誤ったキーでも検索できる情報検索システム†

沼倉 覚†† 田中栄一††† 青木晴海††††
矢野目毅††††† 矢吹勉†††††††

本論文は，誤りを持つキーでも検索できる階層的ファイルの構成法について述べている．B-木あるいはそれを改良した階層的ファイルはよく知られているが，これらは正しく記憶されたデータを正しいキーで速く検索するものである．データが大量になると，データが誤って記憶されることも避けられないし，検索者も常に正しいキーを使うとは限らない．このような事態にいかに対処するかはまだ手探りの状態であるが，最近，HL法が提案されている．本論文では，文字類の誤り傾向に基づいて文字をいくつかの類に分割し，その類名を用いてデータを分類し，階層的ファイルを作る．文字類が1つしかできないときは，適当に文字類を作る．このときは，他の類の文字に誤る（置換誤り）ことが起こりうる（類外置換）．ファイルの大部分あるいはすべてを補助記憶に記憶する．キーがファイルにあるかどうかを調べる．キーがファイルにないとき，レーベンシュタイン距離の意味で最も近いキーを探す．3つの方法を提案し，長さ4～8の1.66万語（辞書A）と10.4万語（辞書B）の上で，実験し，辞書Aの上で，HL法と比較した．キーの誤りは，類外置換，挿入，脱落は高々1つとし，実験では類外置換を含めて，誤り数は高々2としている．(a)すべての場合について，辞書の第1段を主記憶に置いた場合の方が，すべてを補助記憶に置いた場合より検索時間は短い．また，辞書Bの場合でも誤りのあるキーでの検索時間はそれほど低下しなかった．(b)誤りのないキーの検索時間は，HL法に比べて820～1800倍速く，誤りのあるキーでは，70～520倍速い．(c)挿入1，脱落1，脱落1+類内置換1の場合は，HL法が検索率がよく，他の5つの場合は本論文の検索率が高い．HL法はすべての場合について誤検索率が高い．

1. まえがき

大型データを能率よく検索するためのファイルの構成法はよく研究されている[1]．たとえば，B-木[2]およびそれを改良した階層化ファイルはよく知られている．これらの方法は，正しく記憶されたデータを正しいキーで速く検索するものであるが，データが大量になると，データが誤って記憶されることも避けられないし，検索キーも常に正しいとは限らない．このような事態にいかに対処するかはまだ手探りの状態である[3]．最近，伊藤・木沢[4),5]は，この問題について1つの階層的ファイルの構成法を提案した．これは，筆者の知る限り，この問題に対するほとんど唯一の研究である．本論文は，綴りの編集法を大型ファイル構成に応用することについて述べる．綴りの編集法[3),6)~8]は，統計法と辞書法に大別される．統計法はn字組確率や文字・単語の発生確率などの統計的情報を用いるが辞書は用いない方法である．辞書法は辞書を用いる方法であるが，統計的情報を用いないことを意味しない．統計法は辞書法と比べて高速であるが訂正率が低いとされている．最近，栗田・相沢[9]は大語彙環境下で高い訂正率を持つ統計法を提案している．一方，辞書法の高速化の研究もある[10)~13]．綴りの編集法を大型ファイル構成に利用するとき，次のような問題がある．

(1) データが大量になると，補助記憶に格納されるため，主記憶と補助記憶の間にデータの転送が頻繁に起こる．データのどの部分を主記憶に，どの部分を補助記憶に格納するか，また，補助記憶のデータをどのように主記憶に転送するか，が問題になる．主記憶上の操作で高速な綴りの編集法が，主記憶と補助記憶を含めた系で高速とは限らない．そこで，主記憶と補助記憶の間で，どのようにデータを分配するかが問題になる．

(2) 情報検索では，通常，大部分のキーは正しい．綴りの編集法は誤った綴りを訂正するものであるが，誤った綴りの訂正に高速であっても，正しいキーの検索に高速であるとは限らない．一般に，キーが誤っていることは少ないと考えられるから，ファイルは正しいキーの検索でも十分高速で

---

† An Information Retrieval System Accessible by Keys with Errors by SATORU NUMAKURA (Information Science, Faculty of Engineering, Graduate School, Utsunomiya University), EIICHI TANAKA (Department of Information Science, Faculty of Engineering, Utsunomiya University), HARUMI AOKI (Fuji Heavy Industries, Ltd.), TAKESHI YANOME (Sharp Corporation) and TSUTOMU YABUKI (Fujitsu, Ltd.).

†† 宇都宮大学大学院工学研究科情報工学専攻
††† 宇都宮大学工学部情報工学科
†††† 富士重工(株)
††††† シャープ(株)
††††††† 富士通(株)

なければならない.

ここでは，綴りの編集法を誤りのあるキーでも検索できる大型ファイルの構成に応用する．第1の方法は，2種類の類名表記を用いた階層的ファイルで，第2の方法は，1種類の類名表記による階層的ファイルであり，いずれもハッシュ法で検索する．両方法は重みつきレーベンシュタイン距離を用いている．第3の方法は，第2の方法で重みつきレーベンシュタイン距離の代わりにハミング距離を用いるものである．

次章で本方法の基礎になる重みつきレーベンシュタイン距離と類名表記について述べる．第3章で検索法，第4章では実験のための大型データの発生法，第5章で実験結果について述べ，HL法[5]と比較する．

2. 重みつきレーベンシュタイン距離[15]と綴りの類名表記

系列 $X=x_1x_2\cdots x_m$ と $Y=y_1y_2\cdots y_n$ との間に次の条件を満たす写像 M_s が定義されているものとする．x_i が y_j に写像されているとき，(i,j) と書く．

(i)　$(i,j)\in M_s$ のとき，$1\leqq i\leqq m, 1\leqq j\leqq n$.

(ii)　$(i_1,j_1),(i_2,j_2)\in M_s$ のとき，

a)　$i_1=i_2$ iff $j_1=j_2$,

b)　$i_1<i_2$ iff $j_1<j_2$.

条件(ii)は**図1**のような置換を禁止している．

$|M_s|$ で M_s の元の数を表すものとする．u_s は M_s の元 (i,j) のうち，$x_i\neq y_j$ であるものの数とし，$v_s=n-|M_s|, w_s=m-|M_s|$ とする．u_s, v_s, w_s はそれぞれ，置換の数，挿入の数，脱落の数である．このとき，X から Y への重みつきレーベンシュタイン距離(WLD) $D(X,Y)$ は，次のように定義される．

$$D(X,Y)=\min_s\{p*u_s+q*v_s+r*w_s\}.$$

ここで，p,q,r はそれぞれ，置換，挿入，脱落の重みである．通常，$p<q+r$ と仮定する．$D(X,Y)$ は，次の式で帰納的に計算できる．

$$d(i,j)=\min\{d(i-1,j)+r, d(i-1,j-1)+p(i,j), d(i,j-1)+q\}.$$

ここで，$d(i,0)=i*r, d(0,j)=j*q$,

$$p(i,j)\begin{cases}p, & x_i\neq y_j \text{ のとき.}\\ 0, & \text{そうでないとき.}\end{cases}$$

このとき，

$$D(X,Y)=d(m,n).$$

ここで，$q=r$ のときは距離公理を満たすが，そうでないときは，必ずしも $D(X,Y)=D(Y,X)$ が成り立たない．

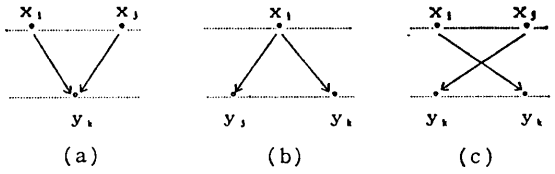

図1　X から Y への写像
Fig. 1　Mapping from X to Y.

文字読取り機 (OCR) などには，読取り誤りに，ある偏りがあるのが普通である．たとえば，荒川[16]の英文字読取りシステムでは，"アルファベットと記号" Σ が次のように分類でき，同じ類の中では読取り誤りがあることがあるが，他の類の文字に誤って読まれることはない．

分　類　1

$A=\{a,h,k,m\}, B=\{n,p,u\}, C=\{b,e,f,g\},$
$D=\{l,q,y,z\}, E=\{i,j,o,r\}, F=\{s,t,v\},$
$G=\{c,d\}, H=\{w,x\}, I=\{\text{-},\text{'}\}.$

ここで，$A,\cdots,I$ を類名という．次に類名表記を定義する[10]．類名表記を用いて，キーの集合を分類し，キー集合全体を探索する代わりに，キー集合の1部分を探索することで，検索率と検索速度を上げることを目指す．いま単語 apple を類名で書くと，$ABBDC$ となるが，これを apple の第1種の類名表記といい，$E1$(apple) と書く．apple が誤って，apule になっても，その類名表記は $E1(\text{apule})=ABBDC$ で，$E1(\text{apple})$ と等しい．分類1を使っていることを明示したいときは，$E1.1(\text{apple})$ のように書く．このように，同じ類内で起こる文字置換を類内置換という．文字誤りの偏り情報が不明なときや，たとえ文字誤りの偏り情報がわかっていても，複数の類に分かれるのではなく，1つの類になってしまうことがある．このときには，適当に文字類を作らなければならない．このとき，文字の置換誤りは，その文字の属する類の文字だけでなく，他の類の文字に誤ることがある．このときの文字置換を類外置換という．

いま，単語 W の第1種の類名表記 $E1(W)$ の中に，$A,\cdots,I$ 類の類名がそれぞれ，$u_A,\cdots,u_I$ 個あるとき，$(u_A,\cdots,u_I)$ を W の第2種の類名表記といい，$E2(W)$ と書く．たとえば，$E2(\text{apple})=(1,2,1,1,0,0,0,0,0)$ である．分類1を用いていることを明示するときは $E2.1(\text{apple})$ と書く．第2種の類名表記でも，類内置換のときは，誤った綴りの類名表記は正しい綴りのそれと等しい．以後，分類1を統合した分類2を

用いる.

分 類 2

$J=\{a,h,k,m,n,p,u\}$,

$K=\{b,e,f,g,l,q,y,z\}$,

$L=\{i,j,o,r,s,t,v\}$, $M=\{c,d,w,x\}$,

$N=\{-,'\}$.

3. ファイル検索法

本章では3種のファイル検索法について述べる. これらの検索法は, キーが誤って他のキーになってしまった場合は考慮していない. このような場合に対処するためには, キーの広い意味での文脈(周辺の情報)を用いないとできない.

(1) 検索法1

第1段に第2種の類名表記, 第2段に第1種の類名表記を用いて, 階層的ファイルを作ることができる[13]が, ここでは, ハッシュ法でファイルにアクセスすることにする. 主記憶に対する負荷を軽くするため, ファイルのアクセス法は文献13)とは異なった方法を用いる. いま, 文字の分類を $T=\{C_1, C_2, \cdots, C_u\}$ とし, 長さ ℓ の単語に関するファイルの見出しになっている第2種の類名表記の1つを $E2_i=(n_1, n_2, \cdots, n_u)$ とする. ここで, n_i は n_{ci} の略記である. $n_1n_2\cdots n_u$ を $E2_i$ の十進数と見て, これを E_{num} とおく. また, ファイル中の長さ ℓ の第2種の類名表記数を $N1(\ell)$ とする. $N1$ の1は第1段の見出しであることを示す. 次のハッシュ関数を作る.

$$h(E2_i)=(E_{num} \bmod N1(\ell))+1.$$

衝突したキーのデータは空き領域に記憶し, 無駄のないハッシュ表を作った. ファイルの一部を**図2**に示す. 第2種の類名表記 $E2(W)$ の下にある第2段の第1種の類名表記の集合を, $E2(W)$ を見出しとする部分辞書といい, $d(E2(W))$ と書く. たとえば, $d((5,1,0,0,0))=\{JJJJJK, JJJJKJ, JJJKJJ\}$.

単語 W が誤って W' になったとする. "類内置換の数には制限を設けないが, 類外置換, 挿入, 脱落の数の和は高々1とする"(条件1). いま, 分類2を用いた類名表記を考え, $E2.2(W')=(n_1, n_2, n_3, n_4, n_5)$ とすると, $E2.2(W)$ は次のいずれかである.

類内置換と仮定したとき

$(n_1, n_2, n_3, n_4, n_5)$.

類外置換と仮定したとき

$(n_1-1, n_2+1, \cdots, n_5), (n_1-1, n_2, n_3+1, \cdots, n_5), \cdots, (n_1-1, n_2, \cdots, n_5+1),$

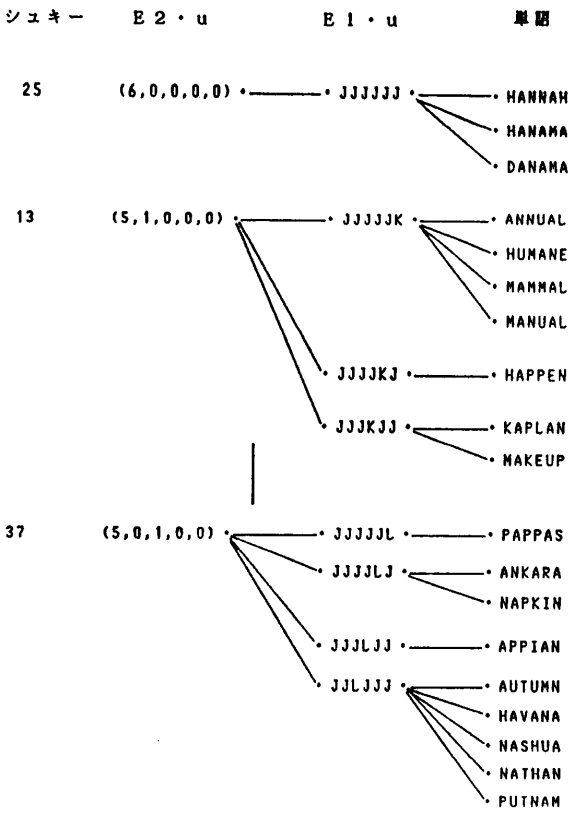

図2 階層化ファイルの一部
Fig. 2 A part of a hierarchical file.

$(n_1+1, n_2-1, n_3, \cdots, n_5), (n_1, n_2-1, n_3+1, \cdots, n_5),$
$\cdots, (n_1, n_2-1, \cdots, n_5+1),$
$\vdots$
$\vdots$
$(n_1+1, n_2, \cdots, n_5-1), (n_1, n_2+1, \cdots, n_5-1), \cdots,$
$(n_1, n_2, \cdots, n_4+1, n_5-1).$

(展開2)

挿入と仮定したとき

$(n_1-1, n_2, \cdots, n_5), (n_1, n_2-1, \cdots, n_5), \cdots,$
$(n_1, n_2, \cdots, n_5-1).$

脱落と仮定したとき

$(n_1+1, n_2, \cdots, n_5), (n_1, n_2+1, \cdots, n_5), \cdots,$
$(n_1, n_2, \cdots, n_5+1).$

ここで $n_i-1<0$ $(i=1,2,\cdots,5)$ のときは, その項がないものとする. たとえば, $(n_1-1, n_2+1, \cdots, n_5)$ で $n_1-1<0$ のときは $(n_1-1, n_2+1, \cdots, n_5)$ は作らない. 検索法1では, ある分類 T を用いるものとし, 特にそれを明記しない.

[検索法1]

0. $E1(W')$ と $E2(W')$ を作る.

1. (W' は誤りがないと仮定し，ファイルを検索する)
   (a) $E2(W')$ のハッシュ値を計算し，ファイルをアクセスする．
   $E2(W')$ がファイルにあれば(b)へ．
   なければ 2. へ．
   (b) $E1(W')$ が，$E2(W')$ を見出しとする第2段の部分辞書 $d(E2(W'))$ にあれば(c)へ．
   なければ 2. へ．
   (c) W' が $E1(W')$ を見出しとする第3段の部分辞書 $d(E1(W'))$ にあれば，W' が辞書にあることを表示し，なければ 2. へ．
2. (W' が辞書にないので，W' と最も近いものを探す)
   (a) $d_{\min} := |W'|$, $S4 := \phi$.
   ここで，$|W'|$ は W' の長さである．
   (b) $E2(W')$ から，(条件1)に合うすべての類名表記を発生させる．
   発生させた類名表記の集合を $S1$ とする．
   (c) $S1$ から類名表記を1つ（それを $E2_k$ とする）取り出し，
   $S1 := S1 - \{E2_k\}$
   とおく．$E2_k$ のハッシュ値を計算し，ファイルをアクセスする．$E2_k$ がファイルにあれば，(d)へ．なければ，$S1 \neq \phi$ のとき，(c)へ．$S1 = \phi$ なら，(g)へ．
   (d) $E1(W')$ と $E2_k$ を見出しとする第2段の部分辞書 $d(E2_k)$ にある類名表記 $E1_h$ との距離を計算し，$D(E1_h, E1(W')) \leqq 1$ となる類名表記の集合 $S2$ を作る．(e)へ．
   (e) $S2$ から類名表記1を（それを $E1_k$ とする）を取り出し，
   $S2 := S2 - \{E1_k\}$, $S3 := d(E1_k)$,
   とおく．(f)へ．
   (f) $S3$ から単語を1つ（それを W_x とする）取り出し，
   $S3 := S3 - \{W_x\}$
   として，W_x から W' への距離を計算し，
   $D(W_x, W') < d_{\min}$ なら
   $d_{\min} := D(W_x, X')$, $S4 := \{W_x\}$,
   とおき，
   $D(W_x, W') = d_{\min}$ なら
   $S4 := S4 \cup \{W_x\}$,
   とおく．
   $S3 \neq \phi$ なら(f)へ．$S3 = \phi$ かつ $S2 \neq \phi$ なら(e)へ．
   $S3 = \phi$ かつ $S2 = \phi$ かつ $S1 \neq \phi$ なら(c)へ．
   (g) $|S4| = 1$ なら，W' を $S4$ にある単語に訂正する．
   $|S4| > 1$ なら，W' を棄却する．
   類外置換がないことが明らかであるときは，2.(b)で(展開2)の類名表記を発生させなくてもよい．

(2) 検索法2

検索法2は第1種の類名表記を使った階層化ファイル[12]をハッシュ法でアクセスする．主記憶に対する負荷を軽くするため，ファイルへのアクセス法は文献13)の方法とは異なる．操作の高速化を図るため，第1段の類名表記を数値で表しているのは文献13)と異なる．分類 $T = \{C_1, C_2, \cdots, C_n\}$ とするとき，C_i を数値 $(i-1)$ で表すと，類名表記を数値で表現できる．このようにすると，類名表記を1つの u 進法で表すことができ，ハッシュ表にその類名表記があるかどうかを調べるのが1回の比較ですむ．たとえば，J, K, L, M, N の類名を持つ分類2を用いた1つの類名表記 $KLJML$ を考える．いま，類名に次のように数字を割り当てる．$J \to 0$, $K \to 1$, $L \to 2$, $M \to 3$, $N \to 4$. このとき，$KLJML$ は 12032 なる数表現を得るが，これを5進数と見なす．長さ5の第1種の類名表記の数 $N2(\ell)$ が525であったとすると，ハッシュ値 $h(KLJML)$ は

$$\begin{aligned} h(KLJML) &= ((12032)_5 \bmod 525) \\ &= 892 \bmod 525 \\ &= 369 \end{aligned}$$

となる．ここで $(12032)_5$ は 12032 が5進数であることを示す．記述を一般的にするために，第1段，第2段で使う分類をそれぞれ T_1, T_2 と書く．T_2 は T_1 の細分類であれば，T_1, T_2 はどのような分類でもよい．いま，$E1.\ T_1(W') = A_1A_2\cdots A_\ell$ とする．これを数値で表したものを $IE1.\ T_1(W')$ と書き，$(F_1F_2\cdots F_\ell)_u$ とする．添字の u は，$F_1F_2\cdots F_\ell$ が u 進法であることを示す．また，T_1 の類数は u とし，(条件1)の下で考えるものとする．$IE1.\ T_1(W')$ から $IE1.\ T_1(W)$ を推定すると，次のいずれかになる．

類内置換と仮定したとき，
$(F_1F_2\cdots F_\ell)_u$.

類外置換と仮定したとき，
$(iF_2\cdots F_\ell)_u (i \neq F_1)$, $(F_1iF_3\cdots F_\ell)_u (i \neq F_2)$, $\cdots$,

$(F_1F_2\cdots F_{\ell-1}i)_u (i \neq F_\ell)$. (条件3)

挿入と仮定したとき,

$(F_2F_3\cdots F_\ell)_u, (F_1F_3\cdots F_\ell)_u, \cdots, (F_1F_2\cdots F_{\ell-1})_u$.

脱落と仮定したとき,

$(iF_1F_2\cdots F_\ell)_u, (F_1iF_2\cdots F_\ell)_u, \cdots, (F_1F_2\cdots F_\ell i)$,

ここで, $0 \leqq i \leqq u-1$ である.

[検索法2]

0. $IE1.T_1(W')$ と $E1.T_2(W')$ を作る.

1. (W' は誤りがないと仮定し, ファイルを検索する)

(a) $IE1.T_1(W')$ のハッシュ値を計算し, ファイルをアクセスする. $IE1.T_1(W')$ がファイルにあれば, (b)へ. なければ2. へ.

(b) $E1.T_2(W')$ が $IE1.T_1(W')$ を見出しとすると第2段の部分辞書 $d(IE1.T_1(W'))$ にあれば, (c)へ. なければ2. へ.

(c) W' が $E1.T_2(W')$ を見出しとする第3段の部分辞書 $d(E1.T_2(W'))$ にあれば, W' が辞書にあることを表示し, なければ2. へ.

2. (W' が辞書にないので, W' と最も近いものを探す)

(a) $d_{\min} := |W'|$, $S4 := \phi$.

(b) $IE1.T_1(W')$ から, (条件1)に合うすべての類名表記を発生させる. 発生させた類名表記の集合を $S1$ とする.

(c) $S1$ から類名表記を1つ (それを $E1.T_{1k}$ とする) 取り出し,

$S1 := S1 - \{E1.T_{1k}\}$

とおく. $E1.T_{1k}$ のハッシュ値を計算し, ファイルをアクセスする. $E1.T_{1k}$ がファイルにあれば, (d)へ. なければ, $S1 \neq \phi$ のとき(c)へ. $S1 = \phi$ なら(g)へ.

(d) $E1.T_2(W')$ と $E1.T_{1k}$ を見出しとする第2段の部分辞書 $d(E1.T_{1k})$ にある類名表記 $E1.T_{2h}$ との距離を計算し, $D(E1.T_{2h}, E1.T_2(W')) \leqq 1$ となる類名表記の集合 $S2$ を作る. (e)へ.

(e) $S2$ から類名表記を1つ (それを $E1.T_{2k}$ とする) を取り出し,

$S2 := S2 - \{E1.T_{2k}\}$,

$S3 := d(E1.T_{2k})$,

とおく.

(f), (g)は, 検索法1に同じ.

類外置換がないことが明らかであるときは, 2. (b)で(*3)を発生させなくてもよい.

(3) 検索法3

検索法3は, 検索法2で WLD を計算する部分をハミング距離 (HD) で置き換えるものである. 置換誤りを仮定したときは, WLD をそのまま HD で置き換える. 挿入誤りを仮定したとき, たとえば, $E = F_1\cdots F_{i-1}F_{i+1}\cdots F_\ell$ がファイルにあったとする. このとき, $d(E)$ にある1つの類名表記を $B_1B_2\cdots B_{\ell-1}$ とすると, $HD(B_1B_2\cdots B_{\ell-1}, \tilde{E}1.T_2(W'))$ を計算する. ここで, $E1.T_2(W') = G_1G_2\cdots G_\ell$ とするとき, $\tilde{E}1.T_2(W') = G_1\cdots G_{i-1}G_{i+1}\cdots G_\ell$ である. 脱落誤りを仮定したとき, たとえば, $E = F_1\cdots F_iF_jF_{i+1}\cdots F_\ell$ がファイルにあったとする. このとき, $d(E)$ にある類名表記を $D_1D_2\cdots D_{\ell+1}$ とすると, $HD(D_1D_2\cdots D_{\ell+1}, \tilde{E}1.T_2(W'))$ を計算する. ここで, $\tilde{E}1.T_2(W') = G_1\cdots G_i\$G_{i+1}\cdots G_\ell$ である. \$ は Σ に属さない文字である.

(4) 検索法の相違

検索法1と検索法2はファイルの構成法が異なっており, したがって検索の方法も異なっている. 検索法3は検索法2と同じファイルを使うが, 検索法2が WLD を使って検索するのに対して, 検索法3は HD を使って, 検索率を落とさずに検索速度を上げることを目指している. 検索法は, ファイル構成・変更の容易さ, 検索速度, 検索率などで評価される. 3方法と

表1 2種類の辞書の語数と類名表記数の分布
Table 1 The distributions of words and class name expressions of the two dictionaries.

単語長	辞書A				辞書B			
	単語数	第1種類名表記数		第2種類名表記数	単語数	第1種類名表記数		第2種類名表記数
		分類2	分類3	分類2		分類2	分類3	分類2
4	2135	215	23	37	4000	247	28	42
5	3091	668	41	59	10000	959	75	82
6	3782	1679	78	84	30000	3444	154	130
7	4025	2714	144	105	30000	8142	281	172
8	3567	3070	265	120	30000	14790	508	224
計	16600	8346	551	405	104000	27582	1046	650

も，ファイル構成・変更は同程度に容易である．検索速度，検索率などは，実験で比較する．

4. 大型データの発生法

いま，長さ m の単語の集合（語数 N_0）から，第 i 文字が a，第 $(i+1)$ 文字が b であるものの数を調べ，それを $d_i(a,b)$ とする．$d_0(\$,a)$ は先頭の文字が a であるものの語数である．$d_i(a,b)(i=0,1,\cdots,m)$ を“位置を決めた2文字組”(positional-digram) という．

$$\sum_{i=0}^{m}\sum_{j=1}^{28}\sum_{k=1}^{28} d_i(x_j,x_k)=(m+1)*N_0$$

ここで，$x_j, x_k \in \Sigma$ である．N_z 個の文字列の発生は，次のようにする．

（1） 先頭文字の発生

乱数を28個発生させ，それを $q_1,\cdots,q_{28}$ とする．

$q_j*d_0(\$,x_j)(j=1,2,\cdots,28)$

のうち，最大の値をとるもの（それを，$q_s*d_0(\$,x_s)$ とする）を選び，x_s を第1文字とする．最大値をとる

表 2　辞書Aを用いた類外置換を認めない場合の検索結果
Table 2　The results of retrieval for the dictionary A with no outer class substitution.

検索法	誤りの数			検索率 (%)	誤検索率 (%)	棄却率 (%)	検索時間1 (秒)	検索時間2 (秒)
	類内置換	挿入	脱落					
検索法 1				100.0	0.0	0.0	0.023	0.037
	1			86.4	0.0	13.6	1.124	1.312
	2			57.6	3.2	39.2	1.272	1.486
		1		97.6	0.0	2.4	1.471	1.697
	1	1		88.4	0.4	11.2	1.332	1.551
			1	49.2	0.0	50.8	0.611	0.700
	1		1	16.0	16.0	68.0	0.811	0.923
検索法2　3段				100.0	0.0	0.0	0.021	0.025
	1			86.4	0.0	13.6	1.053	1.126
	2			57.6	3.2	39.2	1.159	1.237
		1		97.6	0.0	2.4	1.028	1.110
	1	1		88.4	0.4	11.2	1.016	1.105
			1	49.2	0.0	50.8	0.729	0.772
	1		1	16.0	16.0	68.0	0.937	0.999
検索法2　2段				100.0	0.0	0.0	0.013	0.016
	1			86.4	0.0	13.6	0.196	0.314
	2			57.6	3.2	39.2	0.212	0.340
		1		97.6	0.0	2.4	0.124	0.332
	1	1		88.4	0.4	11.2	0.102	0.316
			1	49.2	0.0	50.8	0.252	0.311
	1		1	16.0	16.0	68.0	0.286	0.380
検索法3　3段				100.0	0.0	0.0	0.021	0.025
	1			86.4	0.0	13.6	0.390	0.463
	2			58.4	3.2	38.4	0.427	0.509
		1		97.6	0.0	2.4	0.370	0.439
	1	1		90.0	0.4	9.6	0.364	0.438
			1	49.2	0.0	50.8	0.303	0.343
	1		1	16.8	16.0	67.2	0.375	0.427
検索法3　2段				100.0	0.0	0.0	0.013	0.016
	1			86.4	0.0	13.6	0.095	0.212
	2			58.4	3.2	38.4	0.103	0.236
		1		97.6	0.0	2.4	0.069	0.277
	1	1		90.0	0.4	9.6	0.064	0.269
			1	49.2	0.0	50.8	0.112	0.182
	1		1	16.8	16.0	67.2	0.131	0.219

表 3 辞書Aを用いた類外置換を認めた場合の検索結果
Table 3 The results of retrieval for the dictionary A with an outer class substitution.

検索法	誤りの数				検索率 (%)	誤検索率 (%)	棄却率 (%)	検索時間1 (秒)	検索時間2 (秒)
	置換		挿入	脱落					
	類外	類内							
検索法1					100.0	0.0	0.0	0.023	0.037
		1			79.6	0.0	20.4	2.165	2.546
		2			39.2	3.6	57.2	2.579	3.022
	1				80.4	0.0	19.6	1.939	2.325
	1	1			52.8	6.8	40.4	2.181	2.625
			1		94.0	0.0	6.0	3.364	4.031
		1	1		83.6	0.4	16.0	3.107	3.775
				1	36.4	0.0	63.6	1.058	1.195
		1		1	6.8	19.2	74.0	1.376	1.561
検索法2 3段					100.0	0.0	0.0	0.021	0.025
		1			79.6	0.0	20.4	1.704	1.807
		2			39.2	3.6	57.2	1.988	2.115
	1				80.4	0.0	19.6	1.459	1.572
	1	1			52.8	6.8	40.4	1.720	1.846
			1		94.0	0.0	6.0	1.805	1.933
		1	1		83.6	0.4	16.0	1.811	1.954
				1	36.4	0.0	63.6	1.082	1.131
		1		1	6.8	19.2	74.0	1.391	1.457
検索法2 2段					100.0	0.0	0.0	0.013	0.017
		1			79.6	0.0	20.4	0.315	0.504
		2			39.2	3.6	57.2	0.364	0.584
	1				80.4	0.0	19.6	0.258	0.449
	1	1			52.8	6.8	40.4	0.305	0.524
			1		94.0	0.0	6.0	0.204	0.521
		1	1		83.6	0.4	16.0	0.178	0.503
				1	36.4	0.0	63.6	0.414	0.510
		1		1	6.8	19.2	74.0	0.479	0.624
検索法3 3段					100.0	0.0	0.0	0.021	0.025
		1			79.6	0.0	20.4	0.570	0.673
		2			39.6	3.6	56.8	0.662	0.784
	1				80.4	0.0	19.6	0.491	0.597
	1	1			54.4	6.8	38.8	0.575	0.698
			1		94.0	0.0	6.0	0.567	0.690
		1	1		85.2	0.4	14.4	0.567	0.699
				1	36.4	0.0	63.6	0.414	0.459
		1		1	7.2	19.2	73.6	0.522	0.582
検索法3 2段					100.0	0.0	0.0	0.013	0.017
		1			79.6	0.0	20.4	0.149	0.332
		2			39.6	3.6	56.8	0.172	0.391
	1				80.4	0.0	19.6	0.126	0.312
	1	1			54.4	6.8	38.8	0.146	0.361
			1		94.0	0.0	6.0	0.110	0.422
		1	1		85.2	0.4	14.4	0.102	0.422
				1	36.4	0.0	63.6	0.163	0.280
		1		1	7.2	19.2	73.6	0.195	0.343

ものが2つ以上あるときは再試行する．

$d_0(\$, x_t) := d_0(\$, x_t) - N_0/N_x$ とする．

（2） 第 k 文字の発生

$\alpha_1\alpha_2\cdots\alpha_{k-1}(k<m)$ が作られたとする．新たに発生した28個の乱数を，$q_1, q_2, \cdots, q_{28}$ とする．

$q_j * d_{k-1}(\alpha_{k-1}, x_j)(j=1, 2, \cdots, 28)$

のうち，最大の値をとるもの（それを，$q_t * d_{k-1}(\alpha_{k-1}, x_t)$ とする）を選び，x_t を第 k 文字とする．最大値をとるものが2つ以上あるときは再試行する．

$d_{k-1}(\alpha_{k-1}, x_t) := d_{k-1}(\alpha_{k-1}, x_t) - N_0/N_x$,

とする．

実際には，同じ文字列が発生することがあるので，上記の方法では，N_x 個の文字列を作ることができない．そこで，N_0/N_x の代わりに weight$*N_0/N_x$ (0<weight<1) を用いた．

この系列の発生法を用いると，N_0 と類似の性質を持つ N 個の系列を発生させることができる．たとえば，1つの類名表記 E_k の下にある単語（あるいは系

表 4　辞書Bを用いた類外置換を認めない場合の検索結果
Table 4　The results of retrieval for the dictionary B with no outer class substitution.

検索法	誤りの数			検索率 (%)	誤検索率 (%)	棄却率 (%)	検索時間1 (秒)	検索時間2 (秒)
	類内置換	挿入	脱落					
検索法 1				100.0	0.0	0.0	0.028	0.042
	1			62.8	0.0	37.2	1.925	1.959
	2			31.2	8.0	60.8	3.081	3.109
		1		88.4	0.0	11.6	4.686	4.724
	1	1		63.2	2.4	34.4	4.968	4.985
			1	28.4	0.0	71.6	1.053	1.101
	1		1	4.8	20.0	75.2	1.683	1.744
検索法2　3段				100.0	0.0	0.0	0.026	0.026
	1			62.8	0.0	37.2	1.823	1.845
	2			31.2	8.0	60.8	2.759	2.812
		1		88.4	0.0	11.6	3.136	3.186
	1	1		63.2	2.4	34.4	3.296	3.358
			1	28.4	0.0	71.6	1.125	1.127
	1		1	4.8	20.0	75.2	1.825	1.830
検索法2　2段				100.0	0.0	0.0	0.015	0.019
	1			62.8	0.0	37.2	0.443	0.542
	2			31.2	8.0	60.8	0.665	0.817
		1		88.4	0.0	11.6	—	0.640
	1	1		63.2	2.4	34.4	—	0.663
			1	28.4	0.0	71.6	0.560	0.623
	1		1	4.8	20.0	75.2	0.836	0.923
検索法3　3段				100.0	0.0	0.0	0.026	0.026
	1			62.8	0.0	37.2	0.572	0.612
	2			31.6	8.0	60.4	0.851	0.911
		1		88.4	0.0	11.6	0.896	0.980
	1	1		64.4	2.4	33.2	0.942	1.037
			1	28.4	0.0	71.6	0.374	0.385
	1		1	4.8	20.0	75.2	0.582	0.598
検索法3　2段				100.0	0.0	0.0	0.015	0.019
	1			62.8	0.0	37.2	0.181	0.242
	2			31.6	8.0	60.4	0.265	0.358
		1		88.4	0.0	11.6	—	0.335
	1	1		64.4	2.4	33.2	—	0.359
			1	28.4	0.0	71.6	0.203	0.230
	1		1	4.8	20.0	75.2	0.296	0.336

表 5 辞書Bを用いた類外置換を認めた場合の検索結果
Table 5 The results of retrieval for the dictionary B with an outer class substitution.

検索法	誤りの数				検索率 (%)	誤検索率 (%)	棄却率 (%)	検索時間1 (秒)	検索時間2 (秒)
	置換		挿入	脱落					
	類外	類内							
検索法 1					100.0	0.0	0.0	0.028	0.042
		1			47.2	0.0	52.8	3.242	3.354
		2			14.4	10.8	74.8	5.696	5.815
	1				52.8	0.0	47.2	3.410	3.522
	1	1			18.4	10.0	71.6	5.425	5.618
			1		82.8	0.0	17.2	9.468	9.564
		1	1		54.8	2.8	42.4	10.041	10.075
				1	17.2	0.0	82.8	1.489	1.549
		1		1	3.6	18.4	78.0	2.384	2.453
検索法2 3段					100.0	0.0	0.0	0.026	0.026
		1			47.2	0.0	52.8	2.546	2.582
		2			14.4	10.8	74.8	4.454	4.506
	1				52.8	0.0	47.2	2.668	2.709
	1	1			18.4	10.0	71.6	4.038	4.108
			1		82.8	0.0	17.2	5.130	5.224
		1	1		54.8	2.8	42.4	5.478	5.580
				1	17.2	0.0	82.8	1.401	1.411
		1		1	3.6	18.4	78.0	2.380	2.390
検索法2 2段					100.0	0.0	0.0	0.014	0.019
		1			47.2	0.0	52.8	0.862	0.999
		2			14.4	10.8	74.8	1.380	1.589
	1				52.8	0.0	47.2	0.745	1.038
	1	1			18.4	10.0	71.6	1.266	1.462
			1		82.8	0.0	17.2	—	1.064
		1	1		54.8	2.8	42.4	—	1.462
				1	17.2	0.0	82.8	0.774	0.854
		1		1	3.6	18.4	78.0	1.171	1.286
検索法3 3段					100.0	0.0	0.0	0.026	0.026
		1			47.2	0.0	52.8	0.729	0.769
		2			14.4	10.8	74.8	1.256	1.328
	1				52.8	0.0	47.2	0.756	0.804
	1	1			18.8	10.0	71.2	1.148	1.225
			1		82.8	0.0	17.2	1.346	1.474
		1	1		55.2	2.8	42.0	1.447	1.591
				1	17.2	0.0	82.8	0.442	0.457
		1		1	3.6	18.4	78.0	0.740	0.764
検索法3 2段					100.0	0.0	0.0	0.014	0.019
		1			47.2	0.0	52.8	0.284	0.372
		2			14.4	10.8	74.8	0.453	0.598
	1				52.8	0.0	47.2	0.295	0.396
	1	1			18.8	10.0	71.2	0.411	0.556
			1		82.8	0.0	17.2	—	0.538
		1	1		55.2	2.8	42.0	—	0.585
				1	17.2	0.0	82.8	0.254	0.293
		1		1	3.6	18.4	78.0	0.391	0.454

列）の個数を $|E_k|$ とする．$|E_k|=s$ となる類名表記の数を $n(s)$ とする．原集合（元の数 N_0）の類名表記数分布を $n(1), n(2), \cdots$ と，新しく作った系列集合（元の数 N）の類名表記数 $n'(1), n'(2), \cdots$ がほぼ等しい．

5. 実　　験

（1）実験条件

実験に使った辞書は UNIX の OS 下にある約 2.4 万語の辞書のうち長さ 4〜8 の単語約 1.66 万語の辞書Aと第4章の方法で発生した長さ 4〜8 の綴り 10.4 万語の辞書Bの2種類である．両辞書の語数および類名表記分布を**表1**に示す．ここで，分類3は次のものである．

分　類　3

$O=J\cup K, P=L\cup M, Q=N$

実験は類外置換がないと仮定したときは，6種の誤りパタンと誤りのない場合の7つの場合について，類外置換があると仮定したときは9つの場合について実験した．誤りのある場合には，各々の場合について，長さ6の単語から250の誤った綴りを発生させた．使用した計算機は ACOS-600 S で，プログラムは FORTRAN で書いた．

（2）実験結果

実験結果を**表2〜表5**に示す．2段法では，分類2を，3段法では分類2と分類3を用いた．表で，検索時間1は第1段を主記憶に置いた場合で，検索時間2はすべて補助記憶に記憶した場合である．なお，表4および表5で，検索法2（2段）で挿入のある場合の検索時間1のデータは主記憶容量の制限からとれなかった．

a）すべての場合について，ファイルを主記憶と補助記憶に分散して記憶した方が，補助記憶にだけ記憶する場合に比べて，検索に要する時間は短い．

b）辞書 B の類外置換を認める場合以外は，“類内置換1かつ脱落1”の場合は誤検索率が 16〜20 % と高いが，その他の場合は誤検索率が数 % 以下と低い．辞書Bの場合も，検索速度はそれほど落ちなかった．

c）HL 法で，辞書Aを用いて実験した結果を**表6**に示す．HL 法の検索率は，本論文の方法の検索率と比べて，挿入1，脱落1，脱落 1+類内置換1の場合には高く，他の場合には低い．また HL 法はすべての場合について誤検索率が高い．棄却されるときは，すべての訂正候補を表示して検索者に選ばさせることができるが，誤検索ではそれができないので，誤検索率が高いのは好ましくない．検索時間は，誤りのないキーの場合，本論文の方法が 320〜1800 倍近く，誤りのあるキーでは 70〜520 倍速い．

表6 HL 法による検索率と検索時間
Table 6 The correct retrieval rate and the retrieval time by the HL method.

検索法	誤りの数				検索率 (%)	誤検索率 (%)	棄却率 (%)	検索時間 (秒)
	置換		挿入	脱落				
	類外	類内						
HL 法					100.0	0.0	0.0	30.438
		1			65.2	19.6	15.2	197.639
		2			14.0	48.4	37.6	190.810
	1				64.8	16.8	18.4	194.122
	1	1			14.0	57.2	28.8	186.169
			1		97.6	2.4	0.0	207.887
		1	1		55.6	23.2	21.2	254.983
				1	68.0	20.0	12.0	127.916
		1		1	8.8	53.2	38.0	131.732

6. むすび

誤ったキーでも検索できる階層的ファイルを用いた検索法を3つ提案し，実験してその有効性を確かめた．実験は，類外置換，挿入，脱落誤りが一語中高々1つであるとの制限の下に，長さ 4〜8 の 1.66 万語の辞書を用いて行い，これまで知られていた唯一の方法 (HL 法) と比べ，はるかに高速である．8つの誤りパタンのうち，5つの場合で，提案した方法は HL 法より検索率が高く，すべての場合について，誤検索率が小さい．長さ 4〜8 の 10.4 万語についての実験でも，検索速度はそれほど落ちなかった．

これらの方法に，統計的情報をファイルの構成に用いて高速化を図ったり，誤訂正率を上げないで検索率を高めること，また，単語検索では造語規則を用いて検索率を高めること等は，今後の問題である．さらに，文脈依存類似度[17]が有用な問題に対するファイルの構成法も残された問題である．

参考文献

1) 伊藤：情報検索，昭晃堂 (1986).
2) Bayer, R. and McCreight, E.: Organization and Maintenance of Large Ordered Indexes, *Acta Inf.*, Vol. 1, pp. 173-189 (1972).
3) Hall, P. A. V. and Dowling, G. R.: Approximate String Matching, *Comput. Surv.*, Vol. 12, No. 4, pp. 381-402 (1980).
4) 伊藤，木沢：階層化ファイルによるつづり誤りの訂正，電子通信学会論文誌，Vol. J56-D, No. 8, pp. 1090-1091 (1982).
5) Ito, T. and Kizawa, M.: Hierarchical File Organization and Its Application to Similar-String Matching, *ACM Trans. Database Syst.*, Vol. 8, No. 3, pp. 410-433 (1983).
6) Peterson, J. L.: Computer Programs for Detecting and Correcting Spelling Errors, *CACM*, Vol. 23, No. 12, pp. 676-687 (1980).
7) 川合：英文綴り検査法，情報処理，Vol. 24, No. 4, pp. 507-513 (1983).
8) 伊藤：英文つづり誤り訂正法，情報処理，Vol. 25, No. 5, pp. 471-479 (1984).
9) 栗田，相沢：日本語に適した単語の誤入力訂正法とその大語彙単語認識への応用，情報処理学会論文誌，Vol. 25, No. 5, pp. 831-841 (1984).
10) 田中，小橋口，島村：綴りの置換誤りの高速訂正法，情報処理学会論文誌，Vol. 27, No. 2, pp. 177-182 (1986).
11) Tanaka, E., Toyama, T. and Kawai, S.: High Speed Error Correction of Phoneme Sequence, *Pattern Recogn.*, Vol. 16, No. 5, pp. 407-412 (1986).
12) Tanaka, E. and Kojima, Y: A High Speed String Correction Method Using a Hierarchical File, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 6, pp. 806-815 (1987).
13) 田中，古河原：系列の高速訂正法，情報処理学会文書処理とヒューマンインタフェース研究会，15-1 (1987).
14) Tanaka, E. and Kogawara, A.: High Speed String Edit Methods Using Hierarchical Files and Hashing Techniques, *The 9th Int. Conf. Pattern Recogn.*, Rome, pp. 334-336 (1988).
15) Okuda, T., Tanaka, E. and Kasai, T.: A Method for the Correction of Garbled Words Based on the Levenshtein Metric, *IEEE Trans. Comput.*, Vol. C-25, No. 2, pp. 172-178 (1976).
16) 荒川：オンライン文字認識に関する研究，北海道大学学位論文 (1981).
17) 田中：系列の文脈依存類似度，電子通信学会論文誌，Vol. J67-A, No. 6, pp. 612-613 (1984).

(昭和63年12月28日受付)
(平成元年9月12日採録)

沼倉 覚

昭和38年生．昭和62年3月宇都宮大学工学部情報工学科卒業．平成元年3月同大学院工学研究科情報工学専攻修了．同年4月リコー応用電子研究所(株)に入社．現在，情報システムに関する研究開発に従事．電子情報通信学会会員．

田中 栄一 (正会員)

昭和37年3月大阪府立大学工学部電気工学科卒業．昭和43年1月大阪大学大学院工学研究科博士課程通信工学専攻修了．工学博士．昭和42年4月大阪府立大学工学部電気工学科に勤務．昭和52年4月宇都宮大学工学部情報工学科教授，現在に至る．昭和49～51年パデュー大学客員研究員．言語認識およびパタン認識に関するアルゴリズムの研究に従事．

青木 晴海

昭和40年生．昭和63年3月宇都宮大学工学部情報工学科卒業．同年3月富士重工業(株)に入社．宇都宮製作所航空機工場電算機課に所属．工場内システムの開発，ソフトウェアの調査，テスト等に従事．

矢野目 毅

昭和41年生．昭和63年3月宇都宮大学工学部情報工学科卒業．同年4月シャープ(株)入社．電子機器事業本部に勤務し，映像機器のファームウェア設計に従事．

矢吹 勉

昭和39年生．昭和63年3月宇都宮大学工学部情報工学科卒業．同年4月富士通(株)入社．以来，POSシステムの開発に従事．