情報処理学会論文誌　データベース　Vol. 1　No. 1　49–60 (June 2008)

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[†1]　喜田拓也[†1]　有村博紀[†1]

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いまだ大きな負担である．本論文では，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 to Web Browsing

Takashi Uemura,[†1] Takuya Kida[†1]
and Hiroki Arimura[†1]

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 keyword extraction at the current Web page. The proposed algorithm utilizes a data structure, called *word N-gram tree*, which represents the all 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 北海道大学大学院情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University

1. はじめに

1.1 背　景

ユーザのインターネットによる情報収集が日常的な作業となりつつある昨今，キーワードへのリンク付け[⋆1]やコンテンツへのタグ付け[⋆2]等，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提供する．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現在でも大多数であり，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．

本論文では，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，さらに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きわ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見られる．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，検索作業を効率化することができる．

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．快適な閲覧のためには，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．本論文では，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，映画の題名や特徴的ないい回し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，出現頻度等の統計量を用いることが考えられるが，計算資源の限られた PC で任意の単語 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，

⋆1 URL: http://www.hatena.ne.jp/
⋆2 URL: http://www.flickr.com/

速度の面でも問題が生じる．

1.2 目　　的

このような問題を解決するため，本論文では，与えられた文字列に対し，ある単語数以下のすべての単語 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．

本論文の後半では，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．

1.3 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[4]．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，一般的に TF-IDF 法は単語，特に名詞を対象としており，基本的には任意の単語 N グラムを扱うことは考慮されない．また，2 単語の連接頻度を考慮した手法[9] 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[7] がある．松尾らは，ユーザ個人の閲覧履歴中によく現れる単語（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．

文字列の索引構造としては，与えられた文字列のすべての部分文字列を表す接尾辞木[3),5),6)] 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),2)]，ある単語数以下のすべての部分文字列を表す単語幅限定接尾辞木[8)] 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．

1.4 本論文の構成

本論文の構成は以下のとおりである．2 章ではキーワード抽出に用いるデータ構造である単語 N グラム木を定義する．3 章ではキーワード抽出アルゴリズムを与える．4 章では抽出したキーワードを用いたウェブブラウザのインタフェースの実現方法を提案する．5 章では計算機実験を行った結果を示す．6 章では本論文の結論を述べ，今後の課題をあげる．

2. 単語 N グラム木

本章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 Σ とする．以降では Σ は定数サイズであると仮定する．Σ 上の文字列全体の集合を Σ^* 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**といい，ε で表す．Σ^+ 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．ただし簡単のため，混乱のない場合には省略して xy 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．たと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，$|S|_\#$ 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 の総数であり，$f(S, X)$ と書く．ただし，$X = \varepsilon$ ならば $f(S, X) = \infty$ 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**，**部分文字列**，**接尾辞**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 のすべ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．$root$ は $WST(T, N)$ の根である．V はノードの集合であり，$root \in V$ 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．ただしラベルを連結した文字列に # が 1 つも現れないとき，$\overline{v} = \varepsilon$ と定義する．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，葉はたかだか 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，$|\overline{u}|_{\#} < |x|_{\#} < |\overline{v}|_{\#}$ であるとき，**x は辺 (u, v) に属する**という．任意のノード $v \in V$ 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．

葉はたかだか 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たかだか $2m-1$ 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．したが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，$|\overline{u}|_{\#} < |\overline{v}|_{\#}$ 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．したがって 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，$|w|_{\#} > |u|_{\#}$ であり，$|v|_{\#}$ 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）

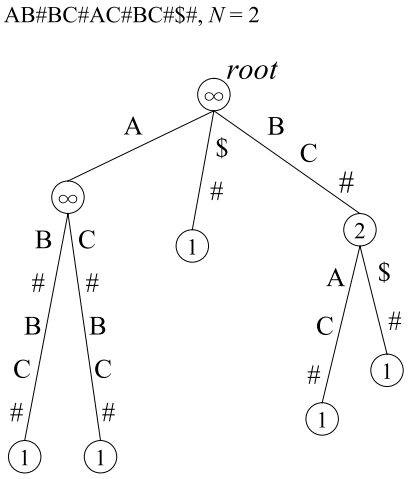

図 1　単語列 $T = AB\#BC\#AC\#BC\#\$\#$ に対する単語 2 グラム木
Fig. 1　The word 2-gram tree of a text $T = AB\#BC\#AC\#BC\#\$\#$.

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 のすべての接尾辞を表すデータ構造である単語接尾辞木[2] 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．本論文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．本論文では簡単のため，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，テキスト長に対する句読点の個数の割合がある閾値 δ_0（$0 < \delta_0 < 1$）以上である段落のみを残し，そのすべてを連結して 1 つの文字列とする．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 とする．

表 1　品詞スコアの例
Table 1　An example of the part-of-speech scores.

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
連帯し	0
フィラー	0
感動詞	0
記号	0
その他	0

3.2　キーワード候補のスコア付け

キーワード候補とは，単語数 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，助詞等を低く設定する．**表 1** に品詞スコアの例を示す．ここで,「名詞」,「名詞，接尾」のように複数あてはまる場合は，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

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

と定義する．ただし，たと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，たと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 文字あたりの長さをそれぞれ 4，2，2，1 とする．

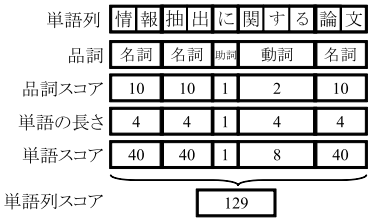

図 2　単語列「情報抽出に関する論文」に対する単語列スコアの計算例
Fig. 2　An example for computing phrase score.

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．**図 2** に，キーワードに対する品詞スコアの例および単語スコアの計算例を示す．ただしこの例では，簡潔さのため区切り記号を省略している．

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 $PF(X) \geq PF(X')$ であることがただちに分かる．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，$|X|_{\#} > |X'|_{\#}$ かつ $f(T, X) = f(T, X')$ となる X' が存在しない N 単語以下のすべての 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，本論文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 つの補題は本論文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 $|\overline{v}|_{\#} > |\overline{u}|_{\#}$ であるすべての 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 $v \in V'$ が存在する．

［証明］単語 N グラム木 $WST(T,N)$ は T のすべての N 単語以下の部分単語列を表すから，すべ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．したがって 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．キーワード候補 W について，W を表すノードが V' に存在しないと仮定する．すなわち，あるノード $v \notin V'$ について $\overline{v} = W$ である．$|W|_{\#} > 0$ であるので，v の先祖で *root* でないノード $u \in V'$ のうちで，u から v へのパス上のラベルに $\#$ が含まれないものがある．すなわち，u について $\overline{u} = \overline{v} = W$ が成り立つ．よって矛盾である．以上より，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 $|\overline{v}|_{\#} > |\overline{u}|_{\#}$ であるすべての 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，$|\overline{v}|_{\#} < |X|_{\#}$ 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，$|\overline{v}|_{\#} < |\overline{p}|_{\#}$ ならば $\mathit{freq}(v) > \mathit{freq}(p)$ であるから，矛盾である．よって，すべてのノード $v \in V'$ 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 $|\overline{v}|_{\#}$，$P(\overline{v})$ 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．すべ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，すべ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．また，スコアが上位 k 個のキーワード候補のみを保持すればよいので，ヒープを用いることでリストを実現する．

手続き **KeywordExtraction** $(T, N, WST(T,N), k)$;
入力：長さ n，単語数 m の入力テキスト T，最大単語数 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** $\mathit{WordCount}(v) > \mathit{WordCount}(u)$ **then**
5　　　$PF(\overline{v}) = \mathit{PhraseScore}(v) \cdot \log(\mathit{freq}(v))$;
6　　　$L = (L \cup \{\overline{v}\})$;
7　　　**if** $|L| > k$ **then** $L = L \setminus \{$ スコアが最少の要素 $l \in L\}$;
8　　**end if**
9　**end**
10　**return** L;

図 3　キーワード抽出の手続き
Fig. 3　The procedure for keyword extraction.

以上のキーワード抽出手続き **KeywordExtraction** を**図 3** 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．

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 3 の手続き **KeywordExtraction** 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．したがって，6 行目のリストへの追加を除くと，2 行目から 8 行目のすべてのノードに対するスコア計算は $O(m)$ 時間で行える．

スコアが上位の要素 k 個を保持するリストは，ヒープを用いることで，キーワード 1 つあたり $O(\log k)$ 時間で管理することができる．したがって，手続き全体でのリストの操作に関する計算量は $O(m \log k)$ 時間である．

以上より，図 3 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同じ単語が複数現れる場合があり，ユーザにとって冗長である．例として，**表 2** の左の列に，青空文庫[⋆1]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上位 20 個を示す．本文中には「弟子の僧」というキーワードが多く現れるため，その部分単語列や,「の」や「は」といった助詞を付加しただけの単語列がキーワードとして上位に出力されてしまう．しかし，単純に同じ単語を 2 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 $|\overline{L}_X|$ とおく．このとき，ある閾値 δ_1，δ_2（$0 < \delta_1$，$\delta_2 < 1$）に対し，

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

かつ

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

であるならば，X をキーワードとして出力し，X に含まれるすべての単語を 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．δ_1，δ_2 は，キーワード間の単語の重複を抑えたいほど 1 に近く設定する．すなわち，キーワードを表示できる個数が限られている場合や，順位が下位でも上位のキーワードに含まれない単語が含まれているキーワードを出力したい場合には，これらの値を大きくとればよい．

表 2 に，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 [⋆2]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，δ_1，δ_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いるが，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．

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
Table 2　The top-20 non-setected and selected keywords extracted from a novel “Hana”, written by Ryunosuke Akutagawa.

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

3.5　口語体文書からのキーワード抽出例

個人が公開するウェブ上の文書は，口調や文体，また表現がそれぞれ異なるため，名詞以外の単語の出現頻度が各ウェブサイトごとに異なり，キーワード抽出に影響を及ぼす可能性がある．そこで，実際のウェブ閲覧を想定した，個人によって書かれたブログからのキーワード抽出例を示す．

対象とするブログは，Google ブログ検索[⋆3]を用いて「Wii Fit」[⋆4]をクエリとして検索し，

⋆1 URL: http://www.aozora.gr.jp/

⋆2 URL: http://mecab.sourceforge.net/

⋆3 http://blogsearch.google.co.jp/

⋆4 http://www.nintendo.co.jp/wii/rfnj/index.html

表 3　ブログからのキーワード抽出例
Table 3　An example for extracted keywords from blogs.

ブログA	ブログB
バランスゲーム	筋トレ
踏み台リズム	バランス年齢
Wii Fit	体重
ヘディング	データ
有酸素運動	ー
リズムボクシング	ながらマラソン
バランススノボー	腹筋も
スキー	画面の
バランス年齢	ウィーボ
体重	休み

Wii Fit について継続的に記事を書いているブログを検索結果の上位から 2 件選択した．さらに，これらの本文を手作業で抽出し，形態素解析を行い，入力テキストを生成した．ただし，ブログの各記事は数十字から数百字程度と短いため，各ブログの古い記事から順に 50 件を入力テキストに追加した．以降ではこれらをそれぞれブログ A，ブログ B と呼ぶ．ブログ A の総文字数は 4821 字，ブログ B の総文字数は 1606 字となった．なお，短いテキストからの抽出に関しては次節で述べる．

表 3 に，ブログ A，ブログ B に対して提案手法によりキーワード抽出を行った結果の上位各 10 件を示す．いずれの抽出結果でも，Wii Fit およびエクササイズに関係したキーワードが抽出された．ブログ B の 5 番目のキーワードが「ー」となっているが，これは形態素解析器によって名詞と判断されたためである．

この結果を見ると，提案手法は，任意の単語 N グラムを候補集合としたうえで明らかな重複による冗長なキーワードを排除したキーワードのリストを出力できているといえる．しかし，「腹筋も」や「画面の」といった適切でない単語の切れ目で抽出してしまっている部分もある．テキストが十分長い場合には，たとえば「腹筋が」や「画面に」というように，いくつかの異なった使われ方により，適切な切れ目が判断可能であるが，ブログでは片寄った使われ方をする場合もおおいにあるため，重複の除去に加えて文法を考慮した選別も検討する必要がある．

一方で,「ながらマラソン」という非常に特徴的で興味深いキーワードも抽出されている．単語重みの割り当て方と結果の再選別法次第では，より個人のブログや掲示板の書き込み等に適したキーワード抽出が実現できる可能性がある．

3.6　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，1 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，本論文では，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，ユーザが新しいウェブページを閲覧する際に，過去に内容の関連したページを閲覧している場合が多いと考え，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．

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．索引のサイズを考慮し，本論文では 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 中のすべ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．

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

ここで，W がある葉 v によって表され，かつ T 中に 1 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，いったん 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．

3.7　短い文書からのキーワード抽出例

短い文書からのキーワード抽出に関する実験として，前節のブログ記事のほかに，新たに短いブログ記事を用いた例を示す．

この実験では，ある項目に関する文書を事前に閲覧した後，同じ項目について記述されているが短い文書を閲覧した場合を想定した．事前に閲覧した文書としては，前節のブログ A，ブログ B を用いる．新たに閲覧する文書は，Google ブログ検索による “Wii Fit” の検索結果から異なる 1 件を選択した．さらにこの本文を手作業で抽出し，形態素解析を行い，入力テキストを生成した．最終的なテキスト長は 533 文字である．以降ではこれをブログ C と呼ぶ．

表 4 に，ブログ C から単独で抽出したキーワード，ブログ A を閲覧履歴としてブログ C から抽出したキーワード，ブログ B を閲覧履歴としてブログ C から抽出したキーワードのそれぞれ上位 10 件を示す．

表 4　短い入力テキストに対するキーワード抽出の例
Table 4　An example for extracted keywords from a short text.

ブログC単独	ブログA を履歴に利用	ブログB を履歴に利用
ジョギング	ジョギング	ジョギング
ペース	Wii Fit	有酸素運動
気に入った	有酸素運動	筋トレ
Wii Fit	筋トレ	ペース
島	バランスボード	気に入った
する	感じ	Wii Fit
ないので	ペース	する
ました	気に入った	島
し	する	バランス
です	ヨガ	画面

この結果から，ブログ C 単独では抽出されなかった,「有酸素運動」や「筋トレ」,「バランスボード」といったキーワードが，閲覧履歴を利用することで抽出されたことが確認された．

しかし一方で,「感じ」や「する」といった一般的な単語のスコアも補正されてしまうことが確認された．このような悪影響を防ぐには，TF-IDF 法を単語に用いる等して，より特徴的な単語に高い重みを割り当てるか，十分長い入力テキストに対しては閲覧履歴を利用しないといった対策をとる必要がある．

4. キーワードを利用したインタフェース

前章のアルゴリズムで抽出したキーワードは，たと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．本章では，抽出したキーワードを利用してさらに二次的な処理を行うことで，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 4** に示す．表示しているのは「情報処理学会論文誌：データベース」に関するページ[⋆1]である．

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，少なすぎると一般的なウェブサイトしか抽出できず，また多すぎると偏ったウェブページしか該当しなくなってしまう．本研究で実装したウェブブラウザでは，5 単語以上のキーワード 1 つか，または各キーワードの単語数の和が 4 以下となるように，上位から順に選択する．

図 4　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
Fig. 4　A screenshot of our Web browser. A snippet is above the Web page, keywords in the top right, related Web sites in the below right, suggested words in the below left.

図 5，**図 6** に，Wikipedia の「Wii Fit」に関するページ[⋆2]および「ラーメン」に関するページ[⋆3]に対して，それぞれ抽出したキーワードを用いて Google[⋆4]により検索して得られた関連サイト上位 10 件のタイトルを示す．以後，これらをそれぞれページ A，ページ B と呼ぶ．ページ A では抽出されたキーワードより上位 2 つの「バランス Wii ボード」と「ゲーム」が選択された．ページ B では抽出されたキーワードより上位 3 つの「ラーメン」と「鶏ガラ」，および「スープ」が選択された．この結果，それぞれのページに関係しているが，現在閲覧しているウェブサイトから直接リンクされていないウェブサイトを抽出し提示することができている．このように関連サイトが適切に抽出されれば，ユーザの手間を省くことだけにとどまらず，新たな興味を喚起することが期待される．また，本手法では適切なキーワードが与えられれば 1 度のウェブ検索により関連サイトが抽出できるため，ユーザのウェブブラウザ側においては非常に少ない追加コストで実現できる．

⋆1 URL: http://ipsjtod.kyoto-su.ac.jp/abouttod.html
⋆2 http://ja.wikipedia.org/wiki/Wii_Fit/
⋆3 http://ja.wikipedia.org/wiki/ラーメン/
⋆4 http://www.google.com/

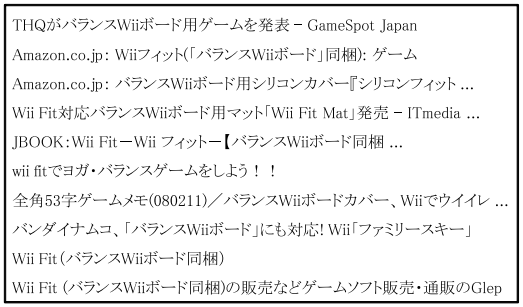

図 5　ページ A の関連サイト
Fig. 5　An example for related sites of the page A.

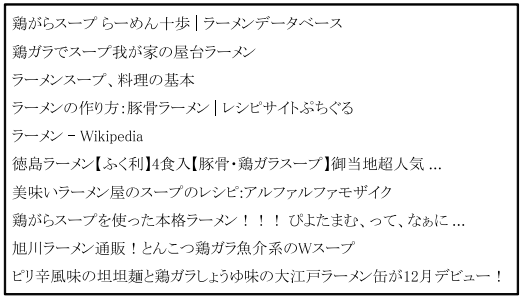

図 6　ページ B の関連サイト
Fig. 6　An example for related sites of the page B.

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．本論文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．

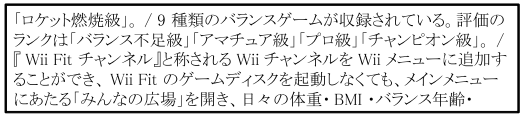

図 7　ページ A のスニペット
Fig. 7　An extracted snippet of the page A.

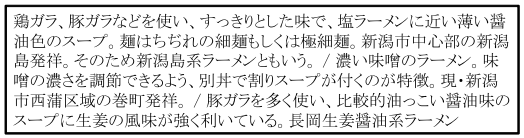

図 8　ページ B のスニペット
Fig. 8　An extracted snippet of the page B.

図 7，**図 8** に，ページ A およびページ B に対して提案手法により抽出したスニペットを示す．ページ A では，キーワードを多く含む部分として，「Wii Fit チャンネル」という項目について書かれた部分を中心に抽出された．ページ B では，各地方のラーメンの特徴について記述された部分が抽出された．これらは必ずしもページ全体の概要部分ではないが，いずれもキーワードに含まれる多くの単語を用いて詳細な記述がなされている．提案システムでは，提示したスニペットにユーザが興味を持ったならば，要約表示部をクリックすることにより，即座に該当部分を表示することが可能である．特に長いウェブページを閲覧する際には，文書要約機能はユーザの負担の軽減が期待できる．

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．

図 9，**図 10** に，提案システムにより当該ページを閲覧し，「Wii」と入力したときの検索語入力支援結果を示す．この結果では，閲覧しているウェブページの中から入力候補を抽出していることから，ユーザが入力した「Wii」に関する一般的な単語全体よりも，より「Wii Fit」に関係の深い単語が提示された．提案システムでは，閲覧しているページに存在する単語しか提示しないため，入力支援の結果を見ながらページ内の検索を行うことで，より無駄のない検索が実現できる．また，この機能はキーワード抽出に用いた単語 N グラム木を用いて，圧縮トライ構造による高速な計算を実現している．

Wii Fit
Wiiボード
Wiiメニュー
Wiiリモコン
Wiiチャンネル
Wiiバランス
Wiiヘルス
Wii伝言板
Wii本体
Wiiの

図 9　ページ A の閲覧中に「Wii」と入力したときの検索語候補
Fig. 9　suggested words followed by “Wii” in browsing the page A.

豚骨ラーメン
豚骨醤油
豚骨白湯
豚骨ベース
豚骨スープ
豚骨系
豚骨味
豚骨・
豚骨は
豚骨が

図 10　ページ B の閲覧中に「豚骨」と入力したときの検索語候補
Fig. 10　suggested words followed by “Tonkotsu” in browsing the page B.

5. 計算機実験

本章では，提案するキーワード抽出アルゴリズムを実装し，計算機実験を行った結果を示す．コンパイラは Visual C++ .NET 2005 付属のものを用いた．実験用のシステムとして，CPU が Pentium M 1.3 GHz，メモリが 1 GB，OS が Windows XP Professional のノート PC を用意した．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おいたため，計算速度についての実験のみを行った．

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 3 M 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，索引のノード数，索引の構築時間，キーワード抽出の時間を測定する．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．品詞スコア，文字種による補正は表 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 ∞ とし，入力のテキスト長を 100 k バイトから 3 M 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 11** に，計算時間の変化のグラフを**図 12** に示す．

図 11 では，最大単語数にともなってノード数が増えていく様子が分かる．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，

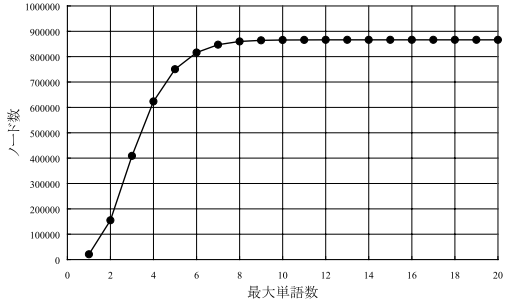

図 11　抽出キーワードの最大単語数に N 対する単語 N グラム木のノード数
Fig. 11　Number of nodes in the word N-gram tree for maximal word count N of extracting keywords.

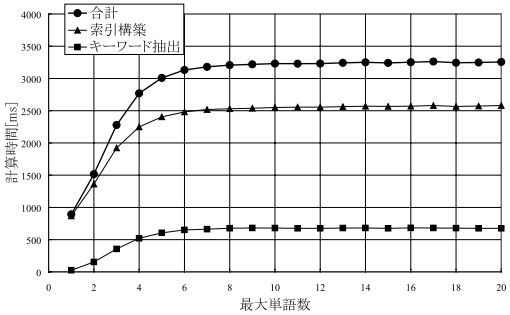

図 12　抽出キーワードの最大単語数に対する計算時間
Fig. 12　Computation time for maximal word count N.

このときの抽出結果は最大単語数を ∞ と等しくとったときと一致する．

図 12 では，最大単語数の増加にともな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分かる．したがって，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 ∞ 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

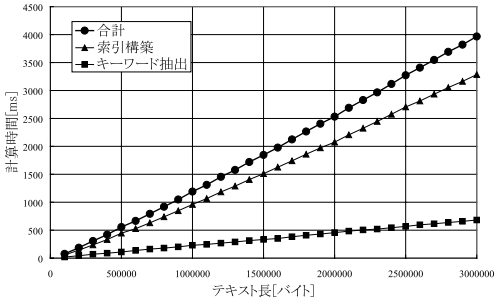

図 13　テキスト長に対する計算時間
Fig. 13　Computation time for the input text.

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，∞ 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 13** 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分かる．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 1 MB 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

本論文では，与えられた文字列に対し任意の N 以下のすべての単語 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，まず単語 N グラム木の構築時間の理論的な解析が必要である．また，キーワード抽出品質については，他手法と比較しての客観的な評価を行う必要がある．また，応用としてあ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べきである．

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，各キーワード候補に対し，

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，単語スコアそのものは任意に定めたものを用いることができる．したがって，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　2 人の査読者の方には非常に有益なコメントをいただきました．ここに感謝の意を表します．また，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ました．

参 考 文 献

1) Andersson, A., Larsson, N. and Swanson, K.: Suffix trees on words, *CPM: 7th Annual Symposium on Combinatorial Pattern Matching* (1996).
2) Inenaga, S. and Takeda, M.: On-Line Linear-Time Construction of Word Suffix Trees, *Proc. 17th Ann. Symp. on Combinatorial Pattern Matching*, pp.60–71 (2006).
3) McCreight, E.M.: A Space-Economical Suffix Tree Construction Algorithm, *J. ACM*, Vol.23, No.2, pp.262–272 (1976).
4) Salton, G., Wong, A. and Yang, C.S.: A vector space model for automatic indexing, *Comm. ACM*, Vol.18, No.11, pp.613–620 (1975).
5) Ukkonen, E.: On-Line Construction of Suffix Trees, *Algorithmica*, Vol.14, No.3, pp.249–260 (1995).
6) Weiner, P.: Linear Pattern Matching Algorithms, *FOCS*, pp.1–11 (1973).
7) 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，人工知能学会論文誌，Vol.18, No.4E, pp.203–211 (2003).
8) 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, pp.5–8 (2006).
9) 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).

(平成 19 年 12 月 20 日受付)
(平成 20 年 4 月 14 日採録)

(担当編集委員　有次 正義，DB Web 2007 推薦論文)

上村　卓史

2006 年北海道大学工学部情報工学科卒業．同年同大学大学院情報科学研究科入学，現在に至る．2006 年度下期 IPA 未踏ソフトウェア創造事業（未踏ユース）「テキストマイニング技術を融合したウェブブラウザの開発」開発代表者．現在，情報検索と，アルゴリズム設計の研究に従事．

喜田　拓也（正会員）

2001 年九州大学大学院システム情報科学研究科情報理学専攻博士後期課程修了．同年九州大学附属図書館研究開発室専任講師を経て，2004 年北海道大学大学院情報科学研究科助教授，現在に至る．2000〜2001 年日本学術振興会特別研究員．2001 年情報処理学会創立 40 周年記念論文賞を受賞．現在，文字列処理アルゴリズム，データ圧縮，電子図書館等の研究に従事．博士（情報科学）．

有村　博紀（正会員）

1990 年九州大学大学院総合理工学研究科修士課程修了．同年九州工業大学情報工学部助手，同助教授を経て，1996 年九州大学大学院システム情報科学研究科助教授，2004 年北海道大学大学院情報科学研究科教授，現在に至る．1996 年ヘルシンキ大学訪問研究員．2005 年リヨン第 1 大学訪問研究員．1999〜2002 年科学技術振興事業団「さきがけ研究 21」研究員．2005 年から文科省科学研究費補助金特別推進研究「半構造マイニング」研究代表者（2007 年まで）．現在，データマイニングと，計算学習理論，情報検索の研究に従事．1999 年人工知能学会 2000 年度優秀論文賞，PAKDD 2000 Paper with Merit Award，電子情報通信学会 DEWS2002，2003，2004 優秀論文賞，2005 年日本データベース学会上林記念研究奨励賞等を受賞．ACM，人工知能学会各会員．博士（理学）．