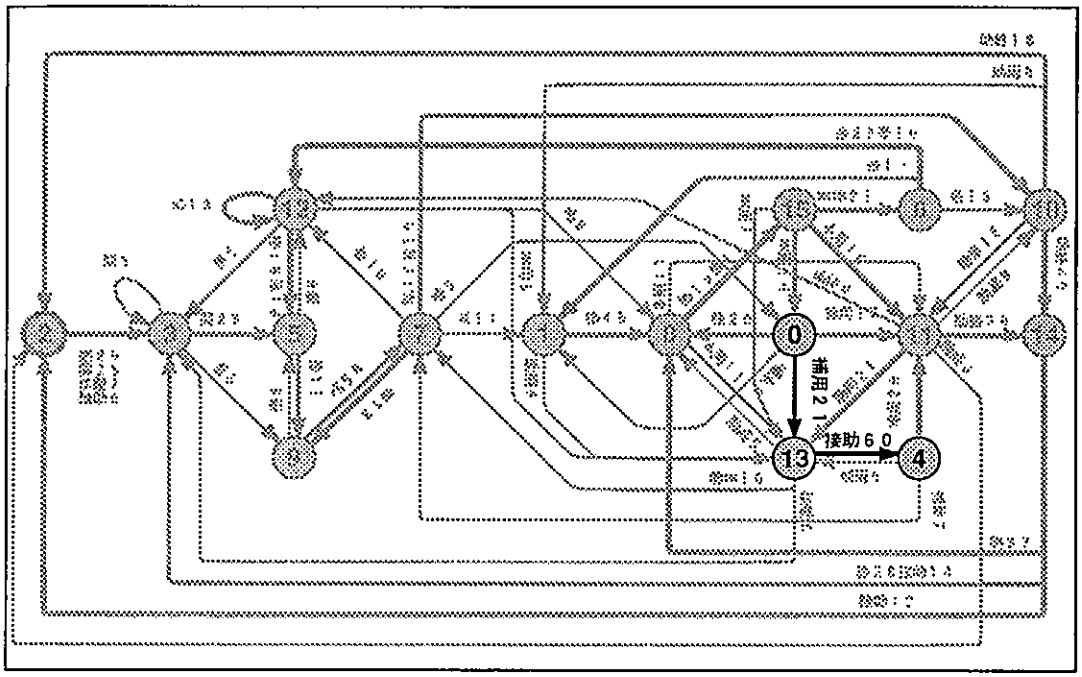


「～して」を表す遷移

状態遷移 ：⑥⇒④⇒⑤

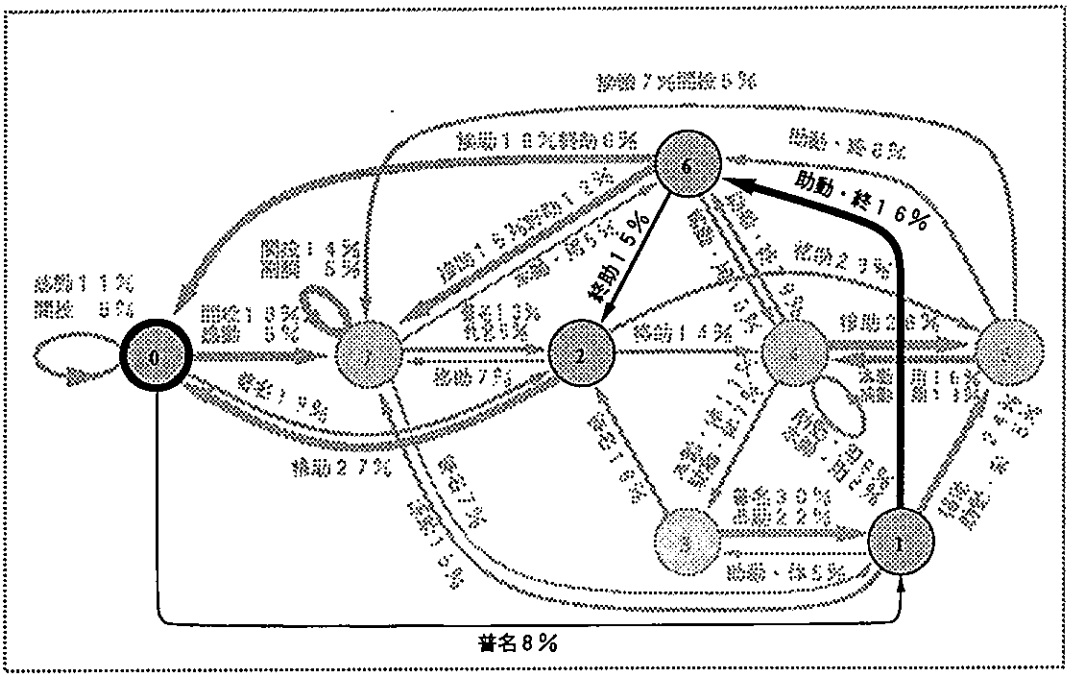


状態遷移 ：⓪⇒⑬⇒④

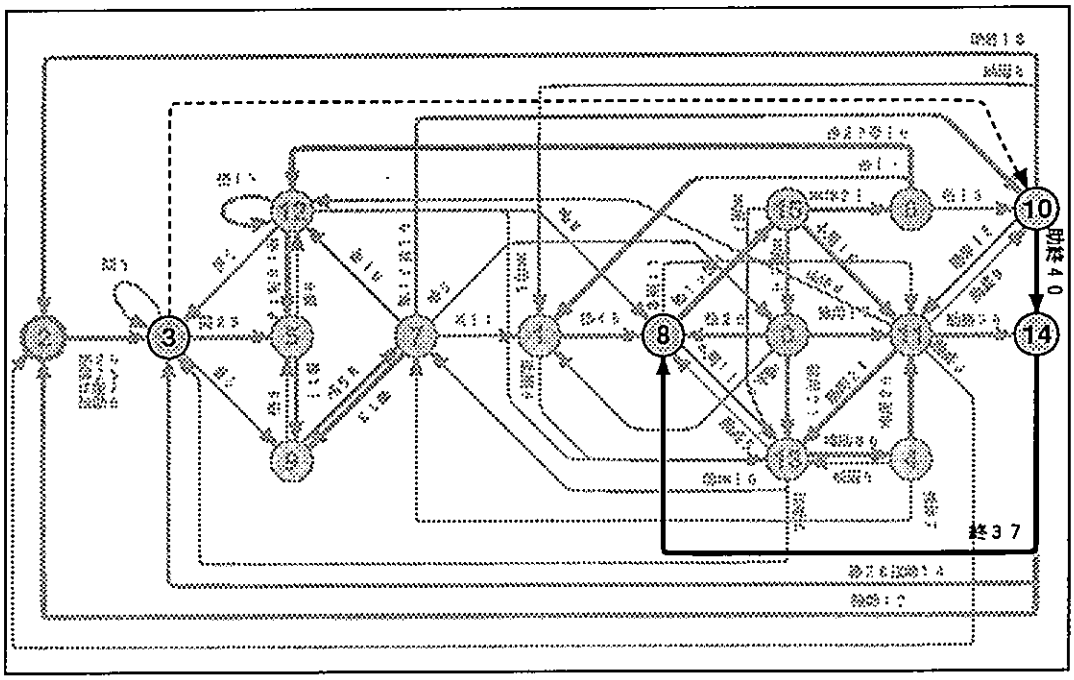


「そうですか」を表す遷移

状態遷移 ：⓪⇒①⇒⑥⇒②

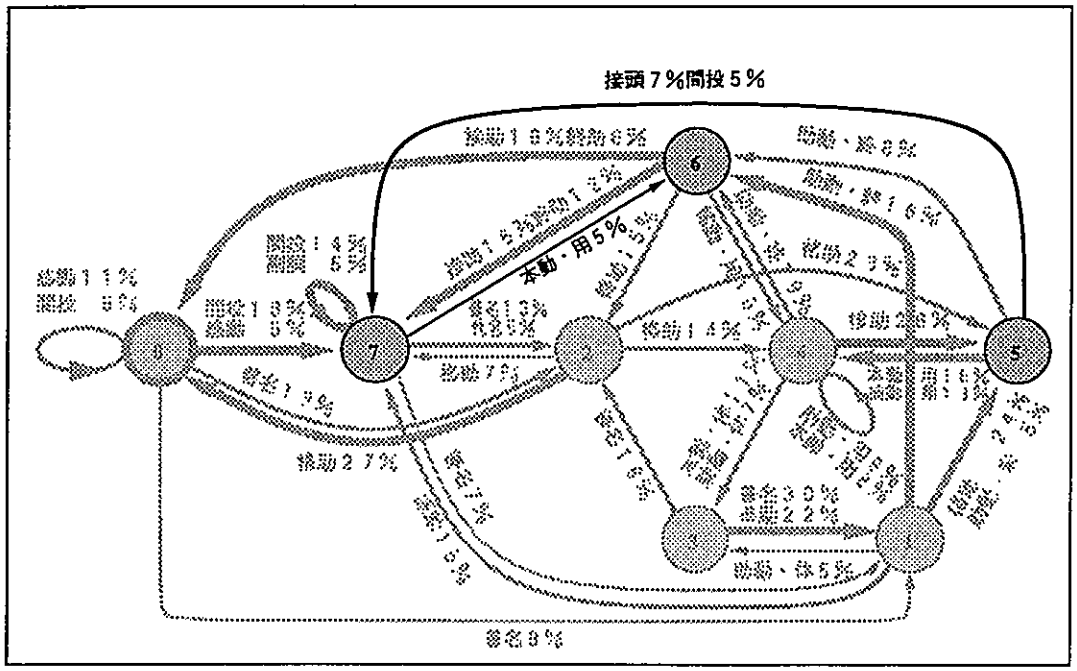


状態遷移 ：③⇒⑩⇒⑭⇒⑧

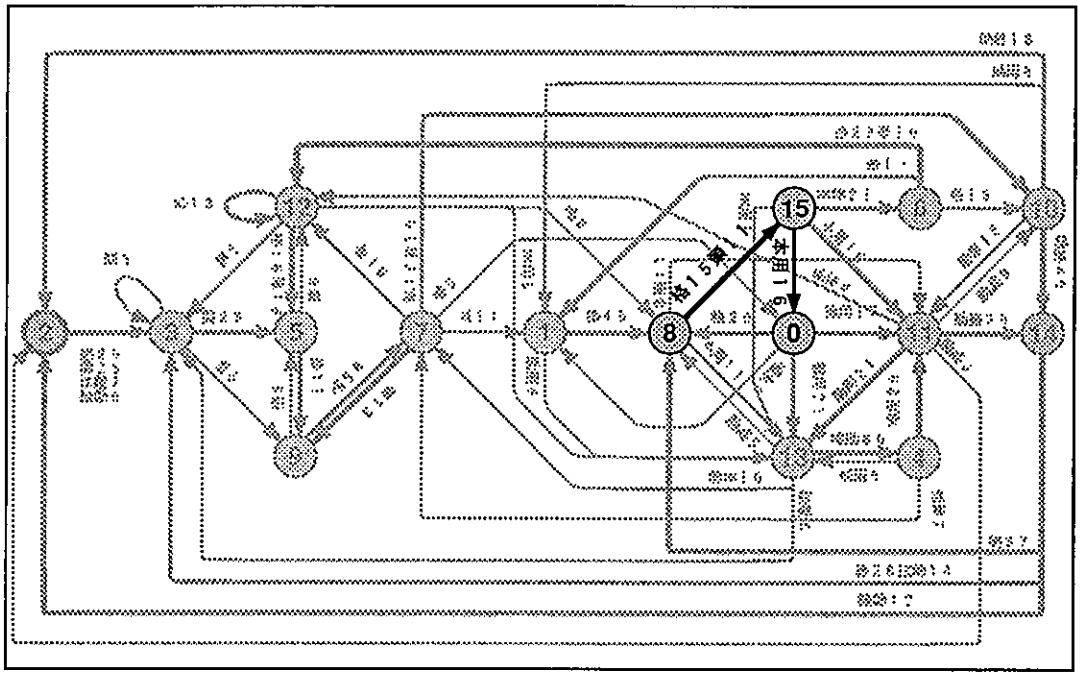


「お送り～」「お待ち～」を表す遷移

状態遷移 ：⑤⇒⑦⇒⑥



状態遷移 ：⑧⇒⑮⇒⓪



3.3.7 文の生成確率・モデルのエントロピー

平均対数文生成確率

学習終了後の ergodic HMM に文（単語列）を入力し、HMM が入力された文を出力する確率を求めたものが文生成確率である。HMM の学習に用いた Baum-Welch algorithm は、尤度（入力データの生成確率）を最大にするようにパラメータを調整する。そこで、学習していないデータ (open data) を生成する確率と学習データを生成する確率と比較することにより、ergodic HMM が獲得した文法の一般性を調べる。

平均対数文生成確率の計算方法

言語モデル生成実験で得られた ergodic HMM が closed data および open data を生成する一文あたりの平均対数確率を、forward algorithm を用いて計算した。なお、単語の出力確率が 0.0 の場合は 10^{-5} でフロアリングした。計算方法を簡単な例で図 3.21 に示す。

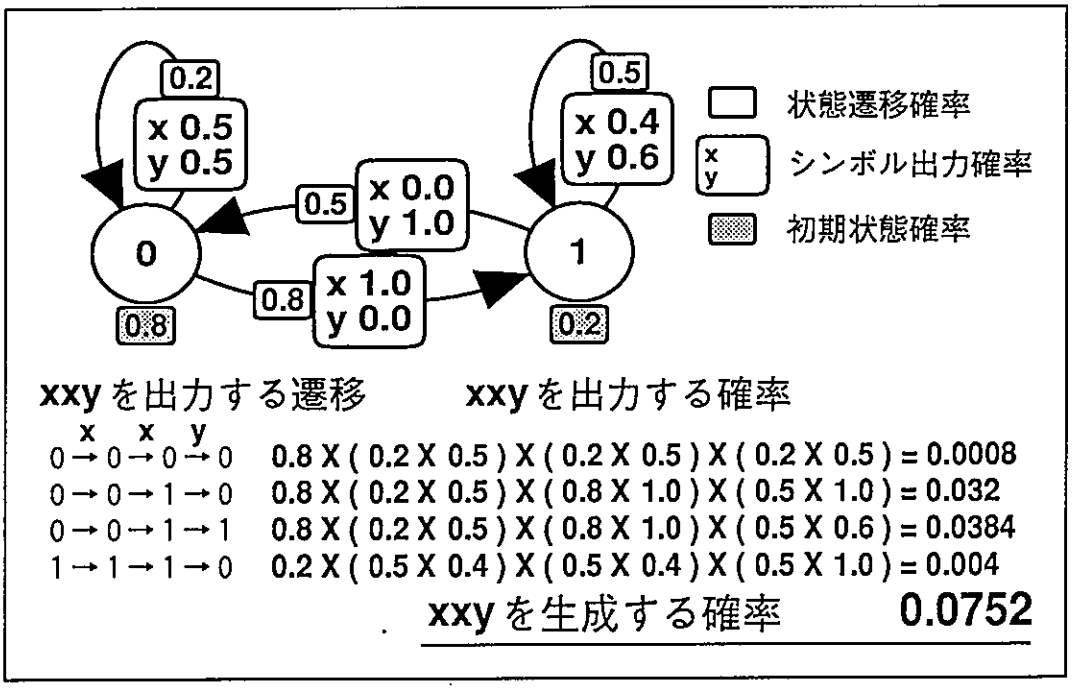


図 3.21: 文の生成確率の計算方法

例えば、出力シンボルが x と y の 2 状態 ergodic HMM の学習終了後の各パラメータが図 3.21 のようになってるとする。この HMM が “xxy” を生成する確率について以下に述べる。

- HMM が “xxy” を出力する全ての経路を列挙する (例では 4 経路)。
- 各経路を通る確率を求める。
- 全ての経路の確率の和を求める (この和が “xxy” を生成する確率)。

以上の手順で求められた生成確率の対数値の和を求め、一文当たりの平均を求めたものを平均対数文生成確率とする。

エントロピー

平均対数文生成確率に加え言語モデルの評価基準としてエントロピーを用いる。エントロピーはモデルの複雑さを表す指標となる。あるモデル λ のエントロピーが $H(\lambda)$ ならば次のシンボルを決定するのに、平均 $H(\lambda)$ 回の yes/no の質問を繰り返さなければならない。いい換えれば、$2^{H(\lambda)}$ 個の等出現確率のシンボルの中から一つのシンボルを決定することになる。すなわち、エントロピーが大きいほど、モデルは複雑であるといえる。

エントロピーの計算方法

$p(v_k \mid i)$ を状態 i から遷移する際に記号 v_k を出力する確率とすれば、HMM が獲得したモデル λ のエントロピーは次のようにして求められる [7]。

$$p(v_k \mid i) = \sum_{j=1}^{N} a_{ij} b_{ij}(v_k) \ldots \text{状態 } i \text{ でシンボル } v_k \text{ を生成する確率} \tag{3.1}$$

$$H(K \mid i) = -\sum_{k=1}^{K} p(v_k \mid i) \log_2 p(v_k \mid i) \ldots 1 \text{ シンボル当たりのエントロピー} \tag{3.2}$$

$$H(\lambda) = \sum_{i=1}^{N} \pi_i H(K \mid i) \ldots \text{モデル } \lambda \text{ のエントロピー} \tag{3.3}$$

ただし、

N	.....	HMM の状態数
K	.....	シンボルの数（種類）
a_{ij}	.....	状態 i から状態 j へ遷移する確率
$b_{ij}(v_k)$	.....	状態 i から状態 j へ遷移する際に v_k を出力する確率
ω_i	.....	状態 i の定常状態確率

である。定常状態確率 ω_i については以下に述べる [10]。Ergodic HMM において、状態 S_i から遷移を開始し、n 回の遷移を繰り返した後に状態 S_j に達する確率 (n 次の遷移確率) を $a_{ij}^{(n)}$ と表すことにする。$a_{ij}^{(n)}$ には次の式が成立する。

$$a_{ij}^{(n+1)} = \sum_{\nu=1}^{N} a_{i\nu} a_{\nu j}^{(n)} \tag{3.4}$$

$$a_{ij}^{(n+m)} = \sum_{\nu=1}^{N} a_{i\nu}^{(n)} a_{\nu j}^{(m)} \tag{3.5}$$

n が大きくなるにつれて、$a_{ij}^{(n)}$ は一定値に近づき、その値は状態 S_j のみで決まり、出発点 S_i には無関係になることが証明できる [10]。すなわち、

$$\lim_{n\to\infty} a_{ij}^{(n)} = \omega_j \tag{3.6}$$

となる。ω_j は、十分な遷移のあとにおいて、任意の瞬間に、この過程が状態 S_j にある確率を表す。定常状態確率 ω_j には次の式が成り立つ。

$$\sum_{j=1}^{N} \omega_i = 1 \tag{3.7}$$

$$\sum_{i=1}^{N} \omega_i a_{ij} = \omega_j \tag{3.8}$$

したがって、定常状態確率はこの式を解けば、遷移確率から求められる。

計算結果

odd4000 を学習させた各状態数の ergodic HMM について open data、closed data それぞれ 4000 文の対数生成確率を求め、一文当たりの平均と HMM のエントロピーを計算した結果を表 3.23 , 図 3.22 , 図 3.23 に示す。

表 3.23: 平均対数文生成確率・エントロピー

ergodic HMM の状態数	エントロピー	平均対数文生成確率	
		closed data	open data
2 状態	7.53	-76.48	-77.37
4 状態	6.70	-69.34	-71.30
8 状態	5.99	-62.93	-67.48
16 状態	5.29	-56.81	-64.50

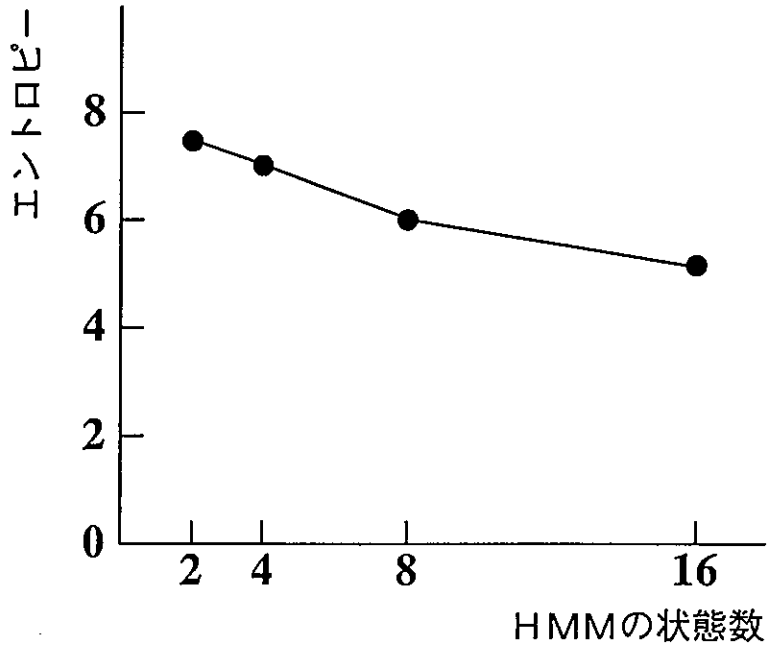


図 3.22: モデルのエントロピー

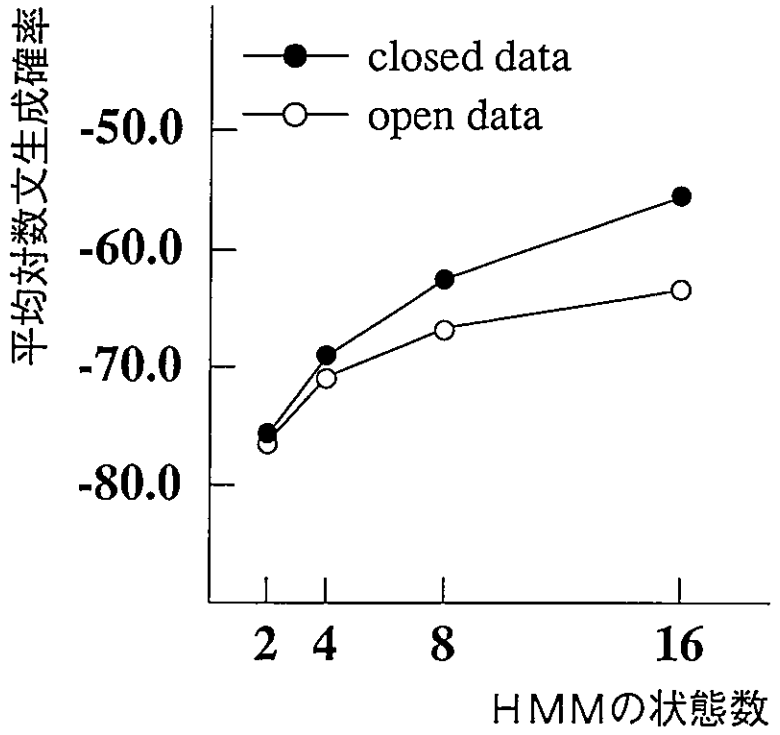


図 3.23: 平均対数文生成確率

表 3.23 , 図 3.22 , 図 3.23 から、HMM の状態数が多いほどエントロピーが下がっているのがわかる。解析結果のネットワーク図において、状態数が増えることによって、HMM が単語をより細かく分類して出力していた。つまり、状態数が増えることによって、一つの遷移で出力される単語の分布の偏りが大きくなり、エントロピーが下がると思われる。また、状態数が大きくなるにつれ、closed data の平均対数文生成確率とともに open data の

平均対数文生成確率も改善されている。状態数が増えるにつれ、closed data と open data の平均対数文生成確率の差が開くことがわかる。この原因として、open data に closed data に存在しない単語（未知語）が多数含まれていることが考えられる。実際に調べた結果、even4000 には、未知語を含む文が 990 文あった。HMM の学習アルゴリズムでは出現しないシンボルの出力確率は 0.0 になるため、未知語を含む文の生成確率は 0.0 になる。これを回避するため、フロアリングを行なった。しかし、フロアリングする値の明確な基準がないため、1.0×10^{-5}(1/6418 よりも 1 桁小さい値で、このくらいが妥当であると判断した) でフロアリングした。状態数が増えれば、1 シンボルあたりの出力確率は大きくなり、すべての状態の HMM に対して同じ値でフロアリングすることは適当ではない。すなわち、状態数の異なる HMM で、フロアリング値を一定にした場合、状態数が増えるにつれてフロアリングされた未知語出力確率は、他の単語の出力確率に対して相対的に小さくなる。これが、open data と closed data との平均対数文生成確率の差が開く原因の一つと推察できる。

3.3.8 学習データ量とモデル化の関係

2状態、4状態、8状態のergodic HMMにodd1000、odd2000、odd4000を学習させた結果について、open dataの平均対数文生成確率を計算し獲得された文法の一般性を調べた。検証データとしてeven4000を用いて対数生成確率の一文当たりの平均を求めた結果を表 3.24 、図 3.24 に示す。また、合わせて各学習データ量におけるモデルのエントロピーを表3.24 に示す。

表 3.24: 学習データ量と平均対数文生成確率の関係

状態数	学習データ	検証データ	平均対数文生成確率	エントロピー
2	odd1000	even4000	-82.60	7.20
	odd2000	even4000	-80.43	7.22
	odd4000	even4000	-77.37	7.53
4	odd1000	even4000	-77.33	6.35
	odd2000	even4000	-75.22	6.34
	odd4000	even4000	-71.30	6.72
8	odd1000	even4000	-76.31	5.58
	odd2000	even4000	-73.35	5.59
	odd4000	even4000	-67.48	6.00

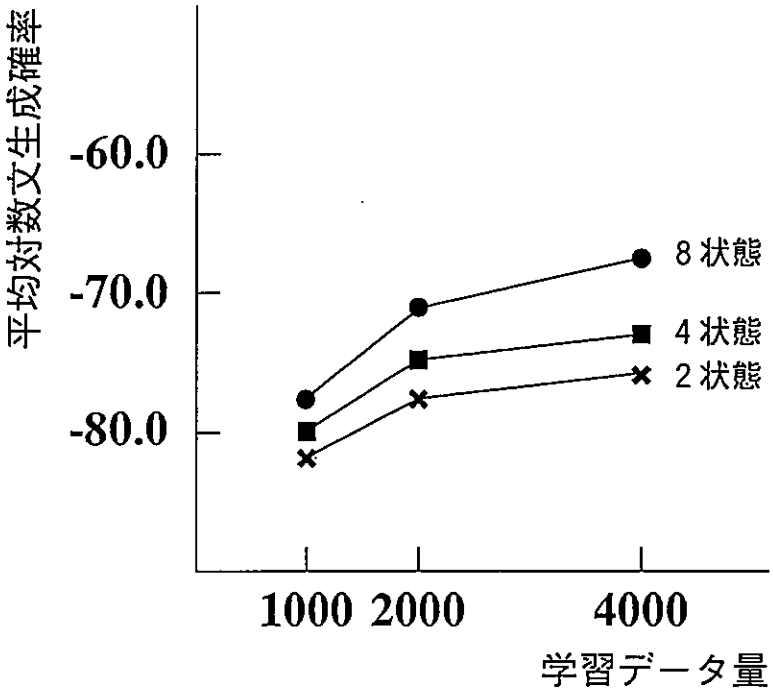


図 3.24: 学習データ量と平均対数文生成確率との関係

表 3.24 、図 3.24 から、全ての HMM で学習データを増加すると、open data の平均対数文生成確率が高くなることが示された。また、データ数の増加にともないエントロピーが上昇することもわかった。この結果から学習データ量を増やすことで、エントロピーは増加するものの、より一般性のあるモデルが獲得されることが明かになった。

3.4 連続音声認識への適用

Ergodic HMM が獲得した確率つきネットワーク文法を連続音声認識に適用し、その有効性を調べた。その概要を次に述べる。

3.4.1 実験条件

表 3.25: 連続音声認識実験の条件

音素モデル数	52 音素
音素音響モデル	4 状態 3 ループ混合分布型 HMM
	混合数は音素ごとに異なる。
	継続時間長制御なし。
話者	男性アナウンサー 1 名 (MAU)
音響パラメータ	log パワー + 16 次 LPC ケプストラム
	+Δlog パワー + 16 次 LPCΔ ケプストラム
音響分析条件	サンプリング周期 12kHz
	フレーム窓長 20ms
	フレーム周期 5ms
ビーム幅	4096
認識語彙数	435 単語
テストデータ	同一話者発声 (MAU)
	38 文

認識実験では、テストデータとして学習データ odd4000 と同一タスクの open data 38 文を用いた。

その他の実験条件を表 3.25 に示す。音響尤度の計算には音素モデル HMM、Viterbi algorithm を用い、これに言語モデルとして odd4000 で学習した状態数 2,4,8 の ergodic HMM を使用した。

同一のテストデータを使って、open data,closed data 両方の文認識を行ない比較するために、odd4000 で学習したパラメータを初期値とし、odd4000 にテストデータ 38 文を加えた 4038 文を学習させた ergodic HMM を closed data に対する文認識に用いた。4038 文で学習させた時の学習条件は、パラメータの初期値以外は odd4000 で学習させた時と同じである。

単語間の接続部分で、対数音響尤度の値に ergodic HMM から Viterbi algorithm で計算

された単語間接続確率の対数値を加えた。この際、ergodic HMM から得られる値に

対数音響尤度 : 対数単語間接続確率 = 1 : 16

の比率で重みをつけた。また、認識時の計算量を少なくするためビーム幅 4096 でビームサーチを行なった。

3.4.2 実験結果

表 3.26 , 図 3.25 に実験結果を示す。比較のため、言語モデルを用いない場合 (音響モデルのみで認識した場合) と言語モデルとして単語 bigram (対数音響尤度: 単語 bigram の対数値 = 1 : 1) を用いた場合を合わせて示す。表中に示した () 内の値は (正解を出力した文数 / 認識に用いた文数) である。言語モデルとして ergodic HMM を用いた場合は単語 bigram を用いた場合には及ばないが、言語モデルがない場合に比べ高い認識率になることがわかる。また、状態数が増えるにつれ、closed データに対する認識率が上がることが確認される。

表 3.26: 認識実験の結果

言語モデル	文認識率	
なし	29.0% (11/38)	
	open data	closed data
2 状態 ergodic HMM	31.5% (12/38)	34.2% (13/38)
4 状態 ergodic HMM	36.8% (14/38)	39.5% (15/38)
8 状態 ergodic HMM	39.5% (15/38)	47.3% (18/38)
16 状態 ergodic HMM	36.8% (14/38)	——
単語 bigram	34.3% (13/38)	52.7% (20/38)

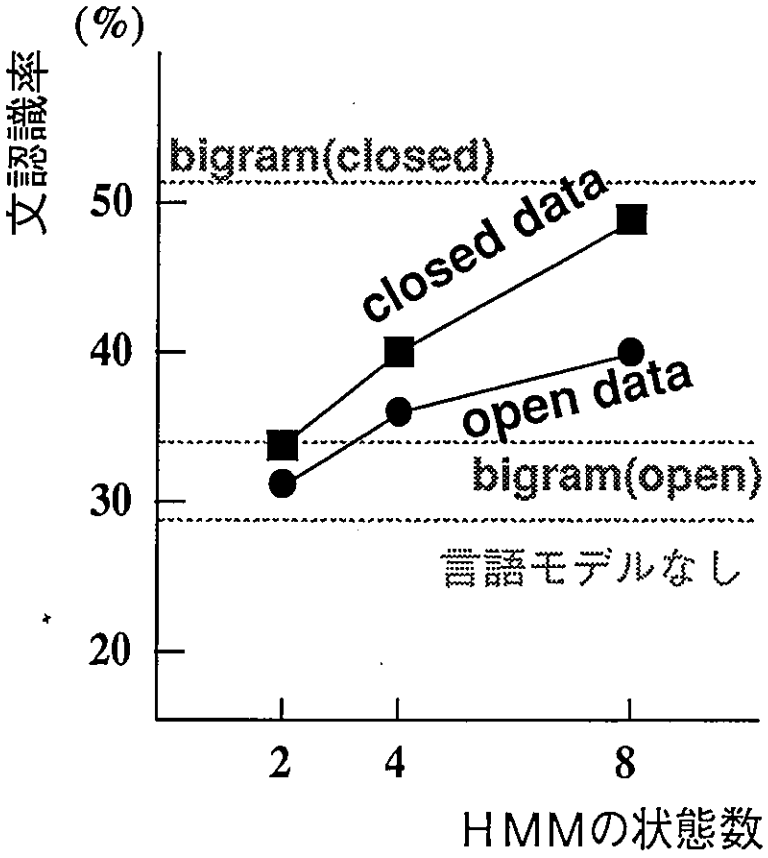

(%)
文認識率
bigram(closed)
50
closed data
40
open data
bigram(open)
30
言語モデルなし
20
2
4
8
HMMの状態数

図 3.25: 文認識結果

3.5 初期パラメータの違いによるモデルの変化

Ergodic HMM のパラメータの初期値の違いによる生成されるモデルの変化を調べるために、初期値の異なる 8 状態 ergodic HMM を 8 個用意し、odd4000 を学習させた。初期値が異なること以外はすべて同一条件で学習を行ない、パラメータの再推定回数 20 回を学習の終了条件とした。初期パラメータは 3.2 節で示した方法で決定した。

表 3.27 に学習終了後の各 ergodic HMM のエントロピー、学習データ odd4000 の平均対数文生成確率、音声認識に用いた場合の closed data に対する認識率などを示す。表中、一番左は各 ergodic HMM を区別するために番号をつけた。1 は、 3.3.4 節で示した、学習終了条件を生成確率の上昇率 1% 未満にした場合の結果である。

表 3.27: 初期値の違いによる生成モデルの変化

	学習回数	エントロピー		平均対数文生成確率		認識率
		初期状態	学習後	初期状態	学習後	
1	749	12.61	5.99	-126.39	-62.93	47.3%
2	20	12.61	6.21	-126.53	-64.48	42.1%
3	20	12.61	6.14	-126.36	-64.36	39.5%
4	20	12.61	6.03	-126.43	-63.68	44.7%
5	20	12.61	6.27	-126.10	-64.71	34.2%
6	20	12.61	6.09	-126.41	-64.95	36.8%
7	20	12.61	6.10	-126.24	-64.06	36.8%
8	20	12.61	6.16	-126.47	-64.62	39.5%
9	20	12.61	6.13	-126.32	-65.07	31.6%

表 3.27 から、パラメータの初期値によって、生成されたモデルのエントロピーや平均対数文生成確率が異なることがわかる。これを認識に用いた場合の結果は、学習回数の異なる 1 を除いた 8 種の ergodic HMM 間で、認識率の最高値 (4 の ergodic HMM) と最低値 (9 の ergodic HMM) の差は差は約 13%(5 文) もあり、初期モデルの違いでかなりの違いが見られる。また、学習後の平均対数文生成確率と認識率の関係を図 3.26 に示す。

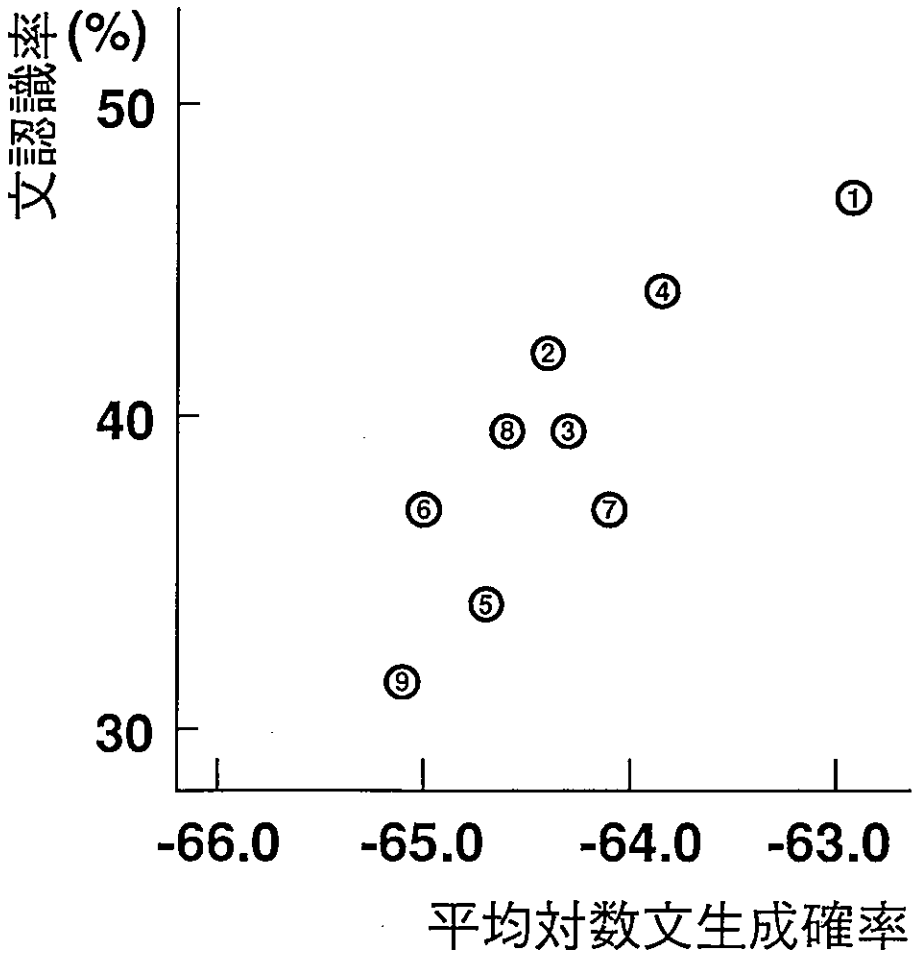


図 3.26: 文認識結果

学習させた回数が 20 回で十分でなかったことと、8 種類と学習を行なった ergodic HMM が少なかったため、明言することはできないが、図 3.26 から、平均対数文生成確率が高い ergodic HMM の認識率が高くなっている。

第 4 章

考察

4.1 HMM による文法の自動獲得の可能性

学習後の ergodic HMM の解析結果で示したように、ergodic HMM が文法的な特徴を獲得しており、open data と closed data の生成確率に大差がないことから、ergodic HMM が言語データを学習することにより、一般性のあるネットワーク文法が自動的に獲得できる可能性があると考えられる。また、体言、活用する品詞、格助詞などの分離に見られるように、ergodic HMM は単語を文中での機能により細かいカテゴリーに分けて表現しており、単語列の学習によって従来の品詞より詳細な単語のカテゴリーを獲得する能力もあると思われる。

また、活用する品詞が 4 状態、8 状態、16 状態と ergodic HMM の状態数が増えるごとに、より細かく分類されたのをはじめ、他の品詞でも状態数を増やすことでより細かく分離して出力された。モデルのエントロピー、open data の生成確率も状態数を増やすことで改善されることも実験結果から明らかになった。今後さらに状態数を多くすれば、エントロピーが小さくなり、モデルの表現能力も高くなることが予想され、より詳細な文法・単語のカテゴリーを獲得できると思われる。

4.2 認識実験の結果について

HMM が生成した確率つきネットワーク文法を用いた連続音声認識実験の結果から、open data、closed data 両方に対し言語モデルを用いない場合より高い認識率を示し、HMM が生成した文法が音声認識実験に有効であることがわかった。しかし、3.5節で示したように、認識率は HMM の初期モデルによって全く異なる値を示すので、初期状態を変えたモデルについて認識実験を行なう必要がある。また、認識時に言語モデルから計算される接続確率につける重みの最適値の決め方も課題の一つである。

4.3 HMMの学習における問題点

4.3.1 学習データ量について

本研究では、ergodic HMMに、学習によって文法・単語カテゴリーを獲得し表現する能力があることが示された。さらに良いモデルを得るためには、パラメータを精度よく推定する必要がある。本研究で行なったergodic HMMの学習では、学習データ量が十分ではなかったと思われる。8状態のergodic HMMにodd4000を学習させた場合を例にとると、8状態ergodic HMMが持つパラメータ数は、初期状態確率が8個、状態遷移確率が64個(＝8状態×8状態)、シンボル出力確率が410752個(＝8状態×8状態×6418単語)で合計410752個である。これに対し、odd4000の総単語数は57354個であった。これから、推定すべきパラメータの数に対し、推定に用いられる学習データが十分でないと考えられる。HMMのパラメータの推定に必要な学習データ量の明確な基準はないが、十分にモデル化するには、少なくともHMMのパラメータ数と同数の学習データ量が必要であると思われる。したがって、パラメータを精度良く推定するには、さらに学習データ量を増やす必要があると思われる。

4.3.2 初期モデルについて

本研究で行なった実験では、状態数の異なる4種の状態遷移出力型のergodic HMMを用いたが、状態数によって表現力が異なり、本研究で行なった範囲では状態数が多いほどエントロピーやopen dataの生成確率が改善され、より良いモデルを得ていていることがわかった。しかし、シンボル数(単語数)に対して最適な状態数などHMMの構造に関しては不明な点が多い。

また、状態数が同じ場合でも、ergodic HMMのパラメータの初期値を変えた学習の実験結果から、初期値によって異なるモデルを生成することがわかった。これはHMMの学習で学習アルゴリズムがローカルミニマムに収束するのが原因である。また、パラメータ数、学習データ量が多くなるとパラメータ推定に膨大な計算量がかかり、初期モデルによって収束するまでにかかる学習回数が異なることが予想される。本研究で行なった実験のように学習データ量が少ない場合でも、学習にはかなりの時間がかかるので、計算量を減らしてより精度よくパラメータを推定するためにもモデルの初期値の決め方が重要な問題になる。効率の良い学習法、最適な初期値の決定法は今のところ知られていないが、考えられる有効な一つの方法として、いくつかの初期モデルでに適当に学習を繰り返した後に、文の生成確率やエントロピーを計算し、結果の良好なものにについて学習を続行することが挙げられ

る。

4.3.3 学習データの作成に関する問題点

3.1節で説明したように、本研究で用いた言語データは表記が同一なものでも品詞や活用形の異なる単語に異なるラベルをつけてある。このことによって単語に品詞情報が加わると考えられるので、正確に“品詞情報のない単語列からの文法および単語カテゴリーの自動獲得”の可能性を検証するためには、同一表記の単語には同じラベルをつけたデータをHMMに学習させる実験を行なう必要があると思われる。

本研究では、3.1 節で説明したように、言語データベースを奇数偶数の 2 set に分け、それぞれ先頭から 1000 文, 2000 文, 4000 文を data set とした。表 3.7 から、even と odd では構成単語数に大差はないがしかし、データの前半部分と後半部分で対話文の長さに違いがあることがわかる。表 ?? に odd4000 の前半と後半の 2000 文、odd1000, odd4000 のデータを示す。(前半 2000 文は odd2000 と同じ。)

表 4.1: 構成単語数 2

set	odd1000	odd2000	odd4000 の後半 2000 文	odd4000
文数	1000	2000	2000	4000
単語数	13299	20730	36624	57354
文平均単語数	13.30	10.37	18.31	14.34
最大単語数	99	99	128	128

表 4.1 から、前半後半で構成単語数がかなり異なり、後半部分の一文あたりの単語長が前半部分の約 2 倍になっている。

odd1000,odd2000 は文平均単語数が odd4000 に比べ少ない。表 3.14 を見ると、odd2000 では「はい」「もしもし」などの一単語のみの文の割合が 19.95% で odd1000(12.70%), odd4000 (15.25%) に比べ多く含まれ、odd2000 で (odd1000 に) 新たに加わった文の中に 30 単語以上の文が 5 文で非常に少ない。odd2000 では 30 単語を越える長い文の比率が odd1000,odd4000 に比べ小さくなっている。また、odd1000 では一単語の文はそれほどないが、3 単語、4 単語の短い文の比率が高く、逆に odd4000 では長い文が多く含まれている。

以下に 8 状態の ergodic HMM に odd1000, odd2000 を学習させた場合の解析結果を図 4.1、図 4.2 に示す。(odd4000 を学習させたものは図 3.4 参照。)

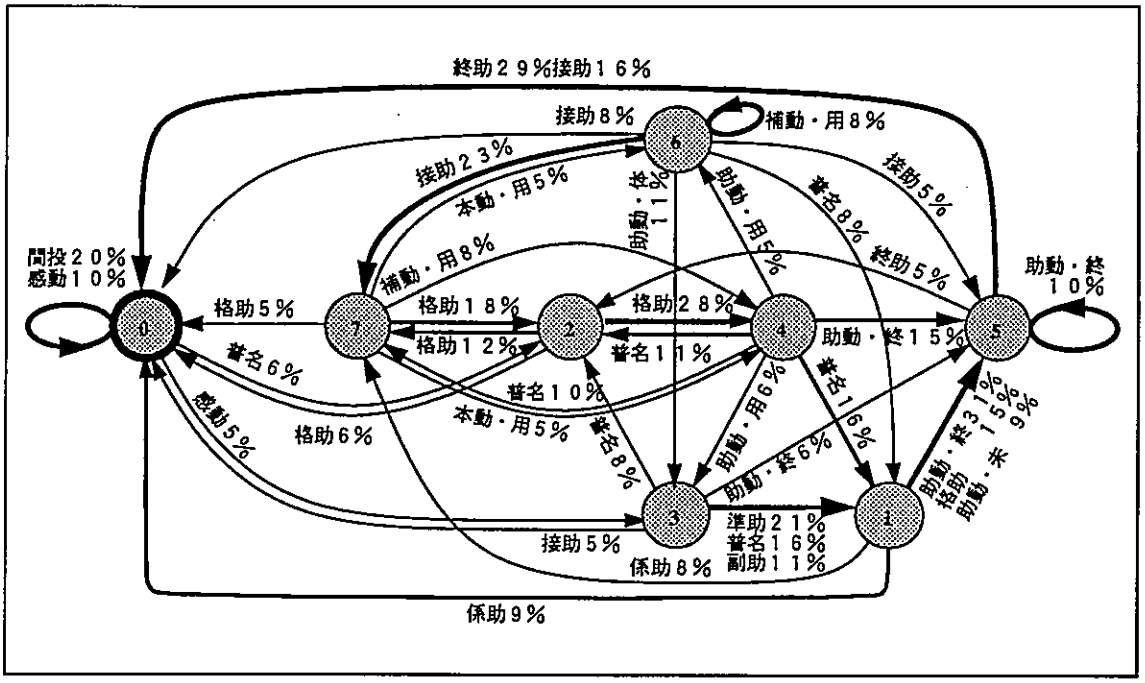


図 4.1: odd1000 を学習させた 8 状態 ergodic HMM

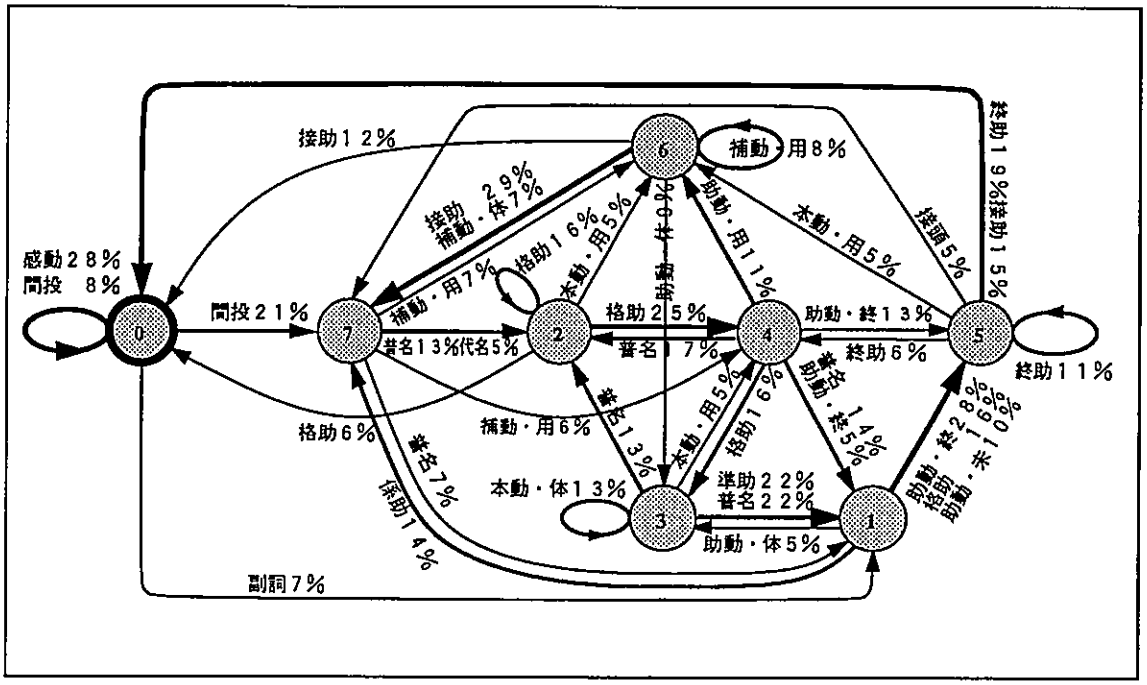


図 4.2: odd2000 を学習させた 8 状態 ergodic HMM

一単語の文が多く含まれる odd2000 で学習した HMM は状態①での自己ループでの感動詞の出力が他の HMM に比べて高く、odd4000 で学習した HMM では感動詞の出力とともに自己ループの遷移確率も低くなっている。学習データ量が異なるので学習データの性質の違いによるものかどうかはわからないが、ネットワークの形態そのものもかなり異なっている。

1000 文、2000 文の set を作成する際に、odd、even それぞれの 4000 文の全体から 4 の倍数番目、2 の倍数番目のように全体から均一に抽出すれば、このような data set による文の性質の違いを緩和できたと思われる。

第 5 章

結論

本論文では、ergodic HMM と確率つきネットワーク文法が類似した構造を持ち、同種のパラメータで表現されること、さらにデータを入力すると HMM のパラメータは 2.3.2節で紹介した Baum-Welch algorithm で学習できることに着目し、ergodic HMM による言語のモデル化の検討を行なった。

3章で、ergodic HMM に品詞情報を持たない単語列を学習させ、3.3.2節〜3.3.5節で学習後の HMM を解析した結果、ergodic HMM の構造は学習データの特徴をとらえた文法的な特徴を示しており、単語を文中での機能によって分類して出力していることがわかった。また、ergodic HMM の状態数が増えるほど詳細な表現が可能となり、より細かく単語の分類を行なっていることがわかった。

3.3.7節で生成確率を調べた結果、open data も closed data と大差なく生成されることから、ergodic HMM が学習によって一般性のある文法を生成していることがわかった。また、状態数が増えることにより、言語モデルの複雑さの指標となるエントロピーが改善され、open data に対する生成確率も高くなることがわかった。

学習データ量を増やすことによって open data に対する生成確率が高くなることが 3.3.8節で明らかになった。

以上のことから、ergodic HMM を用いてテキストデータから一般性のある確率つきネットワーク文法を自動生成し、単語を分類できる可能性が示された。また、一般性のあるよりよいモデルを得るためには、ergodic HMM の状態数を増やし大量の言語データで学習する必要があることが示された。

3.4節で、得られた ergodic HMM を言語モデルとして用いて、連続音声認識実験を行なった結果、open data、closed data ともに言語モデルのない場合に比べ認識率が良くなっており、HMM が獲得した文法は連続音声認識に有効であることが示された。

3.5節では、HMM の学習に問題になる初期パラメータの変化によるモデル化の違いに

ついて調べた。その結果、初期モデルによって生成されるモデルは異なり、それらのモデルを使った文認識実験では、認識率の差は最高値と最低値で約 13% の違いが見られ初期パラメータの設定法が重要であることが明らかになった。文認識率とモデルの文生成確率には相関が見られ、文生成確率のよいモデルほど文認識率は向上する傾向が見られた。このことから、よりよいモデルを少ない計算量で得るための方法として、初期値の異なる複数の ergodic HMM で学習を行ない、学習を数回繰り返したところで文の生成確率やエントロピーを基準に適当と思われる ergodic HMM について学習を続行する方法があげられる。

謝辞

本論文を終えるにあたり、末筆ながらこれまでお世話して下さった方々に感謝の意を表し、お礼を述べたいと思います。

本研究を進めるにあたって、懇切なる御指導を賜わり、ATR 自動翻訳電話研究所での研究の機会を与えて下さった指導教授、葉原耕平教授に心から感謝の意を表します。

また、おりにふれて適切な御助言を頂いた堀内和夫教授ならびに大石進一教授に心から感謝の意を表します。

また、研究の機会を与えて頂いた ATR 自動翻訳電話研究所 樽松明社長、貴重な御助言、御検討を頂いた同研究所音声情報処理研究室、嵯峨山茂樹室長、村上仁一研究員をはじめとした同研究所の皆様に感謝致します。

日頃より、御激励下さった堀内研究室、大石研究室の皆様に感謝致します。

最後に、私事ですが、3 年間に及ぶ私の研究生活を支援してくれた両親に、心から感謝致します。

参考文献

[1] 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 回全国大会予稿集, pp.2-237-238 (1990).

[2] 村瀬功, 上田佳央, 中川聖一 : "文脈自由文法と bigram,trigram による言語のモデル化の検討", 電子情報通信学会技術研究報告, SP90-5, pp.49–56 (1990).

[3] 田本真詞, 村上仁一, 嵯峨山茂樹 : "HMM を利用した言語獲得の可能性について", 人工知能学会研究会資料, SIG-SLUD-9201-6, pp.47–54 (1992).

[4] 山本寛樹, 村上仁一, 嵯峨山茂樹 : "HMM を言語モデルに用いた連続音声認識の検討", 情報処理学会第 45 回全国大会予稿集, vol.3, pp.227–228 (1992.10).

[5] 山本寛樹, 村上仁一, 嵯峨山茂樹 : "HMM を言語モデルに用いた連続音声認識の検討", 日本音響学会秋季講演論文集, vol.1,pp.193–194 (1992.10).

[6] X.D.Haung, Y.Ariki, M.A.Jack : "Hidden Markov models for speech recoginition",Edinburgh University Press (1990).

[7] 中川聖一 : "確率モデルによる音声認識", 電子情報通信学会 (1988).

[8] 江原暉将, 小倉健太郎, 森本逞 : "電話対話ベースの構築", 情報処理学会第 40 回全国大会予稿集, vol.1, pp.486–487 (1990).

[9] 江原暉将 : "対話データベースからの統計情報の抽出", 情報処理学会第 40 回全国大会予稿集, vol.3, pp.83–84 (1990).

[10] 瀧保夫 : "情報論 I ", 岩波 (1978).

1 言語モデルを生成するためのｅｒｇｏｄｉｃ　ＨＭＭ学習プログラム

このプログラムは言語データ（シンボル列）を、
ＨＭＭに学習させるプログラムで、
実行時には学習データ（シンボル番号列）のファイルが必要である。
パラメータの再推定を行なうごとに更新されたパラメータをファイルに保存する。
また、再推定の際にアンダーフロー回避のためスケーリングを行なっている。

1.1 プログラムの構成

```
main program : hmm             : メインプログラム、学習終了判定
sub rotine   : reestimate      : ＨＭＭのパラメータの再推定
               init            : 学習開始時のパラメータ読み込み
               forward         : Forward algorithmで計算されるαの値を返す関数
               backward        : Backward algorithmで計算されるβの値を返す関数
               init_forward    : Forward algorithmによりαの値を計算
               init_backward   : Backward algorithmによりβの値を計算
               initialize      : ＨＭＭの初期パラメータをランダムに設定
               write_data      : ＨＭＭの学習後のパラメータをファイルにセーブ
               cal_entro       : モデルのエントロピーを計算
header file  : hmm_state.h
makefile
```

1.2 ＨＭＭ・学習条件の設定

設定はすべてhmm_state.hで行う。
各定数の意味は次の通りである。

(a)ＨＭＭの設定・学習終了条件

```
MAX_DATA        : 学習させるシンボル列の長さの最大値
STATE_NUMBER    : ergodic HMM の状態数
SYMBOL_NUMBER   : 学習させるシンボルの種類
END_CONDITION   : 学習を終了させる条件（尤度の上昇率）
END_ITERATION   : 学習を終了させる条件（再推定の回数）
INIT_RAND       : 乱数発生の初期値（適当な整数）
MIN             : logを計算できる最小値（アンダーフロー回避のため）
INIT_PROB       : 尤度の初期値
INIT_ITERATION  : 学習回数の初期値
```

例えば、３０００種類の単語で構成される単語列１０００文を学習データとして用い、５状態のergodic HMMに学習させる場合は、１０００文中最も長い文の単語数が５０であったとすると、設定は以下のようになる。

```
MAX_DATA         50
STATE_NUMBER      5
SYMBOL_NUMBER  3000
```

学習は尤度の上昇率が END_CONDITION 未満になるか、再推定の回数（学習回数）が END_ITERATION に達した時に終了する。

例えば、尤度上昇率１０％で学習を終了する場合は
END_CONDITION   0.1
END_ITERATION 99999 (<-- 適当に大きな数字)
とすれば良い。
また、逆に学習回数で制御したい場合は、END_CONDITIONの値を
1.0E-50などのように極端に小さい値にしておけば良い。

最初から学習を行なう場合は尤度、学習回数の初期値を０にし、
途中から行なう（以前に学習した結果を初期モデルとして学習する）
場合はINIT_PROB INIT_ITERATION に相当する数値を代入する。
たとえば、２０回パラメータの推定を繰り返したところで

学習を終了してモデルの尤度が -23124.3478 となったHMM
のデータがあり、さらに学習を繰り返す場合は、
INIT_PROB　　-23124.3478
INIT_ITERATION　　20
とする。
また、この場合初期モデルのパラメータファイル名
FILE_INIT_A,FILE_INIT_B,FILE_INIT_PI は２０回学習を繰り返した
HMMのパラメータがセーブしてあるファイル名に書き換える必要がある。

(b)データのセーブに関する設定

SAVE_STEP　　：学習の途中結果をセーブする再推定回数の間隔
MAKE_ASCII_FILE_A　：状態遷移確率のASCII FILE を作る(1)か作らない(0)か
MAKE_ASCII_FILE_B　：シンボル出力確率のASCII FILE を作る(1)か作らない(0)か
MAKE_ASCII_FILE_PI　：初期状態確率のASCII FILE を作る(1)か作らない(0)か
MAKE_ASCII_FILE_ALL：すべてのパラメータを記述するASCII FILEを作らない(0)
　　作る(1)
　　状態遷移確率、初期状態確率を記述するASCII FILEを作る(2)

SAVE_STEPで学習の途中結果をどのくらいのインターバルで残すかを設定する。
例えば、SAVE_STEPを１０にした場合、状態遷移確率をセーブしたファイル名が
"test.a"だとすると、学習を１０回繰り返すごとに "test.a.10","test.a.20",
"test.a.30"のようにファイル名に".学習回数"をつけたファイルを作り保存する。

(c)セーブするファイル名

FILE_ENT　　：モデルのエントロピーを記録するファイル名 (ASCII FILE)
FILE_A　　：状態遷移確率を記録するファイル名 (BINARY FILE)
FILE_B　　：シンボル出力確率を記録するファイル名 (BINARY FILE)
FILE_PI　　：初期状態確率を記録するファイル名 (BINARY FILE)
FILE_A2　　：状態遷移確率を記録するファイル名 (ASCII FILE)
FILE_B2　　：シンボル出力確率を記録するファイル名 (ASCII FILE)
FILE_PI2　　：初期状態確率を記録するファイル名 (ASCII FILE)
FILE_ALL　　：尤度、エントロピー、パラメータなどを記録するファイル名 (ASCII FILE)
FILE_INIT_A　　：学習開始時の状態遷移確率を記録するファイル名 (BINARY FILE)
FILE_INIT_B　　：学習開始時のシンボル出力確率を記録するファイル名 (BINARY FILE)
FILE_INIT_PI　　：学習開始時の初期状態確率を記録するファイル名 (BINARY FILE)
LEARN_DATA_FILE　　：学習データのファイル名 (ASCII FILE)

1.3 実行形式

hmm_state.h を実験条件に合わせて書き換える。
あとは、以下のようにタイプする。
make <return>
hmm　<return>

2 音声認識プログラムについて

このプログラムは、連続音声文認識を行なうもので、
言語モデルとして ergodic HMM を用いている。
音響尤度の計算にはフレーム同期型 one pass dp アルゴリズムを用い、
単語間の接続部で viterbi algorithm によって言語モデルから得られた
単語間の接続確率に重みをつけたものを加えている。
実行時には、このプログラムの他に言語モデルとして用いるＨＭＭのデータファイル、
認識させる音声のデータファイル、音素ＨＭＭのデータファイル、
単語辞書ファイルが必要となる。

2.1 プログラムの構成
プログラムは、main program と14個の sub rotine 、header file、makefile から構成される。

main program : dp　　：メインプログラム、認識結果の表示
sub rotine　　: read_phoneme_data　　：発声された音声の音響分析地の読み込み
　　: init_parameter　　：尤度の計算に必要なパラメータの更新
　　ctoi　　：charの数字をintに変換

```
                read_phoneme_HMM_data   : 音素HMMのパラメータ読み込み
                viterbi                 : viterbi algorithmによる尤度の計算
                viterbi_fr_0            : viterbi algorithmによる尤度の計算(0 frame)
                viterbi_fr_last         : viterbi algorithmによる尤度の計算(最終frame)
                sort_probability        : 認識結果のソーティング
                read_word_data          : 単語データの読み込み
                read_language_HMM_data  : 言語HMMのデータ読み込み
                cal_data                : 尤度の計算に必要なパラメータを返す関数
                cal_log                 : logの計算（アンダーフロー回避）
                check_lh                : ビームサーチの処理
                calc_b                  : 音素HMMのシンボル出力尤度の計算
header file   : parameter.h
makefile      : makefile
```

2.2 実行形式

必要に応じてparameter.h に書かれている定数やファイル名を変更し、

make (return)

でコンパイル後、

dp <A> <B> (return)
<A> : 認識する文の音声データの最初のフレーム番号
<B> : 認識する文の音声データの最後のフレーム番号

で実行される。

LINE # | TEXT

```
  0  :  -
  1  :  CH1
  2  :  CH2
  3  :  H
  4  :  K1
  5  :  K2
  6  :  N
  7  :  P1
  8  :  P2
  9  :  Q1
 10  :  S
 11  :  SH
 12  :  TS1
 13  :  TS2
 14  :  Ui
 15  :  Uu
 16  :  a
 17  :  a1
 18  :  b1
 19  :  b2
 20  :  by
 21  :  cy
 22  :  d1
 23  :  d2
 24  :  e
 25  :  e1
 26  :  g1
 27  :  gy1
 28  :  hy
 29  :  i
 30  :  i1
 31  :  ky
 32  :  m
 33  :  my
 34  :  n
 35  :  ng
 36  :  ngy
 37  :  ny
 38  :  o
 39  :  o1
 40  :  py
 41  :  r
 42  :  ry
 43  :  sy
 44  :  t1
 45  :  t2
 46  :  u
 47  :  u1
 48  :  w
 49  :  y
 50  :  z
 51  :  zy
```

LINE # TEXT

```
/*---------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <viterbi_fr_last.c>

        calculate likelyhood using viterbi algorithm at last frame

                viterbi を用いた尤度の計算（最終フレーム）

         Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
---------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <strings.h>
#include "parameter.h"

int viterbi_fr_last()
{
  int     i,wn2               ;
  int     br,wn,bws           ; /* ビーム番号,単語番号,単語ＨＭＭ内の状態番号              */
  int     bl,bms              ; /* これまでに認識された単語の数・音語ＨＭＭ内の状態番号  */
  int     ph,ph_st1,ph_st2    ; /* 音素番号,音素ＨＭＭの状態１,音素ＨＭＭの状態２       */
  int     word_change_sw      ; /*   1...change   0...unchange                          */
  int     temp_reco_word      ; /*   現在計算中の単語                                   */
  int     temp_word_state     ; /*   現在計算中の単語ＨＭＭ内の状態番号                 */
  int     temp_word_length    ; /*   これまでに認識された単語の数                       */
  int     temp_model_state    ; /*   現在計算中の音語ＨＭＭ内の状態番号                 */
  double  temp_lh             ; /*   計算中の尤度                                       */
  double  temp_conenction_pr  ; /*   単語間の接続確率（一時）                           */
  double  max_connection_pr   ; /*   単語間の接続確率最大値                             */
  double  log_connection_pr   ; /*   log(単語間の接続確率の最大値）                     */
  int     max_connection_st   ; /*   接続確率が最大値になる遷移先の状態                 */

  /* 関数・サブルーチン */

  int    check_lh() ; /* ＢＥＡＭＳＥＡＲＣＨ 保持するデータの調整                  */
  int    cal_data() ; /* 単語番号,単語ＨＭＭの状態番号,音素,音素ＨＭＭの状態番号を返す */
  double cal_log()  ; /* logを計算
  /* ------------------------ 初期化 initialize ------------------------ */

  for(br=0;br<TEMP_BEAM_RANGE;br++) {
    log_lh[br] = MIN_DOUBLE ;
    model_state[br] = -3 ;
    word_state[br]  = -3 ;
    word_length[br] = -3 ;
    for(i=0;i<MAX_NUMBER_OF_WORD;i++)
      reco_word[br][i] = -3 ;
  }

  /* ------------------------ main start  ------------------------ */

  BEAM_NUM = 0 ;
  for(br=0;br<BEAM_RANGE;br++) {
    cal_data(br,&wn,&bws,&ph,&ph_st1,&ph_st2,&bl,&bms) ;

    if(before_log_lh[br]>MIN_DOUBLE && bws == number_of_state[wn]-1) {

      temp_lh    =   before_log_lh[br]
                   + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
                   + cal_log((double)b[ph][MAX_STATE-1])          ;

      temp_reco_word   = wn ;
      temp_word_state  = bws+1 ;
      temp_word_length = bl ;
      temp_model_state = bms ;
      word_change_sw   = 0 ;

      cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
    }
  }
}
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <viterbi_fr_0.c>

       calculate likelyhood using viterbi algorithm at frame 0

                  viterbi を用いた尤度の計算 ( フレーム 0 )

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
-------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "parameter.h"

int viterbi_fr_0()
{
  int    br,i,j,k,wn,st ;
  double mp_max,mp_temp,log_mp_max,temp_log_lh ;
  int    mp_max_bms ;
  double cal_log() ;

  BEAM_NUM = 0 ;
  for(wn=0;wn<MAX_WORD;wn++) {
    if(wn!=PAUSE) {                              /* 無音以外の単語について */
      mp_max = MIN_DOUBLE  ;
      for(i=0;i<STATE_NUMBER;i++) {
        for(j=0;j<STATE_NUMBER;j++) {            /* 言語ＨＭＭで遷移開始時の尤度計算 */
          mp_temp = model_pi[i] * model_a[i][j] * model_b[i][j][word_number[wn]] ;
          if(mp_temp>mp_max) {                   /* 単語(wn)の出力尤度が最大になる遷移を選ぶ */
            mp_max_bms = j ;                     /* mp_max : 言語ＨＭＭでwnを出力する最大尤度 */
            mp_max = mp_temp ;                   /* mp_max_bms (bms = before_model_state */
          }
        }
      }
    }
    else {                                       /* 無音部が文頭に来た場合について */
      mp_max = 1.0 ;
      mp_max_bms = -1 ;
    }

    log_mp_max = cal_log(mp_max) ;

    for(st=0;st<=1;st++) {
      temp_log_lh  = LANGUAGE_WEIGHT * log_mp_max
                   +cal_log((double)a[model[wn][0]][0][st])
                   +cal_log((double)b[model[wn][0]][0])    ;

      log_lh[BEAM_NUM]      = temp_log_lh ;
      model_state[BEAM_NUM] = mp_max_bms ;
      word_state[BEAM_NUM]  = st ;
      reco_word[BEAM_NUM][0]     = wn ;
      BEAM_NUM++ ;
    }
/*        printf("wn %d : st %d : log_lh : %e \n",wn,st,before_log_lh[wn][st]) ; */
  }

  /* 計算値の数がビーム幅を越えない時は default の値の MIN_DOUBLE が残る */

}
```

```
LINE #  TEXT
241     }
242
243
244
245     }
```

```
LINE #  TEXT
121                         + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
122                         + cal_log((double)b[ph][MAX_STATE-1])
123                         + LANGUAGE_WEIGHT * log_connection_pr ;
124
125               temp_reco_word   = wn2 ;
126               temp_word_state  = 0 ;
127               temp_word_length = bl ;
128               temp_model_state = max_connection_st ;
129               word_change_sw   = 1 ;
130
131               cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
132             }
133             else {
134
135               /* -------------------- 2.2.1.2 ポーズ以外からポーズへ遷移 transit from word to pause --- */
136
137
138               temp_lh    =   before_log_lh[br]
139                            + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
140                            + cal_log((double)b[ph][MAX_STATE-1])          ;
141
142               temp_reco_word   = wn2 ;
143               temp_word_state  = 0 ;
144               temp_word_length = bl ;
145               temp_model_state = bms ;
146               word_change_sw   = 1 ;
147
148               cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
149             }
150           }
151         }
152
153         else {
154
155           /* -------------------- 2.2.2 ポーズから遷移 transit from pause ------------------------- */
156
157           for(wn2=0;wn2<MAX_WORD;wn2++) {
158             if(wn2!=PAUSE) {
159
160               /* -------------------- 2.2.2.1 ポーズからポーズ以外へ遷移 transit from pause to word -- */
161
162               if(bms!=-1) {
163
164               max_connection_pr = MIN_DOUBLE ;
165               max_connection_st = 0 ;
166               for(i=0;i<STATE_NUMBER;i++) {
167                 temp_connection_pr = model_a[bms][i] * model_b[bms][i][word_number[wn2]] ;
168                 if(temp_connection_pr>max_connection_pr) {
169                   max_connection_pr = temp_connection_pr ;
170                   max_connection_st = i ;
171                 }
172               }
173               log_connection_pr = cal_log(max_connection_pr) ;
174
175               temp_lh    =   before_log_lh[br]
176                            + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
177                            + cal_log((double)b[ph][MAX_STATE-1])
178                            + LANGUAGE_WEIGHT * log_connection_pr ;
179
180               temp_reco_word   = wn2 ;
181               temp_word_state  = 0 ;
182               temp_word_length = bl ;
183               temp_model_state = max_connection_st ;
184               word_change_sw   = 1 ;
185
186               cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
187
188               }
189               else {       /* 文頭のポーズからの遷移 */
190
191               max_connection_pr = MIN_DOUBLE ;
192               max_connection_st = 0 ;
193               for(i=0;i<STATE_NUMBER;i++) {
194                 for(j=0;j<STATE_NUMBER;j++) {
195                   temp_connection_pr = model_pi[i] * model_a[i][j] * model_b[i][j][word_number[wn2]] ;
196                   if(temp_connection_pr>max_connection_pr) {
197                     max_connection_pr = temp_connection_pr ;
198                     max_connection_st = j ;
199                   }
200                 }
201               }
202
203               log_connection_pr = cal_log(max_connection_pr) ;
204
205               temp_lh    =   before_log_lh[br]
206                            + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
207                            + cal_log((double)b[ph][MAX_STATE-1])
208                            + LANGUAGE_WEIGHT * log_connection_pr ;
209
210               temp_reco_word   = wn2 ;
211               temp_word_state  = 0 ;
212               temp_word_length = bl ;
213               temp_model_state = max_connection_st ;
214               word_change_sw   = 1 ;
215
216               cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
217
218               }
219             }
220             else {
221
222               /* -------------------- 2.2.2.2 ポーズからポーズへ遷移 transit from pause to pause------- */
223
224               temp_lh    =   before_log_lh[br]
225                            + cal_log((double)a[ph][MAX_STATE-1][MAX_STATE])
226                            + cal_log((double)b[ph][MAX_STATE-1])          ;
227
228               temp_reco_word   = wn2 ;
229               temp_word_state  = 0 ;
230               temp_word_length = bl ;
231               temp_model_state = bms ;
232               word_change_sw   = 1 ;
233
234               cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
235             }
236           }
237         }
238
239       }
240     }
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                        <viterbi.c>

          calculate likelyhood using viterbi algorithm

                viterbi を用いた尤度の計算

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
-------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <strings.h>
#include "parameter.h"

int viterbi()
{
  int     i,j,wn2             ;
  int     br,wn,bws           ; /* ビーム番号,単語番号,単語ＨＭＭ内の状態番号          */
  int     bl,bms              ; /* これまでに認識された単語の数・言語ＨＭＭ内の状態番号 */
  int     ph,ph_st1,ph_st2    ; /* 音素番号,音素ＨＭＭの状態１,音素ＨＭＭの状態２      */
  int     word_change_sw      ; /*   1...change   0...unchange                         */
  int     temp_reco_word      ; /*   現在計算中の単語                                  */
  int     temp_word_state     ; /*   現在計算中の単語ＨＭＭ内の状態番号                */
  int     temp_word_length    ; /*   これまでに認識された単語の数                      */
  int     temp_model_state    ; /*   現在計算中の言語ＨＭＭ内の状態番号                */
  double  temp_lh             ; /*   計算中の尤度                                      */
  double  temp_connection_pr  ; /*   単語間の接続確率（一時）                          */
  double  max_connection_pr   ; /*   単語間の接続確率最大値                            */
  double  log_connection_pr   ; /*   log(単語間の接続確率の最大値）                    */
  int     max_connection_st   ; /*   接続確率が最大値になる遷移先の状態                */

  /* 関数・サブルーチン */

  int    check_lh() ; /* ＢＥＡＭＳＥＡＲＣＨ 保持するデータの調整 */
  int    cal_data() ; /* 単語番号,単語ＨＭＭの状態番号,音素,音素ＨＭＭの状態番号を返す */
  double cal_log()  ; /* logを計算                                                    */

  /* ------------------------ 初期化 initialize ------------------------ */

  for(br=0;br<TEMP_BEAM_RANGE;br++) {
    log_lh[br] = MIN_DOUBLE ;
    model_state[br] = -3 ;
    word_state[br]  = -3 ;
    word_length[br] = -3 ;
    for(i=0;i<MAX_NUMBER_OF_WORD;i++)
      reco_word[br][i] = -3 ;
  }
  BEAM_NUM = 0 ;

  /* ------------------------ main start  ------------------------ */

  for(br=0;br<BEAM_RANGE;br++) {
    if(before_log_lh[br]>MIN_DOUBLE) {
      cal_data(br,&wn,&bws,&ph,&ph_st1,&ph_st2,&bl,&bms) ;

      /* -------------------- 1.自己ループ  self-loop ---------------------- */

      temp_lh    =   before_log_lh[br]
                   + cal_log((double)a[ph][ph_st1][ph_st1])
                   + cal_log((double)b[ph][ph_st1])        ;

      temp_reco_word   = wn ;
      temp_word_state  = bws ;
      temp_word_length = bl ;
      temp_model_state = bms ;
      word_change_sw   = 0 ;

      cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;

      /* -------------------- 2.遷移 transit         ---------------------- */

      /* -------------------- 2.1 同一単語内を遷移 transit to other state in same word -------- */

      if(bws < number_of_state[wn]-1) {

        if(ph_st2==0) ph_st2 = MAX_STATE ;

        temp_lh    =   before_log_lh[br]
                     + cal_log((double)a[ph][ph_st1][ph_st2])
                     + cal_log((double)b[ph][ph_st1])        ;

        temp_reco_word   = wn ;
        temp_word_state  = bws+1 ;
        temp_word_length = bl ;
        temp_model_state = bms ;
        word_change_sw   = 0 ;

        cheack_lh(br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw) ;
      }

      else {

        /* -------------------- 2.2 他の単語へ遷移   transit to other word   ---------------------- */

        if(wn!=PAUSE) {

          /* -------------------- 2.2.1 ポーズ以外から遷移 transit from word without pause -------- */

          for(wn2=0;wn2<MAX_WORD;wn2++) {
            if(wn2!=PAUSE) {


              /* -------------------- 2.2.1.2 ポーズ以外からポーズ以外へ遷移 transit from word to pause --- */

              max_connection_pr = MIN_DOUBLE ;
              max_connection_st = 0 ;
              for(i=0;i<STATE_NUMBER;i++) {
                temp_connection_pr = model_a[bms][i] * model_b[bms][i][word_number[wn2]] ;
                if(temp_connection_pr>max_connection_pr) {
                  max_connection_pr = temp_connection_pr ;
                  max_connection_st = i ;
                }
              }
              log_connection_pr = cal_log(max_connection_pr) ;

              temp_lh    =   before_log_lh[br]
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                        <sort_probability.c>

                       sort recognition result

                       認識結果のソーティング

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
  -------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include <strings.h>

int sort_probability()
{
  int i,max_br,rn,br ;
  double max;
  double temp_prob[NUMBER_OF_RANK] ;

/*################################*/
/*        INITIALIZE              */
/*################################*/

  for(rn=0;rn<NUMBER_OF_RANK;rn++) {
    sorted_label_number[rn] = -1  ;
    temp_prob[rn]           = 0.0 ;
  }

/*################################*/
/*            SORT                */
/*################################*/

  for(rn=0;rn<NUMBER_OF_RANK;rn++) {
    max    = MIN_DOUBLE ;
    max_br = -1         ;

    for(br=0;br<BEAM_RANGE;br++) {
      if((log_lh[br])>max) {
        max    = log_lh[br] ;
        max_br = br         ;
      }
    }
    if(max_br != -1) {
      temp_prob[rn]           = log_lh[max_br] ;
      log_lh[max_br]          = MIN_DOUBLE     ;
      sorted_label_number[rn] = max_br         ;
    }
    else {
      sorted_label_number[rn] = -1 ;
    }
  }

  for(rn=0;rn<NUMBER_OF_RANK;rn++)
    if(sorted_label_number[rn] != -1)
      log_lh[sorted_label_number[rn]]=temp_prob[rn] ;
}
```

```
LINE #  TEXT
     1  #------------------------------------------------------------------------------
     2  #  SENTENCE RECOGNITION PROGRAM CHELL SPRICT
     3  #
     4  #                                <recog.cs>
     5  #
     6  #           Copyright Hiroki YAMAMOTO          10/27/1992
     7  #
     8  #------------------------------------------------------------------------------
     9  dp  1891   2901    > test_007
    10  dp  2943   3719    > test_008
    11  dp  3760   4558    > test_009
    12  dp  4598   4718    > test_010
    13  dp  4758   5201    > test_011
    14  dp  5242   5814    > test_012
    15  dp  5854   6426    > test_013
    16  dp  6467   6843    > test_014
    17  dp  6884   6938    > test_015
    18  dp  6979   7589    > test_016
    19  dp  7630   7746    > test_017
    20  dp  7787   8594    > test_018
    21  dp  8635   8679    > test_019
    22  dp  8720   9300    > test_020
    23  dp  9340   9387    > test_021
    24  dp  9429   9541    > test_022
    25  dp  9582   9864    > test_023
    26  dp  9905  10252    > test_024
    27  dp 10292  10627    > test_025
    28  dp 10669  10803    > test_026
    29  dp 10845  11073    > test_027
    30  dp 11115  11487    > test_028
    31  dp 11528  11757    > test_029
    32  dp 11798  12301    > test_030
    33  dp 12342  12596    > test_031
    34  dp 12636  14092    > test_032
    35  dp 14132  14422    > test_033
    36  dp 14463  15264    > test_034
    37  dp 15304  15430    > test_035
    38  dp 15471  16472    > test_036
    39  dp 16512  16789    > test_037
    40  dp 16831  16992    > test_038
```

```
LINE #  TEXT
     1  #------------------------------------------------------------------------------
     2  #  SENTENCE RECOGNITION PROGRAM CHELL SPRICT
     3  #
     4  #                          <recog.cs>
     5  #
     6  #          Copyright Hiroki YAMAMOTO          10/27/1992
     7  #
     8  #------------------------------------------------------------------------------
     9
    10  dp    21   130   > test_001
    11  dp   170   220   > test_002
    12  dp   261   544   > test_003
    13  dp   584  1159   > test_004
    14  dp  1201  1762   > test_005
    15  dp  1802  1850   > test_006
    16  dp  1891  2901   > test_007
    17  dp  2943  3719   > test_008
    18  dp  3760  4558   > test_009
    19  dp  4598  4718   > test_010
    20  dp  4758  5201   > test_011
    21  dp  5242  5814   > test_012
    22  dp  5854  6426   > test_013
    23  dp  6467  6843   > test_014
    24  dp  6884  6938   > test_015
    25  dp  6979  7589   > test_016
    26  dp  7630  7746   > test_017
    27  dp  7787  8594   > test_018
    28  dp  8635  8679   > test_019
    29  dp  8720  9300   > test_020
    30  dp  9340  9387   > test_021
    31  dp  9429  9541   > test_022
    32  dp  9582  9864   > test_023
    33  dp  9905 10252   > test_024
    34  dp 10292 10627   > test_025
    35  dp 10669 10803   > test_026
    36  dp 10845 11073   > test_027
    37  dp 11115 11487   > test_028
    38  dp 11528 11757   > test_029
    39  dp 11798 12301   > test_030
    40  dp 12342 12596   > test_031
    41  dp 12636 14092   > test_032
    42  dp 14132 14422   > test_033
    43  dp 14463 15264   > test_034
    44  dp 15304 15430   > test_035
    45  dp 15471 16472   > test_036
    46  dp 16512 16789   > test_037
    47  dp 16831 16992   > test_038
```

LINE # TEXT

```
  if(strcmp(s, "w") == 0)       i=48  ;
  if(strcmp(s, "y") == 0)       i=49  ;
  if(strcmp(s, "z") == 0)       i=50  ;
  if(strcmp(s, "zy") == 0)      i=51  ;
  if(i == 100)
    {
      printf("phoneme name error in phoneme_number\n");
      exit() ;
    }
  return(i) ;
}

/*--------------------------------------

    label number --> label

 --------------------------------------*/

char *ntol(i)
     int i ;

{
  char s[10] ;
  switch(i)
    {
    case  0  :  sprintf(s, "-")   ; break ;
    case  1  :  sprintf(s, "CH1") ; break ;
    case  2  :  sprintf(s, "CH2") ; break ;
    case  3  :  sprintf(s, "H")   ; break ;
    case  4  :  sprintf(s, "K1")  ; break ;
    case  5  :  sprintf(s, "K2")  ; break ;
    case  6  :  sprintf(s, "N")   ; break ;
    case  7  :  sprintf(s, "P1")  ; break ;
    case  8  :  sprintf(s, "P2")  ; break ;
    case  9  :  sprintf(s, "Q1")  ; break ;
    case 10  :  sprintf(s, "S")   ; break ;
    case 11  :  sprintf(s, "SH")  ; break ;
    case 12  :  sprintf(s, "TS1") ; break ;
    case 13  :  sprintf(s, "TS2") ; break ;
    case 14  :  sprintf(s, "Ui")  ; break ;
    case 15  :  sprintf(s, "Uu")  ; break ;
    case 16  :  sprintf(s, "a")   ; break ;
    case 17  :  sprintf(s, "al")  ; break ;
    case 18  :  sprintf(s, "b1")  ; break ;
    case 19  :  sprintf(s, "b2")  ; break ;
    case 20  :  sprintf(s, "by")  ; break ;
    case 21  :  sprintf(s, "cy")  ; break ;
    case 22  :  sprintf(s, "d1")  ; break ;
    case 23  :  sprintf(s, "d2")  ; break ;
    case 24  :  sprintf(s, "e")   ; break ;
    case 25  :  sprintf(s, "el")  ; break ;
    case 26  :  sprintf(s, "g1")  ; break ;
    case 27  :  sprintf(s, "gy1") ; break ;
    case 28  :  sprintf(s, "hy")  ; break ;
    case 29  :  sprintf(s, "i")   ; break ;
    case 30  :  sprintf(s, "il")  ; break ;
    case 31  :  sprintf(s, "ky")  ; break ;
    case 32  :  sprintf(s, "m")   ; break ;
    case 33  :  sprintf(s, "my")  ; break ;
    case 34  :  sprintf(s, "n")   ; break ;
    case 35  :  sprintf(s, "ng")  ; break ;
    case 36  :  sprintf(s, "ngy") ; break ;
    case 37  :  sprintf(s, "ny")  ; break ;
    case 38  :  sprintf(s, "o")   ; break ;
    case 39  :  sprintf(s, "ol")  ; break ;
    case 40  :  sprintf(s, "py")  ; break ;
    case 41  :  sprintf(s, "r")   ; break ;
    case 42  :  sprintf(s, "ry")  ; break ;
    case 43  :  sprintf(s, "sy")  ; break ;
    case 44  :  sprintf(s, "t1")  ; break ;
    case 45  :  sprintf(s, "t2")  ; break ;
    case 46  :  sprintf(s, "u")   ; break ;
    case 47  :  sprintf(s, "ul")  ; break ;
    case 48  :  sprintf(s, "w")   ; break ;
    case 49  :  sprintf(s, "y")   ; break ;
    case 50  :  sprintf(s, "z")   ; break ;
    case 51  :  sprintf(s, "zy")  ; break ;
      default  :  printf("phoneme name error in phoneme_number\n") ; exit() ;
    }
  return(s);
}
```

LINE # TEXT

```
481     if(fp==NULL) {
482       printf("file <%s> is not open in read_word_data \n",WORD_FILE_NAME) ;
483       printf("Now wn = %d \n",wn) ;
484       exit() ;
485     }
486
487     for(i=0;i<word_number[wn];i++)
488       fgets(dummy,3000,fp) ;
489
490     fscanf(fp,"%d %s %s %s %s %d %s %s %s %d",
491           &data,dummy,word_name[wn],dummy,dummy,&part_number[wn],
492           word_form[wn],word_type[wn],dummy,&number_of_phoneme[wn]) ;
493
494
495     if(data!=word_number[wn])
496       printf("something wrong in read_word_data \n") ;
497
498     for(j=0;j<number_of_phoneme[wn];j++) {
499       fscanf(fp,"%s",dum) ;
500       phoneme_label_number[wn][j] = lton(dum) ;
501     }
502
503     fclose(fp) ;
504
505  /*####################*/
506  /* MAKE WORD MODEL  */
507  /*####################*/
508
509     number_of_state[wn] = MAX_STATE * number_of_phoneme[wn] ;
510     md = 0 ;
511     for(st=0;st<number_of_state[wn];st++) {
512       model[wn][st] = phoneme_label_number[wn][md] ;
513       state[wn][st] = (st%3) ;
514       if((st%3)==2) md++ ;
515     }
516   }
517
518  /*####################*/
519  /*   DISPLAY  DATA  */
520  /*####################*/
521  /*
522    printf("\n");
523    for(wn=0;wn<MAX_WORD;wn++) {
524      printf("単語番号 %d 単語名 %s  音素数 %d 状態数 %d\n",
525            word_number[wn],word_name[wn],number_of_phoneme[wn],number_of_state[wn]) ;
526      printf("品詞番号 %d  活用形 %s  活用型 %s \n",
527            part_number[wn],word_form[wn],word_type[wn]) ;
528
529      printf("音素列 ") ;
530      for(j=0;j<number_of_phoneme[wn];j++)
531        printf(" %s ",ntol(phoneme_label_number[wn][j])) ;
532      printf("\n") ;
533
534      for(st=0;st<number_of_state[wn];st++)
535        printf("*** wn:st:  %3d:%3d \n",model[wn][st],state[wn][st]) ;
536
537      printf("\n");
538
539    }
540  */
541  }
542
543  /*--------------------------------------
544
545     label --> label number
546
547   --------------------------------------*/
548
549  int lton(s)
550      char s[10];
551  {
552    int i ;
553    if(strcmp(s, "-") == 0)      i= 0  ;
554    if(strcmp(s, "CH1") == 0)    i= 1  ;
555    if(strcmp(s, "CH2") == 0)    i= 2  ;
556    if(strcmp(s, "H") == 0)      i= 3  ;
557    if(strcmp(s, "K1") == 0)     i= 4  ;
558    if(strcmp(s, "K2") == 0)     i= 5  ;
559    if(strcmp(s, "N") == 0)      i= 6  ;
560    if(strcmp(s, "P1") == 0)     i= 7  ;
561    if(strcmp(s, "P2") == 0)     i= 8  ;
562    if(strcmp(s, "Q1") == 0)     i= 9  ;
563    if(strcmp(s, "S") == 0)      i=10  ;
564    if(strcmp(s, "SH") == 0)     i=11  ;
565    if(strcmp(s, "TS1") == 0)    i=12  ;
566    if(strcmp(s, "TS2") == 0)    i=13  ;
567    if(strcmp(s, "Ui") == 0)     i=14  ;
568    if(strcmp(s, "Uu") == 0)     i=15  ;
569    if(strcmp(s, "a") == 0)      i=16  ;
570    if(strcmp(s, "a1") == 0)     i=17  ;
571    if(strcmp(s, "b1") == 0)     i=18  ;
572    if(strcmp(s, "b2") == 0)     i=19  ;
573    if(strcmp(s, "by") == 0)     i=20  ;
574    if(strcmp(s, "cy") == 0)     i=21  ;
575    if(strcmp(s, "d1") == 0)     i=22  ;
576    if(strcmp(s, "d2") == 0)     i=23  ;
577    if(strcmp(s, "e") == 0)      i=24  ;
578    if(strcmp(s, "e1") == 0)     i=25  ;
579    if(strcmp(s, "g1") == 0)     i=26  ;
580    if(strcmp(s, "gy1") == 0)    i=27  ;
581    if(strcmp(s, "hy") == 0)     i=28  ;
582    if(strcmp(s, "i") == 0)      i=29  ;
583    if(strcmp(s, "i1") == 0)     i=30  ;
584    if(strcmp(s, "ky") == 0)     i=31  ;
585    if(strcmp(s, "m") == 0)      i=32  ;
586    if(strcmp(s, "my") == 0)     i=33  ;
587    if(strcmp(s, "n") == 0)      i=34  ;
588    if(strcmp(s, "ng") == 0)     i=35  ;
589    if(strcmp(s, "ngy") == 0)    i=36  ;
590    if(strcmp(s, "ny") == 0)     i=37  ;
591    if(strcmp(s, "o") == 0)      i=38  ;
592    if(strcmp(s, "o1") == 0)     i=39  ;
593    if(strcmp(s, "py") == 0)     i=40  ;
594    if(strcmp(s, "r") == 0)      i=41  ;
595    if(strcmp(s, "ry") == 0)     i=42  ;
596    if(strcmp(s, "sy") == 0)     i=43  ;
597    if(strcmp(s, "t1") == 0)     i=44  ;
598    if(strcmp(s, "t2") == 0)     i=45  ;
599    if(strcmp(s, "u") == 0)      i=46  ;
600    if(strcmp(s, "u1") == 0)     i=47  ;
```

```
LINE #  TEXT
  361     word_number[ 325]=  1260 ;
  362     word_number[ 326]=  1262 ;
  363     word_number[ 327]=  1263 ;
  364     word_number[ 328]=  1293 ;
  365     word_number[ 329]=  1313 ;
  366     word_number[ 330]=  1315 ;
  367     word_number[ 331]=  1325 ;
  368     word_number[ 332]=  1351 ;
  369     word_number[ 333]=  1372 ;
  370     word_number[ 334]=  1373 ;
  371     word_number[ 335]=  1375 ;
  372     word_number[ 336]=  1382 ;
  373     word_number[ 337]=  1388 ;
  374     word_number[ 338]=  1393 ;
  375     word_number[ 339]=  1406 ;
  376     word_number[ 340]=  1420 ;
  377     word_number[ 341]=  1433 ;
  378     word_number[ 342]=  1441 ;
  379     word_number[ 343]=  1473 ;
  380     word_number[ 344]=  1495 ;
  381     word_number[ 345]=  1526 ;
  382     word_number[ 346]=  1553 ;
  383     word_number[ 347]=  1567 ;
  384     word_number[ 348]=  1600 ;
  385     word_number[ 349]=  1621 ;
  386     word_number[ 350]=  1634 ;
  387     word_number[ 351]=  1650 ;
  388     word_number[ 352]=  1652 ;
  389     word_number[ 353]=  1660 ;
  390     word_number[ 354]=  1704 ;
  391     word_number[ 355]=  1715 ;
  392     word_number[ 356]=  1781 ;
  393     word_number[ 357]=  1785 ;
  394     word_number[ 358]=  1792 ;
  395     word_number[ 359]=  1830 ;
  396     word_number[ 360]=  1846 ;
  397     word_number[ 361]=  1851 ;
  398     word_number[ 362]=  1861 ;
  399     word_number[ 363]=  1863 ;
  400     word_number[ 364]=  1886 ;
  401     word_number[ 365]=  1889 ;
  402     word_number[ 366]=  1914 ;
  403     word_number[ 367]=  2011 ;
  404     word_number[ 368]=  2015 ;
  405     word_number[ 369]=  2016 ;
  406     word_number[ 370]=  2018 ;
  407     word_number[ 371]=  2023 ;
  408     word_number[ 372]=  2028 ;
  409     word_number[ 373]=  2038 ;
  410     word_number[ 374]=  2063 ;
  411     word_number[ 375]=  2071 ;
  412     word_number[ 376]=  2073 ;
  413     word_number[ 377]=  2086 ;
  414     word_number[ 378]=  2089 ;
  415     word_number[ 379]=  2121 ;
  416     word_number[ 380]=  2174 ;
  417     word_number[ 381]=  2184 ;
  418     word_number[ 382]=  2198 ;
  419     word_number[ 383]=  2199 ;
  420     word_number[ 384]=  2214 ;
  421     word_number[ 385]=  2246 ;
  422     word_number[ 386]=  2362 ;
  423     word_number[ 387]=  2367 ;
  424     word_number[ 388]=  2429 ;
  425     word_number[ 389]=  2473 ;
  426     word_number[ 390]=  2502 ;
  427     word_number[ 391]=  2523 ;
  428     word_number[ 392]=  2569 ;
  429     word_number[ 393]=  2571 ;
  430     word_number[ 394]=  2614 ;
  431     word_number[ 395]=  2795 ;
  432     word_number[ 396]=  2837 ;
  433     word_number[ 397]=  2916 ;
  434     word_number[ 398]=  2957 ;
  435     word_number[ 399]=  2992 ;
  436     word_number[ 400]=  3060 ;
  437     word_number[ 401]=  3061 ;
  438     word_number[ 402]=  3080 ;
  439     word_number[ 403]=  3083 ;
  440     word_number[ 404]=  3128 ;
  441     word_number[ 405]=  3139 ;
  442     word_number[ 406]=  3140 ;
  443     word_number[ 407]=  3178 ;
  444     word_number[ 408]=  3264 ;
  445     word_number[ 409]=  3343 ;
  446     word_number[ 410]=  3374 ;
  447     word_number[ 411]=  3440 ;
  448     word_number[ 412]=  3458 ;
  449     word_number[ 413]=  3487 ;
  450     word_number[ 414]=  3631 ;
  451     word_number[ 415]=  3649 ;
  452     word_number[ 416]=  3650 ;
  453     word_number[ 417]=  3657 ;
  454     word_number[ 418]=  3658 ;
  455     word_number[ 419]=  3662 ;
  456     word_number[ 420]=  3663 ;
  457     word_number[ 421]=  3667 ;
  458     word_number[ 422]=  3669 ;
  459     word_number[ 423]=  3679 ;
  460     word_number[ 424]=  3685 ;
  461     word_number[ 425]=  3689 ;
  462     word_number[ 426]=  3693 ;
  463     word_number[ 427]=  3774 ;
  464     word_number[ 428]=  3966 ;
  465     word_number[ 429]=  4133 ;
  466     word_number[ 430]=  4329 ;
  467     word_number[ 431]=  5976 ;
  468     word_number[ 432]=  5979 ;
  469     word_number[ 433]=  6354 ;
  470     word_number[ 434]=  6373 ;
  471
  472
  473   /*####################*/
  474   /*     READ DATA      */
  475   /*####################*/
  476
  477
  478     for(wn=0;wn<MAX_WORD;wn++){
  479
  480       fp=fopen(WORD_FILE_NAME,"r") ;
```

LINE #	TEXT
241	word_number[205]= 394 ;
242	word_number[206]= 399 ;
243	word_number[207]= 402 ;
244	word_number[208]= 403 ;
245	word_number[209]= 408 ;
246	word_number[210]= 409 ;
247	word_number[211]= 413 ;
248	word_number[212]= 415 ;
249	word_number[213]= 416 ;
250	word_number[214]= 422 ;
251	word_number[215]= 425 ;
252	word_number[216]= 428 ;
253	word_number[217]= 431 ;
254	word_number[218]= 434 ;
255	word_number[219]= 443 ;
256	word_number[220]= 445 ;
257	word_number[221]= 446 ;
258	word_number[222]= 447 ;
259	word_number[223]= 448 ;
260	word_number[224]= 456 ;
261	word_number[225]= 463 ;
262	word_number[226]= 470 ;
263	word_number[227]= 473 ;
264	word_number[228]= 474 ;
265	word_number[229]= 479 ;
266	word_number[230]= 482 ;
267	word_number[231]= 487 ;
268	word_number[232]= 488 ;
269	word_number[233]= 490 ;
270	word_number[234]= 492 ;
271	word_number[235]= 493 ;
272	word_number[236]= 503 ;
273	word_number[237]= 505 ;
274	word_number[238]= 507 ;
275	word_number[239]= 513 ;
276	word_number[240]= 516 ;
277	word_number[241]= 522 ;
278	word_number[242]= 525 ;
279	word_number[243]= 537 ;
280	word_number[244]= 538 ;
281	word_number[245]= 539 ;
282	word_number[246]= 542 ;
283	word_number[247]= 545 ;
284	word_number[248]= 547 ;
285	word_number[249]= 566 ;
286	word_number[250]= 568 ;
287	word_number[251]= 583 ;
288	word_number[252]= 588 ;
289	word_number[253]= 590 ;
290	word_number[254]= 594 ;
291	word_number[255]= 625 ;
292	word_number[256]= 626 ;
293	word_number[257]= 630 ;
294	word_number[258]= 631 ;
295	word_number[259]= 639 ;
296	word_number[260]= 647 ;
297	word_number[261]= 648 ;
298	word_number[262]= 650 ;
299	word_number[263]= 669 ;
300	word_number[264]= 681 ;
301	word_number[265]= 683 ;
302	word_number[266]= 688 ;
303	word_number[267]= 694 ;
304	word_number[268]= 716 ;
305	word_number[269]= 717 ;
306	word_number[270]= 739 ;
307	word_number[271]= 747 ;
308	word_number[272]= 749 ;
309	word_number[273]= 751 ;
310	word_number[274]= 767 ;
311	word_number[275]= 776 ;
312	word_number[276]= 794 ;
313	word_number[277]= 806 ;
314	word_number[278]= 807 ;
315	word_number[279]= 822 ;
316	word_number[280]= 828 ;
317	word_number[281]= 830 ;
318	word_number[282]= 833 ;
319	word_number[283]= 834 ;
320	word_number[284]= 866 ;
321	word_number[285]= 867 ;
322	word_number[286]= 874 ;
323	word_number[287]= 875 ;
324	word_number[288]= 881 ;
325	word_number[289]= 890 ;
326	word_number[290]= 892 ;
327	word_number[291]= 893 ;
328	word_number[292]= 901 ;
329	word_number[293]= 903 ;
330	word_number[294]= 908 ;
331	word_number[295]= 909 ;
332	word_number[296]= 916 ;
333	word_number[297]= 922 ;
334	word_number[298]= 929 ;
335	word_number[299]= 943 ;
336	word_number[300]= 944 ;
337	word_number[301]= 946 ;
338	word_number[302]= 949 ;
339	word_number[303]= 963 ;
340	word_number[304]= 972 ;
341	word_number[305]= 994 ;
342	word_number[306]= 998 ;
343	word_number[307]= 1008 ;
344	word_number[308]= 1012 ;
345	word_number[309]= 1013 ;
346	word_number[310]= 1021 ;
347	word_number[311]= 1055 ;
348	word_number[312]= 1061 ;
349	word_number[313]= 1069 ;
350	word_number[314]= 1090 ;
351	word_number[315]= 1092 ;
352	word_number[316]= 1094 ;
353	word_number[317]= 1107 ;
354	word_number[318]= 1121 ;
355	word_number[319]= 1150 ;
356	word_number[320]= 1153 ;
357	word_number[321]= 1199 ;
358	word_number[322]= 1212 ;
359	word_number[323]= 1221 ;
360	word_number[324]= 1251 ;

```
LINE #  TEXT
   121  word_number[  85]=  111 ;
   122  word_number[  86]=  112 ;
   123  word_number[  87]=  113 ;
   124  word_number[  88]=  114 ;
   125  word_number[  89]=  115 ;
   126  word_number[  90]=  117 ;
   127  word_number[  91]=  120 ;
   128  word_number[  92]=  121 ;
   129  word_number[  93]=  122 ;
   130  word_number[  94]=  125 ;
   131  word_number[  95]=  127 ;
   132  word_number[  96]=  129 ;
   133  word_number[  97]=  131 ;
   134  word_number[  98]=  132 ;
   135  word_number[  99]=  134 ;
   136  word_number[ 100]=  136 ;
   137  word_number[ 101]=  137 ;
   138  word_number[ 102]=  138 ;
   139  word_number[ 103]=  141 ;
   140  word_number[ 104]=  143 ;
   141  word_number[ 105]=  145 ;
   142  word_number[ 106]=  147 ;
   143  word_number[ 107]=  150 ;
   144  word_number[ 108]=  151 ;
   145  word_number[ 109]=  152 ;
   146  word_number[ 110]=  153 ;
   147  word_number[ 111]=  154 ;
   148  word_number[ 112]=  156 ;
   149  word_number[ 113]=  157 ;
   150  word_number[ 114]=  158 ;
   151  word_number[ 115]=  161 ;
   152  word_number[ 116]=  162 ;
   153  word_number[ 117]=  163 ;
   154  word_number[ 118]=  164 ;
   155  word_number[ 119]=  166 ;
   156  word_number[ 120]=  167 ;
   157  word_number[ 121]=  168 ;
   158  word_number[ 122]=  169 ;
   159  word_number[ 123]=  170 ;
   160  word_number[ 124]=  171 ;
   161  word_number[ 125]=  172 ;
   162  word_number[ 126]=  175 ;
   163  word_number[ 127]=  179 ;
   164  word_number[ 128]=  180 ;
   165  word_number[ 129]=  181 ;
   166  word_number[ 130]=  182 ;
   167  word_number[ 131]=  184 ;
   168  word_number[ 132]=  185 ;
   169  word_number[ 133]=  186 ;
   170  word_number[ 134]=  187 ;
   171  word_number[ 135]=  190 ;
   172  word_number[ 136]=  192 ;
   173  word_number[ 137]=  194 ;
   174  word_number[ 138]=  195 ;
   175  word_number[ 139]=  196 ;
   176  word_number[ 140]=  197 ;
   177  word_number[ 141]=  198 ;
   178  word_number[ 142]=  202 ;
   179  word_number[ 143]=  206 ;
   180  word_number[ 144]=  207 ;
   181  word_number[ 145]=  208 ;
   182  word_number[ 146]=  209 ;
   183  word_number[ 147]=  211 ;
   184  word_number[ 148]=  215 ;
   185  word_number[ 149]=  216 ;
   186  word_number[ 150]=  219 ;
   187  word_number[ 151]=  225 ;
   188  word_number[ 152]=  226 ;
   189  word_number[ 153]=  227 ;
   190  word_number[ 154]=  228 ;
   191  word_number[ 155]=  232 ;
   192  word_number[ 156]=  233 ;
   193  word_number[ 157]=  234 ;
   194  word_number[ 158]=  236 ;
   195  word_number[ 159]=  238 ;
   196  word_number[ 160]=  240 ;
   197  word_number[ 161]=  242 ;
   198  word_number[ 162]=  244 ;
   199  word_number[ 163]=  251 ;
   200  word_number[ 164]=  256 ;
   201  word_number[ 165]=  257 ;
   202  word_number[ 166]=  258 ;
   203  word_number[ 167]=  259 ;
   204  word_number[ 168]=  262 ;
   205  word_number[ 169]=  264 ;
   206  word_number[ 170]=  265 ;
   207  word_number[ 171]=  271 ;
   208  word_number[ 172]=  272 ;
   209  word_number[ 173]=  278 ;
   210  word_number[ 174]=  279 ;
   211  word_number[ 175]=  283 ;
   212  word_number[ 176]=  284 ;
   213  word_number[ 177]=  286 ;
   214  word_number[ 178]=  290 ;
   215  word_number[ 179]=  291 ;
   216  word_number[ 180]=  297 ;
   217  word_number[ 181]=  299 ;
   218  word_number[ 182]=  300 ;
   219  word_number[ 183]=  303 ;
   220  word_number[ 184]=  308 ;
   221  word_number[ 185]=  310 ;
   222  word_number[ 186]=  311 ;
   223  word_number[ 187]=  314 ;
   224  word_number[ 188]=  316 ;
   225  word_number[ 189]=  318 ;
   226  word_number[ 190]=  321 ;
   227  word_number[ 191]=  324 ;
   228  word_number[ 192]=  328 ;
   229  word_number[ 193]=  332 ;
   230  word_number[ 194]=  335 ;
   231  word_number[ 195]=  340 ;
   232  word_number[ 196]=  349 ;
   233  word_number[ 197]=  351 ;
   234  word_number[ 198]=  364 ;
   235  word_number[ 199]=  366 ;
   236  word_number[ 200]=  369 ;
   237  word_number[ 201]=  374 ;
   238  word_number[ 202]=  376 ;
   239  word_number[ 203]=  383 ;
   240  word_number[ 204]=  386 ;
```

LINE # TEXT

```
/*-------------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <read_word_data.c>

              read word data from dictionary & make word HMM

              辞書から単語のデータ読み込み＆単語ＨＭＭを作る

           Copyright Hiroki YAMAMOTO          10/27/1992

                                                    called from <dp.c>
  -------------------------------------------------------------------------------*/
#include <math.h>
#include <stdio.h>
#include <strings.h>
#include "parameter.h"

read_word_data()
{
  int i,j,k,wn,st,md ;
  int data ;
  int lton() ;
  char a,b,c[1000] ;
  char *ntol() ;
  char dummy[3000],dum[10] ;
  FILE *fp ;

  for(wn=0;wn<MAX_WORD;wn++)
    word_number[wn] = 0 ;

/*###################*/
/* SET WORD NUMBER */
/*###################*/

  word_number[   0]=     0 ;
  word_number[   1]=     7 ;
  word_number[   2]=     8 ;
  word_number[   3]=     9 ;
  word_number[   4]=    10 ;
  word_number[   5]=    11 ;
  word_number[   6]=    12 ;
  word_number[   7]=    13 ;
  word_number[   8]=    14 ;
  word_number[   9]=    16 ;
  word_number[  10]=    17 ;
  word_number[  11]=    18 ;
  word_number[  12]=    19 ;
  word_number[  13]=    20 ;
  word_number[  14]=    21 ;
  word_number[  15]=    22 ;
  word_number[  16]=    25 ;
  word_number[  17]=    26 ;
  word_number[  18]=    27 ;
  word_number[  19]=    29 ;
  word_number[  20]=    30 ;
  word_number[  21]=    31 ;
  word_number[  22]=    32 ;
  word_number[  23]=    33 ;
  word_number[  24]=    34 ;
  word_number[  25]=    37 ;
  word_number[  26]=    38 ;
  word_number[  27]=    39 ;
  word_number[  28]=    40 ;
  word_number[  29]=    41 ;
  word_number[  30]=    42 ;
  word_number[  31]=    43 ;
  word_number[  32]=    44 ;
  word_number[  33]=    45 ;
  word_number[  34]=    46 ;
  word_number[  35]=    47 ;
  word_number[  36]=    48 ;
  word_number[  37]=    49 ;
  word_number[  38]=    50 ;
  word_number[  39]=    51 ;
  word_number[  40]=    52 ;
  word_number[  41]=    53 ;
  word_number[  42]=    55 ;
  word_number[  43]=    56 ;
  word_number[  44]=    57 ;
  word_number[  45]=    58 ;
  word_number[  46]=    60 ;
  word_number[  47]=    61 ;
  word_number[  48]=    62 ;
  word_number[  49]=    63 ;
  word_number[  50]=    64 ;
  word_number[  51]=    65 ;
  word_number[  52]=    66 ;
  word_number[  53]=    67 ;
  word_number[  54]=    68 ;
  word_number[  55]=    69 ;
  word_number[  56]=    70 ;
  word_number[  57]=    71 ;
  word_number[  58]=    72 ;
  word_number[  59]=    74 ;
  word_number[  60]=    75 ;
  word_number[  61]=    76 ;
  word_number[  62]=    77 ;
  word_number[  63]=    78 ;
  word_number[  64]=    79 ;
  word_number[  65]=    81 ;
  word_number[  66]=    82 ;
  word_number[  67]=    83 ;
  word_number[  68]=    84 ;
  word_number[  69]=    85 ;
  word_number[  70]=    86 ;
  word_number[  71]=    87 ;
  word_number[  72]=    88 ;
  word_number[  73]=    89 ;
  word_number[  74]=    93 ;
  word_number[  75]=    94 ;
  word_number[  76]=    95 ;
  word_number[  77]=    96 ;
  word_number[  78]=    97 ;
  word_number[  79]=   101 ;
  word_number[  80]=   103 ;
  word_number[  81]=   105 ;
  word_number[  82]=   107 ;
  word_number[  83]=   108 ;
  word_number[  84]=   110 ;
```

LINE # | TEXT

```
121
122  /*****************************/
123  /* READ DELTA_CEPSTRUM DATA */
124  /*****************************/
125
126    sprintf(data_file_name,"%s%s",DIR_MAU,DELTA_CEPSTRUM);
127    fp=fopen(data_file_name,"r") ;
128    if(fp==NULL) {
129      printf("File <%s> is not opened in main \n",data_file_name) ;
130      fclose(fp) ;
131      exit() ;
132    }
133
134    start_point = DATA_SIZE * NUMBER_OF_CEPSTRUM * fn ;
135    number_of_data = NUMBER_OF_CEPSTRUM ;
136
137    fseek(fp,start_point,0) ;
138    fread(data,DATA_SIZE,number_of_data,fp);
139    fclose(fp);
140
141  /*****************************/
142  /* WRITE DELTA CEPSTRUM DATA */
143  /*****************************/
144
145    for(i=0;i<NUMBER_OF_CEPSTRUM;i++)
146      p_data[2*NUMBER_OF_POWER+NUMBER_OF_CEPSTRUM+i]=data[i] ;
147
148  }
149
150
151
```

LINE # TEXT

```
/*-------------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <read_phoneme_data.c>

                            read phoneme data

                       発声された音声の音響分析値の読み込み

            Copyright Hiroki YAMAMOTO          10/27/1992

                                                          called from <dp.c>
  -------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "parameter.h"

read_phoneme_data(fn,p_data)
     long int fn ;
     float p_data[DIM_OF_VECTOR] ;
{
  int i,j,v ;
  long start_point ;
  int number_of_data ;
  char data_file_name[100] ;
  float data[10000],b,c ;
  FILE *fp ;

/*---------------------------------------*/
/* INITIALIZE                            */
/*---------------------------------------*/

  for(v=0;v<DIM_OF_VECTOR;v++)
    p_data[v] = 0.0 ;

/*-------------------------------------------------------------------------------*/

/*#######################*/
/* READ LOG POWER DATA */
/*#######################*/

      sprintf(data_file_name,"%s%s",DIR_MAU,LOG_POWER);
      fp=fopen(data_file_name,"r") ;
      if(fp==NULL) {
        printf("File <%s> is not opened in main \n",data_file_name) ;
        fclose(fp) ;
        exit() ;
      }

  start_point = DATA_SIZE * fn ;
  number_of_data = NUMBER_OF_POWER ;

  fseek(fp,start_point,0) ;
  fread(data,DATA_SIZE,number_of_data,fp);
  fclose(fp);

/*########################*/
/* WRITE LOG POWER DATA */
/*########################*/

  for(j=0;j<NUMBER_OF_POWER;j++)
    p_data[j]=data[j] ;

/*-------------------------------------------------------------------------------*/

/*######################*/
/* READ CEPSTRUM DATA */
/*######################*/

  sprintf(data_file_name,"%s%s",DIR_MAU,CEPSTRUM);
  fp=fopen(data_file_name,"r") ;
  if(fp==NULL) {
    printf("File <%s> is not opened in main \n",data_file_name) ;
    fclose(fp) ;
    exit() ;
  }

  start_point= DATA_SIZE * NUMBER_OF_CEPSTRUM * fn ;
  number_of_data = NUMBER_OF_CEPSTRUM ;

  fseek(fp,start_point,0) ;
  fread(data,DATA_SIZE,number_of_data,fp);
  fclose(fp);

/*######################*/
/* WRITE CEPSTRUM DATA */
/*######################*/

  for(j=0;j<NUMBER_OF_CEPSTRUM;j++)
    p_data[j+NUMBER_OF_POWER]=data[j] ;

/*-------------------------------------------------------------------------------*/

/*########################*/
/* READ DELTA POWER DATA */
/*########################*/

  sprintf(data_file_name,"%s%s",DIR_MAU,DELTA_POWER);
  fp=fopen(data_file_name,"r") ;
  if(fp==NULL) {
    printf("File <%s> is not opened in main \n",data_file_name) ;
    fclose(fp) ;
    exit() ;
  }

  start_point=DATA_SIZE*fn ;
  number_of_data=NUMBER_OF_POWER ;

  fseek(fp,start_point,0) ;
  fread(data,DATA_SIZE,number_of_data,fp);
  fclose(fp);

/*#######################*/
/* WRITE LOG POWER DATA */
/*#######################*/

  for(j=0;j<NUMBER_OF_POWER;j++)
    p_data[j+NUMBER_OF_POWER+NUMBER_OF_CEPSTRUM]=data[j] ;

/*-------------------------------------------------------------------------------*/
```

LINE # | TEXT

```
    for(i=0;i<MAX_STATE;i++) {
      for(j=i;j<=(i+1);j++) {
        fscanf(fp,"%s",dummy) ;
        fscanf(fp,"%s",dummy) ;
        fscanf(fp,"%f",&data) ;
        a[ln][i][j]=data ;
        fscanf(fp,"%s",dummy) ;
      }
    }
    fclose(fp) ;
  }
}
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                    <read_phoneme_HMM_data.c>

                     read phoneme HMM data

                  音素ＨＭＭのパラメータ読み込み

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                    called from <dp.c>
-------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <strings.h>
#include "parameter.h"


int read_phoneme_HMM_data()
{
  int   ln,i,j,k,data_int      ;
  int   st,mx,v                ;
  float data                   ;
  char  HMM_data_file_name[100];
  char  dummy[20]              ;
  FILE  *fp                    ;

/*##############*/
/* INITIALIZE */
/*##############*/

  data = 0.0 ;
  for(ln=0;ln<MAX_LABEL;ln++) {
    number_of_mixture[ln] = 0 ;
    sprintf(phoneme[ln], "NOTHING") ;
  }
  for(ln=0;ln<MAX_LABEL;ln++) {
    for(i=0;i<MAX_STATE;i++) {
      for(j=0;j<=MAX_STATE;j++)
        a[ln][i][j]=0.0 ;

      for(j=0;j<MAX_NUMBER_OF_MIXTURE;j++) {
        height_of_distribution[ln][i][j]=0.0 ;
        for(k=0;k<DIM_OF_VECTOR;k++) {
          expectation[ln][i][j][k]=0.0 ;
          variance[ln][i][j][k]=0.0 ;
        }
      }
    }
  }

/*########################*/
/* READ LABEL FROM FILE */
/*########################*/

  sprintf(HMM_data_file_name,"%slabel",HMM_data_dir_name) ;

  fp=fopen(HMM_data_file_name,"r") ;
  if(fp==NULL) {
    printf("File <%s> is not opened at READ LABEL FROM FILE in read HMM data \n",HMM_data_file_name) ;
    fclose(fp) ;
    exit() ;
  }

  for(i=0;i<MAX_LABEL;i++)
    fscanf(fp,"%s",phoneme[i]) ;

  fclose(fp) ;

/*#########################*/
/* READ HMM DATA FROM FILE */
/*#########################*/

  for(ln=0;ln<MAX_LABEL;ln++) {
    sprintf(HMM_data_file_name,"%shmm_%s",HMM_data_dir_name,phoneme[ln]) ;

    fp=fopen(HMM_data_file_name,"r") ;
    if(fp==NULL) {
      printf("File <%s> is not opened in read_HMM_data",HMM_data_file_name) ;
      exit() ;
    }

    for(i=0;i<5;i++)
      fscanf(fp,"%s",dummy) ;

/*#########################*/
/* READ number_of_mixture  */
/*#########################*/

    fscanf(fp,"%d",&data_int) ;
    number_of_mixture[ln] = data_int ;


    for(i=0;i<2;i++)
      fscanf(fp,"%s",dummy) ;

/*#########################################*/
/* READ expectation[ln][st][mx][v]         */
/*      variance[ln][st][mx][v]            */
/*      height_od_distribution[ln][st][mx] */
/*#########################################*/

    for(st=0;st<MAX_STATE;st++) {
      for(mx=0;mx<number_of_mixture[ln];mx++) {
        fscanf(fp,"%s",dummy) ;
        fscanf(fp,"%f",&data) ;
        height_of_distribution[ln][st][mx]=data ;
        for(v=0;v<DIM_OF_VECTOR;v++) {
          fscanf(fp,"%f",&data) ;
          expectation[ln][st][mx][v]=data ;
          fscanf(fp,"%f",&data) ;
          variance[ln][st][mx][v]=data ;
        }
      }
    }

    for(i=0;i<6;i++)
      fscanf(fp,"%s",dummy) ;
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                        <read_language_HMM_data.c>

                   read parameter of language HMM

                   言語ＨＭＭのパラメータ読み込み

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
  -------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "parameter.h"

int read_language_HMM_data()
{

  int i,j,k ;
  char file_name[100] ;
  FILE *fd ;

/* INITIALIZE */

  for(i=0;i<STATE_NUMBER;i++) {
    model_pi[i] =0.0 ;
    for(j=0;j<STATE_NUMBER;j++) {
      model_a[i][j] = 0.0 ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        model_b[i][j][k] = 0.0 ;
    }
  }

/* read language HMM data */

  sprintf(file_name,"%s%s",MODEL_DIR,FILE_A) ;
  fd=fopen(file_name,"r");
  if(fd==NULL) {
    printf("File <%s> is not opened in read_HMM_word_model \n",file_name) ;
    fclose(fd) ;
    exit() ;
  }
  fread(model_a,8,STATE_NUMBER*STATE_NUMBER,fd);
  fclose(fd);

  sprintf(file_name,"%s%s",MODEL_DIR,FILE_B) ;
  fd=fopen(file_name,"r");
  if(fd==NULL) {
    printf("File <%s> is not opened in read_HMM_word_model \n",file_name) ;
    fclose(fd) ;
    exit() ;
  }
  fread(model_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd);
  fclose(fd);

  sprintf(file_name,"%s%s",MODEL_DIR,FILE_PI) ;
  fd=fopen(file_name,"r");
  if(fd==NULL) {
    printf("File <%s> is not opened in read_HMM_word_model \n",file_name) ;
    fclose(fd) ;
    exit() ;
  }
  fread(model_pi,8,STATE_NUMBER,fd);
  fclose(fd);
}
```

LINE # TEXT

```
/*------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM ORIGINAL HEADER FILE

                      <parameter.h>

                  main parameter & constant number

                  連続文認識プログラムに用いた変数・定数

          Copyright Hiroki YAMAMOTO          10/27/1992

 ------------------------------------------------------------------------*/
#define MIN_DOUBLE -9.9999999999e99             /* double の最低値                */
#define MAX_DOUBLE  9.9999999999e200            /* double の最大値                */
#define MIN_LOG     1.0e-300                    /* log を計算できる最小値         */

#define BEAM_RANGE      4000                     /* ビームサーチのビーム幅          */
#define TEMP_BEAM_RANGE 8627                     /* 一時的に計算する尤度の数( = BEAM_RANGE*2 + MAX_WORD) */
#define LANGUAGE_WEIGHT 16.0                     /* 言語モデルの情報に掛ける重み    */

/* 言語モデル用定数  (fixed number of HMM LANGUAGE MODEL)*/
#define STATE_NUMBER    8                       /* 言語モデルのＨＭＭの状態数      */
#define SYMBOL_NUMBER 6420                      /* 言語モデルのＨＭＭのシンボル数  */
#define MODEL_DIR "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000+38.nf/" /* 言語ＨＭＭのデータのある directory */
#define FILE_A  "test.a"                        /* 状態遷移確率のデータファイル    */
#define FILE_B  "test.b"                        /* シンボル出力確率のデータファイル */
#define FILE_PI "test.pi"                       /* 初期状態確率のデータファイル    */



/* 音素データ  (fixed number of phoneme data) */
#define DIR_MAU "/NFS/atrq12/q12/users/xyama/rec_data/MAU_M01.5mS."     /* 音素データのある directory */
#define LOG_POWER "lpow"                        /* file name of log-power      */
#define CEPSTRUM "cep16"                        /* file name of cepstrum       */
#define DELTA_POWER "dpow"                      /* file name of delta-power    */
#define DELTA_CEPSTRUM "dcep16"                 /* file name of delta-cepstrum */
#define DATA_SIZE 4                             /* data size                   */

#define NUMBER_OF_POWER 1                       /* dimension of power data     */
#define NUMBER_OF_CEPSTRUM 16                   /* dimension of cepstrum data  */
#define DIM_OF_VECTOR 34                        /* 2*(NUMBER_OF_POWER+NUMBER_OF_CEPSTRUM) */

/* 音素ＨＭＭモデル (fixed number of phoneme-HMM-model) */
#define HMM_data_dir_name "/NFS/atrq12/q12/users/xyama/bigram/MAU/"  /* phoneme-HMM-model data directory */
#define MAX_LABEL 52                                                 /* number of phoneme-label     */
#define MAX_STATE 3                                                  /* number of state phoneme-HMM-model */
#define MAX_NUMBER_OF_MIXTURE 20                                     /* max number of mixture */

#define NUMBER_OF_RANK 5                                             /* 表示する尤度の順位制限    */

/* 単語ＨＭＭモデル作成用定数 (fixed number for making word-HMM-model) */
#define WORD_FILE_NAME "/NFS/atrq12/q12/users/xyama/bigram/dic1" /* 単語辞書ファイル名 file name of word-dictionary */
#define MAX_NUMBER_OF_PHONEME 60                                 /* max number of phoneme for one ward-HMM-model */
#define MAX_WORD          435                                    /* max number of word-model */
#define MAX_NUMBER_OF_WORD 20                                    /* max number of recoginized word */

/* 言語モデル用変数 (variable number of L-MODEL) */
double  model_a[STATE_NUMBER][STATE_NUMBER];                     /* state transition probability */
double  model_b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];      /* output probability */
double  model_pi[STATE_NUMBER];                                  /* initial state distribution */

double before_log_lh[BEAM_RANGE] ;
double log_lh[TEMP_BEAM_RANGE] ;
int before_word[BEAM_RANGE][MAX_NUMBER_OF_WORD] ;
int reco_word[TEMP_BEAM_RANGE][MAX_NUMBER_OF_WORD] ;
int word_state[TEMP_BEAM_RANGE] ;
int before_word_state[BEAM_RANGE] ;
int word_length[TEMP_BEAM_RANGE] ;
int before_length[BEAM_RANGE] ;
int model_state[TEMP_BEAM_RANGE] ;;
int before_model_state[BEAM_RANGE] ;;
int log_lh_pointer[TEMP_BEAM_RANGE] ;

int BEAM_NUM ;
int PAUSE ;

int model[MAX_WORD][MAX_STATE*MAX_NUMBER_OF_PHONEME] ;
int state[MAX_WORD][MAX_STATE*MAX_NUMBER_OF_PHONEME] ;


int number_of_mixture[MAX_LABEL] ;
int sorted_label_number[NUMBER_OF_RANK] ;

float a[MAX_LABEL][MAX_STATE+1][MAX_STATE+1] ;
double b[MAX_LABEL][MAX_STATE+1] ;

int word_number[MAX_WORD] ;
int number_of_state[MAX_WORD] ;
int number_of_phoneme[MAX_WORD] ;
char word_name[MAX_WORD][100] ;
int  part_number[MAX_WORD] ;
char word_form[MAX_WORD][5] ;
char word_type[MAX_WORD][5] ;
phoneme_label_number[MAX_WORD][MAX_NUMBER_OF_PHONEME] ;


float height_of_distribution[MAX_LABEL][MAX_STATE][MAX_NUMBER_OF_MIXTURE] ;
float expectation[MAX_LABEL][MAX_STATE][MAX_NUMBER_OF_MIXTURE][DIM_OF_VECTOR] ;
float variance[MAX_LABEL][MAX_STATE][MAX_NUMBER_OF_MIXTURE][DIM_OF_VECTOR] ;

char phoneme[MAX_LABEL][10] ;
```

LINE # TEXT

```
#-------------------------------------------------------------------------------
#  SENTENCE RECOGNITION PROGRAM MAKE-FILE
#
#                          <makefile>
#
#           Copyright Hiroki YAMAMOTO        10/27/1992
#
#-------------------------------------------------------------------------------

CC      = cc
CFLAGS  = -g

MAIN    = dp
SUBR    = read_phoneme_data.c ctoi.c read_phoneme_HMM_data.c calc_b.c viterbi.c viterbi_fr_0.c viterbi_fr_last.c\
          sort_probability.c read_word_data.c read_language_HMM_data.c cal_data.c cal_log.c check_lh.c \
          init_parameter.c
SUBR2   = read_phoneme_data.o ctoi.o read_phoneme_HMM_data.o calc_b.o viterbi.o viterbi_fr_0.o viterbi_fr_last.o\
          sort_probability.o read_word_data.o read_language_HMM_data.o cal_data.o cal_log.o check_lh.o \
          init_parameter.o
LIB     = -lm
#-lLPC#


$(MAIN): $(MAIN).c
        $(CC) $(CFLAGS) $(MAIN).c $(SUBR) $(LIB) -o $(MAIN)

$(MAIN).o $(SUBR2): parameter.h
```

LINE # TEXT

```
        *(log_lh+ii)=large ;
        log_lh_pointer[ii] = large_pointer ;
      }
      else {
        hantei=0 ;
        if(*(log_lh+ii)<st) ml=ii-1  ;
        else ml=ii ;
      }
    }

    *start_beam_num=kk;
    ii-- ;
  }

  m2=ml+1 ;


  if( ( *(&ss) <= BEAM_RANGE -1 ) && ( BEAM_RANGE -1 <= *(&ml)) ) {
    quick(&ss,&ml);
  }
  if( ( *(&m2) <= BEAM_RANGE -1 ) && ( BEAM_RANGE -1 <= *(&ee)) ) {
    quick(&m2,&ee) ;
  }
}
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <init_parameter.c>

                sort & initialize parameter of each state

                  one pass DP で必要なパラメータのソート・更新

          Copyright Hiroki YAMAMOTO          10/27/1992

                                                        called from <dp.c>
-------------------------------------------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <strings.h>
# include "parameter.h"

int  init_parameter()
{

  int  j,br,start_beam_num,end_beam_num,i ;

  for(br=0;br<BEAM_NUM;br++) {
    log_lh_pointer[br]=br ;
  }

/*
  printf("更新前のパラメータ\n") ;
  for(br=0;br<TEMP_BEAM_RANGE;br++) {
    printf("log_lh[%d]= %e log_lh_poiter = %d ",br,log_lh[br],log_lh_pointer[br]) ;
    printf(" ws = %2d ms = %d wl = %d ",word_state[br],model_state[br],word_length[br]) ;
    for(j=0;j<MAX_NUMBER_OF_WORD;j++) {
      if(reco_word[br][j]>=0)
        printf("%d %5s ",reco_word[br][j],word_name[reco_word[br][j]]);
    }
    printf("\n") ;
  }
  printf("\n") ;
*/
  start_beam_num = 0 ;
  end_beam_num   = BEAM_NUM-1 ;

  quick(&start_beam_num,&end_beam_num) ;                    /*  クイックソート */


  for(br=0;br<BEAM_RANGE;br++) {
    before_log_lh[br]        =  log_lh[br]      ;
    before_length[br]        =  word_length[log_lh_pointer[br]] ;
    before_word_state[br]    =  word_state[log_lh_pointer[br]]  ;
    before_model_state[br]   =  model_state[log_lh_pointer[br]] ;
    for(j=0;j<MAX_NUMBER_OF_WORD;j++)
      before_word[br][j] = reco_word[log_lh_pointer[br]][j]     ;
  }
}

/* 以下 デバッグ用 */
/* printf("更新後のパラメータ\n") ;
  for(br=0;br<BEAM_RANGE;br++) {
    before_log_lh[br]        =  log_lh[br]                    ;
    before_length[br]        =  word_length[log_lh_pointer[br]] ;
    before_word_state[br]    =  word_state[log_lh_pointer[br]]  ;
    before_model_state[br]   =  model_state[log_lh_pointer[br]] ;
    printf("log_lh[%d]= %e log_lh_poiter = %d ",br,before_log_lh[br],log_lh_pointer[br]) ;
    printf(" ws = %2d ms = %d wl = %d ",before_word_state[br],before_model_state[br],before_length[br]) ;
    for(j=0;j<MAX_NUMBER_OF_WORD;j++) {
      before_word[br][j] = reco_word[log_lh_pointer[br]][j]     ;
      if(before_word[br][j]>=0)
        printf("%d %5s ",before_word[br][j],word_name[before_word[br][j]]);
    }
    printf("\n") ;
  }
  printf("\n") ;
}
*/



quick(start_beam_num,end_beam_num)

    int *start_beam_num,*end_beam_num ;
{
  int    i,ii,jj,kk,ll,m1,m2,ss,ee,small_pointer,large_pointer,hantei ;
  double small,large,st ;
  if(*end_beam_num<= *start_beam_num) return(0) ;
  hantei = 1 ;
  st= *(log_lh + *start_beam_num) ;
  ii=ee= *end_beam_num ;
  kk=ss= *start_beam_num ;

  while(hantei) {

    while((st>= *(log_lh+ii))&&(ii>kk)) ii-- ;
    if(ii>kk) {
      small= *(log_lh+ii) ;
      small_pointer = log_lh_pointer[ii] ;
      for(jj=ii;jj>kk;jj--) {
        *(log_lh+jj)= *(log_lh+jj-1) ;
        log_lh_pointer[jj] = log_lh_pointer[jj-1]  ;
        }
      *(log_lh+kk)=small ;
      log_lh_pointer[kk] = small_pointer ;
    }
    else {
      hantei=0 ;
      if ((*(log_lh+kk)<st)) m1=kk-1 ;
      else m1=kk ;
    }

    if(hantei!=0) {
      kk++ ;
      while((*(log_lh+kk)> st)&&(ii>kk)) kk++ ;
      if(ii>kk) {
        large= *(log_lh+kk) ;
        large_pointer = log_lh_pointer[kk] ;
        for(ll=kk;ll<ii;ll++) {
          *(log_lh+ll)= *(log_lh+ll+1) ;
          log_lh_pointer[ll] = log_lh_pointer[ll+1] ;
        }
```

LINE # TEXT

```
/*-------------------------------------------------------------------------*/

  for(v=0;v<DIM_OF_VECTOR;v++)                    /* INITIALIZE */
    p_data[v] = 0.0 ;
  ip = read_phoneme_data(start_frame,p_data) ;
  ip = calc_b(p_data)                          ;
  ip = viterbi_fr_0()                          ;

/*-------------------------------------------------------------------------*/
/*                    frame = 1 to (number_of_frames-2)                    */
/*-------------------------------------------------------------------------*/

  for(fr=1;fr<(number_of_frames-1);fr++) {
    if((fr%5)==0)
      printf("%4d frames  / %4d FINISHED\n",fr,number_of_frames) ;

    for(v=0;v<DIM_OF_VECTOR;v++)
      p_data[v]=0.0                ;
    ip = init_parameter()          ;
    fn = start_frame + fr          ;
    ip = read_phoneme_data(fn,p_data) ;
    ip = calc_b(p_data)            ;
    ip = viterbi()                 ;
    }

/*-------------------------------------------------------------------------*/
/*                              end frame                                  */
/*-------------------------------------------------------------------------*/

  for(v=0;v<DIM_OF_VECTOR;v++)                    /* INITIALIZE */
    p_data[v]=0.0                ;
  ip = init_parameter()          ;
  fn = start_frame + number_of_frames -1 ;
  ip = read_phoneme_data(fn,p_data)      ;
  ip = calc_b(p_data)            ;
  ip = viterbi_fr_last()         ;

/*-------------------------------------------------------------------------*/
/*                           DISPLAY RESULT                                */
/*-------------------------------------------------------------------------*/

  ip=sort_probability() ;
  printf("RECOGINITION RESULT\n\n");
  for(i=0;i<NUMBER_OF_RANK;i++) {
    br=sorted_label_number[i] ;
    j = (i%10)+1 ;
    switch(j) {
    case 1 : sprintf(ord,"st. ") ; break ;
    case 2 : sprintf(ord,"nd. ") ; break ;
    case 3 : sprintf(ord,"rd. ") ; break ;
      default  :  sprintf(ord,"th. ") ;
    }
    printf("------------------------------ %2d%s result ------------------------------\n\n",j,ord) ;
    if(br!=-1) {
      for(k=0;k<word_length[br];k++) {                          /* 認識単語の表示 */
        temp = reco_word[br][k] ;
        if(temp>=0 && temp!=PAUSE)
          printf("%s",word_name[temp]);
      }
      printf("\n\n") ;
      printf("単語番号 ") ;                                     /* 単語番号の表示 */
      for(k=0;k<word_length[br];k++) {
        temp = reco_word[br][k]       ;
        printf("%d ",word_number[temp]);
      }
      printf("\n") ;
      printf("Likelyhood = %30.15f \n",log_lh[br]) ;            /* 尤度の表示     */
      printf("Beam Number %2d : word length %d \n",br,word_length[br]) ; /* 文の長さの表示 */
    }
    else
      printf("候補なし\n") ;
    printf("\n") ;
  }
}
```

LINE # TEXT

```
/*-------------------------------------------------------------------------

                    SENTENCE RECOGNITION PROGRAM

              USING ONE PASS DP & BEAM SEARCH & LANGUAGE-HMM

            Copyright Hiroki YAMAMOTO          10/27/1992

                        連続文認識プログラム

                                                    山本 寛樹

  -------------------------------------------------------------------------*/

/*-------------------------------------------------------------

   Hit key like as follows.
   実行形式 : dp  <A>  <B>
                  <A>.....start frame of voice data
                  <B>.....end frame

   parameter & some valuables are defined in "parameter.h"
   変数・定数は "parameter.h" 参照

  -------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include "parameter.h"

  main(argc,argv)
     int argc     ;
     char *argv[] ;              /* argv[1]...start frame argv[2]...end frame */
{

/* ローカル変数     */
/* local valuables */

  long  int fn                ;   /* フレーム番号          */
  long  int start_frame       ;   /* スタートフレーム番号  */
  int   number_of_frames      ;   /* 最終フレーム番号      */
  int   i,j,k,ip              ;
  int   br,wn,fr,v,st,temp    ;
  float p_data[DIM_OF_VECTOR] ;   /* 音素特徴パラメータ    */
  FILE  *fp                   ;
  char  file_name[100]        ;
  char  ord[10]               ;

/* 関数・サブルーチン     */
/* function & sub routin */

  int read_language_HMM_data() ; /* 言語ＨＭＭのデータ読み込み          */
  int read_phoneme_data()      ; /* 音素パラメータのデータ読み込み      */
  int read_word_data()         ; /* 認識単語のデータ読み込み            */
  int read_phoneme_HMM_data()  ; /* 音素ＨＭＭのデータ読み込み          */
  int calc_b()                 ; /* 音素シンボル出力尤度計算            */
  int viterbi_fr_0()           ; /* frame = 0 の時の viterbi            */
  int viterbi()                ; /* viterbi                             */
  int viterbi_fr_last()        ; /* 最終 frame の時の viterbi           */
  int init_parameter()         ; /* viterbi で計算するパラメータの更新  */
  int sort_probability()       ; /* 最終尤度の sorting                  */
  long int ctoi()              ;

/*-------------------------------------------------------------------------*/
/*                        INPUT ERROR CHECK                                */
/*-------------------------------------------------------------------------*/

  if(argc<2){
      printf("input error in main.c\n") ;
      exit() ;
    }

/*-------------------------------------------------------------------------*/
/*                        READ LANGUAGE HMM                                */
/*-------------------------------------------------------------------------*/

  read_language_HMM_data() ;

/*-------------------------------------------------------------------------*/
/*                              READ DATA                                  */
/*-------------------------------------------------------------------------*/

  start_frame = ctoi(argv[1]) ;                       /* STRAT FRAME NUMBER */
  number_of_frames = ctoi(argv[2])-ctoi(argv[1])+1  ; /* NUMBER OF FRAMES */

  printf("\n start frame = %d \n",start_frame) ;
  printf(" number of frames = %d \n",number_of_frames) ;

  printf("Now read HMM data\n");
  ip = read_phoneme_HMM_data() ;                      /* READ HMM DATA */

  printf("Now read word data\n");
  ip = read_word_data() ;

  printf("Now start ONE PASS DP\n");

/*-------------------------------------------------------------------------*/
/*                             INITIALIZE                                  */
/*-------------------------------------------------------------------------*/

  for(br=0;br<TEMP_BEAM_RANGE;br++) {
    log_lh[br]          =  MIN_DOUBLE ;   /* ある単語のある状態の尤度 */
    word_state[br]      =  0          ;   /* 単語ＨＭＭ内の状態番号   */
    word_length[br]     =  1          ;   /* それまでに認識された単語の長さ */
    model_state[br]     = -1          ;   /* 言語ＨＭＭで計算中の単語の遷移先の状態 -1 : 文頭 */
    for(j=0;j<MAX_NUMBER_OF_WORD;j++)
      reco_word[br][j]  = -1 ;            /* それまでに認識された単語番号 */
    if(br<BEAM_RANGE) {
      before_log_lh[br]      =  MIN_DOUBLE ;
      before_word_state[br]  =  0          ;
      before_model_state[br] = -1          ;
      before_length[br]      =  1          ;
      for(j=0;j<MAX_NUMBER_OF_WORD;j++)
        before_word[br][j] = -1 ;
    }
  }

/*-------------------------------------------------------------------------*/
/*                            frame = 0                                    */
```

LINE # TEXT

```
#include <stdio.h>
#include <math.h>

long int ctoi(s)
char *s ;
{
  long int w,t=0 ;

  while(*s)
    {
      switch(*s)
        {
        case '1' : w=1 ; break ;
        case '2' : w=2 ; break ;
        case '3' : w=3 ; break ;
        case '4' : w=4 ; break ;
        case '5' : w=5 ; break ;
        case '6' : w=6 ; break ;
        case '7' : w=7 ; break ;
        case '8' : w=8 ; break ;
        case '9' : w=9 ; break ;
        default:   w=0 ;
        }
      t=t*10+w;
      s++ ;
    }
  return(t) ;
}
```

LINE # TEXT

```
/*-------------------------------------------------------------------------
  SENTENCE RECOGNITION PROGRAM SUB-ROUTIN

                          <check_lh.c>

                      main process of beam search

                          ビームサーチの処理

          Copyright Hiroki YAMAMOTO          10/27/1992

                      called from <viterbi.c><viterbi_fr_0.c><viterbi_fr_last.c>
  -------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "parameter.h"

int cheack_lh (br,temp_lh,temp_reco_word,temp_word_state,temp_word_length,temp_model_state,word_change_sw)
  int br               ; /*   番号                                        */
  int word_change_sw   ; /*   1...change   0...unchange                   */
  int temp_reco_word   ; /*   現在計算中の単語                            */
  int temp_word_state  ; /*   現在計算中の単語ＨＭＭ内の状態番号          */
  int temp_word_length ; /*   これまでに認識された単語の数                */
  int temp_model_state ; /*   現在計算中の音節ＨＭＭ内の状態番号          */
  double temp_lh       ; /*   計算中の尤度                                */

{
  int br2,i,min_br,check ;
  double min_temp_lh ;

  check = 0 ;                          /* 計算されている状態の処理が終了:1 未処理:0 */
  for(br2=0;br2<BEAM_NUM;br2++) {
   if(log_lh[br2] > MIN_DOUBLE) {
     if((temp_reco_word==reco_word[br2][word_length[br2]-1]) &&  (temp_word_state==word_state[br2]))

       /* 今までに計算されている状態の場合 */

       {
         check = 1 ;
         if(temp_lh>log_lh[br2]) {
           log_lh[br2]      = temp_lh          ;
           model_state[br2] = temp_model_state ;
           word_state[br2]  = temp_word_state  ;
           word_length[br2] = temp_word_length ;
           for(i=0;i<MAX_NUMBER_OF_WORD;i++)
             reco_word[br2][i] = before_word[br][i] ;
           if(word_change_sw == 1) {
             reco_word[br2][temp_word_length]=temp_reco_word ;
             word_length[br2]++ ;
           }
         }
       }
   }
   if(check == 1) break ;
  }

  if(check == 0) {                          /* 今までに計算されていない状態の場合 */
    log_lh[BEAM_NUM]      = temp_lh          ;
    model_state[BEAM_NUM] = temp_model_state ;
    word_state[BEAM_NUM]  = temp_word_state  ;
    word_length[BEAM_NUM] = temp_word_length ;
    for(i=0;i<MAX_NUMBER_OF_WORD;i++)
      reco_word[BEAM_NUM][i] = before_word[br][i] ;
    if(word_change_sw == 1) {
      reco_word[BEAM_NUM][temp_word_length]=temp_reco_word ;
      word_length[BEAM_NUM]++ ;
    }
    BEAM_NUM++ ;
  }
}
```

```
LINE #  TEXT
     1  #include <stdio.h>
     2
     3  main()
     4  {
     5    int i,j,k,flag,o_length,number_of_sentence ;
     6    char data_file_name[25] ;
     7    FILE *fp,*fpp ;
     8
     9    printf("*****************************************************\n");
    10    printf("*       DATA FILE --------> data                    *\n");
    11    printf("*****************************************************\n\n");
    12
    13    while(1)
    14    {
    15      j=1 ;
    16      printf("Input data file name ----");
    17      scanf("%s",data_file_name);
    18      printf("\n");
    19      printf("\n O.K.? (yes=1/no=0) ");
    20      scanf("%d",&j) ;
    21      if(j!=0) break ;
    22      printf("\n\n");
    23    }
    24    fp=fopen(data_file_name,"r") ;
    25    fpp=fopen("data","w") ;
    26    number_of_sentence=0 ;
    27    while(1)
    28      {
    29        flag=fscanf(fp,"%d",&o_length) ;
    30        if(flag<0) break ;
    31        number_of_sentence=number_of_sentence+1 ;
    32        fprintf(fpp,"%6d",o_length);
    33        printf("%6d  ",o_length);
    34        for(i=0;i<o_length;i++)
    35          {
    36            fscanf(fp,"%d",&k);
    37            fprintf(fpp,"%6d",k);
    38            printf("%6d",k);
    39          }
    40        fprintf(fpp,"\n");
    41        printf("\n");
    42
    43      }
    44    printf("Number of DATA is %d\n",number_of_sentence) ;
    45    fclose(fp);
    46    fclose(fpp);
    47    printf("\n*****************************************************\n");
    48    printf("*    CHANGE <%20s>   INTO <data>    *\n",data_file_name);
    49    printf("*****************************************************\n");
    50
    51  }
    52
    53
```

```
LINE #  TEXT
121             for(k=0;k<SYMBOL_NUMBER;k++)
122               fprintf(fd,"b[%2d][%2d][%3d] = %30.25f \n",i,j,k,before_b[i][j][k]);
123         fclose(fd);
124       }
125       if(MAKE_ASCII_FILE_PI == 1) {
126         fd=fopen(FILE_PI2,"w");
127         for(i=0;i<STATE_NUMBER;i++)
128           fprintf(fd,"pi[%d] = %30.25f\n",i,before_init_value[i]);
129         fclose(fd);
130       }
131
132       if(MAKE_ASCII_FILE_ALL != 0) {
133         fd=fopen(FILE_ALL,"w");
134         fprintf(fd,"##################################################################################################\n") ;
135         fprintf(fd,"                    HMM ver 4.1 "     ) ;
136         fprintf(fd,"                    ITERATION     = %-3d\n",iteration) ;
137         fprintf(fd,"                    END_CONDITION = %-2e\n",END_CONDITION) ;
138         fprintf(fd,"                    STATE_NUMBER  = %-3d  \n",STATE_NUMBER) ;
139         fprintf(fd,"                    SYMBOL_NUMBER = %-5d  \n",SYMBOL_NUMBER) ;
140         fprintf(fd,"                    MAX_DATA      = %-6d  \n",MAX_DATA) ;
141         fprintf(fd,"                    PROBABILITY   = %-25.15f  \n",prob) ;
142         fprintf(fd,"                    DIF           = %-10.8f     %-5.2f%%UP \n",dif,100.0 * (exp(dif)-1.0)) ;
143         fprintf(fd,"                    ENTROPY       = %-25.15f \n",entropy) ;
144         fprintf(fd,"##################################################################################################\n") ;
145         for(i=0;i<STATE_NUMBER;i++) {
146           fprintf(fd,"\n---------------------------------- %d STATE ---------------------------------- \n",i);
147           fprintf(fd,"\npi[%2d] =%23.20f \n\n",i,before_init_value[i]);
148           for(j=0;j<STATE_NUMBER;j++) {
149             fprintf(fd,"a%1d%1d =%23.20f \n",i,j,before_a[i][j]);
150             if(MAKE_ASCII_FILE_ALL == 1) {
151               for(k=0;k<SYMBOL_NUMBER;k++)
152                 if(before_b[i][j][k]!=0.0)
153                   fprintf(fd,"b%d%d[%3d] =%23.20f \n",i,j,k,before_b[i][j][k]);
154             }
155           }
156         }
157         fclose(fd);
158       }
159     }
160
161
162
163
164
165
166
167
168
169
```

LINE # TEXT

```
/*---------------------------------------

        Write Data in File

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                10/19/1992
  ---------------------------------------*/

# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"

write_data()
{
  char fname[100] ;
  FILE *fd ;
  int  i,j,k,ip ;

  if(iteration%SAVE_STEP ==0 ) {

    /*--------------------    Write Binary data in File   --------------------*/

    sprintf(fname,"%s.%d",FILE_A,iteration);
    fd=fopen(fname,"w");
    ip=fwrite(before_a,8,STATE_NUMBER*STATE_NUMBER,fd);
    fclose(fd);
    sprintf(fname,"%s.%d",FILE_B,iteration);
    fd=fopen(fname,"w");
    ip=fwrite(before_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd);
    fclose(fd);
    sprintf(fname,"%s.%d",FILE_PI,iteration);
    fd=fopen(fname,"w");
    ip=fwrite(before_init_value,8,STATE_NUMBER,fd);
    fclose(fd);

    /*--------------------    Write Ascii data in File    --------------------*/

    if(MAKE_ASCII_FILE_A == 1) {
      sprintf(fname,"%s.%d",FILE_A2,iteration);
      fd=fopen(fname,"w");
      for(i=0;i<STATE_NUMBER;i++)
        for(j=0;j<STATE_NUMBER;j++)
          fprintf(fd,"a[%2d][%2d] = %30.25f \n",i,j,before_a[i][j]);
      fclose(fd);
    }
    if(MAKE_ASCII_FILE_B == 1) {
      sprintf(fname,"%s.%d",FILE_B2,iteration);
      fd=fopen(fname,"w");
      for(i=0;i<STATE_NUMBER;i++)
        for(j=0;j<STATE_NUMBER;j++)
          for(k=0;k<SYMBOL_NUMBER;k++)
            fprintf(fd,"b[%2d][%2d][%3d] = %30.25f \n",i,j,k,before_b[i][j][k]);
      fclose(fd);
    }
    if(MAKE_ASCII_FILE_PI == 1) {
      sprintf(fname,"%s.%d",FILE_PI2,iteration);
      fd=fopen(fname,"w");
      for(i=0;i<STATE_NUMBER;i++)
        fprintf(fd,"pi[%d] = %30.25f\n",i,before_init_value[i]);
      fclose(fd);
    }
    if(MAKE_ASCII_FILE_ALL != 0) {
      sprintf(fname,"%s.%d",FILE_ALL,iteration);
      fd=fopen(fname,"w");
      fprintf(fd,"##########################################################################################\n") ;
      fprintf(fd,"                HMM Double Ver 4.1  "      ) ;
      fprintf(fd,"                ITERATION     = %-3d\n",iteration) ;
      fprintf(fd,"                END_CONDITION = %-2e\n",END_CONDITION) ;
      fprintf(fd,"                STATE_NUMBER  = %-3d  \n",STATE_NUMBER) ;
      fprintf(fd,"                SYMBOL_NUMBER = %-5d  \n",SYMBOL_NUMBER) ;
      fprintf(fd,"                MAX_DATA      = %-6d  \n",MAX_DATA) ;
      fprintf(fd,"                PROBABILITY   = %-25.15f  \n",prob) ;
      fprintf(fd,"                DIF           = %-10.8f     %-5.2f%%UP \n",dif,100.0 * (exp(dif)-1.0)) ;
      fprintf(fd,"                ENTROPY       = %-25.15f \n",entropy) ;
      fprintf(fd,"##########################################################################################\n") ;
      for(i=0;i<STATE_NUMBER;i++) {
        fprintf(fd,"\n---------------------------------- %d STATE ---------------------------------- \n",i);
        fprintf(fd,"\npi[%2d] =%23.20f \n\n",i,before_init_value[i]);
        for(j=0;j<STATE_NUMBER;j++) {
          fprintf(fd,"a%1d%1d =%23.20f \n",i,j,before_a[i][j]);
          if(MAKE_ASCII_FILE_ALL == 1) {
            for(k=0;k<SYMBOL_NUMBER;k++)
              if(before_b[i][j][k]!=0.0)
                fprintf(fd,"b%d%d[%3d] =%23.20f \n",i,j,k,before_b[i][j][k]);
          }
        }
      }
      fclose(fd);
    }
  }


  fd=fopen(FILE_A,"w");
  ip=fwrite(before_a,8,STATE_NUMBER*STATE_NUMBER,fd);
  fclose(fd);
  fd=fopen(FILE_B,"w");
  ip=fwrite(before_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd);
  fclose(fd);
  fd=fopen(FILE_PI,"w");
  ip=fwrite(before_init_value,8,STATE_NUMBER,fd);
  fclose(fd);

  if(MAKE_ASCII_FILE_A == 1) {
    fd=fopen(FILE_A2,"w");
    for(i=0;i<STATE_NUMBER;i++)
      for(j=0;j<STATE_NUMBER;j++)
        fprintf(fd,"a[%2d][%2d] = %30.25f \n",i,j,before_a[i][j]);
    fclose(fd);
  }
  if(MAKE_ASCII_FILE_B == 1) {
    fd=fopen(FILE_B2,"w");
    for(i=0;i<STATE_NUMBER;i++)
      for(j=0;j<STATE_NUMBER;j++)
```

LINE # TEXT

```
241        fscanf(fp,"%s",HINSHI[i]) ;
242      }
243    fclose(fp);
244
245    /*----------------- main ------------------------*/
246
247
248    for(i=0;i<129;i++)
249      sen[i]=0 ;
250    for(i=0;i<6420;i++)
251      sym[i]=0;
252    sprintf(sen_data_name,"sen_%s",argv[1]) ;
253    fp=fopen(sen_data_name,"r") ;
254    tango_max = 0 ;
255    bunsyou_suu=0 ;
256    tango_suu=0 ;
257    while(fscanf(fp,"%d ",&sen_length)>=0)
258      {
259        bunsyou_suu++ ;
260        sen[sen_length]++ ;
261        tango_suu = tango_suu + sen_length ;
262        if(sen_length>tango_max)
263          tango_max = sen_length ;
264
265        for(i=0;i<sen_length;i++)
266          {
267            fscanf(fp,"%d",&dummy1) ;
268            sym[dummy1]++ ;
269          }
270      }
271    fclose(sen_data_name) ;
272
273  /*  統計処理 表示  */
274  /*  statistic procedure & dispray */
275
276    for(ln=0;ln<MAX_LABEL;ln++)
277      Hinshi_suu[ln] = 0 ;
278
279    for(ln=0;ln<MAX_WORD;ln++)
280      {
281        num = label[ln] ;
282        Hinshi_suu[num] = Hinshi_suu[num] + sym[ln] ;
283      }
284    for(i=0,i<=9;i++)
285      for(j=0;j<=8;j++)
286        Hinshi_suu[160+i] = Hinshi_suu[160+i] + Hinshi_suu[60+10*i+j] ;
287    Hinshi_suu[170] = Hinshi_suu[157] + Hinshi_suu[158] ;
288    for(i=0,i<5;i++)
289      {
290        Hinshi_suu[19] = Hinshi_suu[19] + Hinshi_suu[160+i] ;
291        Hinshi_suu[32] = Hinshi_suu[32] + Hinshi_suu[165+i] ;
292      }
293    Hinshi_suu[32] = Hinshi_suu[32] + Hinshi_suu[170] ;
294    for(i=0,i<=7;i++)
295      {
296        Hinshi_suu[12] = Hinshi_suu[12] + Hinshi_suu[50+i] ;
297        Hinshi_suu[01] = Hinshi_suu[01] + Hinshi_suu[40+i] ;
298      }
299
300    for(rn=1;rn<=NUMBER_OF_RANK;rn++)
301      {
302        sorted_label_number[rn]=0 ;
303        temp_Hinshi_suu[rn]=0 ;
304      }
305
306    for(rn=1;rn<=NUMBER_OF_RANK;rn++)
307      {
308        max = 0 ;
309        max_ln=0 ;
310
311        for(ln=0;ln<MAX_LABEL;ln++)
312          {
313            if((Hinshi_suu[ln])>max)
314              {
315                max=Hinshi_suu[ln] ;
316                max_ln=ln ;
317              }
318          }
319        temp_Hinshi_suu[rn]=Hinshi_suu[max_ln] ;
320        Hinshi_suu[max_ln] = 0 ;
321        sorted_label_number[rn]=max_ln ;
322      }
323    printf("-------------------------------------\n") ;
324    printf("   文章データ名 %s\n",argv[1]) ;
325    printf("-------------------------------------\n\n") ;
326    printf("文章数 : %d \n単語数 : %d \n平均単語数 : %5.2f \n最大単語数 : %d\n",bunsyou_suu,tango_suu,((float)tango_suu/(float)bunsyou_suu),tang
       o_max) ;
327
328    printf("\n# 品詞別ランキング\n\n") ;
329    printf("順位:品詞名              出現数   割合     出現度（一文当たり）\n") ;
330    for(rn=1;rn<=NUMBER_OF_RANK;rn++)
331      {
332        num = sorted_label_number[rn] ;
333        Hinshi_suu[num]=temp_Hinshi_suu[rn] ;
334        printf("%4d:%-20s %5d  %5.2f%%  %6.3f\n",rn,HINSHI[num],Hinshi_suu[num],(float)Hinshi_suu[num]/(float)tango_suu*100.0,(float)Hinshi_suu[
           num]/(float)bunsyou_suu) ;
335      }
336    printf("\n") ;
337
338    printf("\n# 文章の構成単語数比較\n\n") ;
339    printf("  単語数  :  頻度   :  割合\n") ;
340
341    l=0 ;
342    for(i=0;i<=(tango_max/10);i++)
343      {
344        if(i<1)
345          {
346            for(j=1;j<=30;j++)
347              printf("%8d : %6d : %5.2f%%\n",j,sen[j],100.0*(float)sen[j]/(float)bunsyou_suu);
348            printf("\n");
349          }
350        l=0;
351        for(j=1;j<=10;j++)
352          l=l+sen[i*10+j] ;
353        printf("%3d～%3d : %6d : %5.2f%%\n",i*10+1,i*10+10,l,(float)l/(float)bunsyou_suu*100.0);
354      }
355  }
```

```
LINE #  TEXT
121                 if(strcmp(word_type[wn],"04")==0) label[wn] = 74 ;
122                 if(strcmp(word_type[wn],"05")==0) label[wn] = 75 ;
123                 if(strcmp(word_type[wn],"06")==0) label[wn] = 76 ;
124               }
125             if(strcmp(word_form[wn],"02")==0)
126               {
127                 if(strcmp(word_type[wn],"00")==0) label[wn] = 80 ;
128                 if(strcmp(word_type[wn],"01")==0) label[wn] = 81 ;
129                 if(strcmp(word_type[wn],"02")==0) label[wn] = 82 ;
130                 if(strcmp(word_type[wn],"03")==0) label[wn] = 83 ;
131                 if(strcmp(word_type[wn],"04")==0) label[wn] = 84 ;
132                 if(strcmp(word_type[wn],"05")==0) label[wn] = 85 ;
133                 if(strcmp(word_type[wn],"06")==0) label[wn] = 86 ;
134                 if(strcmp(word_type[wn],"ff")==0) label[wn] = 87 ;
135               }
136             if(strcmp(word_form[wn],"03")==0)
137               {
138                 if(strcmp(word_type[wn],"00")==0) label[wn] = 90 ;
139                 if(strcmp(word_type[wn],"01")==0) label[wn] = 91 ;
140                 if(strcmp(word_type[wn],"02")==0) label[wn] = 92 ;
141                 if(strcmp(word_type[wn],"03")==0) label[wn] = 93 ;
142                 if(strcmp(word_type[wn],"04")==0) label[wn] = 94 ;
143                 if(strcmp(word_type[wn],"05")==0) label[wn] = 95 ;
144                 if(strcmp(word_type[wn],"06")==0) label[wn] = 96 ;
145                 if(strcmp(word_type[wn],"ff")==0) label[wn] = 97 ;
146               }
147
148             if(strcmp(word_form[wn],"04")==0)
149               {
150                 if(strcmp(word_type[wn],"00")==0) label[wn] = 100 ;
151                 if(strcmp(word_type[wn],"01")==0) label[wn] = 101 ;
152                 if(strcmp(word_type[wn],"02")==0) label[wn] = 102 ;
153                 if(strcmp(word_type[wn],"03")==0) label[wn] = 103 ;
154                 if(strcmp(word_type[wn],"04")==0) label[wn] = 104 ;
155                 if(strcmp(word_type[wn],"05")==0) label[wn] = 105 ;
156                 if(strcmp(word_type[wn],"06")==0) label[wn] = 106 ;
157               }
158             if(strcmp(word_form[wn],"02")==0) label[wn] = 107 ;
159           }
160         if(part_number[wn]==32)
161           {
162             if(strcmp(word_form[wn],"00")==0)
163               {
164                 if(strcmp(word_type[wn],"00")==0) label[wn] = 110 ;
165                 if(strcmp(word_type[wn],"01")==0) label[wn] = 111 ;
166                 if(strcmp(word_type[wn],"02")==0) label[wn] = 112 ;
167                 if(strcmp(word_type[wn],"03")==0) label[wn] = 113 ;
168                 if(strcmp(word_type[wn],"04")==0) label[wn] = 114 ;
169                 if(strcmp(word_type[wn],"05")==0) label[wn] = 115 ;
170                 if(strcmp(word_type[wn],"06")==0) label[wn] = 116 ;
171                 if(strcmp(word_type[wn],"ff")==0) label[wn] = 117 ;
172               }
173             if(strcmp(word_form[wn],"01")==0)
174               {
175                 if(strcmp(word_type[wn],"00")==0) label[wn] = 120 ;
176                 if(strcmp(word_type[wn],"01")==0) label[wn] = 121 ;
177                 if(strcmp(word_type[wn],"02")==0) label[wn] = 122 ;
178                 if(strcmp(word_type[wn],"03")==0) label[wn] = 123 ;
179                 if(strcmp(word_type[wn],"04")==0) label[wn] = 124 ;
180                 if(strcmp(word_type[wn],"05")==0) label[wn] = 125 ;
181                 if(strcmp(word_type[wn],"06")==0) label[wn] = 126 ;
182                 if(strcmp(word_type[wn],"12")==0) label[wn] = 127 ;
183                 if(strcmp(word_type[wn],"ff")==0) label[wn] = 128 ;
184               }
185             if(strcmp(word_form[wn],"02")==0)
186               {
187                 if(strcmp(word_type[wn],"00")==0) label[wn] = 130 ;
188                 if(strcmp(word_type[wn],"01")==0) label[wn] = 131 ;
189                 if(strcmp(word_type[wn],"02")==0) label[wn] = 132 ;
190                 if(strcmp(word_type[wn],"03")==0) label[wn] = 133 ;
191                 if(strcmp(word_type[wn],"04")==0) label[wn] = 134 ;
192                 if(strcmp(word_type[wn],"05")==0) label[wn] = 135 ;
193                 if(strcmp(word_type[wn],"06")==0) label[wn] = 136 ;
194               }
195             if(strcmp(word_form[wn],"03")==0)
196               {
197                 if(strcmp(word_type[wn],"00")==0) label[wn] = 140 ;
198                 if(strcmp(word_type[wn],"01")==0) label[wn] = 141 ;
199                 if(strcmp(word_type[wn],"02")==0) label[wn] = 142 ;
200                 if(strcmp(word_type[wn],"03")==0) label[wn] = 143 ;
201                 if(strcmp(word_type[wn],"04")==0) label[wn] = 144 ;
202                 if(strcmp(word_type[wn],"05")==0) label[wn] = 145 ;
203                 if(strcmp(word_type[wn],"06")==0) label[wn] = 146 ;
204               }
205
206             if(strcmp(word_form[wn],"04")==0)
207               {
208                 if(strcmp(word_type[wn],"00")==0) label[wn] = 150 ;
209                 if(strcmp(word_type[wn],"01")==0) label[wn] = 151 ;
210                 if(strcmp(word_type[wn],"02")==0) label[wn] = 152 ;
211                 if(strcmp(word_type[wn],"03")==0) label[wn] = 153 ;
212                 if(strcmp(word_type[wn],"04")==0) label[wn] = 154 ;
213                 if(strcmp(word_type[wn],"05")==0) label[wn] = 155 ;
214                 if(strcmp(word_type[wn],"06")==0) label[wn] = 156 ;
215               }
216             if(strcmp(word_form[wn],"05")==0)
217               {
218                 if(strcmp(word_type[wn],"03")==0) label[wn] = 157 ;
219                 if(strcmp(word_type[wn],"06")==0) label[wn] = 158 ;
220               }
221             if(strcmp(word_form[wn],"06")==0) label[wn] = 159 ;
222             if(strcmp(word_form[wn],"ff")==0) label[wn] = 160 ;
223
224           }
225
226         if(data!=word_number[wn])
227           printf("something wrong in read_word_data \n") ;
228       }
229     fclose(fp) ;
230
231     sprintf(filename,"%s",LABEL_FILE);
232     fp=fopen(filename,"r");
233     if(fp==NULL)
234       {
235         printf("file <%s> is not open at reading a \n",filename) ;
236         exit() ;
237       }
238     for(ln=0;ln<LABEL_DATA;ln++)
239       {
240         fscanf(fp,"%d",&i) ;
```

```
LINE #  TEXT
     1  4!/*-------------------------------------------
     2
     3            Analize sentence data
     4                      6/30/1992
     5
     6    -------------------------------------------*/
     7
     8  #include <math.h>
     9  #include <stdio.h>
    10  #include <strings.h>
    11
    12  #define SYMBOL_NUMBER 6420
    13  #define WORD_FILE_NAME "/NFS/atrq12/q12/users/xyama/bigram/dic1" /*<--- 単語ファイル名 */
    14  #define LABEL_FILE "/NFS/atrq12/q12/users/xyama/EXP_DATA/label"
    15
    16  #define MAX_WORD 6418              /*<--- 単語の最大数   */
    17  #define MAX_NUMBER_OF_WORD 150
    18  #define MAX_LABEL 171               /*<--- 品詞の数   */
    19  #define LABEL_DATA 130
    20  #define NUMBER_OF_RANK  130
    21
    22  int word_number[MAX_WORD] ;
    23  int number_of_state[MAX_WORD] ;
    24  int number_of_phoneme[MAX_WORD] ;
    25  char word_name[MAX_WORD][100] ;
    26  int  part_number[MAX_WORD] ;
    27  char word_form[MAX_WORD][5] ;
    28  char word_type[MAX_WORD][5] ;
    29  int  Hinshi_suu[MAX_LABEL] ;
    30  int  temp_Hinshi_suu[MAX_LABEL] ;
    31  int max ;
    32  int sorted_label_number[NUMBER_OF_RANK+1] ;
    33
    34  char HINSHI[MAX_LABEL][30] ;
    35  int label[MAX_WORD] ;
    36
    37  main(argc,argv)
    38  int argc ;
    39  char *argv[] ;
    40  {
    41    int dummy1,sen_length,tango_max,bunsyou_suu;
    42    static int tango_suu ;
    43    char sen_data_name[20] ;
    44    int sen[129],sym[6420] ;
    45    int i,j,k,l,m,wn,st,st2,ln,md ;
    46    int max_ln,rn ;
    47    int data,num ;
    48    char dummy[3000],dum[10] ;
    49    char filename[100] ;
    50    FILE *fp ;
    51
    52
    53  /*##################*/
    54  /* SET WORD NUMBER */
    55  /*##################*/
    56
    57    for(wn=0;wn<MAX_WORD;wn++)
    58      word_number[wn] = wn ;
    59
    60  /*####################*/
    61  /*     READ DATA      */
    62  /*####################*/
    63
    64    fp=fopen(WORD_FILE_NAME,"r") ;
    65    if(fp==NULL)
    66      {
    67        printf("file <%s> is not open in read_word_data \n",WORD_FILE_NAME) ;
    68        printf("Now wn = %d \n",wn) ;
    69        exit() ;
    70      }
    71    for(wn=0;wn<MAX_WORD;wn++)
    72      {
    73        fscanf(fp,"%d %s %s %s %s %d %s %s  ",
    74               &data,dummy,word_name[wn],dummy,dummy,&part_number[wn],
    75               word_form[wn],word_type[wn]) ;
    76        fgets(dummy,3000,fp) ;
    77  /*      printf("case%5d : num = %2d semicoron end semicoron \n",wn,part_number[wn]); */
    78
    79        label[wn] = part_number[wn] ;
    80        if(part_number[wn]==1)
    81          {
    82            if(strcmp(word_form[wn],"00")==0) label[wn] = 40 ;
    83            if(strcmp(word_form[wn],"01")==0) label[wn] = 41 ;
    84            if(strcmp(word_form[wn],"02")==0) label[wn] = 42 ;
    85            if(strcmp(word_form[wn],"03")==0) label[wn] = 43 ;
    86            if(strcmp(word_form[wn],"04")==0) label[wn] = 44 ;
    87            if(strcmp(word_form[wn],"05")==0) label[wn] = 45 ;
    88            if(strcmp(word_form[wn],"06")==0) label[wn] = 46 ;
    89            if(strcmp(word_form[wn],"ff")==0) label[wn] = 47 ;
    90          }
    91
    92        if(part_number[wn]==12)
    93          {
    94            if(strcmp(word_form[wn],"00")==0) label[wn] = 50 ;
    95            if(strcmp(word_form[wn],"01")==0) label[wn] = 51 ;
    96            if(strcmp(word_form[wn],"02")==0) label[wn] = 52 ;
    97            if(strcmp(word_form[wn],"03")==0) label[wn] = 53 ;
    98            if(strcmp(word_form[wn],"04")==0) label[wn] = 54 ;
    99            if(strcmp(word_form[wn],"05")==0) label[wn] = 55 ;
   100            if(strcmp(word_form[wn],"06")==0) label[wn] = 56 ;
   101          }
   102
   103        if(part_number[wn]==19)
   104          {
   105            if(strcmp(word_form[wn],"00")==0)
   106              {
   107                if(strcmp(word_type[wn],"00")==0) label[wn] = 60 ;
   108                if(strcmp(word_type[wn],"01")==0) label[wn] = 61 ;
   109                if(strcmp(word_type[wn],"02")==0) label[wn] = 62 ;
   110                if(strcmp(word_type[wn],"03")==0) label[wn] = 63 ;
   111                if(strcmp(word_type[wn],"04")==0) label[wn] = 64 ;
   112                if(strcmp(word_type[wn],"05")==0) label[wn] = 65 ;
   113                if(strcmp(word_type[wn],"06")==0) label[wn] = 66 ;
   114              }
   115            if(strcmp(word_form[wn],"01")==0)
   116              {
   117                if(strcmp(word_type[wn],"00")==0) label[wn] = 70 ;
   118                if(strcmp(word_type[wn],"01")==0) label[wn] = 71 ;
   119                if(strcmp(word_type[wn],"02")==0) label[wn] = 72 ;
   120                if(strcmp(word_type[wn],"03")==0) label[wn] = 73 ;
```

```
121         if(before_a[i][j]==0.0)
122           before_b[i][j][s]=b[i][j][s];
123         else
124           before_b[i][j][s]=child_b[i][j][s]/child_a[i][j];
125
126
127
128     }
129
130   /*
131         sum3=0.0;
132
133         for(j=0;j<STATE_NUMBER;j++)
134         sum3=sum3+child_a[i][j];
135
136   */
137   /*
138           printf( " child error \n");
139           printf( " %d %d %e %e %e \n",i,j,before_a[i][j],child_a[i][j],sum3);
140           printf( " %d %d %e %e %e \n",i,j,1.0/before_a[i][j],1.0/child_a[i][j],1.0/sum3);
141   */
142   /*
143         if(sum3 != mother_a[i])  {
144           printf(" a error \n");
145           printf(" %e %e \n",sum3,mother_a[i]);
146           for(j=0;j<STATE_NUMBER;j++)  {
147             printf(" %e  %e \n",child_a[i][j],mother_a[i]);
148           }
149         }
150   */
151
152     }
153     for(i=0;i<STATE_NUMBER;i++)
154       before_init_value[i]=before_init_value[i]/sum2 ;
155     return(answer_p);
156   }
157
158
```

```
LINE #  TEXT
     1  /*-----------------------------------------
     2
     3          Reestimate parameter
     4
     5          Ergodic HMM Lerning Program
     6
     7          Double valuable version
     8          Using Scaling
     9          No Flooring
    10
    11          Copyright Hiroki YAMAMOTO
    12                                  08/26/1992
    13  -----------------------------------------*/
    14
    15  # include <stdio.h>
    16  # include <math.h>
    17  # include "hmm_state.h"
    18
    19  double reestimate()
    20  {
    21    int    i,j,k,t,s,Ot ;
    22    double  answer_p,aij ;
    23    double  sum,sum2,sum3 ;
    24    double  forward(),backward();
    25    FILE   *fd;
    26    int    flag;
    27
    28    /*############ INITIALIZE ##############*/
    29
    30    for(i=0;i<STATE_NUMBER;i++) {
    31      before_init_value[i]=0.0 ;
    32      mother_a[i]=0.0 ;
    33      for(j=0;j<STATE_NUMBER;j++) {
    34        child_a[i][j]=0.0 ;
    35        for(s=0;s<SYMBOL_NUMBER;s++)
    36          child_b[i][j][s]=0.0 ;
    37      }
    38    }
    39
    40    fd=fopen(LEARN_DATA_FILE,"r") ;
    41    flag=0 ;
    42    answer_p=0.0 ;
    43    while(1) {
    44      clear_o() ;
    45      flag = fscanf(fd,"%d",&o_length);
    46      if(flag<0) break;
    47      for(k=0;k<o_length;k++) {
    48        fscanf(fd,"%d",&o[k]);
    49        if(o[k]>SYMBOL_NUMBER) {
    50          printf(" symbol number is too big in main_a \n");
    51          exit();
    52        }
    53      }
    54      init_forward();
    55      init_backward();
    56
    57      /*######### CHECK SCALE ERROR & CALCULATE PROBABILITY ##########*/
    58      sum = 0.0 ;
    59      for(t=0;t<=o_length;t++) {
    60        if( scaling_value[t] <= 0.0 ) {
    61          sum2 = 0.0 ;
    62          printf("scaling value error in <reestimate>\n") ;
    63          printf("scaling value = %f\n",scaling_value[t]) ;
    64          exit() ;
    65        }
    66        else {
    67          if( scaling_value[t] <= MIN ) {
    68            sum2 = log(MIN) ;
    69            printf("scaling value error2 in <reestimate>\n") ;
    70            printf("scaling value = %f\n",scaling_value[t]) ;
    71          }
    72          else
    73            sum2 = log(scaling_value[t]) ;
    74        }
    75        sum = sum - sum2 ;
    76      }
    77
    78      answer_p = answer_p + sum ;
    79      if(sum==0.0) {
    80        printf("scale error p= %f in reestimate \n",sum);
    81        exit();
    82      }
    83
    84      /*##########  CALCULATE a b pi ##################*/
    85
    86      for(i=0;i<STATE_NUMBER;i++) {
    87        before_init_value[i]=before_init_value[i]+forward(0,i)*backward(0,i) ;
    88        sum=0.0 ;
    89        for(t=1,t<=o_length;t++)
    90          sum=sum+forward(t-1,i)*(backward(t-1,i)/scaling_value[t-1]) ;
    91        mother_a[i]=mother_a[i]+sum ;
    92        for(j=0;j<STATE_NUMBER;j++) {
    93          aij=a[i][j];
    94          for(t=1,t<=o_length,t++) {
    95            Ot=o[t-1] ;
    96            sum2=0.0 ;
    97            sum2=forward(t-1,i)*(aij*b[i][j][Ot]*backward(t,j)) ;
    98
    99            for(s=0;s<SYMBOL_NUMBER;s++)
   100              if(Ot==s)  child_b[i][j][s]=child_b[i][j][s]+sum2 ;
   101
   102            child_a[i][j]=sum2+child_a[i][j] ;
   103          }
   104        }
   105      }
   106    }
   107    fclose(fd);
   108
   109    sum2=0.0 ;
   110    for(i=0;i<STATE_NUMBER;i++) {
   111      sum2=sum2+before_init_value[i] ;
   112      for(j=0;j<STATE_NUMBER;j++) {
   113  /*
   114            sum3 = 0.0 ;
   115            for(s=0;s<SYMBOL_NUMBER;s++)
   116              sum3 = sum3 + child_b[i][j][s] ;
   117  */
   118        before_a[i][j]=child_a[i][j]/mother_a[i] ;
   119
   120        for(s=0;s<SYMBOL_NUMBER;s++)
```

LINE # TEXT

```
# ----------------------------------------
#
#       Make File for
#
#       Ergodic HMM Lerning Program
#
#         Float valuable version
#         Using Scaling
#         No Flooring
#
#       Copyright Hiroki YAMAMOTO
#                                 08/26/1992
# ----------------------------------------

!CC = cc
CFLAGS = -O2

hmm:    hmm.o reestimate.o init.o forward.o backward.o \
        init_forward.o init_backward.o initialize.o write_data.o cal_entro.o
        cc  hmm.o reestimate.o init.o forward.o backward.o  \
                init_forward.o init_backward.o  \
                        initialize.o write_data.o cal_entro.o -lm -o hmm
reestimate.o hmm.o init.o forward.o backward.o  \
        init_forward.o init_backward.o initialize.o write_data.o cal_entro.o: hmm_state.h
```

LINE # TEXT

```
/*---------------------------------------

        Calculate Initialvalue of HMM

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  ---------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"
# include <stdlib.h>

double    init_for[STATE_NUMBER];
double    init_trs[STATE_NUMBER][STATE_NUMBER];
double    init_sym[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];

initialize()
{
  int    i,j,k,tp;
  double  sum;
  int    rand() ;
  void   srand() ;
  long   data_ran;
  FILE   *fd;

  srand(INIT_RAND) ;
  for(i=0;i<STATE_NUMBER;i++)
    init_for[i]=1.0/(double)STATE_NUMBER ;


  for(i=0;i<STATE_NUMBER;i++) {
    for(j=0;j<STATE_NUMBER;j++) {
      data_ran=rand() ;
      init_trs[i][j]=(double)data_ran ;
    }
    sum=0.0 ;
    for(j=0;j<STATE_NUMBER;j++)
      sum=init_trs[i][j]+sum ;
    for(j=0;j<STATE_NUMBER;j++)
      init_trs[i][j]=init_trs[i][j]/sum;
  }

  for(i=0;i<STATE_NUMBER;i++) {
    for(j=0;j<STATE_NUMBER;j++) {
      for(k=0;k<SYMBOL_NUMBER;k++) {
        data_ran=rand() ;
        if(data_ran==0)  data_ran=1 ;
        init_sym[i][j][k]=(double)data_ran ;
      }
      sum=0.0;
      for(k=0;k<SYMBOL_NUMBER;k++)
        sum=init_sym[i][j][k]+sum ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        init_sym[i][j][k]=init_sym[i][j][k]/sum ;
      sum=0.0 ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        sum=init_sym[i][j][k]+sum ;
      printf(" b [%d][%d][%d]sum = %f \n",i,j,k,sum) ;
    }
  }

  fd=fopen(FILE_INIT_PI,"w") ;
  tp=fwrite(init_for,8,STATE_NUMBER*STATE_NUMBER,fd) ;
  fclose(fd) ;

  fd=fopen(FILE_INIT_A,"w") ;
  tp=fwrite(init_trs,8,STATE_NUMBER*STATE_NUMBER,fd) ;
  fclose(fd) ;

  fd=fopen(FILE_INIT_B,"w") ;
  tp=fwrite(init_sym,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd) ;
  fclose(fd) ;

}
```

```
/*-----------------------------------------

        Calculate forward-value

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  -----------------------------------------*/

# include <stdio.h>
# include <math.h>
# include "hmm_state.h"

double pre_alpha[STATE_NUMBER] ;
double alpha_star[STATE_NUMBER] ;

int init_forward()

{
  int t,i,j,ip,Ot ;
  double sum,sum2,Ct ;
  FILE *fd;

  sum=0.0;
  for(i=0;i<STATE_NUMBER;i++)
    sum=init_value[i]+sum ;


/*  printf(" SIGMA pi in init_forward=%30.25f \n",sum);  */
  if((sum<0.9999) || (sum > 1.00001)) {
    printf(" init pro error %30.25f in init_forward\n",sum);
    for(i=0;i<STATE_NUMBER;i++)
      printf(" %d %f \n",i,pro[i]);
    exit();
  }

  /*############ INITIALIZE ##############*/
  for(t=0;t<=MAX_DATA;t++)
    scaling_value[t]=0.0 ;

  for(i=0;i<STATE_NUMBER;i++) {
    pre_alpha[i]=0.0;
    alpha_star[i]=0.0 ;
  }

  /*############ t = 0 #############*/
  scaling_value[0]=1.0 ;
  for(i=0;i<STATE_NUMBER;i++) {
    pre_alpha[i]=init_value[i] ;
    forward_value[0][i]=init_value[i] ;
  }

  /*######## 1<= t <= T (=o_length) ########*/
  for(t=1;t<=o_length;t++) {
    Ot=o[t-1];
    Ct=0.0 ;
    sum=0.0 ;

    for(i=0;i<STATE_NUMBER;i++) {
      alpha_star[i]=0.0 ;
      sum2=0.0 ;
      for(j=0;j<STATE_NUMBER;j++)
        sum2=sum2+pre_alpha[j]*a[j][i]*b[j][i][Ot] ;
      alpha_star[i]=sum2 ;
      sum=sum+sum2 ;
    }
    Ct=1.0/sum ;

    if(Ct==0.0) {
      printf("scaling value error in <init_forward>\n") ;
      printf("Ct = %20.10f sum = %e \n",Ct,sum) ;
      exit();
    }

    scaling_value[t]=Ct ;


    for(i=0;i<STATE_NUMBER;i++) {
      pre_alpha[i]=alpha_star[i]*Ct ;
      forward_value[t][i]=pre_alpha[i] ;
    }
  }

  /*      printf(" SIGMA alpha[t=%2d] = %30.25f \n",t,sum);  */
}
```

LINE # TEXT

```
/*-----------------------------------------

        calculate backward-value

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                  08/26/1992
  -----------------------------------------*/
# include <stdio.h>
# include <math.h>
# include "hmm_state.h"
double pre_beta[STATE_NUMBER] ;
double beta[STATE_NUMBER] ;


int init_backward()
{

  int t,i,j,Ot ;
  double sum,sum2,Ct ;

  for(i=0;i<STATE_NUMBER;i++) {
    pre_beta[i]=0.0;
    beta[i]=0.0;
  }

  /*############ t = T (= o_length)###############*/
  Ct=scaling_value[o_length] ;
  for(i=0;i<STATE_NUMBER;i++) {
    pre_beta[i]=Ct ;
    backward_value[o_length][i]=Ct ;
  }

  /*############ 0 <= t < T ###############*/
  for(t=o_length-1;t>=0;t--) {
    Ct=scaling_value[t] ;
    Ot=o[t] ;
    for(i=0;i<STATE_NUMBER;i++) {
      sum=0.0;
      for(j=0;j<STATE_NUMBER;j++)
        sum=sum+a[i][j]*b[i][j][Ot]*pre_beta[j] ;
      beta[i]=sum*Ct ;
      backward_value[t][i]=beta[i] ;
    }
    for(i=0;i<STATE_NUMBER;i++) {
      pre_beta[i]=beta[i] ;
      beta[i]=0.0 ;
    }
  }
}
```

LINE # TEXT

```
/*-----------------------------------------

        Read Initial value of HMM
            (Relearn Version)

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  -----------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"
# define  NEW_VALUE 1.0e-10

init()
{
  int    i,j,k,ip ;
  double sum ;
  FILE   *fd ;

  fd=fopen(FILE_INIT_A,"r") ;
  ip=fread(before_a,8,STATE_NUMBER*STATE_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++)  {
    sum=0.0 ;
    for(j=0;j<STATE_NUMBER;j++)
      sum=sum+before_a[i][j] ;
    printf(" init %d trs sum = %f \n",i,sum) ;
  }

  fd=fopen(FILE_INIT_B,"r") ;
  ip=fread(before_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++) {
    for(j=0;j<STATE_NUMBER;j++) {
      before_b[i][j][1372] = NEW_VALUE ;
      before_b[i][j][2018] = NEW_VALUE ;
      before_b[i][j][2571] = NEW_VALUE ;
      before_b[i][j][2992] = NEW_VALUE ;
      before_b[i][j][3080] = NEW_VALUE ;
      before_b[i][j][3650] = NEW_VALUE ;
      before_b[i][j][3667] = NEW_VALUE ;
      before_b[i][j][3685] = NEW_VALUE ;
      before_b[i][j][3693] = NEW_VALUE ;
      before_b[i][j][5976] = NEW_VALUE ;

      sum=0.0 ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        sum=sum+before_b[i][j][k] ;
      printf("SIGMA b[%d][%d]= %f \n",i,j,sum) ;
    }
  }

  fd=fopen(FILE_INIT_PI,"r") ;
  ip=fread(before_init_value,8,STATE_NUMBER,fd) ;
  fclose(fd) ;

  sum=0.0 ;
  for(i=0;i<STATE_NUMBER;i++)
    sum=sum+before_init_value[i] ;
  printf(" init  sum = %f \n",sum) ;
}



clear_o()
{
  int i;
  for(i=0;i<MAX_DATA;i++)
    o[i]=0;
}
```

```
/*----------------------------------------

        Read Initialvalue of HMM

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
 ----------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"

init()
{
  int    i,j,k,ip ;
  double  sum ;
  FILE   *fd ;

  fd=fopen(FILE_INIT_A,"r") ;
  ip=fread(before_a,8,STATE_NUMBER*STATE_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++)  {
    sum=0.0 ;
    for(j=0;j<STATE_NUMBER;j++)
      sum=sum+before_a[i][j] ;
    printf(" init %d trs sum = %f \n",i,sum) ;
  }

  fd=fopen(FILE_INIT_B,"r") ;
  ip=fread(before_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++) {
    for(j=0;j<STATE_NUMBER;j++) {
      sum=0.0 ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        sum=sum+before_b[i][j][k] ;
      printf("SIGMA b[%d][%d]= %f \n",i,j,sum) ;
    }
  }

  fd=fopen(FILE_INIT_PI,"r") ;
  ip=fread(before_init_value,8,STATE_NUMBER,fd) ;
  fclose(fd) ;

  sum=0.0 ;
  for(i=0;i<STATE_NUMBER;i++)
    sum=sum+before_init_value[i] ;
  printf(" init  sum = %f \n",sum) ;
}



clear_o()
{
  int i;
  for(i=0;i<MAX_DATA;i++)
    o[i]=0;
}
```

LINE # | TEXT

```
/*---------------------------------------

        Read Initialvalue of HMM

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  ---------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"

init()
{
  int    i,j,k,ip ;
  double  sum ;
  FILE   *fd ;

  fd=fopen(FILE_INIT_A,"r") ;
  ip=fread(before_a,8,STATE_NUMBER*STATE_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++)  {
    sum=0.0 ;
    for(j=0;j<STATE_NUMBER;j++)
      sum=sum+before_a[i][j] ;
    printf(" init %d trs sum = %f \n",i,sum) ;
  }

  fd=fopen(FILE_INIT_B,"r") ;
  ip=fread(before_b,8,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd) ;
  fclose(fd) ;

  for(i=0;i<STATE_NUMBER;i++) {
    for(j=0;j<STATE_NUMBER;j++) {
      sum=0.0 ;
      for(k=0;k<SYMBOL_NUMBER;k++)
        sum=sum+before_b[i][j][k] ;
      printf("SIGMA b[%d][%d]= %f \n",i,j,sum) ;
    }
  }

  fd=fopen(FILE_INIT_PI,"r") ;
  ip=fread(before_init_value,8,STATE_NUMBER,fd) ;
  fclose(fd) ;

  sum=0.0 ;
  for(i=0;i<STATE_NUMBER;i++)
    sum=sum+before_init_value[i] ;
  printf(" init  sum = %f \n",sum) ;
}


clear_o()
{
  int i;
  for(i=0;i<MAX_DATA;i++)
    o[i]=0;
}
```

LINE # TEXT

```
/*----------------------------------------

        Valuable & FILE_NAME of

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  ----------------------------------------*/

# define MAX_DATA           130
# define STATE_NUMBER       8
# define SYMBOL_NUMBER      6420
# define END_CONDITION      0.1
# define END_ITERATION      20
# define INIT_RAND          900
# define MIN                1.0E-300
# define INIT_PROB          0.0
# define INIT_ITERATION     0

# define SAVE_STEP          50
# define MAKE_ASCII_FILE_A   0
# define MAKE_ASCII_FILE_B   0
# define MAKE_ASCII_FILE_PI  0
# define MAKE_ASCII_FILE_ALL 2  /* 0..NO    1..SAVE   2..SAVE without bij[k]   */

# define FILE_ENT        "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.ent"
# define FILE_A          "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.HP.a"
# define FILE_B          "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.HP.b"
# define FILE_PI         "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.HP.pi"
# define FILE_A2         "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.aa"
# define FILE_B2         "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.bb"
# define FILE_PI2        "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.pipi"
# define FILE_ALL        "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/test.all"
# define FILE_INIT_A     "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/init.HP.a"
# define FILE_INIT_B     "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/init.HP.b"
# define FILE_INIT_PI    "/NFS/atrq12/q12/users/xyama/EXP_DATA/STATE_08/odd4000.nf9/init.HP.pi"
# define LEARN_DATA_FILE "sen_odd4000"

/*
# define FILE_ENT        "test.ent"
# define FILE_A          "test.a"
# define FILE_B          "test.b"
# define FILE_PI         "test.pi"
# define FILE_A2         "test.aa"
# define FILE_B2         "test.bb"
# define FILE_PI2        "test.pipi"
# define FILE_ALL        "test.all"
# define FILE_INIT_A     "test.a.0"
# define FILE_INIT_B     "test.b.0"
# define FILE_INIT_PI    "test.pi.0"
# define LEARN_DATA_FILE "data"
*/

int     o_length;
int     o[MAX_DATA];

int     iteration ;
double  entropy,before_ent,dif_ent,prob,dif,before_prob ;
double  forward_value [MAX_DATA+1][STATE_NUMBER];
double  backward_value[MAX_DATA+1][STATE_NUMBER];
double  a[STATE_NUMBER][STATE_NUMBER];
double  b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];
double  before_a[STATE_NUMBER][STATE_NUMBER];
double  before_b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];
double  pro[STATE_NUMBER],pre_pro[STATE_NUMBER],pro_star[STATE_NUMBER];
double  init_value[STATE_NUMBER];
double  before_init_value[STATE_NUMBER];
double  scaling_value[MAX_DATA+1];
double  mother_a[STATE_NUMBER];
double  child_a[STATE_NUMBER][STATE_NUMBER];
double  child_b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];
```

```
/*----------------------------------------

        Main Program of

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
                                $Log: hmm.c,v $
 * Revision 1.1  1992/10/20  02:32:26  xyama
 * Initial revision
 *
 ----------------------------------------*/
# include <stdio.h>
# include <math.h>
# include <sys/file.h>
# include "hmm_state.h"

main()
{
  int   i,j,k,ip ;
  int   initialize(),cal_entro(),write_data() ;
  double reestimate();
  FILE  *fd;


  setbuf(stdout,NULL);

  if(INIT_ITERATION==0)
    ip = initialize();
  ip = init();
  prob = INIT_PROB ;
  iteration = INIT_ITERATION ;
  dif = 0.0 ; before_ent = 0.0 ;
  cal_entro() ;
  while(1) {
    if(iteration==END_ITERATION) break ;
    before_prob = prob ;
    iteration++ ;
    for(i=0;i<STATE_NUMBER;i++) {
      init_value[i] = before_init_value[i] ;
      for(j=0;j<STATE_NUMBER;j++) {
        a[i][j] = before_a[i][j] ;
        for (k=0;k<SYMBOL_NUMBER;k++)
          b[i][j][k] = before_b[i][j][k] ;
      }
    }

    prob = reestimate() ;

    dif = prob-before_prob ;
    printf("%3d: PROB = %-22.13f",iteration,prob);
    cal_entro() ;
    if(iteration>=2) {
      write_data () ;
      printf("  DIF = %-10.5f %6.2f%%up  ENT = %6.3f\n",dif,100.0*(exp(dif)-1.0),entropy);
      if(dif < log(1.0+END_CONDITION)) break ;
    }
    else printf("\n");
  }
}
```

LINE # | TEXT

```
1  /*----------------------------------------
2
3          Return forward-value
4
5          Ergodic HMM Lerning Program
6
7          Double valuable version
8          Using Scaling
9          No Flooring
10
11         Copyright Hiroki YAMAMOTO
12                                   08/26/1992
13   ----------------------------------------*/
14 # include <stdio.h>
15 # include <math.h>
16 # include "hmm_state.h"
17
18 double forward(time,j)
19
20 int j;
21 int time;
22 {
23 /*
24   printf(" t %d j %d value %f \n",time,j,log(forward_value[time][j]));
25 */
26   return(forward_value[time][j]);
27 }
```

```
LINE #  TEXT
    1
    2   # include <stdio.h>
    3   # include <math.h>
    4
    5   # define STATE_NUMBER     8
    6   # define SYMBOL_NUMBER  6420
    7   float  before_a[STATE_NUMBER][STATE_NUMBER];
    8   float  before_b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];
    9   float  before_init_value[STATE_NUMBER];
   10
   11   main()
   12   {
   13     int i,j,k ;
   14     int ip;
   15     float entropy_p,entropy_HK,entropy,temp ;
   16     FILE *fd;
   17
   18     setbuf(stdout,NULL);
   19
   20
   21
   22     fd=fopen("test.a","r");
   23     ip=fread(before_a,4,STATE_NUMBER*STATE_NUMBER,fd);
   24     fclose(fd);
   25     fd=fopen("test.b","r");
   26     ip=fread(before_b,4,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd);
   27     fclose(fd);
   28     fd=fopen("test.pi","r");
   29     ip=fread(before_init_value,4,STATE_NUMBER,fd);
   30     fclose(fd);
   31
   32     entropy=0.0 ;
   33     for(i=0;i<STATE_NUMBER;i++)
   34       {
   35         entropy_HK=0.0 ;
   36         for(k=0;k<SYMBOL_NUMBER;k++)
   37           {
   38             entropy_p=0.0 ;
   39             for(j=0;j<STATE_NUMBER;j++)
   40               {
   41                 entropy_p=entropy_p+before_a[i][j]*before_b[i][j][k] ;
   42               }
   43             if(entropy_p<=1.000000e-37) temp = 0.0 ;
   44             else temp = (float)log((double)entropy_p) ;
   45   /*        printf("entropy = %e (float)log((double)ent) = %e \n",entropy_p,temp) ; */
   46             entropy_HK=entropy_HK-entropy_p/(float)log((double)2.0)*temp ;
   47           }
   48         entropy=entropy+before_init_value[i]*entropy_HK ;
   49       }
   50
   51
   52     fd=fopen("test.ent","w");
   53     fprintf(fd,"ENTROPY      = %-25.15f \n",entropy) ;
   54     fprintf(fd,"PERPLEXITY   = %-25.15f \n",pow(2.0,entropy)) ;
   55     fclose(fd);
   56   }
```

```
121             {
122               pro[j] = 0.0 ;
123               temp = 0 ;
124               for(i=0;i<STATE_NUMBER;i++)
125                 if ((pre_pro[i] != 0.0) && (a[i][j] > FL_A) && (b[i][j][ot] > FL_B))
126                   temp = 1 ;
127               if(temp == 1)
128                 pro[j] = 1.0 ;
129               else
130                 pro[j] = 0.0 ;
131             }
132           for(j=0;j<STATE_NUMBER;j++)
133             pre_pro[j] = pro[j] ;
134         }
135       temp = 0 ;
136       for(j=0;j<STATE_NUMBER;j++)
137         if(pre_pro[j] != 0.0)
138           temp = 1 ;
139       if(temp == 1)
140         COV++ ;
141     }
142   printf("odd%s Coverage is %d / %s \n",SEN_NUM,COV,SEN_NUM) ;
143 }
144
145
146
147
148
149
150
151
152
```

```
LINE #  TEXT
   1  /*################################################*/
   2  /*    coverage 1992/07/06                         */
   3  /*################################################*/
   4
   5  # include <stdio.h>
   6  # include <math.h>
   7  # include <strings.h>
   8  # define STATE_NUMBER  4
   9  # define SEN_NUM  "4000"
  10  # define RESULT_DIR "/NFS/atrq12/q12/users/xyama/HMM_sc5/"
  11  # define DATA_DIR "/NFS/atrq12/q12/users/xyama/sentence/data_base/sen_even4000"
  12  # define MAX_DATA       130
  13  # define SYMBOL_NUMBER  6420
  14  # define FL_A 0.0
  15  # define FL_B 0.0
  16
  17
  18
  19  int     o_length;
  20  int     o[MAX_DATA];
  21
  22  float  a[STATE_NUMBER][STATE_NUMBER];
  23  float  b[STATE_NUMBER][STATE_NUMBER][SYMBOL_NUMBER];
  24
  25  float  pro[STATE_NUMBER],pre_pro[STATE_NUMBER] ;
  26  float  init_value[STATE_NUMBER];
  27  main()
  28  {
  29    int    flag;
  30    int    i,j,k,s,t,sen,ip;
  31    int temp,ot,COV ;
  32
  33    float sum ;
  34    FILE   *fd;
  35    char   file_name[80] ;
  36
  37
  38    /*############ INITIALIZE ##############*/
  39    for(i=0;i<STATE_NUMBER;i++)
  40      {
  41        init_value[i]=0.0 ;
  42        for(j=0;j<STATE_NUMBER;j++)
  43          {
  44            a[i][j]=0.0 ;
  45            for(s=0;s<SYMBOL_NUMBER;s++)
  46              b[i][j][s]=0.0 ;
  47          }
  48      }
  49
  50
  51    sprintf(file_name,"%stest.a",RESULT_DIR) ;
  52    fd=fopen(file_name,"r");
  53    ip=fread(a,4,STATE_NUMBER*STATE_NUMBER,fd);
  54    fclose(fd);
  55
  56    for(i=0;i<STATE_NUMBER;i++)
  57      {
  58        sum=0.0;
  59        for(j=0;j<STATE_NUMBER;j++)
  60          sum=sum+a[i][j];
  61        printf(" init %d a sum = %f \n",i,sum);
  62      }
  63
  64    sprintf(file_name,"%s/test.b",RESULT_DIR) ;
  65    fd=fopen(file_name,"r");
  66    ip=fread(b,4,STATE_NUMBER*STATE_NUMBER*SYMBOL_NUMBER,fd);
  67    fclose(fd);
  68
  69    for(i=0;i<STATE_NUMBER;i++)
  70      {
  71        for(j=0;j<STATE_NUMBER;j++)
  72          {
  73            sum=0,0 ;
  74            for(k=0;k<SYMBOL_NUMBER;k++)
  75              sum=sum+b[i][j][k];
  76            printf("SIGMA b[%d][%d]= %f \n",i,j,sum);
  77          }
  78      }
  79
  80
  81    sprintf(file_name,"%stest.pi",RESULT_DIR) ;
  82    fd=fopen(file_name,"r");
  83    ip=fread(init_value,4,STATE_NUMBER,fd);
  84    fclose(fd);
  85
  86    sum=0.0;
  87    for(i=0;i<STATE_NUMBER;i++)
  88      sum=sum+init_value[i];
  89    printf(" pi  sum = %f \n",sum);
  90
  91
  92    sprintf(file_name,"%s",DATA_DIR) ;
  93    fd=fopen(file_name,"r");
  94    flag=0;
  95
  96    COV = 0 ;
  97
  98    while(1)
  99      {
 100        for(i=0;i<MAX_DATA;i++)
 101          o[i]=0;
 102
 103        flag=fscanf(fd,"%d",&o_length);
 104        if(flag<0) break;
 105        for(k=0;k<o_length;k++)
 106          {
 107            fscanf(fd,"%d",&o[k]);
 108            if(o[k]>SYMBOL_NUMBER)
 109              {
 110                printf(" symbol number is too big in main_a \n");
 111                exit();
 112              }
 113          }
 114        for(i=0;i<STATE_NUMBER;i++)
 115          pre_pro[i] = init_value[i] ;
 116
 117        for(t=0;t<o_length;t++)
 118          {
 119            ot = o[t] ;
 120            for(j=0;j<STATE_NUMBER;j++)
```

LINE # TEXT

```
# include <stdio.h>
# include <math.h>
# include "hmm_state.h"

double  matrix[STATE_NUMBER][STATE_NUMBER+1] ;
double  con_st_prob[STATE_NUMBER] ;

cal_entro()
{
  int i,j,k,ip;
  double entropy_p,entropy_HK,log_p ;
  FILE *fd;

  for(i=0;i<STATE_NUMBER-1;i++) {
    for(j=0;j<STATE_NUMBER+1;j++) {
      if(j!=STATE_NUMBER) {
        if(i!=j)
          matrix[i][j] = before_a[j][i] ;
        else
          matrix[i][j] = before_a[j][i] - 1.0 ;
      }
      else
        matrix[i][j] = 0.0 ;
    }
  }

  for(j=0;j<STATE_NUMBER+1;j++)
    matrix[STATE_NUMBER-1][j] = 1.0 ;

  for(k=0;k<STATE_NUMBER;k++) {
    for(j=k+1;j<=STATE_NUMBER;j++)
      matrix[k][j] = matrix[k][j]/matrix[k][k] ;
    for(i=0;i<STATE_NUMBER;i++)
      if(i!=k)
        for(j=k+1;j<=STATE_NUMBER;j++)
          matrix[i][j] = matrix[i][j] -matrix[i][k] * matrix[k][j] ;
  }

  for(i=0;i<STATE_NUMBER;i++)
    con_st_prob[i] = matrix[i][STATE_NUMBER] ;

  entropy = 0.0 ;
  for(i=0;i<STATE_NUMBER;i++) {
    entropy_HK = 0.0 ;
    for(k=0;k<SYMBOL_NUMBER;k++) {
      entropy_p = 0.0 ;
      for(j=0;j<STATE_NUMBER;j++)
        entropy_p = entropy_p+before_a[i][j]*before_b[i][j][k] ;
      if(entropy_p<=MIN) log_p = 0.0 ;
      else log_p = log(entropy_p) ;
      entropy_HK = entropy_HK-entropy_p/log(2.0)*log_p ;
    }
    entropy = entropy+con_st_prob[i]*entropy_HK ;
  }

  dif_ent = entropy - before_ent ;
  fd=fopen(FILE_ENT,"a");
  ip=fprintf(fd,"%4d PROB = %20.10f  dif = %7.4f  ENTOROPY = %8.5f   dif = %6.4f\n",iteration,prob,dif,entropy,dif_ent);
  fclose(fd);
  before_ent = entropy ;
}
```

LINE # TEXT

```
/*-----------------------------------------

        Return backward-value

        Ergodic HMM Lerning Program

        Double valuable version
        Using Scaling
        No Flooring

        Copyright Hiroki YAMAMOTO
                                08/26/1992
  -----------------------------------------*/
# include <stdio.h>
# include <math.h>
# include "hmm_state.h"

double backward(time,j)

int j;
int time;
{
/*
  printf(" beta[%d][%d]= %f \n",time,j,backward_value[time][j]);
*/
  return(backward_value[time][j]);
}
```