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―

4D－6

福島 俊一† 下村 秀樹† 森 義和‡
(NEC† NEC情報システムズ‡)

1 はじめに

郵便物の宛名住所のようにフリーピッチで書かれた手書き文字列は、字形が多様で、文字サイズにばらつきがあり、文字の接触・入組みなどもよく起きる。したがって、その読取りでは、誤切出し / 誤認識によって欠落した正解文字を補完可能な知識処理[1]が不可欠である。

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、さらに、単語の並びとしての妥当性を判定して読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段目の単語照合で虫食い照合を行うことで対処する。

しかし、このような従来の枠組みは、フリーピッチ手書き文字列の読取りを正確かつ効率よく行うのに、まだ十分なものとは言えない。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段目の虫食い照合で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、単語辞書の全探索あるいは候補文字の組合せ爆発が起きる。そうでなければ、虫食い照合に1文字不一致のみのような制限を付けて、可能性を切り捨てることになる。1段目の単語照合に限ってみれば、各文字位置から単語へのインデックスをもつ松本らの手法[2]が効率よい虫食い照合を可能にしているが、そのままではフリーピッチの単語列読取りには適用できない。

本稿では、上記のような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。手書き宛名住所から都道府県名・市区郡名・町名の並びを読み取る応用を例に概要を紹介する。

2 知識データの構造

文字タグ法で用いる知識データは、次のような単語テーブル(図1)と文字インデックス(図2)とで構成する。例えば、図1から「宮崎」には「宮前区」が前接可能で、図2から、「崎」という文字は「川崎市」の3文字中の2文字目、あるいは「宮崎」の2文字中の2文字目になり得ることがわかる。

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

0: [”川崎市”, {}]
1: [”宮前区”, {0}]
2: [”梶ケ谷”, {1}]
3: [”宮崎”, {1}]
4: [”宮前平”, {1}]
5: [”馬絹”, {1}]
6: [”野川”, {1}]
7: [”有馬”, {1}]

図 1: 単語テーブル

ケ:	[2,2,3]			川:	[0,1,3]	[6,2,2]
梶:	[2,1,3]			前:	[1,2,3]	[4,2,3]
宮:	[1,1,3]	[3,1,2]	[4,1,3]	谷:	[2,3,3]	
区:	[1,3,3]			馬:	[5,1,2]	[7,2,2]
絹:	[5,2,2]			平:	[4,3,3]	
崎:	[0,2,3]	[3,2,2]		野:	[6,1,2]	
市:	[0,3,3]			有:	[7,1,2]	

図 2: 文字インデックス

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

3 文字タグ法のアルゴリズム

入力データを次のような形式で記述し、処理過程で次のような形式のデータ(「タグ」と呼ぶ)を作成する。

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

タグ: $tag(n)$: $[ttop_n, twid_n, twrd_n, tpos_n, tlen_n, tpre_n, tcst_n, tflg_n]$
　n はタグ番号, $ttop_n$ 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 $tag(n)$ に対応する単語コード・単語内位置・長さ
　$tpre_n$ は $tag(n)$ が連結する最良のタグの番号
　$tcst_n$ は $tag(n)$ までのタグ連鎖の累積コスト
　$tflg_n$ は $tag(n)$ がタグ連鎖の末端か否か(フラグ)

文字タグ法のアルゴリズムは以下の通りである。図3には、例を用いた処理概要を示す。

Step1: ダミータグ $tag(0)$: [0, 1, -1, 1, 1, -1, 0, 1] を作成する;

Step2: 先頭側のセグメントから順にすべてのセグメント $seg(i)$ について Step2.1 ～ Step2.2 を実行する;

　Step2.1: $seg(i)$ 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;

An Address Elements Search Algorithm for a Hand-Written Address Reader

Toshikazu Fukushima†, Hideki Shimomura† and Yoshikazu Mori‡ (NEC Corporation† and NEC Informatic Systems Ltd.‡)

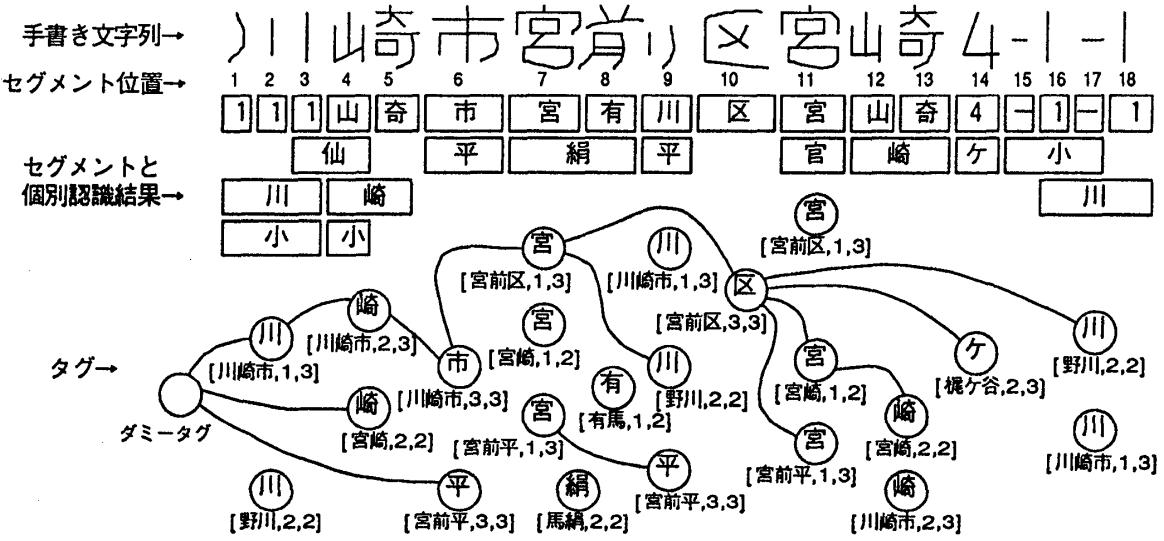

図 3: 文字タグ法による処理例

Step2.2: 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 $tag(n)$: [top_i, wid_i, wrd, pos, len, -1, 初期コスト+ $rcst_{i,j}$, 1] を作成する;

Step3: 先頭側のタグから順にすべてのタグ $tag(n)$ について Step3.1 ～ Step3.5 を実行する;

Step3.1: $ttop_m + twid_m \leqq ttop_n$ かつ $twrd_m = twrd_n$ かつ $tpos_m < tpos_n$ が成立するすべての $tag(m)$ を検索して, 連結したときのコスト $ccst(m,n)$ を計算する;

Step3.2: 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;

Step3.3: $ttop_m + twid_m \leqq ttop_n$ かつ $twrd_m = pwrd_k$ が成立するすべての $tag(m)$ を検索して, 連結時コスト $ccst(m,n)$ を計算する;

Step3.4: Step3.1 と Step3.3 で求めたコスト $ccst(m,n)$ が最小となる $tag(m)$ を求める;

Step3.5: Step3.4 で求めた $tag(m)$ に対して $ccst(m,n) - tcst_m \leqq$ 敷居値 ならば $tpre_n = m$, $tcst_n = ccst(m,n)$, $tflg_m = 0$ とし, それ以外の場合 ($tag(m)$ が存在しなかった場合も含む) は $tpos_n \neq 1$ ならば $tcst_n$ に先頭補正コストを加える;

Step4: $tflg_n = 1$ かつ $tpos_n \neq tlen_n$ であるようなすべてのタグ $tag(n)$ について $tcst_n$ に末端補正コストを加える;

Step5 $tflg_n = 1$ のタグ $tag(n)$ 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タグ連鎖を読取り結果とする;

4 コストの定義

2タグ ($tag(m)$ と $tag(n)$) 間の論理的間隔と物理的間隔を定義し、次のようなコストを計算する (値が小さいほど良いものとする)。

論理的間隔: タグ間の読み飛ばした文字数。すなわち、$twrd_m = twrd_n$ のとき $tpos_n - tpos_m - 1$、$twrd_m \neq twrd_n$ のとき $tlen_m - tpos_m + tpos_m - 1$。

物理的間隔: タグ間の領域 (セグメント位置 $ttop_m + twid_m$ から $ttop_n - 1$ まで) のサイズ。

初期コスト: 読めた文字が多い方が確からしいものと考え、ある程度大きい負の値とする。

連結時コスト: $tcst_n$ に $tcst_m$、および、論理的間隔に応じて大きくなるコスト、物理的間隔に応じて大きくなるコスト、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などを加算する。

先頭補正コスト: $tpos_n - 1$ に応じたコストを加算する。

末端補正コスト: $tlen_n - tpos_n$ に応じたコストを加算する。

5 おわりに

フリーピッチ手書き文字列 (特に宛名住所) の読取りのために、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、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、正解文字の欠落も想定した候補文字の組合せ的探索を動的計画法を利用して効率よく実行する。さらに、虫食い個所のトップダウン検証も自然に取り込める拡張性・柔軟性も特長である [3]。

参考文献

[1] 西野、文字認識における自然言語処理、情処学会誌:34(10)、1993 年。
[2] 松本ほか、連想統合型照合による単語あいまい検索、情処 34 全大:4E-7、1987 年。
[3] 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。