自然言語処理 117－6
（1997. 1. 20）

# ギャップのある $n$-gramによる言回しの抽出

國吉芳夫　　中西正和
yoshio@nak.math.keio.ac.jp　czl@nak.math.keio.ac.jp

慶應義塾大学大学院 理工学研究科 計算機科学専攻

ギャップを許す任意の $n$ についての $n$-gram の検索に基づいたアルゴリズムにより、ギャップを含む言回しを自動抽出するための一手法について述べる。これは単語のペアを繰り返し拡張することにより間接的に連語表現の統計量を推定する方法とは異なり、連語の統計量を直接扱うものである。本手法では左右のエントロピーによって $n$-gram を抽出し、連語を構成する $n$-gram の相関性を評価する関数を用いて連語の順位付を行う。基本となるアルゴリズムは多次元の範囲問合せのアルゴリズムに資源を意識した連語抽出のための制約を加えたものである。英文コーパスに対する抽出実験により、提案した手法が有効であることが確かめられた。

# Extracting Collocations Using Gapping $N$-gram

Yoshio Kuniyoshi　Masakazu Nakanishi

Department of Computer Science
Graduate School of Science and Technology
Keio University

We describe a method for automatic extraction of interrupted collocations. Rather than based on repeated extension of word pairs, proposed method is based on $n$-grams for arbitrary $n$ with gaps called gapping $n$-gram. The method is a kind of simple algorithm for multi dimensional range query problem in addition to several special constraints for collocation retrieval which implements *resource conscious* way of extraction. Also, we propose a kind of function which provides association value of the $n$-gram and show the effectiveness of our definition experimentally.

# 1 はじめに

自然言語における言回しの表現は規則で統一的に扱うことが難しい。また、分野によって使用される言回しも異なったものになり、手作業で抽出することが困難であるため、自動抽出に対する強い要求がある。

多くの研究は、2単語の組の相関度を求め、関連性の高いペアを順次拡張していくことによって長い単語列を連語として抽出する方法を取っている [1, 2]。一方で、最近は任意の $n$ についての $n$-gram に基づき、連語を直接抽出する方法も提案されるようになってきている [3, 4]。

特に、池原らの手法は、連鎖型及び離散型の連語表現を比較的効率良く網羅することができるという点で特筆すべきものである [5]。しかしながらその手法はすべての連語を効率良く網羅することを目的としているため、抽出の評価基準が抽出手法に組み込まれており、他の抽出基準を取り入れるように変更することは難しい。

本稿では、ギャップのある $n$-gram に基づいて離散型の連語を抽出するための一手法を提案する。この手法は探索型であり、抽出基準が手法に組み込まれた網羅型のアプローチとは異なり、抽出基準として任意の手法を取り入れることが可能である。また、本手法で提案する連鎖型の $n$-gram の組の相関性を評価する関数と資源を意識した抽出法の連語抽出に対する有効性を示す。

# 2 $N$ が大きい $n$-gram の求め方

大きな $n$ に対する $n$-gram を求めることは、時間的/空間的計算量の面からあまり実用的ではないとされてきたが、長尾らは十分に大きい $n$ に対しても $n$-gram が効率良く求められることを示した [3]。彼らの手法は、半無限文字列 (sistring – semi infinite string) [6, 7] に基づくデータ構造である suffix array [8] を用いたものである。Sistring とは、コーパスを一つの文字列とみなしたとき、その文字列のある部分以降からなる部分文字列のことで、コーパス中を指すポインタとして実装される。Suffix array とは、コーパスに含まれるすべての sistring からなる配列を各 sistring に関して辞書順に並べ替えたものである。

Suffix array では同じ先頭部分を持つ $n$-gram は必ず一つの範囲に入っているために、単純な2分探索アルゴリズムによって上限と下限を求めることにより、任意の $n$ に対する $n$-gram の頻度を $O(\log n)$ 時間で求めることができる [9]。

# 3 ギャップのある $n$-gram の定義

まず最初に、一般のギャップのある $n$-gram を定義し、次に、いくつかの特別な制約を課したギャップのある $n$-gram を定義する。

## 3.1 ギャップのある $n$-gram

ギャップのある $n$-gram は、任意の数の任意の単語で中断されることを許した $n$ 単語の列で、形式的には以下に示す単語列の同一視として定義される。

$$w_1 w_2 \cdots w_n \text{ where } d(w_i, w_{i+1}) \geq 0 \qquad (1)$$

ここで、$d(w_i, w_j)$ は単語 $w_i$ と $w_j$ の間に入る任意の単語の数である。

## 3.2 Multi-$n$-gram

Multi-$n$-gram とは、ギャップのある $n$-gram の特別な形で、ギャップの配置の異なる $n$-gram を区別するものである。すなわち、multi-$n$-ngram は (連鎖型の) $n$-gram の列で、任意の数の任意の単語が各 $n$-gram の間に入ることを許したもので、形式的には以下に示す列の同一視として定義される。

$$ws_1 ws_2 \cdots ws_m \text{ where } d(ws_i, ws_{i+1}) \geq 0 \qquad (2)$$

ここで、$d(ws_i, ws_j)$ は $n$-gram $ws_i$ と $ws_j$ の間に入る単語の数である。

## 3.3 Bi-$n$-gram

Bi-$n$-gram とは連鎖型の $n$-gram を2つしか含まないように制限された multi-$n$-ngram である。

# 4 Multi-$n$-gram の求め方

Multi-$n$-gram の頻度は多次元の範囲問合せ問題を解くことで求められる [10]。ただし、これだけでは multi-$n$-ngram を構成する各 $n$-gram の一部また

は全部を重複して数え上げてしまう場合が生じてしまう。我々は頻度とはコーパス中での最大同時出現数 (現れ方は複数有り得る) であると考え、互いに重なり合わない multi-$n$-ngram のみを数え上げるような制約を導入した。

コーパスが次のような列であるとして、multi-$n$-ngram $A\ldots\ B\ldots A$ の頻度を計数することを考える。

$$A_1\ldots A_2\ldots B_1\ldots B_2\ldots A_3\ldots A_4$$

このとき、互いに異なる要素からなる multi-$n$-ngram の数は 8 である。この数を頻度としてしまうと各 $n$-gram $A, B, C$ はそれぞれ 4 回数え上げに使われることになる。自然言語の文では、ある表現が他の表現の一部であることはあっても、同じ表現の一部になることはない。したがって、この場合の頻度は 2 とするのが妥当であると考えられる。さらに、コーパスが次のような列であった場合、重複を許さないような数え上げを考えると、頻度は 1 とするべきである[1]。

$$A_1\ldots B_1\ldots A_2\ldots B_2\ldots A_3$$

このような数え上げは multi-$n$-ngram の最大同時出現数を数えれば良い。

このような数え上げを、**資源を意識した数え上げ**と呼ぶ。また、複数の文に跨るような multi-$n$-ngram は頻度に入れるべきではない。以下に示すアルゴリズム 1 は多次元の範囲問合せ問題にこれらの制約を取り入れたものになっている。

**アルゴリズム 1**

```
proc countMNgram(ws_0, ..., ws_{n-1}) ≡
  count := 0;
  for i := 0 to n − 1 do
      rng_i := range(ws_i);
      copyRange(rng_i, a_i);
      sortRange(rng_i, a_i);
      p_i := 0; e_i := size(rng_i)
  od;
  while p_0 ≤ e_0 do
        done := true;
        for i := 1 to n − 1 do
            while p_i < e_i ∧
                  tl(i − 1) + mingap > hd(i)
                  do p_i++ od;
            if p_i ≥ e_i
              then break label2
            fi
            if hd(i) > tl(i − 1) + maxgap
              then done := false;
                   break label1
            fi
        od:label1;
        if done ∧ within(hd(0), hd(n − 1))
          then
               count++;
               for i := 0 to n − 1 do
                   lasttail := tl(i);
                   while p_i < e_i ∧
                         hd(i) < lasttail do
                         p_i++
                   od
               od
          else
               p_0++
        fi
  od:label2.
where
const mingap                          min gap length
const maxgap                          max gap length
funct range(ws) ≡
  range search for n-gram ws.
funct copyRange(rng, ar) ≡
  copy the range of suffix array specified by
  rng into array ar .
funct sortRange(rng, ar) ≡
  sort the copied range ar according to sistring.
funct size(rng) ≡
  the number of items in the range.
funct hd(j) ≡
  a_i[p_j].                  p_j th index in the copied region
funct tl(j) ≡
  a_i[p_j] + len(ws_j).
funct within(x, y) ≡
  x and y point into the same sentence.
funct len(ws) ≡
  the length of n-gram ws.
```

多次元の範囲問合せ問題をより高速[2]に求めるアルゴリズム [10] も提案されているが、それらのアルゴリズムはより大きな空間を要求し、大規模なコーパスには適用しにくいので採用しなかった。

# 5 離散型の共起表現の抽出法

## 5.1 抽出過程の概要

離散型の共起表現の抽出は以下の段階を経てなされる。

1. テキストコーパスを後の処理で扱いやすい形式に整形する。
2. 異なり語彙をすべて抽出して辞書を作成する。

[1] すべての組合せを考えた場合は 4 になる。

[2] $O(\log n)$

3. 生テキストコーパスの各単語を辞書の登録番号で置き換えて、番号の配列にしたものに変換する。
4. $N$-gram 表 (suffix array) を作成する。
5. エントロピー表 (後述) を作成する。
6. コーパス中の各文に対して左右のエントロピーを利用して独立な単語列を抽出する。
7. 独立な単語列を組み合わせて multi-$n$-ngram を作成し、相関性を評価関数によって計算する。
8. 高い評価値を持つ multi-$n$-ngram を出力する。

## 5.2 連鎖型の連語の抽出

### 5.2.1 左右のエントロピー

連鎖型の連語は周囲の環境と独立であると考えられる。独立な単語列に隣接する単語はそうでない単語列よりも多様であると考えられる。すなわち、もし $n$-gram が独立な単語列であるとすると、任意の単語 $j$ に対して $n$-gram$_i w_j$ の頻度は $n$-gram$_i$ の頻度よりも著しく小さく、また $w_j$ の種類は多くなる。

森と長尾は独立な単語列に隣接する単語の多様性を捉える尺度として左右のエントロピーを提案している [11]。$N$-gram$_i$ の左エントロピー $H_l$ と右エントロピー $H_r$ は次のように定義される。

$$H_l(ws_i) = -\sum_j P_l(w_j|ws_i) \log P_l(w_j|ws_i) \quad (3)$$

$$H_r(ws_i) = -\sum_k P_r(w_k|ws_i) \log P_r(w_k|ws_i) \quad (4)$$

ここで $w_j$ は $n$-gram $ws_i$ の左側に隣接する単語であり、$w_k$ は $n$-gram $ws_i$ の右側に隣接する単語である。

### 5.2.2 エントロピー表

左右のエントロピーを毎回求めるのは非効率である。また、コーパス中のすべての $n$-gram についてエントロピーを求める必要はなく、異なる環境を持つ $n$-gram のみ計算すれば良い。

異なる環境を持つ $n$-gram は次のようにして定義される。suffix array のある項目 X が指す単語列とその直後の項目 Y が指す単語列の先頭からの一致部分の長さを X の一致単語列長と呼ぶ[3]。一致単語列長 $n$ が 0 より大きい項目を長さ $n$ の $n$-gram とみなす。このような項目は少なくとも 2 種類以上の環境を持つことが保証されている。

通常の suffix array は sistring の先頭を右側に置いているが、逆方向の sistring に基づく suffix array を考えることにより左のエントロピーも容易に計算できる。

左右に異なる環境を持たない $n$-gram は独立して抽出する必要がなく、そのような $n$-gram に対するエントロピーは 0 として良い。したがって、エントロピーの登録は一致単語列長が 0 より大きい項目に対してのみ行えば良い。

ところで、suffix array は辞書順に並んでいるだけなので、ここにエントロピーの値を書き込んでも、検索するためには計算対象の $n$-gram と一致する範囲について長さが等しい項目を線形探索する必要があり、効率良く検索することができない。そこで、suffix array を $n$-gram の長さに関する安定な整列を施したエントロピー表を作成し、これを用いて効率良くエントロピーの値を検索することを提案する。

エントロピー表は $n$-gram の長さ、sistring、左右のエントロピーの値の 3 つ組を要素とする配列である。エントロピー表は次に示すアルゴリズム 2 にしたがって作成できる。

**アルゴリズム 2**

```
proc createEntropyTable(corpus) ≡
  逆向 sistringに関して整列した suffix array L を作成;
  左のエントロピーを計算し L に書き込む;
  L を長さに関して安定に整列し、
  重複する項目を圧縮;
  正方向の sistringに関して整列した suffix array R
  を作成;
  右のエントロピーを計算し R に書き込む;
  R を長さに関して安定に整列し、
  重複する項目を圧縮;
  foreach item ∈ R do
    e := getEntropy(item, L);
    if e > 0 then
                  e を R に書き込む
    fi
  od;
  左右のエントロピーが存在するもののみを
  残すように R を圧縮.
```

エントロピー表を用いた $n$-gram のエントロピー値の検索は次のようにして行う。

**アルゴリズム 3**

```
proc getEntropy(ws, table) ≡
  rng := rangeLen(len(ws), table);
```

[3] 長尾ら [12, 3] はこの値を特定の $n$ に関する $n$-gram 表を作成するのに利用している。

```
  item := binSearch(ws, rng);
  if item = null
    then
        return 0
    else
        return item_{H_l} + item_{H_r}
  fi.
where
funct rangeLen(l, table) ≡
  range search for length l on table.
funct binSearch(ws, rng) ≡
  search n-gram ws into range specified by
  rng into the table.
```

最初のステップの範囲の検索と、第 2 のステップの $n$-gram の検索は $n$ を表のサイズとした時、$O(\log(n))$ 時間で行えるので、エントロピー値の検索は $O(\log(n))$ 時間でできる。

## 5.3 Multi-$n$-gram の評価

コーパスに含まれる可能な multi-$n$-ngram の数は膨大であるため、これらのすべてを抽出するのは実用的とはいえない。そこで、multi-$n$-ngram を評価するための手法が必要とされる。我々の採用した方法はこれを 2 段階で行うものである。まず、連鎖型の単語列を左右のエントロピーの値の和で評価する。さらに、こうして得られた単語列を組み合わせて得られた multi-$n$-ngram の相関性を評価する。

### 5.3.1 評価関数

これまでに、相互情報量 [1]、Kay の類似度 [13] など、単語間の相関を評価するための手法がいくつか提案されている。

2 単語の相互情報量 $\mathrm{MI}(x,y)$ は次のように定義される。

$$\mathrm{MI}(x,y) = \log \frac{P(x,y)}{P(x)P(y)} \tag{5}$$

ここで $P(x,y)$ は単語 $x$ と $y$ の結合確率であり、$P(x)$ は単語 $x$ の確率である。MI は一方を知ることによって与えられる他方に関する情報の量のことであるから、この値が大きいほど両者は類似しているということができる。MI を使用すると類似度が情報量の絶対値として与えられるため、理論的に扱い易い。

しかし、上の式の分母は 2 単語の確率の積であるため、どちらかの単語の頻度が大きい時には、その値は急速に減少する性質を持ち、例え同一の単語であっても頻度が大きいと抽出され難くなってしまう。逆に、稀に出現する単語ほど高い値になる傾向が強い。これは MI が情報量として与えられることから当然予想される性質であるが、多くの場合、稀にしか現れない単語はより頻度の高い単語よりも興味深いとはいえない上に、統計量としての信頼性にも欠けるため、類似度を与える関数として使用する場合は何らかの補正を加えることが多い。また、2 単語の相互情報量を $n$ 単語に拡張することは、単純にはできない。

Kay の類似度 $\mathrm{sim}(x,y)$ は次のように定義される。

$$\mathrm{sim}(x,y) = \frac{2c(x,y)}{c(x)+c(y)} \tag{6}$$

ここで $c(x,y)$ は単語 $x$ と $y$ の共起頻度であり、$c(x)$ は単語 $x$ の頻度である。

この関数の値域は [1, 0] であり、使いやすい性質を持つが、これにも相互情報量における問題と共通する問題がある。すなわち、定義自体は頻度に影響されないようになっているにも関らず、頻度の高い単語同士の類似度が低くなる傾向がある。これは、頻度の高い単語は、結び付きの (直観的な) 強さとは無関係に、様々な単語と結び付く可能性が強いからである。

我々は連語を抽出するための相関度を評価する関数 $\mathrm{assoc}(x,y)$ を次のように定義することを提案する。

$$\mathrm{assoc}(x,y) = \frac{c(x,y)}{\min(c(x),c(y))} \tag{7}$$

ここで $c(x,y)$ は単語 $x$ と $y$ の共起頻度であり、$c(x)$ は単語 $x$ の頻度である。

この関数は、次のような仮定に基づいている。

> 「単語または単語列の組の関係は、互いに強く関連している場合であっても対称とは限らない」

この仮定のもとでは、$x$ と $y$ の組の関連が強いのは、次のいずれかの場合である。

- $x$ の出現は $y$ の出現を予測する。または、
- $y$ の出現は $x$ の出現を予測する。

この関数 $\mathrm{assoc}(x,y)$ を multi-$n$-ngram に拡張するのは簡単であり、次のように定義される。

$$\mathrm{assoc}(mn) = \frac{c(mn)}{\min_i(c(ws_i))} \tag{8}$$

ここで $c(mn)$ は multi-$n$-ngram の頻度であり、$c(ws_i)$ はその multi-$n$-ngram を構成する各 $n$-gram $ws_i$の頻度である。

## 6 実験結果

我々は、語彙数約6万語、単語数約110万語からなる英語のテキストコーパスに対して連語の抽出実験を行った。実際に抽出を行ったのは multi-$n$-ngram の特別な場合である、bi-$n$-gram である。

抽出に当たっては、計算時間を考慮して抽出する bi-$n$-gram を制限するための様々なパラメターを導入した。以下に、パラメターとそれに関与する条件を挙げる。これらのいずれかの条件に該当する bi-$n$-gram は抽出しない。また、以下で左右の $n$-gram とは bi-$n$-gram を構成する2つの $n$-gram のことを指すものとする。

**unigram の最小頻度:** 100
左右どちらかの $n$-gram が unigram で、かつその頻度がこの閾値より小さい。

**bigram の最小頻度:** 20
左右どちらかの $n$-gram が bigram で、かつその頻度がこの閾値より小さい。

**$n$-gram の最小頻度:** 10
左右どちらかの $n$-gram の頻度がこの閾値より小さい。

**$n$-gram の最大頻度:** 5000
左右どちらかの $n$-gram の頻度がこの閾値より大きい。
この制約は極めて頻度の高い単語 (例えば、"the", "of", "and" など) だけからなる unigram を排除し、計算時間を節約することを目的としている。

**$n$-gram の最小エントロピー値:** 2.0
左右どちらかの $n$-gram のエントロピーの値がこの閾値よりも小さい。

**bi-$n$-gram の最小類似度値:** 0.1
相関度評価関数 assoc(bi-$n$-gram) の値がこの閾値よりも小さい。

**bi-$n$-gram の長さの最小値:** 2
左右の $n$-gram の長さの和がこの閾値よりも小さい。

**bi-$n$-gram のギャップの長さの最大値:** 20
左右の $n$-gram の間に入る単語の数がこの閾値よりも大きい。

**bi-$n$-gram のギャップの長さの最小値:** 1
左右の $n$-gram の間に入る単語の数がこの閾値よりも小さい。
この値が0であると、より長い $n$-gram の部分列が無用に抽出されてしまう結果となる。

各制約の項目の最初の行の右端の数値はシステムの既定値であり、以下の実験結果を得るのに使用された値である。抽出処理は、コーパスのすべての文 (56401個) に対して行われた。抽出された bi-$n$-gram の数は234605であった。表1は得られた連語のうちの上位50個のものである。ただし、句読点などの記号を含んだり、意味が不明の単語を含む連語は取り除いてある。

## 7 評価

"as ... as he could" のような連語が第1位に来ているが、このような結果は Kay の類似度や、相互情報量のような対称性のある関数を用いては得られないものである。コーパスに現れる "as" の頻度は1092であったのに対して、連語 "as he could" の頻度は11であった。このように著しく頻度に差のある $n$-gram の組は、相互情報量の考え方からは非常に小さい値になるために抽出され難い。また、"as" のように様々な使われ方をする単語の場合、Kay の類似度を用いてもかなり小さい値になってしまうため、やはり相互情報量の考え方を用いた場合と同様に抽出され難くなってしまう。一方、直観的に考えて、"as ... as he could" は離散型の連語表現として妥当なものであると考えられる。本手法で提案する相関度評価関数のための非対称性の仮定が、このような連語の抽出に対して非常に有効に働くことが分かった。

## 8 結論

離散型の連語表現を自動的に抽出する手法を提案した、また、本手法で考案した相関度の評価関数が連語の抽出に有効に働くことを実験的に示した。

## 9 今後の課題

少なくとも次のような分野について、更に研究を進める必要があると考えている。

表 1: 抽出された bi-$n$-gram(上位 50 個)

| | | |
|---|---|---|
| as | ... | as he could |
| either | ... | or |
| either a | ... | or |
| as | ... | as possible |
| there | ... | be no |
| the *government of the | | |
| | ... | *states |
| there | ... | little doubt that |
| not only in | ... | but |
| dominant stress will | ... | be on |
| from time | ... | time |
| not only | ... | but |
| in the basic | ... | rate |
| not | ... | but also |
| which | ... | and which |
| not only the | ... | but |
| as | ... | as any |
| the *export-*import *bank | | |
| ... *department of *economic *affairs | | |
| of ... *united *states of *america | | |
| not | ... | but as a |
| there | ... | doubt that |
| of | ... | *states of *america |
| so | ... | as to be |
| was not only | ... | but |
| not | ... | but rather |
| as a matter | ... | fact |
| the *government of the ... of | | |
| the *export-*import *bank ... of | | |
| is not only | ... | but |
| only | ... | but also |
| not only | ... | but also |
| the *lord is | ... | my |
| but at the | ... | time |
| who | ... | and who |
| those | ... | and those who |
| what | ... | going to do |
| the *government of | | |
| ... *united *states of *america | | |
| the *government | | |
| ... *united *states of *america | | |
| government | | |
| ... *united *states of *america | | |
| one | ... | to another |
| to prevent the | ... | from |
| the | ... | *states of *america |
| the year of | ... | one |
| one | ... | one hundred and |
| hundred and | ... | one hundred and |
| hundred | ... | one hundred and |
| but if you | ... | you |
| the *government of | ... | *united *states |
| the *government of | ... | *united |
| the *government of | ... | *states |
| united | ... | of *america |

### 9.1 評価関数の改良

現在のシステムは、multi-$n$-ngram を組み立てるところでしか左右のエントロピーを用いていない。また、相関度評価関数 assoc は頻度や $n$-gram の長さといった情報を全く用いていない。より精度の高い抽出を行うためには、これらの情報を有効に用いることが要求されると思われる。

$n$-gram の長さと頻度の関係は、概ね頻度の対数が長さに反比例すると考えられる。そこで、関数 assoc を $\sum \log(c(ws_i)) * \mathrm{len}(ws_i)$ で重み付けしたものを新たな評価関数とすることが考えられる。

### 9.2 Multi-$n$-gram の抽出実験

現在のシステムは 3 つ以上の要素からなる multi-$n$-ngram を作成する部分ができていないため、実際に抽出できるのは bi-$n$-gram である。要素数が 3 以上の multi-$n$-ngram を抽出する実験をする必要がある。

### 9.3 他言語への応用

現在のシステムは単語を 2 バイトの ID で表現した英語コーパスを対象としている。しかし、本手法は特に言語に依存する部分はなく、また日本語の漢字は一字を 2 バイトで表現することができることから日本語に応用することも容易である。

### 9.4 より規模の大きい実験

現状では 100 万語のコーパスで実験を行っているが、これはかなり比較的コーパスである。さらに大きいサイズのコーパスでの実験による評価をする必要がある。

### 9.5 他の言語解析への応用

単語のクラスタリング、統語解析、統語規則の獲得など、他の言語解析への応用を考えたい。

## 参考文献

[1] Kenneth Church and Patrick Hanks. Word

association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pp. 76–83, Vancouver, Canada, 1989.

[2] Frank A. Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, Vol. 19, No. 9, pp. 143–177, 1993.

[3] Makoto Nagao and Shinsuke Mori. A new method of n-gram statistics for large number of n and automatic extraction of words and phrases from large text data of japanese. In *Proceedings of The 15th International Conference on Computational Linguistics (COLING94)*, pp. 611–615, Kyoto, Japan, 1994.

[4] 新納浩幸, 井佐原均. 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, Vol. 36, No. 1, pp. 32–40, 1995.

[5] 池原悟, 白井諭, 河岡司. 大規模日本語コーパスからの連鎖型および離散型の共起表現の自動抽出法. 情報処理学会論文誌, Vol. 36, No. 11, pp. 2584–2595, 1995.

[6] Guston H. Gonnet, Ricardo A. Baeza-Yates, and Tim Snider. New indices for text: Pat trees and pat arrays. In W.B. Frakes and Ricardo A. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 5, pp. 66–82. Prentice-Hall, 1992.

[7] Guston H. Gonnet. Unstructured data bases or very efficient text searching. In *2nd ACM SIGACT-SIGMOD Symp. PODS*, pp. 117–124, 1983.

[8] Udi Manber and Gene Myers. Suffix arrays: A new method for on-line string searches. In *1st ACM-SIAM Symposium on Discrete Algorithms*, pp. 319–327, San Francisco, 1990.

[9] Guston H. Gonnet and Ricardo A. Baeza-Yates. *Handbook of Algorithms and Data Structures in Pascal and C.* Addison-Wesley, 2 edition, 1991.

[10] Udi Manber and Ricardo A. Baeza-Yates. An algorithm for string matching with a sequence of don't cares. *Information Processing Letters*, Vol. 37, No. 3, pp. 133–136, Feb. 1991.

[11] 森信介, 長尾眞. タグ付きコーパスからの統語規則の獲得. 情報処理学会論文誌, Vol. 37, No. 9, pp. 1688–1696, 1996.

[12] 長尾眞, 森信介. 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. 情報処理学会 自然言語処理研究会 研究報告 96-1, pp. 1–8, 1993.

[13] Martin Kay and Martin Röschesen. Text translation alignment. *Computational Linguistics*, Vol. 19, No. 1, pp. 121–142, 1993.