データベースシステム 116－20
（1998. 7. 8）

最適パタン発見に基づくテキストデータマイニング：大規模テキスト索引における高速な実装方式

笠井 透, 有村 博紀, 藤野 亮一, 有川 節夫
九州大学 大学院システム情報科学研究科 情報理学専攻
〒812-8581 福岡市東区箱崎 6-10-1
Tel: 092–642–2688, Fax: 092–642–2698
E-mail: {kasai,arim,fujino,arikawa}@i.kyushu-u.ac.jp

与えられた大量の文書の集積から，分類精度を最大にする文字列パタンを見つける問題を考察する．複数の文字列の近接した出現を記述する近接語相関パタンを導入し，一様確率で生成されたランダムテキストに対して，分類精度を最大化する (d,k)-語相関パタンを平均計算時間 $O(k^{d-1}n\log^{d+1}n)$ および領域 $O(k^{d-1}n)$ で計算するアルゴリズムを与える．さらに，大規模テキストデータ向けの索引構造である接尾語配列上での実装方法についても考察する．

データマイニング, 分類誤差最小化問題

Text Data Mining Based on Optimal Pattern Discovery — Towards a Scalable Data Mining System for Large Text Databases —

Toru Kasai, Hiroki Arimura, Ryoichi Fujino, Setsuo Arikawa
Dept. of Informatics, Kyushu University
Hakozaki 6-10-1, Fukuoka, 812-8581 Japan
Tel: 092–642–2688, Fax: 092–642–2698
E-mail: {kasai,arim,fujino,arikawa}@i.kyushu-u.ac.jp

We consider the problem to find a pattern over strings that maximizes the precision of classifying the given positive- and negative-labeled strings. We introduce a notion of patterns for specifying associations between strings, and present an efficient algorithm that finds a best pattern that maximizes the precision and runs in $O(k^{d-1}n\log^{d+1}n)$ expected time and with $O(k^{d-1}n)$ space for uniformly random texts. We also discuss an implementation technique for the algorithm on the suffix array indexing structure.

data mining, maximum agreement problem

1 はじめに

データマイニング (Data mining) とは，データベースに蓄積された大量のデータから，自明でない規則性やパタンを半自動的にとりだす方法についての科学的研究である．データマイニングは，現在，ビジネス分野や科学技術分野をはじめとするさまざまな対象分野で，その適用が盛んにおこなわれている．近年，発展の著しいテキストデータベースに関しては，

1. 明示的な構造をもたない，
2. 多様な内容をもつ電子化文書の，
3. 数ギガバイトから数テラバイトにおよぶ膨大なデータの集積である

などの理由から，従来の方法は適用できない．そのため，われわれは，テキストデータマイニングを情報検索の逆問題として定式化し，構造のないテキストから2語相関パタンと呼ばれる単純なパタンを発見するアルゴリズム [2] を開発してきた．

本研究では，これまでの研究に基づき，大規模なテキストデータベースへの適用を目指して，以下のような拡張を行う．はじめに，任意定数個の部分語からなる相関パタンを扱えるよう拡張する．つぎに，平均値解析の枠組みを採用し，ゲノムデータにみられるランダムなテキストに対して，とくに高速な計算をおこなうアルゴリズムを開発する．さらに，接尾語配列上での実装法についても議論する．

2 語相関パタン

本稿では，アルファベット Σ 上の任意の文字列を語 (word) とよぶ．正整数 d, k に対して，**語相関パタン**またはパタン (proximity word-association pattern) とは，与えられたテキストの d 個の部分語と非負整数（距離パラメータ）k から構成される単純なパタン $\pi = (p_1, \ldots, p_d; k)$ である．このパタンは，テキスト中に，これらの部分語が文字数 k 以下の距離で，指定された順に連続して出現するという制約を表す．以後，近接度が k で d 個の語からなる語相関パタンを，(d, k) 語相関パタンと呼ぶ．

以下は2語相関パタンの例である．

(TATA, AGGAGGT, CACA; 30).

(“data mining”, “large”, “databases”; 8).

従来用いられてきた単純なキーワードの論理積と比較した場合，語相関パタンは，(i) 与えられたテキストの任意の部分語が扱えることと (ii) 文脈情報を表現できるという特徴がある．これらの特徴から，語相関パタンはウェブ検索や，ゲノム情報学で有用だと考えられている．

形式的には，語相関パタンの意味はつぎのように定める．

定義 1. 語相関パタン π がテキスト $s \in \Sigma^*$ に照合する とは，位置の列 $i_1, \ldots, i_d$ が存在し，任意の $1 \leq j < l$ に対して以下の (i), (ii) が成立することをいう．

(i) 各 p_j は位置 i_j に出現する s の部分語である．

(ii) それぞれの位置はたかだか k しか離れていない，すなわち $0 \leq i_{j+1} - i_j \leq k$ が成立する．

π が s に照合する（照合しない）とき，$\pi(s) = 1$ と（$\pi(s) = 0$ と）書く．2つのパタン $\pi = (\alpha; k)$ と $\tau = (\beta; k)$ の連接を $\pi\tau = (\alpha\beta; k)$ と定義する．整数 $i \leq j$ に対して，

3 最適パタン発見問題

テキストデータマイニングを，情報検索の逆問題として定式化する．入力データは，文字列の有限集合である例集合 (sample) $S = \{s_1, \ldots, s_m\}$ とラベルづけ関数 (objective condition function) $\xi : S \to \{0, 1\}$ として与えられる．関数値 $\xi(s_i)$ は，文書 s_i の正負の分類値 (label) を表す．パタン π と文書 s_i に対して，$\pi(s) = \xi(s)$ のとき π は s を**正しく分類する** (π agree with s_i) という．

(d, k)-語相関パタンによる分類精度最大化問題 (Maximizing Agreement by d-Words k-Proximity Association)

入力: アルファベット Σ および例集合 $S \subseteq \Sigma$, ラベルづけ関数 $\xi : S \to \{0, 1\}$, 非負整数 d, k.

問題: S に関する分類精度

$$\sum_{s \in S} |\pi(s) - \xi(s)|$$

を最大化する (d, k)-語相関パタン π を見つけよ．

この最適化問題は，統計的決定理論で知られている実効リスク最小化に他ならない．頑健な確率学習 (Agnositic PAC-learning) に関する最近の研究からは，この問題を効率よく解くアルゴリズムは，データに雑音やエラーが含まれる場合にも，分類例を生成した未知の規則をきわめてうまく近似できることがわかっている [4].

4 ランダムテキストに対する高速アルゴリズム

(d,k)-語相関パタンによる分類精度最大化問題は，すべての (d,k)-語相関パタンを探索する自明な方法を用いて，$O(n^{2d+1})$ 時間でとける．しかし，本研究では平均的により高速なアルゴリズムを開発する．

4.1 接尾語木

アルゴリズムの主要なデータ構造である接尾語木について説明する．テキスト $t = a_1 \cdots a_{m-1}\$$ に対して，位置 p からはじまる t の接尾語を $t_p = a_1 \cdots a_{m-1}\$$ で表す．テキスト t の **接尾語木** (suffix tree) T_t とは，t の空でない接尾語全体 $\{t_1, \ldots, t_n\}$ を表す圧縮トライ (compacted trie) である．ここで，圧縮トライとは，通常のトライ (trie) から，子を一つしかもたない内部節点を取り除き，辺のラベルを合併することを繰り返して得られる木である．

接尾語木は，$O(n)$ 時間で計算可能であり，$O(n)$ 領域を使用する (McCreight [6]). $Word(v)$ で，根から節点 v にいたるパス上のラベルを連結して得られる語を表す．以下では，内部節点 v に対して $s = Word(v)$ と表される文字列 s を **分岐語** (branching word) という．

4.2 アルゴリズム

図 1 に，最適パタンを計算するアルゴリズムをしめす．簡便のため，以下ではパタン πに照合する S 中の正ラベル付の例数 $count_1$ と 負ラベル付の例数 $count_0$の差 $\Delta_{S,\xi}(\pi)$ を最大化する問題を考える．これは，明らかに元の問題と等価である．

まず入力データ $S = \{s_1, \ldots, s_m\}$ と $\xi : S \to \{0,1\}$ を受け取ると，アルゴリズムは S の例すべてを連結して，一本のテキスト $t = s_1\$s_2\$\cdots s_m\$$ を構成する $(n = |A|)$．ここに，\$ は，$\$ \notin \Sigma$ かつ $\$ \neq \$$ をみたす互いに異なる区切り文字である．各位置 $1 \le p \le n$ に対して，p が文書 s_i 中の位置ならば $\delta(p) = i$ と定義する．

このアルゴリズムは，テキスト t の接尾語木 T_t を作り，これを用いて文書の部分語を管理する．T_t の葉は t の接尾語を表しており，左から右へそれが表す接尾語の辞書式順序に並んでいる．これらの接尾語 t_p の開始位置 p をこの辞書式順序に従っておさめた配列を $suf[1..n]$ とし，その逆関数を表す配列を $pos[1..n]$ とする．

接尾語木 T_t の節点の列 $u_1, \ldots, u_d$ に対して，

$$\pi = (Word(u_1) \ldots, Word(u_d); k)$$

の形で表されるパタンを，**正規形パタン**という．ここに，節点 u_j が葉のときは，$Word(u_j)$ は，$Word(u_j)$ の\$ を含まない最長の接頭語であると再定義しておく．$\perp_S \in \Sigma^*$ を，S 中のどの文書よりも真に長い任意の文字列とする．

補題 1. 正規形のパタンまたは $\perp_S$ が，$\Delta_{S,\xi}(\pi)$ を (d,k) 語相関パタンに対して最大化する．

補題 1 から，正規形のパタンだけに注目すれば，最適パタンを発見できることがわかる．しかし，単純に接尾語木を巡回しながら正規形のパタンを枚挙し，テストするだけでは，平均時および最悪時に $O(n^{d+1})$ 時間の計算を要する．

そこで，平均時間の意味でより高速に働くアルゴリズムを与えるため，本アルゴリズムでは，鍵となるアイディアとして，d 語相関パタンの探索をテキストの接尾語を辞書式順序にならべた空間における d 次元直方体の探索に帰着する．

例集合 S に対して，**幅 k の d 次元対角要素集合** $Diag_{d,k,S} \subseteq [1..n]^d \times [1..m]$ とは，任意の $1 \le j \le d$ に対して，$0 \le i_{j+1} - i_j \le k$ かつ $\delta(i_j) = \delta$が成立するような，ラベル付けされた点 $(i_1, \ldots, i_d; \delta)$ 全体のなす集合である．ここに，$i_0 = 0, i_{d+1} = n$ とする．

さらに，**d次元ランク集合** $Rank_{d,k,S} \subseteq [1..n]^d \times [1..m]$ を，ある $(i_d, \ldots, i_1; \delta) \in Diag_{d,k,S}$ に対して，任意の $x_j = pos(i_j)$ について $1 \le j \le d$ をみたすようなラベル付けされた点 $(x_d, \ldots, x_1; \delta)$ 全体のなす集合と定義する．集合 $Diag_{d,k,S}, Rank_{d,k,S}$ の要素数は共に $N = k^{d-1}n$ である．

接尾語木の節点 v に対して，語 $Word(v)$ の出現位置全体はランク空間 $[1..n]$ 上の連続した

Algorithm. *Find_Best_Pattern*:
Input: An alphabet Σ, an integer $k \geq 0$, a sample $S \subseteq \Sigma^*$ and an objective condition $\xi : S \to \{0,1\}$.
Output: A (d,k)-proximity pattern π over Σ that maximizes $\Delta_{S,\xi}(\pi)$.

1 Let $A = s_1\$\cdots\$s_m\$$ be the input text associated with S. Compute the suffix tree $Tree_A$ for A, the suffix arrays suf and $pos = suf^{-1}$.

2 Let $Diag_{d,k,S}$ be the d-dimensional diagonal set of width k. Then, compute $Rank_{d,k,S} = \{ (x_d,\ldots,x_1;\delta) \mid x_j = pos(i_j)$ for every $1 \leq j \leq d$ and $(i_d,\ldots,i_1;\delta) \in Diag_{d,k,S} \}$ by transforming the labeled points in $Diag_{d,k,S}$ from position space to the rank space.

3 Call $Discover(d, Rank_{d,k,S}, \rho_d)$ with the empty pattern $\rho_d := (;k)$.

4 Return the best patterns π found if the maximum of Δ is positive. Otherwise, return $\perp_S$.

Figure 1: The algorithm for computing the maximum agreement over (d,k)-proximity patterns

部分区間 $I(v) = [x_1..x_2]$ を占める．そこで，正規形の (d,k) 語相関パタン

$$\pi = (Word(v_1),\ldots,Word(v_d);k)$$

に対して，ランク空間上の d 次元直方体を

$$Box(\pi) = I(v_1) \times \cdots \times I(v_d)$$

とし，その分類誤差を $\Delta_{Q,\xi}(B(\pi)) = count'_1 - count'_0$ と定義する．ここに，$\alpha \in \{0,1\}$ に対して，$count'_\alpha$ は点 $(x_1,\ldots,x_d;i) \in Q \cap Box(\pi)$ で $\xi(i) = \alpha$ をみたすものの数を表す．

補題 2. 任意の正規形 (d,k) 語相関パタン π に対して，$\Delta_{S,\xi}(\pi) = \Delta_{Q,\xi}(Box(\pi))$ が成立する．

図 2 に，$\Delta_{Q,\xi}(Box(\pi))$ を計算する再帰手続き *Discover* を示す．この手続きは，動的計画法を用いて接尾語木上で直交領域質問を行いながら，Δ を最大化する直方体を高速に計算する．さらに，平均計算量を下げる工夫として，分割統治法を用いて正の分類値を与える直方体だけに探索を制限する．これにより，$O(n^d)$ 個ある可能な d 個の部分語の組み合わせ全体から，一部のパタンだけを探索することが可能になる．

定理 3. 入力 (Σ,S,ξ,d,k) に対して，図 1 のアルゴリズム *Find_Best_Pattern* は，分類誤差を最大化する正規形 (d,k) 語相関パタンすべてを，最悪時でも $O(k^{d-1}h^d n\log n)$ 時間と $O(k^{d-1}n)$ 領域で計算する．ここに n は S の文書のサイズの総和であり，$h \leq n$ は t の接尾語木の高さ（葉までの最長パスの長さ）である．

接尾語木の高さ h は，一般のデータに対してそれほど大きくないことが経験的に知られている．実際，一様分布によるランダムテキストに対しては，接尾語木の高さがテキスト長 n の対数程度になる (Devroye 1992). これは，ゲノムデータに対してもよくあてはまる．

系 4. 一様分布によって生成されたランダムテキストの集合を S とする．このとき，図 1 のアルゴリズム *Find_Best_Pattern* は平均実行時間 $O(dk^{d-1}n\log^{d+1}n)$ と領域 $O(k^{d-1}n)$ で動く．ここに n は S の文書のサイズの総和である．

Proof. $\Sigma\cup\{\$\}$ 上の長さ n のランダムテキストを t とおき，H_n を t の接尾語木の高さを与える確率変数とする．Devroye *et al.* [3] は，任意の $h \geq 0$ に対して $H_n \leq h$ の確率の上限を以下のように与えた:

$$\mathbf{P}(H_n \leq h) \leq 2n(q^{h+1}/(1-q) + nq^{2h})$$

ここに $q = (1/b)^{1/2}$ である．このとき，平均実行時間は，$\mathbf{E}(T(n)) = \sum_A T(n,h)\cdot P(H_n = h)$ となる．ここに，和は長さ n のテキスト全体でとるとし，$T(n,h) = dk^{d-1}h^d n\log n)$ である．この和を，$H_n \leq 2\log_b n$ と $H_n > 2\log_b n$ の二つの場合に分けて計算すると，平均実行時間 $\mathbf{E}(T(n)) = O(dk^{d-1}n\log^{d+1}n)$ を得る． □

Procedure. *Discover*(d, Q, ρ_d):

1. The case that $d \geq 1$:
   (a) Compute the set X_{coord} of the first coordinates of the points in Q and sort it.
   Let $H := \emptyset$ and $H_{\mathrm{new}} := \emptyset$.
   (b) For each coordinate $x \in X_{\mathrm{coord}}$ do:
   Attach the set $Q_{\lambda_x} = \{ (x_{d-1}, \ldots, x_1; \delta) \mid (x_d, x_{d-1}, \ldots, x_1; \delta) \in R_d \}$ to the leaf λ_x of rank x, and then append λ_x to the right end of H.
   (c) For each level $l = 0, 1, \ldots, h$ do:
   (i) Scanning H from left to right, for each $v \in H$ do:
   Let w be the parent of v. Then, update Q_w by $Q_w := Q_w \cup Q_v$ in sorted order in the $(d-1)$st coordinate (in δ if $d = 1$). If v is the leftmost child of w then append w to the right end of H_{new}.
   If $l \geq 0$, then recursively call *Discover*$(d-1, Q_v, \rho_{d-1})$ with $\rho_{d-1} := \rho_d \cdot (Word(v); k)$.
   (ii) Discard Q_v for all nodes $v \in H$. Let $H := H_{\mathrm{new}}$ and $H_{\mathrm{new}} := \emptyset$.
2. The case that $d = 0$:
   Compute $count_\alpha := |\{\delta \in Q \mid \xi(\delta) = \alpha\}|$ for each $\alpha \in \{0, 1\}$. If $\Delta := count_1 - count_0$ is larger than the largest value seen so far, record the pattern π_0.

Figure 2: The procedure *Discover*

5 接尾語配列を用いた実現法

接尾語配列 (suffix array) は，接尾語木の葉だけを 1 次元配列に格納したデータ構造である．前節で用いた配列 suf が接尾語配列である．接尾語配列は，接尾語木の 1/2〜1/3 以下の記憶容量をもち，2 次記憶での実装も容易なため，現在，多くの大規模テキストデータベースが接尾語配列を用いて実装されている．そこで，接尾語木を用いるデータマイニング手続きを，そのまま接尾語配列上に効率よく実装する方法を考察する．

この際の問題点は，接尾語配列が，元の接尾語木がもっていた木構造を失っていることである．ここではまず，接尾語木の節点の巡回や，その上での動的計画法の適用といった接尾語木に対する基本的な計算を，部分語統計問題として定式化し，接尾語配列を用いた高速な計算方法を与える．

本節の方法は，語相関パタンの発見アルゴリズムを実装するのに直接適用できるわけではないが，これを基礎としてより一般的な方法の開発を目指している．

5.1 部分語統計問題

テキストのすべての部分語の出現回数を数える問題を一般化して，部分語統計問題を定義する．長さ n のテキストを $A = a_1 \cdots a_{n-1}\$$ とする．ここに，\$ はどの文字ともことなる区切り記号である．このとき，部分語統計は 3 つ組 $(D, \oplus, B)$ で指定される:

- 値の集合 D.
- 結合的な 2 項演算 $\oplus : D \times D \to D$. D は，任意の $e \in D$ に対して $\bot = \bot \oplus e = e \oplus \bot$ となる空要素 $\bot$をもつ.
- A 中のすべての位置 $1 \leq p \leq n$ に対する初期割り当て $B : [1..n] \to D$.

部分語 αに対して，α の A 中の出現位置すべてを $p_1, \ldots, p_r$ とおき，部分語統計を $C(\alpha) = B(p_1) \oplus \cdots \oplus B(p_r)$ と定義する．

定義 2. 部分語統計問題 (subword statistics problem) とは，テキスト A と割り当て B が与えられたとき，Aのすべての部分語 αに対して $C(\alpha)$ を計算する問題である．

たとえば，部分語の出現数を求める問題は，D を自然数の集合とし，任意の位置 p に対して $B(p) = 1$，演算 $\oplus$ を自然数の加算と定義して得られる．また，分岐語 $Word(v)$ に対応する区間 $I(v) = (L, R)$ を枚挙する問題は，$D = \{(L, R) \mid 1 \leq L \leq R \leq n\}$ とし，$B(p) = (p, p)$, $(L, R) \oplus_1 (L', R') = (L, R')$ と定義して得られる．

5.2 単純なアルゴリズム

部分語統計は，接尾語木の深さ優先探索を用いてつぎのように計算できる．

1. 葉を左から右に走査しながら，i 番目の葉 v に対して $(i := 1, \ldots, n)$，リスト $C_v := B(Pos(i))$ を対応づける．
2. 接尾語木を葉から根へと走査しながら，各内部節点 v に対して，つぎを実行する：v に対して，リスト $C_v := C_{v_1} \oplus \cdots \oplus C_{v_m}$ を対応づける．ここに，節点 v の子供を $v_1, \ldots, v_m$ とする．

このアルゴリズムを接尾語配列上で模倣するのに，各節点 v を区間 $I(v) = [L..R]$ で表現し，節点 v の子供 $v_1, \ldots, v_m$ を見つけるのに接尾語配列上での2分探索を用いることができる．木の走査には，スタックを用いる．しかし，この方法は，テキスト A の圧縮しない接尾語木 $\bar{T}_A$ の走査に対応しており，計算時間は $O(n \log n + Q) + M$ となる．ここに，Q は $\bar{T}_A$ の節点数 $Q = O(n^2)$ であり，M は $\oplus$ 演算の所要時間の合計である．

5.3 より高速なアルゴリズム

節点での2分探索を用いない，より高速なアルゴリズムを以下の図に示す．このアルゴリズムは，接尾語配列を左から右に一度走査しながら，スタックを用いて $O(n) + M$ 時間ですべての部分語統計を計算する．

ここで $Hgt[1..n]$ は，任意の $1 \leq i \leq n$ に対して $A_{Pos[i]}, A_{Pos[i+1]})$ の最長接頭語の長さを $Hgt(i)$ の値とする $Hgt(i)$ 長さ n の配列である．ただし，$Hgt(n) = -1$ と定義する．Hgt は接尾語配列から線形時間で計算できる．

対応する接尾語木の高さ h に対して，使用する領域は接尾語配列に $O(n)$，スタックに $O(h)$ である．実際のテキストデータベースでは $h \leq 128$ 程度と考えられているので，一度の走査しかしない接尾語配列は2次記憶に，スタックは主記憶に格納する記憶管理法が考えられる．

6 おわりに

本稿では，テキストデータマイニングのためのパタン発見アルゴリズムの高速化について論じた．はじめに，遺伝子配列データベースのようなランダムなテキストに対して，平均時に高速に働くアルゴリズムを与えた．さらに，接尾語配列上での実装法についても論じた．

謝辞: 本研究の一部は特定領域研究「高度データベース」による．

Algorithm Subword_Statistics

1. Compute $Hgt[1..n]$ and $S := \emptyset$.
   Push $(\perp, -1)$ into S.
2. For each $i = 1, \ldots, n-1$, do:
   (a) $C_{\text{new}} := B(i)$ and $H_{\text{new}} := Hgt(i)$.
   Let (C, H) be the top of S.
   (b) While $H_{\text{new}} < H$, do:
   (i) Report $(C \oplus C_{\text{new}}, H)$.
   (ii) $C_{\text{new}} := C \oplus C_{\text{new}}$. Then, pop S.
   Let (C, H) be the top of S.
   (c) If $H_{\text{new}} = H$ then
   Pop (C, H) from S, and then push $(C \oplus C_{\text{new}}, H)$ into S.
   (d) Else if $H_{\text{new}} > H$ then
   Push $(C_{\text{new}}, H_{\text{new}})$ into S.

Figure 3: An algorithm for computing string statistics using the suffix array

References

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases. Proc. 1993 SIGMOD, 207–216 (1993).

[2] H. Arimura, A. Wataki, R. Fujino, S. Arikawa, A Fast Algorithm for Discovering Optimal String Patterns in Large Text Databases. (To appear in Proc. 9th ALT, LNAI, 1998)

[3] L. Devroye, W. Szpankowski, B. Rais, A note on the height of the suffix trees. *SIAM J. Comput.*, 21, 48–53 (1992).

[4] M. J. Kearns, R. E. Shapire, L. M. Sellie, Toward efficient agnostic learning. *Machine Learning*, 17, 115–141, (1994).

[5] U. Manber, R. Baeza-Yates, An algorithm for string matching with a sequence of don't cares. *IPL*, 37, 133–136 (1991).

[6] E. M. McCreight, A space-echonomical suffixtree construction algorithm. *J. ACM*, 23, 262–272 (1976).