論　文

グラフを利用して誤りを含む記号列を分類する方法とその音声認識への応用*

関口芳廣** 重永　実**

Abstract

Classification of symbol strings is one of the important problems in pattern recognition. For this problem, usually the input string is compared with some reference strings and identified with the reference string having the highest similarity. But when the input string contains some errors or some extra symbols, its classification becomes difficult. We propose a method which overcomes this difficulty by using graph theory, and explain the algorithm for its performance and verify the completeness of the algorithm. Next, we have applied this method to speech recognition and obtained good results. By comparing with another method using dynamic programming (so-called DP method), it has become clear that in our method it takes a little more time to classify input samples than the DP method; but when input samples contain some errors or extra symbols, the classification score of our method is much better than the DP method.

1. まえがき

パターン認識でよく取り上げられる問題の1つに記号列の分類の問題がある．この問題に対する常套手段は標準パターンを用意しておき，それらと入力パターンとのマッチングを行い最も類似していると思われるカテゴリーに分類する方法である．入力と標準パターンとのマッチングの方法としては評価関数を定義し，ダイナミック・プログラミングの手法などを用いて最良の解を得ようとするのが代表的なものであろう．しかし入力の記号列がいくつかの誤りと余分な部分を含んでいる場合，上記のような方法はあまり良い方法とはいい難い．そこで本論文ではグラフを利用して全体の記号列をみながら最適な解を得る方法を提案し，その実現のための簡単なアルゴリズムを示す．またアルゴリズムの完全性についても証明する．さらにこの方法を実際の音声認識に応用した場合の例にもふれる．すなわち実際の音声認識ではまず音声波を音素列（またはそれに類似な記号列）に変換し，その音素列と，用意してある辞書とのマッチングを行い，認識を進めて行くという方法がよく取られるが[1)~3)]，これらのシステムでは音素列の誤り，欠如などがしばしば見られる．これはある程度さけられないもので，これらの誤りや欠如が存在しても正確な認識が行われていくことが要求される．筆者らは本論文で提案する方法を音声認識システムに導入し非常によい結果を得ており，本方法が有効なことも実証済みである．

2. 二つの記号列間のマッチング

2.1 定　義

今後必要となるいくつかの定義を行う．

［定義1］ 入力記号列 $\{x_i\}$ とは分類されるべき対象となるデータのことで，それに含まれる記号の数を s とすると

$$\{x_i\}=(x_1, x_2, x_3, \cdots, x_s)$$

と表わされる．

［定義2］ 分類可能なカテゴリーの数は p 個で，各

* A Graph Theory Method of Classification of Symbol Strings with Some Errors and its Application to Speech Recognition by Yoshihiro SEKIGUCHI and Minoru SHIGENAGA (Department of Computer Science, Faculty of Engineering, Yamanashi University).

** 山梨大学工学部計算機科学科

カテゴリーには $1, 2, \cdots, p$ と番号がついている．l $(1 \leq l \leq p)$ の番号がついたカテゴリーをカテゴリー l という．

[定義3] カテゴリー l はそれを代表する標準記号列 $\{S_j{}^l\}$ を持っている．カテゴリー l の標準記号列の要素の数を m_l とすると

$$\{S_j{}^l\}=(S_1{}^l, S_2{}^l, S_3{}^l, \cdots, S_{m_l}{}^l)$$

と表わされる．

[定義4] 記号列 $\{\alpha_i\}$ と $\{\beta_j\}$ の順序つきマッチングとは **Fig. 1** に示すように要素 α_i と β_j をグラフの節点とみなし，$\alpha_i=\beta_j$ なる節点間を枝で結び，次の条件を満足する最も多くの枝の数 c_l を求める操作である．

(条件) 1. 枝は交差しない．2. 1つの節点から2つ以上の枝は出ない．

2.2 入力データの識別

まず入力記号列 $\{x_i\}$ がどのカテゴリーに属するかを識別する問題について考える．ただし本論文では入力および標準記号列が次のような仮定を満たすような場合について考える．

(仮定) 1. 入力記号列 $\{x_i\}$ は余分な記号や誤りを含んでおり，あるカテゴリーの標準記号列と完全に一致するという場合は非常に希である．2. 各カテゴリーの標準記号列の要素の数は種々様々である．

そこで入力記号列 $\{x_i\}$ のカテゴリー l に対する類似度 h_l を次のように定義する．

[定義5] 入力記号列 $\{x_i\}$ のカテゴリー l に対する類似度を

$$h_l=c_l/m_l \qquad (1)$$

で表わす．

この類似度を利用して，

[定義6] 入力記号列 $\{x_i\}$ は次の条件を満たすときカテゴリー l に属するという．

(条件) $h_l=\max\{h_1, h_2, \cdots, h_p\}$ (2)

$$m_l=\max\{m_k \mid h_k=h_l\}_{1\leq k\leq p} \qquad (3)$$

[定理1] 各カテゴリーの標準記号列の要素の数 m_k $(k=1, 2, \cdots, p)$ がすべて異なる場合，入力記号列がどのカテゴリーに属するかは一意に定まるが，そうでない場合には一意に決定されるとは限らない．

(証明) 定義6の式(2)は $h_1, h_2, \cdots, h_p$ の最大値を選ぶことなのでこれは必ず求められる．次に

i) $h_k=h_l$ なる k が1つしか存在しない場合．

式(3)より入力記号列は明らかにどのカテゴリーに属するか一意に決まる．

ii) $h_k=h_l$ なる k が複数個存在する場合．

それに対応する m_k を下のようにする．

$$\{m_{kl}\}=(m_{k1}, m_{k2}, m_{k3}, \cdots, m_{kn}) \qquad (4)$$

① m_k $(k=1, 2, \cdots, p)$ がすべて異なる場合．

$m_{k1}\neq m_{k2}\neq m_{k3}\neq\cdots\neq m_{kn}$ であるので $\{m_{kl}\}$ の要素のうち最大のものは一つに決まる．つまり式(3)を満足する m_l は一つしかない．よって $\{x_i\}$ は必ずある一つのカテゴリーに属することになる．

② m_k の中に同一のものが存在する場合．

$\{m_{kl}\}$ の中に同一のものが含まれる可能性があり，その場合式(2)，(3)を満足する l が複数個存在する．

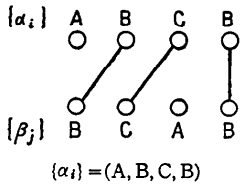
[証明終]

このようにこの識別方法では各カテゴリーの標準パターンの要素の個数がすべて異なる場合を除いては，入力記号列が複数のカテゴリーに属しているというあいまいな結果が出る場合もあり得る．これは実際のパターン認識のシステムではかなり不利なように思われる．しかしこれは我々人間にもどのカテゴリーか判断し難いような場合であり，むしろこのような結果が出る方が望ましいといえよう．そして実用的なパターン認識システムでは他の情報（意味，構文，背景など）を利用して最終的な判断を下すことがより適当であろう．

2.3 アルゴリズム

以上のような考えより枝の数 c_l を速く簡単に求める必要がある．以下にそのアルゴリズムを示す．

まず $\{x_i\}$, $\{S_j{}^l\}$ に対して **Fig. 2** のような行列



$\{\alpha_i\}=(A, B, C, B)$
$\{\beta_i\}=(B, C, A, B)$

Fig. 1 Matching between symbol strings.

	x_1	x_2	x_3	x_4 ……	x_s
$S_1{}^l$	a_{11}	a_{12}	a_{13}	a_{14}	a_{1s}
$S_2{}^l$	a_{21}	a_{22}	a_{23}	a_{24}	a_{2s}
$S_3{}^l$	a_{31}	a_{32}	a_{33}	a_{34}	a_{3s}
⋮					
$S_m{}^l$	a_{m1}	a_{m2}	a_{m3}	a_{m4}	a_{ms}

Fig. 2 Matrix for pattern matching between two symbol strings $\{S_j{}^l\}$ and $\{x_i\}$.

$|a_{jk}|$ を考える．ここで

$$a_{jk}=\begin{cases}1: S_j'=x_k\\ 0: S_j'\neq x_k\end{cases} \qquad (5)$$

である．さて Fig. 2 の行列で $a_{jk}=1$ となる要素に注目し，これをグラフの節点と考える．そして次の定義により有向グラフ G を作る．

[定義 7] 行列 $|a_{jk}|$ で 1 の値を持つ任意の要素 $a_{uv}, a_{u'v'}$ をとる．下の条件を満足するとき a_{uv} から $a_{u'v'}$ に有向線分が引ける．

(条件) $u<u'$ かつ $v<v'$

ただし $1\leq u, u'\leq m,\ 1\leq v, v'\leq s$

[定理 2] $\{x_i\}$ と $\{S_j'\}$ の最良のマッチングを取るということは有向グラフ G で最長のパス（最も多くの節点を通る道）を見つけることと同じである(**Fig. 3**)．

(証明) ① 有向グラフ G 中の節点は式(5)より $x_i=S_j'$ となる対を表わしている．② 定義 7 は定義 4 の条件を満たす（何故ならば，条件 1 は明らか，2 は $u=u'$ または $v=v'$ となるものは除かれているから）．①②より有向グラフ G を求めることは $\{x_i\}$ と $\{S_j'\}$ の順序つきマッチングを行ったのと同じ．また G 中の最長のパスに含まれる節点の数が $\{x_i\}$ と $\{S_j'\}$ の要素の最も多く一致した数に等しいのは明らか．

[証明終]

以下に有向グラフ G 中で最長パスを求めるアルゴリズムを示す．まず行列 $|a_{jk}|$ の要素 a_{jk} が 1 となる点に対して次のような 4 通りの分類を行う（**Fig. 4** 参照）．各点の名前を次のように定義する．

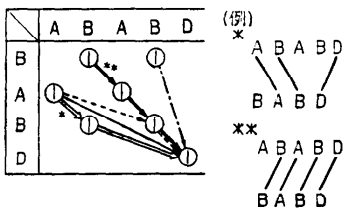
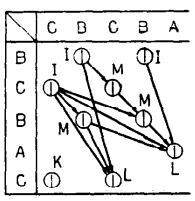


Fig. 3 Correct matching path (＊＊) and other possible pseudo matching paths.



①：$a_{jk}=1$ となる点
I：初期点，K：孤立点
L：最終点，M：中間点

Fig. 4 Illustration of each point

[定義 8]

(i) 初期点 I：必ずパスの第一要素となる点．

(ii) 最終点 L：必ずパスの最終要素となる点．

(iii) 孤立点 K：パスの第一要素でありかつ最終要素にしかなり得ない点．

(iv) 中間点 M：パスの第一または最終要素とはなり得ない点．

それぞれの点に分類するのは簡単で次のようにして行い得る．$a_{jk}=1$ となるような要素に関して二つの行列 $\boldsymbol{a}^1=|a_{bc}|, \boldsymbol{a}^2=|a_{de}|$ を考える（**Fig. 5** 参照）．ここで $|a_{bc}|, |a_{de}|$ はそれぞれ

$$b<j, c<k;\ d>j, e>k$$

なる行列である．そのとき

$1\notin \boldsymbol{a}^1$ かつ $1\in \boldsymbol{a}^2$ ならば a_{jk} は初期点．

$1\in \boldsymbol{a}^1$ かつ $1\notin \boldsymbol{a}^2$ ならば a_{jk} は最終点．

$1\notin \boldsymbol{a}^1$ かつ $1\notin \boldsymbol{a}^2$ ならば a_{jk} は孤立点．

$1\in \boldsymbol{a}^1$ かつ $1\in \boldsymbol{a}^2$ ならば a_{jk} は中間点．

最長のパスの要素の数 c_l を求めるアルゴリズムを **Fig. 6**(次頁参照)に示す．まず初期値として $c_l=0$ とする．次に $a_{jk}=1$ の点を 4 種類に分類し，初期点 I，最終点 L が存在する場合には $c_l=c_l+2$ とする．I，L が存在しない場合には孤立点 K の存在を調べ，存在する場合には $c_l=c_l+1$ とする．次に中間点 M の存在を調べ，M が存在しない場合には c_l を最長のパスの要素の数として出力する．M が存在する場合には I, L, K の点を消し去り，再び点の分類を行い以上の過程をくりかえす．

以上の過程の例を **Fig. 7** (次頁参照)に示す．この例では最長のパスの要素の数 $c_l=5$ となり，$m_l=7$ であるので，この場合の類似度 h_l は式(1)より $h_l=c_l/m_l=5/7$ となる．このようにして $l=1, 2, 3, \cdots, p$ な

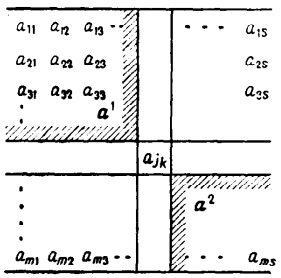


Fig. 5 Illustration of matrixes $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$ for classifying point a_{jk}.

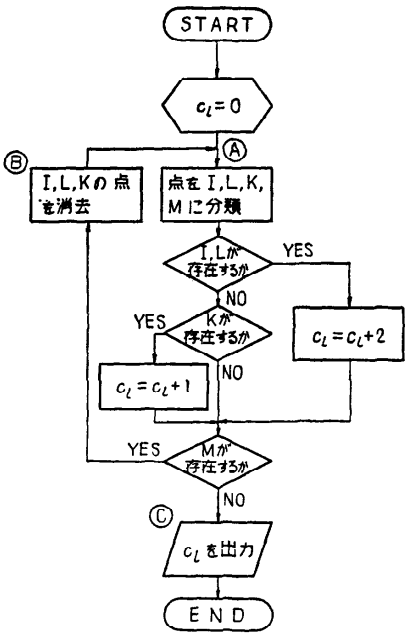


Fig. 6 Algorithm evaluating number of elements included in the path having the maximum length.

る各カテゴリーに対して h_i を計算し最大の h_i を取るカテゴリーに分類する.

2.4 アルゴリズムの完全性

上記のアルゴリズムで最良のマッチングつまり必ず最長のパスが得られることを証明する. いま有向グラフ G 中で最も多くの節点を持つパスの節点の数を最長のパスの長さと呼び $l(G)$ と表わす.

[事実1] I, L, K が存在しない. $\Rightarrow G$ は空グラフ (M もない) $\Rightarrow l(G)=0$.

[事実2] I と L が存在せず, K が存在する. $\Rightarrow G$ は辺を持たない. (空ではない.) $\Rightarrow l(G)=1$. $\Rightarrow G$ から I, L, K を消去した残りを G' とすると $l(G')=0=l(G)-1$.

[事実3] I が存在する. $\Leftrightarrow L$ が存在する. $\Rightarrow G$ から I, L 及び K を消去した残りを G' とすると $l(G')=l(G)-2$.

以上の [事実1]~[事実3] は自明.

[定理3] Fig. 6 のアルゴリズムが終了した時点では $c_l=l(G)$ が成り立つ.

(証明) 最初に与えられたグラフ G を消去によって変形して得られるグラフを G' で表わす. 最初は $G'=G$ であるが, Fig. 6 のⒷの消去のブロックを通過するたびに G' は変化する. しかし Fig. 6 のⒶの時点では常に

$$l(G')+c_l=l(G) \qquad (6)$$

が成り立つ.

式(6)は通過する回数についての帰納法で証明できる.

[式(6)の証明] ① 第1回目: 最初は $G'=G, c_l=0$ だから明らか.

② 第 r 回目: $r-1$ 回目まで式(6)が成り立つと仮定すると,

(i) I, L が存在する場合.

$c_l=c_l+2$ とし, 事実3から $l(G')$ は2減少し, c_l' は2増加する. よって $l(G')+c_l$ の値は変化せず $l(G)$ に等しい.

(ii) I, L が存在せず, K が存在する場合.

事実2から同様にして式(6)は成り立つ.

(iii) 上記以外の場合 (I, L, K なし) は M も存在しないからⒶにもどってはこない. ゆえにⒶにもどる場合には必ず式(6)が保証される. [式(6)の証明終り].

次に Fig. 6 で©の出力に進む時, 次の三つの場合

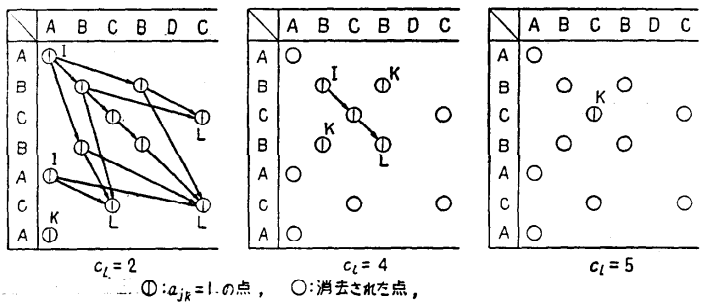


Fig. 7 Example of process of evaluating number of elements included in the path having the maximum length.

が考えられる.

(i) I, L, K が存在しない場合.

この場合 M も存在しないから $G'=\phi$ (空), したがって $l(G')=0$ で式(6)から $c_l=l(G)$.

(ii) I, L が存在せず, K のみ存在する場合.

G' は K のみ含むから $l(G')=1$. ゆえに $c_l=c_l+1$ を実行する前のⒶの時点では式(6)より $1+c_l=l(G)$. ゆえに $c_l=c_l+1$ を実行した後の出力Ⓒの時点では, $c_l=l(G)$.

(iii) I, L が存在し, M が存在しない場合.

$c_l=c_l+2$ を実行し, かつ I, L, K の消去をも実行したとすれば, G' は ϕ (空) となる. 従って $l(G')=0$. ゆえに式(6)より $0+c_l=l(G)$. すなわち $c_l=l(G)$.

実際にはこの場合 I, L, K の消去は実行されないが, それは c_l の値に影響を及ぼさないから Fig. 6 では省略してある. 省略した方がスピードが速い. [証明終]

以上 Fig. 6 のアルゴリズムが終了した時点で必ず最長のパスが得られることが証明された.

3. 音声認識への応用

3.1 連続音声認識への応用

連続音声の認識では単語単位への区切りがあいまいであり, また文末の発声はかなり不正確なものになるので, どうしても入力に余分な記号や誤りが入りやすい. そこで前述の方法が有効に働く. **Fig. 8** に日本語連続音声の認識システムを示すが, ここでは単語の単位での応用と文の単位での応用との二段階で前項で述べた分類の方法を使用している[4]. まず **Table 1** に示すような辞書を用意しておく. ①音声は音響的な処理がほどこされ, 音素列に変換されている (この音素列には誤りが含まれる場合が多い). ②この音素列を音響的な特徴を加味しながら簡略化する. ③また比較

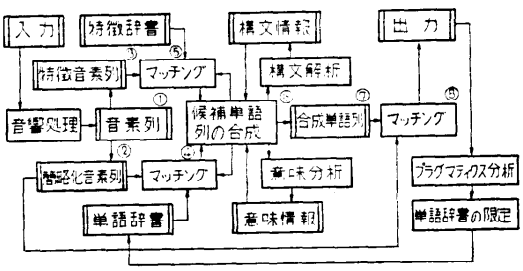


Fig. 8 Brock diagram of speech recognition system for spoken Japanese sentences.

Table 1 Some items of the word lexicon.

単語	単語辞書	特徴辞書
英語	EINO	EIO
数学	SUNA・PU	SUA・U
グラフ	BURAFU	UAU
と	・PO	・O
の	NO	O

的発声が安定な音素を中心に特徴となる音素を記号列としてとっておく. ④⑤この簡略化された音素列 $\{B_i\}$ と特徴音素列 $\{Q_i\}$ に対し, 辞書中の単語 (単語 e は $\{W_j{}'\}_{j=1,2\cdots,\alpha}$ なる音素列で表わされている) の語頭の特徴などより候補の単語 e を選択し, $\{W_j{}'\}$ と, その単語 e に対応して $\{B_i\}$ から作られた $\{B_k{}'\}_{k=1,2,\cdots,\alpha+3}$ とのマッチングを行う. 同時に単語 e の特徴辞書 $\{A_j{}'\}$ と, それに対応する $\{Q_i\}$ から作られた $\{Q_k{}'\}_{k=1,2,\cdots,\beta}$ とのマッチングも行う. ただし一般に $\{B_k{}'\}$ は次のようにして作られる. 現在までに $\{B_i\}$ の n 番目の音素までが認識に使われており, 次にマッチングしようとする単語 e の音素数を α とすると

$$\{B_k{}'\}_{k=1,2,\cdots,\alpha+3}=(B_n, B_{n+1}, \cdots, B_{n+\alpha+1}, B_{n+\alpha+2})$$

$\{Q_k{}'\}$ も同様にして作られる. ⑥これである程度一致したものに対し, 次に続く候補の単語を選び, マッチングを行い, 候補単語列を作っていく. ⑦最終的にいくつかの候補の単語列が作られるが, ⑧その単語列と入力 $\{B_i\}$ とのマッチングを行い最も適合していると思われる単語列を最終の出力とする. 以上のマッチングの部分に前述の分類の方法を使用して, 連続音声の認識に成果をあげている.

なお現在このシステムで利用できる単語数は 99 単語であり, 処理の単位は 1～8 単語で, 長い文は適当な部分にポーズを入れて発声している. 実験結果の例を **Fig. 9, 10** (次頁参照) に示す. データは「数学と英語の」と一息で発声されたもので, その簡略化された音素列が $\{B_i\}$ である. $\{B_i\}$ から各単語に対する $\{B_k{}'\}$ が作られ, それぞれの単語辞書とマッチングが行われる様子を示している. なおこの例では一つずつの候補単語が選ばれ, 最後の候補単語列は一つしかできなかったが, 一般には複数個合成され, そのうち最も得点の高いものが出力として出される. なお単語単位のマッチングの際は辞書を基準に類似度を測るが, 合成された単語列のマッチングの際には入力データを基準にしている.

またこの方法は JIS 7000 レベルのフォートラン

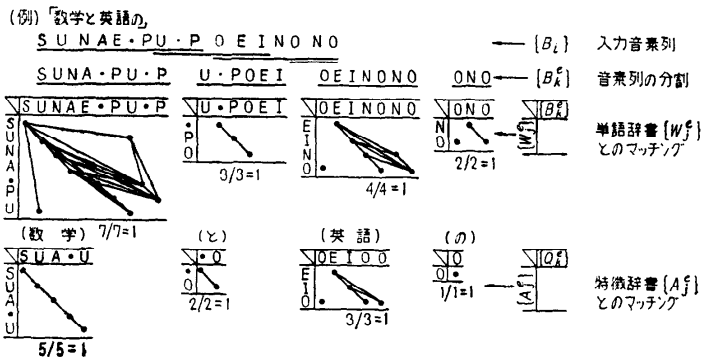


Fig. 9 Example of matching processes in continuous speech recognition.

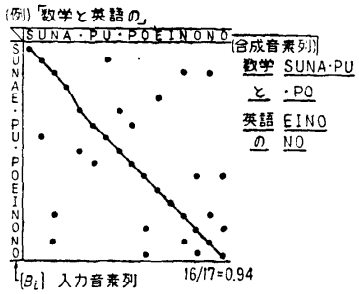


Fig. 10 Example of matching between an input phoneme string and a word string in continuous speech recognition.

プログラムの音声認識システムでも効果的に使われている[5),6)].

3.2 文章中のキー・ワード検出への応用

音声理解においては,話された文章中に或るキー・ワードが存在するか否かを検出することが必要である.この場合にもキー・ワードの単語辞書の音素列を入力音素列の先頭(場合によっては途中)から順次ずらしながら,前述の方法によってマッチングを取っていくことにより目的を達し得るであろう.

またコンパイラ言語のように文法が明確な文章においては,入力文章が Read 文であるとか,算術代入文であるとかいった文の型を知ることは,認識率の向上や,誤り訂正に非常に役立つのは言うまでもない.そして文の型を知るには,各文型に関するキー・ワード群(例えば Read 文では 'READ, (, 数字,)' を順序つきで)が存在するかどうかを調べればよい.筆者らはこの方面への応用として,フォートラン・プログラムの音声認識システムにおける誤り訂正機能として,上記のようなキー・ワード群の存在を調べており,好結果を得ている[3),7)].

4. 検 討

記号列を分類する方法はいろいろ存在するが,ここでは特に扱う対象が余分な部分や誤りを含んでいる可能性が強い記号列であるということを考慮すれば,直接探索法と較べた場合明らかに時間的にも計算の量的にもこの方法の方が優っているといえよう.またパターンマッチングの代表的な方法であるダイナミック・プログラミングの方法に対しても,その標準的な方法と較べ,(i)初期状態,最終状態が複数個あっても全く影響ない.(ii)最適な評価関数をさがす繁雑さがない.(iii)探索のステージ数は最長パスの要素の数を c_1 とすると,必ず $c_1/2$ または $c_1/2+1$ ですむ.(iv)本方法では計算はほとんど簡単な大小判断のみで行える.などの点から,少なくとも入力にかなりのあいまい性が含まれている場合の方法としては,本方法の良さが確認できると考えられる.この点を更に明確にするために現在使用されている他の方法との比較を試みる.

音声認識システムでは音声波を音素列に変換してからダイナミック・プログラミングの手法を応用して辞書とのマッチングを行うものも多い[1)~3)].これらのシステムでは(i)端点をフリーにする.(ii)最適な評価関数を工夫する.などによりかなり良い結果が得られている.

実際の音声認識システムでは音素間の類似度や発声の速さなどの音響的な情報を評価関数に導入する場合が多く,同一条件のもとでの比較は困難である.文献3)における単語のマッチング法はマッチングの際音響的な情報を使用していない.そこで同一条件のもとで

この方法と比較検討することにする.

(実験)

i) 方法: 比較するマッチングの方法は以下のようである. [方法A] 本論文で提案したグラフを利用する方法. [方法B] ダイナミック・プログラミングを用いた文献 3) にある方法. この方法Bは簡単には **Fig. 11** に示すように先頭からマッチングを行い, 評価関数としては前の点からの距離を加算して最少のものを取る. ただし出発点は重要なので, 最初同一の距離があった場合には複数個の道を進んで最後に最良のものを取るようにしてある. なおこの方法は評価関数を簡単にしてスピードを上げるようになっている.

分類されるべきカテゴリーは **Table 2** に示すような日本の都市名 20 である*. 入力は誤りや余分な部分を含んだ資料を使用する. この資料については下に示す. これらの資料を入力し, 都市名の分類を行った. 同時に必要な時間の測定も行った. なお使用した計算機は FACOM 230-45 S である.

ii) 資料: 検討用の資料は辞書をもとに作成した. 連続音声中の単語の識別に近い状況にするため, 資料は **Fig. 12** のような手順で辞書から作り出した. まず①のようなメモリを設け, その4番目から, 6, 8… と1つおきに辞書の音素を入れる. このときメモリの x 番目までつまったとする. 次に前後の単語との間に生起するかも知れない過渡的な音素を想定して②のようにメモリの2番目と $x+2$ 番目に使用可能な音素記号をランダムに選び入れる. 次に③のようにメモリの1番目から $x+3$ 番目までの間に任意の位置を選び, そこに任意の記号 (誤り) を入れる. この誤りの音素記号 (欠如の記号も含む) もランダムに選ばれる. 最後に④でいらないメモリの部分をつめ, これを資料として用いる. なお③の誤りの挿入個数は各単語に対して 0～4 個のものを作った.

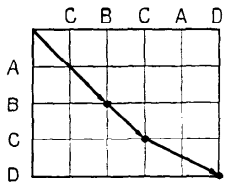


Fig. 11 Method B (Dynamic programming).

Table 2 Example of words for experiment.

	カテゴリー	辞書		カテゴリー	辞書
1	札幌	SA・PORO	11	金沢	・PANASAUOA
2	青森	AONORI	12	甲府	・POFU
3	盛岡	NORIO・PA	13	岐阜	BIFU
⋮			⋮		
10	富山	・POIEANA	20	広島	HIROSINA

* モーラ数の割合, 全音素の出現などが考慮してある文献 8) で使用している単語群を使用した.

(例) 甲府 ・POFU

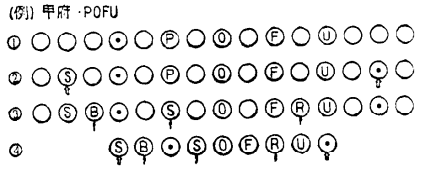


Fig. 12 Process of data production for the experiment (a case where a sample with three errors is produced from '・POFU'). ↑shows an erroneous phoneme added. ⇑shows additional phoneme added before and after the word.

iii) 実験と結果: 上記の資料に対し, 方法A, Bでそれぞれ識別実験を行った. その結果を資料に含まれる誤りの個数別に示したのが **Fig. 13** である. 方法Bは誤りが増加すると正答率はかなり低下するが, 方法Aではあまり悪くならない. また識別に必要な時間は誤りの個数にはほとんど関係なく, 1資料識別するのに平均, 方法Aでは 593 ミリ秒, 方法Bでは 242 ミリ秒かかった (なおこの時間は資料をディスクから読む時間, 結果を出力する時間など全行程にかかった時間を含む. **Table 3** 参照). (実験終)

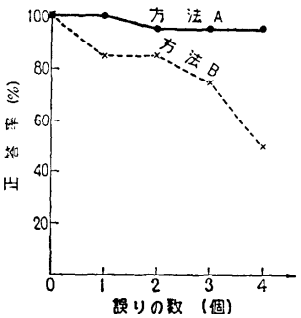


Fig. 13 Comparison between the abilities of two matching methods. Method A: by our method, Method B: by dynamic programming.

Table 3 Mean times necessary to identify each sample.

方法＼誤り(個)	0	1	2	3	4	平均
A (ミリ秒)	594	602	599	587	586	593
B (ミリ秒)	242	240	242	243	243	242

以上二つの方法で単語音声の識別のシミュレーションを行ったが，本論文で提案した方法は比較的機械的な方法でよい識別結果が得られることがわかった．

しかしダイナミック・プログラミングの方法に比して時間が2.4倍ほどかかるので，これを改良する必要がある．なお文献1)で使用されるダイナミック・プログラミングの方法では各ステージで複雑な評価関数をいくつも計算することにより，音響処理段階におけるあいまいさをかなり救済できるので，システム全体としての結果は良くなるであろうが，時間は前述の方法Bよりかなりかかることになる．

5. あとがき

以上余分な部分や誤りを含む記号列を分類する方法について一つの方法を提案，音声認識への応用を試みたが，本方法は比較的機械的な操作で良い結果が得られるから応用範囲が非常に広いと考えられる．なお本方法はグラフ理論でいわれるバイパータイトグラフにおけるマッチングの場合で，節点に順序関係があり，枝に半順序関係がある特殊な場合と考えられる[9]．本方法は様々な場合に応用できるが，それぞれの場合に応じアルゴリズムに少し工夫をこらすことにより，より速く，かつ小容量にすることができる．たとえばFig. 2の $a_{jk}=1$ なる点の選出にあたって $|j-k|$ が閾値より大きい場合は計算をはぶくことによりかなり高速化され，容量も少なくなる．また容量を少なくするため，入力記号列から標準記号列にない記号をあらかじめ抜き去る方法なども効果がある．

末筆ながらアルゴリズムの完全性について，詳しくご教示いただいた山梨大学，野崎昭弘教授に深謝します．

参考文献

1) S. Nakagawa: A Machine Understanding System for Spoken Japanese Sentences. 京都大学学位論文，1976年10月．
2) 松岡，城戸：スペクトルのピークを用いた単語音声の認識，日本音響学会講演論文集，1975年10月．
3) 関口，大輪，青木，重永：フォートラン・プログラムの音声認識システム，情報処理，第18巻，第5号，1977年5月．
4) 関口，重永：単語列の合成による連続音声の認識，日本音響学会，音声研究会資料，S 76-26，1976年11月．
5) M. Shigenaga, Y. Sekiguchi, H. Oowa, K. Aoki: Speech Recognition of FORTRAN Programs. 9th ICA, 1977年7月．
6) 関口，大輪，青木，重永：フォートラン・プログラムの音声認識，電子通信学会，情報部門全国大会講演論文集，1977年8月．
7) 関口，大輪，青木，重永：フォートランプログラムの音声認識における誤り自動修正法，日本音響学会音声研究会資料 S 76-03，1976年6月．
8) 松岡，城戸：音声スペクトルのローカルピークの静特性のもつ音韻情報に関する検討．日本音響学会誌，第32巻，第1号，1976年1月．
9) 服部，小澤：グラフ理論解説，昭晃堂，1974年11月．

(昭和51年11月26日受付)
(昭和53年3月6日再受付)