文科系の計算機利用　III
—旧字体活字の文書の自動読み取りの改良

小野芳彦

1．はじめに

OCR（光学文字読み取り装置）は印刷文字の自動的な計算機入力に威力を発揮する機器である。性能は、現代の明朝体活字に対して最大になるように最適化されている。これは情報機器を商品として提供する場合、当然の方策である。必要な情報が活字化されているとしても現在文科系の諸分野においては昭和30年までの旧字体の活字を用いた文献が大半を占めている。特に、資料集として辞書的な利用をするもの、つまり、データベース化のメリットの高いものが旧字体活字で印刷されていることが多い。従来、通常の方法では、これら旧字体活字の文書はOCR読み取りが実用にならないと考えられてきた。本稿は、読み取り装置そのものの改良ではなく、後処理を効率化することによって実用的な読み取り率を得ようとする試みである。

1．1　予備的調査の結果

図1～2は、御触書寛保集成[(1)]巻四十二の冒頭部1ページを国際日本文化研究センターに設置の日本語OCR[(2)]で読み取った結果と、それを人手で訂正したテキストである。図1には、OCRが認識を誤った文字に3種類の印を振ってある。線で囲んであるものは1文字を分解して2文字と認識したものである。（同様に、1文字を前後に分解した部分が前あるいは後の文字と合成されて誤認するケースもある。この例には出現していない。）波線を振ってある文字は起こしやすい誤認の結果であり、かなり再現する。傍線を振ってある文字は、誤認の結果がばらついているものである。

この図の例の場合、空白を含めない文字全体の正解率は82%である。行ごとに正解率を求めその標準偏差を計算すると、0.09あり、標準偏差の2倍の範囲で64%から100%の正解率に落ち着くことを予想させる。したがって、OCR読み取りのまま後編集を行なうと、（以下、空白文字を含めた計算上）平均で約6文字に1文字の訂正が必要で、悪い場合は約3文字に1文字の訂正ということもありうると予想される。

ところが、再現性の高い前2者のタイプの誤認を自動的に訂正する手段が適用できると、後編集の状況はかなり改善できることになる。本例でこのような自動訂正が可能であると仮定すると、予想認識率は93%（標準偏差0.07）であり、後編集で平均約14文字に1文字の訂正率となる。

以上のように、理想的な状況がもし得られるならば、旧字体の印刷物のOCR読み取りに実用化の道を開く可能性がある。

御鯛書寛保集成四十二

一廻船井川船等之部
一御堀井所々橋塵芥拾場等之部
廻船井川船等之部可石之御胸書、ケ條入交り、外之部二有之候分
一明暦一丸未年十一丹、廻船船逆を明ゲ、舟つかへ候ハぬ株二可掛世依、御堀#所々橋等之部二右之、
一天和一丸酉年十月、廻船之塵芥川え不拾、傳嬬船二て永代蛙御定之埃松場え避可巾儀、御堀井所々橋等之部に右之、
一一丸祇三午年七且、倦形船御定之寸尺より大ク作間敷儀、井二般三艇一ツにもやひ巾問敷儀、和撲耀笠之部二有之、
一正徳二二巳年三且、二挺立三挺立之船倅止、屋形舟先年より百艇こ限り候問、彌共旨を守候様二可巾付儀二什御害付、靴之部二右之、
一何年同月、二挺立三挺立舟ヶ速解船二可仕、屋形舟焼印可申付儀に付町燭、雑之部二有之、
一享保五子年三月、品川穆　御成之節、諸廻船のけ置候内を猟船往行仕問敷義、御成之部二右之、
一享保六丑年九月、品川沖二て諸廻船之水主と馴合、出宜出買致凹敷儀、諸商崗之部二有之、
一享保十三巾年正丹、旧光　御祇參御溜守中舟改之儀二付御嶼書、御祇參之部こ右之、

廻船井川船等之部「薇誹塔土史」

1炭年五丹より

呂四００　堯永十五旗年五且

一五百石以上之船倅止と此以血被　仰出候、今以共通候、然共商買船は御ゆるし被成候、共段心徂可巾事、
一此以前御法度書二何遍之事右之とゆふとも、共處をまもれと被　仰出之段は、仞所におゐて面々私之事喉、公俵を逆覗仕候もの盗賊等之甑なとは各別候、以來其旨を心徂巾へし、若國法をそむくもの右之り、隣國之面々ヶ速巾付へし、二小身の衆は共品により、近所之衆合僥いたし、租計中へき堊、

図1　OCR読み取りテキスト
本図では、原テキストに近いサイズに文字サイズを変更している。
OCRが出力するテキストには大きさを変えるコードは挿入されていない。

御触書寛保集成四十二

一廻船并川船等之部
一御堀并所々橋塵芥捨場等之部
廻船并川船等之部可有之御触書、ケ條入交り、外之部ニ有之候分
一明暦元未年十一月、廻船船道を明ケ、舟つかへ候ハぬ様ニ可掛置儀、御堀并所々橋等之部ニ有之、
一天和元酉年十月、廻船之塵芥川え不捨、傳馬船ニて永代嶋御定之埃捨場え遣可申儀、御堀并所々橋等之部に有之、
一元禄三午年七月、尾形船御定之寸尺より大ク作間敷儀、并二艘三艘一ツにもやひ申間敷儀、相撲躍等之部ニ有之、
一正徳三巳年三月、二挺立三挺立之船停止、屋形舟先年より百艘ニ限り候間、彌其旨を守候様ニ可申付儀ニ付御書付、雑之部ニ有之、
一同年同月、二挺立三挺立舟早速解船ニ可仕、屋形舟焼印可申付儀に付町触、雑之部ニ有之、
一享保五子年三月、品川辺　御成之節、諸廻船のけ置候内を猟船往行仕間敷義、御成之部ニ有之、
一享保六丑年九月、品川沖ニて諸廻船之水主と馴合、出売出買致間敷儀、諸商売之部ニ有之、
一享保十三申年正月、日光　御社參御留守中舟改之儀ニ付御触書、御社參之部ニ有之、
廻船并川船等之部「寛永十五寅年五月より
寛保二戌年十一月迄」
二四〇〇　寛永十五寅年五月
一五百石以上之船停止と此以前被　仰出候、今以共通候、然共商買船は御ゆるし被成候、其段心得可申事、
一此以前御法度書ニ何遍之事有之とゆふとも、其處をまもれと被　仰出之段は、国所におゐて面々私之事候、
公儀を違背仕候もの盗賊等之事なとは各別候、以來其旨を心得申へし、若國法をそむくもの有之ハ、隣國之面々早速申付へし、但小身の衆は其品により、近所之衆合属いたし、相計申へき事、

図2　読み誤りを訂正したテキスト

1．2 基本方針

上記自動訂正に必要なデータの採取方法について述べる。データとして必要なものは、再現性の高い誤認識の正誤の文字対である。これを経験的に使用者が集めることも可能であるが、それでは使い勝手が悪い。つまり、ワープロで辞書登録をするような程度の便利さが、少なくとも必要であろう。後編集の過程の中でエディタ・プログラムに自動的に記録をさせるというような機械化は可能であるが、そのようなエディタ・プログラムは市販の製品にはない。開発するとしても、エディタ・プログラム自体の開発費用の方が高くつくであろうから、ここでは、独立した専用の収集プログラムを開発することにした。

まず、最初はいく分かのOCR入力テキストを人手で編集した正しいテキストに直す作業を行なう。この作業自身は、市販のワープロ（プログラム）など、使用者の慣れたシステムを使う。次に、今回開発の差分プログラムにより、OCRテキストと編集後テキスト（こちらが本来の元テキスト）の文字列単位の差分を取り出す。さらに、その差分を統計的に処理するプログラム（新規開発）によって、頻出する正誤訂正対を文字単位で抽出する。この訂正対を基本に、3番目のプログラム（新規開発）が、もう一度本来のテキストを調べて、自動的に適用することができるような編集コマンド群を作成する。このようにして作成された編集コマンド群は、SEDというUNIXワークステーションやMS-DOSパソコンに既存のプログラムにパラメーターとして渡され、新規のOCR入力のテキストの自動（部分）訂正を可能にする。以上の処理を図3に示す。

この処理は、繰り返しによってさらに効率・精度を上げていけるようになっている。つまり、OCRテキストの自動訂正後に行なわれる確認編集で得られた編集後テキストも使って、訂正の候補を増やすことができる。ただし、精度には上限があり、ほどなく処理時間だけが増えて得られる成果がほとんどないという状況になると予想される。

2．システムの概要

2．1 差分プログラムの概要

パソコン（MS-DOS）やワークステーション（UNIX）の世界では、テキストの差分を取るプログラムが存在するが、これらは行を対象とする。これは、ソースプログラムの訂正を判定したり履歴を保存したりするのに利用することを想定しているためであり、本稿のような文字単位の編集コマンドを抽出するような応用は想定外となっている。

2つの文字列の差分は、一致・不一致を検出することから始まる。2つの文字列の差分検出アルゴリズムとして、Longest Common Subsequence（最長共通部分文字列、以下LCSと略す）検出法の応用が知られている。本プログラムはこのアルゴリズムを前処理と本処理の基幹としている。

LCSは、差分をとるそれぞれの文字列の文字一つ一つを縦と横に並べて、その交点に一定の計算を施す行列である（図4）。それぞれの交点の情報は、先頭からその交点までにいくつ一致しているペアの順序列を見つけることができるのかを示した値である。ある交点（j、k）で縦（j）と横（k）とが一致していれば、一致ペア数は交点（j－1、k－1）に1加えたものである。つまり、縦と横の文字列の最後尾をはがして得られるそれぞれの文

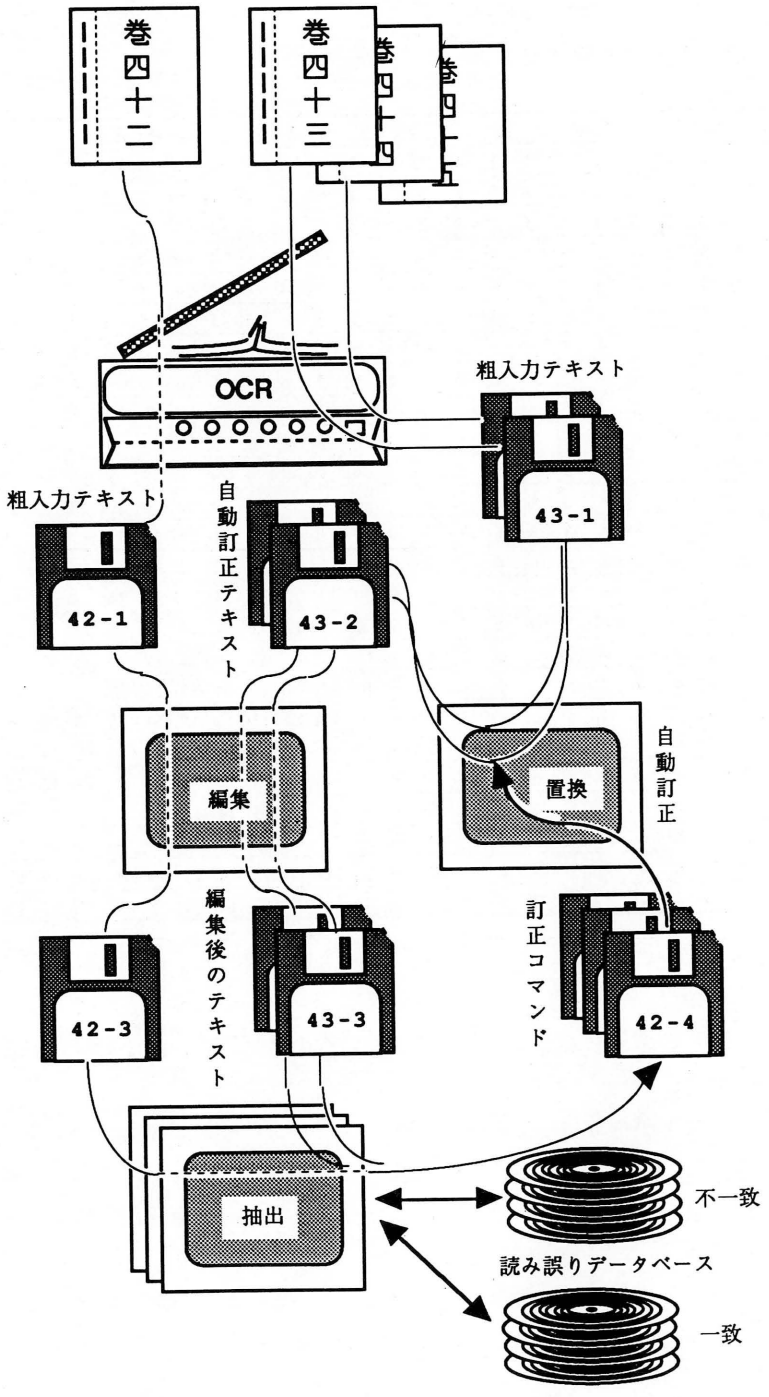


図3　OCR 入力と訂正の処理の流れ

図４　LCS を計算する行列の例

	一	正	徳	二	一	巳	年	三	且	、	二	挺	立	三	挺	立	之	船	停	止	、	屋	形	舟	先	年
一	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
正	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
徳	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
三	1	2	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
巳	1	2	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
年	1	2	3	3	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
三	1	2	3	3	3	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
月	1	2	3	3	3	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
、	1	2	3	3	3	4	5	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
二	1	2	3	4	4	4	5	6	6	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
挺	1	2	3	4	4	4	5	6	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
立	1	2	3	4	4	4	5	6	6	7	8	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10
三	1	2	3	4	4	4	5	6	6	7	8	9	10	11	11	11	11	11	11	11	11	11	11	11	11	11
挺	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	12	12	12	12	12	12	12	12	12	12	12
立	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	13	13	13	13	13	13	13	13	13	13
之	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	14	14	14	14	14	14	14	14	14
船	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	15	15	15	15	15	15	15
停	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	15	15	15	15	15	15	15
止	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	16	16	16	16	16	16
、	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	17	17	17	17	17
屋	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	18	18	18	18	18
形	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	19	19	19
舟	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	20	20	20
先	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	20	21	21
年	1	2	3	4	4	4	5	6	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	20	21	22

字列の最長共通部分に、最後のペアをくっつけたものになるということをボトムアップに計算している。その交点で一致していなければ、交点（ｊ－１、ｋ）と交点（ｊ、ｋ－１）のうちの大きいほうの値になる。このようにして、求めた行列の、右下の交点の値が共通部分列の最大長になる。

この行列を、右下からたどってみると、同じ数字のブロックが左上方向に角を出していることが分かる。この角のところが、真に一致している場所である。図４に影を着けた部分が縦横で文字が一致している所であるが、角になっていないペアを採用すると、最長の部分文字列は得られない。

このような角が見つかれば、角から角までは一致している文字がない。従って、その間を差分とすれば、目的の文字単位の差分の種がえられる。角をたどるアルゴリズムなどの詳細は割愛する。

図１に二重末梢線で示したように、OCR 入力には主に傍注[3]が原因で行単位の認識不能が存在する。このため、単純に行単位での差分を順々に取るわけにはいかない。まず、複数の行を対象に行の対応を発見してから、次に差分をとる必要がある。本差分プログラムでは、２行以上のずれがないことを仮定して行の対応を見つけるアルゴリズムを採用している。こ

表1　差分プログラムの能力

一致	不一致		脱落		延べパターン		異なりパターン		
文字数	編集前	編集後	編集前	編集後	編集前	編集後	検出	有効	有効率
15536	3674	3499	1342	450	3216	3086	685	161	0.24
12390	2401	2514	959	250	2082	2018	321	63	0.20
15161	2446	2366	1789	246	2101	2024	203	39	0.19
11328	2240	1963	2428	231	1789	1690	128	23	0.18
10043	1774	1427	1184	125	1377	1286	87	20	0.23
14350	2333	2059	997	161	1964	1879	140	29	0.21
11896	2098	1949	1758	378	1768	1702	140	25	0.18
19917	3494	3408	1031	847	2973	2841	180	27	0.15
12587	1538	1491	983	108	1403	1337	87	22	0.25

表2　自動訂正の有無による編集文字数の差

巻	前処理	後編集	合計	直接編集	差	脱落	省力率1	省力率2	冗長度
42	—	—	—	—	—	—	—	—	—
43	1351	1615	2966	2401	565	959	0.33	0.23	1.24
44	1809	1290	3099	2446	653	1789	0.47	0.27	1.27
45	1691	1267	2958	2240	718	2428	0.43	0.21	1.32
46	1343	907	2250	1774	476	1184	0.49	0.29	1.27
47	1736	1174	2910	2333	577	997	0.50	0.35	1.25
48	1708	1064	2772	2098	674	1758	0.49	0.27	1.32
49	2856	1625	4481	3494	987	1031	0.53	0.41	1.28
50	1289	713	2002	1538	464	983	0.54	0.33	1.30

前処理　自動訂正で置換される文字数
後編集　後編集で置換する文字数
合計　前処理＋後編集
直接編集　自動訂正なしの編集で置換する文字数
差　合計－直接編集
脱落　対応不詳の文字数

省力率1　(前処理－差)／直接編集
　文字置換編集のうち前処理が有効な割合
省力率1　(前処理－差)／(直接編集＋脱落)
　全編集のうち前処理が有効な割合
冗長度　合計／直接編集
　前処理のために増えた編集の手間

図5　行の一致の検出

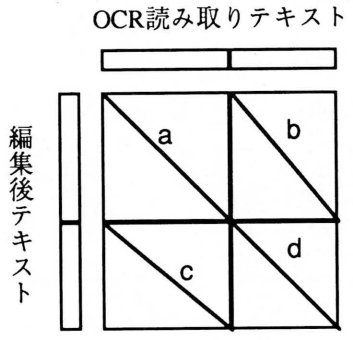
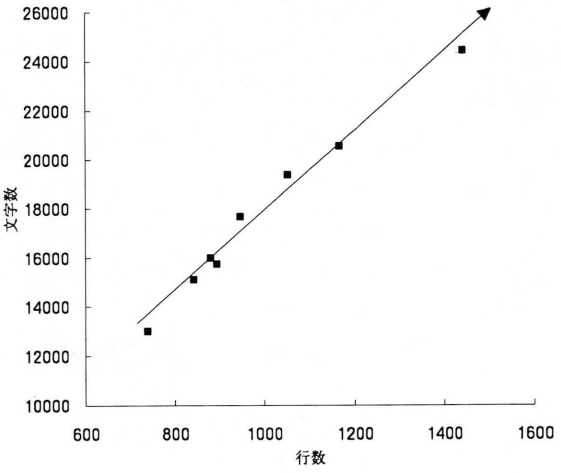


図6　御触書テキストの行数と文字数の相関



れを有限行に拡大することは、単純なアルゴリズムの拡張で行なえる。アルゴリズムは、2行分の文字列のLCS行列の最長一致面を見つけることであるが、図5をみれば明らかなように、4本の対角線a, b, c, d上の一致数をLCS値から求めて、最大のものを選べばよい。aが選ばれることはお互いの頭部行が対応することを意味し、bが選ばれることは編集前の尾部行と編集後の頭部行が対応すること（即ち編集前の頭部行が削除されたこと）を意味し、cが選ばれることは編集前の頭部行と編集後の尾部行が対応すること（即ち編集後の頭部行が挿入されたこと）を、それぞれ意味する。dが選ばれるのは、お互いの頭部行が一致しないことを意味するだけでなく、尾部行で次の一致があることをある程度保障する。つまり、何らかの原因で2行以上の差ができてしまった場合に、次々と互いの頭部行を削除しあって、永遠に差が縮まらないという失敗をおかすこともありえるのであるが、その可能性を少なくするために有用である。具体的には、どれも一致すると言えるほどの一致がない場合に、その2行を続けて1行とみなし、新たな尾部行を読み込んで照合を続けることにするのである。

差分プログラムは、結局、行の対応が見つけられなかった部分を捨てることになる。これは、今回の目的のための措置であり、一般には、これらも差分として検出して、後の処理に利用することも可能である。

本差分プログラムの性能を概算すると、表1のようになる。本差分プログラムは編集前のテキストの約71%から83%、編集後では80%から89%の文字の一致対応を発見できる。また、逆に、その一致文字列の間にある不一致を検出することになる。行の対応を見つけられなかったのは、編集後の総文字数の1%から4%にあたる。

2．2　差分から置換パターンの作成

前節で示したように、差分は一致しなかった部分文字列のペアであり、そのまま置換パターンとすることはできない。これを文字の読み取り誤りの訂正用として文字単位の置換パターンとするのが次の置換パターン作成プログラム群である。

第1のパスは、基本的な読み取り誤りのデータベース作成である。まず、差分が両者1文字である場合は読み取り誤りであると判断して、ほぼ間違いない。さらに、2文字から1文字あるいは1文字から2文字の差分も、かなり高い確立で読み取り誤りである。（たとえば、図1にある「元」を2文字に読み誤る例はこれで検出される。）

第2のパスで、長い差分の文字列を分解する。前や後から1ないし2文字を切り取って、第1のパスで作成したデータベースで登録済みの読み誤りペアかどうか確認する。そうなら、出現回数を更新した後それを剝がす。これをできるかぎり繰り返し、残りが登録の条件（長さ1ないしどちらか一方が2）を満たせば、あらたなペアとみなしてデータベースに追加する。条件を満たさなかった差分の部分列は、ファイルに書き出す。

第3のパスは、第2のパスで残された差分に対して、読み誤りペアが検出されなくなるまで、第2のパスを繰り返すことである。これは、データベースへの登録の順序の前後が不定であること、前後からしかデータベースと照合しないことによる。照合を1度に行なうには複雑なアルゴリズムを必要とする。繰り返しは、せいぜい3回目で収束するので、プログラムを複雑にしないほうが計算も楽である。

第４のパスでは、差分検出時に一致した文字の字数を数えて、データベースにする。正しく認識されたと考えられる文字を、置換の対象としないためである。

第５のパスで、読み誤りペアを整列して、同一字の読み誤りが複数の場合に分かれるものを検出する。第４のパスで検証した正しく認識される場合も読み誤りが複数である場合に含む。この情報を勘案して、有効な置換パターンを作りだす。有効である条件は、(1)同一の読み誤りが３件以上あること（偶然の誤りではないこと)、(2)読み誤りの元（OCR）文字が共通のもののうち発生件数が最大であること。(3)読み誤りの元文字について、同文字が読み誤られなかった件数をオーバーしていること、の３件である。この条件を満たす読み誤りペアを置換パターンの形式でファイルに出力する。検出できた読み誤りペアの延べ件数および異なり数と有効ペアの異なり数を、表１に示す。異なり数にして約20％しか有効なペアはないことがわかった。

この後は、１．２節（図３）に示すように、置換パターンをSEDコマンドのパラメータとするようなコマンドをOCR読み取りデータにかければよい。(4)

３．実際の適用諸データ

本プログラムによって作られるデータベースをサンプリングすることなどで、実際の適用の様子をある程度調べることができる（表１・表２）。御触書寛保集成巻四十二から巻五十までのテキストで、本システムを順に適用する様子をシミュレートする。各巻の諸データは表１の通り。また、OCR読み込みの巻ごとの行数と文字数の相関は図６のようになっていて、大きな過疎の差異はない。

巻四十二を人の手で編集する。その結果、行数にして20行、文字数にして約1100文字の減少があった。差分プログラムが対応を発見したのは、文字数にして正対応が約15000字、誤対応が約3700字（OCR側）と約3500字（編集後側）であった。450字（約２パーセント）は、プログラムの判断ミスにより、対応を発見できなかった。

差分検出プログラムによって抽出された差分のかたまりは2767個である。これを置換パターンに直すと、総数3082件、異なりパターンが685であった。このうち、有効な置換パターンは、異なりパターンが161に減少する。

巻四十三の編集には、OCRの読み取りテキストに対して巻四十二で作った置換パターンがまず適用される。計算上では、約1000件強の文字列の書き換えを起こすが、正しいにもかかわらず変更してしまう約100件強の誤変換があるため、実効は800件弱の分を機械が肩代りすることになる。このあと人間が編集を加えて、目的のテキストを得る。

巻四十三を前処理しないものとしたものとでは、一致文字数が12313対13188と約470文字の差がでる。また、人間の編集分は件数で約1300件対1700件、字数にして約1600字対2600字である。ちなみに、計算上では800件の差があるはずであるが、誤変換が２件以上つながって、１件の編集に計上されているためであると考えられる。字数の差が1000字あることは、そのことを示唆する。全文字数15750字のうち12390字の認識であったものが、14572字のうち13209の認識に変わったのであるから、形式的な認識率は79％から90％に向上したことになる。

次に、巻四十四の自動編集には巻四十三の編集結果をも使って新たな置換コマンドを作成する。これを次々と繰り返す。そのようにした場合の性能を、表2に示す。

4．おわりに

前章で示した90パーセントという認識率は、まだ満足のいく数字ではない。ただ、読み取りのバースト的な誤りを除外すれば、目標の90パーセント台後半に到達する。つまり、連続して読み誤りが多発している部分は、正読み取り率が低くても編集するという行為に対する心理的負担に大きな差はない。読み取り誤りが分散しているような部分でこそ、誤り率が低いほど心理的負担を和らげることができる。

今後は、英文OCRが取りいれているようなスペルチェックとの連動を導入することが必要であるが、肝心の日本語スペルチェックはまだ実用になっているものがない。将来の課題であろう。その予備的な段階として、冗長な結果となっている自動訂正を正確なものとするために、2文字連を2文字連に訂正するように本方法を拡張することを構想している。これは、一種の文脈を勘案する自動訂正と言え、より正確な訂正が可能になるであろう。続報としたい。

本研究は総合研究大学院大学共同研究『日本語テキストデータベースの利用法に関する研究』（研究代表者　村井康彦　国際日本文化研究センター教授）の研究の一部として、筆者が分担して行なった。また、利用したテキストは、平成3年度文部省科学研究費データベース作成費『江戸幕府御触書データベース』（代表者　速水融　国際日本文化研究センター教授）によるものを利用し、平成4年度文部省科学研究費データベース作成費『江戸幕府法令データベース』（代表者　笠谷和比古　国際日本文化研究センター助教授）のデータ入力に一部応用した。

注

(1) 高柳真三、石井良介編『御触書寛保集成』岩波書店、1934年。

(2) ハードウエア：富士電気株式会社製　日本語ドキュメントプロセッサXP-70。ソフトウエア：同社製　XPal。

(3) 今回の入力では傍注を除外しているため、これらは単純な行抹消作業を行なうだけでよい。

(4) SEDのパターン数に制限があるため、パターンが多くなってくると、SEDコマンドを複数回適用する必要が生じることがある。