

逐次検索システムと言語について

九大理 情報研 有川節夫

1. はじめに

逐次サーチに基づく情報検索システムは走査するデータの量に少くとも比例する時間と検索時に必要とある。このために多数の利用者を対象にする一般的な IR システムとしては不向きである。しかし最近の廉価で高速な外部メモリの開発と普及によって、利用者に新しい問題意識が生れてきた。そして効率的な逐次サーチに関する理論的研究の進展によって、こうした問題の解決が可能になりつつある。

本稿では、数年来筆者等が開発して使用している一方向逐次サーチに基づく情報検索システム TEXTIR (Text Information Retrieval System) について、その概要及び能力について考察する。

2. 準備

Σ を字母とする。コードは Σ^* の要素であり、ファイルは Σ^* の部分集合であるといえる。ファイルはある規則 (方針) に従って集められるので、その規則を文法 G と考えらると、ファ

ファイルは文法 G によって生成される言語 $L(G)$ ということになる。通常ファイルは $L(G)$ の有限な部分集合である。

逐次検索システムにおいて質問 Q と満たすレコードの集合を検索することは、ファイル $L(G)$ と質問 Q が定義する文法 G' によって定まる言語 $L(G')$ との共通部分 $L(G) \cap L(G')$ を作ることに対応する。質問が $L(G)$ 又は Σ^* の或る明確な部分族と定義できれば、システムの能力を知る上で都合がよい。システムと質問文によって定義される言語族が正規集合族と一致するとき、システムを R -万能ということにする。逐次検索システムとこのように形式言語との対応関係で把えることによって、システムの能力の定性的評価が可能になる。

さて、TEXTIR は逐次サーチの部分に Aho-Corasick [1] によるパターン・マッチングマシンを使用している。パターン・マッチングマシンは Σ^* の要素としてのテキスト系列を左から右へ一方向に一度だけ走査して、与えられたキー (Σ^* の元) の有限集合 K の要素の存在場所とことごとく知らせてくれる一種の有限オートマトンである。従って、検索システムには、このように出来上がったオートマトンを系列上に走らせる部分と、与えられた K から有限オートマトンを作る部分が必要である。後者は goto 関数構成部と failure 関数構成部から成りたっている。

3. TEXTIR の機能と特徴

TEXTIR はパターンマッチングマシンを土台にして、次のことに留意して構成した。即ち、

- a) パターンマッチングマシンを効率良く実現すること。
- b) 機能をできるだけ拡張追加すること。
- c) しかし、そのために速さと効率と犠牲にしないこと。

その結果、結局もとの Aho-Corrick システムのもつ機能に加えて次の機能と利点をもつものになった。

1) $\Sigma^* x_1 \Sigma^* x_2 \Sigma^* \dots \Sigma^* x_n \Sigma^*$ 形の言語の認識。この機能はキーの出現順序が意味をもつような検索において特に有効である。例えば、 x followed by y の形の検索は、いわゆるトリプル・ドットを含むキー $A=x \dots y$ と論理式 $Z=A$ なる単純な質問で可能である。

2) 自由なレコード形式。このシステムでは検索の対象となるレコード形式を特定のものに限定していない。ファイル全体を1つの長い文字列とみて検索を行う。必要なレコードは、出力区切り語と通常のキーと同様に登録して、2つの出力区切り語で囲まれた区間として、検索時に自動的に取出ることができる。

3) 複数質問の同時処理。TEXTIR は一方向の逐次サーチに基づいているために、検索時間は必然的に走査対象のフai

ルの長さに比例する。そこで、一連の検索における全体の検索時間を短縮するために、最高15個までの箇所を同時に処理できるようにした。この場合も作成されるパターン・マッチングマシンは唯1個である。

4) ユニバースの制限。箇所は Σ^* のある部分集合を定義するわけであるが、字母 Σ は実際には、計算機で利用できる全ての文字や記号からなる。箇所によっては、このユニバース Σ^* と適当な部分集合、例えば、 $\Sigma^* (\Sigma \subset \Sigma)$ に制限したくなる場合もある。これは理論的にはいつでも可能であるが、実際には箇所において多数のキーを登録しなければならないため実用的ではない。そこで TEXTIR では *no reset* というモードを指定することによって、初期状態への *failure transition* と、次の出力区切り語を検出するまで禁止することにした。これによって、キー及び入力・出力区切り語に使用された語の集合を X とあるとき、ユニバースをほぼ X^* に制限することが可能になった。このモードの選択による検索時間への影響はない。

5) メモリの節約。Aho-Corasick の技法と直接的に採用すると、パターン・マッチングマシンを実現するには、*goto* 関数と状態図で表わした場合の「ノード数」+「終点ノード数」= n となるとき、サイズ $3n$ の遷移表が必要である。しかしこの表の

goto関数の行は、任意の状態 k に対して $goto(k) = k+1$ となるように出来るので、除去できる。したがって表のサイズとこれに絡めることができる。

6) 検索時間の短縮。遷移表のサイズを固定して、無駄なε-動作と出来るだけ減少させるようにした。(文献[4]参照)

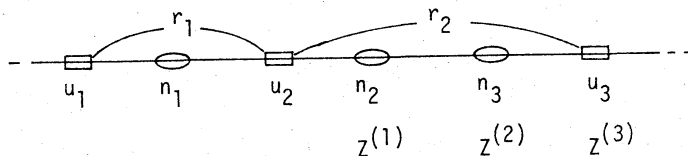
7) R-万能性。式変数とε-変数^④と論理式に許容とによって R-万能性と得た。したがって、このシステムで少なくとも理論的にはすべての正規言語を定義することが出来る。更に正規文法で書かれたレコードの構文チェックも可能となった。

4. 質問文と検索

TEXTIRでは、区切り語やキー、論理式の登録に先立って、トリアル・ドットと表わす記号、復改用の記号、エラー訂正用の記号、reset/noresetの指定を行うことが出来る。勿論システムには標準的な記号、モードが仮定されているので、これらの指定を省略することも出来る。

1) 区切り語。区切り語は Σ^+ の要素であり、出力区切り語と入力区切り語とがある。出力区切り語は長いテキスト系列の中に仮想的なレコードを設定して、入力区切り語は論理式を評価すべき区間を指定する。通常は、出力区切り語は入力区切り語としても働く。システムは入力区切り語を検出するたびに論理式を評価し、出力区切り語と検出した時点での論理

式の値が 1 (true) であるような質問に対する検索結果として、最近の二つの出力区切り語で囲れた仮想的レコードと出力する。図 1 の場合には、「通常の」検索においては、出力区切り語 u_3 を検出した時点で論理式 Z の論理和が



$$Z^{(1)} + Z^{(2)} + Z^{(3)} = 1$$

であるときに、出力区切り語 u_2 と u_3 で囲れたレコード r_2 をこの

図 1 出力/入力区切り語

□ : 出力区切り語 (u_1, u_2, u_3)

○ : 入力区切り語 (n_1, n_2, n_3)

$Z^{(i)}$: 論理式の値

論理式に対する検索結果として出力する。

キー、入力区切り語、出力区切り語には、この順序で優先順位がある。したがって、ある記号列がキーでもあり、入力区切り語でもあり、更に出力区切り語でもある場合には、システムは、先ずキーであることをマークして、論理式と詳細し、仮想的レコードの作り出しを行う。

2) キー。キーには通常のキーとトリプル・ドット (これを $\circ$ で表わす) を含む $x_1 \circ x_2 \circ \dots \circ x_n$ の形のキーとがある。キー、区切り語の登録はシステムが発行する変数 $A_1, A_2, \dots$ に続いて行う。

3) 論理式。「通常の」検索における論理式は式変数 $Z_1, Z_2, \dots$ に続いてキー変数 $A_1, A_2, \dots$ と論理和 (+), 論理積 ($\cdot$), 否定 ($-$) 及び括弧 [,] を用いて登録する。一般に TEXTIR では論理

式に式変数 A_i と ε -変数 $@$ を使うこともできる。 ε -変数 $@$ は特別な働きをする。 $@$ の値はシステムが出力区切り語とまじに離れようとしているときだけ 1 で、その後は次の出力区切り語と検出するまで 0 である。

論理式の評価は $Z_1, Z_2, \dots$ の順序で行われ、出力区切り語を検出して次の仮想レコード上を走査する直前に、 $Z_i := \text{inv}Z(i)$ によって初期化される。ここに $\text{inv}Z(i)$ は

$$Z_i = f_i(@; A_1, \dots, A_n; Z_1, \dots, Z_m)$$

とあるとき、

$$\text{inv}Z(i) = f_i(1; 0, \dots, 0; \text{inv}Z(1), \dots, \text{inv}Z(i-1), 0, \dots, 0)$$

である。

箇所リストに $@$ が使用されていないで、どの論理式 $Z_i = f_i$ にも式変数 Z_j ($j \geq i$) が使用されていないときには、「通常の」方式で検索する。

5. システムの能力

これまでに Aho-Corasick のパターン・マッチング・マシンに論理演算 (+, ·, -) だけと使えるシステムを始めとして、「通常の」検索とある TEXTIR, 一般の TEXTIR の3種類の逐次情報検索システムを考えてきたことになる。本節ではこれらの能力について考えてみる。

「通常の」TEXTIR においては、キー $A=x$ は言語

$$\|A\| = \|x\| = \Sigma^* x \Sigma^*$$

と定義し、キー $A = x_1 \circ x_2 \circ \dots \circ x_n$ は言語

$$\|A\| = \|x_1 \circ x_2 \circ \dots \circ x_n\| = \|x_1\| \cdot \|x_2\| \dots \|x_n\|$$

を定義すると考えられる。この解釈 $\|\cdot\|$ は一般の論理式にも

$$\|\alpha + \beta\| = \|\alpha\| \cup \|\beta\|$$

$$\|\alpha \cdot \beta\| = \|\alpha\| \cap \|\beta\|$$

$$\|-\alpha\| = \Sigma^* - \|\alpha\|$$

によって拡張できるから、論理式 $Z_i = f(\alpha_1, \dots, \alpha_n)$ は言語

$$\|Z_i\| = \|f(\alpha_1, \dots, \alpha_n)\|$$

と定義することになる。

一般の TEXTIR の質内に対しては多少工夫を要するが、同様に $\|Z_i\|$ でもって、論理式 Z_i が規定する言語を表わすことになる。そこで、次のような3種類のシステムによって定義される Σ 上の言語族を考えてみよう。

$A(\Sigma) = \text{Aho-Corasick システムによって定義される言語族,}$

$R(\Sigma) = \text{「通常の」 TEXTIR による言語族,}$

$W(\Sigma) = \text{一般の TEXTIR による言語族.}$

さらに $\text{Reg}(\Sigma)$ で Σ 上の正規言語の族を表わすことになる。

こうすると、族 $A(\Sigma)$ や $R(\Sigma)$, $W(\Sigma)$ の大さき、或いは

$\text{Reg}(\Sigma)$ との関係でもって、システムの検索能力を知ること

ができる。ここでは $\Sigma := \Sigma - \{a\}$ とし、変数の個数、キーや区切り語の長さに関しては、それが有限であることを除いて特に制限はないものと仮定する。

- 〈定理1〉 (1) $A(\Sigma) = R(\Sigma) \subsetneq W(\Sigma)$ ($\#\Sigma = 1$ のとき),
 (2) $A(\Sigma) \subsetneq R(\Sigma) \subsetneq W(\Sigma)$ ($\#\Sigma \geq 2$ のとき),
 (3) $W(\Sigma) = \text{Reg}(\Sigma)$.

略証. 定義から $A(\Sigma) \subset R(\Sigma) \subset W(\Sigma) \subset \text{Reg}(\Sigma)$ であることは明らかである。また $\#\Sigma = 1$ のときは、

$$\|x_1 \circ x_2 \circ \dots \circ x_n\| = \|x_1 x_2 \dots x_n\|$$

であるから、 $A(\Sigma) = R(\Sigma)$ である。更に $\Sigma = \{a\}$ とするとき、言語 $L_1 = (aa)^*$ は $R(\Sigma)$ に属さないことが次のようにして示される。 L_1 が $R(\Sigma)$ に属すると仮定して、キーのリスト

$$A_i = a^{r_i} = aa \dots a \quad (r_i \text{ 個}, 1 \leq i \leq n)$$

をもつ関数 $Z = f(A_1, A_2, \dots, A_n)$ が L_1 を定義するとする。

論理式 Z は積和形式

$$Z = f(A_1, A_2, \dots, A_n) = \sum_i A_1^{e_{i1}} \cdot A_2^{e_{i2}} \dots A_n^{e_{in}}$$

で表わされる。ここに $e_{ij} = 0$ 又は 1 であり、 $A_j^0 = -A_j$, $A_j^1 = A_j$ である。いま $e_{ij} = 0$ とすると、

$$\begin{aligned} \|A_1^{e_{i1}} \dots A_j^{e_{ij}} \dots A_n^{e_{in}}\| &\subset \|A_j^0\| \\ &= a^* - \|A_j\| \end{aligned}$$

$$= \{\varepsilon, a, aa, \dots, a^{n-1}\}$$

となる。 L_1 は無限集合であるから、少くとも1つの項

$A_1^{e_{k_1}} \dots A_n^{e_{k_n}}$ に対しては、すべての j に対して $e_{k_j} = 1$ で

なければならぬ。そうすると

$$\begin{aligned} \|A_1^{e_{k_1}} \dots A_n^{e_{k_n}}\| &= \|a^{\max(k_1, \dots, k_n)}\| \\ &= a^{\max(k_1, \dots, k_n)} \cdot a^+ \end{aligned}$$

となり、 $m > \max(k_1, \dots, k_n)$ に対して、

$$a^m, a^{m+1} \in \|A_1^{e_{k_1}} \dots A_n^{e_{k_n}}\| \subseteq L_1$$

となり、 L_1 が L_1 を定義することに反する。したがって L_1

は $R(\Sigma)$ に属さない。 L_1 が $W(\Sigma)$ に属することは L_1 が正

規であることと (3) に依る。これで (1) が証明された。

(2) は $\Sigma = \{a, b\}$ とするとき、 $L_2 = \Sigma^* a \Sigma^* a \Sigma^*$ が $R(\Sigma)$ に属すが、 $A(\Sigma)$ には属しないことに依る。実際 L_2 はキー $A = a \circ a$ と論理式 $Z = A$ によって定義される。 $A(\Sigma)$ に属しないことは上の L_1 に関する議論とほぼ同様にして示される。

(3) は任意の有限オートマトンを TEXTIR の中でシミュレートできることを示せばよい。 $M = (K, \Sigma, \delta, s_1, F)$ を有限オートマトンとする。 $K = \{s_1, s_2, \dots, s_m\}$, $\Sigma = \{a_1, \dots, a_n\}$ として、 $a_1, a_2, \dots, a_n$ を入力区切り語として登録して、各キー変数にキー a_i と $A_i = a_i$ ($1 \leq i \leq n$) と割り当ててお

く、勿論モードは *noreset* を選んでおく。そして質向リストを

$$Z_1 = \sum_{(j,k): \delta(\delta_k, a_j) = \delta_1} A_j \cdot Z_{n+k} + @,$$

$$(j,k): \delta(\delta_k, a_j) = \delta_1$$

$$Z_i = \sum_{(j,k): \delta(\delta_k, a_j) = \delta_i} A_j \cdot Z_{n+k} \quad (1 \leq i \leq m),$$

$$(j,k): \delta(\delta_k, a_j) = \delta_i$$

$$Z_{m+i} = Z_i \quad (1 \leq i \leq m),$$

$$Z_{2m+1} = \sum_{\delta_i \in F} Z_i,$$

とある。 $\|Z_{2m+1}\| = L(M)$ であることは明らかであろう。

定理1(3)はTEXTIRがR-不能であることを示している。したがってこれを正規文法で書かれたLコードの構文チェックにも用いることができる。即ち文法と表現する質向リストを作り、テキスト系列としてそのLコードと与え、それが検索結果として出力されるならば、文法に適合していて、そうでなければ文法的誤りがあると判定できる。

実際、例えば図2で与えられる文法Nに対しては、次のようなTEXTIRの質向が対応する。モードを *noreset*

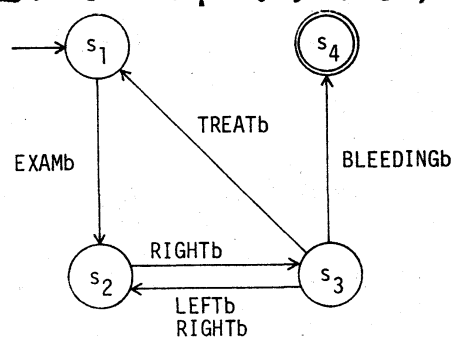


図2 不完全有限オートマトンN
(正規文法)

に選い、入力区切り語としてはスペースと登録し、キーを $A_1 = \text{TREAT}$, $A_2 = \text{BLEEDING}$, $A_3 = \text{LEFT}$, $A_4 = \text{RIGHT}$, $A_5 = \text{EXAM}$ とする。そして論理式と

$Z_1 = Z_2 \cdot A_2$, $Z_2 = Z_3 + A_4$, $Z_3 = Z_4 \cdot A_5 + [A_3 + A_4] \cdot Z_5$,
 $Z_4 = Z_5 \cdot A_1 + @$, $Z_5 = Z_2$ とある。そうすると $\|Z_1\| = L(N)$
 となり, Z_1 でこの文法の構文チェックができる。

さて、この定理からトリポルドットの機能は他の論理演算
 では表わせないことも分った。トリポルドットに因しては便
 に次の定理が成立する。

$R^{(R)}(\Sigma) =$ 「通常の」TEXTIRにおいて、トリポルドットと
 1個のキーの中で高々1個しか使用できら
 いような質問によって定義される言語族

とある。

(定理2) $A(\Sigma) = R^{(0)}(\Sigma) \subsetneq R^{(1)}(\Sigma) \subsetneq \dots \subsetneq R(\Sigma)$.

略証. $L_n = \|a \circ a \circ \dots \circ a\|$ (n 個) は $R^{(n)}(\Sigma)$ に属するが、
 $R^{(n-1)}(\Sigma)$ には属しないことが示される。

6. おわりに

パターンマッチングマシンを利用する一方向逐次サーチに基
 づく情報検索システム TEXTIR について概説し、その能力を
 考えてみた。システムは「通常」版は FORTRAN で一般の方
 は Coral 66 で書かれている。今後の方向として、ここで導
 入した区切り語の概念を一般化して、更にアクションも導入
 して、検索だけでなく、一般性のあるテキスト編集のための
 システムにある問題がある。

7. 参考文献

- [1] Aho, A.V. and Corasick, M.J. Efficient string matching: An aid to bibliographic search, CACM 18 (1975), 333-340.
- [2] 有川, 武谷, 石橋. パターン・マッチング・マシンを用いた例文検索システム, 情報処理学会論文集 (1976), 119-120.
- [3] Takeya, S. and Arikawa, S. TEXTIR - A text information retrieval system using pattern matching machines, Res. Rept. Res. Inst. Fund. Inform. Sci., Kyushu Univ. 83 (1978), 1-17.
- [4] Arikawa, S., Hill, A.G. and Townsend, H.R.A. An information retrieval system based on one-way sequential search, Res. Rept. MIRU, Univ. of Edinburgh (to appear).