# ウェブ閲覧における効率的なキーワード抽出とその利用

上村卓史[†]　喜田拓也[†]　有村博紀[†]

インターネットからの効率的な情報収集においてウェブブラウザの果たす役割は大きい．しかし，膨大な文書の検索および閲覧はユーザにとって未だ大きな負担である．本稿では，情報収集の補助を目的として，ユーザが閲覧するウェブページからのキーワード抽出という問題について考察する．ここでのキーワードとは，任意の単語の連結 (単語 $N$ グラム) である．このため，接尾辞木をもとにした，任意の単語 $N$ グラムを線形領域で表すデータ構造である単語 $N$ グラム木を提案し，これを利用した高速な抽出アルゴリズムを与える．また，抽出されたキーワードを利用することで，得られたキーワードの表示，関連するウェブサイトの表示，検索語の入力候補の提示といった，ユーザのウェブ閲覧を補助するインタフェースを実現する手法を示す．

# Efficient Keyword Extraction and its Applications on Web Browsing

Takashi UEMURA,[†] Takuya KIDA[†] and Hiroki ARIMURA[†]

Web browsers play an important role in information-gathering on the Internet. However, searching and browsing documents are still time-consuming jobs for many users. In this paper we present an algorithm for keyphrase extraction at the current page. The proposed algorithm utilizes a data structure, called *word $N$-gram tree*, which represents concatenations of words at most $N$ in an inputed text. We also present browsing user interfaces based on the extraction, for showing related sites, for finding summary paragraph, and for suggesting input queries for web search.

## 1. はじめに

個人のインターネットからの情報収集が日常的なものとなりつつある昨今，キーワードへのリンク付け☆やコンテンツへのタグ付け☆☆など，ウェブサービスによる情報の付加が多く試みられるようになった．これらのサービスは，ユーザに対して情報への効率的なアクセスを支援する．しかし一方で，このようなウェブサービスに対応していないページが大多数であり，ウェブブラウザは効率的な情報収集の実現に関していまだに重要な位置を占めている．

本稿では，ウェブ閲覧による情報収集を効率化するためのアプローチとして，まず任意の複合語を考慮したウェブページからのキーワード抽出手法を与え，抽出したキーワードを利用して高度なユーザインタフェースを実現する手法を提案する．

ウェブページに出現するキーワードの集合は，ページの内容を端的に表す極めて価値の高い情報であると考えられる．ウェブページの内容は様々であり，特に長い文書や難解な文書では，適切なキーワードの提示が内容理解の大きな手助けとなりうる．また，ウェブページの閲覧中には，そのページまたはウェブ全体から文字列検索を行う場面が頻繁に起こる．このとき，ウェブブラウザの画面に検索したい文字列がキーワードとして表示され，マウスのクリック等で瞬時に検索可能な状態にすることができれば，情報収集の効率化につながると考えられる．

一方で，ウェブページからのキーワード抽出にはいくつかの解決すべき問題がある．まず重要なのが高速性である．快適な閲覧を考えると，前もってキーワードが計算されている場合を除き，ユーザの閲覧に合わせて実時間で抽出可能である手法が求められる．次に，品質の問題がある．本稿では，ウェブからの情報収集におけるキーワードとして，単に名詞だけではなく，映画の題名等のような任意の単語の連接 (単語 $N$ グラム) を想定している．キーワード抽出におけるスコア付けには，出現頻度などの統計量を用いることが考えられるが，任意の単語 $N$ グラムに対する統計量を高速に求めることは困難である．また，$m$ 個の単語の連接からなる文書に対し，単語の個数が $k$ 以下の部分

† 北海道大学大学院情報科学研究科

☆ URL: http://www.hatena.ne.jp/
☆☆ URL: http://www.flickr.com/

文字列は $O(km)$ 個存在するため，単純な方法でスコアを計算すると，速度の面でも問題が生じる．

このような問題に対し，本稿では，与えられた文字列に対し，ある単語数以下の全ての単語 $N$ グラムを表す索引構造である単語 $N$ グラム木を提案し，これを用いたキーワード抽出アルゴリズムを示す．

本稿の後半では，索引構造と抽出したキーワードを利用することで，関連のあるウェブサイトの検索や重要部分の抜粋，検索における入力支援といったウェブ閲覧のための高度なインタフェースが実現できることを示す．

また，提案するキーワード抽出アルゴリズムを実装し，抽出速度に関する実験を行った結果，実時間処理に耐えうる速度でキーワード抽出が行えることを確認した．

### 1.1 関 連 研 究

キーワード抽出において広く知られたスコア付けの手法として，TF-IDF 法がある [7]．この手法は，対象文書に多く現れ，他の文書にはあまり現れないようなキーワードを高くスコア付けする．しかし，TF-IDF 法は単語を単位としており，基本的には任意の単語 $N$ グラムを扱うものではない．また，2 単語の連接頻度を考慮した手法 [6] が提案されているが，この手法は名詞および複合名詞を対象としている．

ウェブ閲覧におけるキーワード抽出および情報収集の支援としては，松尾らの研究 [4] がある．松尾らは，ユーザ個人の閲覧履歴中に良く現れる単語 (身近語) と共起するキーワードを高くスコア付けするキーワード抽出手法を提案している．この手法では，名詞，動詞，形容詞，未知語のみを扱うため，任意の単語 $N$ グラムを扱う本提案手法とは異なる．また，サーバ側での処理を前提としており，ユーザの PC 上での実時間処理は考慮していない．

文字列の索引構造の研究としては，与えられた文字列の全ての部分文字列を表す接尾辞木 [9][8][5] や，単語の先頭から始まる部分文字列を表す単語接尾辞木 [1][3]，ある単語数以下の全ての部分文字列を表す単語幅限定接尾辞木 [10] がある．単語 $N$ グラム木は，単語の先頭から始まり，かつある単語数以下である部分文字列を扱うので，これらともまた異なるデータ構造である．また，単語 $N$ グラム木はこれらの索引構造よりさらにノード数を抑えることができる．

## 2. 単語 $N$ グラム木

本節ではキーワード抽出に用いる索引構造を導入する．

アルファベットを $\Sigma$ とする．以降では $\Sigma$ は定数サイズであると仮定する．$\Sigma$ 上の文字列全体の集合を $\Sigma^*$ で表す．文字列 $x \in \Sigma^*$ について，$x$ の長さを $|x|$ と表す．特に長さが 0 の文字列を**空語**（*empty word*）といい，$\varepsilon$ で表す．$\Sigma^+$ を $\Sigma^* \setminus \{\varepsilon\}$ と定義する．文字列 $x, y \in \Sigma^*$ の連結を $x{\cdot}y$ で表す．**単語** $w$ を $w \in \{W{\cdot}\#|W \in \Sigma^+\}$ とする．$\#$ は**区切り記号**であり，$\# \notin \Sigma$ である．例えば英語のテキストにおいて，$\#$ は空白記号を表す．以降では，文字列 $S$ は単語の並び $S = w_1 \cdots w_m$ であると仮定する．このような文字列を特に**単語列**と呼ぶ．ここで，$w_m$ は $w_1 \cdots w_{m-1}$ に現れない単語とする．単語列 $S$ の**単語数**とは，$S$ 中の $\#$ の個数であり，$|S|_{word}$ と書く．**単語列 $X$ が $S = w_1 \cdots w_m$ 中に位置 $i$ で出現する**とは，$X = w_i \cdots w_j (1 \leq i, j \leq m)$ である $i, j$ が存在することをいう．また，$S$ 中における $X$ の**出現回数**とはそのような位置 $i$ の個数であり，$f(T, X)$ と書く．ただし，$X = \varepsilon$ ならば $f(T, X) = \infty$ と定義する．

ある文字列 $w \in \Sigma^*$ に対して，$w = xyz$ となる $x, y, z \in \Sigma^*$ が存在するとき，$x, y, z$ をそれぞれ $w$ の**接頭辞**（*prefix*），**部分文字列**（*substring*），**接尾辞**（*suffix*）と呼ぶ．$w$ の $i$ 番目の文字を $w[i]$ と表し，$w$ の $i$ 番目から $j$ 番目までの部分文字列を $w[i \ldots j]$ で表す．$i > j$ のとき，便宜的に $w[i \ldots j] = \varepsilon$ と定義する．また，ある単語列 $T = w_1 \cdots w_m$ に対して，$w_i \cdots w_j (1 \leq i, j \leq m)$ を $T$ の**部分単語列**と呼ぶ．ある整数 $N > 0$ に対し，文字列 $T = w_1 \cdots w_m$ の**単語 $N$ グラム集合** $WS(T, N)$ とは，単語数が $N$ 以下の $T$ の全ての部分単語列の集合，すなわち $\{w_i \cdots w_j \mid 1 \leq i \leq m, j \leq min(m, i + N - 1)\}$ である．

与えられた長さ $n$，単語数 $m$ の文字列 $T = w_1 \cdots w_m$ と整数 $N > 0$ に対し，$T$ の**単語 $N$ グラム木**とは，$T$ の単語 $N$ グラム集合 $WS(T, N)$ を表す圧縮トライ $WST(T, N) = \{V, root, E, f\}$ である．$V$ はノードの集合である．$E \subseteq V^2$ は辺の集合である．ある 2 つのノード $u, v \in V$ に対し，辺 $(u, v) \in E$ が存在するとき，$u$ を $v$ の**親**，$v$ を $u$ の**子**と呼ぶ．同様に，$u$ から $v$ への道が存在するとき，$u$ を $v$ の**祖先**，$v$ を $u$ の**子孫**と呼ぶ．任意のノード $v \in V$ は，子を 2 つ以上持つ**分岐ノード**か，子を 1 つも持たない**葉**のどちらかである．辺 $e \in E$ には $T$ の部分文字列 $T[i \ldots j] (1 \leq i, j \leq n)$ が**ラベル**として割り当てられる．ラベルは $T$ 中の位置を参照する組 $(i, j)$ で表される．ノード $v \in V$ から $v$ の子 $u$ へ向かう辺はそれぞれ異なる文字で始まるラベルを持つ．したがって，子 $u$ へ向かう辺のラベルを $label(u)$ で表す．

**ノード** $v \in V$ **が表す文字列**とは，根から $v$ への道 $(v_0 = root, v_1), \ldots, (v_{k-1}, v_k = v)$ の各辺のラベルを連結した文字列 $label(v_1) \cdots label(v_k)$ の，# で終わる最長の接頭辞であり，これを $\overline{v}$ で表す．任意の葉は $\{w_i \cdots w_j \mid 1 \leq i \leq m, j = min(m, i+N-1)\}$ の要素のいずれかを表す．すなわち，葉は高々$m$ 個しか存在しない．任意のノード $v \in V$，$v$ の親 $u$，$v$ の接頭辞である任意の単語列 $x$ について，$|\overline{u}|_{word} < x < |\overline{v}|_{word}$ であるとき，$x$ **は辺** $e$ **に属する**という．任意のノード $v \in \{root\} \cup V$ に対し，$freq$ は，$T$ 中の $\overline{v}$ の出現回数 $f(T, \overline{v})$ を割り当てる．任意のノード $v \in V$ は $T$ の部分単語列を表すから，$v \in V$ に対し $freq(v) > 0$ であることに注意する．

葉は高々$m$ 個であり，葉でないノードは分岐ノードであるから，根を含めたノードの個数は高々$2m-1$ 個である．また，ノードはラベルを表すための位置のペアを保持するから，1 つあたり $O(\log n)$ 領域を要する．従って次の定理が成り立つ．

**定理 1**　長さ $n$，単語数 $m$ の文字列 $T$ と，整数 $N > 0$ に対し，単語 $N$ グラム木は $O(m \log n)$ 領域で構築できる．

また，単語 $N$ グラム木の各ノードが表す文字列の出現回数に関し，次の 2 つの補題が成り立つ．

**補題 1**　文字列 $T$ の単語 $N$ グラム木を $WST(T, N)$ とする．任意のノード $v \in V$ とその親 $u$ について，$|\overline{u}|_{word} < |\overline{v}|_{word}$ ならば $freq(u) > freq(v)$ が成り立つ．

[証明] 定義より $u$ は分岐ノードであり，$v$ でない子を持つ．従って $v$ でない $u$ の子孫の葉 $w$ が存在する．$label(w)$ は # で終わる文字列であるから，$|w|_{word} > |u|_{word}$ であり，$|v|_{word}$ の接頭辞ではない単語列である．また $\overline{u}$ は $\overline{v}$ と $\overline{w}$ の共通の接頭辞であるから，少なくとも $freq(v) + freq(w)$ 回 $T$ に出現する．ここで任意のノード $p \in V$ に対し $freq(p) > 0$ が成り立つから，$freq(u) \geq freq(v) + freq(w) > freq(v)$ が成り立つ．(証明終わり)

**補題 2**　文字列 $T$ の単語 $N$ グラム木を $WST(T, N)$ とする．任意のノード $v \in V$ とその親 $u$，辺 $e = (u, v)$ に属する任意の単語列 $X$ に対して，$f(T, X) = freq(v)$ が成り立つ．

[証明] $X$ は $\overline{v}$ の接頭辞であるから，少なくとも $freq(v)$ 回 $T$ に出現する．ここで $f(T, X) > freq(v)$ であると仮定すると，$X$ を接頭辞とし，$\overline{v}$ を接頭辞としない $T$ の部分単語列 $Y$ が少なくとも 1 つ存在する．このとき，$Y$ を接頭辞とする葉 $w \in V$ が存在するから，$X$ を接頭辞とする単語列を表す，$v$ と $w$ の共通の祖先 $p$ が存在する．しかしそのような $p$ が存在するならば，$u$ と $v$ の道の間に位置するから，$e$ は $p$ によって 2 つの辺に分割される．よって矛盾である．以上より $f(T, X) = freq(v)$ が成り立つ．(証明終わり)

$T$ = AB#BC#AC#BC#$#, $N$ = 2

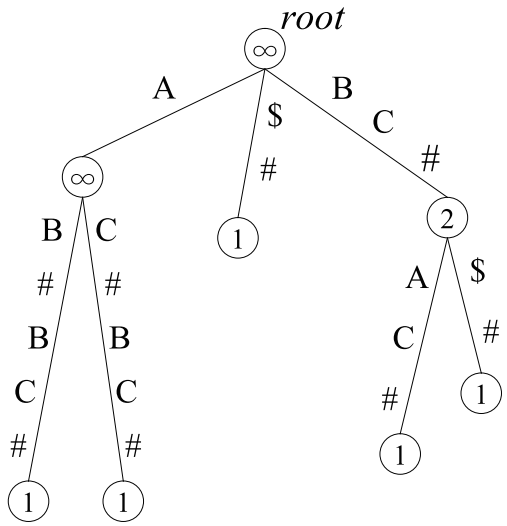


**図 1**　入力テキスト $T = AB\#BC\#AC\#BC\#\$\#$ に対する単語 2 グラム木．

文字列 $T$ =AB#BC#AC#BC#$# に対する単語 2 グラム木を図 1 に示す．ノード $v$ の中に記された数字は $f(v)$ を示す．なお，$N = \infty$ とすると，単語の先頭から始まる $T$ の全ての接頭辞を表すデータ構造である単語接尾辞木 [3] と同形の木となる．

## 3. キーワード抽出

### 3.1 入力テキスト

キーワード抽出する対象の文字列を**入力テキスト**と呼ぶ．本稿で提案するキーワード抽出アルゴリズムは，単語間に明示的な区切りを持つことを前提としている．また，ウェブページには広告等の不要な要素が多く含まれるため，以下の処理を施したものを入力テキスト $T$ とする．

今，ウェブページのテキスト部分を $d$ とする．**段落**とは意味上のつながりを持った $d$ の部分文字列である．本稿では簡単のため，ウェブページを HTML 文書と仮定し，各ブロック要素を段落とする．そして，$d$ の各段落のうち，句読点の個数の割合がある閾値 $\delta_0 (0 < \delta_0 < 1)$ 以上である段落のみを残し，連結する．最後に，日本語のように単語の区切りが明示的になされていない場合，形態素解析し分かち書きを行う．以上の処理を行って得られる文字列を入力テキスト $T$

**表 1** 品詞スコアの例

| 品詞 | 品詞スコア |
|---|---|
| 名詞 | 10 |
| 名詞, 接尾 | 1 |
| 名詞, 数 | 1 |
| 名詞, 代名詞 | 0 |
| 名詞, 非自立 | 1 |
| 形容詞 | 2 |
| 形容詞, 接尾 | 1 |
| 形容詞, 非自立 | 1 |
| 動詞 | 2 |
| 動詞, 接尾 | 1 |
| 動詞, 非自立 | 1 |
| 助動詞 | 1 |
| 助詞 | 1 |
| 副詞 | 1 |
| 接頭詞 | 1 |
| 接続詞 | 0 |
| 連体詞 | 0 |
| フィラー | 0 |
| 感動詞 | 0 |
| 記号 | 0 |
| その他 | 0 |

とする.

### 3.2 キーワード候補のスコア付け

**キーワード候補**とは，長さ $m$ の入力テキスト $T = w_1 \cdots w_m$ の空でない部分単語列 $X = w_i \cdots w_j (1 \leq i \leq j \leq m)$ である.

まず，任意の単語 $x \in \{w \cdot \#|w \in \Sigma^+\}$ に対し，$x$ の品詞によって**品詞スコア** $C(x)$ を定義する．任意の単語に対し，$C(x) \geq 0$ であると仮定する．一般には名詞がキーワードとなりやすいので，品詞スコアは名詞を高く，助詞などを低く設定する．表 1 に品詞スコアの例を示す．ここで,「名詞」,「名詞，接尾」のように複数当てはまる場合は，最も長く一致したものの品詞スコアを採用することにする.

次に単語 $x$ に対し，$x$ の**単語スコア** $W(x)$ を，品詞スコアと $x$ の長さの積で

$$W = C(x) \cdot (|x| - 1)$$

と定義する．ただし，例えば漢字とひらがなでは 1 文字が表す意味の量が違うので，文字種を考慮し，単語 $x$ の長さ $|x|$ は，例えば次のように適宜補正する.

**例 1** 漢字，ひらがな，カタカナ，英数字に対し，1 文字当たりの長さをそれぞれ 4,2,2,1 とする.

単語スコアを用いて，任意のキーワード候補 $X = x_1 \cdots x_k$ に対する**単語列スコア** $P(X)$ を，$X$ に含まれる各単語の単語スコアの和で

$$P(X) = \sum_{i=1}^{k} W(x_i)$$

と定義する．品詞スコア，単語スコア，単語列スコアは，その文字列のみによって定まることに注意する.

以上で定義される単語列スコアと，キーワード候補の $T$ 中の出現回数より，キーワード候補 $X$ に対する最終的なスコア $PF(X)$ を

$$PF(X) = P(X) \cdot \log(f(T, X))$$

で定義する．以降は混乱のない限り $PF(X)$ を単に $X$ の**スコア**と呼ぶ.

ここで，任意の品詞スコアは非負であることから，次の補題が成り立つ.

**補題 3** 任意の単語列 $X$ と，$X$ の $\#$ で終わる接頭辞 $X'$ について，$P(X) \geq P(X')$ が成り立つ.

補題 3 より，キーワード候補 $X$ と，$X$ の自身より短い任意の接頭辞 $X'$ に対し，$f(T, X) = f(T, X')$ ならば $PF(X) \geq PF(X')$ であることが直ちにわかる．このような $X$ が存在する $X'$ は $X$ **に代表される**という．よって，入力テキスト $T = w_1 \cdots w_m$ の $N$ **単語以下のキーワード候補集合** $K(T, N)$ を，$X$ を接頭辞とする $T$ の部分単語列 $X'$ について，$|X|_{word} > |X'|_{word}$ かつ $f(T, X) = f(T, X')$ となる $T'$ が存在しない $N$ 単語以下の全ての $T$ の部分単語列 $X = x_i \cdots x_j (1 \leq i \leq j \leq m)$ の集合であると定義する.

すなわち，本稿におけるキーワード抽出とは，次のような問題である.

**単語 $N$ グラムキーワード抽出問題**: 与えられた入力テキスト $T$ に対し，$N$ 単語以下のキーワード候補集合のうち，上位 $k$ 個をスコアの降順で出力せよ.

### 3.3 単語 $N$ グラム木を用いたキーワード抽出アルゴリズム

単語 $N$ グラム木を用いて，各キーワード候補のスコアを計算し，上位のキーワードを列挙するアルゴリズムを示す．次の 2 つの補題は本稿で提案するアルゴリズムの正当性について重要である.

**補題 4** 入力テキストを $T = w_1 \cdots w_m$，抽出するキーワードの最大単語数を $N$ とする．単語 N グラム木のノード $v \in V$ の中で，$v$ の親 $u$ に対し $|\overline{v}|_{word} > |\overline{u}|_{word}$ である全ての $v$ の集合を $V'$ とする．このとき，$T$ の $k$ 単語以下の任意のキーワード候補 $W \in K(T, N)$ に対し，$W = \overline{v}$ であるノードが $V'$ に存在する.

[証明] 単語 $N$ グラム木 $WST(T, N)$ は $T$ の全ての $N$ 単語以下の部分単語列を表すから，全てのキー

ワード候補はノードによって表されるか，辺に属するかのどちらかである．ここでまず，キーワード候補 $W = w_i \cdots w_j$ が辺 $e = (u, v)$ に属すると仮定する．補題 2 より $f(T, W) = \mathit{freq}(v)$ であるから，$W$ は $\overline{v}$ に代表される．従って $W$ はキーワード候補集合の要素の定義を満たさないから，矛盾である．よって $W$ はあるノード $v \in V$ によって表される．次に $W$ を表すノードが $V'$ に存在しないと仮定する．このとき，$W = \overline{v}$ となるノードが $v \in V \setminus V'$ に存在する．ところが，$|W|_{word} > 0$，$|\overline{root}|_{word} = |\varepsilon|_{word} = 0$ であるから，$root$ から $v$ への道の間に，$\overline{p} = W$ となるような，$v$ を含む全てのノード $p$ の道 $p_1, \ldots, p_l (p_1, \ldots p_l \in V)$ が存在する．ここで $p_1$ の親を $p_0$ とすると，$|\overline{p_1}|_{word} > |\overline{p_0}|_{word}$ が成り立つ．これより $p_1 \in V'$ となり，矛盾である．よって，任意のキーワード候補 $W$ に対し $W \in K(V')$ が成り立つ．(証明終わり)

**補題 5**　入力テキストを $T = w_1 \cdots w_m$ とする．抽出するキーワードの最大単語数を $N$ とする．単語 $N$ グラム木のノード $v \in V$ の中で，$v$ の親 $u$ に対し $|\overline{v}|_{word} > |\overline{u}|_{word}$ である全ての $v$ の集合を $V'$ とする．このとき，任意のノード $v \in V'$ に対し，$\overline{v} = W$ となるキーワード候補 $W \in K(T, N)$ が存在する．

[証明] $\overline{v} \notin K(T, N)$ であると仮定する．定義より，$v$ は $T$ の部分単語列を表すから，$v$ を接頭辞とし，$|\overline{v}|_{word} > |X|_{word}$ かつ $\mathit{freq}(v) = f(T, X)$ となる $T$ の $N$ 単語以下の部分単語列 $X$ が存在する．このとき $\overline{p} = X$ となる $v$ 以外のノード $p$ が存在する．しかし，補題 1 より，$|\overline{v}|_{word} < |\overline{p}|_{word}$ ならば $\mathit{freq}(v) > \mathit{freq}(p)$ であるから，矛盾である．よって，全てのノード $v \in V'$ について $\overline{v} \in K(T, N)$ が成り立つ．(証明終わり)

効率的なスコア計算のため，単語 $N$ グラム木の構築時に，各ノード $v \in V$ に $|\overline{v}|_{word}$，$P(\overline{v})$ を保持しておくことにし，これらをそれぞれ $\mathit{WordCount}(v)$，$\mathit{PhraseScore}(v)$ で表す．全てのキーワード候補のスコアを計算するには，単語 N グラム木を構築し，全てのノードを深さ優先探索でたどり，各ノード $v \in V'$ について $\mathit{PhraseScore}(v) \cdot \log(\mathit{freq}(v))$ を計算すればよい．スコアの上位 $k$ 個のキーワード候補をキーワードとして出力するならば，次のようにすることで効率的に計算できる．まず，最大 $2k$ 個のキーワード候補を保持するリストを用意する．適宜キーワード候補をリストに追加し，要素数が $2k$ 個となる度にスコアの降順でソートして上位 $k$ 個を残す．最後に残った全要素をソートし，上位 $k$ 個を出力する．

以上のキーワード抽出手続き **KeyphraseExtraction** を図 2 に示す．　補題 4，補題 5 より，本論文の主結果である次の定理が示される．

**手続き KeyphraseExtraction** $(T, N, WST(T, N), k)$;
**入力**: 長さ $n$，単語数 $m$ の入力テキスト $T$，最長単語数 $N$，
　　$T$ の単語 $N$ グラム木 $WST(T, N)$，抽出個数 $k$;
**出力**: $T$ の $N$ 単語以下のキーワード上位 $k$ 個;
1 $L = \emptyset$;
2 $WST(T, N)$ の根を除くノードを深さ優先探索し以下を行う:
3　　$v$ = 対象のノード，$u = v$ の親;
4　　**if** $\mathit{WordCount}(v) > \mathit{WordCount}(u)$
5　　　　$PF(\overline{v}) = \mathit{PhraseScore}(v) \cdot \log(\mathit{freq}(v))$;
6　　　　$L = L \cup \{\overline{v}\}$;
7　　　　**if** $L$ の要素数 $= 2k$
8　　　　　　$L$ をソートし，スコアの上位 $k$ 個を残す;
9　　**end if**
10 **end**
11 $L$ をソートし，スコアの上位 $k$ 個を残す;
12 **return** $L$;

**図 2**　キーワード抽出の手続き

**定理 2**　与えられた単語数 $m$ の入力テキスト $T$，最大単語数 $N$，抽出個数 $k$，$T$ の単語 $N$ グラム木 $WST(T, N)$ に対し，図 2 の手続き **KeyphraseExtraction** は，単語 $N$ グラムキーワード抽出問題を $O(m \log k)$ 時間で正しく計算する．

[証明] 補題 4，補題 5 より正当性は明らかである．計算時間に関して，まず 1 行目は初期化であり $O(1)$ 時間で行える．定理 1 より，ノードの個数は $O(m)$ 個である．深さ優先探索により，各ノード $v \in V$ について，リスト $L$ の操作を除くと，$O(1)$ 時間でスコアを計算することができる．$\mathit{WordCount}(v)$ および $\mathit{WordCount}(u)$ はノードに保持されているから，4 行目の判定は $O(1)$ 時間で行える．また，各ノードは単語列スコアと出現回数を保持しているから，5 行目のスコアの計算も $O(1)$ 時間で行える．従って，6 行目のリストへの追加と 7，8 行目のリストのソートを除くと，2 行目から 10 行目の全てのノードに対するスコア計算は $O(m)$ 時間で行える．

2 行目から 10 行目の 1 回のループにつき，$L$ の要素数は高々1 増える．よって，8 行目でソートする回数は高々$\lfloor \frac{m}{2k} \rfloor$ 回である．7 行目の判定は $O(1)$ 時間で行え，1 回のソートは，例えばヒープソートを用いることで $O(2k \cdot \log(2k))$ 時間で行える．また，下位 $k$ 個を削除するのは $O(k)$ 時間で行える．よってソートにかかる時間は 2 行目から 10 行目のループ全体で $O(m \log(2k))$ 時間である．最後に 11 行目で高々$2k$ 個の要素をソートし，下位の高々$k$ 個を削除するのは，$O(2k \cdot \log(2k))$ 時間で行える．

以上より，図 2 の手続き **KeywordExtraction** の

計算時間は $O(m \log k)$ 時間である．(証明終わり)

### 3.4 キーワード抽出結果の選別

前節の抽出アルゴリズムでは，抽出されたキーワード間には同じ単語が複数現れる場合があり，ユーザにとっては冗長である．しかし，例えば同じ単語を二度出力しないということにすると，本来出力するべきであったキーワードが失われてしまう可能性がある．そこで，スコアが上位のキーワードから順に以下の処理を行い，冗長なキーワードを排除することで一覧性を向上させる．

$L$ を単語の集合とする．初期状態では，$L$ は空とする．キーワード $X = w_i \cdots w_j$ について，単語の多重集合 $\overline{L}_X = \{w_l | i \leq l \leq j, w_l \notin L\}$ を求める．ここで，$\overline{L}_X$ の要素数を $|\overline{L}_X|$ と置く．このとき，ある閾値 $\delta_1, \delta_2 (0 < \delta_1, \delta_2 < 1)$ に対し，

$$\frac{|\overline{L}_X|}{|X|_{word}} \geq \delta_1$$

かつ

$$\frac{\sum_{w \in \overline{L}_X} W(w)}{\sum_{n=i}^{j} W(w_n)} \geq \delta_2$$

であるならば，$X$ をキーワードとして出力し，$X$ に含まれる全ての単語を $L$ に追加する．以上の処理を上位のキーワードから順に行う．

表 2 に，青空文庫[☆]より入手した芥川龍之介の「鼻」の本文に対するキーワード抽出結果と，$\delta_1 = \delta_2 = 0.5$ として選別したときの上位 20 個のキーワードを示す．ただし，形態素解析に MeCab0.96[☆☆]を，品詞スコアに表 1 の値を，文字種による長さの補正に例 1 の値を用いた．また，$\delta_1, \delta_2$ は予備実験により求めた値を用いた．選別前は長いキーワードの部分単語列が多く現れてるが，選別後は,「弟子の僧」と「池の尾」や,「内供」と「禅智内供」のように，単語の重複を許しつつ明らかな重複を取り除くことができており，自然な結果が得られているといえる．

### 3.5 閲覧履歴の利用

ウェブ上で閲覧する文書には，極端に短いものも存在する．このような短い文書においては，キーワードの出現回数のような統計情報が信頼できない．また，一度しか現れない複合語が多く存在し，単語の切れ目を推定することも困難である．このような問題に対し，本稿では，ユーザの直近の閲覧履歴を利用することでキーワードの出現頻度を補正する手法を提案する．この手法の基本的なアイディアは，ユーザは関連性のある一連の文書を閲覧しているという仮定の下，関連性のある文書によく出現する文字列を重要視する，というものである．

**表 2** 芥川龍之介「鼻」に対するキーワード抽出結果上位 20 個と選別後の結果上位 20 個

| 選別前のキーワード | 選別後のキーワード |
|---|---|
| 弟子の僧 | 弟子の僧 |
| 弟子の僧の | 内供 |
| 内供 | 鼻 |
| 鼻 | 禅智内供 |
| 供 | 自分 |
| 弟子 | 池の尾 |
| 弟子の | 顔 |
| 弟子の僧は | 中童子 |
| 内供は | 上唇の上から顋の下まで |
| 供は | 木の片 |
| 鼻を | 法 |
| 弟子の僧が | 寺 |
| 禅智内供 | 湯 |
| 自分 | 眼 |
| 池の尾 | 手 |
| 僧 | 鏡 |
| の僧 | 心もち |
| 鼻の | 自尊心 |
| 内供の | 鼻を粥の中へ落した |
| 供の | 気に |

入力テキストを $T$ とする．**閲覧履歴**とは $T$ を閲覧する以前に閲覧した文書から生成した入力テキストの集合であり，これを $D$ とおく．直近に閲覧した文書ほど関連が深いと考えられるため，$D$ の要素数は定数個程度と仮定する．任意のキーワード候補 $W$ に対し，$W$ の $D$ 中の全ての入力テキスト中の合計の出現回数を $f(D, W)$ と書く．このとき，$f(T, W)$ を次のように補正する．

$$f'(T, W, D) = f(T, W) \cdot \{1 + \ln(f(D, W))\}$$

ここで，$W$ がある葉 $v$ によって表され，かつ $T$ 中に一度しか現れない単語列であるとき，$W$ は本来抽出されるべきキーワードを接頭辞とする部分単語列であり，適切なキーワードが得られない場合が考えられる．そこで，一旦 $W$ をキーワード候補から除外する．次に，$W'$ を $f(D, W') > 1$ である最長の $W$ の接頭辞とする．ここで，$W = W'$ であるか，または $W'$ が $v$ に向かう辺に属しているならば，$W'$ を新たなキーワード候補とする．

短い文書からのキーワード抽出に対する閲覧履歴の利用について，理想的な閲覧履歴が得られた場合の例

☆ URL: http://www.aozora.gr.jp/
☆☆ URL: http://mecab.sourceforge.net/

**表 3** 「吾輩は猫である」のあらすじ (139 字) に対する，閲覧履歴の利用によるキーワード出現回数の補正の例.

| 単独で抽出 | | 閲覧履歴を利用して抽出 | | | 本文から抽出 | |
|---|---|---|---|---|---|---|
| キーワード | $f$ | キーワード | $f$ | $f'$ | キーワード | $f$ |
| 主人 | 3 | 苦沙弥先生 | 1 | 3.833 | 苦沙弥先生 | 24 |
| する | 2 | 主人 | 3 | 23.22 | 武右衛門 | 27 |
| と | 4 | の人 | 1 | 4.219 | 迷亭先生 | 34 |
| に | 2 | 人間は | 1 | 2.792 | 主人 | 839 |
| を | 2 | 門下生 | 1 | 2.386 | 天然居士 | 21 |
| | | 結婚 | 1 | 2.609 | 独仙 | 48 |
| | | の娘 | 1 | 3.485 | 人間 | 257 |
| | | 南無阿弥陀仏と | 1 | 1.693 | 寒月 | 215 |
| | | 猫の | 1 | 2.946 | 細君 | 191 |
| | | 寒月と | 1 | 2.099 | 首縊りの力学 | 11 |

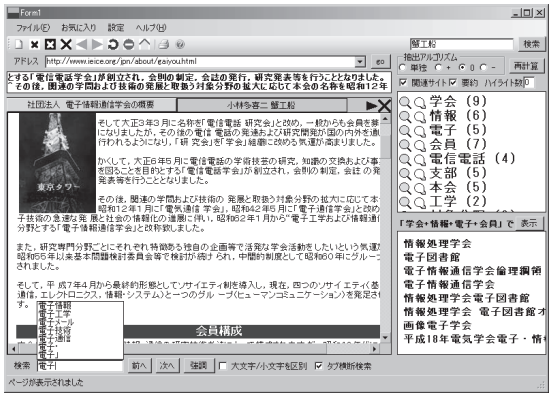

**図 3** 実装したウェブブラウザのスクリーンショット．ページ上部にスニペット，右上にキーワード，右下に関連サイト，左下に検索語入力候補を表示している．

を示す．表 3 に，三省堂のウェブサイト☆ に記載されている，夏目漱石の「吾輩は猫である」のあらすじを入力テキストとしたときの，単独でのキーワード抽出結果と，閲覧履歴として青空文庫より入手した「吾輩は猫である」本文を用いたときの結果，さらに，参考として「吾輩は猫である」本文から抽出した結果を示す．抽出の条件は表 2 と同様である．入力テキストは 139 字のごく短い文章であるため，一度しか現れない単語が多く，単独での抽出ではうまく抽出ができず，5 個のみの抽出となっている．閲覧履歴を利用した場合,「の人」等，区切りの適切でないキーワードが含まれている一方,「苦沙弥先生」のように，あらすじ中に一度しか現れない複合語が取り出せている．このように，閲覧履歴が抽出対象の文書とよく関連しているときには，短い文書からのキーワード抽出品質を改善することができる．

## 4. キーワードを利用したインタフェース

前節のアルゴリズムで抽出したキーワードは，例えばマウスによるクリックで即座に検索可能な状態にして一覧表示することで，検索語の入力の手間を省略することができる．本節では，抽出したキーワードを利用してさらに二次的な処理を加えることで，情報収集を補助するインタフェースを実現する手法を提案する．また，以下に紹介するインタフェースをウェブブラウザに実装し，実際にウェブ閲覧を行ったときのスクリーンショットを図 3 に示す．表示しているのは電子情報通信学会の概要ページ☆☆ である．

☆ URL: http://tb.sanseido.co.jp/kokugo/kokugo/dokusho/j-tb/writer10.html

☆☆ URL: http://www.ieice.org/jpn/about/gaiyou.html

### 4.1 関連サイト表示

上位のキーワードをいくつか用いてウェブから検索を行うことで，内容の関連性が深いと考えられるウェブサイトを抽出する．これを**関連サイト**として表示する．検索に用いるキーワードは，4 個以下かつキーワード中の単語の総数が 15 以下となるように，上位から順に選択する．

### 4.2 文 書 要 約

文書中で重要な部分の抜粋を**スニペット** *(snippet)* と呼ぶ．本稿では上位のキーワードを多く含む部分が重要であるという考えの下，この重要さを数値化してスニペットとして抜粋する部分を計算する．

まず上位のキーワードに対し点数を与える．ただし計算の簡潔さのため，キーワードに含まれる各単語に点数を与えることでこれに代える．ここでは 1 位のキーワードから順に 5,4,3,2,1 という点数を各キーワード中に含まれる単語に与える．ただし，複数の順位のキーワードに含まれる単語は，より高い点数を選択する．以上で点数が与えられなかった単語には 0 を与える．各単語 $w$ に与えた点数を $s(w)$ とする．入力テキスト $T = w_1 \cdots w_m$ の任意の部分単語列 $X = w_i \cdots w_j (1 \leq i, j \leq m)$ に対し，$X$ のスニペットとしてのスコア $S(X)$ を

$$S(X) = \sum_{k=i}^{j} s(w_k)$$

と定義する．$T$ の長さ $l$ のスニペットとは，長さが $l$ 以下の $T$ の部分単語列 $X$ で $S(X)$ が最大のものである．$T$ に対し幅 $l$ のスライド窓を考えると，先頭の単語を削除し，可能な限り末尾に新しく単語を付け加えることで，テキスト長に比例した時間でスコアが最大の部

分文字列をスニペットとして抽出することができる．

### 4.3 検索語入力支援

単語 $N$ グラム木は接尾辞木と同様，文字列の高速な検索と，後に続く文字列の列挙が可能である．この利点を生かし，ユーザが検索語を入力したとき，その直後に続く単語を列挙し，入力候補として提示する機能を実現する．

今，ユーザが文字列 $X$ を入力したとする．簡単のため，$X$ は単語列 $X = x_1 \cdots x_k$ であると仮定する．$T$ 中で $X$ の後に続いて現れる単語は，単語 $N$ グラム木を用いることで容易に列挙することができる．まず木の根からたどり，$X$ に対応するノードまたは辺を求める．求めた位置から下に続く部分木が，$X$ の後に続く文字列の集合を表す木である．この部分木を探索し，$X$ より 1 単語長い文字列と，そのキーワードスコアを計算する．得られた文字列をスコアの降順にソートし，上位を入力候補として出力する．

## 5. 実　　験

本節では，提案するキーワード抽出アルゴリズムを実装し，計算機実験を行う．コンパイラは Visual C++ .NET 2005 付属のものを用いた．実験用のシステムは，CPU が Pentium M 1.3GHz，メモリが 1GB，OS が Windows XP Professional のノート PC である．今回はウェブブラウザに組み込んで実時間処理を実現することを主眼に置いたため，計算速度についての実験結果のみを示す．

### 5.1 デ ー タ

データとして，青空文庫のウェブサイトより紫式部の「源氏物語」の与謝野晶子による現代語訳の本文からルビを除いたものを用意した．これを MeCab0.96 を用いて形態素解析し，空白記号で分かち書きをしたものを UTF-8 でエンコーディングすることで入力テキストを生成した．空白を除く文字数は約 88 万文字であり，ファイルサイズは約 3M バイトである．今回実装したプログラムでは 1 バイト単位でテキストを扱うため，以降ではテキスト長を入力テキストのバイト数とする．

### 5.2 方　　法

#### 5.2.1 実　験　1

入力テキストに対し，キーワードの最大単語数を 1 から 20 まで変化させ，索引の構築時間，キーワード抽出の時間，索引のノード数を測定する．時間に関しては 10 回の実行の平均値をとる．抽出する個数は 30 とする．その他の条件は表 2 と同様とする．キーワードの計算が完了した時点までを抽出の時間とし，出力にかかる時間は含めないものとする．

#### 5.2.2 実　験　2

最大単語数を $\infty$ とし，入力のテキスト長を 100k バイトから 3M バイトまで変化させ，索引の構築時間，キーワード抽出の時間を測定する．その他の条件は実験 1 と同様である．

### 5.3 結　　果

#### 5.3.1 実　験　1

ノード数の変化のグラフを図 4 に，計算時間の変化のグラフを図 5 に示す．

図 4 では，最大単語数に伴ってノード数が増えていく様子がわかる．4 単語目あたりまでは急激にノード数が増え，それ以降は収束に向かっている．なお，最大単語数が 18 のときはじめて，最大単語数をテキスト長としたときと同じノード数となった．すなわち，このときの抽出結果は最大単語数を $\infty$ と等しくとったときと一致する．

図 5 では，最大単語数の増加に伴い実行時間が増えているが，$N = 8$ 程度で収束している．また，計算時間は索引構造の構築が占める割合が大きい．なお，形態素解析等による入力テキストの生成にかかる時間が約 2.2 秒であったので，$N = \infty$ のとき，キーワード抽出全体で 5.5 秒程度の時間を要する．

これらの結果から，最大単語数を小さくとると，計算時間およびメモリ領域を節約できることがわかる．従って，自然言語処理において単語バイグラムや単語トライグラムに対する統計量を計算する際，効率的な索引構造として適用可能である．また，最大単語数を $\infty$ としても，現実的な計算時間でキーワード抽出が可能であるといえる．なお，本実験による実装では 1 ノードあたり 28 バイトを消費するので，$N = 2, 3, \infty$ のとき，メモリ領域はそれぞれ 1 文字あたり 1.4 バイト，3.8 バイト，8.1 バイト程度必要となる．

#### 5.3.2 実　験　2

結果を図 6 に示す．この結果から，テキスト長にほぼ比例した時間で処理していることがわかる．一般的なウェブページでは，文書の長さは数百から数万字程度である．実験に用いたシステムはごく一般的なノート PC であり，1 秒間に 1MB 程度処理可能であることを考えると，提案手法によって十分実時間でのキーワード抽出が可能であるといえる．

## 6. おわりに

本稿では，与えられた文字列に対し任意の $N$ 以下の全ての単語 $N$ グラムを表すデータ構造である単語 $N$ グラム木を提案した．また，単語 $N$ グラム木を用

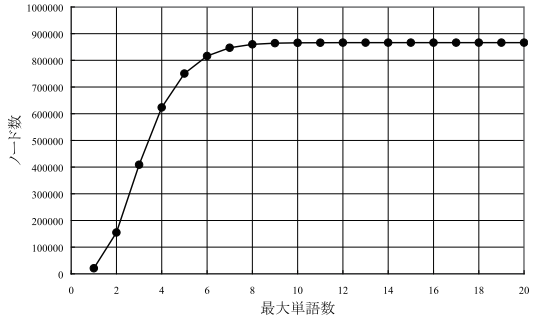


**図 4** 抽出キーワードの最大単語数に対する単語 N グラム木のノード数

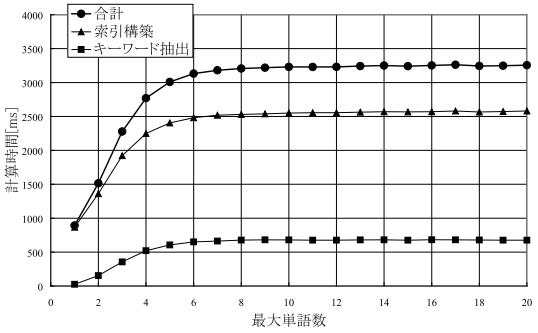


**図 5** 抽出キーワードの最大単語数に対する計算時間

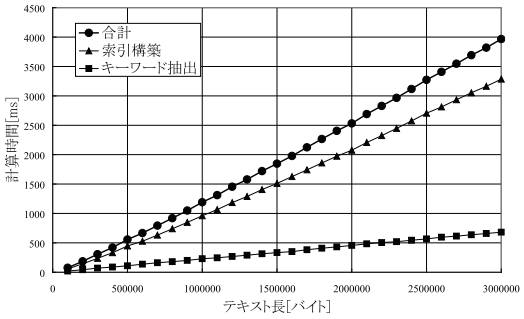


**図 6** テキスト長に対する計算時間

いて，任意の単語 $N$ グラムをキーワードとして抽出するアルゴリズムを提案した．さらに，これらを利用して実現する機能を用いて，ウェブ閲覧を補助するインタフェースを提案した．計算機実験では，提案したアルゴリズムが実際に実時間でのキーワード抽出に適用可能であることを確認した．

今後の課題としては，まずキーワード抽出品質について，他手法と比較しての客観的な評価が必要である．また，応用として挙げた各機能についても，品質と速度，有用性に関して他手法と比較して評価をする必要がある．

単語 $N$ グラム木を用いたキーワード抽出アルゴリズムでは，各キーワードに対し，単語に対するスコアの総和で全体のスコアが与えられる形であれば，単語スコアは任意に定めることができる．従って，単語に対して他の尺度を提案手法に適用した場合について，その品質と速度の評価を行っていくことも今後の課題である．

## 謝　　辞

本研究の一部は IPA 2006 年度下期未踏ソフトウェア創造事業「未踏ユース」の支援を受けて行われた．

## 参 考 文 献

1) A. Andersson, N.J. Larsson, and K. Swanson, “Suffix trees on words, ” 7th Annual Symposium on Combinatorial Pattern Matching, Laguna Beach, USA, June 1996 .
2) J.C. Na, A. Apostolico, C.S. Iliopoulos, and K.Park, “Truncated suffix trees and their application to data compression, ” Theoretical Computer Science, vol.304, 87–101, July 2003.
3) S.Inenaga and M.Takeda, “On-line linear-time construction of word suffix trees, ” Proc.of 17th Ann. Symp. on Combinatorial Pattern Matching, LNCS4009, pp. 60–71, 2006.
4) 松尾豊，福田隼人，石塚満，“ユーザ個人の閲覧履歴からのキーワード抽出によるブラウジング支援”，人工知能学会誌，Vol.18，No.4E，pp.203-211，2003.
5) E.M. McCreight, “A space-economical suffix tree construction algorithm, ” J. ACM, 23:262–272, 1976.
6) 中川裕志，森辰則，湯本紘彰，“出現頻度と連接頻度に基づく専門用語抽出”，自然言語処理，Vol.10 No.1，pp. 27 - 45，2003.
7) G. Salton, A. Wong, and C. S. Yang, A Vector Space Model for AutomaticIndexing, Communications of the ACM, 18(11):613.620, 1975.
8) E. Ukkonen, “On-line construction of suffix trees, ” Algorithmica, vol.14, no.3, 249–260, 1995.
9) P. Weiner, “Linear pattern matching algorithms, ” Proc. IEEE 14th Annual Symposium on Switching and Automata Theory, pp. 1–11, 1973.
10) 上村卓史，喜田拓也，有村博紀，“単語幅を制約した接尾辞木の効率のよい構築アルゴリズム, ” 情報科学技術レターズ (第 5 回情報科学技術フォーラム講演論文集)，vol.5, 5–8，2006.