総出現数による文字列の頻出パタンマイニングと共通パタン発見への応用

池田，大輔
九州大学附属図書館

山田，泰寛
九州大学システム情報科学府

出版情報：第4回データマイニングワークショップ，pp.33-40，2004-09-30
バージョン：accepted
権利関係：

総出現数による文字列の頻出パタンマイニングと共通パタン発見への応用

池田 大輔
九州大学附属図書館
daisuke@lib.kyushu-u.ac.jp

山田 泰寛
九州大学システム情報科学府
yshiro@matu.cc.kyushu-u.ac.jp

概要

本稿では、文字列の集合が与えられた時、頻出する文字列パタンを探す頻出パタンマイニングの新たな枠組みとマイニングアルゴリズムを提案する。提案する枠組みの特徴は、頻度による部分文字列の表現方法と総出現数による高頻度・低頻度の峻別にある。つまり、アルゴリズムはある頻度 f 回出現する全ての部分文字列をまとめて f により表し、また、頻度 f の大小により頻出か否かを判断するのではなく、f 回出現する部分文字列の総出現数 $F(f)$ を用いて高頻度と低頻度を峻別する。そのため、通常の頻出パタンマイニングでは枝刈り及び頻出パタンの評価に用いられる最小サポート等のしきい値は不要で、特異的に頻出するパタンを発見可能である。頻出パタンとして様々なパタンが考えられるが、本稿では最も基本的な部分文字列を考え、頻出部分文字列の集合を発見する線形時間アルゴリズムを与える。このアルゴリズムを情報抽出におけるテンプレートの特定問題へ応用し、様々な言語の実データを用いた実験を行なう。特に、複数の言語で記述されたファイルを同時に与えても、問題なくテンプレートが特定できることを示す。

1 はじめに

ゲノム配列、HTML/XML 等の半構造化データ、メール・ニュース等のテキストデータが増大している。このようなデータから、求めるテキストを発見する検索技術、有用な知識を発見する技術、コンテンツを抽出し加工・統合する技術等の重要性が高まっている。これらの研究分野はテキストマイニングと呼ばれる分野の一部として、現在盛んに研究されている。

大量のデータを対象に有用な知識を発見するための研究として、データベースを対象にしたデータマイニングが以前から研究されている。データマイニングでは、最小サポートと呼ばれるしきい値より多く出現する結合規則を有用な知識・ルールとして出力する頻出パタンマイニングの枠組みがよく用いられる。これは非常に一般的な枠組みであり、データベースだけではなく文字列や時系列パタン等に応用した研究もある [1, 6, 9, 15]。

構造化されたデータベースと比較して、テキストマイニングの対象となるテキストは、構造が無いか、または緩い構造しか持たず、データマイニングの手法を直接援用できるわけではない。また、そもそも頻出パタンマイニングには最小サポートを指定する必要がある。このしきい値は、出力されるルールを制限することで意味のあるルールとそうでないルールを数値で峻別する役割を果たしているが、その値は試行錯誤的に決めるため、何回もアルゴリズムを走らせる必要がある。同時に、このしきい値はアルゴリズムの枝刈りにも用いられるため、値によっては非常に時間がかかることになる。

本稿では、テキストを対象にした新たな頻出パタンマイニングの枠組みと、実際のマイニングアルゴリズムを提案する。このアルゴリズムは**部分文字列増幅法**と呼ばれ、もともと、筆者らにより共通部分を高速に特定するために開発された [4, 13, 14]。その基本アイデアは頻度の高い部分文字列は高い確率で共通部分に一致するというもので、頻出パタ

ンマイニングと同様の枠組みとして考えるほうが自然である。そこで、本稿では新たな文字列上の頻出パタンマイニングの枠組みとして提案する。

提案する枠組みの特徴は、部分文字列の表現方法と総出現数を用いた興味深いパタンの定義の方法にある。ある文字列の部分文字列 w を表す方法として、w そのものや、接尾辞木における接尾辞の接頭辞、また、出現位置で表すことが一般的である。一方、総出現数による頻出パタンマイニングでは、頻度（出現回数）f により部分文字列を表す。部分文字列の総数は $O(|w|^2)$ 程度にもなるが、この表現によりコンパクトに部分文字列全てを表現でき、頻出な部分文字列を線形時間で発見することが可能になる。

もう一つの特徴は総出現数である。高頻度なものを探すには、単純に頻度の順に並べればよく、実際、頻出パタンマイニングや自然言語における n グラム統計等ではこのような手法がとられる。しかし、最も頻度の高いものは、短く単純なパタンや語であるため、知識として役にたたないことが多い。そのため、通常の頻出パタンマイニングでは最小サポートにより意味のあるパタンを規定し、一方、n グラム統計ではストップワードによる不要語の除去や n を適切に決めることで意味のあるものを探そうとする。

一方、提案する総出現数による頻出パタンマイニングでは、頻度 f ではなく、f 回出現する部分文字列の総出現数 $F(f)$ を計算する。何度も現われる部分文字列は少ないため、f を大きくすると $F(f)$ は急激に小さくなる。しかし、その中で頻出する部分文字列の総出現数は大きくなる。特に、ある程度長い部分文字列 w の場合、w の中の全ての部分文字列も少なくとも同じ回数出現することになる。そのため、w の出現回数 f_w に対し $F(f_w)$ の値は $O(|w|^2)$ 程度に増幅され、長い頻出部分文字列がより見つけやすくなる。頻出パタンマイニングにおいても、同じ頻度であればより長いパタンを出力しようとする閉パタンの研究が盛んになってきている [3, 8, 10, 11]。つまり、本手法には閉パタンの考えの一部があらかじめ枠組みとして取りいれられていると言える。

テキストマイニングでは、様々な形式のパタンが抽出すべき知識やルールとして提案されている。総出現数による頻出パタンマイニングにおいても、文字列を基本とした様々なパタンが考えられるが、本稿では最も基本的な部分文字列を探す問題を考える。単なる部分文字列では表現力に乏しいため、一部の研究 [15] を除き、より複雑なパタンを探しているようである。しかし、共通部分の表現方法として、例えば、情報抽出やラッパー生成アルゴリズムの入力となるテンプレートを共有するファイル群を発見する手法 [5] の一部に組み込まれて、その有用性が示されている。

提案アルゴリズムを、Web 上で収集した文書を対象にした共通パタン発見へ応用した実験を紹介する。収集したファイルは新聞記事のファイルで、6 サイトで収集した。部分文字列増幅法のそもそもの開発動機は共通部分の高速な特定であったから [4, 13, 14]、同様の実験はすでに行なっている。しかし、用いられる言語は日本語に限られ、原理的には様々な言語に対して応用可能であるはずのアルゴリズムであったが、実証されていたわけではない。部分文字列増幅法は、後述するようにコンテンツ部分の文字列の頻度分布に依存するため、他の言語での実証実験は重要である。

本稿では、英語、ドイツ語、中国語の 3 ヶ国語を用いて実験を行ない、これらの言語に対する非依存性を確認した。特に、複数の言語からなるファイルを同時に与えても、問題なく共通パタンを発見できることが示された。このことは、リンクを辿って収集することが一般的である Web 上のデータを扱う際には、どのような言語のファイルが混入しているか不明なため、Web 上のデータを扱うテキストマイニングアルゴリズムには重要な特徴である。

2 準備

D を文書集合とする。$V(f)$ と $F(f)$ を、それぞれ、D 中にちょうど f 回出現する部分文字列の異なり数と総出現数とする。つまり、$F(f) = f \times V(f)$ である。

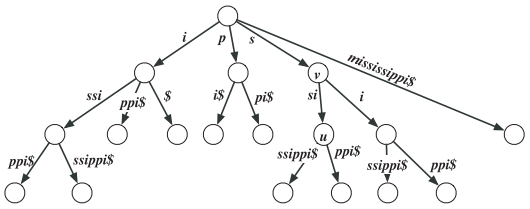

図 1　*mississippi*$ に対する接尾辞木

$ を、任意の文字 $a \in \Sigma$ に対し $a \neq \$$ となる特別な文字とする。文字列 w に対し $A = w\$$ とおき、A_p $(1 \leq p \leq |A|)$ で A の p 文字目から始まる接尾辞を表す。$A_{p_1}, A_{p_2}, \ldots, A_{p_n}$ を、辞書式順序[*1]に並べた A の全ての接尾辞とする。w に対する**接尾辞木**とは $A_{p_1}, A_{p_2}, \ldots, A_{p_n}$ に対するコンパクトなトライのことである [7]。接尾辞木のノード v に対し、根から v に至るパス上のラベルの文字列を連接してできた文字列を $BS(v)$ で表し、**分岐語**と呼ぶ。

図 1 に、*mississippi* に対する接尾辞木を示す。ノード u と v に対し、それぞれ $BS(u) = ssi$ と $BS(v) = s$ である。

u を任意の接尾辞木のノードとする。$BS(u)$ の出現回数は u の子孫となる葉の数である。実際、$BS(u) = ssi$ の子孫となる葉は 2 つであり、*mississippi* 中における出現回数も 2 回である。

v を u の親ノードとすると、$BS(v)$ は $BS(u)$ の（真に短い）接頭辞である。さらに、$BS(u)$ の接頭辞で、$BS(v)$ を真に含むものの出現回数は $BS(u)$ の出現回数と等しい。つまり、部分文字列の頻度を数える場合、全ての部分文字列を数える必要はなく、分岐語のみを数えれば十分である。

3　総出現数による頻出パタンマイニング

3.1　基本的な考え方

本節では、具体的な例を用いて総出現数による頻出パタンマイニングの考え方を説明する。基本的には、頻出するパタンとそうでないパタンの頻度の差を用いるが、頻度 f で比較するのではなく、総出現数 $F(f)$ で差を測る。実験で用いる Web 上の HTML ファイルにおいては、非頻出部分が自然な文章やコンテンツに、頻出部分がテンプレートに対応する。まず、非頻出部分の $F(f)$ がどのようなグラフになるか見ていこう。

図 2 は、英語と日本語の小説を対象とした $F(f)$ のグラフである。これらのグラフの縦軸を $V(f)$ にして両対数でプロットすると、グラフは直線になる。これは、自然言語処理の分野では Zipf の（第二）法則として知られる。ただし、数える単位は単語ではなく全ての部分文字列であることに注意する。そのため、どのような言語であっても前処理の必要はない。

一方、図 3 左は、産経新聞[*2]から取得した 50 個の記事ファイルを D とした時の $F(f)$ のグラフである。各ファイルは基本的に日付、見出し、サブ見出し、記事本文の 4 種類のコンテンツから構成され、それ以外は全て同じである。つまり、固定されたテンプレートの中にコンテンツが埋め込まれている。この場合、テンプレートが頻出する部分として検出したい部分である。

このグラフにおいて、f が増加するにつれ $F(f)$ は急激に小さくなっているが、この部分はコンテンツにより生成された $F(f)$ である。実際、入力ファイルからテンプレートを除去して抽出したコンテンツのみのデータから生成した $F(f)$ は、図 3 右のようになる。

基本的に $F(f)$ は急激に小さくなっているが、その中に特異的に大きな $F(f)$ が何か所かある。まず、$f = 50$ に特異的に高い $F(f)$ が存在することが分かる。この数はファイル数と同じであり、ある程度長い部分文字列がちょうど 50 回出現していることを示している。$F(f)$ は、全ての部分文字列の出現回数を足しあわせて得られるので、何度も繰り返し出現する文字列から生成される $F(50)$ は非常に大きい値となる。そのため、この手法を部分文字列**増幅**法と呼ぶ。

D 中にちょうど 50 回出現する部分文字列をつなげて得られる文字列は多くの部分文字列を含ん

[*1] 任意の文字列 $w \in \Sigma^*$ に対し、w を $w\$$ より先に並べることにする。

[*2] http://www.sankei.co.jp/

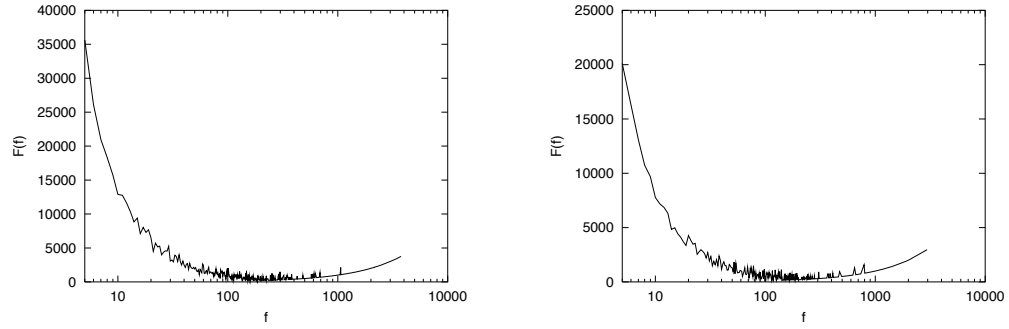

図 2　英語の小説（左）と日本語の小説（右）から生成した $F(f)$ のグラフ。横軸（対数スケール）は頻度 f で、縦軸は総出現数 $F(f)$

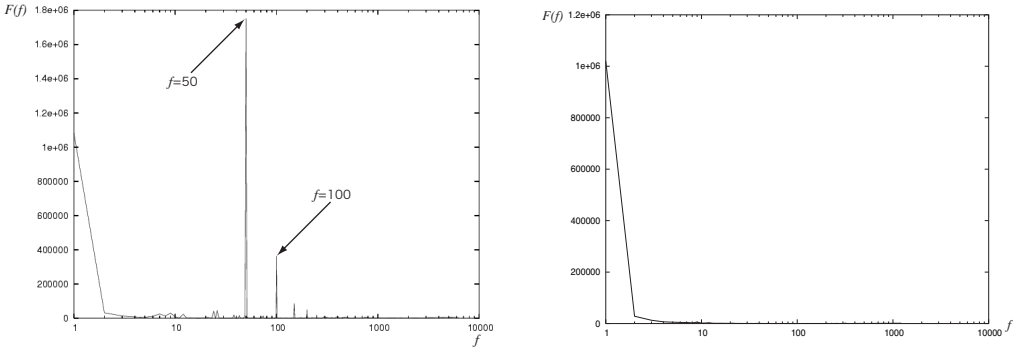

図 3　産経新聞の記事ファイル 50 個から生成した $F(f)$ のグラフ（左）と、コンテンツ部分のみ対象に生成した $F(f)$ のグラフ（右）

```
</b></i><br><P>
<center>
<table width=467>
<tr>
<td>

<!---------ヘッダ情報終了---------->

<!------------★★ここから入れ替えてね・・---------------->
<font color="#8b0000">■</font><b>
```

図 4　コメント、空白記号、テキスト要素 “■” を含んだ産経新聞のテンプレートの一部

でいる。例えば、図 4 の文字列は 72 文字であり、テンプレートの一部であった。また、このテンプレートからコメントやタグ、あるいはテキスト要素 “■” もテンプレートの一部として抜きだされていることが分かる。この中に含まれる部分文字列の総出現数は $72 \cdot (72-1)/2 = 2556$ となり、また、全ての記事がこの文字列を含んでいることから少なくとも $F(50)$ は 2556×50 となることが分かる。さらに、$f = c \times 50\,(c = 2, 3, \ldots)$ にも高い値が見える。これらは、50 回出現するテンプレートの中で 2 度、3 度と出現する部分文字列による。

以上より、$F(f)$ が特異的に大きな値を持つ f の値が分かれば、ちょうど f 回出現する部分文字列からテンプレートが見つけられることが分かる。

3.2　部分文字列増幅法

特異的に大きな $F(f)$ を持つ f が見つかったとしよう。この時の f が表しているのは、部分文字列の集合である。そこで、本節では、複雑なパタンではなく、頻出する部分文字列の集合を見つける**部分文字列増幅法**と呼ばれるアルゴリズムを提案する。

図 5 に、部分文字列増幅法の擬似プログラムコードを示す。部分文字列増幅法の入力は文字列の集合 `D` であり、出力は `D` に現われる分岐語の集合である。そのため、同一頻度であっても分岐語の接頭辞は自動的に排除される。

まず `Count(D)` により、`D` から接尾辞木を構成

```
function Amplification(var D):
    set of strings
  var
    V, F: ハッシュ表;
  begin
    V:=Count(D); { 分岐語の頻度をカウント }
    for f∈ keys(V); { 全ての頻度に対し }
       F(f):=0;
       for w∈ V(f);
         F(f) += f· |w|;
       end;
    end ;
    f_p:= FindPeaks(F);
    return(V(f_p));
  end ;
```

図 5　部分文字列増幅法の擬似プログラム

し、接尾辞木の全ての分岐語の頻度をカウントする。次に、頻度 f をキーに持ち、ちょうど f 回出現する分岐語の集合を値として持つハッシュ V[*3]を構成する。

次に V(f) から F(f) を計算する。このために F(f) = f· |V(f)| を計算するが、これだけだと f 回出現する分岐語の個数しかカウントされない。よって、まず F(f):=0 と初期化し、各分岐語 w∈ V(f) に対し、分岐語と等価な接頭辞の出現回数は等しいので、F(f) += f· |w| を計算する。

次に FindPeaks(F) は、ピークとなる頻度を返す。具体的には、どのような基準でピークとすべきかは議論の余地があるが、例えば、各 f に対し、$F(f-1)$ と $F(f+1)$ に対する $F(f)$ の変化率等が妥当であろう。このような基準では、ピークか否かは各 f に対しその近傍の F の値のみで決定できることに注意する。最後に、D 中に f_p 回出現する分岐語の集合を V(f_p) を返す。

接尾辞木におけるノードの数から、次の補題を示すことができる。

[*3] ここで V(f) は異なり数ではなく集合であることに注意する。

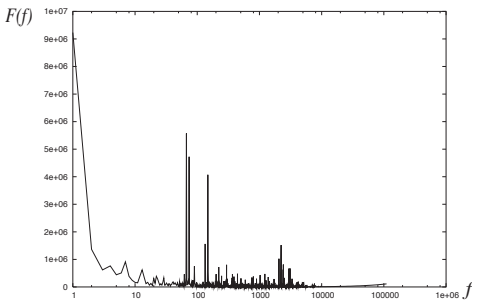

図 6　washingtonpost.com の記事ファイル 74 個から生成した総出現数 $F(f)$ のグラフ

補題 1 D 中に現われる部分文字列の出現回数の異なり数は $O(n)$ である。ただし、n は D 中の文字列長の総和である。

この補題を用いて、部分文字列増幅法が線形時間アルゴリズムであることが示される。

定理 1 部分文字列増幅法の時間計算量は $O(n)$ である。ただし、n は D 中の文字列長の総和であり、ピークは各 f に対しその近傍の総出現数のみで決定できるものとする。

4　共通パタン発見への応用

本節では、さきほどの部分文字列増幅法を用い、共通パタンへの応用可能性を、Web 上から収集したデータを用いて実証する。

用いたデータは 6 つの新聞社から取得した新聞記事ファイルである（表 1 参照）。どのサイトも、記事ファイルを人手で選別し、アルゴリズムに与えた。その意味で、ノイズなしのデータである。

どのサイトのデータからも共通のテンプレートが発見できたが、 そのうちいくつかについては $F(f)$ グラフを紹介する（図 6, 7, 8 参照）。

どの図においてもファイル数と一致する f において、最も高いピークが得られた。しかし、それ以外にも全ファイルに共通するわけではない、部分的なピークが見つかり、部分的なテンプレートの存在

表 1　データセット

サイト名	URL	使用言語	ファイル数
washingtonpost.com	http://www.washingtonpost.com/	英語	74
Financial Times	http://www.ft.com/	英語	50
Financial Times Deutschland	http://www.ftd.de/	ドイツ語	101
Frankfurter Rundschau	http://www.zeit.de/	ドイツ語	50
中国新聞網	http://www.chinanews.com.cn/	中国語	171
人民日報	http://www.peopledaily.co.jp/	中国語	127

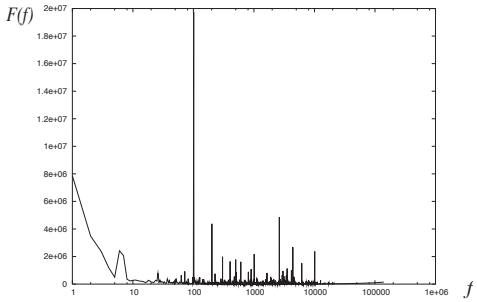

図 7　Financial Times Deutschland の記事ファイル 101 個から生成した総出現数 $F(f)$ のグラフ

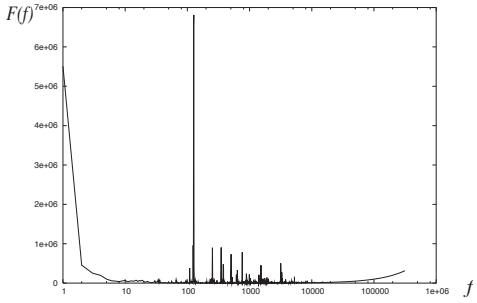

図 8　人民日報の記事ファイル 127 から生成した総出現数 $F(f)$ のグラフ

が明らかになった。また、高いピークを得るためには、ファイル数は多いほうが望ましいが数十から百数十程度で十分認識できるピークが得られることが分かる。

上のグラフのデータを混合したファイルを対象にした場合も、それぞれのテンプレートが適切に発見できた。一般に Web 上の文書を収集する場合はリンクを辿ることが多く、テキストマイニングアルゴリズムは様々な言語のファイルを同時に扱える必要がある。混合しても、単にグラフを重ねあわせることと同等なので、もともと高いテンプレートは混合されたグラフにおいても高くなる。

5　関連研究

部分文字列増幅法は、単に頻度を数えることで特徴的な文字列を見つけ、これらの文字列により入力データを分類する手法である。新しい手法であるではあるが、既存の多くの手法や枠組みと類似点を持つ。本節では、類似すると思われると部分文字列増幅法との関連について述べる。

クラスタリング

部分文字列増幅法は、距離として頻度を採用したクラスタリングと考えることができる。同一の距離のみを同じクラスタと考えるため、通常のクラスタリングで問題になる、ある要素をクラスタにいれるかどうかの距離に関するしきい値を考慮する必要はない。また、クラスタ数に依存しないので、どれだけのクラスタが含まれていても問題ない。

n グラム統計

自然言語処理における n グラム統計では、主に頻度 f の順に順位づける手法と、異なり数 $V(f)$[*4]を f の順番で並べる手法がよく用いられる。自然言語の文書における単語を対象にすると、前者は Zipf の第一法則が、後者は Zipf の第二法則が対応する。部分文字列増幅法は、Zipf の第二法則と関連が深いが、$V(f)$ ではなく $F(f) = f \times V(f)$ を用い、また

[*4] 頻度の頻度として知られている。

縦軸は対数スケールではないところに相違がある。さらに、単語ではなく部分文字列を数えること、固定した n ではないので n を事前に決める必要がないこと等、アルゴリズムを動かすのに必要な前提知識の有無も異なる。そのため、本稿で示したように任意の言語に対する処理を一度に行なうことが可能となる。

情報抽出

Web マイニングの一分野として、情報抽出がある。これは、与えられた複数のテキストから共通な部分をテンプレートとして特定し、テンプレートの間に埋め込まれたコンテンツを抽出する手法を開発する分野である。この分野には、訓練例を必要とする機械学習による手法と、与えられたテキスト間の頻度や繰り返しパタンのみからテンプレートを特定する手法が存在する。部分文字列増幅法の開発動機も、テンプレート部分の高速な特定であり [13, 14]、後者の手法と類似点は多い。しかし、従来手法と比較すると単なるテキストを扱っているため、木構造として扱ったり、単語やタグごとのトークンに区切る必要がない。もちろん、部分文字列増幅法の前処理や後処理にこのような知識を用いた処理を行なうことは可能であるが、テンプレートを構成する文字列を特定するためには、タグや単語に区切ることが本質的に必要なわけではないことが分かる。

また、より重要な相異点として、どの程度頻度があればテンプレートと考えるか、について新たな観点を提供している点が挙げられる。例えば、Arasu と Garcia-Molina は頻度によるテンプレートの特定において、頻度の高いトークンはテンプレートである可能性が高いとしながら、どの程度の頻度があればよいかについては言及していない [2]。一方、部分文字列増幅法においては、頻度 f ではなく $F(f)$ により判断するため、より容易にテンプレートであるかどうか判断できる。

6 まとめ

本稿では、総出現数を基準とする文字列上の頻出パタンマイニングを提案し、発見するパタンとして部分文字列を採用した場合の線形時間アルゴリズムを与えた。この枠組みにおいては、最小サポート等は不要であり、枝刈りも行なわない。また、ある文字列における部分文字列を頻度により表現するというアイデアで、簡単に線形時間アルゴリズムを得ることができた。

本稿では、この枠組みを共通部分発見に援用したが、ストリームデータや時系列データ等に適用することは今後の課題である。その場合、非頻出パタンに対し、Web 上のコンテンツにおける頻度分布のように “自然” な分布が成立するかどうかが第一の問題となる。そのため、様々なデータにおける総出現数に相当する概念の頻度分布を調べる必要がある。

本稿で提案した部分文字列増幅法は、文字列のパタンとしては最も単純な部分文字列の集合を出力するものである。しかし、その単純さにもかかわらず、ラッパー生成の前処理 [12] や情報抽出アルゴリズムの入力となるテンプレートを共有するファイル群の発見 [5] として利用できる。特に、後者については、情報抽出そのものはよく研究されているが、情報抽出の対象となるファイルをどのようにして獲得するかに言及した論文はほとんどなく、重要な技術と言える。

部分文字列増幅法の対象はテキストであれば何でもよく、本稿で扱った半構造化テキストに限定するものではない。また、長い頻出部分文字列であるほど $F(f)$ の値が増加することと、パタンの長さは計算量に影響しないため、長いパタンのほうがより都合がよい。逆に短いパタンは見つけにくいと予想される。例えば、ゲノム配列もそのまま入力として扱うことはできるが、文字数の少なさや一致する部分が短い場合は十分に $F(f)$ の値が大きくならないと予想される。そのため、ギャップを扱えるように拡張し、ギャップを含めて一つの文字列として扱うことができるようにすることが必要である。

本稿で紹介した総出現数 $F(f)$ のグラフは、ノイズを含まないデータから作成されたものである。そのため、最大のピークは非常に明瞭であった。ノイズが混入する場合は、ノイズがなすピークのよう

な $F(f)$ も存在する可能性が高く、ノイズとの峻別のための境界値が必要になる[*5]。

参考文献

[1] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering*, pp. 3–14, 1995.

[2] A. Arasu and H. Garcia-Molina. Extracting Structured Data from Web Pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 337–348, 2003.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhai. Mining Frequent Patterns with Counting Inference. *SIGKDD Explorations*, 2(2):66–75, 2000.

[4] D. Ikeda. *Autoschediastic Text Mining Algorithms*. PhD thesis, Graduate School of Information Science and Electrical Engineering, Kyushu University, March 2004.

[5] D. Ikeda and Y. Yamada. Gathering Text Files Generated from Templates. In *Proceedings of Workshop on Information Integration on the Web (IIWeb-04)*, August 2004. `http://cips.eas.asu.edu/iiweb.htm` (to appear).

[6] H. Mannila and H. Toivonen. Discovering Generalized Episodes Using Minimal Occurrences. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 146–151, 1996.

[7] E. M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *JACM*, 23(2):262–272, 1976.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 24(1):25–46, 1999.

[9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings of the 17th International Conference on Data Engineering*, pp. 215–226, April 2001.

[10] T. Uno, T. Asai, Y. Uchida, and H. Arimura. LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003.

[11] J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 236–245, 2003.

[12] 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.

[13] 池田, 山田, 廣川. 文字列の頻度分布による共通パタン発見. 第 72 回情報処理学会情報学基礎研究会, pp. 25–32, September 2003.

[14] 池田, 山田, 廣川. 文字列増幅法による共通パタン発見アルゴリズム. 第 47 回情報処理学会数理モデル化と問題解決研究会, pp. 45–48, December 2003.

[15] 坪井. 頻出部分文字列のマイニング. 第 158 回情報処理学会自然言語処理研究会, pp. 147–154, 2003.

[*5] ただし、これは最小サポートとは異なり枝刈り等はしないため $F(f)$ そのものの分布は影響は受けないことに注意する。