英文タッチタイピング練習プログラムにおける誤り検出アルゴリズム†

竹田尚彦†† 押切実†††*
河合和久†††† 大岩元†††**

英文タッチタイピング練習システムの誤り指摘方法には，入力時に誤りが生ずるとただちに警告を発する即時警告方式と，練習の区切りごとに一括して練習者にエラーの起こった箇所と種類，成績を報告する一括処理方式がある．即時警告方式は練習者のタイピング動作への集中を妨げるため，われわれの開発した英文タッチタイピング練習システム・TYPING では一括処理方式を採用した．このため，問題文と練習者の打鍵結果を比較して，誤り検出をする一括処理型の誤り検出アルゴリズムが必要となった．被打鍵文字列と打鍵文字列を比較することにより，タイプ誤りを検出，分類するアルゴリズムは，タイピング特性の解析やタイピング練習ソフトウェアにおける誤り検出に応用することができる．この誤り検出アルゴリズムには，練習者がタイプ動作中に誤ったと感じた箇所を正確に指摘する誤り同定の妥当性が要求される．従来の誤り検出アルゴリズムでは，2つの文の間の最長一致（LCS）を求め，問題文，打鍵結果および LCS を比較することにより，誤り検出をおこなっている．しかし，この方法では，LCS を求めるために $O(n^2)$ 計算量がかかり，さらに誤り同定のための後処理が複雑になる．筆者らは，タイプ動作中に生ずるエラーパターンに注目し，簡便なパターンマッチングにより，誤り検出が可能であると考えた．本論文では，タイピング時に生ずる誤りの特性を考慮し，誤りパターンのマッチング順序を工夫することにより，高々3字程度の先読みをほどこすだけで，比較の回数にして $2L$（L：入力テキストの長さ）程度の計算量で，人間の感覚に合致した誤り検出ができるアルゴリズムについて述べる．

1. まえがき

タイピング誤りの検出，分類するアルゴリズムを考えるとき，熟練タイピストを対象とするものと，初心者を対象とするものを別々に考える必要がある．熟練タイピストのタイピング動作は，テキストの先読み，指の運動キュー，実際の 10 本の指の運動が複雑にからみあっている．このため，熟練者の情報処理モデルを仮定して解釈しなければならないような誤りが生ずる[1]．一方，英文タイピング練習ソフトウェアを使用するような初心者には，熟練者にみられるような複雑な誤りは生じず，もっぱら，キー位置の記憶や，目標のキーへの指の運動の不正確さによるものが主体になる．本論文では，このような初心者のためのタイピング練習ソフトウェアに適用するタイピング誤り検出アルゴリズムについて考える．

初心者の場合，キー位置の記憶や指の運動の不正確さから，練習中に多くの誤りを生ずる．このため，タイピング練習ソフトウェアにおいては，練習中に生じた誤り情報を，どのような方法でタイピング動作への集中を妨げることなく練習者へ伝えるかが問題となる．ある種のタイピング練習システムでは，練習者の入力を逐次チェックし，即時にリアルタイムに警告を発し，正しい入力をおこなうまで先に進めないものが大部分である．しかし，この誤り指摘方法では，誤りを多くおかすであろう初心者は，タイピングの動作に集中することができない．

そこで練習者が一定時間，練習テキストをタイプし終るまで誤り指摘をおこなわず，終ってから一括しておこなう方法をとれば，練習者はタイピング動作に集中することができる．この場合には，練習テキストと打鍵文字列を比較することによりタイプ誤りを検出，分類するアルゴリズムが必要になる．

この誤り検出アルゴリズムには，1)誤り箇所の同定が正確であること，2)誤りの種類の同定が正確であること，3)高速であることが要求される．特に，1)，2)

† A Fast Algorithm for Detecting Errors of Novice Typists by NAOHIKO TAKEDA (Computer Center, Toyohashi University of Technology), MINORU OSIKIRI (Department of Information and Computer Science, Toyohashi University of Technology), KAZUHISA KAWAI (Department of Knowledge-based Information Engineering, Toyohashi University of Technology) and HAJIME OHIWA (Department of Information and Computer Science, Toyohashi University of Technology).
†† 豊橋技術科学大学情報処理センター
††† 豊橋技術科学大学情報工学系
†††† 豊橋技術科学大学知識情報工学系
* 現在 ヤマハ(株)
Present Affiliation YAMAHA Co. Ltd.
** 現在 慶應義塾大学環境情報学部
Present Affiliation Faculty of Enivironmental Information, Keio University

に関しては，練習者が練習時に誤ったと感じたところと一致している，つまり人間の感覚に適合するような誤り検出結果が得られなければならない．

われわれはこのような誤り指摘方法を用いたタイピング練習システム・TYPING を開発した[2),3)]．TYPING は，アルファベット小文字，大文字およびセミコロン，コンマ，ピリオドのタッチタイプを習得するための，CAI システムである．全課程を終了するには，約5時間を要する．豊橋技術科学大学では，電気・電子および情報工学系の学生に対して，プログラミング教育の導入部に，この CAI システムを用いたタイピング教育をおこなっている．

TYPING で使用するテキストは，P. S. Pepe の "Personal Typing in 24 hours"[4)] から引用している．このテキストは，すべてよく使用される英単語で構成されているため，テキストを単語あるいは句単位で指の運動パターンに変換していく機構が形成されやすい．われわれは，このテキストと熟練タイピストの認知モデルをもとに TYPING の指導方針，「手元を見ない・打ったテキストを見ない・誤りを気にしない・リズミカルに打つ」を決定した[4),5)]．

誤り指摘のための1つの方法として，問題文と打鍵結果を比較し一致する部分を見つけ，不一致の部分について，誤り解析をする方法がある[6)]．この方法では，まず2つの文字列の最長一致を見つける．最長一致を求めるアルゴリズムとして有名なものは，LCS (Longest Common Subsequence)[7)~9)] がよく知られている．LCS を求めることにより，挿入，削除は容易に検出できる．しかし，文字の交換エラーや，誤打鍵の検出には，後処理が必要となり，誤り検出アルゴリズムは複雑になる．また，一般に LCS を求めるには，$O(n^2)$ の計算量が必要であるため，LCS を用いる方法では処理に時間がかかる．

本論文で提案する誤り検出アルゴリズムは，LCS や問題文の単語情報を用いず，しかも，高速に誤りを検出することができる．このアルゴリズムでは，あらかじめ練習者が起こすであろう誤りパターンを想定しておき，問題文と練習者の打鍵の間に不一致が生じたら，誤りの生起確率の高い順に，誤りパターンの適合を調べていく．この方法では，誤りパターンの適合を調べるための先読みは最大3文字である．

本論文では，まず，TYPING の練習方式と，練習者に対する誤り指摘の方法について述べる．次に LCS を用いた誤り指摘の方法を紹介し，その問題点について述べた後，ファイル比較アルゴリズム・N/M 法に基づき，あらかじめ予想した生起確率の順に，誤りパターンのマッチングをおこなう，2/3 アルゴリズムを提案する．最後に，このアルゴリズムの評価と問題点について考察する．

2. TYPING の練習方法と誤り指摘

2.1 TYPING の練習方法

TYPING は，英字の位置を覚えるのに全10課の練習が必要で，各課は原則として3つのセクションに分かれている．各セクションは1分間ずつの練習を8回繰り返すようになっており，1課分の練習時間は約30分，全体の教程を終了するのに要する時間は，約5時間である．この教程では，アルファベット，コンマ，ピリオド，セミコロンと大文字の打ち方を習得することができる．

各課では，最初に新出キーを記憶するための導入部があり，その後で各セクションごとの練習に入る．1セクションの練習方法は，数個の文を1分間にできるだけ多くタイプするという練習を8回繰り返す，というものである．

1分間練習時の画面表示を図1に示す．この画面では，打った文字は，画面表示されない．練習者が1行入力を終了した時点で改行キーを押すと，次にタイプすべきテキストが表示される．

1分間の練習が終了すると，誤り指摘画面となり，各練習テキストと，練習者の打鍵列，誤り情報が表示され，最後に1分間の総打鍵数，誤り数および得点が表示され，練習者の励みとするようになっている．

2.2 タイプ誤り指摘

TYPING では，1分間の練習セクションを終了した後，元の練習エキストと練習者の入力を，誤り検出アルゴリズムにより比較することにより，タイプ誤りの指摘をおこなう．誤り指摘画面では，練習テキスト3行分に対し，

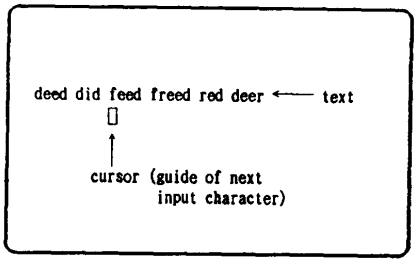


図1 練習画面
Fig. 1 Display of training session.

正解打鍵列　練習者が入力すべき練習テキスト
入力打鍵列　練習者が実際に入力した打鍵系列
誤り情報列　誤りの位置と種類を示したもの

を表示する．誤り指摘画面の表示を**図2**に示す．なお，誤り検出アルゴリズムで同定できる誤りの種類については次章で述べる．

練習者に対する誤り指摘は，練習者自身が，どこを間違えやすいかを知る目的に利用されている．練習者は，誤りの種類まで細かくチェックする必要はない．間違えた箇所を正しく打ってみる程度で先に進んでよい．このため，練習者にとっては，誤りの種類の報告は，あまり重要な意味は持たない．

しかし，この誤り指摘画面に表示される情報は，TYPING の使用記録として保存され，エラーの解析や，練習記録の分析に用いられる．このため，単に誤ったエラー箇所を指摘するだけでなく，誤りの種類を同定しておくことは，タイピング習得の特性解析のために重要である．

3. タイプ誤りとその検出

本章では，タイプ時に生ずる誤りについて検討を加え，TYPING における誤り検出アルゴリズムで検出できる誤りの種類について述べる．

3.1 タイピング時に生ずる誤りの種類

タイピング時に生ずる誤りは，いくつかの種類に分類される[10)~12)]．

イ）移動

イ-1. 2文字の交換

例：the → hte
　　a job → j aob

イ-2. 数文字内の交換

例：major → jamor

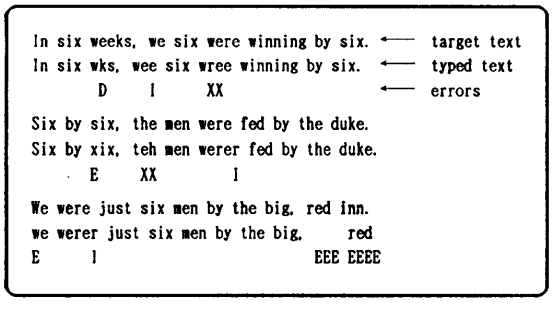


図2 誤り指摘
Fig. 2 Display of error information.

この例では，右手で打つ‘m’‘j’の交換が生じた

イ-3. 1文字の移動

例：that → atht

ロ）脱落　打つべき文字が欠落した

例：omit → omt

ハ）挿入　余分な文字が挿入された

例：and → asnd

ニ）置換　打つべきキーを違うキーで打った

ホ）多重打ち　同じキーを余分に打った

ヘ）繰返しパターン　繰返しパターンで，パターンの順序が入れ替わっている

例：these → thses

これらの誤りは文字単位で生ずる誤りである．この他に，単語単位での挿入／脱落／置換なども生じると考えられる．しかし，TYPING を使用するような初心者では，まだ単語あるいは句単位で指の運動パターンに変換する機構が十分できあがっていないため，単語単位での誤りは生じにくい．したがって，ここでは単語単位での誤りは考慮しないことにする．

ここであげた誤りの種類を，正解列と入力列から正確に求めることは難しい．単語単位で誤っているか，指のポジションおよび指使いの誤りであるのか，単なる打鍵タイミングの制御の誤りであるのかなど，いくつかの解釈が考えられるからである．このため，正確な判断をするためには，ビデオによる記録が必要になる[12)]．

3.2 誤り検出アルゴリズムで扱う誤りの種類

われわれの開発した誤り検出アルゴリズムは，正解列と入力列を1パスするだけで，誤り検出をおこなうことを目的としている．このため，入力列中に誤りがある場合は，あらかじめ想定した誤りパターンとのマッチングをとることにより，誤りを同定することにした．

このため，多くの誤りパターンを想定すると，パターンマッチングが複雑になったり，何通りかに解釈される場合が生じる．そこで，われわれの誤り検出アルゴリズムでは，次の4通りの誤りの種類に単純化して考えることにした．

誤打鍵　正解打鍵列のある文字が，別の文字で入力された．‘E’で表す
挿入　正解打鍵列にはない文字が挿入された．誤り情報列では，‘I’で表す
脱落　正解打鍵列であるべき文字が入力されず脱落した．‘D’で表す

交換　正解打鍵列の続いた2文字が，入力打鍵列では順序が入れ替って入力されている誤り情報列では，'X' で表す

2文字の交換以外の移動パターンや置換は，挿入および脱落，誤打鍵が複合して起きたとして扱うことにする．たとえば，前記の these→thses と誤った繰返しパターン誤りの例は，Rumelhart らによれば特殊な3つ組打鍵プログラムの起動誤りと解釈されている[1]が，本アルゴリズムでは，単純に 'e' が脱落し，末尾に余分な 's' が挿入されたと解釈する．

4. LCS のタイプ誤り検出アルゴリズムへの適用

本章では，LCS について述べた後，LCS のタイプ誤り検出への適用と，その問題点について述べる．

4.1 LCS アルゴリズム

LCS アルゴリズムは，2つの文字列を比較し，その部分文字列の最長一致を見つけるためのアルゴリズムである．

ここで，文字列A，B，Cを考え，これらの文字列の長さは，それぞれ m, n, p であり，文字列は

$A = a_1 a_2 \cdots a_m$

$B = b_1 b_2 \cdots b_n$

$C = c_1 c_2 \cdots c_p$

であると仮定する．

このとき，文字列Aのうち $m-p$ 個の文字を削除した文字列がCと等しいとき，文字列Cは文字列Aの部分文字列 (subsequence) であるという．文字列Aのうちから $m-p$ 個の文字，文字列Bから $n-p$ 個の文字を削除した文字列が，どちらも文字列Cになるとき，これをAとBの共通部分文字列 (common subsequence) という．LCS (Longest Common Subseqence) は，これらの共通部分文字列のうちの，もっとも長いものをさす．

例えば，2つの文字列

typing　(1)

と

jtypgn　(2)

の LCS は，

LCS = {typg, typn}　(3)

の2つが求められる．LCS を求めるアルゴリズムは種々の検討[7]がなされており，Hirschberg[8),9)] が高速なアルゴリズムを開発している．

4.2 タイプ誤り検出への適用

前節の例で，(1)を正解打鍵列，(2)を練習者の入力打鍵列とする．(3)は，この2つから求められた LCS である．この場合，同じ長さの共通部分文字列が得られるので，LCS が2つ存在する．

タイプ誤りを検出するには，(3)と(1)，(3)と(2)の文字列の一致を，先頭から順に調べる．この結果から，正解打鍵列に対して，どの文字が挿入／脱落して入力打鍵列に変化したかを知ることができる．挿入／脱落は，

挿入された文字　(2)にあって(3)にない文字

脱落された文字　(1)にあって(3)にない文字

により判定することができる．

前出の例では，LCS が2つ存在するので，図3のように2通りの結果を得る．この結果から，先頭に "j" が挿入され，"ing" の文字列のうち，"i" が脱落し，"ng" の文字列の変化が2通りに解釈されているのが分る．

4.3 問題点

LCS を用いた誤り検出方法で，LCS を求めることができれば，挿入／脱落レベルの誤り検出は簡単におこなえることが分った．しかし，この方法では，次にあげるような欠点がある．

1) LCS を用いたアルゴリズムでは，挿入/脱落しか検出できない．

前出の誤り検出結果1では，"in" が削除され，末尾に "g" が付加されたと解釈される．しかし，"i" が脱落し，"g" と "n" に交換が生じたと解釈するほうが自然である．なぜならば，"n" と "g" は，左手-右手の交互打ちとなり，打鍵動作のタイミング制御のミスにより，交換が生じることは，よく知られている[10]からである．

また，

abc → adc

のように，単純に文字 "b" を文字 "d" と，ミスタイプした場合，"b" が脱落したと解釈される．

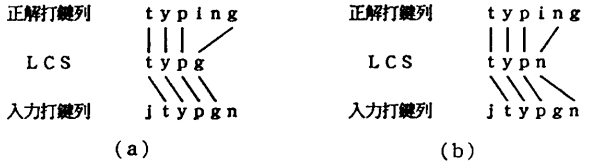


図3 LCS を用いた挿入／削除の検出

Fig. 3 An example of detecting insertion/deletion errors by LCS method.

このような事態を避けるためには，求められたLCS同士を比較するか，あるいは削除された文字と元の打鍵列との関係を調べることにより，交換やミスタイプを検出する必要がある[6]．したがって，LCSを求めた後，エラー箇所の同定のための，なんらかの後処理が必要となる．

2) 計算量が大きい

LCSを求めるための計算量は，最悪の場合，$O(mn)$になることが知られている[7]．ここで，mおよびnはLCSを求めたい文字列のそれぞれの長さである．Hunt[13]やHirschberg[9]は，計算量を小さくするための改良アルゴリズムを提案している．Hirschbergは，この論文のなかで，2つのアルゴリズムを提案しており，それぞれの計算量は，$O(pn+n\log n)$，$O((m+1-p)p\log n)$，(ただし，pは求められたLCSの長さ，$m \geqq n$と仮定)であり，$N\log N$のオーダであるとしている．

LCSを用いた誤り検出は，まずLCSを求め，LCSから文字レベルでの挿入／脱落を求め，さらに誤入力や文字の交換を検出するという3パスが必要となる．このため，アルゴリズムが複雑になり，LCSを求める計算量に後処理の時間が加わるため，計算量はさらに増大する．

5. 誤りの生起確率を考慮した誤り検出アルゴリズム

TYPINGでは，ファイル比較アルゴリズム・N/M法[14]に基づいた，誤り検出アルゴリズムを使用している．

本章では，N/M法について簡単な説明をした後，タイプ誤りの生起確率についての仮定をたてる．次に，この仮定をもとに誤りのパターンマッチングをおこなうことにより，誤り検出をおこなう2/3アルゴリズムについて述べる．

5.1 ファイル比較アルゴリズム・N/M法

われわれの開発した誤り検出アルゴリズムは，N/M法と呼ばれる，ファイル照合アルゴリズムに基づいている．このアルゴリズムは，DEC社のFILECOMという，2つのファイル間の相違を調べるツールに用いられている．

N/M法は，2つのファイルを先頭から一行ずつ比較しながらスキャンしていく．もし，不一致点が検出されたならば，N行の連続して一致する部分が見つかるまで，ファイルの後の方をスキャンしていく．ただし，先読みをいくらしても，N行の連続した一致が見つからないことがあるので，先読みはあらかじめ指定したM行目までの範囲に限っておこなうものとする(図4)．このM行内にN行の一致が見つからなければ，$M+1$行目より再び一行ずつの比較をおこなう．

N/M法は，N(連続して一致する行数)，M(先読み行数)の決めかたによって，結果が大きく左右されるため，ユーザが問題に合わせて，NおよびMを調整する必要がある[14]．

5.2 誤りの生起確率についての仮定

正解打鍵列が，いくつかの基本操作によって，入力打鍵列に変化すると仮定する．ある打鍵1文字に注目した時，次のような操作パターンが考えられる．

イ) 正しい打鍵　正しい文字の入力がなされた
ロ) 誤打鍵　別の文字を打鍵した
ハ) 挿入　余分な文字が打鍵された
ニ) 脱落　打つべき文字が打鍵されなかった

これを，表1に示す．

ここで，正しい打鍵がおこなわれる確率をPn，誤打鍵をする確率をPe，挿入が生ずる確率をPi，脱落が生ずる確率をPdとする．この時，これらの生起確率の間には，次のような関係が生ずると仮定する．

$$Pn \geqq Pe \geqq Pi \geqq Pd \qquad (4)$$

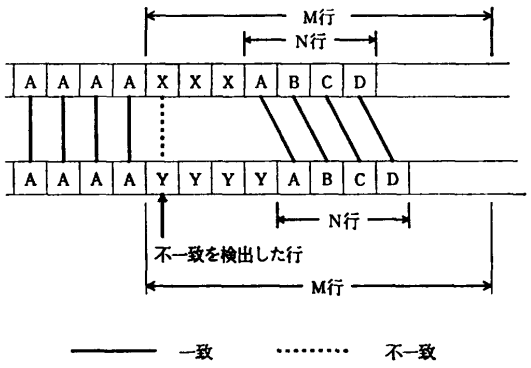


図4 N/M法の概要
Fig. 4 Concept of N/M method.

表1 検出できるタイプエラーの種類
Table 1 Kind of detected typing errors.

エラーの種類	例	エラー指摘時の表示	誤りの生起確率
正しい打鍵	deed→deed		Pn
誤打鍵	deed→dead	E	Pe
挿入	deed→deedd	I	Pi
削除	deed→ded	D	Pd

これらの生起確率の大小関係は，個人差が生じるため一様ではないと考えられる．Grudin[12]の報告によれば，熟練タイピストのエラー特性は，個人差があるものの，挿入エラーが多く，次に脱落と誤打鍵がほぼ同じ程度のエラーの発生率になっていると報告されている（表 2）．しかし，この結果をみる限りでは，個人差が大きく順序関係は，一定していない．一方，初心者のタイピストでは，キー位置が正確に把握されていないためか，誤打鍵，挿入，脱落の順である．実際に，初心者の練習記録を分析してみると，6章で述べるように，誤りの頻度が誤打鍵，挿入，脱落の順に生じていることが報告された．

表 2 エラーの種類とその割合
Table 2 Typing speed and error rates.

被験者	タイプ速度 (wpm)	エラー率 (%)	エラーの種類と全エラーに対する割合 (%)				
			挿入	脱落	置換	移動	その他
1	90.4	1.1	28	21	27	10	13
2	65.6	0.5	17	11	69	3	0
3	76.1	1.9	75	5	6	4	10
4	74.9	1.0	47	15	4	4	29
5	61.3	0.4	53	19	9	11	9
6	81.9	0.8	35	12	21	11	20
初心者	20.0	3.2	9	4	75	4	11

from Grudin[11] p. 126

タッチタイピング練習プログラムは，キーボードの経験がないものに対して実施する．このため，(4)の仮定は，熟練タイピストの特性を表してはいないが，初心者のエラー特性をよく表している．

5.3 2/3 アルゴリズム

本来の N/M 法は，ファイルを行単位に比較し，相違点を見つけるためのアルゴリズムである．われわれは，これを文字単位に比較することにより，2つの文字列の比較に応用した．本アルゴリズムは，2文字の連続した一致，1文字の一致および文字の交換を，最大先読み3文字以内で一致を見つけることから，N(連続する一致の最大長さ)=2, M (先読みの最大数)=3 の場合に相当する．このことから，本英文誤り指摘アルゴリズムのことを 2/3 アルゴリズムと呼ぶことにする．

本アルゴリズムを図 5 に示す．

2/3 アルゴリズムは，入力打鍵列と正解打鍵列を先頭から順に比較していく．もし，文字の不一致が見つかれば，予測している誤りパターンと一致するかどうか調べる．もし，一致する誤りパターンがあれば，誤り情報列に書き込み，次へポインタを進める．もし，予め想定した誤りパターン以外ならば，1字誤りとして正解打鍵列および入力打鍵列を，ともに1文字読み飛ばし，次の一致を確かめる．

```
R[] : 正解打鍵列
I[] : 入力打鍵列
E[] : エラー情報列

{初期化する}
rp = 正解打鍵列の先頭
ip = 入力打鍵列の先頭
ep = エラー情報列の先頭

while R[rp],I[ip]のいずれもEOSでない do begin
  if R[rp] = I[ip] then begin
    E[ep] := 空白
    rp,ip,epを1つ進める end
  else begin
    {エラーパターンのマッチングをする}
        2／2誤りパターンマッチングをする
        文字の交換誤りマッチングをする
        1／3誤りパターンマッチングをする
    E[ep]にエラー種別を書く
    必要分だけrp,ip,epを進める
  end
end
```

図 5 2/3 アルゴリズム
Fig. 5 Algorithm of 2/3 method.

誤りパターンのマッチングは，

1) 2/2 誤りパターンマッチング
2) 文字の交換誤りマッチング
3) 1/3 誤りパターンマッチング

の順におこなう．

1/3 誤りパターンマッチングは，5.2 節で述べた誤りの生起確率に基づいて設計した誤りパターンマッチングである．2/2 および文字交換のパターンマッチングは，1/3 では，検出することのできない誤りをあらかじめ検出しておく，いわばプリプロセッサ的な役割をするものである．

5.4 誤りパターンのマッチング

5.4.1 1/3 誤りパターンのマッチング

正解打鍵列と入力打鍵列を，文字列の先頭から照合していき，不一致が検出されたと仮定する．この時，生じた誤りが誤打鍵による誤りで，例えば

bc → dc

のように，“b”を“d”と誤って打鍵し，次の文字は正しく打鍵された場合を考える．このような文字列が生じる確率は，

$$PePn \qquad (5)$$

となる．

同様に，1文字挿入が生じた次の文字は正しく打鍵された場合，1文字脱落が生じて次の文字が正しく打

鍵された場合の生起確率はそれぞれ

PiPn

PdPn

となる．これを誤り生起確率の仮定から大小関係をきめると

$$PePn \geqq PiPn \geqq PdPn \quad (6)$$

となる．

打鍵誤りを検出するために，起こり得る確率の高いものからマッチングを調べていけば，誤りを早く検出でき，効率のよいアルゴリズムになる．

この時，誤りが複合して生じたり，文字の挿入により一致点がずれてしまうことがあるので，さらに3文字先まで先読みをして誤りパターンのマッチングをおこなう．この時，マッチングをとる順序は，

$$\begin{aligned} &PePn \geqq PiPn \geqq PdPn \\ &\geqq PePePn \geqq PePiPn = PiPePn \\ &\geqq PePdPn = PdPePn \geqq PiPiPn \\ &\geqq PdPdPn \geqq PePePePn \cdots \end{aligned} \quad (7)$$

とする．このことから，誤りパターンとの一致のチェックを**図6**のような順序でおこなうことにする．

5.4.2 1/3 アルゴリズムの問題点

この誤りパターンマッチングで，大部分の誤りを検出することができるが，次のような誤りパターンが検出することができない．

イ）同一文字の2字連続

英文タイプの練習テキストでは，“deed” や “feed” などのような，同一文字が連続して出現する場合がある．

例えば，

deed→ddeed

と誤った場合，先頭にdが余分に挿入されたと解釈すべきである．しかし，前述のアルゴリズムでは，**図7**-(a)のように誤り検出がおこなわれ，2文字目の“d”と，4文字目の“e”の2文字が間違ったと解釈される．また，スペースをはさんで，同一文字が続くような場合も，図7-(b)のように誤って解釈される．

ロ）文字の交換

文字の交換誤りは，1/3 では複合的な誤りとして解釈してしまう．例えば，

directory→driectory

は，rの前にiが挿入され，eの前のrが脱落されたと解釈される（図7-(c)）．

5.4.3 2/2 誤りパターンマッチング

同一文字の2字連続，文字の交換のような連続した2文字について，誤りを検出するためには，2文字の連続した部分について比較をすればよい．このために，本アルゴリズムでは，1/3 で一致を検出する以前に，2/2で2文字の連続した一致を検出する．

2/2 での誤りチェック順序を**図8**に示す．2字の連続した一致を検出するために，誤打鍵，挿入，脱落の順にチェックする．このチェック順序も，先に述べた初心者タイピストの誤り特性にもとづいている(図8-(a)，(b)，(c))．

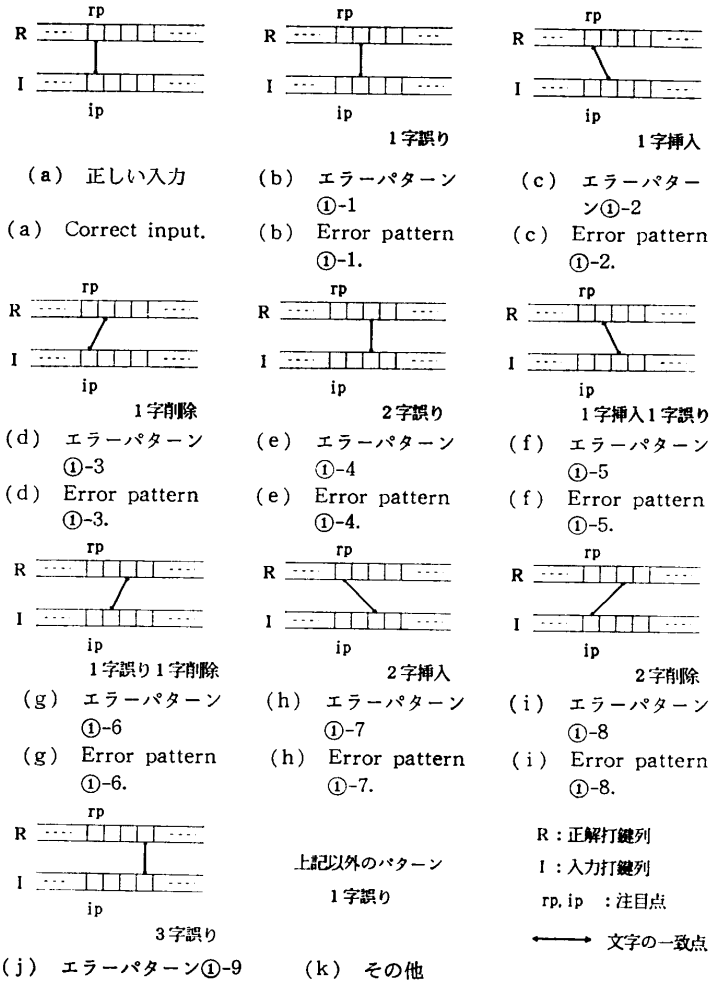


図6 1/3 エラーパターン・マッチング
Fig. 6 Error pattern matching by 1/3 method.

5.4.4 文字交換のマッチング

タイピング動作の中で，顕著に起こる誤りの例として，文字の交換がある．右手-左手の交互打ちになっている場合，打つべきキーは正しいが，タイミングが制御されていない時，文字の前後交換が生じる．

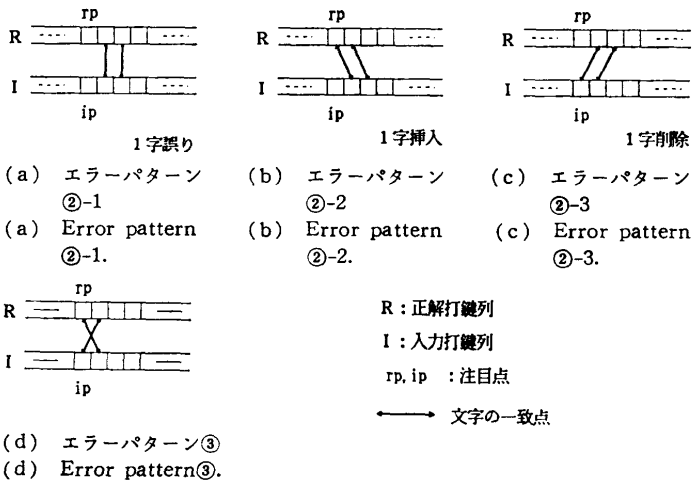
このため，文字の交換は1/3誤りパターンマッチングをする前に，マッチングをとっておく（図 8-(d)）．

5.4.5 同一文字の多重打ちの検出

初心者が TYPING を使用して，タイプ練習をおこなうと，同じキーを押し続ける誤りが顕著に現れることが分った．この誤りは，すべての誤りの約 10% に達している（6章参照）．

誤りパターンのマッチングをとるだけのアルゴリズムでは，先読みよりも長い同一文字の多重打ちが入力打鍵列中に生ずると，テキストとの一致点が見つけられず，多重打ち発生以降の誤りを正しく検出することができない．この例を，**図9**に示す．

同一キーを3文字以上連続して打つ単語は，一般の英文に現れてくることは稀であり，初心者に顕著な誤りであることから，入力打鍵列に3文字以上の同一キーの多重打鍵が生じた場合，これを読み飛ばすように改良した．実際のインプリメントでは，2/3 アルゴリズムの前処理として，同一キーの多重打ちを検出しておき，その後，2/3 アルゴリズムによる誤りパターンマッチングをおこなうようになっている．

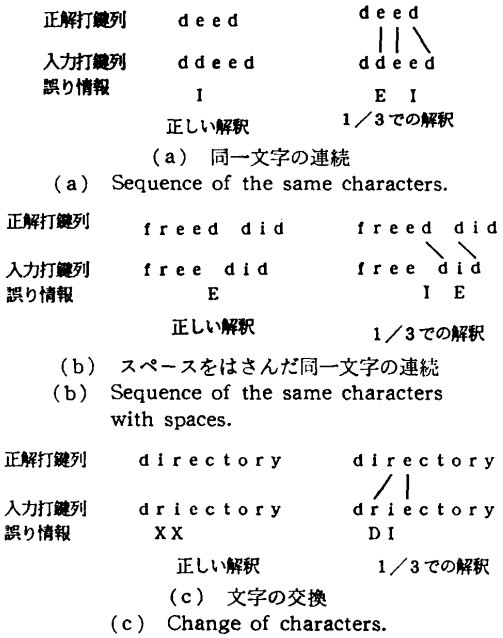


(a) エラーパターン②-1
(a) Error pattern ②-1.
(b) エラーパターン②-2
(b) Error pattern ②-2.
(c) エラーパターン②-3
(c) Error pattern ②-3.
(d) エラーパターン③
(d) Error pattern③.

図 8 2/2 エラーパターンマッチング ((a)～(c)) と交換エラーパターンマッチング (d)
Fig. 8 Error pattern matching by 2/2 method ((a)～(c)) and matching of change two characters (d).

```
正解打鍵列：  the big men deeded the big, red inn
入力打鍵列：  the bbbbbbbbbbbbbig men deeded the big, red inn
誤り情報列：        EEEEEEEEEEEEEE E I EEEEEE EE DI EEEEEEEEEE
```

図 9 同一文字の多重打ちの誤解釈例
Fig. 9 Example of wrong detection on doubling errors.



(a) Sequence of the same characters.
(b) Sequence of the same characters with spaces.
(c) Change of characters.

図 7 1/3 パターンマッチングで生ずる誤った解釈例
Fig. 7 Examples of wrong detection of errors by 1/3 method.

6. 誤り検出アルゴリズムの計算量

TYPING を使用した6人の初心者の練習記録について，本誤り検出アルゴリズムにより誤り検出をおこなった．この結果，全課程を通じての総打鍵数，総エラー数，およびエラー率は**表3**のとおりであった．エラー率は，最大15.0%から最小3.1%と個人差によっ

表 3 練習者のエラー率
Table 3 Trainees' type error rates.

練習者	総打鍵数	総エラー数	エラー率(%)
1	34334	5156	15.0
2	36975	1587	4.3
3	37948	1176	3.1
4	41143	5024	12.2
5	32341	3623	11.2
6	34678	2478	7.1
平 均	36236.5	3174	8.8

てばらついている．平均エラー率は8.8%であった．

表4は，全エラーに対する各誤りパターンの発生率を示したものである．2/2アルゴリズムによる，誤打鍵，挿入，脱落の順にエラー発生が多く，次第に減少しており，ほぼアルゴリズム設計時の最初の仮定どおりの結果となっている．

また，前処理で検出される同一キーの多重打ち誤りの発生率は，全エラーに対して11%であった．

2/3アルゴリズムの計算量を，入力打鍵列の長さをLとして，エラーの平均発生率を10%，エラーの分布は，実データの表4にしたがうと仮定し，次式により計算した．

$$\{0.90\pm0.10(r_1c_1+r_2c_2+\cdots+r_nc_n)\}L$$

ここで，r_nは，パターンの発生率，c_nは，そのエラーパターンを認識するために必要な比較の回数である．r_nc_nの部分を計算したものを，表4に示す．

この結果，

$$(0.90+0.10*8.57)L=1.76L$$

となり，約$2L$回の比較回数で，計算できることが分かった．

7. 誤り検出アルゴリズムの評価

7.1 誤り検出例

本アルゴリズムで，検出した誤りの例を図10に示す．

図10のa)からd)までは，典型的な誤りの解析結果である．

a)では，文字の挿入および脱落を検出している．

b)では，スペースとcが交換され，“th”が誤打鍵であると解釈している．

c)，d)では，交換と挿入の誤りが確認されている．練習者は，誤り検出の表示を見ることにより，“quite”の“te”を交換して打鍵したり，右手のホームポジションにおいて，小指がしばしば動いて“;”を打鍵してしまうなどの，練習者自の癖を確認することができる．

本アルゴリズムは，不一致が検出された時点から，タイプ時の誤り特性に従って，誤りパターンにあてはまるかどうかを，調べていくだけの単純なアルゴリズムである．しかし，ほぼ人間の主観にあった誤り検出をおこなっており，実用上問題は生じていない．

しかし，本アルゴリズムでは，単語レベルでの誤りチェックをおこなっていないため，図10-e)のような単語の欠落が生じた場合，それ以降の解釈が正しくなされない場合がある．

7.2 誤り同定の正確さ

本アルゴリズムでは，あらかじめ想定した誤りパターンに適合しないものは，約10%であった．これらのパターンにあてはまらない誤りを分析してみると，以下のようにいくつかに分類された．

イ）手の位置，指の位置の間違い

表4 エラーパターンとエラー分布
Table 4 Error distribution.

	エラーパターン	エラー分布 r_n (%)	比較回数 c_n	$r_nc_n/100$
1字誤り	②-1	26.10	3	0.78
1字挿入	②-2	17.62	5	0.88
1字削除	②-3	3.19	7	0.22
文字交換	③	0.93	9	0.08
1字誤り	①-1	6.74	11	0.74
1字挿入	①-2	4.65	13	0.60
1字削除	①-3	1.77	14	0.25
2字誤り	①-4	5.37	15	0.81
1字挿入 1字誤り	①-5	4.49	16	0.72
1字誤り 1字削除	①-6	2.05	17	0.35
2字挿入	①-7	2.36	18	0.42
2字削除	①-8	0.80	19	0.76
3字誤り	①-9	3.78	20	0.76
その他	others	9.05	20	1.81
多重打ち	—	11.1	—	—

```
a) 正解打鍵列:  deed did feed freed red deer
   入力打鍵列:  deed did feedd reed redd deer
   誤り情報列:              I D       I

b) 正解打鍵列:  the judge did free the civic three
   入力打鍵列:  the judig did free the ciji cc ree
   誤り情報列:       I D              E XXEE

c) 正解打鍵列:  The man were quite happy; the pay was quite ample.
   入力打鍵列:  THhe man were quiet happy, yhe pay was quiet ample.
   誤り情報列:   I              XX      E E             XX

d) 正解打鍵列:  we were six, the first six in this sixty
   入力打鍵列:  wew were dix ,;the ;fir;st ;six ;in this sixty
   誤り情報列:    I      E  XXI    I   I   I    I

e) 正解打鍵列:  the big judge did free the big duke
   入力打鍵列:  the big judge free the big duke
   誤り情報列:              EEEE EE DEEE EEEE
```

図10 誤り検出の例
Fig. 10 Examples of error report by 2/3 method.

ホームポジションや使用する指が誤っている場合.

ロ）入力文の末尾での誤り

入力打鍵列の末尾の文字を間違えた.

ハ）単語単位での挿入や脱落

単語を余分に打鍵したり，打つべき文字列のうちの，ある単語や単語の一部を打鍵しなかった場合.

ニ）テキストの連結

改行を忘れて次の行のテキストあるいは同じテキストを打った場合.

ホ）無意味な打鍵

テキストと全く関係のない文字列が入力された場合.

ヘ）先読みで同定可

先読みを3ないし4文字先まですれば，誤り同定が可能だったもの.

これらの誤りの同定できなかった誤り全体に対する割合を表5に示す.

ポジション誤りは，ホームポジションがずれているため全く一致が見つからずに，その他の誤りとなる. 同様に末尾での誤り，無意味な打鍵や余分なテキストの連結も，前方にマッチする部分が見つからないので，すべて誤りとなる. これらは本質的に誤りと解釈してよく，現在のアルゴリズムのように，「その他の誤り」と解釈してよい. これらの占める割合は，約60% であった.

単語あるいは一部の文字列単位の挿入や脱落は，現在のアルゴリズムでは検出することができない. しかし，TYPING のように，初心者にテキストを忠実にタイプすることを要求するようなソフトウェアでは，単語あるいは一部の文字列単位の挿入や脱落は，あまり生じない. 実際，この種の誤りはマッチングできなかった誤りのうちの約 40% を占めるが，全誤りのうちの4% にすぎない. ただし，本アルゴリズムをタイピングの特性解析に使用する場合には，さらに正確な誤りチェックが必要となる. 単語レベルでの誤り検出を加えることにより，ほとんどの誤りを検出できると考えられる.

表 5 マッチングできなかったエラーの要因と割合
Table 5 Ratio and factors of error that did not match error patterns.

種　　類	割合 (%)
ポジション誤り	17.6
文字列の挿入	16.8
文字列の脱落	20.2
テキストの連結	9.6
末尾の誤り	11.6
無意味打鍵	16.0
先読みで同定可	3.0

いくつかのマッチしない誤りのうち3% が，先読みを3ないし4文字まですれば，解釈可能なものであった. しかし，これは誤り全体の約 0.3% 程度にすぎない. このことから，先読みの文字数は，現状の最大3文字で実用上，十分であると考えられる.

8. ま と め

本論文では，英文タッチタイプ練習システム・TYPING で使用している，誤り検出アルゴリズムについて述べた. この誤り検出アルゴリズムは，初心者のタイピングの誤り特性をあらかじめ考慮に入れることにより，高々3文字の先読みをし，パターン・マッチングをするのみである. このアルゴリズムを，実際の初心者の練習記録に適用し，実データをもとに計算の手間を計算してみると，入力打鍵列の長さ L に対して，比較回数にして $2L$ 程度の手間で，誤り検出の種果を得ることができる.

われわれが最初に仮定した誤りパターンで，約90% の誤りが検出できることが分った. 残りの検出できない誤りは，単語の欠落等である. しかし，本アルゴリズムを用いているブラインドタッチ練習システムでは，練習者には自然な英文の練習文をなるべく正確に入力することを求めており，単語の欠落は，打つべきものが打つべき時に打鍵されなかったとして，以降の入力を誤りとして実用上問題がない.

本アルゴリズムは，最初 20 台の TSS 端末を有する MELCOM 700 (0.7 MIPS) 上の TYPING の誤り検出ルーチンとして開発された. このため，誤り検出アルゴリズムが高速であることは，重要な条件の1つであった. しかし，ハードウェアの計算能力が向上した現在では，高速性は重要な条件ではない. これによって生じた計算能力の余力を単語レベルのチェック等に振り向けることにより，さらに誤り同定の正確さを増すことができ，タイピング特性解析などにも応用することができる.

現状のアルゴリズムでは，高速化のために誤りパターンのマッチング順序が固定になっており，アルゴリズムの変更のためには，再コンパイルが必要である. 今後は，誤りパターンとマッチング順序を，ルールとして外部から与えられるように改良し，問題の対象ごとに，マッチングルールを変更できるようにした

いと考えている．この改良により，個々の練習者のタイピング特性に合せた，誤り検出も可能であろう．

謝辞 本論文をよりよいものにするために，適切かつ丁寧なご助言を多数いただいた査読者の方に感謝いたします．

参考文献

1) Rumelhart, D. E. and Norman, D. A.: Simulating a Skilled Typist—A Study of Skilled Cognitive-Motor Performance, *Cognitive Science*, Vol. 6, pp. 1-36 (1982).
2) 大岩 元，高嶋孝明：TSS によるタッチタイプトレーニングシステム，電子通信学会研究会技術報告，ET 79-12, pp. 37-42 (1979).
3) Takeda, N., Kawai, K., Takashima, T. and Ohiwa, H.: TYPING: A Computer Program for Touch-Typing, *Proceedings International Conference on Advanced Research on Computer in Education*, pp. 175-180 (1990).
4) Pepe, P. S.: *Personal Typing in 24 Hours* (Rev. 5 th ed.), McGraw-Hill (1985).
5) 河合和久，大岩 元：タイピング教育のための認知モデル，人工知能学会研究会資料，SIG-HICG-8801-2 (1988).
6) 平賀 譲，小野芳彦，山田尚勇：日本語タッチタイプ用の練習システムの作成，第 25 回情報処理学会全国大会論文集，pp. 1089-1090 (1983).
7) Aho, A. V., Hirschberg, D. S. and Ullman, J. D.: Bound on the Complexity of Longest Common Subsequence Problem, *J. ACM*, Vol. 23, No. 1, pp. 1-12 (1976).
8) Hirschberg, D. S.: A Linear Space Algorithm for Computing Maximal Common Subsequences, *Comm. ACM*, Vol. 18, No. 6, pp. 341-343 (1975).
9) Hirschberg, D. S.: Algorithms for Longest Common Subsequence Problem, *J. ACM*, Vol. 24, No. 4, pp. 664-675 (1977).
10) Norman, D. A. and Rumelhart, D. E.: Studies of Typing form the LNR Research Group, *Cognitive Aspects of Skilled Typing* (Cooper, W. E. ed.), pp. 45-65, Springer-Verlag (1983).
11) Gentner, D. R., Grudin, J. T., Larochelle, S., Norman, D. A. and Rumelhart, D. E.: A Glossary of Terms Including a Classification of Typing Errors, *Cognitive Aspects of Skilled Typing* (Cooper, W. E. ed.), pp. 39-43, Springer-Verlag (1983).
12) Grudin, J. T.: Error Pattern in Novice and Skilled Transcription Typing, *Cognitive Aspects of Skilled Typing* (Cooper, W. E ed.), pp. 121-143, Springer-Verlag (1983).
13) Hunt, J. W. and Szymanki, T. G.: A Fast Algorithm for Computing Longest Common Subsequences, *Comm. ACM*, Vol. 20, No. 5, pp. 350-353 (1977).
14) Heckel, P.: A Technique for Isolating between Files, *Comm. ACM*, Vol. 21, No. 4, pp. 264-268 (1978).

(平成 3 年 11 月 22 日受付)
(平成 4 年 7 月 10 日採録)

竹田 尚彦（正会員）
1958 年生．1982 年名城大学工学部電気工学科卒業．同年，(株)金陵入社．メカトロニクス関連ソフトウェアの開発に従事．1984 年より社員資格のまま豊橋技術科学大学大学院へ進学．1990 年同大学博士後期課程単位取得退学．工学修士．現在，同大学情報処理センター助手．HCI，ソフトウェア工学に関する研究に徒事．ソフトウェア開発における人間的要因について興味をもつ．

押切 実
1957 年生．1981 年豊橋技術科学大学工学部情報工学科卒業．1983 年同大学院修士課程修了．同年ヤマハ(株)入社．以来，LSI 設計システムの開発に従事．

河合 和久（正会員）
1958 年生．1981 年大阪大学基礎工学部情報工学科卒業．1986 年同大大学院基礎工学研究科博士課程修了．工学博士．同年豊橋技術科学大学情報工学系助手．1991 年同大知識工学系講師，現在に至る．計算機を用いた教育，創造的活動における計算機支援，HCI などの研究に従事．電子情報通信学会，人工知能学会，AAAI，日本ソフトウェア科学会などの会員．

大岩 元（正会員）
1942 年生．1965 年東京大学理学部物理学科卒業．1971 年同大学院博士課程修了．理学博士．同年東京大学理学部助手．1978 年豊橋技術科学大学情報工学系講師．1980 年同助教授．1985 年同教授．1992 年慶應義塾大学環境情報学部教授．1974 年～1976 年英国ケンブリッジ大学キャベンディッシュ研究所客員研究員（ブリティッシュ・カウンシル・スカラ），1979 年～1980 年米国コーネル大学応用物理学科客員準教授．キーボード入力，情報教育，ソフトウェア工学などの研究に従事．