社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2003－FI－ 72　(4)
2003－NL－157　(4)
2003／9／29

文字列の頻度分布による共通パタン発見

池 田 大 輔 †† 山 田 泰 寛 † 廣川 佐千男††

パタンを定数と変数からなる文字列とする。パタン中の変数を定数文字列で置きかえて得られる文字列をそのパタンから生成される語とする。本稿では、未知のパタンから生成された語の有限集合が与えられた時に、そのパタンの定数部分を見つける問題 (テンプレート発見問題) を考察する。未知パタンの定数部分が適当な長さを持ち、変数へ代入される定数文字列が自然な確率分布に従っているならば、パタンから生成される語において、定数部分と変数に代入された文字列の部分文字列の出現頻度の差を利用してテンプレートを効率よく発見できることを示す。さらに、Web 上の HTML ファイルでの予備的な実験結果を紹介する。

Pattern Discovery from Distributions of String Frequency

Daisuke Ikeda †† ,Yasuhiro Yamada † and Sachio Hirokawa††

A pattern is a string over constant and variable symbols. A string generated by a pattern is one obtained by replacing all variables by some constant strings. In this paper, we consider the template discovery problem which is, given a set of strings generated by some fixed but unknown pattern, to find all constant parts of the pattern. If any constant part is long enough and replacing variables follows some natural probabilistic distributions, we show that there exist an efficient algorithm for the problem, using disparity of string frequencies among constant parts and replaced parts. We also show accuracy and effectiveness by experiments by using HTML files collected from the Web.

1. はじめに

複数の対象が与えられたときに、これらの多くに共通する性質あるいは頻出する部分を見つける問題を解くことは、非常に一般的であり、様々な分野で研究されてきた。本稿では、文字列を対象にして、これらに共通なパタンを探す問題を考える。

1.1 関連研究

文字列から共通・頻出パタンを探す問題は古くから文字列処理の分野で研究されてきた。例えば、最長共通部分文字列や最長共通部分列問題などが共通パタン発見問題として、また、最大反復などが頻出パタン発見問題として、理論と実践の両面から詳しく研究されてきた。これらの研究から、パタンの形式が単純な最長共通部分文字列問題 [7] や最大反復 [11] 等は高速に解けることが示されたが、パタンが部分文字列の列となる最長共通部分列問題となると NP 完全になることが知られている。

ゲノム情報処理では、進化の過程や生物学的な機能を DNA 配列等から推測するために、共通するパタンを発見するアプローチが重要な役割を果たしている。モチーフ発見 [16]、最長共通部分列問題やその拡張 [5),8]、ローカル及びグローバルアラインメント、最大反復 [6),12] などを解くアルゴリズムが、実際に配列データに応用され、成果をあげてきた。しかし、パタンの形式は複雑であり、ペアワイズアラインメントのように入力が 2 本に固定されているような場合はともかく、大規模な配列への適用は困難である。

自然言語処理における定型表現の抽出も多くの文書に共通するフレーズの発見と考えられる。機械学習における正例からのパタン言語の学習 [2] も、与えられた例に共通する性質やパタンの探索といえる。圧縮やキャッシュアルゴリズムは、オンラインで頻出するパタンを予測するアルゴリズムである。

最近ではデータマイニングの分野において、共通・頻出パタンの発見手法が多く提案されている [1),3),4]。データマイニングやテキストマイニングにおいては、共通するパタンを見つけることが大きな目標の一つといってよい。Web マイニングの分野において情報抽出が盛んに研究されているが [13),17),18]、これも共通部分をラッパーとして取り出す問題と考えることができる。

このように共通・頻出パタン発見問題は、非常に一般

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

的で応用範囲が広いことが分かる。しかし、この問題は、探すパタンの形式にも依存するが、明確に定義されるパタンを探す場合は、一般に計算量が高いことが明らかになっている。例えば、最長共通部分列は NP 完全であり、パタン言語の学習も変数の出現等を制限しないと現実的に解けないことが知られている [2),10),14)]。

1.2 頻度によるパタン発見

一方、探す対象を明確に定義しにくい場合等に、ヒューリスティクスを用いて、高速に対象を探す手法も提案されている。このような対象として、例えば、自然言語処理における未知語や定型表現の抽出がある。未知語や定型表現そのものは明確に定義しにくいが、おおまかな傾向としては、ある程度の長さでまとまって度々使用されるものといえる。このことを利用して、意味的な解析を行なわず字面処理だけでフレーズや定型表現を抽出する試みがなされている [20),21),23)]。長尾と森は、長さ n の特定の文字列の頻度が高く、かつ、その文字列の左や右に現われる文字の種類が多い場合は、その特定の文字列がひとかたまりで使われているであろうことに着目した [23)]。新納と井佐原は、長尾らのアイデアに助詞的定型表現が持つ特有の性質やヒューリスティクスを加え、助詞的な定型表現を抽出した [21)]。下畑らも、同様に、定型表現とその前後の文字の出現頻度の差を利用し、定型表現を抽出した [20)]。

このように完全に字面処理だけで意味のある語句や定型表現を求める場合、基本的に出現回数が多いものを抽出する。しかし、それだけでは長さ 1 の単なる文字が最も頻度が高くなるので、抽出すべき定型部分とその前後の文字を加えた文字列の頻度の差などを利用している。これに対し、対象が Web 上の半構造化データと棋譜データではあるが、定型表現や意味のある語句はある程度の長さを持つと仮定し、部分文字列の頻度のみで抽出を試みた研究もある [9),22)]。

中村は、囲碁の棋譜データを符号化された文字列とみなし、また定石と呼ばれる定型的に用いられる普遍的な手順を定型表現とみなし、文字列の出現頻度を用いて囲碁の棋譜データから定石を発見する手法を提案した [22)]。この手法は、定石が (1) 多くの棋譜データによく現われること、(2) 単位性のある一連の手順から構成されること、(3) ある程度の長さを持つことに着目したいる。そして、棋譜データを文字列データに符号化し、定石の発見問題を定型的な部分文字列の発見に置きかえた。その上で、長さを適当に固定し、その長さの部分文字列における頻度をプロットした。定石はある程度の長さをもつので、定石に含まれる部分文字列の頻度は高いが、定石以外での頻度は低くなるので、定石がうまく発見できる。

ただし、この手法では長さを適当に定める必要がある。実際、文献 [22)] では、長さを変化させると、定石部分がみつかりにくくなることもある。そのため、本質的な解決策ではないが、複数の長さをあらかじめ集合として定められるようにし、一つの長さではなく、これら複数の長さに対する頻度の平均で定石を発見している。

筆者らは、自動的に長さを決めるために、交代数という新たな計数を導入し、共通部分を特定するアルゴリズムを提案し、Web 上のデータに対してその有効性を確認した [9)]。このアルゴリズムは、自動的に整数のペア (n, a) を決定する。このペアを用いて、長さ n のすべての部分文字列で、頻度の上位 a パーセントに入るものが高頻度であると定義される。ある (n, a) における交代数とは、高頻度とそうでない部分との境界線の数である。

(n, a) を決めるアルゴリズムは $(1, 1)$ から始めて、現在のペアにおける交代数と、n または a のどちらかの値を 1 増やした時の交代数とを比較する。現在のペアの交代数が最小であれば停止し、このペアを出力する。そうでなければ、より小さい交代数となるペアのほうへ遷移し、また比較を繰り返す。

このアルゴリズムは自動的に長さを決めるが、問題点も含んでいる。100 刻みのパーセントで文字列を分割するが、何故 100 刻みなのかの説明を与えない。また、一度 n または a の値が増えると、アルゴリズムは以後値を減らすことをしないから、初期の段階で最適な値を “飛び越え” てしまうことがある。この場合、最適な値を見つけることができない。他にも、遷移の際に (n, a), $(n + 1, a)$, $(n, a + 1)$ における交代数を比較するが、最小なものを選択する妥当性についても、実験的にうまくいくことが多いというだけである。

1.3 頻度分布の差による共通パタン発見

筆者らは、共通パタンを持つ文書と持たない文書を対象に、すべての部分文字列の頻度を数え、頻度 f とちょうど f 回出現する部分文字列の数 $V(f)$[☆] との関係を調べた [19)]。これによると、共通パタンを持たない小説 (日本語と英語) では、$\log V(f)$ と $\log f$ は線形の関係となり、ジップの (第 2) 法則を満すことが分かった。数えたのは単語ではなく、“すべての” 部分文字列[☆☆] であることに注意する。一方、共通パタンを持つものは、共通パタン中の部分文字列の頻度が特異的に高いことから、ジップの法則に従わないことが分かった。

頻度 f と頻度の頻度 $V(f)$ の関係により、共通パタンを持つか持たないかの判定はできるが、具体的にパタンを抽出することはできなかった。そこで、本稿では、まず共通パタン発見として問題を定義し、頻度や頻度の頻度との関係を考察する。

問題は、複数の文字列が与えられたときに、これを適当なテンプレートから生成された文字列とみなし、テン

☆ “頻度の頻度” とも呼ばれる。

☆☆ 実際には長さ 50 までのすべての部分文字列である。

プレート部分を見つけることである。これは、パタン言語 [2] の正例からの学習に似ているが、変数部分の対応は無視しているので、各変数の出現が高々1 回に制限される正則パタン [15] の学習と同じである。

しかし、正則パタンに制限しても、現実的な時間内に学習することは難しいことが知られている。そこで、本稿では、テンプレート部分 (パタンの定数文字列部分) とそうでない部分 (変数に代入された文字列部分) の頻度の差を利用し、テンプレートを発見する手法を提案する。

2. パタンとテンプレート発見問題

この節では、パタンとパタンで定義される言語を定義し、テンプレート発見問題を定式化する。テンプレートとは、直感的に言えば、複数の文字列に共通する部分文字列の列である。また、テンプレート発見問題とパタン言語の学習の関係についても述べる。

2.1 パタン言語

Σ を有限アルファベットとする。Σ の要素の列 $a_1a_2\ldots a_n$ $(a_i \in \Sigma)$ を (Σ 上の) **文字列**と呼ぶ。Σ 上の文字列の集合を Σ^* で表す。文字列 $w \in \Sigma^*$ に対し、$tuv = w$ なる文字列 t, u, v が存在するとき u を w の**部分文字列**と呼ぶ。

$V = \{x_1, x_2, \ldots,\}$ を Σ と交わらない集合とする。V の要素を**変数**と呼ぶ。V との対比から、Σ 上の文字列を**定数**文字列とも呼ぶ。

$\Sigma \cup V$ 上の文字列を**パタン**と呼ぶ。各変数が高々1 回しか出現しないパタンを**正則**パタンという。

$w_i \in \Sigma^*$, $X_i \in V^*$ $(1 \leq i \leq m)$ からなるパタン $p = w_1X_1 \cdots w_mX_m$ が与えられたとき、文字列の列 $(w_1, \ldots, w_m)$ を**テンプレート**と呼び、$t(p)$ で表す。テンプレートの**幅**をテンプレート中で最も短い文字列の長さと定義し、$|t(p)|$ で表す。つまり、$|t(p)| = \min\{|w_i| \mid w_i \in t(p)\}$ である。

V から Σ^* への写像を**代入**と呼ぶ。代入 θ が変数 x へ w を代入することを $[x := w]$ と書き、複数の代入を一度に $[x_1 := w_1, \ldots, x_n := w_n]$ と表す。

パタン p と代入 $\theta = [x_1 := w_1, \ldots, x_n := w_n]$ に対し、$p\theta$ により、パタン中の変数 x_i を w_i で置きかえて得られる文字列を表す。パタン p の**言語**とは、文字列の集合 $\{w \in \Sigma^* \mid \exists\theta; p\theta = w\}$ のことであり、$L(p)$ で表す。

文字列の有限集合 $S \subset \Sigma^*$ を**サンプル**と呼ぶ。サンプル S が与えられたとき、$S \subseteq L(p)$ かつ p とは異なるパタン p' で $L(p') \subset L(p)$ なるものが存在しないとき、パタン p は S に対し**極小**であるという。

ここでテンプレート発見問題を以下のように定式化する。

定義 1 (Angluin[2], Shinohara[15]) テンプレート発見問題とは、与えられたサンプルに対し極小であるような正則パタン p を見つける問題である。

テンプレート発見では定数部分のみを見つければ十分である。しかし、正則パタンの場合、変数は高々1 回しか出現しないので、テンプレート $(w_1, \ldots, w_m)$ の間に適当な変数を挿入すれば正則パタンになる。逆に、パタンが見つかれば、アルファベット Σ 以外の文字 (つまり、変数) を除けば、テンプレートが得られる。つまり、変数部分は無視されるので、特に正則にパタンに制限しなくてもよい。

2.2 テンプレート発見問題

前節の議論により、テンプレート発見問題は正則パタンの学習と考えることができる。パタン言語の学習は 1980 年代以降によく研究されてきた。その結果によると、パタンを正則に限ったとしても、現実的な時間内の学習は容易ではないことが知られている。そこで本節では、現実的に許容できる仮定を加え、テンプレート発見問題の新たな解法を考える。

共通なパタンを見つけようとする時は、大量の文字列データから規則性をテンプレートという形式で探しだそうとしている。ここで、一般に以下の 2 つのことが暗黙のうちに仮定されている。(1) 文字列はテンプレートとそうでない部分からなる、(2) テンプレートでない部分は、規則性がないので、ランダムであったり、適当に “自然” な文字列からなる。

(1) の仮定は当然であり、すべてがテンプレートであれば、問題は非常に易しい。一方、実際のデータを考えた場合、(2) の仮定も当然と言ってよいだろう。例えば、Web マイニングの一つである情報抽出 (ラッパー生成問題) をテンプレート抽出と考えた場合、テンプレートでない部分は通常自然言語で記述された文章などのコンテンツである。コンテンツ内には不自然なパタンは現われにくいであろう。他に、例えば、DNA 配列などで考えた場合も、生物学的な機能を司る部分や意味のある部分をテンプレートとした場合、それ以外の配列には規則性は見つけにくいだろう。

そこで、パタンの変数へ代入する定数文字列に適当な確率分布を仮定する。こうすることにより、テンプレート以外の部分のランダムさや自然さが表現可能となる。また、確率を考えるから、テンプレート以外の部分には、規則性が現われにくいだけで、現われる確率は 0 ではないと考えられる。

PAC 学習では、例を与えるための確率分布が導入されている。しかし、PAC 学習では任意の例の与え方 (確率分布) に対して、アルゴリズムは学習できなければならない。一方、本稿では特定の確率分布を仮定するので、制限がきびしいようにも思える。しかし、テンプレート以外の部分に、自然言語のコンテンツを埋め込むことを考えると、自然言語らしい確率分布を考えることは妥当で

ある。さらに、自然言語の場合は、単語と頻度に関して、例えば、ジップの法則 (Zipf's law) などが経験的な分布として知られている。

ここで仮定する確率分布がどのような性質を満さなければならないかは、今後の課題である。しかし、テンプレート部分の文字列は常に同じであり、そうでない部分は自然に変化しているものを考える。ここで、確率分布が**自然**であるとは、短い文字列は適当な確率で現われるが、長い文字列は現われにくいものとしておく。

テンプレート部分はサンプルの全ての文字列に共通して現われる。しかし、テンプレート部分が短かければ、テンプレート以外の部分にたまたまその文字列が頻出することも考えられる。逆に、テンプレートが十分に長ければ、テンプレート以外には、このような長い文字列は現われないので、その頻度は特徴的であろう。そこで、各サンプル文字列に埋め込まれているテンプレートも適当な幅を持っていると仮定する。

以上の議論から、テンプレート発見問題を以下のように定義しなおす。

定義 2 (テンプレート発見問題) テンプレートの幅が k である未知のパタン $p = w_1X_1 \cdots w_mX_m$ から自然な確率分布 $D(X_1, \ldots, X_m)$ に従って生成されたサンプル S が与えられたとき、テンプレート発見問題とは、サンプルに対し極小であるようなパタン p のテンプレート $t = (w_1, \ldots, w_m)$ を探す問題である。
確率分布そのものは与えられていないことに注意する。

3. ジップの法則とパタン発見手法

サンプル S が与えられた時の部分文字列とその頻度との関係を考えてみる。f を頻度 (出現回数) を表す正の整数とする。S 中の (適当な長さ、または適当な基準を満す) 部分文字列の頻度を数え、ちょうど f 回出現した異なる部分文字列の数を $V(f)$ で表し、$F(f)$ でちょうど f 回出現した部分文字列の重複を含めた数を表すことにする。つまり $F(f) = f * V(f)$ である。

S を英文テキストとし、部分文字列を数える単位として英単語を選択すると、頻度 f と $V(f)$ の間にはジップの法則 (Zipf's law) が成立することが知られている。これは、適当な定数 $a > 0$ と b が存在して

$$\log V(f) = -a \log f + b \qquad (1)$$

が成立する、というもので、特に低頻度の単語に対してよく成立することが知られている。対数をとっているが、基本的には高頻度 (f が大きい) になればなるほど、それだけ出現する部分文字列の種類 (つまり $V(f)$ の値) は小さくなる、ことを主張している。

筆者らは、すでに、共通なパタンを持つ文書とそうでない文書について、f と $V(f)$ の関係を調べた [19]。この時、数えあげる部分文字列は単語ではなく、全ての部分文字列である。これにより、単語や特定の適当な長さといった前提知識は不要なり、様々な分野への応用が可能になると考えられる。

全ての部分文字列の頻度に対しても、共通パタンを持たない文書 (実際には日本語や英語の小説) に対しては、うまく式 (1) のジップの法則が成立し、一方、共通パタンがある文書に対しては、共通パタンの頻度が特異的に高いため、ジップの法則の成立を妨げていることが示された [19]。

しかし、上述の f と $V(f)$ の関係だけでは、パタンの存在の有無は分かっても、どの部分文字列が共通パタンなのかということは未解決のままであった。そこで、本稿では $V(f)$ ではなく $F(f)$ を使うことで、共通パタンそのものを発見することを考える。

式 (1) を変形すると $\log(V(f)f^a) = b$ が得られる。つまり、適当な a に対し、$V(f)f^a$ は頻度 f にかかわらず定数となる。a はサンプル S の種類などによって変わると考えられるが、ここでの問題は、共通パタンがある場合とない場合に違いがでればよいので、a を求めることはせずに、$a = 1$ として扱うことにする。すると、$F(f) = fV(f)$ が $a = 1$ の場合の入力に依存して決まる定数を表していることになり、これを調べることによりサンプル S の性質を推測することになる。

実際、4 節で示すように、f を横軸に (対数で) とり、$F(f)$ を縦軸にグラフを描くと、共通パタンの現われる回数のところにピークが現われるし、また、このちょうど回数だけ現われる部分文字列が共通パタンそのものを構成していることも分かった。

そこで、共通パタン発見の新たな手法として、すべての部分文字列に対する頻度を数え、各頻度 f に対する f 回出現する文字列の総数 $F(f)$ をプロットする。このとき、ピークを形成する部分文字列を共通パタンの一部を成す部分文字列として出力する。

長さ n の入力文字列に対し、すべての部分文字列の数は最悪で $O(n^2)$ 存在する。しかし、適当な長さ以上の部分文字列の数が何度も頻出することは稀であるため、適当な数をあらかじめ決めておき、そこまでの長さの部分文字列の頻度を数えることにする。4 節では、長さ 30 までの部分文字列を数えている。

4. 実　　験

本節では、Web 上から取得したファイルを対象に、頻度 f と $F(f)$ のグラフをプロットし、単一のテンプレートを持つ入力、複数のテンプレートを持つ入力、テンプレートとは無関係なファイルも同時に与えられる場合などにも、提案手法がうまくテンプレート部分を見つけられること示す。

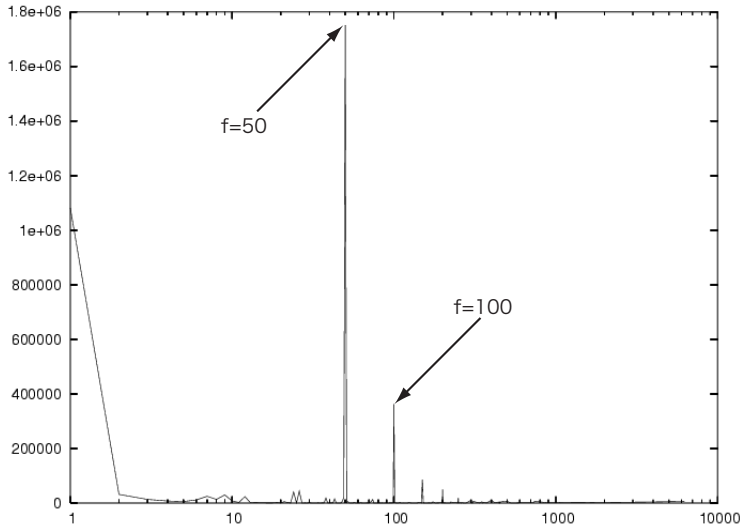

図 1 産経新聞の $F(f)$ グラフ

4.1 単一フォーマット

まず最初に一つのテンプレートから作成されたと考えられるファイルを対象に実験を行なった。入力は、産経新聞から取得した新聞記事の HTML ファイル 50 個であり、$F(f)$ のグラフは図 1 のようになった。

図 1 より、$f = 50, 100, 150, 200$ と 50 おきにピークがあることが分かる。また、最も高いピークは $f = 50$ の時である。これより、入力として与えた 50 ファイルの各ファイルには、テンプレートがあり、このテンプレート部分が 50 回出現していることが分かる。実際、ちょうど 50 回出現する部分文字列、つまり、最も高いピークを構成している文字列がどのように使われるか調べてみると、ほぼテンプレート部分と一致することが分かった。

これは単なるタグの部分をテンプレートとしているわけではなく、タグ以外の部分でもテンプレートとなり得る。産経新聞では、記事のカテゴリに応じて、title タグに “Sankei-itimen” や “Sankei-international” などと表示される。この中のハイフン (“-”) より左側はすべての記事に共通であり、右側が記事によって多少違う。そのため、“Sankei-” は頻出となったが、右側はそうではなかった。

$f = 50$ のピーク以外に、50 おきに小さくなりながらピークが出現しているのは、テンプレート部分に同じ文字列が含まれているからである。例えば、title タグの場合 “<title>” や “</title>” という文字列は、各ファイルに一度しか現われないが、“title” という文字列は各ファイルに 2 回づつ現われている。そのため、“title” はちょうど 100 回出現することになるが、このような文字列の種類は、50 回出現する文字列より少ないため、より小さなピークとなって現われている。

4.2 出現範囲が異なる複数フォーマット

次に、Yahoo!☆の検索結果を対象に実験を行なった。Yahoo!に限らず、検索結果のページが複数ある場合は、意味の異なる (少なくとも)2 種類のテンプレートが存在すると予測される。一つは各ファイルごとに共通のテンプレートであり、もう一つは各検索されたページごとのテンプレートであり、これは 1 ファイル中に複数回現われる。この実験は、このような入力に対して、どのようなグラフが得られるかを確かめるものである。

図 2 が Yahoo!の検索結果 46 ファイルを入力とした $F(f)$ のグラフである。検索結果を取得するために、検索語として Yahoo!のディレクトリのカテゴリ名を使った。すべてのカテゴリ名からランダムに 50 個抜きだし、一つのカテゴリ名を検索語として与え、検索結果の最初のページを取得した。ただし、検索結果のうち 4 つは取得に失敗したため、結果的に 46 ファイルが集まった。 各ファイルには、基本的に 20 件の検索結果が存在するが、14 件しかないものと 19 件しかないものが存在したため、検索結果の総数は 913 件であった。

図 2 から、ファイル数と同じ $f = 46$ の時に一番大きなピークが存在することが分かる。また、ちょうど 2 倍の $f = 92$ ではなく $f = 91$ の時にピークが見つかった。これは、ほぼすべてのファイルに同じ文字列が 2 回づつ出現したが、たまたま 1 つのファイルではそうではなかったためと予想した。しかし、共通パタンを成す文字列を確認すると、当初は予想しなかった別のテンプレートが存在することが分かった。Yahoo!の検索結果には、検索語にマッチした Web ページだけでなく、検索語にマッチした Yahoo!のカテゴリ名も表示される。このマッチしたカテゴリの総数が 91 であり、このテンプレートの部分文

☆ http://www.yahoo.com/

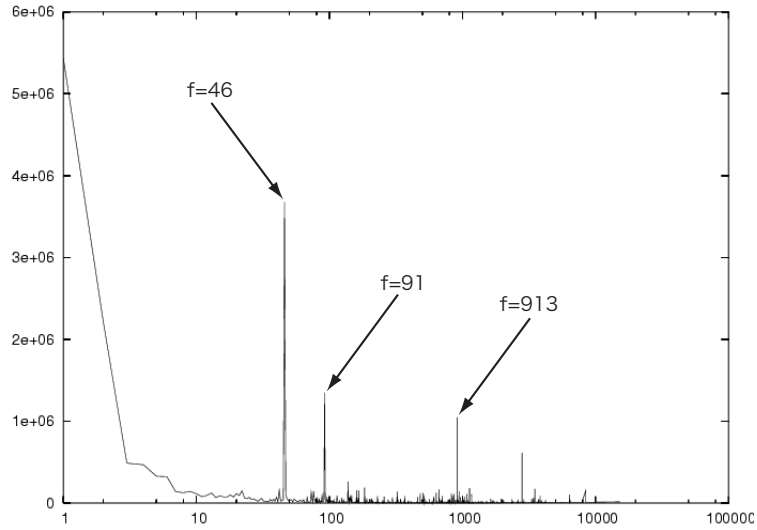

図 2 Yahoo!の検索結果の $F(f)$ グラフ

字列がピークを成していた。

さらに、図 2 には、検索結果の数と同じ $f = 913$ の時にもピークが存在する。このように複数のレンジの違うテンプレートが存在していても、すべてのテンプレートを見つけられることが分かる。

4.3 複数フォーマット

次には、3 つの異なるテンプレートから独立に生成されたファイルが混在する場合について実験を行なった。実験は、先の産経新聞 50 ファイルに加え、朝日新聞☆104 ファイル、読売新聞☆☆140 ファイルを同時に与えて行なった。産経新聞のファイル数に対し、朝日新聞のファイル数はおよそ 2 倍、読売新聞のそれはおよそ 3 倍になっていて、全体で見ると産経新聞のファイル数は 1/6 程度である。

これに対する $F(f)$ のグラフが図 3 である。 各新聞社のファイル数に応じて、3 つのピークが存在する。したがって、複数のテンプレートが混在していても、さらに、ファイル数が大きく異なっていても、すべてのテンプレートを見つけることができることが分かる。

図 3 では、予想していなかったピークも観測された。$f = 49$ と $f = 103$ のピークである。当初、これらはそれぞれ産経新聞と朝日新聞のファイル数 50 と 104 に近いため、これらの新聞社のテンプレートで、1 ファイルにだけ存在しないものが見つかったのかと予想した。

しかし、これらのピークを成す部分文字列を見てみると、例えば $f = 49$ のほうは、たまたま 49 回出現する文字列が朝日新聞と読売新聞に独立にあり、これらがある程度大きなピークを形成していた。つまり、たまたま朝日新聞と読売新聞で同じ回数ではあったが、それぞれに内容が異なる部分テンプレートを予期せず発見することができた。

4.4 ノイズを含んだデータ

ここまでの 3 つの実験では、あらかじめ適当なテンプレートが存在することが予測されるものを対象としたものであった。一方、最後の実験では、九州大学のトップページからリンクを 3 段階辿って取得したファイル 598 個を対象にした。

大学の Web ページは、それぞれの学部や学科、あるいは研究室や個人が独立に作っており、特に共通なテンプレートが存在しないことが多い。仮に、Web ページの制作単位で共通なテンプレートを使用したとしても、大学内のページ数からいうと少数にとどまり、大きなピークは見つからないだろうと予測していた。

図 4 が実験結果である。 図には、予想に反して明らかなピークが存在する。これは、いくつかの事務系の部署において、大学のトップページをコピーして使用しているところが 60 ファイルほどあったためである。テンプレートを持つファイル数は、入力ファイルの 1 割程度しかないが、それでも十分ピークが確認できる。

5. ま と め

本稿では、共通部分を持つ入力が与えられたときに、この共通部分を特定する問題をテンプレート発見問題として定式化した。このとき、与えられる入力はパタンの変数に対し自然な代入がなされるものと仮定する。具体的にどのような条件を満せば本稿で示したように明確なピークがでるのかを示すことは今後の課題である。

テンプレート発見問題を解く手法として、上記仮定と、共通部分とそうでない部分に現われる部分文字列の頻度分布の差を利用して、高速に共通部分を特定する手法を

☆ http://www.asahi.com/
☆☆ http://www.yomiuri.co.jp/

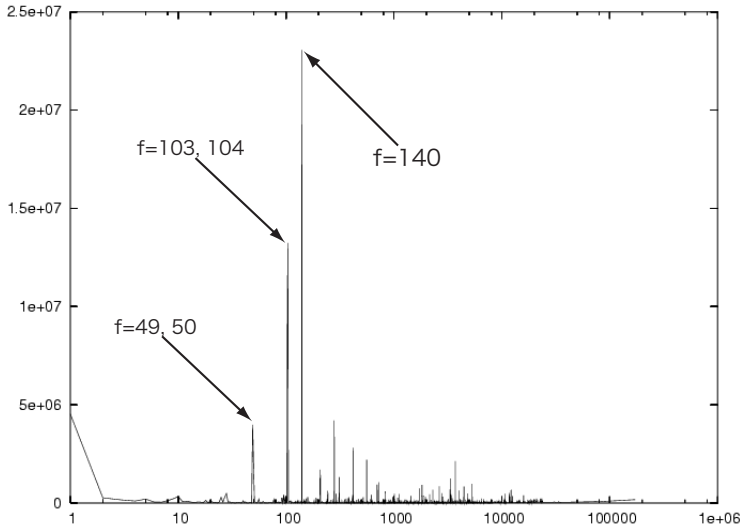

図 3 3 社混合の $F(f)$ グラフ

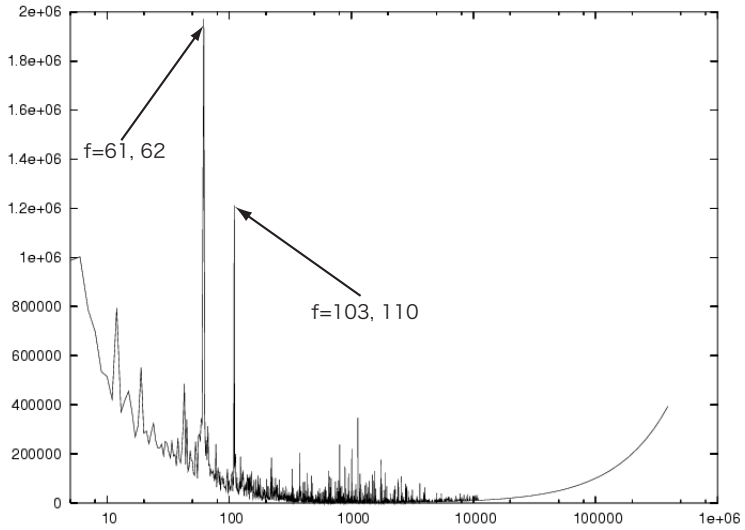

図 4 九州大学内の Web ページの $F(f)$ グラフ (ただし $f \geq 5$)

提案した。また、Web 上のテンプレートを持つ入力ファイルに対し実験を行なった。これにより、テンプレートを持たないファイルが 9 割程度あったとしても、また、複数のテンプレートが存在する場合でも、問題なくテンプレートを発見することができた。

現状では、ピークの発見は人手で行なっており、これを補助する対話的なプログラム、または、自動的なピークの抽出が今後の課題である。特に自動的なピークの抽出には、どの程度でピークと判定するかのしきい値も必要で、これができれば正の例ではない入力が与えられた時でも、「共通パタンがない」と答えることができる。

提案手法が探すパタンは、実際には頻出する部分文字列の集合である。これを部分文字列の列のように決められた形式のパタンにすることは、ピークを検出した後に簡単に行なうことができる。この時、単に部分文字列の列にするのではなく、より表現力の高いパタンに拡張することも考えられる。例えば、ゲノム情報処理の分野などでよく用いられる $[AC]$ のように複数の文字から選択できるようにするなど、探すパタンの拡張は重要な課題である。

本稿の実験では、Web 上の HTML のみを対象としたが、他の分野のデータに対する実験も行ないたい。本稿で実験したデータは、テンプレート部分の幅が十分にあり、文字種も多く、提案手法がうまく扱えるデータであったと言える。逆に、文字種が少ない DNA 配列は、Web 上のデータと対照的である。文字種が少ない場合は、長さを固定した時に異なる文字列の数が少ないので、テンプレート内の文字列とテンプレートでない部分にたまた

ま出現した文字列が一致する可能性が高くなる。そのため、テンプレートの幅がある程度必要とされると予想される。

参 考 文 献

1) R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pp. 487–499, September 1994.
2) D. Angluin. Finding Patterns Common to a Set of Strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
3) H.Arimura and S.Shimozono. Maximizing Agreement with a Classification by Bounded or Unbounded Number of Associated Words. In *Proceedings of the 9th International Symposium on Algorithms and Computation*, Lecture Notes in Artificial Intelligence 1533, pp. 39–48. Springer-Verlag, 1998.
4) T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering Frequent Substructures in Large Unordered Trees. In *Proceedings of the 6th International Conference on Discovery Science (to appear)*, 2003.
5) H. Bannai, S. Inenaga, A. Shinohara, M. Takeda, and S. Miyano. A String Pattern Regression Algorithm and Its Application to Pattern Discovery in Long Introns. In *Genome Informatics (GIW2002)*, Vol. 13, pp. 3–11, 2002.
6) M.Giraud and G.Kucherov. Maximal Repetitions and Application to DNA sequences. In *Proceedings of the Journes Ouvertes: Biologie, Informatique et Mathmatiques*, pp. 165–172, May 2000.
7) D.Gusfield. *Algorithms on Strings, Trees and Sequence*. Cambridge University Press, New York, 1997.
8) M. Hirao, H. Hoshino, A. Shinohara, M. Takeda, and S. Arikawa. A Practical Algorithm to Find Best Subsequence Patterns. In *Proceedings of the 3rd International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence 1967, pp. 141–154. Springer-Verlag, December 2000.
9) D. Ikeda, Y. Yamada, and S. Hirokawa. Eliminating Useless Parts in Semi-structured Documents using Alternation Counts. In *Proceedings of the 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence 2226. Springer-Verlag, November 2001.
10) M. Kearns and L. Pitt. A Polynomial-Time Algorithm for Learning k-variable Pattern Languages from Examples. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pp. 57–71, 1989.
11) R. Kolpakov and G. Kucherov. Finding Maximal Repetitions in a Word in Linear Time. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 596–604, 1999.
12) S. Kurtz and C. Schleiermacher. REPuter: Fast Computation of Maximal Repeats in Complete Genome. *Bioinformatics*, 15(5):426–427, 1999.
13) N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
14) R. E. Schapire. Pattern Languages Are Not Learnable. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pp. 122–129, 1990.
15) T. Shinohara. Polynomial Time Inference of Extended Regular Pattern Languages. In *RIMS Symposium on Software Science and Engineering (1982)*, Lecture Notes in Computer Science 147, pp. 115–127. Splinger-Verlag, 1983.
16) E. Tateishi, O. Maruyama, and S. Miyano. Extracting Best Consensus Motifs from Positive and Negative Examples. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 1046, pp. 219–230. Springer-Verlag, February 1996.
17) Y. Yamada, D. Ikeda, and S. Hirokawa. SCOOP: A Record Extractor without Knowledge on Input. In *Proceedings of the 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence 2226, pp. 482–487. Springer-Verlag, November 2001.
18) Y. Yamada, D. Ikeda, and S. Hirokawa. Automatic Wrapper Generation for Multilingual Web resources. In *Proceedings of the 5th International Conference on Discovery Science*, Lecture Notes in Computer Science 2534, pp. 332–339. Splinger-Verlag, 2002.
19) Y.Yamada, D.Ikeda, and S.Hirokawa. Frequency Analysis of Semi-structured Documents with Structural Similarity. In *Proceedings of the 2nd Forum on Information Technology (FIT2003)*, 2003.
20) 下畑, 杉尾, 永田. 隣接文字の分散値を用いた定型表現の自動抽出. 第 110 回情報処理学会自然言語処理研究会, pp. 71–78, 1995.
21) 新納, 井佐原. 擬似 N グラムを用いた助詞的定型表現の自動抽出. 情報処理学会論文誌, 36(1):32–40, January 1995.
22) 中村. 着手記号列の出現頻度に基づく囲碁棋譜からの定型手順獲得. 情報処理学会論文誌, 43(10):3030–3036, October 2002.
23) 長尾, 森. 大規模日本語テキストの n グラム統計の作り方と語句の自動抽出. 第 96 回情報処理学会自然言語処理研究会, pp. 1–8, 1993.