既存知識を用いたＨＭＭのパラメータ推定

茨城大学 工学部 システム工学科

大内 弘和　92T7017R

新納 浩幸　(担当教官)

平成８年３月４日

目 次

1 はじめに **1**

1.1 研究の目的 1

1.2 本論文の概要 1

2 形態素解析について **2**

2.1 形態素解析 2

2.2 既存の形態素解析の方法 4

2.2.1 右方向最長一致法 4

2.2.2 字種切り法 4

2.2.3 文節数最小法 5

3 HMM **7**

3.1 マルコフ過程 7

3.2 基本 HMM の定式化 11

3.3 viterbi アルゴリズム 15

3.4 Viterbi アルゴリズムの例 17

3.5 HMM の学習 17

3.5.1 前向きパスアルゴリズム 17

3.5.2 EM アルゴリズム 18

3.5.3 パラメータ推定アルゴリズム 20

4 HMM の形態素解析への応用 **27**

4.1 HMM による形態素解析 27

4.2 HMM による形態素解析の長所と短所 28

4.3 既存知識の利用 28

5 HMM と Viterbi アルゴリズムによる形態素解析 **30**
5.1 方法 30

6 解析結果 **33**
6.1 まとめ 35

図目次

2.1 英語の文章 2
2.2 日本語の文章 3
2.3 日本語の文章の形態素解析 3
2.4 右方向最長一致法による形態素解析 4
2.5 文節数最小法で用いる表 6

3.1 単純マルコフ過程の例 9
3.2 2 重マルコフ過程の例 10
3.3 ユニフィラーな有限状態情報源 11
3.4 HMM の例 12
3.5 ナル遷位の除去 24
3.6 HMM におけるシンボル系列 abb に対する 最適パス算出例 25
3.7 $P(abb)$ のトレリス上の計算 25
3.8 HMM における $P(abb)$ に対する後ろ向きパスアルゴリズム 26

4.1 HMM の形態素解析への応用 28

5.1 24 に分類した単語品詞番号と品詞 32

6.1 二つのパラメータ sptran1.dat と sptran2.dat の解析結果 33

Chapter 1

はじめに

1.1 研究の目的

自然言語処理では入力文の構文，意味構造を知るためにその文がどのような単語列から構成されているかを，まず，調べる必要がある．この処理は形態素解析と呼ばれている．英語の場合は，基本的に空白により単語区切りがされているので，主に単語の品詞を推定する処理となるが，日本語の場合は膠着文と呼ばれる単語区切りがなされていない文法のため単語区切りと品詞付与の処理となり，比較的複雑な処理である．この形態素解析は自然言語処理の必須の要素技術であり，従来さまざまな手法が研究されてきた [1]．そして近年形態素解析は HMM[2] によってモデル化できることが示された [3].

HMM を利用した形態素解析はいくつかの長所があるが，パラメータの学習に膨大なデータと時間が必要であるという欠点がある [4].

本研究の目的は，HMM のパラメータ学習を小さなデータで効率良く行なうことである．そのために既存の知識を HMM の初期パラメータに反映させ，学習量を少なくする事を検証した.

1.2 本論文の概要

本研究では形態素解析と HMM を扱うため、第 2 章では形態素解析の目的とその方法を、第 3 章では HMM の説明と viterbi アルゴリズム、HMM の学習の方法を述べる。第 4 章では HMM を形態素解析に応用するために必要な定義を示す。そして第 5 章と第 6 章で今回行なった実験について説明する。

Chapter 2

形態素解析について

2.1 形態素解析

形態素解析の目的は，文を構成する形態素 (morpheme) を認定する事である．形態素とは，正確に言えば，言語学的に意味がある (文を構成する) 最小言語単位の事であるが，以下では辞書に登録されている単語 (単に語と呼ぶこともある) を形態素と呼ぶこともある．

自然言語解析の通常の流れは，文の形態素解析をおこなってから次のステップに進む．文中の形態素が認定できなければ次の処理が不可能なため，形態素解析は自然言語処理の要素技術といえる．

欧米の言語の文では，単語と単語の間に空白をおく言語であるため，文中の語形変化がない単語は辞書を引く事により直ちに認識可能な事と，たとえ語形変化した単語があっても，規則変化形である限り，その解析が比較的容易である．

Rome was not built in a day.

Figure 2.1: 英語の文章

ところが日本語のような，単語と単語の間に空白による切れ目を置かない膠着語の場合には，形態素解析は難しい問題になる．例えば漢字をふくまない文を単語単位に分割し，単語を認定する事は難しい．これは文を構成する単語が語形変化しておらず，すべて辞書に登録されている場合でも問題になる．

現在の形態素解析の技術では，統語的並びに意味的に妥当な形態素の系列が得られるかどうかを必ずしも問題にしない．しかし最近の形態素解析では，抽出可能なあらゆる形態

ローマは一日にしてならず.

Figure 2.2: 日本語の文章

素の系列を得るだけではなく，統語的な情報や様々なヒューリスティクスを用いて，それらをしぼりこむ手法が研究されている．意味解析を組み入れようとする試みもあるが，極めて限定された範囲での意味解析であり，本格的な意味解析とはまだ言えない．

ローマ は 一日 に して ならず .
名詞 副助詞 数詞 接尾辞 格助詞 動詞 動詞 助動詞

Figure 2.3: 日本語の文章の形態素解析

これまでのほとんどの形態素解析の手法は，統語解析とは独自のヒューリスティクスに基づくアルゴリズムを採用してきた．次の章ではこれらの既存の形態素解析の手法を紹介しよう．

2.2 既存の形態素解析の方法

2.2.1 右方向最長一致法

右方向最長一致法は，文を左から右の向きに見て，もっとも長い形態素を優先させて分割するものである．例えば「アルプスの少女は美しい」を最長一致法を用いて分割してみる．

まず始めに，文全体も最長の形態素の候補であるが，それは辞書にない．そこで最右端の一文字「い」を削除した「アルプスの少女は美し」について同じことを繰り返す．これも辞書にないから最右端の一文字を更に削除していくと，最終的に「アルプス」が右方向最長の形態素として認定される．次は「アルプス」を取り除いた残りの「の少女は美しい」について同様な事を繰り返すと，「の」，「少女」，「は」，「美しい」を得る．このようにして正しい形態素結果が得られる．

(a) アルプスの少女は美しい

最長一致部	残りの部分
アルプス	の少女は美しい
の	少女は美しい
少女	は美しい
は	美しい
美しい	

(b) アルプスのやまは美しい

最長一致部	残りの部分
アルプス	のやまは美しい
のやま	は美しい
は	美しい
美しい	

Figure 2.4: 右方向最長一致法による形態素解析

2.2.2 字種切り法

我々は平仮名で書かれた文より，適当に漢字が交じった文の方が読みやすい．これは漢字が形態素解析に何らかの手助けをしているからだろう．字種切り法はこの事実を利用する．

字種切り法では，ひらがなからほかの字種への変わり目，区切り符号の前後，非平仮名列から数字列への変わり目，数字列から非平仮名列への変わり目に強制的に切れ目を入れる．　もちろんこれは第一近似としての形態素である．強制的に入れた切れ目が誤りである事も多い．送り仮名のために「読み手」が「読み　手」に，「申し分ない話」が「申し

分ない　話」のように，誤った位置に余計な切れ目が入ってしまう．また漢字だけからなる副詞も問題を引き起こす.「今回直接会談した結果不可能であると判断する」は「今回直接会談した　結果不可能であると　判断する」のように分割される．これには切れ目が不足している．この様な場合,「今回」「直接」などと言う漢字からなる副詞には，改めてその前後に切れ目を入れる必要がある．助動詞が相互承接した平仮名文字列についても，適当な箇所に切れ目を入れなければならない．

2.2.3　文節数最小法

文節数最小法は横型探索をベースにしたヒューリスティクスで，入力文に含まれるあらゆる単語の候補からなる表を用意し，その表を左から右に走査して，文節数が最小となる単語の系列を優先して出力する方法である．例えば,「くるまでまつ」に大して，あらゆる単語の候補を切り出した表を始めにつくる．図 2.5の右には，入力文を作り出す単語の列の脇に文節数が書かれている．文節数最小法は，文節数 2 の単語列を文節数 3 の単語列に優先させるということである．図からわかるように，文節数最小法のヒューリスティクスは，入力文が短く，文を構成する文節数が少ない場合にはさほど有効に働かない．文節数最小法は，入力文の長さが長いほど有効であるとされているが，不完全な方法である事には変わりがない．またこの方法は膠着語一般に適用可能な物ではなく，文節と言う概念が日本語に特有な物であること，もともと平仮名のべた書き文にたいして考えられた方法である事に注意しなければならない．

			文節数
車で	待つ		2
車で	松		2
来るまで	待つ		2
来るまで	松		2
繰るまで	待つ		2
繰るまで	松		2
狂るまで	待つ		2
狂るまで	松		2
来る	間で	待つ	3
来る	間で	松	3

Figure 2.5: 文節数最小法で用いる表

Chapter 3

HMM

3.1 マルコフ過程

まず始めに、HMM 法 (Hidden Markov Model, 隠れマルコフモデル) の基礎となるマルコフ過程について述べる.

時刻 t における確率変数を $Y_t(t = 1, 2, ...)$, 観測値を y_t としよう. $Y_1 = y_1, Y_2 = y_2, ...Y_t = y_t$を知ったときの, ナル確率が次式で与えられるとき, n 重マルコフ過程という.

$$P(Y_{t+1} = y_{t+1}|\mathbf{Y}_1^t = \mathbf{y}_1^t) = P(Y_{t+1} = y_{t+1}|\mathbf{Y}_{t-n+1}^t = \mathbf{y}_{t-n+1}^t \tag{3.1}$$

ここで, $\mathbf{Y}_1^t(\mathbf{y}_1^t)$ は $Y_1, Y_2, ..., Y_t(y_1, y_2, ..., y_t)$ の時系列をあらわし, $\mathbf{Y}_1^t = \mathbf{y}_1^t$は, $Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t$ を表すものとする。特に, 1 重マルコフ過程のときには単純マルコフ過程という.

確率過程 $\{Y_t\}$ のとりうる値を有限個または可付番号無限個として、これを非負の数だけに限っても一般性を失わない. 単純マルコフ過程においては, $Y_1 = n_1, Y_2 = n_2, ..., Y_t = n_t$ のとき, $Y_{t+1} = n_{t+1}$ なる条件を考えれば,

$$P(Y_{t+1} = n_{t+1}|\mathbf{Y}_1^t = \mathbf{n}_1^t) = P(Y_{t+1} = n_{t+1}|Y_t = n_t) \tag{3.2}$$

が成立する. これが常に成立するものとして、

$$P_{ij}(t, t+1) = P(Y_{t+1} = j|Y_t = i) \tag{3.3}$$

と書くと,

$$P_{ij}(t,t+1) \geq \sum_j P_{ij}(t,t+1) = 1 \tag{3.4}$$

である. $P_{ij}(t,t+1)$ は、時刻 t に状態 i であったものが時刻 $t+1$ で状態 j に遷移する確率を表す. この遷位確率が t に無関係なとき, $Y_1, Y_2, ...$ を定常な遷位確率を持つマルコフ過程と いう. 以下、定常性を仮定して $P_{ij}(t,t+1)$ を単に P_{ij} と書くことにする. また、初期状態確率πを次式で定義する.

$$\pi_i = P(Y_1 = i), \sum_j \pi = 1 \tag{3.5}$$

n 重マルコフ過程を考えるとき, 一個の確率変数 Y_tの代わりに n 次元ベクトル $(Y_t, Y_{t+1}, ..., Y_{t+n-1}) = \mathbf{Y}_t$に着目すれば、新しいベクトル確率過程 $\{\mathbf{Y}_t\}$ が得られる。この過程では $\mathbf{Y}_{t-1}$の最初のベクトル成分は y_{t-1}であり, それ以外の成分は $\mathbf{Y}_t$に含まれている成分と同じであって、現在の事象はその一つ前の事象だけの影響下にあることとなり、確率変数の n 次元のベクトルを考えるだけで n 重マルコフ過程は単純マルコフ過程に 帰着されることになる。

$\{Y_t\}$ のとりうる値を $L_1, L_2, ..., L_s$という S個のシンボルとすれば、$\mathbf{Y}_t = (Y_t, Y_{t+1}, ..., Y_{t+n-1})$ のとりうる値は たかだか S^n(S^nの状態) のシンボルとなる。

定常な遷位確率 P_{ij}を持った単純マルコフ過程を考えよう. 遷位確率 P_{ij}を要素とする行列

$$\mathbf{P} = \begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ P_{20} & P_{21} & P_{22} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \tag{3.6}$$

を遷位確率マトリクスという. 一般に次の関係が成立する.

$$P_{ij}(m+n) = \sum_v^{\infty} P_{iv}(m) P_{vj}(n) \tag{3.7}$$

ここで、左辺はあるとき i であった場合に $(n+m)$ 回の遷位で jになる確率である. これを行列形式でかけば,

$$\mathbf{P}(m+n) = \mathbf{P}(m)\mathbf{P}(n) = (\mathbf{P})^{m+n} \tag{3.8}$$

となり、これは Kolmogorov-Chapman の方程式と言われている. $t=1$ における初期状態の確率を $\sum_i$ とすれば, $Y_t = j$となる確率は,

$$P_j(t) = \sum_i \pi_i \cdot P_{ij}(t-1) \tag{3.9}$$

となる．$P_{ij}(t) > 0$ であるような t が存在する事は，状態 i から状態 j へいつかは到達可能である事を意味し、任意の i, j について $P_{ij}(t) > 0$ ならこのマルコフ過程は正則であるという．どのような状態から出発しても何回かの後には必ず他の任意の状態に移る事のできるマルコフ過程はエルゴード的であるという．正則なマルコフ過程はエルゴード的である．正則なマルコフ過程では、初期上位の確率に関係なく、サンプル系列 $\{n_1, n_2, ..., n_t\}$ にたいして状態 j を通過する回数を $u_j(t)$ とすれば、状態 j を通過する確率 $u_j(t)/t$ は，t を無限大にすれば一定値となる．これはエルゴードの定理と呼ばれている．

n 重マルコフ過程は状態遷位図で表現する事ができる．ベクトル $(y_t, y_{t+1}, ..., y_{t+n-1})$ がとりうる値の一つを $(L_{i0}, L_{i1}, ..., L_{in-1})$ とすればこれを 1 状態に対応させる．また、各状態について、その状態から 1 回の遷位で到達する事ができる状態 (たかだか $(L_{i1}, L_{i2}, ..., L_{in-1}, L_k), k = 1, 2, ..., S$ の S 個の状態) を方向を持った線で結ぶ．また、この遷位に対して、状態遷位確率を付随させ、さらに、一つのシンボル L_k を発生するものとする．

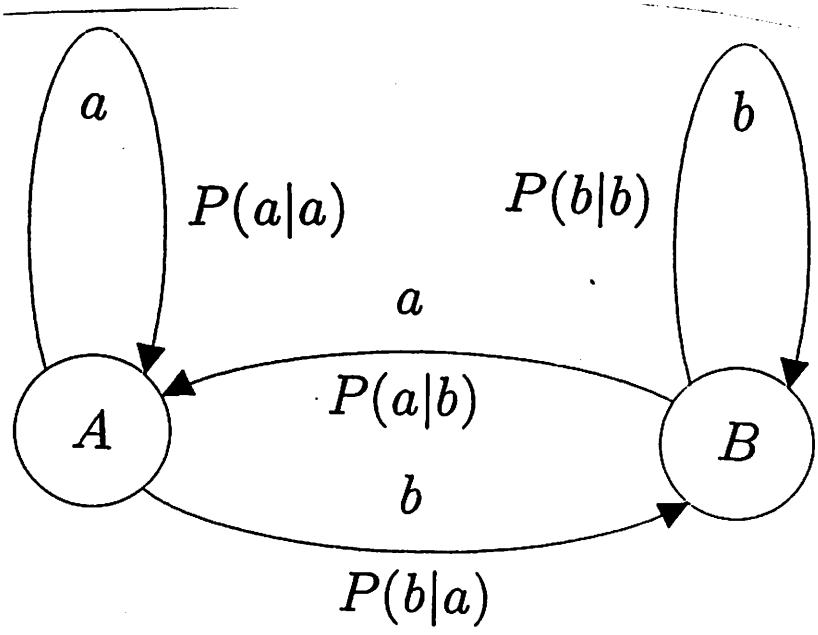

Figure 3.1: 単純マルコフ過程の例

図 3.1，図 3.2は，$\{a, b\}$ の二つの出力シンボルを持つ単純マルコフ過程，2 重マルコフ過程の状態遷位図である．状態遷位確率は，このマルコフ過程によって生成されうる長い訓

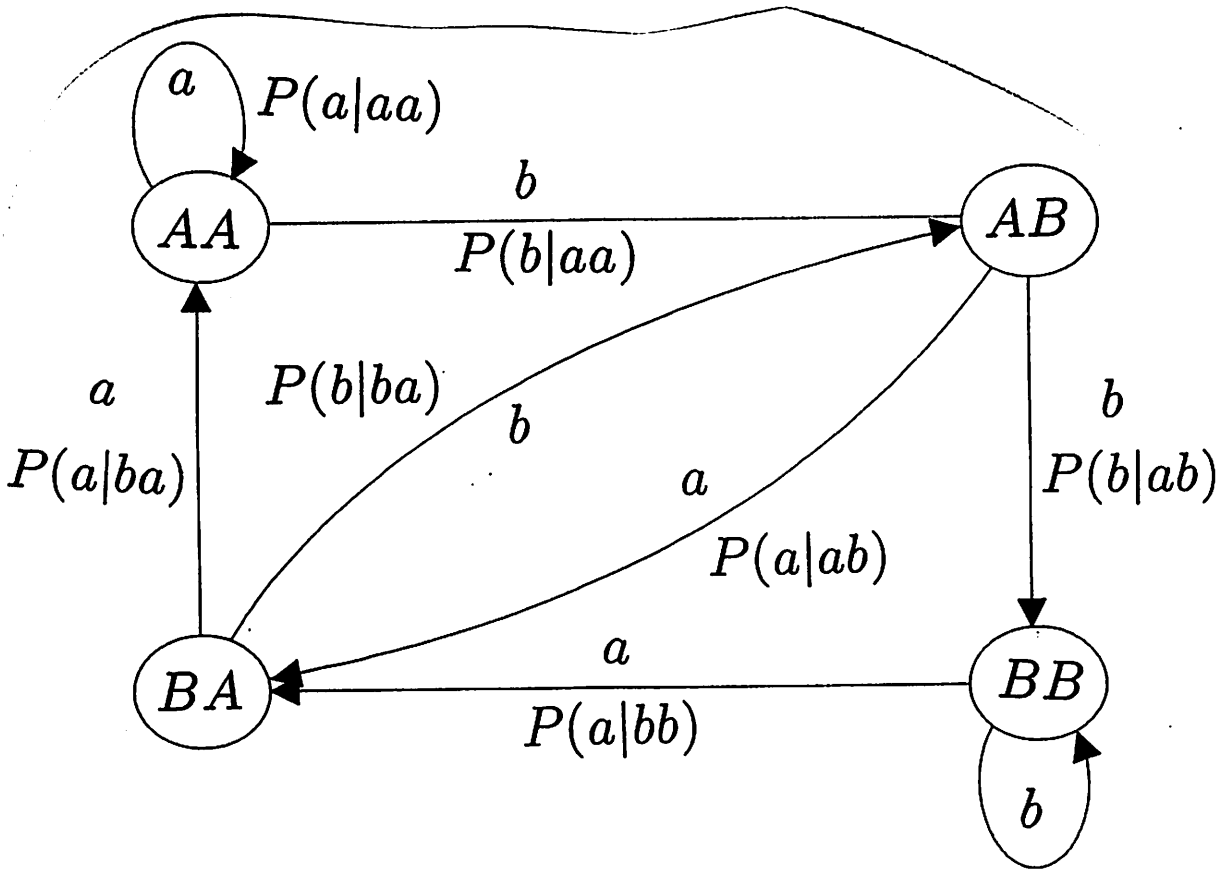

Figure 3.2: 2 重マルコフ過程の例

練用サンプル系列を用いる事によって容易に求められる．例えば，図 3.1で N_Aを状態 A を通った回数，$N_A(b)$ をシンボル b が状態 A から生成された回数とすれば，

$$P(b|a) = \frac{N(a,b)}{N(a)} = \frac{N_A(b)}{N_A} \tag{3.10}$$

同様に，図 3.2で N_{AB}を状態 ABを通った回数，$N_{AB}(b)$ をシンボル b が状態 ABから生成された回数とすると，

$$P(b|ab) = \frac{N(a,b,b)}{N(a,b)} = \frac{N_{AB}(b)}{N_{AB}} \tag{3.11}$$

となる．マルコフ過程のように、各状態から出ているすべての遷位がそれぞれ異なったシンボルを生成する有限状態情報源はユニフィラー (unifilar) と呼ばれる．

ユニフィラーな有限情報源から生成されるシンボル系列に対しては、唯一の状態遷位系列が対応する．それゆえ、このシンボル系列を生成する確率は容易に求められる．たとえば、図 3.3の例で，状態 1 が初期状態、状態 3 が最終状態であるとき，シンボル系列 aabaaac の生成確率は,.

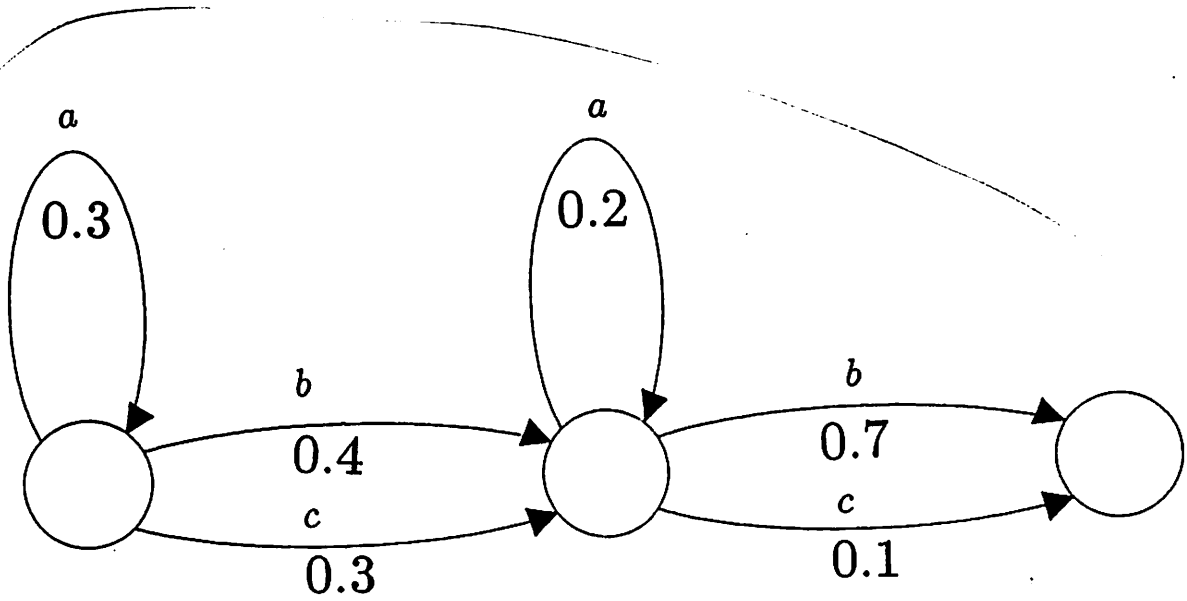

Figure 3.3: ユニフィラーな有限状態情報源

$$P(aabaaac) = 0.3 \times 0.3 \times 0.4 \times 0.2 \times 0.2 \times 0.2 \times 0.1 = 2.88 \times 10^{-5} \tag{3.12}$$

となる.

3.2 基本 HMM の定式化

HMM(Hidden Markov Model, 隠れマルコフモデル) は，出力シンボルによって一意に状態遷位が決まらないという意味での非決定有限状態オートマトンとして定義される (一般に，マルコフモデルは最終状態の概念はないが，今回の実験での形態素解析に用いたマルコフモデルは初期状態，最終状態を設定する). 定義からとうぜん HMM はユニフィラーではない. すなわち，出力シンボルが与えられても状態遷位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事から hidden マルコフモデルと呼ばれている.

簡単な例を図 3.4に示す. 図中のアーク上のスカラ値は状態遷位確率を，アーク上のベクトル値はシンボル a,b の状態遷位による条件付きの出力確率を示している. すなわち，ベクトルの第一要素が a の，第二要素が b の出力確率を示している. この場合，観測シンボル系列が abb であった場合，可能性のある状態遷位系列は $S_1S_1S_2S_3$ と $S_1S_2S_2S_3$ の二通りがある. 今回は出力シンボル系列は最終状態に達するものとして扱う. 通常 HMM は全状態が最終状態となりうるが，形態素解析では一部分の状態のみが最終状態となる.

一般性を失う事なく，状態遷位にナル遷位 (シンボルを何も出力しない) は存在せず二つ

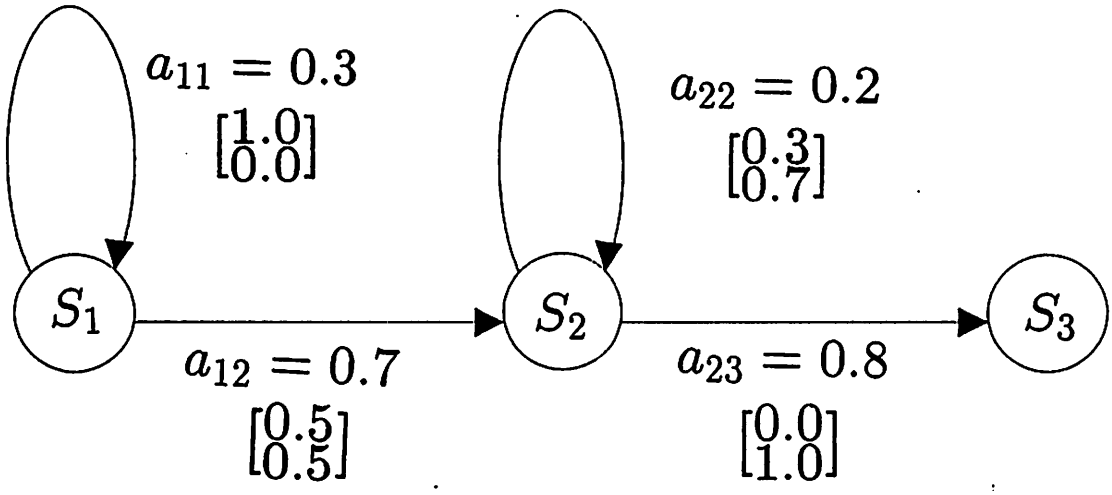

Figure 3.4: HMM の例

の状態間での遷位はたかだか一つと仮定できる．前者の場合は，ナル遷位のときは空記号 λが出力されたと考えれば良い．

また，図 3.5で示すようにナル遷位を除去する事もできる．これによって後者が満たされなくなるが，二つの状態遷位 $S_2 \rightarrow S_3$に対して，新たな状態$\dot{S}_3$を設けて，一方の遷位を $S_2 \rightarrow \dot{S}_3$に変換し，状態$\dot{S}_3$の遷位先を状態 S_3の遷位先に等しくすれば良い．また，y,x をそれぞれ出力と状態の確率変数値の系列としよう．今考えている HMM を M とすれば，マルコフモデルの仮定から

$$P(x_i|\mathbf{x}_i^{i-1}, \mathbf{M}) = P(x_i|x_{i-1}, \mathbf{M}) \tag{3.13}$$

また，非決定性 (hidden) の仮定から，

$$P(y_i|\mathbf{x}_1^i, \mathbf{y}_1^{i-1}, \mathbf{M}) = P(y_i|x_{i-1}, x_i, \mathbf{M}) = P(y_i|x_{i-1}, \mathbf{M}) \tag{3.14}$$

となる．

式は状態遷位によって y_iの出力確率が定まるとするもので，式は遷位元の状態によって定まるとするものである．混同のない限り HMMを示す M の条件項を省く事にする．単語 w_iに対する HMMを

上述の M とすれば，式の $\mathbf{y}_{t_{i-1}}^{t_i}$

をあらためて $\mathbf{y}_1^T$とおけば，$P(\mathbf{y}_1^T|\mathbf{M})$ は，

$$P(\mathbf{y}_1^T) = \sum_x P(\mathbf{y}_1^T|\mathbf{x})P(\mathbf{x}) \tag{3.15}$$

となる．また，

$$P(\mathbf{x}) = \prod_i P(x_i|\mathbf{x}_1^{i-1}) = \prod_i P(x_i|x_{i-1}) \tag{3.16}$$

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}_1^i, \mathbf{y}_1^{i-1} = prod_i P(y_i|x_{i-1}, x_i) \tag{3.17}$$

から，

$$P(\mathbf{y}) = \sum_x \prod_i P(x_i|x_{i-1}) \cdot P(y_i|x_{i-1}, x_i) \tag{3.18}$$

となる．

図 3.4の例で P(abb) の値を式を用いて求めて見よう．先にも述べたとおり $\prod_i P(y_i|x_{i-1}, x_i)$ が 0 にならないのは二つの状態系列だけであったので，この各々の系列による値を $P_1(abb), P_2(abb)$ とすれば，

$$P_1(abb) = 0.3 \times 1.0 \times 0.7 \times 0.5 \times 0.8 \times 1.0 = 0.084 \tag{3.19}$$

$$P_2(abb) = 0.7 \times 0.5 \times 0.2 \times 0.7 \times 0.8 \times 1.0 = 0.0392 \tag{3.20}$$

故に，$P(abb)$ は，

$$P(abb) = P_1(abb) + P_2(abb) = 0.084 + 0.0392 = 0.1232 \tag{3.21}$$

となる．

以上から，形態素解析に用いる HMM **M** は次の六つの組

$$\mathbf{M} = (\mathbf{S}, \mathbf{Y}, \mathbf{A}, \mathbf{B}, \pi, \mathbf{F})$$

で定義される.

- **S**: 状態の有限集合 ; s_i
- **Y**: 出力シンボルの集合
- **A**: 状態遷移確率の集合; **A**=a_{ij}; a_{ij}は状態 s_iから状態 s_jへの遷移確率，ここで $\sum_j a_{ij}$+1.
- **B**:出力確率の集合; **B**=$b_{ij}(k)$; $b_{ij}(k)$ は状態 s_iから状態 s_jへの遷移の際にシンボル kを出力する確率．ここで，

  $\sum_k b_{ij}(k) = 1$ (離散 HMM)

  $\int_{-\infty}^{\infty} b_{ij}(k)dk = 1$ (連続 HMM)

- π : 初期状態確率の集合; $\pi = \{\pi_i\}$; π_iは初期状態が s_iである確率.

  $\sum_j \pi_j = 1$.

- **F**:最終状態の集合.

3.3 viterbi アルゴリズム

二つの単語 w_1 と w_2 に対応する HMM $\mathbf{M}_1$ と $\mathbf{M}_2$ を連結して 一つの HMM $\mathbf{M}_3$ を構成したとしよう. この時, $\mathbf{M}_1$ の 最終状態から $\mathbf{M}_2$ の初期状態へナル遷位を設けて連結するものとしよう (あるいは, $\mathbf{M}_1$ の最終状態を $\mathbf{M}_2$ の初期状態に縮退させても良い). $\mathbf{M}_3$ と入力文字列に対応する出力シンボル系列 $y_1, y_2, ..., y_T$ に 対して前述のアルゴリズムで $P(y_1, y_2, ..., y_T|\mathbf{M}_3$ を 求めることができる. これは一般に,

$$P(y_1, y_2, ..., y_T)|\mathbf{M}_3 = \sum_t P(y_1, y_2, ..., y_t)|\mathbf{M}_1 \times P(y_{t+1}, y_{t+2}, ..., y_T)|\mathbf{M}_2 \tag{3.22}$$

と表現できる. これは, $\mathbf{M}_1$ の最終状態からほかの状態への遷位先は $\mathbf{M}_2$ の初期状態にかぎられる事から明らかである. 上式の関係から, $P(y_1, y_2, ..., y_T)$ の算出の最には, $w_1(\mathbf{M}_1)$ にもっともよく対応する出力シンボル系列区間 $y_1, y_2, ..., y_t$ と $w_2(\mathbf{M}_2)$ にもっともよく対応するシンボル系列区間 $y_{t+1}, y_{t+2}, ..., y_T$ を決定する事はできない.

ところが形態素解析のためにこの最適な単語境界時点 t を知りたい. これは次のような定義

$$\dot{P}(y_1, y_2, ..., y_T|\mathbf{M}_3) \tag{3.23}$$

から求められる.

$$\dot{P}(y_1, y_2, ..., y_T|\mathbf{M}_3) = max_t P(y_1, y_2, ..., y_t|\mathbf{M}_1) \times P(y_{t+1}, y_{t+2}, ..., y_T|\mathbf{M}_2) \tag{3.24}$$

上式の定義は近似的ではあるが妥当な定義とも考えられる. 言い換えれば y_t を HMM $\mathbf{M}_3$ の唯一の状態遷位に対応させたことになる. この考えをすべての y_t に拡張すれば, 式の

$$P(y_1, y_2, ..., y_T) = \sum_x \prod_i P(x_i|x_{i-1}) \cdot P(y_i|x_{i-1}, x_i) \tag{3.25}$$

の代わりに,

$$\ddot{P}(y_1, y_2, ..., y_T) = max_x \{\prod_i P(x_i|x_{i-1}) \cdot P(y_i|x_{i-1}, x_i)\} \tag{3.26}$$

を求めることになる. これは式の {} 内の項を最大にする状態遷位系列上での確率で $P(y_1, y_2, ..., y_T)$ を近似するものである. これによって, 任意の y_t は唯一の状態遷位に対応付けられる. この状態遷位系列を最適パスと呼ぶ.

最適パスとこのパス上での確率を求めるためには，動的計画法を用いる．このアルゴリズムは最初 Viterbi によって提案された事から，Viterbi アルゴリズムと呼ばれている．$\mathbf{Q}(i,t)$ を$\ddot{P}(y_1,y_2,...,y_T) = P(y_1,y_2,...,y_t,\{\mathbf{x}\})$ を最大にするパス (状態遷位系列)，$\mathbf{f}(i,t)$ をその確率と定義すると Viterbi アルゴリズムは以下で与えられる．

viterbi アルゴリズム

(1) 初期化

すべての状態 i に対して $f(i,0) = \pi_i$

(2)$t = 1,2,...,T$に対して (3),(4) を実行.

(3) すべての状態 i に対して (4) を実行.

(4)$\hat{i}$ =argmax$f(k,t)a_{ki}$

$f(i,t)$ =max$\{f(\hat{j},t-1)a_{\hat{j}i} \cdot b_{\hat{j}i}(y_t)\}$

$Q(\hat{j},t-1) \otimes \hat{j}$

(5)$\hat{\hat{i}}$ = argmax$_{i \in F} f(i,T)$

$\ddot{P}(y_1,y_2,...,y_T) = P(y_1,y_2,...,y_T,Q(\hat{i}\,T)) = f(\hat{i},T)$

ここで，$\otimes$ は状態遷位系列と状態を連結し，新たな状態遷位系列をつくるオペレータである．最適状態遷位系列は (最適パス) は $\mathbf{Q}(\hat{i},T)$ で与えられる．

HMM 法による認識では，P(y) をもちいる代わりに，この $\mathbf{f}(i,T)$ を用いて最大値を与えるモデルのカテゴリーを認識結果とする方法もあり，同等の認識精度が得られている．

3.4 Viterbi アルゴリズムの例

形態素解析にもちいる HMM のモデル上では，ある状態からある状態へ遷移するさいにあるシンボルを出力する．ある状態への遷移は存在する状態の数だけとることができ，それぞれ確率によって決まっている．初期状態から最終状態まで遷移し，観測できるのは出力シンボルのみである．

HMM は出力シンボルが与えられても状態遷移系列は唯一に決まらない．さきの 3.4の例で HMM のモデルを考えてみる．

このようなモデルがあったとき，最大の状態遷移確率と状態遷移系列を同時に求めるものが viterbi アルゴリズムである．つまり，このアルゴリズムによって状態遷位を最適にする状態遷位系列と，この系列上での確率をもとめることができる．

図 a5 のモデルの上で，シンボル系列が abb を出力する場合の最適パスを viterbi で解くと，図 3.6のように求められ，最適パスは $S_1S_2S_2S_3$ であり，この時の遷移確率は 0.084 である．

3.5 HMM の学習

3.5.1 前向きパスアルゴリズム

図 3.4について考えてみる．入力が abb の場合，可能な二つの状態遷位系列にたいして式で示したように各々5 回の乗散が必要であった．また、$P(abb)$ の計算回数は 10 回の乗散と 1 回の加算であった．この計算は次のように書き換えられる．

$$P(abb) = [P(ab, S_1 \to S_1 \to S_2) + P(ab, S_1 \to S_2 \to S_2)] \times P(b, S_2 \to S_3) \qquad (3.27)$$

このようにすると，全計算回数は 8 回の乗散と 1 回の加算ですむ．一般的に次に示す前向きパスアルゴリズムによって効率良く $P(y_1, y_2, ..., y_T)$ を求めることができる．ここで $\alpha(i,t)$ を $y_1, y_2, ..., y_T$を生成して状態 i に達する確率としよう。また，F を最終状態の集合とする。すると，

$$P(y_1, y_2, ..., y_T) = \sum_{i \in F} \alpha(i, T) \qquad (3.28)$$

となる．$\alpha(i,T)$ の算出アルゴリズムを示す．

前向きパスアルゴリズム

(1) 初期化

すべての状態 i に対して$\alpha(i,0) = \pi_i$

(2)$t = 1, 2, ..., T$に対して (3),(4) を実行.

(3) すべての状態 i に対して (4) を実行.

(4)$\alpha(i,t) = \sum_{j(\text{ナル遷位を除く})} \alpha(j,t-1)a_{ji}b_{ji}(y_t) + \sum_{j(\text{ナル遷位})} \alpha(j,t)a_{ji}$

(5)$P(y_1, y_2, ..., y_T) = \sum_{i \in F} \alpha(i,T)$

これは、出力シンボル系列に対応する時間経過を横軸に，各状態を縦に並べて許される状態遷位を示すトレリス (trellis) の上で考えると理解しやすい．図 3.7は図 3.4の HMM で出力シンボル系列が abb の場合のトレリス上での$\alpha(i,t)$ の値を示している．これより $P(abb) = \alpha(S_3, 3) = 0.1232$ となる事が理解できるだろう.

3.5.2 EM アルゴリズム

HMM の最尤推定は，出力シンボル系列に対し，状態系列が直接観測できず，各状態遷位が何回生じたか数えられないので容易ではない.

これは、観測データが不完全な場合の最尤推定と本質的に同じ問題を含んでいる．不完全な観測データ yに対して尤度関数 $L(\theta; y)$ を最大にするパラメータθを決定するために yに付随して得られる付加データ x を用いる場合を考えよう. yと x を用いて推定されたパラメータ$\tilde{\theta}$と初期値θに対して $Baum$ は次の重要な定理を証明した.

$$L(\tilde{\theta}; y) \geq L(\theta; y) \tag{3.29}$$

ここで、等号が成立する必要十分条件は$\theta 0 \tilde{\theta}$である．故に，このをあらたにとしての推定を繰り返せばある値 (最適値または局所的最適値) に収束することになる。この手法は EM(Expectation-Maximization) アルゴリズムとしてよく知られてるものである．まず最初に，次のようにして尤度関数に非観測データ x を導入しよう.

$$L(\hat{\theta}; y) = L(\hat{\theta}; y) \cdot \frac{L(\hat{\theta}; x, y)}{L(\hat{\theta}; x, y)} = \frac{L(\hat{\theta}; x, y)}{L(\hat{\theta}; x|y)} \tag{3.30}$$

確率変数 X のすべてにわたって、あらかじめ設定されている推定値を使って期待値をとると,

$$\log L(\hat{\theta};y) = \log L(\hat{\theta};x,y) - logL(\hat{\theta};x|y) \tag{3.31}$$

$$E_\theta[\log L(\hat{\theta};y)]_{X|y} = \sum_x L(\theta;x|y)\log L(\hat{\theta};y) = \log(L(\hat{\theta};y) \tag{3.32}$$

これから、

$$\log L(\hat{\theta};y) = E_\theta[\log L(\hat{\theta};X,y)]_{X|y} - E_\theta - [\log L(\hat{\theta};X|y)]_{X|y} \tag{3.33}$$

が得られる. また,

$$\begin{aligned} E_\theta[\log L(\hat{\theta};X|y)]_{X|y} &= \sum_x L(\hat{\theta};x|y)\log L(\hat{\theta};x|y) \qquad (3.34)\\ &\le \sum_x L(\theta;x|y)\log L(\theta;x|y)\\ &= E_\theta[\log L(\theta;X|y)]_{X|y} \end{aligned}$$

(3.35)

なる関係が成立する. (等号は$\theta = \hat{\theta}$のとき) から, もし

$$E_\theta[\log L(\hat{\theta};X,y)]_{X|y} \ge E_\theta[\log L(\theta;X,y)]_{X|y} \tag{3.36}$$

になるような$\hat{\theta}$を用いれば,

$$\log L(\hat{\theta};y) \ge \log L(\theta;y) \tag{3.37}$$

となる. 以上より, 次のアルゴリズムは尤度関数の局所最大値または鞍点に収束する事がわかる.

EM アルゴリズム

(1) パラメータθの初期値を設定

(2)$E_\theta[\log L(\hat{\theta};X,y)]_{X|y}$を最大にする$\hat{\theta}$を最尤推定

(3)θを$\hat{\theta}$に設定

(4) 収束条件が満たされなければステップ (2) へ、満たされれば終了

ステップ (2) は期待値操作 (Expectation Step)，ステップ (3) は最大値操作 (Maximization Step) と呼ばれており、EM の名の由来はここから来ている．

3.5.3 パラメータ推定アルゴリズム

次に，EM アルゴリズムを HMM のパラメータ推定に適用する場合を考えてみる．HMM では，観測されるデータ yはモデルからの出力シンボルだけである．yに付随して得られる非観測データ x は状態遷位系列である．まず，ステップ (1) で、初期パラメータ (初期状態確率π_0，状態遷位確率 A，出力シンボル確率 B) を設定する．ステップは次の二つの部分からなる．まず，各時間 (フレーム) ごとに, y の生成時にとられる各状態確率を求める．次に，これらの確率を用いて $E_\theta[\log L(\hat{\theta}; X, y)]_{X|y}$を最大にするパラメータの更新値$\hat{\theta}$を最尤推定によって求める．以下、その具体的アルゴリズムを求める．

まず、先に定義した$\alpha(i,t)$ と双対な $beta(i,t)$ を状態 i から始まる状態遷位に よって y_{t+1}^T が生じる確率としよう.$\beta(i,t)$ は 次の後ろ向きパスアルゴリズムによって求めることができる．

後ろ向きパスアルゴリズム

(1) 初期化

$$\beta(i,T) = 1 \; if \; i \in F; \; \beta(i,T) = 0 \; if \; i \notin F$$

(2)$t = T, T-1, ..., 2, 1, 0$ に対して (3),(4) を実行.

(3) すべての状態 i に対して (4) を実行.

(4)$\beta(i,t) = \sum_{j(\text{ナル遷位を除く})} \beta(j,t+1) a_{ij} b_{ij}(y_{t+1}) + \sum_{j(\text{ナル遷位})} \alpha(j,t) a_{ij}$

定義から明らかなように，

$$\sum_{i \in F} \alpha(i,T0 = \sum_i \beta(i,0) \pi_i \tag{3.38}$$

ここで,πは状態 i の初期状態である．

このとき、次の関係式が成立する.

$$P(Y_1^T = y_1^T, X_{t-1} = i, X_t = j) = \alpha(i, t-1) \cdot a_{ij} \cdot b_{ij}(y_t) \cdot \beta(j, t) \tag{3.39}$$

$$\beta(i, t) = \sum_j a_{ij} b_{ij}(y_{t+1}) \beta(j, t+1) \tag{3.40}$$

図3.4にたいする後ろ向きパスアルゴリズムの計算例を図3.8に示す. 以上のような前向きパス、後ろ向きパスアルゴリズムでα, β を求めるアルゴリズムをForwardBackward アルゴリズムと呼ぶ.

次に, 出力シンボル系列$y = y_1, y_2, ..., y_T$に対して状態iから状態jへの遷位が時刻tで生じた確率を$\gamma(i, j, t)$ と記せば,

$$\begin{aligned} \gamma(i, j, t) &= P(X_{t-1} = i, X_t = j | y_1^T) \\ &= \frac{\alpha(i, t-1) a_{ij} b_{ij}(y_t) \cdot \beta(j, t)}{\sum_{i \in F} \alpha(i, T)} \\ &= \frac{\alpha(i, t-1) a_{ij} b_{ij}(y_t) \cdot \beta(j, t)}{\sum_i \alpha(i, t) \beta(i, t)} \end{aligned} \tag{3.41}$$

となる.αに関して前向きパスアルゴリズム,βに関して後ろ向きパスアルゴリズムを用いれば, 上式はTの線形オーダで計算できる. パラメータθを用いて求められたα, β, γの確率を用いて, 次式を最大にするパラメータθを求めればよい.

$$E_\theta[\log L(\hat{\theta}; X, y)]_{X|y} = \sum_x L(\theta; x|y) \log L(\hat{\theta}; x, y) \tag{3.42}$$

これは, 次式を最大にするθと等価である.

$$\prod_x L(\hat{\theta}; x, y)^{L(\theta; x|y)} \tag{3.43}$$

これは、与えられたyに対し状態遷位系列xが期待値として$L(\theta; x|y)$ 回生じたと解釈した場合, 新たな尤度関数になっていると考えられよう. 上式を最大にする初期状態iの確率は, 時刻1で状態iをとる次の期待値によって与えられる.

$$\hat{\pi} = \frac{\sum_j \gamma(i, j, 1)}{\sum_{i,j} \gamma(i, j, 1)} \tag{3.44}$$

また、状態遷位確率は, 状態iから状態jへの遷位をとる次の期待値によって与えられる.

$$\hat{a}_{ij} = \frac{\sum_t \gamma(i,j,t)}{\sum_j \sum_t \gamma(i,j,t)} \tag{3.45}$$
$$= \frac{\sum_t \alpha(i,t-1)a_{ij}b_{ij}(y_t)\cdot\beta(j,t)}{\sum_t \alpha(i,t)\beta(i,t)}$$
$$= \frac{a_{ij}\partial P/\partial a_{ji}}{\sum_j a_{ij}\partial P/\partial a_{ji}}$$

最後に、出力集合 $b_{ij}(k)$ に関しては,

$$\hat{b}_{ij}(k) = \frac{\sum_{t:y_t=k} \alpha(i,t-1)a_{ij}b_{ij}(y_t)\beta(j,t)}{\sum_t \alpha(i,t-1)a_{ij}b_{ij}(y_t)\beta(j,t)} \tag{3.46}$$
$$= \frac{b_{ij}(k)\partial P/\partial b_{ij}(k)}{\sum_k b_{ij}(k)\partial P/\partial b_{ij}(k)} \tag{3.47}$$

で与えられる．ここで,

$$P(y_1^T) = P(\{a_{ij}, b_{ij}(k)\}) = \sum_i \alpha(i,t)\beta(i,t) \tag{3.48}$$

Baum らは一般に $\sum Z_i = 1$ なる条件で, Z_iの次の多変数関数 $P(\{Z_i\})$ は,

$$P(\{Z_i\}) = \sum_{\mu_1,\mu_2,\cdots,\mu_n} C_{\mu_1,\mu_2,\cdots,\mu_n} Z_1\mu_1 Z_2\mu_2,\cdots,Z_n\mu_n(\text{但し},\mu_1+\mu_2+\ldots\mu_n = d) \tag{3.49}$$

$$\{\hat{Z}_i\} = \frac{Z_i\partial P/\partial Z_i}{sum_j Z_j\partial P/\partial Z_j} \tag{3.50}$$

と写像する事によって極大値または鞍点に収束する事を証明している．式から，$a_{ij} \neq 0$ なら $\hat{a}_{ij} \neq 0$ ，$a_{ij} = 0$ なら $\hat{a}_{ij} = 0$ となり，HMM の構造 (状態遷位の入出力関係) は保存されながら学習される事がわかる．

なお、ある状態遷位確率 a_{ij}と他の状態遷位確率 a_{mn}などを等しくおく HMMの場合には、これらの状態遷位 $i \to j$と $m \to n$ の関係は結び (tied) という．これらの i,m の集合を A_i，これらの状態遷位集合を T_{ij}で記せば,

$$\hat{a}_{ij} = \frac{\sum_{m\to n\in T_{ij}} \sum_t \gamma(m,n,t)}{\sum_{m\in S_i} \sum_n \sum_t \gamma(m,n,t)} \tag{3.51}$$

となる．一方，同様に $b_{ij}(k)$ と $b_{mn}(k)$ をすべての kについて等しくおく HMMの場合は，出力確率に対して「結ばれている」という．この様な $\{i,j\}\{m,n\}$ を D_{ij}と記す．この様な場合は各 D_{ij}に関して次式を最大にするような出力確率分布を求めればよい.

$$g(\phi) = \sum_{m\to n\in D_{ij}} \sum_t \gamma(m,n,t)\cdot\log P_\phi(Y_t = y_t) \tag{3.52}$$

離散分布の場合は,

$$b_\phi(k) = P_\phi(y_t = k); \sum_k b_\phi(k) = 1 \tag{3.53}$$

である．次のラグランジュの未定乗数法を用いると,

$$\frac{\partial}{\partial b_\phi(k)}[\sum_{m\to n\in D_{ij}} \sum_t \gamma(m,n,t) \cdot \log P_\phi(Y_t = y_t) - \lambda(\sum_k b_\phi(k) - 1)] = 0 \tag{3.54}$$

から,

$$\hat{b}_\phi(k) = \frac{\sum_{m\to n\in D_{ij}} \sum_{t;y_t=k} \gamma(m,n,t)}{\sum_{m\to n\in D_{ij}} \sum_t \gamma(m,n,t)} \tag{3.55}$$

となる．

上述のパラメータ推定値の更新アルゴリズムは Baum によって提案されたことから、Baum アルゴリズムあるいは Baum-Welch アルゴリズムと呼ばれている．本アルゴリズムは Forward-Backward アルゴリズムを基本としているから単に Forward-Backward アルゴリズムと呼ぶ事もある．

なお、最尤推定のための EM アルゴリズムは与えられた全観測サンプル系列 $y = y_1, y_2, ..., y_T$からのパラメータ推定法であった事に注意を要する．一般に形態素解析のための HMM のパラメータ推定には数多くの観測サンプル系列 $\{y^n = y_1^n, y_2^n, ..., y_{T_N}^n\}$ を訓練サンプルとする．厳密に EM アルゴリズムを適用するためには，これらの訓練サンプル1セットごとにパラメータを再推定しなければならない．すなわち $P(y^1, y^2, ..., y^N) = P(y^1 \cdot y^2 \cdot \ldots \cdot P(y^N)$ を最大にするようにパラメータθを推定しなければならない．この場合は y^nによる $\alpha(i,t), \beta(i,t), \gamma(i,j,t)$ をそれぞれ $\alpha^n(i,t), \beta^n(i,t), \gamma^n(i,j,t)$ とすれば,

$$\hat{a}_{ij} = \frac{\sum_n \sum_t \gamma^n(i,j,t)}{\sum_n \sum_j \sum_t \gamma^n(i,j,t)} = \frac{\sum_n \sum t\alpha^n(i,t-1) a_{ij} b_{ij}(y_t^n) \beta^n(j,t)}{\sum_n \sum_t \alpha^n(i,t)\beta^n(i,t)} \tag{3.56}$$

$$\hat{b}_{ij} = \frac{\sum_n \sum_{t:y_t=k} \gamma^n(i,j,t)}{\sum_n \sum_t \gamma^n(i,j,t)} = \frac{\sum_n \sum_{t:y_t=k} \alpha^n(i,t-1) a_{ij} b_{ij}(y_t^n) \beta(j,t)}{\sum_n \sum_t \alpha^n(i,t-1) a_{ij} b_{ij}(y_t^n \beta^n(j,t)} \tag{3.57}$$

となる．

Forward-Backward アルゴリズムにおける$\alpha(i,t), \beta(i,t)$ を精度よく能率良く計算できる事は重要である．$\alpha(i,t)$ は時間 t とともに小さな値となりアンダーフローが生じないようにスケーリングが必要である．$\beta(i,t)$ についても同様であり，その値のダイナミックレンジは 1,000 のオーダに達する．

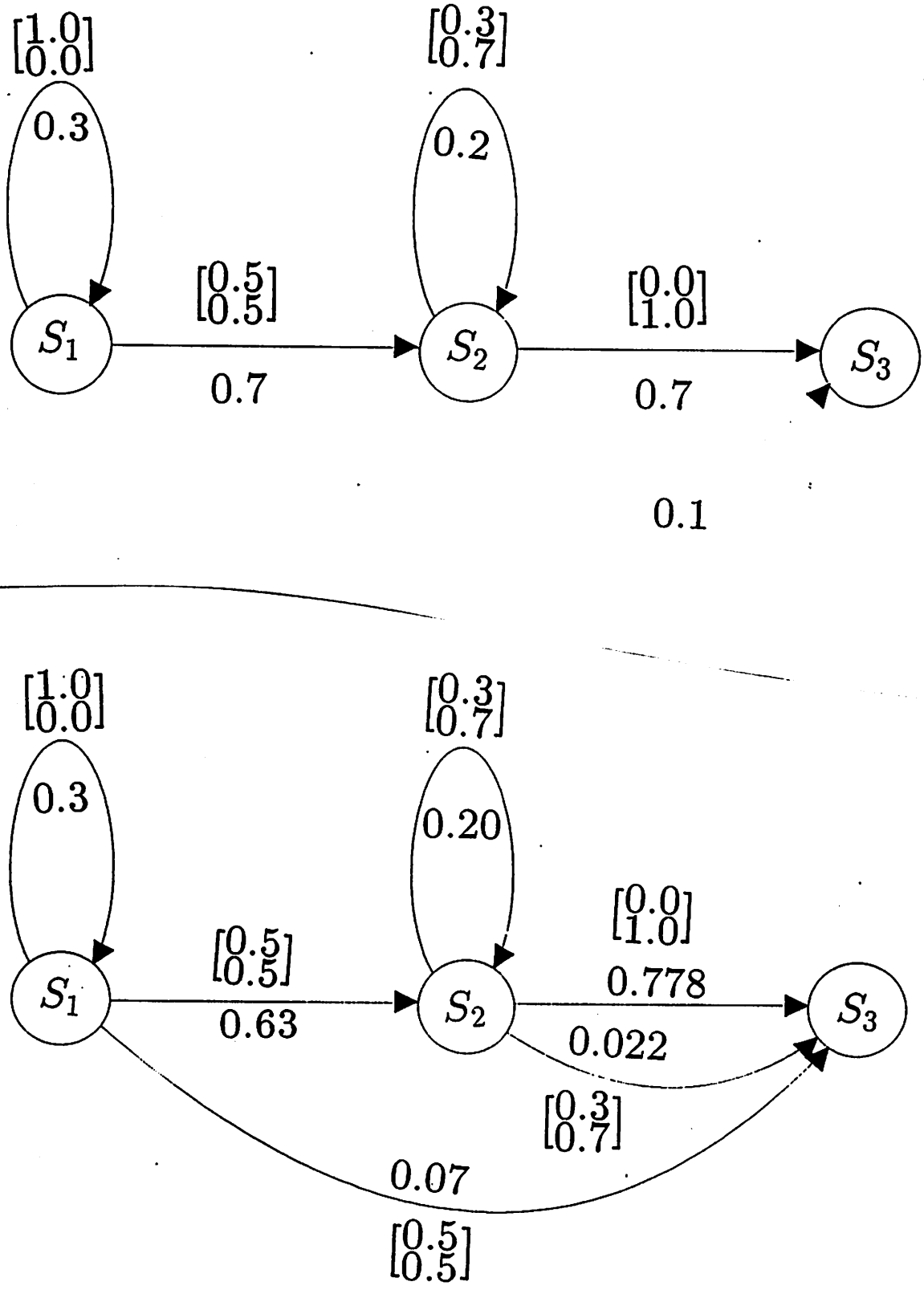

Figure 3.5: ナル遷位の除去

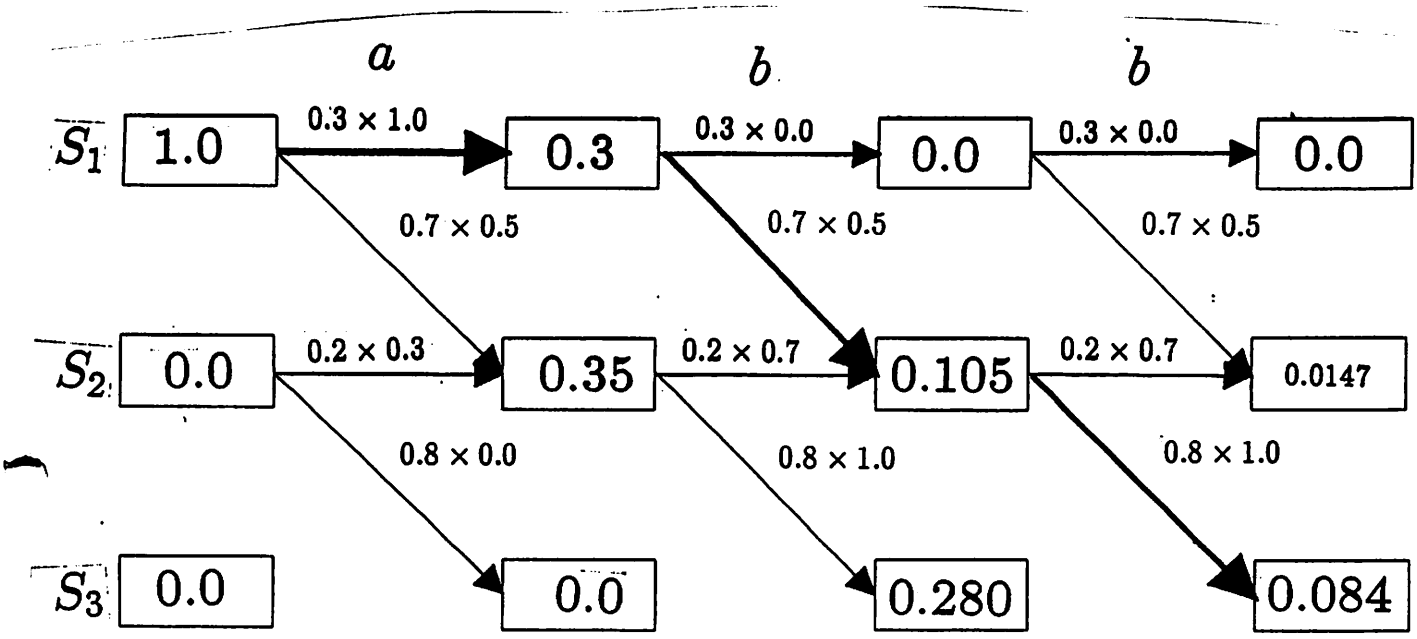

Figure 3.6: HMM におけるシンボル系列 abb に対する 最適パス算出例

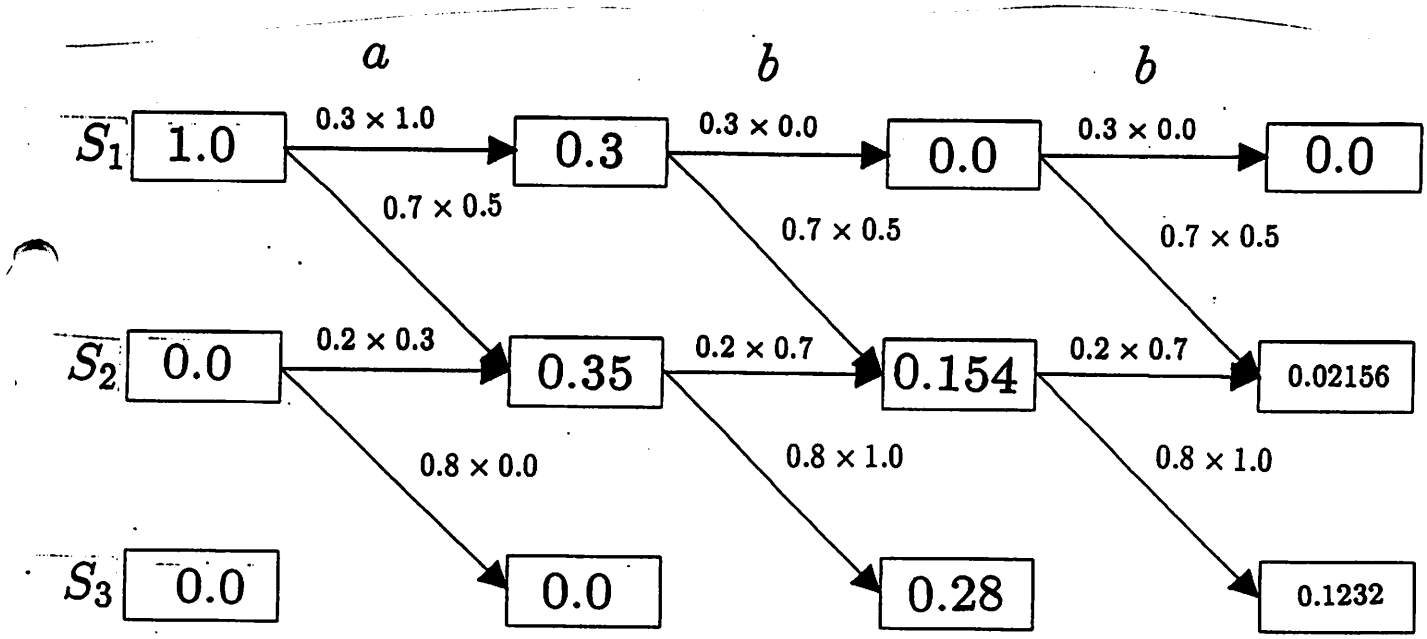

Figure 3.7: $P(abb)$ のトレリス上の計算

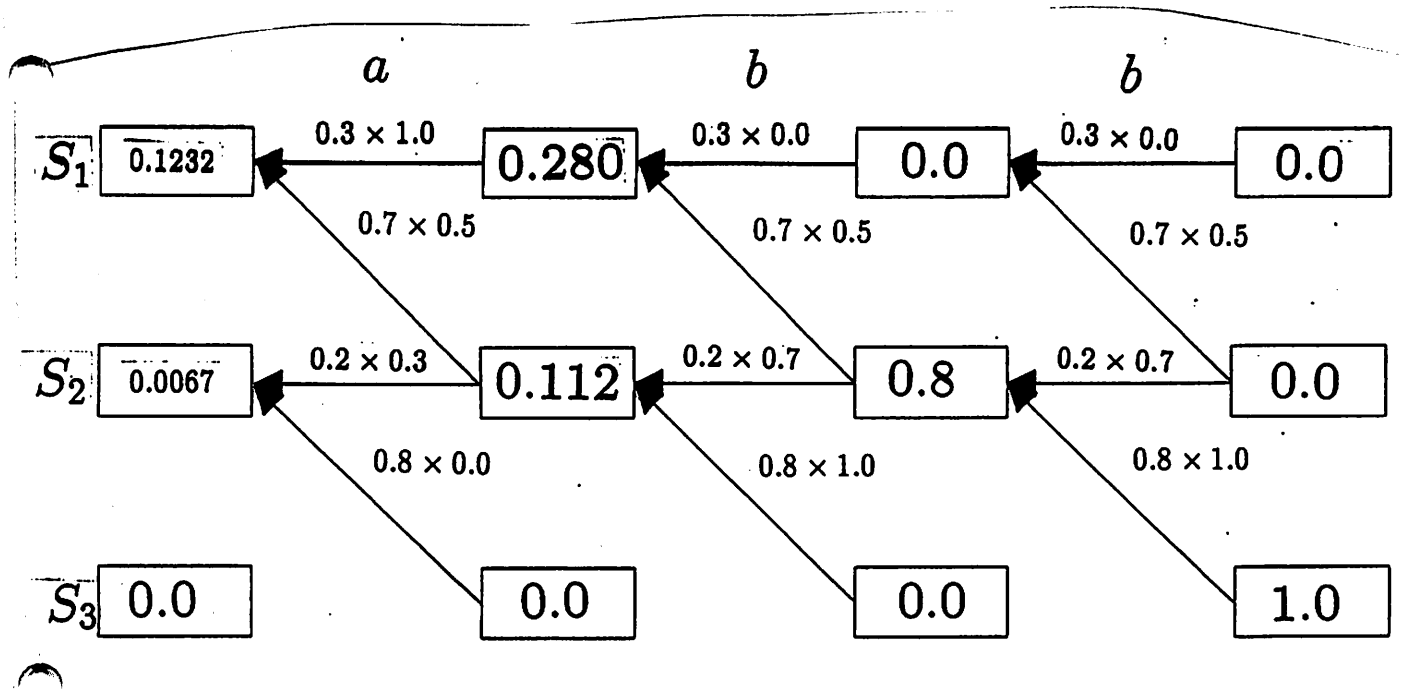

Figure 3.8: HMM における $P(abb)$ に対する後ろ向きパスアルゴリズム

Chapter 4

HMMの形態素解析への応用

4.1 HMMによる形態素解析

HMMのモデルを用いて形態素解析を解くことを考えてみよう．HMMは状態から状態への遷位の際にシンボルを出力する．これを形態素解析で利用するには，自然言語の文は品詞の列とみなす．

いま文を構成するそれぞれの単語の品詞をHMMの状態と考え，単語の字面を出力シンボルと考えると，文の先頭から文末まで品詞の状態を渡り歩き，その際に単語の字面を出力することで，ある文が出力されるモデルが作られる．

このように捉えると，HMMは自然言語を生成するモデルになっている．

はじめにで述べたように，英語の形態素解析は品詞付けに対応する．いま「I like English」という文を形態素解析する場合，それぞれの単語には以下のような品詞が存在し，正しい品詞を選定するのが目的の処理となる．

この例をHMMというモデルで考えよう。名詞や動詞、形容詞といった品詞をHMMの状態、「I」や「like」、「English」という単語をHMMの出力シンボルと置き換える。

するとこの問題は、与えられた出力シンボル系列「I like English」を生じさせる遷移状態列（品詞列）のなかで，最大の確率を持つ品詞列を求める問題に置き換えられる．これは先ほど述べたviterbiアルゴリズムで解くことができる．つまりHMMのパラメータを適切に設定すれば，形態素解析の問題はHMMのviterbiアルゴリズムで解くことができることがわかる．

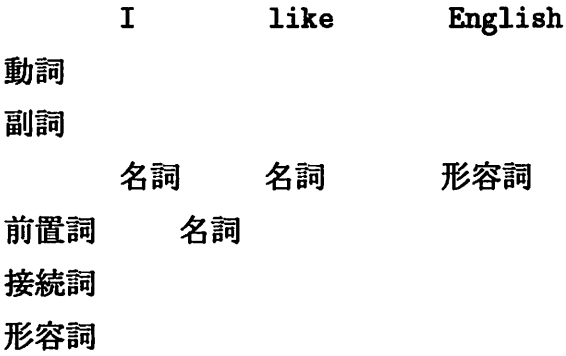

Figure 4.1: HMM の形態素解析への応用

4.2 HMM による形態素解析の長所と短所

HMM の長所はパラメータが自動学習できる点である.

日記などに使われる文章と法律などで扱う文章のように文章の分野が異なる場合は、文の構造が微妙に違う。このため従来の形態素解析解析では文章の分野ごとのデータを作成する必要がある。だがHMMを用いると一つの分野のパラメータしか持っていない場合でも、目的とする分野の文章を学習させる事により容易にその分野の文章に柔軟に対応させることが出来る.

また以前の形態素解析ではおよそ経験的な規則により解析しているために，なぜこの規則により形態素解析を行い良い結果が出るのかという証明が難しい．だがHMM ではパラメータをすべて確率で持っているためにその様な証明も比較的容易にできる.

逆に問題点としてはHMMの学習には膨大な例文と時間が必要となり，そのため満足な精度を出すためにはかなりの手間と時間を必要とすることがあげられる.

4.3 既存知識の利用

HMM のパラメータ学習にともなう問題の解決のための 1 つとして，既存知識をパラメータ学習の初期パラメータに反映させることを考えている，これによって学習量を減らすことができると予想される．本研究では右方向最長一致法，字種切り法，ほかに文節数最小法からの応用を考えている.

右方向最長一致法は文を左から右の向きに見て，最も長い形態素を優先させて分割する

ものである．HMM への応用としては長い形態素への遷移は確立が高いと考える．

字種切り法では，平仮名から他の字種への変わり目，区切り符合の前後，非平仮名列から数字列への変わり目，数字列から非平仮名列への変わり目に切れ目を入れて形態素に分ける．このことから，字種が変わるような遷移は確率が高いと考える．

文節数最小法は入力文を単語列に分けた時に，単語列の少ない方を優先させる．これから，なるべく単語列を少なくさせるような遷移は確率を高くする．

これらの知識を応用して HMM の初期パラメータに反映させる．

Chapter 5

HMM と Viterbi アルゴリズムによる形態素解析

5.1 方法

前述したように、HMM における状態と出力シンボルは，形態素解析においてはそれぞれ品詞と単語に相当する．品詞は図 5.1で示すように 24 に分け，それぞれを設定した．

HMM には 6 個のパラメータが必要である．今回は以下のようなパラメータを作成した．

- 出力シンボルの集合 outset.dat

  出力シンボルである単語を、入力する文に現れる単語をすべて登録する．

- 出力シンボルの集合及び品詞辞書 outset.dic

  outset.dat にある単語すべてに品詞の情報を持たせる．厳密に HMM のアルゴリズムに従うには状態 s_iから状態 s_jへの遷位の際に出力シンボル kを出力する確率を持たなければならないが，今回は処理の軽減とデータ量の減少のために状態 s_jへの遷位の際に出力シンボル kを出力する確率を持つ事にした．さらにこの確率を 0 か 1 で表したので，実際には出力シンボル kが状態 s_jであるかどうか の 2 値を記述してあることになる。具体的には単語の後に続く 2 バイトの数字で品詞を表している．ある単語がある品詞をとるならばそのビットを立てる．このようにする事で日本語の特色でもある多品詞に対応させる事ができる．

- 状態遷位確率の集合 sptran1.dat sptarn2.dat

状態遷位確率の集合は通常の初期パラメータであるすべての状態に等確率で遷移する状態遷位確率の集合 sptran1.dat と，既存の知識をパラメータに反映させた状態遷位確率の集合 sptran2.dat の二つを作成した.

- 初期状態の集合 inistate.dat

  24 ある品詞のうち、文の先頭にあると思われるものに確率を与え，それ以外を 0 とした。

- 最終状態の集合 final.dat

  品詞は 24 あるが必ず句点でおわるようにしたために、句点の番号のみを 1，それ以外を 0 とした。

このパラメータを viterbi アルゴリズムにより解析し，sptran1.dat sptran2.dat の二つのパラメータの違いによる変化を比較する.

ここではそのために例文を 8 文用意した.

入力文の構造

入力文	正答性	長さ
1 かれ が くる まで はこ を はこ び 升 .	×	10
2 かれ が くる までは こ を はこ び 升 .	×	10
3 かれ が くる まで はこ を はこび 升 .	×	9
4 かれ が くる まで はこ を はこび ます .	○	9
5 かれ が くる までは こ を はこび ます .	△	9
6 かれ が くるま で はこ を はこび ます .	○	9
7 かれ が くるま では こ を はこび ます .	△	9
8 かれ が くるま では こ を はこび 升 .	×	9

正答性とは，人間が見て，正しいと思われるものを○，ほぼ正しいと思われるものを△，間違いであるものを×としている.

長さは文章の単語の数である．これには句点も含まれている．この文を Viterbi アルゴリズムにて解析する.

単語番号	品詞	例
0	句点	。
1	読点	、
2	記号	「 」 （ ）
3	動詞	食べる，する，ある，立つ
4	形容詞	美しい，悲しい，
5	判定詞	だ
6	助動詞	ます，させる，られる，ことだ，つもりだ，です
7	名詞	茨城，りんご，カレンダー，ジェフリー
8	数詞	1，2，3，4
9	形式名詞	こと もの、とき、はず、ため
10	副詞的名詞	ため，おかげ，まま，うち，まえ
11	指示詞	この，あの，その
12	副詞	いきなり，かねて、はっきり、しばらく、きっと
13	格助詞	が，から，を，に，へ、の
14	副助詞	は，こそ
15	引用助詞	と
16	名詞接続助詞	と，そして，並びに，および，や
17	述語接続助詞	まで，より，とか
18	終助詞	か，かしら，ね
19	接続詞	あわせて，一方，かくして，しかし
20	連体詞	ある，いかなる，いろんな，おかしな
21	感動詞	ああ，あら
22	接頭辞	御，小，大，第，再
23	接尾辞	個，冊，匹

Figure 5.1: 24 に分類した単語品詞番号と品詞

Chapter 6

解析結果

	sptran1.dat		sptran2.dat	
文	確率	遷移パス	確率	遷移パス
1	-36.700	wrong	fault	fault
2	-36.700	wrong	fault	fault
3	-33.030	true	fault	fault
4	-33.030	true	-22.918	true
5	-33.030	true	-22.918	true
6	-33.030	true	-22.838	true
7	-33.030	true	-22.656	true
8	-33.030	true	fault	fault

Figure 6.1: 二つのパラメータ sptran1.dat と sptran2.dat の解析結果

viterbi アルゴリズムはその仕様により遷位パスが深いほど確率が小さくなる．このため sptran1.dat で解析した 1,2 文は他の文に比べて確率が低くなっている．

sptran1.dat では間違った文においても遷位パスが出てしまい，正解と不正解の区別が付かない．また状態遷位確率は文の長さにより変わるのみで，どのような遷位パスでも変わることはなかった．

sptarn2.dat では 1,2,3,8 の文では遷位パスを出力せず，この文は不正解であることが解った．遷位パスが出る他の文では 7 の文が一番状態遷位確率が高い結果が出ている．これは格助詞→名詞 と 副助詞→名詞 の遷位確率は 副助詞→名詞 の方が確率が高いためにこのような結果になったと思われる．

4,5 は同じ結果が出たが，これは状態遷位パスが二つとも同じためである．今回の実験ではデータの簡略化と計算量の縮小のために出力確率の集合を，本来は状態から状態への遷位の際にシンボルを出力する確率であるところを，状態への遷位の際にシンボルを出力する確率とした．このため出力シンボルが違っても遷移パスが同じ場合には同じ状態遷移確立を得る．

また，この結果を以下の方法で検討した．入力文 i が順位 k であるとき，本当の答が○ならば d=k-1.5、△ならば d=k-3.5、×ならば d=k-6.5 をすべての i について行ない，d を求め両者を比較する．d の値の小さい方が正答率が良いと言える．

	sptran1.dat	sptran2.dat
d	17.0	4.0

この結果 sptran2.dat の方が正答率が良いことが解った．

今回既存知識を応用して作成したパラメータ sptran2.dat では名詞→句点への遷位確率は 0 になるようにしたために文 1,2,3,8 の、名詞から句点への遷位でおわる文章は遷位パスを得る事ができていない．

6.1 まとめ

本研究の目的はHMMを形態素解析に利用する際にパラメータ学習を効率良く行なうことである．そのための方法として既存の知識を応用して初期パラメータを作成して学習のための時間を少なくした。

viterbiアルゴリズムによる形態素解析プログラムを作成し，実際に例文を解析させ，最適パスとその時の状態遷位確率を求めることができた．その結果なにも学習させていない初期パラメータよりも既存の知識を応用して作成した初期パラメータの方がよりよい状態遷位確率と正しい遷位パスを導き出すことが出来ることが解った．今回は通常のHMMで使われる初期状態と既存の知識を用いて作成した初期状態を相対的に見て正答率を調べたが，本来は既存の知識を応用して作成したsptran2.datがどの程度の学習量の節約になったかを調べるべきである．

謝辞

本論文の遂行にあたり、御指導下さった茨城大学工学部システム工学科新納浩幸教官に深く感謝の意を表します。またシステム工学科計算機応用講座の教官の方々にも深く感謝致します。

Bibliography

[1] 田中穂積：“自然言語解析の基礎”, 産業図書, 東京, 1989.

[2] 中川聖一：“確率モデルによる音声認識”, 電子情報通信学会, 東京, 1989.

[3] 永田昌明：“自然言語の確率モデルと統計的学習”, 電子情報通信学会, 日本ソフトウエア科学会共催「自然言語処理における学習」シンポジウム論文集,pp.182–189(1994).

[4] 竹内孝一, 松本裕治：“HMM を用いた形態素解析のパラメータ学習”, 情報処理学会第 50 全国大会, **1R-4**(1995).

付録

final.dat

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

inistate.dat

0 0 0.142 0.142 0.142 0.1 0 0.142 0.142 0 0 0.142 0.142 0.1 0 0 0 0 0 0 0 0 0 0

outset.dat

かれ
が
くる
くるま
まで
までは
では
で
こ
はこ
はこび

を
び
ます
升
。

outset.dic

かれ 128
が 8192
くる 8
くるま 128
まで 131072
までは 131072
では 16384
で 8192
こ 128
はこ 128
はこび 8
を 8192
び 128
ます 64
升 128
。 1

sptran1.dat

24×24 の行列からなるファイルで，それぞれの要素はすべて $1/24 = 0.042$ である．

sptran2.dat

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0.16 0 0 0 0.16 0.16 0 0 0.16 0.16 0 0 0 0 0 0 0 0 0 0.16 0
0.111 0 0.111 0 0 0.111 0 0.111 0 0.111 0 0 0 0.111 0.111 0.111 0 0 0 0 0 0
0 0.111
0.0769 0.0769 0.0769 0.0769 0.0769 0.01 0.0769 0.0769 0 0.0769 0.0769 0 0.076
0.0769 0 0.0769 0 0.0769 0 0 0 0 0 0
0 0 0 0 0 0 0 0.5 0 0 0 0 0 0 0 0.5 0 0 0 0 0 0 0 0
0.33 0.33 0 0 0 0 0 0.33 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.25 0 0 0.25 0 0 0 0.25 0 0 0 0 0.25 0 0 0 0 0 0 0 0 0 0 0
0 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.125 0.0625 0 0 0 0 0.0625 0.125
0.0625 0.125 0.0625 0 0 0 0 0.0625 0.0625
0 0 0 0 0 0 0 0.2 0.4 0 0 0 0 0 0 0 0 0 0 0 0 0 0.4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.5 0.5 0 0 0 0 0 0 0 0 0
0 0.5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.5 0 0 0 0 0 0
0 0 0 0 0 0 0 0.5 0 0 0 0 0 0.5 0 0 0 0 0 0 0 0 0 0
0 0.1666 0 0.1666 0.1666 0.1666 0 0.1666 0 0 0 0.1666 0 0 0 0 0 0 0 0 0 0 0
0
0 0.0833 0.0833 0.0833 0.0833 0.0833 0 0.1666 0.0833 0 0 0 0.0833 0.0833 0.08
0 0 0 0.0833 0 0 0 0 0
0 0.1428 0.1428 0.1428 0.1428 0 0 0.1 0 0 0 0.1428 0.1428 0 0 0 0.1428 0 0 0
0 0 0 0
0 0 0.3333 0.3333 0.3333 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0.1666 0.1666 0 0 0 0.3333 0 0 0.1666 0 0 0 0 0 0 0 0 0 0 0 0.1666 0
0 0 0 0.25 0 0 0 0.25 0 0.25 0 0 0 0 0.25 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0.25 0 0 0.25 0 0 0 0.25 0 0 0.25 0 0 0 0 0 0 0 0 0

0 0.1666 0 0 0 0.1666 0 0.1666 0 0 0 0 0 0.1666 0 0.1666 0 0.1666 0 0 0 0 0

0

viter.c

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <float.h>

#define N 24 /*品詞(状態)の総数*/
#define TT 68 /*入力単語数の最大値*/
#define SYUTURYOKU_NUMBER 310
#define BUN_MAX 68 /* 入力単語数の最大数 */
#define LOG_LIMIT -200000 /* ログの下限限界 */
#define LOG_MAX -0.0000000001 /* ログの最大値 */

struct Markov{ /*マルコフモデルの宣言 Y と B は抜かしてある*/
int S[N];
double A[N][N];
double PAI[N];
double F[N];
};

typedef struct Markov MK;

MK VIT_1(MK ,double [][]);
MK VIT_2(MK,double [][],int [],int [][][]);
MK VIT_5(MK,double [][],int [][][]);
MK DATA_READ(MK);
double B_get(int,int,int,int []);
void NYURYOKU_READ(int [],char *); /* 入力文を読みます */
void Q_plus(int [],int [],int Jh); /* Q の配列に足し込む関数 */
void quit(char *);
```

```
int T;

void main(int argc, char *argv[])
{
        extern int T;

int i,j,k,l;
MK MAR;
double alpha[N][TT];
int Q[N][TT][BUN_MAX]; /* 三次元配列で持つ 番号なので int 型 */
int BANGOU[BUN_MAX];

      if(argc != 3) quit("引数の数が違う"); /* prg file T */

      T = char2num(argv[2]);

       for(i=0;i<BUN_MAX;i++){ BANGOU[i]=-1;}

for(i=0;i<N;i++){ /* 変数 Q の初期化  */
for(j=0;j<T;j++){
for(k=0;k<BUN_MAX;k++){
Q[i][j][k]=-1;
}
}
}

NYURYOKU_READ(BANGOU,argv[1]);
MAR=DATA_READ(MAR);
MAR=VIT_1(MAR,alpha);
MAR=VIT_2(MAR,alpha,BANGOU,Q);
MAR=VIT_5(MAR,alpha,Q);
}
```

```
MK VIT_1(mar,alp)
  MK mar;
  double alp[N][TT];
{
  int i,j,k,l;
  for(i=0;i<N;i++){ for(j=0;j<T;j++){ alp[i][j]=LOG_LIMIT; } }
  for(i=0;i<N;i++){ alp[i][0]=mar.PAI[i]; }
  return mar;
}


MK VIT_2(mar,alp,Bangou,Q)
  MK mar;
  double alp[N][TT];
  int Bangou [BUN_MAX];
  int Q[N][TT][BUN_MAX];
{
   extern int T;
  int t,i,j,k,l,m,jj;
  double P,bg;
  int Jh=0;
  double AM[N];

  for(i=0;i<N;i++) AM[i]=0;
  for(t=1;t<T;t++){
     for(i=0;i<N;i++){
          for(j=0;j<N;j++){
             if (mar.A[j][i] > LOG_LIMIT) {
                 bg = B_get(j,i,t,Bangou);
                 if (bg != 0.0) {
                   AM[j] =(alp[j][t-1] + mar.A[j][i] + B_get(j,i,t,Bangou));
```

```
            } else {
              AM[j] = -100000000.0;
            }
          } else  {
            AM[j] = -100000000.0;
          }
        }
        Jh = 0; /* shinnou */
        for(k=0;k<N;k++){
            if (AM[Jh] > AM[k]) Jh=Jh; else Jh=k;
        }
        if (AM[Jh] != -100000000.0) {
          bg = B_get(Jh,i,t,Bangou);
          if (bg != 0.0) {
             alp[i][t]=(alp[Jh][t-1] + mar.A[Jh][i] + B_get(Jh,i,t,Bangou))
          } else {
             alp[i][t]= -100000000.0;
          }
          Q_plus(Q[i][t],Q[Jh][t-1],Jh);

          for(jj = 0; jj<N;jj++) {
            printf("Q_pre[%d][%d][%d] = %d\n",Jh,t-1,jj,Q[Jh][t-1][jj]);
            printf("Q_ato[%d][%d][%d] = %d\n",i,t,jj,Q[i][t][jj]);
          }

} else {
          alp[i][t] = -100000000000.0;
          for(jj =0;jj<N;jj++)  Q[i][t][jj] = -1;
          for(jj = 0; jj<N;jj++) {
            printf("Q_ato[%d][%d][%d] = %d\n",i,t,jj,Q[i][t][jj]);
          }
        }
```

```
    }
    for(i=0;i<N;i++) AM[i]=LOG_LIMIT;
  }
 return  mar;
}

MK VIT_5(mar,alp,Q)
 MK mar;
 double alp[N][TT];
 int Q[N][TT][BUN_MAX];
{
 extern int T;
 int i,j,k,l;
 int Ih=0;
 j=0;

 for(i=0;i<N;i++){ if(mar.F[Ih]>mar.F[i]) Ih=Ih; else Ih=i; }
 printf("Ih は%d です",Ih);
 printf("答え=%e\n",alp[Ih][T-1]);
 j=0;
 printf("遷位パスは");
 while(Q[Ih][T-1][j] != -1){
     printf("%d,",Q[Ih][T-1][j]); j++;
 }
 printf("\n");
 return mar;
}

void  NYURYOKU_READ(Bangou,filename)
  int Bangou[BUN_MAX];
  char *filename;
{
```

```
 extern int T;
FILE *fp1,*fp2,*fp3;
char syutu[BUN_MAX];
char Tango[BUN_MAX];
int l,m,n;
int i,j,k;
int r=0;

if((fp1=fopen("outan.dat","r")) == NULL){
  printf("出力集合ファイルがオープンできませんでした\n");
  exit(1);
}
if((fp2=fopen(filename,"r")) == NULL){
  printf("入力文ファイルがオープンできませんでした\n");
  exit(1);
}
while(fscanf(fp2,"%s",&Tango) != EOF){
  if(ferror(fp2)){
printf("read error\n"); exit(-1);
  }
  if(feof(fp2)) exit(-1);
  for(i=0;i < SYUTURYOKU_NUMBER;i++){
    if(fscanf(fp1,"%s",&syutu) == EOF){
      if(ferror(fp1)){
          printf("read error\n"); exit(-1);
      }
    }
    if(strcmp(Tango,syutu) == 0){
      Bangou[r]=i; r++;
      fseek(fp1,0,SEEK_SET); break;
    } else { } /* if out */
  } /* for out */
```

```
  } /* while out */
printf("while clear\n");
  fclose(fp1);
  fclose(fp2);

  printf("Bangou==");
  for(i=0;i<BUN_MAX;i++){
     if(Bangou[i]==-1) break;
     printf("%d,",Bangou[i]);
  }
  printf("\n");
}

MK DATA_READ(mar)
  MK mar;
{
  extern int T;
  FILE *fp;
  double a;
  int b,i,j;
  double log(double);

  if((fp=fopen("sptran2.dat","r")) == NULL){
     printf("sptran2.dat のオープンに失敗しました\n");
     exit(1);
  }
  for(i=0;i<N;i++){
     for(j=0;j<N;j++){
       fscanf(fp,"%lf",&a);
       if(a==0){
          mar.A[i][j]=LOG_LIMIT;
```

```
    } else {
        mar.A[i][j]=log(a);
    }
  }
}
fclose(fp);

if((fp=fopen("inistate.dat","r")) ==  NULL){
    printf("inistate.dat のオープンに失敗しました\n");
    exit(1);
}

for(i=0;i<N;i++){
  fscanf(fp,"%lf",&a);
  if(a==0){
       mar.PAI[i]=LOG_LIMIT;
  } else if(a==1){
       mar.PAI[i]=LOG_MAX;
  } else {
       mar.PAI[i]=log(a);
  }
  printf("mar.PAI[%d]=%f\n",i,mar.PAI[i]);
}
fclose(fp);

if((fp=fopen("final.dat","r")) == NULL){
  printf("final.dat のオープンに失敗しました\n");
  exit(1);
}
for(i=0;i<N;i++){
   fscanf(fp,"%lf",&a);
   if(a==0){
```

```
        mar.F[i]=LOG_LIMIT;
      } else {
        mar.F[i]=log(a);
      }
   }
   fclose(fp);

  return mar;
}

double B_get(int a,int b,int c,int Bangou [BUN_MAX])
{
  extern int T;
  double d;
  int p,i,j,k;
  unsigned int o=0,q=0;
  char bun[BUN_MAX];
  FILE *fp;
  d=0.1;
  q=a;
  if((fp=fopen("outan.dic","r")) == NULL){
     printf("出力集合.DIC のオープンに失敗しました\n");
     exit(1);
  }
  j=Bangou[c-1];
  do{
    fscanf(fp,"%s%d",bun,&o);
    j--;
  } while(j >= 0);
  fclose(fp);

  q=(unsigned int)pow(2,a);
```

```
  if(o & q) {
     return(log(0.5));
  }
  return 0.0;
}

void Q_plus(Q_new ,Q_old ,Jh )
  int Q_new[BUN_MAX];
  int Q_old[BUN_MAX];
  int Jh;
{
  extern int T;
  int i,j,k;
  i=0;
  k=0;
  i=0;
  while(Q_old[i] != -1){
     Q_new[i]=Q_old[i];
     i++;
  }
 printf("I is %d\n",i);
  Q_new[i]=Jh;
}

void quit(char *s)
{
  printf(s); putchar('\n');
  exit(1);
}
```

char2num.c

```
#include <stdio.h>

int char2num(char *str)
{
  int pos,sum;
  sum = 0;
  for(pos = 0; str[pos] != NULL; pos++)
     sum = sum*10 + char_to_num(str[pos]);
  return sum;
}

int char_to_num(char c)
{
  if ( c == '0') return 0;
  if ( c == '1') return 1;
  if ( c == '2') return 2;
  if ( c == '3') return 3;
  if ( c == '4') return 4;
  if ( c == '5') return 5;
  if ( c == '6') return 6;
  if ( c == '7') return 7;
  if ( c == '8') return 8;
  if ( c == '9') return 9;
}
```