再帰遷移ネットワーク型文字列照合方式の高速化

古川 直広† 池田 尚司† 酒匂 裕† 加藤 陽介‡
†(株)日立製作所 中央研究所 ‡(株)日立製作所 公共システム事業部

1. はじめに

文字認識で利用される表記知識照合の高速化のために、再帰遷移ネットワーク型文字列照合(RTN 方式)を高速テキスト照合との組合せによって拡張した新方式を提案する。従来の RTN 方式では再帰遷移ネットワークの出次数が大きい場合、探索範囲が広大になり照合時間が掛かってしまう問題があった。この問題に対し、ビットの加算演算とシフト演算を利用した高速テキスト照合である Shift-Add 法を文字認識向けに拡張し、文字候補テーブルの先読み手段として RTN 方式と組合せることによって、より高速な文字列照合方式(RTN-SA 方式)を開発した。本稿ではその基本原理と評価実験結果について報告する。

2. 文字認識における文字列照合の概要

2.1. 文字認識

文字認識は主に下記 3 ステップからなる(図 1)。

(1) 文字切出し:入力画像から文字らしいパターンを抽出、
(2) 文字識別:各文字パターンを文字の識別、
(3) 文字列照合:文字識別結果を意味のある文字列に解釈。

文字切出しにおいて、文字パターン形状だけでは切出しが一意に決定しない場合があるため、考えうる切出しの仮説を文字切出しグラフで表現する。また文字識別においても、第 1 位候補にいつも正しい文字候補が出力されるとは限らないため、第 1 位候補のみでなく下位候補を含む複数個の文字識別候補を出力することが一般的である。

2.2. 文字列照合

文字切出しグラフの中から認識対象の文字列を検出する文字列照合の方式として、DP を用いた方法[1][2]、TRIE 表現された言語モデルを利用する方法[3]等がある。

また文字列照合処理を、文字切出しパス探索＋ラティス文字列照合の２段階の処理に分けて実行する方式もある。まず文字切出しグラフ中から最も良い文字パターンの組合せ(文字切出しパス)を探索し、そのパス上の各文字識別結果を順に並べて出来るテーブル(文字候補ラティス)を作成する。次に文字候補ラティスの中から認識対象の文字列を見つけ出す、という処理手順となる。この２段階処理での文字列照合として KMP 法や BM 法などがある。また高速高精度な手法として、有限オートマトン(Finite State Automaton: FSA)を用いた方式[5]がある。この FSA 方式では文字候補ラティスからオートマトンを生成し、それによって受理される単語を検出し、それら単語の階層関係を用いて統合していくことによって最終的な文字列を得る方式である。この方式は言語モデルとして単語単位の木構造を採用している。さらに再帰遷移ネットワーク(Recursive Transition Network: RTN)を言語モデルとする RTN 方式[6]も提案されている(図 2)。この方式は、(1)文脈自由文法に多くの表記をコンパクトに実装可能、(2)文字切出し・識別の不完全性許容により高精度の認識を実現、の特長を有す。

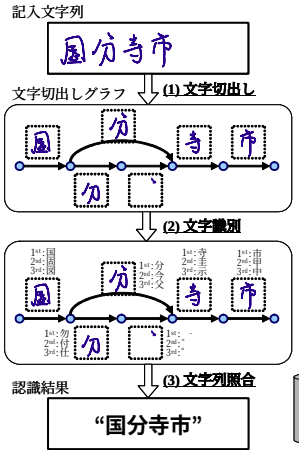

図 1 文字認識の流れ

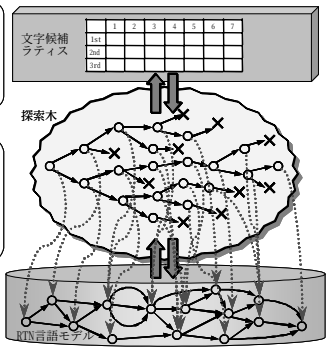

図 2 RTN 文字列照合

3. 課題とアプローチ

3.1. RTN 方式の問題点

しかし、RTN 方式はノードの出次数(out-degree)が大きい場合、探索空間が広がってしまい照合時間が掛かってしまう問題があった。特にこの方式は RTN を left-to-right の方向で探索していくため、RTN の始点付近に高出次数のノードが存在すると、探索木が大きくなってしまう。たとえば、全国の地名表記において都道府県の省略を許容すると開始ノードの出次数が 2,000 以上と大きな数となってしまうため探索時間が掛かってしまう問題があった。

3.2. 課題に対するアプローチ

前記課題に対し、まず下記 2 アプローチを検討した。

(a) left-to-right 以外の探索アルゴリズムの採用、
(b) 出次数が多いノードでの探索制限。

アプローチ(a)の例として、ボトムアップによる RTN 利用照合[7]がある。金額数字列のように RTN のサブグラフ数が少ない場合には有効な探索手法である。しかし地名表記ではサブグラフ数が非常に多いため、ボトムアップは不適である。また地名表記は階層関係があるため、やはり left-to-right が有利である。

したがって本稿ではアプローチ(b)を採用し、ラティスの先読みによって探索すべき RTN のエッジを限定することで探索範囲を制限する方針をとった。そのラティス先読み手段として、ビットの加算演算とシフト演算を利用して計算機アーキテクチャに特化したアルゴリズムである Shift-Add 法[4]を文字認識向けに拡張し適用した。

4. RTN-SA 型文字列照合方式

4.1. Shift-Add 法の基本原理

Shift-Add(SA)法の詳細は文献[4]で述べられている。その基本アイディアは、テキスト上のある位置からのパターンとの一致、不一致を 1 ビットで表し、そのビットの列を後述の Or (Addition) 演算、Shift 演算によって次の状態に次々と遷移させることにより、高速に文字列照合を解くものである。SA 法ではテキスト t の i 番目の文字まで照合したときの結果を、状態 $state_i \in \{0,1\}^{|p|}$ であらわす。

The Improvement of RTN String Matching Method,
Naohiro Furukawa†, Hisashi Ikeda†, Hiroshi Sako† and Yosuke Kato‡,
†Central Research Laboratory, Hitachi, Ltd., ‡Government & Public Corporation Information Systems Division, Hitachi, Ltd.

$state_i$ の各ビットがそれぞれパターン p の何文字目まで現在マッチングしているかを示す。各ビットで 0 はマッチング、1 のときはアンマッチングとなる。たとえば $(state_i)_k = 0$ のときは、テキスト t の i 番目においてパターン p の $|p|-k+1$ 番目までとマッチングしていることを意味する。このとき $state_i$ の各ビットが逆順に並んでいる点に注意する。逆順に並んでいる理由は、逆順の方が計算機に実装したときに高速であるためである。$(state_i)_1 = 0$ のとき、パターン p の全ての文字とマッチングしたことを意味するため、パターン p はテキスト t の中で、シフト $s = i - |p|$ で出現したことを意味する。状態 $state_i$ の初期状態と状態遷移規則は以下のとおりである。

初期状態: $state_0 = 11\ldots1$

状態遷移規則: $state_i = (state_{i-1} << 1) \mid T[t_i]$

ここで T は遷移関数であり、パターン p から作成される長さ $|\Sigma|$ の配列である。その作成規則を下記に示す。

$$\forall a \in \Sigma,\quad T[a] \in \{0,1\}^{|p|},\ T[a]_k = \begin{cases} 0 \cdots p_{|p|-k+1} = a, \\ 1 \cdots p_{|p|-k+1} \neq a \end{cases}$$

4.2. SA 法の拡張（RTN-SA 型文字列照合）

文字認識への適用を考えた場合、パターンを文字列ではなくラティスに対応しなければならない。文字切出しパス長(文字切出しパスに含まれる文字パターン数)を l、文字識別の最大候補数を r とおくと、文字候補ラティス L は r 行 l 列のテーブルとなる。ラティス L の j 行 i 列番目の要素を $L_{i,j}$ で表す。ここで SA 法のパターン p をラティス L の各要素を用いて下式のように定義した。なお、文字クラスを $[u]$ で表す $(u \in \Sigma^+)$ 。これは u 中の任意の文字が該当することを意味する。

$$p = [L_{1,1}L_{1,2}\cdots L_{1,r}][L_{2,1}L_{2,2}\cdots L_{2,r}]\cdots[L_{l,1}L_{l,2}\cdots L_{l,r}]$$

ラティスからパターンに対する遷移関数を作成し、RTN 探索時に出次数が大きいノードに対して、そのノードから出ている全エッジのラベル文字列をテキストとして SA 法による文字列照合を実行し、正当なシフトがなかったエッジに関してはその先の探索を停止することにより高速化を図ることが RTN-SA の基本処理である(図 3)。

なお、このような探索の枝刈りにおいて、SA 法以外の照合方式の適用も考えられる。しかし SA 法は、(1)一度遷移関数を作成しておけば照合対象テキストが多数存在しても高速に照合、(2)文字クラスや不一致を許容する照合方式(かつその場合の速度低下がない)、の利点から、SA 法が今回の利用目的に適していると判断した。

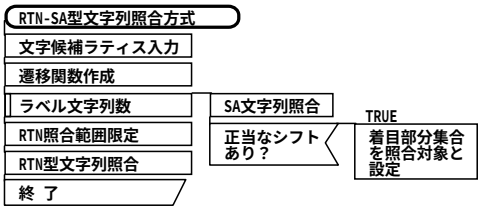

図 3 RTN-SA 型文字列照合方式処理フロー

5. 評価実験および考察

本方式の有効性を検証するために評価実験を行った。その実験条件と結果を以下に示す。

表 1 評価実験条件

項 目		内 容
実験装置	CPU	Pentium III XEON 1.0 GHz
	Memory	512 MB
表記	全国地名	約 42 万表記、都道府県省略許容
評価サンプル	サンプル数	306 サンプル
	平均文字列長	9.35 文字
	入力デバイス	Chat pen (アノトペン)

表 2 評価実験結果

項 目		RTN 方式	RTN-SA 方式
照合処理時間	平 均	58.2ms	16.6ms
	最 大	187.0ms	125.0ms
	最 小	31.0ms	0.0ms
参照率 (平均)		100.0%(951/951)	5.2%(49/951)
探索木サイズ(平均)		43,986	13,062

認識率は両方とも 93.1%であり変化がなかった。したがて、提案方式は精度低下なしに処理時間を 58.2ms から 16.6ms へと 3.5 倍高速化できたことが分った。SA 法導入により、RTN-SA 方式では全国地名表記の 951 個の部分集合うち実際に参照した平均数は 49 個であり、全体の 5.2%しか参照しなかったためである。また RTN-SA の最小照合時間から分るように SA 部分の時間は計測できないほど高速であった。

なお、今回考案の RTN-SA 方式の SA 法に着目して考えると、SA 法の状態の各ビットはラティスから作成された FSA 方式の単語照合での 1 状態と考えることができる。つまり SA 法の状態のビット列は FSA 方式におけるラティスの各シフトからの照合の状態を一纏めに表現したものであり FSA 方式の並列版と見なせる。したがって SA 法は一回の照合で FSA 方式のラティスの全シフトからの照合を実行できるため、SA 法単体でも FSA 方式よりも高速な文字列照合である。

参考文献

[1] F. Kimura, S. Tsuruoka, Y. Miyake, M. Shridhar, “A lexicon directed algorithm for recognition of unconstrained handwritten words”, IEICE TRANS. INF. & SYST., Vol. E77-D, No. 7, pp. 785-793, 1994.

[2] 古川直広, 今泉敦博, 藤尾正和, 酒匂裕, “星座認識による帳票識別方式”, 信学技報 PRMU 2001-125, Vol. 101, No. 421, pp. 85-92, 2001.

[3] M. Koga, R. Mine, H. Sako, and H. Fujisawa, "Lexical Search Approach for Character-String Recognition", Proc. IAPR Workshop on Document Analysis Systems, DAS'98, Nagano, Nov. 19, pp. 237-251, 1998.

[4] R. Baeza-Yates, G. H. Gonnet, "A New approach to text searching", Communications of the ACM, Vol. 35, No. 10, pp. 74-82, 1992.

[5] 丸川勝美, 古賀昌史, 嶋好博, 藤澤浩道, “手書き漢字住所認識のためのエラー修正アルゴリズム”, 情報処理論文誌, Vol. 35, No. 6, pp. 1101-1110, 1994.

[6] H. Ikeda, N. Furukawa, M. Koga, H. Sako, H. Fujisawa, “Context-Free Grammar-Based Language Model for String Recognition”, International Journal of Computer Processing of Oriental Languages, Vol. 15, No. 2, pp. 149-163, 2002.

[7] 古賀昌史, 嶺竜治, 安江司, 酒匂裕, 藤澤浩道, “チェックライター金額文字列認識の一手法”, 電子情報通信学会論文誌 D-II, Vol. J86-D-II, No. 6, pp. 836-845, 2003.