自然言語処理 107－6
（1995．5．26）

手書き文字列読み取りのための単語列探索アルゴリズム —文字タグ法—

福島 俊一† 下村 秀樹† 森 義和‡

NEC 情報メディア研究所† NEC 情報システムズ‡

本論文では、フリーピッチ手書き文字列の読み取りのために、新しい知識処理アルゴリズムである「文字タグ法」を提案する。字形が多様で、文字サイズ・文字ピッチにばらつきがあり、文字の接触・入り組みなどもよく起きる手書き文字列の読み取りでは、誤切り出しや誤認識によって欠落した正解文字を補完する知識処理が不可欠である。従来の知識処理方式は、単語辞書と候補文字列とを照合して単語候補を抽出した上で、その並びの妥当性を判定する2段構成である。このような従来法では、単語境界が不確定なケースをうまく扱えないことや、候補文字列と単語辞書との虫食い照合における組み合わせ爆発を避けると、強引に候補を切り捨てることになって最良解を保証できないことなどが大きな問題になっている。これに対して、本論文で提案する文字タグ法は、文字を基本単位としてタグを付与し、その位置関係をチェックしながら連結していく戦略をとる。単語内の文字の連結と単語間の文字の連結とを同等に扱って動的計画法を適用することで、最良解を保証し、かつ、入力文字列の長さ L と候補多重度 M に対して $O(L^2 \cdot M^2)$ または $O(L \cdot M^2)$ の時間計算量を達成している。さらに、手書き宛名住所の地名領域の読み取りに文字タグ法を応用し、文字切り出しや個別文字認識のあらゆる組み合わせと正解文字欠落の可能性のなかから最良解を高速に探索する文字タグ法の能力を確認した。

A Word-Sequence Search Algorithm for a Hand-Written Character Reader

Toshikazu FUKUSHIMA† Hideki SHIMOMURA† Yoshikazu MORI‡

Information Technology Research Laboratories, NEC Corporation†

NEC Informatic Systems Ltd.‡

This paper proposes a new algorithm for post-processing in a hand-written character reader. Hand-written characters have such characteristics as various styles, irregularity in size and pitch, frequency of character overlapping, and so on. These characteristics bring difficulty into hand-written character reading systems. Post-processing to correct mis-segmentation and mis-recognition by linguistic information is an important approach to accurate reading. Conventional post-processing methods have two steps. In the first step, word candidates are extracted by word dictionary looking-up. In the second step, combinations of words are evaluated. These conventional methods have the following problems. The first problem is that they don't work well when word boundary segmentation is missed. The second one is combinational time complexity, required for examinations of all combinations of character segmentation candidates and character recognition candidates by approximate matching. In the algorithm proposed in this paper, character candidates are tagged with position-in-word information, and the position-in-word tags are connected by a dynamic programming method. This algorithm has the advantage of time complexity $O(L^2 \cdot M^2)$ or $O(L \cdot M^2)$ for optimum path search, where L is input length, and M is average number of segmentation and recognition candidates per character. This paper also describes its implementation and evaluation results in Japanese address reading.

1 はじめに

コンピュータで手書き文字列を読み取る技術は、郵便番号による郵便物の自動区分や帳票類のデータ入力などで早くから実用化されている。しかし、これらの例は、読み取り対象文字種を数字や英字・片仮名に限定したり、文字枠で書き込み位置を制限するなど[1][2]、利用者側に制約を課すことで手書き文字列読み取りが本来もつ困難さを回避したものである。

手書き文字列は、人間が情報を発信・伝達・蓄積するために、自然で欠くことのできないメディアである。それゆえに、手書き文字列のもつ自由度と個性を失わせてしまうような制約を与えることなく、高精度な読み取りを実現する技術が強く望まれている。

自由に書かれた手書き文字列は、字形が多様で、文字サイズ・文字ピッチにばらつきがあり、文字の接触・入り組みなどもよく起きる。そのため、読み取りの際には、1文字に対応するセグメントが正確に切り出せないケースや、各セグメントの個別文字認識結果のなかに正解文字が入らないケースなどが頻繁に発生してしまう。したがって、高精度な読み取りを実現するためには、誤切り出しや誤認識によって欠落した正解文字を補完する知識処理(文脈処理や後処理とも呼ばれる)が不可欠である[3][4][5][6][7]。

現在主流となっている知識処理の枠組みは、まず、各文字位置(セグメント)に複数通りの可能性(候補文字)を許した認識結果文字列と単語辞書とを照合し、次に、単語の並びとしての妥当性を判定して読み取り結果を決定する2段構成である。正解文字の欠落には、1段目の単語照合において虫食い照合(部分的な不一致を許した照合)を行なうことで対処する。

このような2段構成の知識処理は、活字文字列や文字枠内にていねいに書かれた文字列の読み取りで実績を上げている[2][8][9][10][11][12][13]。しかし、フリーピッチ手書き文字列の読み取りを正確かつ効率よく行なうのには、まだ十分なものではない。以下のような点に課題がある。

第一に、例えば「川崎市宮前区」の「市宮」が接触して1セグメントとされてしまったときなど、単語の境界位置を確定できないようなケースがうまく扱えない。

第二に、2段目の単語列探索で最良解が保証されるように、1段目の虫食い照合の際に正解文字欠落のあらゆる可能性を求めておこうとすると、最悪の場合、単語辞書の全探索や候補文字の組み合わせ爆発が起きる。そこで、現実的には、虫食い照合に1文字不一致のみのような制限を加えたり、文字切り出しは正しいと仮定して知識処理を実行し、それが失敗したときに限って別な切り出し候補も調べるなど、強引に可能性を切り捨てている。

第三に、第二の問題の結果として、読み取り候補を絞り込むための基準が、1段目の切り捨て基準と2段目の妥当性判定基準の2通りに分かれてしまい、バランスのよい基準設定や統一的な調整が難しくなっている。

本論文では、上述の問題を解決するため、従来の2段構成とはまったく異なる知識処理の枠組みとして、「文字タグ法」と名付けた新しいアルゴリズムを提案する[14]。従来法が単語の単位を強く意識しているのに対して、文字タグ法は、文字を基本単位とし、その位置関係をチェックしながら、動的計画法によるコスト計算を行なう。それによって、上記第二の問題に挙げた組み合わせ爆発を回避し、最良解を保証した効率のよい探索が実現できている。また、コスト計算では単語内の文字の連結と単語間の文字の連結とを同等に扱うことで、上記第三の問題も解決できている。さらに、途中を読み飛ばした文字の連結を許しているので、上記第一点として指摘した単語境界の不確定ケースも自然に扱える。

以下、第2章では、文字タグ法のアルゴリズムを記述する。本論文では、フリーピッチ手書き文字列の高精度な読み取りが要求される応用として、特に、手書き宛名住所の地名領域(都道府県名・市区郡名・町名の並び)の読み取りに文字タグ法を適用した。第3章では、その性能評価結果を報告し、文字タグ法の有効性を検証する。

2 文字タグ法

2.1 知識データの構造

文字タグ法で用いる知識は、読み取り対象文字列に出現し得る単語のリストと、それらの単語間の接続可否に関する情報である。このような知識を次のような単語テーブルで記述する。

単語テーブル: $wcode_i$: $[wstr_i, \{pwrd_{i,j}\}]$
$wcode_i$ は単語を表すコード
$wstr_i$ は単語 $wcode_i$ の表記文字列
$pwrd_{i,j}$ は単語 $wcode_i$ に前接可能な単語のコード

図1は単語テーブルの例である。図1は、読み取り対象文字列に「川崎市」「宮前区」「梶ケ谷」「宮崎」…「有馬」という単語が出現し得て、それらの間の接続関係については、「宮前区」には「川崎市」(単語コード:1)、「梶ケ谷」「宮崎」…「有馬」には「宮前区」(単語コード:2)が前接可能であることを表わしている。

さらに、文字タグ法では、文字を基本単位として知識を参照するため、次のような文字インデックスを用意する。この文字インデックスは、単語テーブルから機械的な変換によって作成できる。

文字インデックス: chr_i: … $[wrd_{i,j}, pos_{i,j}, len_{i,j}]$ …
chr_i は文字
$wrd_{i,j}$ は文字 chr_i を含む単語の1つ(単語コード)

1: ["川崎市", { }]
2: ["宮前区", {1}]
3: ["梶ケ谷", {2}]
4: ["宮崎", {2}]
5: ["宮前平", {2}]
6: ["馬絹", {2}]
7: ["野川", {2}]
8: ["有馬", {2}]

図 1: 単語テーブル

ケ:	[3,2,3]	川:	[1,1,3] [7,2,2]
梶:	[3,1,3]	前:	[2,2,3] [5,2,3]
宮:	[2,1,3] [4,1,2] [5,1,3]	谷:	[3,3,3]
区:	[2,3,3]	馬:	[6,1,2] [8,2,2]
絹:	[6,2,2]	平:	[5,3,3]
崎:	[1,2,3] [4,2,2]	野:	[7,1,2]
市:	[1,3,3]	有:	[8,1,2]

図 2: 文字インデックス

$pos_{i,j}$ は単語 $wrd_{i,j}$ における文字 chr_i の位置 (単語内位置)
$len_{i,j}$ は単語 $wrd_{i,j}$ の長さ

図 2は、図 1の単語テーブルから作成した文字インデックスの例である。図 2によれば、例えば「崎」という文字は「川崎市」(単語コード:1) の 3 文字中の 2 文字目、あるいは「宮崎」(単語コード:4) の 2 文字中の 2 文字目になり得ることがわかる。

2.2 入力データと文字タグの形式

文字タグ法では、文字切り出しと個別文字認識の各々に複数通りの可能性 (候補) を許した入力データ (図 4の (c) 参照) を、次のような形式で記述する。

入力データ: $seg(i)$: $[top_i, wid_i, \{rchr_{i,j}\}, \{rcst_{i,j}\}]$
i はセグメント番号
top_i と wid_i は $seg(i)$ の先頭位置と幅
$rchr_{i,j}$ は $seg(i)$ に対する第 j 位の候補文字
$rcst_{i,j}$ は $rchr_{i,j}$ の個別文字認識コスト

図 3は、図 4の (c) 入力データを上記の形式で記述したものである。ここでは、セグメント $seg(i)$ の先頭位置 top_i と幅 wid_i を、最も細かいセグメント境界位置 (すべてのセグメントの境界位置の和集合に先頭側から連番を付けたもの、図 4の (b) 参照) を単位として数えている。

次に、文字タグ法の処理過程で作成する「文字タグ」と呼ぶデータの形式を、次のように定義する。

文字タグ: $tag(n)$: $[ttop_n, twid_n, twrd_n, tpos_n, tlen_n, tpre_n, tcst_n, tflg_n]$

seg(1): [1, 1, { 1 }, {0}]
seg(2): [1, 3, { 川, 小 }, {0,16}]
seg(3): [2, 1, { 1 }, {0}]
seg(4): [3, 1, { 1 }, {0}]
seg(5): [3, 2, { 仙 }, {0}]
seg(6): [4, 1, { 山, 小 }, {0,32}]
seg(7): [4, 2, { 崎 }, {0}]
seg(8): [5, 1, { 奇 }, {0}]
.....
seg(23): [16, 3, { 川 }, {0}]
seg(24): [17, 1, { − }, {0}]
seg(25): [18, 1, { 1 }, {0}]

図 3: 入力データの形式

n は文字タグ番号
$ttop_n$ と $twid_n$ は $tag(n)$ に対応するセグメントの先頭位置と幅
$twrd_n$ と $tpos_n$ と $tlen_n$ は $tag(n)$ に対応する単語のコードと単語内位置と長さ
$tpre_n$ は $tag(n)$ が連結する最良の文字タグの番号
$tcst_n$ は $tag(n)$ までの文字タグ連鎖の累積コスト
$tflg_n$ は $tag(n)$ が文字タグ連鎖の末端フラグ

2.3 文字タグ法のアルゴリズム

文字タグ法では、第 2.2節に示した形式の入力データを対象として、第 2.1節に示した知識データを参照しながら、第 2.2節で定義した文字タグを作成していく。文字タグ法のアルゴリズムは以下の通りである。なお、この記述では、値が小さいほどコストが良いものとしている。

Step1: 先頭側のセグメントから順にすべてのセグメント $seg(i)$: $[top_i, wid_i, \{rchr_{i,j}\}, \{rcst_{i,j}\}]$ について Step1.1 ～ Step1.2 を実行する;

Step1.1: $seg(i)$ のすべての候補文字 $rchr_{i,j}$ について文字インデックスを検索する;

Step1.2: 文字インデックスから得られたすべての 3 項情報 $[wrd, pos, len]$ に対して各々 $tag(n)$: $[ttop_n = top_i, twid_n = wid_i, twrd_n = wrd, tpos_n = pos, tlen_n = len, tpre_n = NULL, tcst_n =$ 初期コスト $icst(rcst_{i,j}), tflg_n = 1]$ を作成する;

Step2: 先頭側の文字タグから順にすべての文字タグ $tag(n)$: $[ttop_n, twid_n, twrd_n, tpos_n, tlen_n, tpre_n, tcst_n, tflg_n]$ について Step2.1 ～ Step2.5 を実行する;

Step2.1: $ttop_m + twid_m \leqq ttop_n$ かつ $twrd_m = twrd_n$ かつ $tpos_m < tpos_n$ が成立するすべ

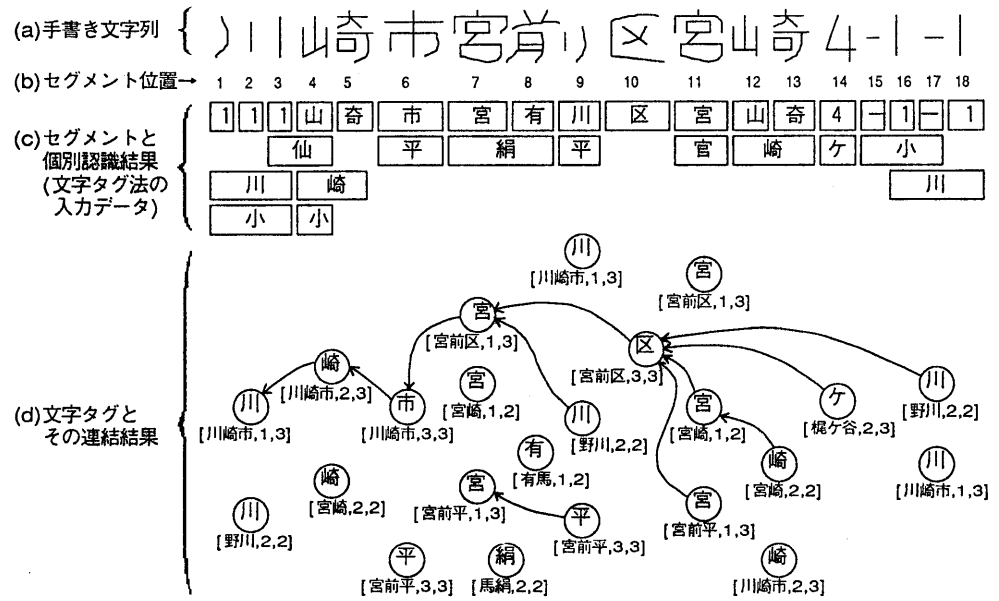

図 4: 文字タグ法による処理例

ての $tag(m)$: $[ttop_m, twid_m, twrd_m, tpos_m, tlen_m, tpre_m, tcst_m, tflg_m]$ を検索して, $tag(m)$ に $tag(n)$ を連結したときのコスト $ccst(m,n)$ を計算する;

Step2.2: 単語テーブルを検索して $twrd_n$ に前接可能な単語の集合 $\{ pwrd_k \}$ を求める;

Step2.3: $ttop_m + twid_m \leqq ttop_n$ かつ $twrd_m = pwrd_k$ が成立するすべての $tag(m)$ を検索して, $tag(m)$ に $tag(n)$ を連結したときのコスト $ccst(m,n)$ を計算する;

Step2.4: Step2.1 と Step2.3 で検索された $tag(m)$ のなかで連結時コスト $ccst(m,n)$ が最小となる $tag(m)$ を求める;

Step2.5: Step2.4 で求めた $tag(m)$ に対して $ccst(m,n) - tcst_m \leqq$ 閾値 CST_{thresh} が成立するならば, $tag(n)$ について $tpre_n = m$, $tcst_n = ccst(m,n)$, $tflg_m = 0$ とする;
それ以外の場合 (連結可能な $tag(m)$ が存在しなかった場合も含む) は, $tag(n)$ について $tpos_n \neq 1$ ならば $tcst_n$ に先頭補正コスト $scst(n)$ を加える;

Step3: $tflg_n = 1$ かつ $tpos_n \neq tlen_n$ であるようなすべての $tag(n)$ について, $tcst_n$ に末端補正コスト $ecst(n)$ を加える;

Step4 $tflg_n = 1$ の $tag(n)$ のなかで $tcst_n$ が最良のものから $tpre_n$ をたどって得られる文字タグ連鎖から取り出した単語の並びを読み取り結果とする;

図 4に文字タグ法による処理例を示す。

Step1 では、全セグメントの全候補文字 (図 4では (c) の各□がそれに対応する) に対して各々、文字インデックスを検索して、順次、文字タグを作成していく。Step2 では、Step1 で作成された各文字タグについて、先頭側から順次、$tpre_n$ (最良の連結先) を決定していく。この $tpre_n$ の決定には、第 2.4節で述べるようなコスト定義にもとづいて動的計画法を適用する。この Step2 の結果、図 4では (d) の○の間をつなぐリンクができる。

Step2.5 では、閾値 CST_{thresh} によって文字タグの連結をカットしている。Step2.1 や Step2.3 における位置関係の条件は、2 つの文字タグの位置がどんなに離れていても関係なく成立し得る。Step2.5 のカット条件は、遠く離れた箇所に偶然に現われた候補文字と強引に連結させてしまうことを避ける役割をもつ。

図 4で末端になっている文字タグのうち (位置 12, 長さ 2) の [宮崎,2,2] のコストが最良だったとすると、Step4 では、それから前にたどって得られる文字タグ連鎖 [宮崎,2,2] − [宮崎,1,2] − [宮前区,3,3] − [宮前区,1,3] − [川崎市,3,3] − [川崎市,2,3] − [川崎市,1,3] を取り出し、「川崎市宮前区宮崎」を読み取り結果とする。

2.4 コストの定義

まず、コスト定義の準備として、連結対象の2つの文字タグ(前側の文字タグ $tag(m)$ と後側の文字タグ $tag(n)$)の間の論理的間隔と物理的間隔を定義しておく。

論理的間隔 $ldis(m,n)$: $tag(m)$ と $tag(n)$ の間の読み飛ばした文字数。すなわち、$twrd_m = twrd_n$ のとき $ldis(m,n) = tpos_n - tpos_m - 1$、$twrd_m \neq twrd_n$ のとき $ldis(m,n) = tlen_m - tpos_m + tpos_m - 1$。

物理的間隔 $pdis(m,n)$: $tag(m)$ と $tag(n)$ の間の領域(セグメント位置 $ttop_m + twid_m$ から $ttop_n - 1$ までの領域)のサイズ。

図4の例において、(7,1) の [宮前区,1,3] と (10,1) の [宮前区,3,3] との間は、[宮前区,2,3] に相当する1文字分が読み飛ばされているので論理的間隔は1である。物理的間隔は (7,1) と (10,1) の間にある (8,2) に相当する領域サイズを指す。

ここで、第2.3節のアルゴリズムで導入した各種コストを定義する。

初期コスト $icst(rcst)$: 文字タグが1つ作成された(1文字読めた)こと自体にもたせるコスト。文字タグに対応する候補文字の個別文字認識コスト $rcst$ を含む。$icst(rcst) = rcst + CST_{init}$ (読めた文字が多い方が確からしいものと考えて CST_{init} はある程度大きい負の定数とする)がその一例である。

連結時コスト $ccst(m,n)$: 2つの文字タグ(前側の文字タグ $tag(m)$ と後側の文字タグ $tag(n)$)を連結させたときに、その結果として $tag(n)$ が新たにもつことになる $tcst_n$ の値。前側の $tag(m)$ のもつ $tcst_m$ ($tag(m)$ までの文字タグ連鎖の累積コスト)に、後側の $tag(n)$ の $tcst_n$ (この時点では単純に初期コストが代入されただけのもの)を加算し、さらに、論理的間隔 $ldis(m,n)$ または物理的間隔 $pdis(m,n)$ に応じて大きくなるコスト(間隔コスト:$dcst(m,n)$ とする)や、論理的間隔と物理的間隔の比が不自然だと大きくなるコスト(間隔不整合コスト:$bcst(m,n)$ とする)などを加える。すなわち $ccst(m,n) = tcst_m + tcst_n + dcst(m,n) + bcst(m,n)$ である。

先頭補正コスト $scst(n)$: 単語の先頭側が読めなかった分について補正するためのコスト。$tag(n)$ に対応する単語 $twrd_n$ の単語内位置 $tpos_n$ より前方の文字数 ($= tpos_n - 1$) に応じてコストを大きくする。

末端補正コスト $ecst(n)$: 単語の末尾側が読めなかった分について補正するためのコスト。$tag(n)$ に対応する単語 $twrd_n$ の単語内位置 $tpos_n$ より後方の文字数 ($= tlen_n - tpos_n$) に応じてコストを大きくする。

以上のようなコスト定義の結果、文字タグ法では、文字を基本単位として連結していく際に、単語内の文字の連結と単語間の文字の連結とを同等に扱えている。

3 評価

3.1 計算量

L を入力データの文字列長、M を入力データの候補多重度としたとき、第2.3節に記述したアルゴリズムの計算量は、次のように表わすことができる。ここで、候補多重度 M とは、全セグメントの候補文字数の合計を文字列長 L で割ったもので、各正解文字が平均何個の候補文字をもつかを表わす(図3・図4でいえば $L = 18$ で $M = 31/18 = 1.7$)。

時間計算量: 入力データに対して $O(L^2 \cdot M^2)$。
通常のコスト定義では $O(L \cdot M^2)$。

メモリ使用量: 入力データに対して $O(L \cdot M)$。

まず、時間計算量について説明する。

Step1は、全セグメントの全候補文字 ($L \cdot M$ 個) に対して文字タグの作成処理を実行するので $O(L \cdot M)$ である。Step1で作成された文字タグの個数は $O(L \cdot M)$ と考えてよいから、Step2.1とStep2.3は、その時点で着目している文字タグより前方のすべての文字タグをスキャンすれば $O(L \cdot M)$ である。Step2全体では、これを文字タグの個数 $O(L \cdot M)$ の分だけ繰り返すので $O(L^2 \cdot M^2)$ となる。Step3もStep4も、該当する文字タグを探すために、基本的にはすべての文字タグをスキャンするので $O(L \cdot M)$ である。以上のことから、本アルゴリズムの時間計算量は $O(L^2 \cdot M^2)$ となる。

ただし、Step2.5には文字タグの連結カット条件がある。これを考慮すると、間隔不整合コスト $bcst(m,n) \geqq 0$ で、間隔コスト $dcst(m,n)$ を物理的間隔 $pdis(m,n)$ に対して単調増加になるように定義すれば、ある定数 $PDIS_{thresh}$ に対して $pdis(m,n) > PDIS_{thresh}$ ならば必ずカット条件を満たすようになる。このとき、Step2.1やStep2.3では、物理的間隔が $PDIS_{thresh}$ 以内の文字タグだけをスキャン対象とすればよいことになるので、スキャン対象の文字タグ数は $O(M)$ である。したがって、アルゴリズム全体の時間計算量は $O(L \cdot M^2)$ に抑えられる。このようなコスト定義に無理はなく、文字タグ法の自然な実装では、通常、時間計算量は $O(L \cdot M^2)$ になると考えてよい。

一方、メモリ使用量は、文字タグの数に依存するので、入力データに対して $O(L \cdot M)$ である。

3.2 性能評価

フリービッチ手書き文字列の高精度な読み取りが要求される代表的な応用として、特に、手書き宛名住所の地名領域(都道府県名・市区郡名・町名の並び)の読み取りに文字タグ法を適用した。

3.2.1 評価対象データ

実験には、葉書に書かれた縦書きの手書き宛名住所データを用いた。文字タグ法では地名領域を読み取るが、今回の実験の入力データは地名領域だけでなく、番地・氏名などの書かれた別領域も含んだものとした[1]。そのため、地名領域の文字数は5～12文字であったが、入力データに対応する領域はその倍程度となった。

読み取り対象の地名としては2つのセットを用意した。セットKは東京都国分寺市の全域、セットFは静岡県富士市の全域である。各セットの読み取りのために単語テーブルに登録した住所要素(都道府県名・市区郡名・町名など)の規模は、セットK: 49個、セットF: 231個である。市区郡名については、読み取り対象の市区郡名だけでなく、その周辺の市区郡名も登録した。また、同一町名の異表記(例えば「一の宮」「一ノ宮」「一宮」、「大淵」「大渕」、「青島」「青島町」など)は別の町名としてカウントした(以降の正解判定でも別町名として扱う)。

入力データの特性を決定する文字切り出しと個別文字認識の方式には、石寺らの文字切り出し方式[15]と、津雲の文字認識方式(多段弾性照合法)[16][17]を用いた。今回の評価データにおける文字切り出しや個別文字認識の多候補の度合いは次の通りである。

セグメント多重度: 地名領域において、正解文字数に対するセグメント候補数の比は平均1.9個程度。

文字多重度: 各セグメント当たりの候補文字数は平均2.8個(最大5、最小1、各セグメントで一律ではない)。

これら2つの多重度をかけ合わせたものが、第3.1節で定義した入力データの候補多重度Mに相当する。今回の評価データでは$M = 1.9 \times 2.8 = 5.3$程度となる。

3.2.2 コストの詳細定義

今回の実験では第2.4節のコスト定義を以下のように詳細化した。ここで述べる定義は、筆者の直感にしたがって定めたものであり、最適なパラメータ設定がされているという保証はない。また、評価対象データに合わせた特別な調整もいっさい加えていない。

初期コスト $icst(rcst) = rcst + CST_{init}$ において、文字認識コスト$rcst$の値域は$0 \leqq rcst \leqq 50$とした。$CST_{init} = -500$とした。

連結時コスト $ccst(m,n) = tcst_m + tcst_n + dcst(m,n) + bcst(m,n)$ において、間隔コストは $dcst(m,n) = (10 \cdot leval(n) + 100) \cdot pdis(m,n)$ とした。ここで、$leval(n)$は$tag(n)$に対応する住所要素の階層レベル(都道府県レベル:0、市区郡レベル:1、町レベル:2)を表わす。間隔不整合コスト$bcst(m,n)$については、次のようにケースを分けた。$ldis(m,n)$と$pdis(m,n)$のいずれかが0のとき: $bcst(m,n) = 1000 \cdot ldis(m,n) + 1000 \cdot pdis(m,n)$。$ldis(m,n)$と$pdis(m,n)$のいずれも0でないとき: $bcst(m,n) = 300 \cdot |\, ldis(m,n) - pdis(m,n) \,|$。カット条件の閾値は$CST_{thresh} = 1500$とした。

先頭補正コスト $scst(n) = (10 \cdot leval(n) + 100) \cdot (tpos_n - 1)$ とした。

末端補正コスト $ecst(n) = (10 \cdot leval(n) + 100) \cdot (tlen_n - tpos_n)$ とした。

上記のようなコストの詳細定義に合わせると、$tag(m)$と$tag(n)$の間の領域サイズを表わす物理的間隔$pdis(m,n)$は、1文字の平均的サイズを基本単位として数えるのが望ましい。しかし、今回は近似的に、最も細かいセグメント単位で数えた$tag(m)$と$tag(n)$の間の距離($ttop_n - ttop_m - twid_m$)を物理的間隔$pdis(m,n)$とにした。

3.2.3 読み取り精度

文字タグ法の読み取り精度は、正解地名文字列の構成文字が入力データ中にどれくらい含まれていたかの割合Rと関係付けて測定した[2]。図5に測定結果を示す。図5に示したRの値域ごとに50件ずつのデータをピックアップして測定した。文字タグ法で計算した最良コストの読み取り候補(第1候補)が、正解の地名文字列と完全一致したかどうかで判定した。

図5で第1候補が不正解となった原因には次のようなものがあった。その比率は図6に示す。

a) 住所要素消失: ある住所要素(市区郡名、町名など)のどの文字も入力データ中に含まれていない。

b) 誤字マッチ: 入力データ中に正解文字が少なく、誤認識文字を組み合わせて偶然にできた地名候補の方が読めた文字数が多い。

c) ノイズ化: 長く伸びたストローク(「市」の中棒など)が独立のセグメントとなり、ノイズとして振る舞う。

d) カット条件: 市区郡名が先頭1文字程度しか読めていないときに、文字タグ連結カット条件が働いて、読み取り結果から市区郡名が欠落する(町名のみ読み取られたことになる)。

[1] フリービッチ手書き文字列の読み取りの実際の場面では読み取り対象領域検出の曖昧性に対する頑健性も求められるためである。

[2] 図4の例ならば、正解地名文字列は「川崎市宮前区宮崎」という8文字で、そのうち「前」を除く7文字については入力データ中に含まれているので、$R = 7/8 = 87.5\%$ となる。

正解文字数のうち 入力データ内に含まれた割合 R	セットK		セットF	
	1位正解	不正解	1位正解	不正解
$0\% \leqq R < 40\%$	5件	45件	4件 (4件)	46件
$40\% \leqq R < 60\%$	21件 (3件)	29件	13件 (6件)	37件
$60\% \leqq R < 80\%$	43件 (6件)	7件	34件 (9件)	16件
$80\% \leqq R < 100\%$	44件 (3件)	6件	47件 (15件)	3件
$R = 100\%$	50件	0件	47件	3件

(注) 1位正解の () 内の値は1位が同点で複数あったもの

図 5: 文字タグ法の読み取り精度

e) コスト調整: 微妙なコスト差による。あるいは、物理的間隔の近似や補正コストの定義などで改良の余地がある。

f) 番地マッチ: 地名領域だけでなく番地領域も入力データに含んでいたため、番地領域と部分一致する長い町名が存在した (例えば「今泉三丁目」に「今泉三条」がマッチするケース)。

g) 異表記: 同一町名の異表記の方が一致の度合いが高くなった (例えば「一ノ宮」の「ノ宮」が統合セグメントになり、その文字認識結果が「宮」となった結果、「一宮」が第1候補になるようなケース)。

3.2.4 処理時間とタグ生成数

文字タグ法のプログラムは、C言語でコーディングし、UNIX ワークステーション NEC EWS4800/330 (CPU: R4400、クロック:67MHz、主記憶:64MB) 上で実行した。処理時間の測定は10ミリ秒を最小単位として実施したが、セットK・セットFのいずれのデータでも1件の処理に要した時間は0または10ミリ秒であった。

また、文字タグの生成数は、セットKで平均76.7個/件、セットFで平均217.5個/件となった。

3.3 考察と議論

文字タグ法の入力データに対する時間計算量は、単語辞書と候補文字列とを組み合わせ照合する従来法に比べて明らかに優位である。従来法は、文字切り出しに曖昧性がないとした場合でも $O(L \cdot M^W)$ の時間計算量は必要であり (W は単語長に関わるファクタ)、文字切り出しの曖昧性をかけ合わせたり、虫食い照合のあらゆる可能性を調べようとすると、時間計算量は $O(L \cdot M^L)$ や $O(L^L \cdot M^L)$ のような組み合わせ爆発を起こす。文字タグ法では、すべての可能性を考慮した最良解計算を保証した上で、時間計算量 $O(L^2 \cdot M^2)$ または $O(L \cdot M^2)$ を達成している。

第3.2節の性能評価結果は、手書きの宛名住所読み取りにおいて、文字タグ法が、文字切り出しの多候補や個別文字認識の多候補のあらゆる組み合わせを十分に高速に探索できていることを示している。この実験では入力データの候補多重度 $M = 5.3$ であるから、文字列長が8の場合を単純な目安として取り上げると $5.3^8 = 60$ 万通りの組み合わせのなかから最良なものを探索する能力をみていることになる。図5によれば、入力データが上述のような候補多重度をもっていたとしても、そのなかに正解文字数の8割以上が含まれていれば、文字タグ法の第1候補正解率は94%になる[3]。

原因	a	b	c	d	e	f	g	合計
$0\% \leqq R < 40\%$	69	22						91
$40\% \leqq R < 60\%$	11	36			19			66
$60\% \leqq R < 80\%$	7	3	4	2	2	3	2	23
$80\% \leqq R < 100\%$		3	1		3	1	1	9
$R = 100\%$						3		3
合計	87	64	5	2	24	7	3	192

図 6: 読み取り不正解の内訳

図5で第1候補が不正解になったものでも、図6の原因分類でe～gに該当するケースでは、第1候補の読み取り結果と僅差のコスト値をもつ別候補のなかに正解の読み取り結果が含まれている。この点を考慮して、正解文字の欠落 (虫食い) がある不完全な入力データから可能性の高い読み取り候補 (複数通り) を推定する能力という観点でみると、入力データ中に正解文字数の6割以上が含まれていれば90%以上のケース[4]において文字タグ法は有効な探索結果を出力している。また、図6の原因a・bに該当したケースは、文字タグ法の第1候補が正解でな

[3] 図5において $R \geqq 80\%$ の件数は合計200件、1位正解の合計は188件であるから、$188/200 = 94\%$ となる。

[4] 図5において $R \geqq 60\%$ の件数は合計300件、1位正解の合計265件に図6で $R \geqq 60\%$ かつ原因e～gの合計15件を加えると280件であるから、$280/300 = 93\%$ となる。

かったとしても、入力データとして与えられた情報から最良の解釈を選択するという文字タグ法の探索能力を否定するものではない。

上述のような文字タグ法の推定能力を利用して、より高精度な読み取りを実現するには、トップダウン的な検証[18][19][20]との組み合わせが有効だと考えている。すなわち、文字タグ法によって虫食い箇所が特定され、その領域の文字列推定がなされるので、その推定を検証する機構を起動することで、不適切な推定を棄却し、虫食いを含む複数通りの推定のなかから、より信頼性の高いものに決定することが可能になる。このようなトップダウン的検証機構を自然に取り込める拡張性・柔軟性も文字タグ法の特長である。

文字タグ法の時間計算量やメモリ使用量で問題になるファクタは、入力データに依存するLやMよりもむしろ、それにかかる係数の方であろう。各文字に対してどれくらいの数の文字タグが生成されるかが、その係数を決定する。一般の日本語文章を想定した数万語オーダーの単語セットを扱うと、上記の係数値はかなり大きなものになって現実的ではないかもしれない。しかし、宛名住所読み取りの実際的な応用を考えると問題にはならない。なぜならば、通常、宛名住所と併記される郵便番号によって地域を限定でき、数十語から数百語程度(多くても千語程度)の単語セットを扱えばよいからである。

4 おわりに

フリーピッチ手書き文字列の読み取りのために、新しい知識処理アルゴリズムである「文字タグ法」を提案した。従来法が単語の単位を強く意識しているのに対して、本アルゴリズムでは文字を基本単位としてタグを付与し、その位置関係をチェックしながら連結していく。単語内の文字の連結と単語間の文字の連結とを同等に扱い、動的計画法による最良解探索を実行するので、効率性と精度を保証している。文字切り出しや個別文字認識の誤りによる正解文字の欠落を制限しない頑健性もそなえている。従来法は正解文字欠落のあらゆる可能性を想定したときに単語虫食い照合で組み合わせ爆発を起こしてしまうが、文字タグ法での計算量は、入力文字列の長さLと候補多重度Mに対して$O(L^2 \cdot M^2)$または$O(L \cdot M^2)$に抑えられる。さらに、手書きの宛名住所の地名領域読み取りに文字タグ法を適用して、その有効性を確認した。また、文字タグ法は、虫食い個所のトップダウン的な検証も自然に取り込める拡張性・柔軟性もそなえている。

本論文で提案した文字タグ法のアルゴリズムでは、常に最良コストのものを読み取り結果として出力し、信頼性の低い場合に読み取り結果を棄却する仕組みはまだそなえていない。今後は、コスト差を分析したり、トップダウン的な検証を組み合わせるなどして、文字タグ法に棄却判定能力をもたせていくことが課題と考えている。

最後に、本研究に関して多くの助言と協力をいただいた山内俊史主任をはじめとするNEC産業オートメーション事業部の関係者、並びに、山田敬嗣課長をはじめとするNEC情報メディア研究所の関係者に深謝する。

参考文献

[1] 鈴木ほか、住所認識装置の選択後処理方式、信学技報:PRU88-154、1988年。
[2] 磯山、住所文字列に対する文字認識後処理方式の検討、情処研報:91-NL-82-3、1991年。
[3] 村瀬ほか、言語情報を利用した手書き文字列からの文字切り出しと認識、信学論:J69-D(9)、1986年。
[4] 仲林ほか、単語あいまい検索法を利用した枠無し文字切り出し手法、情処34全大:4E-8、1987年。
[5] 小林ほか、文字連接情報を利用した手書き文字列認識、信学技報:PRU91-67、1991年。
[6] 木谷、手書き文書の文字認識結果に対する後処理方式、情処研報:91-NL-86-1、1991年。
[7] 丹羽ほか、パターンと記号の統合化処理による文字認識、信学論:J78-D-II(2)、1995年。
[8] 長田ほか、日本語の文脈情報を用いた文字認識、通学論:J67-D(4)、1984年。
[9] 新谷ほか、認識情報及び単語・文節情報を利用した文字認識後処理、信学論:J67-D(11)、1984年。
[10] 池田ほか、手書き原稿認識における語彙および構文の検定、情処論:26(5)、1985年。
[11] 杉村、候補文字補完と言語処理による漢字認識の誤り訂正処理法、通学論:J72-D-II(7)、1989年。
[12] 高尾ほか、日本語文書リーダ後処理の実現と評価、情処論:30(11)、1989年。
[13] 伊東ほか、OCR入力された日本語文の誤り検出と自動訂正、情処論:33(5)、1992年。
[14] 福島ほか、手書き住所読取りのための町名検索アルゴリズム－文字タグ法－、情処50全大:4D-6、1995年。
[15] 石寺ほか、手書き住所読み取りのための文字切り出し方法、信学総大:D-576、1995年。
[16] 津雲、方向パタンマッチング法の改良と手書き漢字認識への応用、信学技報:PRU90-20、1990年。
[17] 津雲、手書き漢字認識、NEC技報:47(8)、1994年。
[18] 下村ほか、手書き住所読取りにおけるパタン処理と連携した住所知識処理方式、情処50全大:4D-1、1995年。
[19] 濱中ほか、変形推定を用いた検証型文字認識の検討、信学総大:D-542、1995年。
[20] 佐瀬ほか、制限付文字列読み取りの一検討、信学技報:PRU88-115、1988年。