レレバンスフィードバックにおける検索語の共起関係推定処理の高速化

中島浩之[†,☆] 木谷強[††,☆☆] 岩城修[†,☆☆☆]

レレバンスフィードバックを実現する手法である Rocchio フィードバックは，文書検索の精度を向上させる有効な手法として知られている．筆者らは検索された文書に適当なスコアを与えるため，決定木学習アルゴリズム ID3 を用いて検索語間の共起関係を抽出し，検索結果の優先順位に反映させることで Rocchio フィードバックの検索精度を向上させる手法をこれまでに提案した．この際，検索者により必要ないし不要の判定をされていない文書（非サンプル文書）を仮想的な不要文書として ID3 に与えることで，より高い精度向上効果が得られることが分かっているが，扱う文書データベース中の文書数に比例して共起推定の処理時間が増加するという欠点があった．本稿では非サンプル文書の集合において，検索語が互いに独立かつ一様な確率で各文書に分布していると仮定することにより ID3 の決定木に登場する非サンプル文書の数を推定し，実際の非サンプル文書集合の代用とすることで，ID3 で処理する学習例数を減少させる手法を提案する．実験の結果，提案手法は従来手法とほぼ同等の検索精度向上を実現し，共起推定処理は 10 倍以上高速化できることが分かった．

Efficient Estimation of Co-occurences of Query Words in Relevance Feedback

Hiroyuki Nakajima,[†,☆] Tsuyoshi Kitani[††,☆☆] and Osamu Iwaki[†,☆☆☆]

Rocchio feedback is known to be effective in improving retrieval accuracy through relevance feedback. The authors proposed to use ID3 inductive learning algorithm for capturing co-occurences of query words in order to correct the order of ranked results of the Rocchio feedback, and experimental results showed that if we regarded non-sample documents as provisional irrelevant samples, estimated co-occurences effectively improved the acccuracy. Although the provisional samples are effective, they increase the processing time of estimating co-occurences, because the ID3 requires processing time proportional to the number of samples. In this paper, we assume that query words appear in the non-sample documents following independent and uniform probability distributions. Under this assumption, we estimate the number of non-sample documents appear in the ID3, and substitute the estimated number for the non-sample documents themselves to reduce the number of samples which are dealt by ID3. Experimental results show that the proposed method improves the processing time by 10 times, and the improvement of retrieval accuracy is not effected much.

1. はじめに

文書データベースから必要な文書を検索するためには，検索者の意図を正確に表現する検索式を作成する必要がある．しかし検索式の作成は，検索業務を専門とするサーチャですら試行錯誤を必要とする困難な作業であり[1)]，また検索者自身の検索意図は検索式として表現できるほどには明確でない場合が多い．レレバンスフィードバックはこの問題を解決する手法であり，検索者が必要文書と不要文書を選択することで，選択された文書からシステムが検索式を作成する．これはシステムと検索者が協調して検索式を作成するもので

† 株式会社 NTT データオープンシステムセンタ
Open System Center, NTT Data Corporation
†† 株式会社 NTT データ北米技術センタ
Technical Center of California, NTT Data Corporation
☆ 現在，日本電信電話株式会社
Presently with Nippon Telegraph and Telephone Corporation
☆☆ 現在，NTT DATA AgileNet L.L.C.
Presently with NTT DATA AgileNet L.L.C.
☆☆☆ 現在，株式会社 NTT データ開発本部
Presently with Research and Development Headquarters, NTT Data Corporation

あり，検索者にとって容易かつ高い精度で文書検索を行う有効な手段となる[2]．

レレバンスフィードバックを実現する代表的なアルゴリズムである Rocchio フィードバック[3]は，検索要求文および検索対象の文書をベクトルとして表現するベクトル空間モデル[4]（Vector Space Model）において，検索者によるフィードバック情報を用いて文書検索の精度を向上させる手法である．検索者は検索要求文による検索結果の一部について必要か不要かを判断し，システムにフィードバックする．システムはフィードバックされた文書中の単語を用いて再度検索を行う．次に検索要求文を表すベクトルを修正し，検索された各々の文書を表すベクトルとの内積をスコアとして，検索された文書をスコアの高い順に順位付けして呈示する．検索者は高い順位の文書から閲覧するため，システムが必要な文書に高い順位を与えれば，閲覧される文書が必要な文書である可能性が高くなる．つまり検索者が閲覧する文書の集合が高い検索精度を持つことになる．

Rocchio フィードバックは高精度の文書検索を実現する手段として知られており，多くの研究者からその有効性が報告されている[5)~7)]．しかしベクトルの修正において検索語間の関係は考慮されないため，複数の検索語が 1 つの文書中に現れる（共起する）ことで具体的な内容を指す文書に対しては，適切なスコア付けが行われないことがあった．

筆者らはフィードバックされた文書から決定木学習アルゴリズム ID3[8]を用いて検索語間の重要な共起を推定し，推定した共起を含む文書の順位を上昇させることで，Rocchio フィードバックの検索精度を向上させる手法を提案している[9]．ID3 での共起推定に用いる文書として，検索者によりフィードバックされる文書が少数であることを考慮し，検索者により必要ないし不要の判定をされていない文書（非サンプル文書）を仮想的な不要文書として用いる手法により，高い精度向上効果が得られることを実験により示した．しかし，この手法は文書データベース中のすべての文書を学習例として扱うため，ID3 による共起推定の処理量は膨大となり，大規模な文書データベースを対象とした場合に処理時間が長くなるという問題があった．

本稿では非サンプル文書の集合において，検索語が互いに独立かつ一様な確率で各文書に分布していると仮定する．この仮定を用いることで，ID3 の決定木の各条件を満たす非サンプル文書数を推定し，実際の非サンプル文書数の代用とする．これにより，ID3 で実際に非サンプル文書を処理する必要がなくなり，結果として共起推定処理を高速化することができることを示す．また NPL テストコレクションにより提案手法の評価実験を行い，従来手法と同等の検索精度を実現しながら，処理時間を大幅に短縮できることを示す．

2. 従来技術の概略

本章ではレレバンスフィードバックを実現する Rocchio フィードバックについて述べる．また ID3 による検索語の共起推定手法，および推定された共起関係により検索結果の優先順位を変更する手法について述べる．

なお本稿では以下の用語を用いる．

サンプル文書 検索者が必要または不要の判定をした文書を指す．検索者からシステムにフィードバックされる文書である．

非サンプル文書 検索対象である文書データベース中の文書のうち，サンプル文書以外の文書を指す．レレバンスフィードバックによる検索での検索対象の文書である．

2.1 Rocchio フィードバック

Rocchio フィードバック[3]はベクトル空間モデルと TF/IDF 法[4]を用いた文書検索システムにおいて，レレバンスフィードバックを実現する．

ベクトル空間モデル[4]は文書や検索要求文をベクトル空間上のベクトルとして表現する．このベクトル空間は単語ごとに独立した次元を持ち，文書や検索要求文のベクトルが各次元で持つ値は，単語が文書や検索要求文中で持つ “重み” を表す．

TF/IDF 法は，文書データベース中の多くの文書に登場する語は重要でなく，特定の文書において多く登場する語は重要とすることで単語の “重み” を決定するものである[4),10),11)]．本稿では，この原理に基づいて提案されている複数の重みの計算式の 1 つを用いる[11)]．文書 d_j 中に単語 t_i が出現する回数 $f_{i,j}$（Term Frequency，TF）および単語 t_i が出現する文書データベース中の文書数 n_i の逆数（Inverted Document Frequency，IDF）を用いて文書 d_j 中の単語 t_i の重み $w_{i,j}$ を

$$w_{i,j} = \frac{(\log(f_{i,j}) + 1.0) * \log(\frac{|DB|}{n_i})}{\sqrt{\sum_{k=1}^{N} (\log(f_{k,j}) + 1.0) * \log(\frac{|DB|}{n_k})^2}}$$

とする．なお $|DB|$ は文書データベース中の文書総数である．

検索要求文に対する文書のスコアは検索式のベクトルと文書のベクトルとの内積によって計算され，検索システムはスコアの高い順に文書を順位付けしてユー

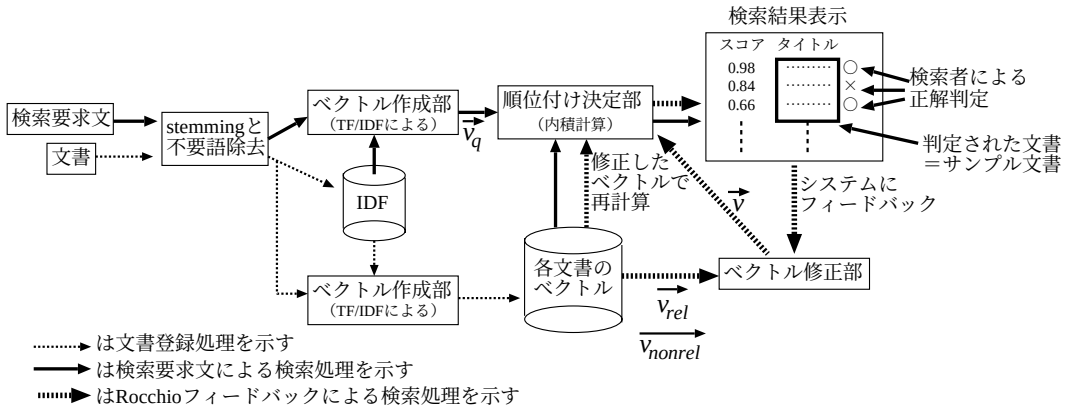

図 1 Rocchio フィードバック
Fig. 1 Rocchio feedback.

ザに呈示する．

Rocchio フィードバック[3]はサンプル文書のベクトルを用いて検索要求文のベクトルを修正することで，検索者の意図を検索式に反映する（**図 1**）．検索要求文のベクトルを $\vec{v}_q$，提示した文書中から検索者が選択した必要文書 num_{rel} 件の持つベクトルの和を $\vec{v}_{rel}$，検索者が選択しなかった文書（不要文書）num_{nonrel} 件の持つベクトルの和を $\vec{v}_{nonrel}$ としたとき，新たなベクトルは

$$v = \alpha\vec{v}_q + \frac{\beta\vec{v}_{rel}}{num_{rel}} - \frac{\gamma\vec{v}_{nonrel}}{num_{nonrel}}$$

となる（ここで α，β，γ は定数）．

Rocchio フィードバックで作成されるベクトル v はベクトル間の加減算によって作成されるため，検索語間の共起は反映されない．そのため検索語の重要な共起が意味を持つ文書に高いスコアが与えられない可能性がある．

2.2 決定木学習アルゴリズム ID3 による共起語推定

ID3 は相互情報量を尺度として用いて学習例を属性の有無で分割し，決定木を作成するアルゴリズムである[8]．学習例を各文書，属性を各文書に登場する検索語とすると，決定木は検索式を木構造で表現したものと考えることがでる．

ID3 のアルゴリズムを以下に示す．

(1) 入力された必要文書と不要文書の番号からなる集合を Set_0 とする．

(2) 集合 Set_0 に “未分割” の印をつける．

(3) “未分割” の印がついた集合のうち任意の集合 Set_i 中の必要文書，不要文書中の検索語 $t_j(1 \leq j \leq N)$ について，以下の式によって相互情報量 $I(t_j)$ を計算する（“未分割” の集合がなければ終了）．

$$I(t_j) = H - H(t_j)$$

ここで

$$\begin{aligned}
p_i &= Set_i\text{中の必要文書数} \\
n_i &= Set_i\text{中の不要文書数} \\
s_i &= p_i + n_i \\
p_i(t_j) &= Set_i\text{で } t_j\text{を含む必要文書数} \\
n_i(t_j) &= Set_i\text{で } t_j\text{を含む不要文書数} \\
s_i(t_j) &= p_i(t_j) + n_i(t_j) \\
p_i(\overline{t_j}) &= Set_i\text{で } t_j\text{を含まない必要文書数} \\
n_i(\overline{t_j}) &= Set_i\text{で } t_j\text{を含まない不要文書数} \\
s_i(\overline{t_j}) &= p_i(\overline{t_j}) + n_i(\overline{t_j}) \\
h(a,b,c) &= -\{\frac{a}{c}\log_2(\frac{a}{c}) + \frac{b}{c}\log_2(\frac{b}{c})\}
\end{aligned}$$

とし，H と $H(t_j)$ は

$$\begin{aligned}
H &= h(p_i, n_i, s_i) \\
H(t_j) &= \frac{s_i(t_j)}{s_i}h(p_i(t_j), n_i(t_j), s_i(t_j)) \\
&\quad + \frac{s_i(\overline{t_j})}{s_i}h(p_i(\overline{t_j}), n_i(\overline{t_j}), s_i(\overline{t_j}))
\end{aligned}$$

とする．

(4) 検索語 $t_j(1 \leq j \leq N)$ から $I(t_k)$ を最大にする t_k を選ぶ（複数ある場合は任意の 1 つ）．$I(t_k) > 0$ の場合，t_k を持つ文書からなる集合を $Set_{i'}$，持たない文書からなる集合を $Set_{i''}$ とし，それぞれに “未分割” の印をつける．i', i'' はすでに集合 $Set_{i'}$，$Set_{i''}$ が存在しなければ任意の数でよい．$I(t_k) \leq 0$ の場合は分割しない．

(5) 集合 Set_i から “未分割” の印を除き，3 へ戻る．

上記アルゴリズムで作成した決定木において，必要

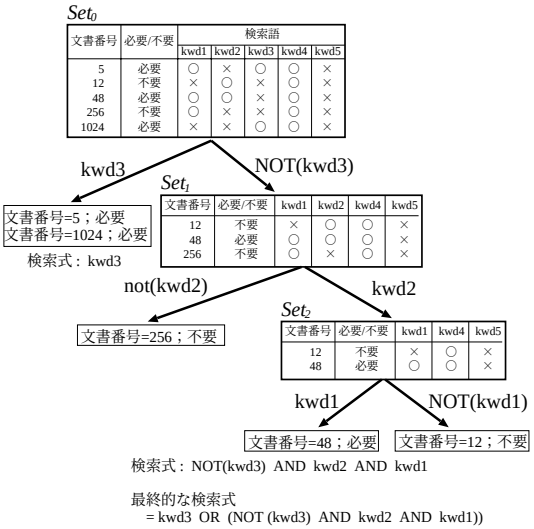

図 2 ID3 による検索式作成
Fig. 2 Producing a query using ID3.

文書を得るパスで用いた検索語を演算子 AND で結合して検索式を作成する．さらに各パスで得られた検索式を演算子 OR で結合したものを最終的な検索式とする（**図 2**）．

作成される検索式によって検索される文書は，演算子 AND により結合された各検索語が共起する文書になる．そのため ID3 によって得られる検索式は，必要文書に存在し，不要文書には存在しない検索語の共起を表していると考えることができる．

2.3 順位付けの補正

検索語の重要な共起を正しく推定できれば，推定された重要な共起を含む文書は含まない文書より重要と考えることができる．しかし，すべての文書について検索者による必要/不要の判定が行われているわけではないので，すべての重要な共起が推定されるとは限らない．そのため推定された共起を含む文書のみを検索結果とすると，一部の必要文書しか得られない可能性がある．Rocchio フィードバックは検索語間の共起をスコア計算に反映していないため，重要な共起を含む文書であっても与えられるスコアは必ずしも大きくならない．そのため重要な共起を含む文書が高い順位を持つとは限らないが，Rocchio フィードバックは検索精度を向上させる手段として有効性が確認されており，共起の扱いを除けば妥当な順位付けが行われていると考えることができる．

そこで，筆者らは推定された共起を Rocchio フィードバックによる順位付けの補正に利用することで，レレバンスフィードバックの精度向上を図る手法を提案

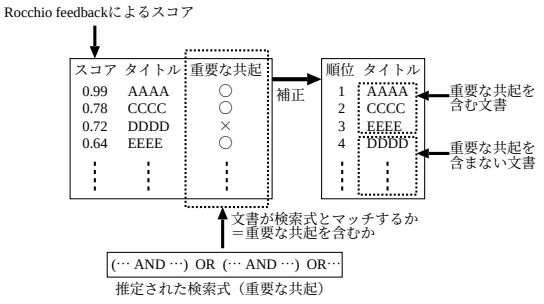

図 3 順位付けの補正
Fig. 3 Modification of ranked results.

した[9]（**図 3**）．

(1) Rocchio フィードバックにより各文書にスコアを与える．

(2) スコアを与えられた文書のうち，共起を含む文書を $d_1, d_2, \cdots, d_m$，共起を含まない文書を $d'_1, d'_2, \cdots, d'_n$ とする．

(3) $d_1, d_2, \cdots, d_m$ を Rocchio フィードバックによるスコアの高い順にソートして順位 $1, 2, \cdots, m$ 位を与える．

(4) $d'_1, d'_2, \cdots, d'_n$ を Rocchio フィードバックによるスコアの高い順にソートして順位 $m+1, m+2, \cdots, m+n$ 位を与える．

上記の手法では，重要な共起を含む文書に共起を含まない文書より高い順位が与えられ，なおかつ重要な共起を含む文書間，および共起を含まない文書間では Rocchio フィードバックによる順位付けが維持される．

2.4 学習例の追加

文書データベース中の全文書について必要文書または不要文書が判定されている場合，判定された文書を ID3 に学習例として与えることで，必要文書に登場し，不要文書に登場しない検索語の共起を得ることができる．しかしサンプル文書集合は文書データベースのごく一部分であり，ID3 により推定される共起は非サンプル文書集合中の不要文書にも存在する共起となる可能性がある．

そこで筆者らはすべての非サンプル文書を不要文書と仮定し，ID3 に与える学習例を増加させることで，サンプル文書数の不足を補う手法を提案した[9]．すべての非サンプル文書を不要文書として，必要文書と不要文書を判別する決定木を作成すると，サンプル文書集合中の必要文書のみを得る検索式が作成される．これでは検索精度の向上には役立たないため，決定木の深さがある程度深くなったところで文書集合の分割を停止させ，その段階で必要文書を得るパスを検索式作成に用いる．

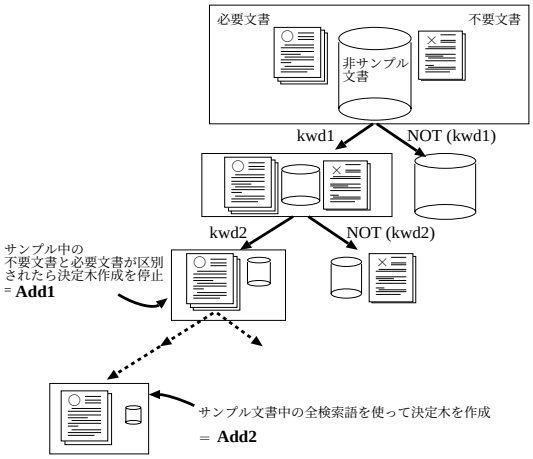

図 4 決定木作成の停止条件
Fig. 4 Stopping criterions in producing a decision tree.

分割の停止条件として，以下の 2 種類を用いる（**図 4**）．

(1) サンプル文書集合中の必要文書と不要文書が区別できたら停止する（以降 **Add1** と呼ぶ）．
(2) サンプル文書に登場するすべての検索語を用いて分割する．分割に用いていない検索語がなくなったら停止する（以降 **Add2** と呼ぶ）．

Add1，**Add2** の両手法とも，必要文書を含む集合を得るパスで用いた検索語を演算子 AND で結合して検索式を作成する．さらに各パスで得られた検索式を演算子 OR で結合したものを最終的な検索式とする．

3. 文書数推定による高速化

3.1 ID3 による共起推定処理時間

2.4 節で述べた **Add1**，**Add2** はサンプル文書に加えて非サンプル文書を不要文書として学習例に用いるため，ID3 には文書データベース中の全文書が与えられる．大量の学習例を用いる場合，計算機のメモリ実装量の制約から学習例をメモリ上に読み込むことが困難になり，処理速度は大きく低下すると予想される．また ID3 の計算量はおおよそ学習に用いる文書数に比例して増加する[12),13)]．NPL テストコレクション（約 1 万文書）を用いた実験（後述する本稿 4 章を参照）では **Add2** を用いる場合に 1 質問あたり平均 0.2 秒程度の処理時間が必要であるが，現在では文書数が 1000 万を超える文書データベースも運用されており，また文書データベースが扱う文書数はつねに増加する傾向にある．このため文書数に比例する処理時間を必要とする **Add1**，**Add2** を大規模な文書データベースに適用するのは現実的ではない．

3.2 非サンプル文書数の推定手法

Add1，**Add2** で ID3 に与えられる大部分の文書は非サンプル文書である．非サンプル文書を仮想的な不要文書として ID3 に与えると，学習例の集合を分割する検索語の決定にのみ用いられ集合分割の停止条件には使われない．このため，ID3 での学習例の集合中で各検索語を含む非サンプル文書の数を十分な精度で推定できれば，非サンプル文書を学習例として用いる必要がなくなり，処理時間の短縮が見込める．

本節では非サンプル文書の集合において，検索語が互いに独立かつ一様な確率で各文書に分布していると仮定する[14)]ことで，ID3 により分割された学習例の集合中で各検索語を持つ非サンプル文書数を推定する．

なお複合語を構成する各単語間や，係り受け関係にある単語間では相関関係が無視できないため[15)]，この仮定に基づいて非サンプル文書数を推定すると，現実の文書数から大きく乖離する可能性がある．特に検索者の入力する検索文から検索語を抽出して用いる場合，検索語の組合せによっては推定が不正確となる恐れがある．このように推定精度は必ずしも明らかではないが，本稿では NPL テストコレクションでの共起推定に利用することで，その影響を検証する．

以下に上記の仮定に基づく非サンプル文書数の推定手法，および **Add1**，**Add2** への適用を示す．

準備

Add1 および **Add2** では ID3 に与える学習例として検索者により必要ないし不要と判定された文書に加え，すべての非サンプル文書を用いる．これらすべての学習例すべてを含む集合を Set_0 とする．**Add1** および **Add2** の決定木作成において，学習例の集合を分割するために用いる検索語 t_j を選択するために用いられる相互情報量 $I(t_j) = H - H(t_j)$ は集合 Set_i について，

u_i = Set_i中の非サンプル文書数
$s_i = p_i + n_i + u_i$
$u_i(t_j)$ = Set_i中の t_jを含む非サンプル文書数
$s_i(t_j) = p_i(t_j) + n_i(t_j) + u_i(t_j)$
$u_i(\overline{t_j})$ = Set_i中の t_jを含まない非サンプル文書数
$s_i(\overline{t_j}) = p_i(\overline{t_j}) + n_i(\overline{t_j}) + u_i(\overline{t_j})$

とすると，H，$H(t_j)$ は

$$H = h(p_i, n_i + u_i, s_i)$$

$$H(t_j) = \frac{s_i(t_j)}{s_i} h(p_i(t_j), n_i(t_j) + u_i(t_j), s_i(t_j)) + \frac{s_i(\overline{t_j})}{s_i} h(p_i(\overline{t_j}), n_i(\overline{t_j}) + u_i(\overline{t_j}), s_i(\overline{t_j}))$$

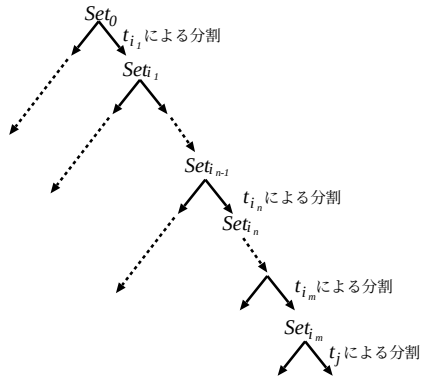

図 5 集合分割と検索語
Fig. 5 Dividing training set with query words.

となる．

集合 Set_0 から Set_i を得るまでの分割に用いられる検索語を $t_{i_1},\cdots,t_{i_m}$，分割で生成される集合を $\mathrm{Set}_{i_0},\cdots,\mathrm{Set}_{i_m}$（ただし $\mathrm{Set}_{i_0}=\mathrm{Set}_0$，$\mathrm{Set}_{i_m}=\mathrm{Set}_i$）とする（検索語と集合の関係を**図 5** に示す）．なお，$0<n\leq m$ である任意の整数 n について $\mathrm{Set}_{i_{n-1}}$ の分割に t_{i_n} が用いられて Set_{i_n} が得られるものとする．また t_j についての関数 $f(t_j)$ を

$$f(t_j)=(I(t_j)-H)s_0=-s_0H(t_j) \quad (1)$$

とする．H，s_0 は各集合において属性によらず一定であるから，$f(t_j)$ の大小比較を行うことで $I(t_j)$ の大小比較を行うことができる．

文書データベース全体の中で単語 t_j が登場する文書数を $df(t_j)$ とする．

Add1 および **Add2** は各集合において u_i，$u_i(t_j)$，$u_i(\overline{t_j})$ を用いるため，非サンプル文書の集合を決定木の分割にともなって分割し，各集合中で特定の検索語を含む非サンプル文書を数え上げる必要がある．

非サンプル文書数の推定

非サンプル文書集合中では各検索語が各文書に一様に分布しているとする仮定を用いて推定できる Set_i 中の非サンプル文書数 u_i，$u_i(t_j)$，$u_i(\overline{t_j})$ を，それぞれ真の値と区別するため $\hat{u}_i$，$\hat{u}_i(t_j)$，$\hat{u}_i(\overline{t_j})$ と記述する．

非サンプル文書集合中で各検索語が各文書に一様に分布するため，非サンプル文書集合を任意の検索語の有無によって分割して生成される文書集合でも，検索語が登場する文書数と文書集合中の全文書数の割合は分割前と等しい．すなわち任意の t_j（t_j は $t_{i_1},\cdots,t_{i_m}$ 以外の検索語）において，任意の分割対象の集合 Set_i で

$$\begin{aligned}\frac{\hat{u}_i(t_j)}{\hat{u}_i}&=\frac{\hat{u}_{i_{m-1}}(t_j)}{\hat{u}_{i_{m-1}}}\\&=\frac{\hat{u}_0(t_j)}{\hat{u}_0}\\&=\frac{\hat{u}_0(t_j)}{u_0}\\&=\frac{df(t_j)-p_0(t_j)-n_0(t_j)}{u_0}\end{aligned} \quad (2)$$

が成り立つ（ここで $\hat{u}_0=u_0=|DB|-p_0-n_0$）．

式 (2) より，集合 Set_{i_k} について

$$\hat{u}_{i_k}(t_{i_{k+1}})=\frac{(df(t_{i_{k+1}})-p_0(t_{i_{k+1}})-n_0(t_{i_{k+1}}))u_{i_k}}{u_0} \quad (3)$$

より $df'(t_{i_k})$ を

$$df'(t_{i_k})=\begin{cases}\frac{df(t_{i_k})-p_0(t_{i_k})-n_0(t_{i_k})}{u_0}\\(\mathrm{Set}_{i_k}\text{ が }t_{i_k}\text{ を含む場合})\\ \\\frac{s_0-df(t_{i_k})+p_0(t_{i_k})+n_0(t_{i_k})}{u_0}\\(\mathrm{Set}_{i_k}\text{が }t_{i_k}\text{を含まない場合})\end{cases} \quad (4)$$

として

$$\hat{u}_{i_{k+1}}=\hat{u}_{i_k}df'(t_{i_{k+1}}) \quad (5)$$

となる．よって

$$\hat{u}_i=u_0\prod_{l=1}^{i}df'(t_{i_l}) \quad (6)$$

$$\hat{u}_i(t_j)=(df(t_j)-p_0(t_j)-n_0(t_j))\frac{\hat{u}_i}{u_0} \quad (7)$$

$$\hat{u}_i(\overline{t_j})=(s_0-df(t_j)+p_0(t_j)+n_0(t_j))\frac{\hat{u}_i}{u_0} \quad (8)$$

を得る．

式 (6)，(7)，(8) は $df(t_i)$ と $p_0(t_i)$，$n_0(t_i)$，s_0 から評価できるが，これらは TF/IDF による重み計算，ないしサンプル文書のみから得られる．

得られた $\hat{u}_i$,$\hat{u}_i(t_i)$,$\hat{u}_i(\overline{t_i})$ を u_i, $u_i(t_i)$, $u_i(\overline{t_i})$ の代用として $f(t_j)$ を評価することで，非サンプル文書を扱うことなく集合分割に用いる検索語を決定できる．

この非サンプル文書推定を用いた決定木作成は以下の手順で行う．

(1) サンプル文書のみからなる集合 Set_0 を作成し，非サンプル文書数 $\hat{u}_0=u_0$ を求める．

(2) 集合 Set_0 に未分割の印を付ける．

(3) (a) 未分割の印が付いた集合 Set_i 中のすべての検索語 t_j について，$\hat{u}_i$ および式 (7)，(8) から得られる $\hat{u}_i(t_j)$，$\hat{u}_i(\overline{t_j})$ を u_i，$u_i(t_j)$，$u_i(\overline{t_j})$ の代用とすることで $f(t_j)$ を計算する．

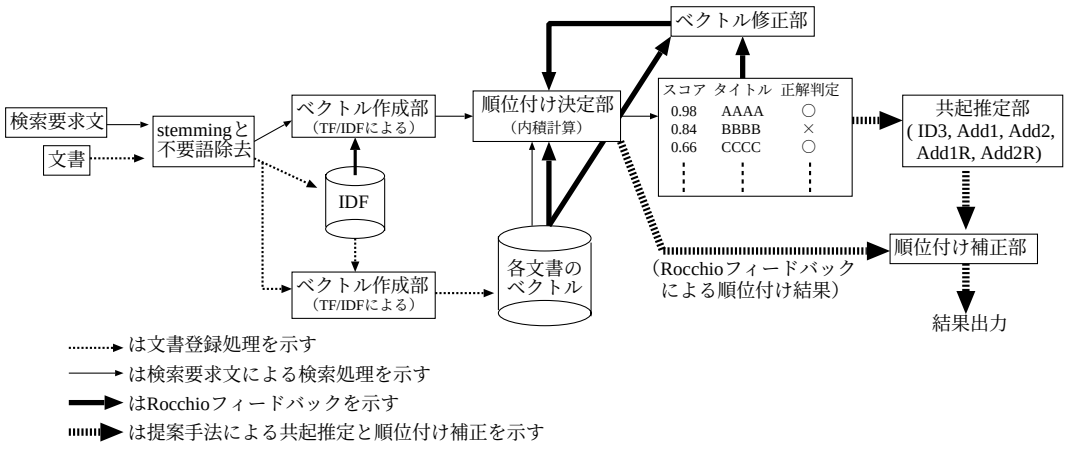

図 6 処理の流れ
Fig. 6 Processing flow.

表 1 NPL テストコレクション
Table 1 Statistics of the NPL test collection.

文書数	文書総量 (MB)	質問数	平均質問語数	平均正解数
11,429	3.1	93	6.7	22.4

(b) $f(t_j)$ を最大とする t_j の有無によって集合 Set_i を $Set_{i'}$（t_j 含む），$Set_{i''}$（t_j 含まない）に分割し，それぞれに “未分割” の印をつける．

(c) 集合 $Set_{i'}$，$Set_{i''}$ 中の非サンプル文書数 $\hat{u}_{i'} = \hat{u}_i(t_j)$，$\hat{u}_{i''} = \hat{u}_i(\overline{t_j})$ を $\hat{u}_i$ と式 (7)，(8) から求める．

(4) 決定木作成の停止を判定する．**Add1** と同じ条件で停止するものを **Add1R**，**Add2** と同じものを **Add2R** とする．停止しない場合 (3) に戻る．

4. 実　　験

本章では実験に用いたデータと実験環境，実験手順について述べる．

4.1 実験に用いたデータ

検索精度の評価には，英文を対象とした文書検索テストコレクションとして広く用いられている NPL テストコレクション[16),17)]を用いた（**表 1**，対象文書は物理分野の文献の要約）．テストコレクションは文書の集合と検索要求文からなり，検索要求文に対して関連する文書（正解）が与えられている．テストコレクションの各質問，文書からは FreeWAIS-sf[18)]の不要語辞書に登場する語を除去後，Porter の stemming アルゴリズム[19)]により語幹を取り出して利用した．

4.2 実 験 環 境

実験に用いた計算機環境を示す．

CPU Celeron300 A（動作周波数は 450 MHz）

メモリ SDRAM256 MB

OS linux2.0.35

HDD OS 等システム用 2 GB と本実験用 4 GB の 2 台（いずれも 5400 rpm，Ultra SCSI 接続）

4.3 実 験 手 順

実験データを用いた処理の流れを**図 6** に示す．以下に実験手順を示す．

(1) 検索要求文から TF/IDF 法を用いて $\vec{v}_q$ を作成し，各文書のベクトルとの内積を計算して各文書のスコアとする（通常の検索，以下 **Query** と呼ぶ）．

(2) スコア上位 n（$n = 10, 30, 50$）件をそれぞれサンプル文書とし，テストセットの正解を用いて正解（= 必要文書）と不正解（= 不要文書）を判定する．

(3) Rocchio フィードバックにより，各文書のスコアを再計算する（以下 **Rocchio** と呼ぶ）．α，β，γ は文献 11) より各々8，16，4 とした．

(4) **ID3**，**Add1**，**Add2**，**Add1R**，**Add2R** により検索要求文中の検索語間の重要な共起を推定する．なお決定木作成に用いる検索語は検索要求文中の検索語に限定した．

(5) 得られた共起を用い，2.3 節で述べた手法を用いて Rocchio フィードバックによる順位を補正する．

(6) 順位付けされた出力の各順位での適合率の平均を求める．評価には trec_eval[20)]を使用した．なお検索結果が検索文に対する正解記事であれば

表 2 適合率平均（%）
Table 2 Average precision (%).

手法	$n = 10$	$n = 30$	$n = 50$
Query	19.77	19.77	19.77
Rocchio	30.67	38.24	43.51
ID3	28.59	39.06	44.25
Add1	32.76	41.39	46.11
Add1R	32.63	41.18	47.06
Add2	34.00	42.28	47.84
Add2R	33.96	42.32	47.85

表 3 推定された共起を含む文書数
Table 3 Number of documents including estimated co-occurences.

手法	$n = 10$	$n = 30$	$n = 50$
ID3	10.04/2964.64	8.06/1534.36	6.56/1345.69
Add1	6.78/388.59	5.66/214.88	5.11/190.61
Add1R	6.49/381.53	5.66/208.88	5.02/189.80
Add2	6.00/102.79	5.39/129.92	4.86/182.40
Add2R	5.82/101.52	5.43/130.32	4.86/181.11

正解とする．

5. 実験結果と考察

5.1 非サンプル文書数推定の検索精度への影響

手法 **ID3**，**Add1**，**Add2**，**Add1R**，**Add2R** により共起を推定し，2.3 節で述べた手法を用いて Rocchio フィードバックの順位を補正した結果を**表 2** に示す．n はサンプル文書数である．

表 2 から共起を推定する各手法は $n = 10$ における **ID3** を除いて **Rocchio** に比べ優れた精度を示している．

非サンプル文書数を推定する **Add1R** および **Add2R** は実際に非サンプル文書を扱う **Add1** および **Add2** とほぼ同等の精度を示している．この結果から 3.2 節で提案した非サンプル文書数を推定する手法は，共起推定にさほど悪影響を与えていないことが分かる．

5.2 共起を含む文書数

各共起推定手法により推定された共起を含む文書数を**表 3** に示す（サンプル文書は含まない）．表中 A/B の A が正解文書数，B が共起を含む文書数を示す．それぞれ 1 質問あたりの平均である．

表 3 から非サンプル文書数を推定する手法 **Add1R**，**Add2R** で推定される共起を含む文書は **Add1**，**Add2** によるものと大きな違いはないことが分かる．

本稿の付録に，推定される非サンプル文書数が実際とどの程度乖離しているか評価した結果を示す．決定木が深くなるにつれて非サンプル文書数の推定精度は低下し，また実際より小さい文書数が推定される傾向にある．しかし表 3 のように **Add1R**，**Add2R** で推定される共起と **Add1**，**Add2** による共起で大きな違いがないことから，非サンプル文書を学習例に用いる手法では，決定木の深い部分において非サンプル文書は共起推定に強い影響を与えていないと考えられる．

表 4 決定木の最大深さ
Table 4 Max depth of decision tree.

手法	$n = 10$	$n = 30$	$n = 50$
ID3	2.19	3.62	4.15
Add1	3.04	4.01	4.42
Add1R	3.01	4.06	4.47
Add2	4.43	5.12	5.43
Add2R	4.42	5.10	5.42

表 5 パス平均長
Table 5 Average length of paths.

手法	$n = 10$	$n = 30$	$n = 50$
ID3	1.96	3.13	3.44
Add1	2.77	3.48	3.68
Add1R	2.77	3.55	3.77
Add2	4.12	4.59	4.72
Add2R	4.10	4.54	4.69

表 3 から **ID3** が非サンプル文書を学習例に用いる手法に比べて，多くの正解文書に含まれる共起を抽出していることが分かる．これは手法 **Add1**，**Add2**，**Add1R**，**Add2R** が非サンプル文書を不要文書として学習例に加えているために，非サンプル文書集合中に存在する共起の抽出に失敗していることを示している．

しかし **ID3** により推定される共起は多くの不要文書に含まれており，共起を含む文書数あたりの正解文書の数は，非サンプル文書を用いる手法と比較して低くなっている．本稿では推定された共起を含む文書の順位を上げることで Rocchio フィードバックの順位付けを修正しているが，推定される共起を含む文書が不要文書である割合が高いと，修正の効果が得られない，もしくは検索精度が低下することがありうる．**ID3** により推定される共起は多くの正解文書に含まれているものの，同時に多くの不要文書に含まれているため，**Add1**，**Add2**，**Add1R**，**Add2R** ほどの精度向上効果が得られない．

5.3 決定木の比較

各共起抽出手法で作成される決定木について，決定木の最大深さ，得られるパスの数（= 得られる重要な共起の数），平均パス長をそれぞれ**表 4**，**表 5**，**表 6** に示す．いずれも各質問ごとに求めた値を平均したものである（表中の n はサンプル文書数）．

ID3 に比べて非サンプル文書を学習例に用いる手法

表 6 平均パス数
Table 6 Average number of paths.

手法	$n = 10$	$n = 30$	$n = 50$
ID3	1.33	2.01	2.46
Add1	1.55	2.25	2.61
Add1R	1.45	2.19	2.63
Add2	2.27	3.90	4.94
Add2R	2.27	3.90	4.94

表 7 共起推定処理時間（CPU 時間（秒））
Table 7 Average CPU time (seconds) of estimation of co-occurences.

手法	$n = 10$	$n = 30$	$n = 50$
ID3	0.007	0.009	0.012
Add1	0.157	0.198	0.216
Add1R	0.010	0.014	0.017
Add2	0.168	0.214	0.236
Add2R	0.010	0.013	0.018

表 8 Rocchio フィードバック処理時間（CPU 時間（秒））
Table 8 Average CPU time (seconds) of Rocchio feedback.

検索語数	$n = 10$	$n = 30$	$n = 50$
10	0.168	0.189	0.202
50	0.384	0.413	0.424
100	0.425	0.553	0.608
1000	0.427	0.600	0.704

がより深く，パス数も長い決定木を得ている．**Add1** と **Add1R**，**Add2** と **Add2R** では大きな違いはない．

5.4 処理時間の比較

精度比較に用いた各手法について共起推定の処理時間を**表 7** に示す．表中の数値は 1 質問あたりの CPU 時間（秒）である（20 回実行し，その平均をとった）．

非サンプル文書数を推定する手法である **Add1R**，**Add2R** は元の **Add1**，**Add2** に比べて 10 倍以上高速化されている．精度比較，および処理時間比較から，**Add1R**，**Add2R** は精度にほとんど悪影響を与えることなく，処理速度を大幅に向上させているといえる．

次に Rocchio フィードバック（共起推定処理なし）の処理時間を，Rocchio フィードバックで用いる検索語を重みの大きい順に 10，50，100，1000 語と制限した場合の処理時間を**表 8** に示す．

共起推定の処理時間と Rocchio フィードバックの処理時間を比較すると，**Add1**，**Add2** は Rocchio フィードバックと同等以上の時間を必要とする場合があるが，3.2 節で提案した **Add1R**，**Add2R** の処理時間は比較的短い．そのため，Rocchio フィードバックの処理に加えて **Add1R** ないし **Add2R** による共起推定処理を行っても，レレバンスフィードバック全体の処理時間は大きく増加しない．

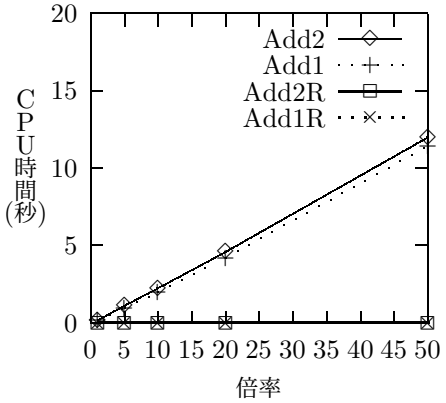

図 7 より大きな文書データベースでの CPU 時間（秒）平均（サンプル文書数 10）
Fig. 7 Average CPU time (seconds) on larger document database (10 samples).

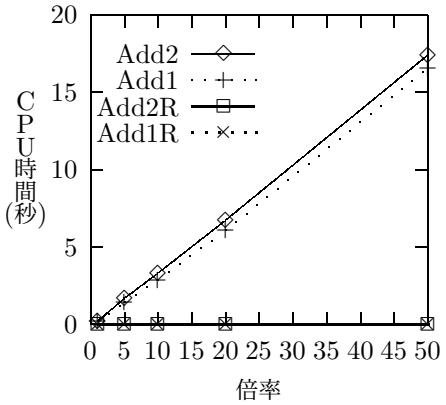

図 8 より大きな文書データベースでの CPU 時間（秒）平均（サンプル文書数 30）
Fig. 8 Average CPU time (seconds) on larger document database (30 samples).

最後に参考として文書データベース中の文書を増加させた場合の処理時間をサンプル文書数 10，30，50 についてそれぞれ**図 7**，**図 8**，**図 9** に示す．評価は NPL テストコレクション中の同一の文書をデータベース中に複数登録することで，非サンプル文書数を増加させて行った．表の “倍率” は文書数を何倍としたかを示す．

Add1，**Add2** は文書数の増加にほぼ比例して処理時間が増加していることが分かる．一方 **Add1R**，**Add2R** は文書数を増加させてもほとんど処理時間が変化しない（表 7 での処理時間とほとんど変わらないため，グラフではほとんど CPU 時間が 0 に見える）．これは **Add1**，**Add2** が非サンプル文書を直接扱うことで非サンプル文書数を求めるのに対して，**Add1R**，**Add2** は式 (6)，(7)，(8) を用い，各検索語がデータ

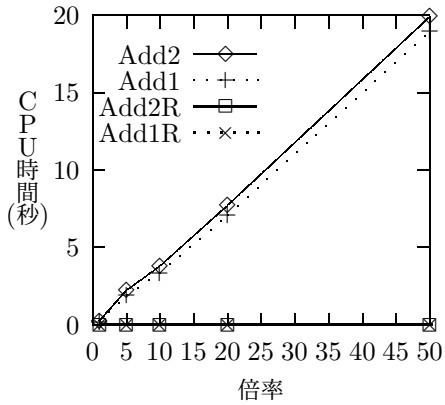

図 9 より大きな文書データベースでの CPU 時間（秒）平均（サンプル文書数 50）
Fig. 9 Average CPU time (seconds) on larger document database (50 samples).

ベース中に登場する頻度を用いて非サンプル文書数を推定するため，非サンプル文書を決定木作成の際に扱う必要がなく，非サンプル文書数が増加しても，処理時間にあまり影響しないためである．

6. おわりに

レレバンスフィードバックにおいて，検索語間の重要な共起関係を ID3 によって推定し，検索精度を向上する手法において，非サンプル文書を仮想的な不要文書として ID3 の学習例として用いる代わりに，各検索語の非サンプル文書集合での分布を仮定することで，ID3 で扱われる非サンプル文書数を推定し，ID3 で実際に取り扱う文書数を減少させる手法を提案した．提案手法により推定した検索語間の共起関係は，実際に非サンプル文書を ID3 で処理する場合と同等の精度向上効果をあげ，また提案手法を用いないで共起推定処理を行った場合に対し 10 倍以上高速化できることを実験により示した．

提案手法では ID3 で扱う学習例を減らすために非サンプル文書数を推定したが，非サンプル文書の集合から適当な数の文書をランダムサンプリングして仮想的な不要文書とする方法や，質問文から作成された検索式によって得られた検索結果の一部を仮想的な不要文書として用いることにより，ID3 に与える非サンプル文書数を減少させる手法も考えられる．今後の課題として ID3 に与える仮想的な不要文書の数と，その際に推定される検索語の共起による精度向上効果の関係を明らかにする必要がある．

一般に，Rocchio フィードバックは多くの検索語を用いる方が精度が向上するが，多くの検索語を用いるほど処理時間が増加する．そのため Rocchio フィードバックで用いられる検索語数は想定する用途により異なり，精度と処理速度のバランスをとって決定される☆．本稿で提案した共起推定による Rocchio フィードバックの精度向上手法はきわめて短い時間で処理が可能であるので，Rocchio フィードバックで用いる検索語数を減らすことで処理時間の短縮を図るとともに，推定した共起によって精度向上を図ることで，一定の精度を保ったまま処理時間を短縮できると考えられる．

推定した重要な共起を利用する手段として，本稿では共起を含むかどうかで文書の順位付けを修正したが，実際には共起ごとに重要度が異なると考えられる．多くの正解文書に含まれる共起や，重みの大きい検索語を含む共起について Rocchio フィードバックのスコアをより大きくすることで精度向上効果を得ており[22]，より適切なスコア付与手法を検討中である．

謝辞 本稿発表にあたってご支援いただきました（株）NTT データ開発本部技術開発部長中村太一博士，ならびに同開発本部ビジネスモデル開発部長山田伸一氏に深く感謝いたします．

参考文献

1) 三輪眞木子：データベースサーチャの視点，情報処理学会誌，Vol.33, No.10 (1992).
2) Spink, A.: Term Relevance Feedback and Query Expansion: Relation to Design, *SIGIR*, pp.81–90 (1994).
3) Rocchio, J.J.: Relevance feedback in information retrieval, *The SMART Retrieval System*, pp.313–323, Prentice-Hall (1971).
4) Salton, G. and McGill, M.J.: *Introduction to Modern Information Retrieval*, McGraw-Hill Advanced Computer Science Series, McGraw-Hill Publishing Company (1983).
5) Harman, D.: Overview of the Second Text REtrieval Conference (TREC2), *The Second Text REtrieval Conference* (*TREC-2*), pp.1–20, Department of Commerce, National Institute of Standards and Technology (1994).
6) Harman, D.: Overview of the Third Text REtrieval Conference (TREC3), *The Third Text REtrieval Conference* (*TREC-3*), pp.1–20, Department of Commerce, National Institute of Standards and Technology (1995).
7) 菅井 猛，和田光教：WWW 上の電子新聞に対する情報フィルタリングとその評価，情報学基礎研究会資料 FI-43-13, 情報処理学会 (1996).

☆ たとえば学習例なし（pseudo-feedback）で用い，通常の検索での精度向上に用いる場合には処理速度が重視されるため 50 語程度であり，十分に学習例が与えられ，精度向上を重視している場合には 100 から 1,000 語程度用いられる[21),22)]．

表 9 推定される非サンプル文書数と実際の文書数の比較（**Add1R**）
Table 9 Comparison between estimated number of non-sample documents and real number of the documents (**Add1R**).

深さ/倍率	～0.1	0.1～0.5	0.5～0.9	0.9～1.1	1.1～1.5	1.5～1.9	1.9～
1	0	0	0	176	0	0	0
2	11	30	18	176	26	5	6
3	30	65	44	69	37	10	10
4	60	79	28	17	18	8	15
5	69	48	9	2	4	5	9
6	46	12	7	0	2	0	3
7	22	6	2	1	0	0	0
8	8	1	0	0	0	0	1

8) Quinlan, J.R.: *C4.5: Programs for machine learning*, Morgan Kaufmann (1993).
9) 中島浩之，木谷 強，岡田 守：検索語間における共起関係の特定によるレレバンスフィードバックの高精度化，情報処理学会論文誌，Vol.40, No.3, pp.1236–1244 (1999).
10) 海野 敏：出現頻度情報に基づく単語重みづけの原理，*Library and Information Science*, pp.67–87 (1988).
11) Buckley, C., Salton, G. and Allan, J.: The Effect of Adding Relevance Information in a Relevance Feedback Environment, *SIGIR*, pp.292–300 (1994).
12) フセインアルモアリム，秋葉泰弘，金田重郎：木構造属性を許容する決定木学習，人工知能学会誌 (1997 年 5 月).
13) 金田重郎：Quinlan, J.R., C4.5: Programs for Machine Learning（書評），人工知能学会誌 (1995 年 5 月).
14) Gravano, L., García-Molina, H. and Tomasic, A.: The Efficacy of Gloss for the Text Database Discovery Problem, STAN-CS-TN-93-2, Stanford Univ. (1994).
15) 日本電子化辞書研究所：EDR 共起辞書. http://www.iijnet.or.jp/edr
16) NPL テストコレクション. ftp://ftp.cs.cornell.edu/pub/smart/npl/
17) Glasgow IDOMENEUS server. http://www.dcs.gla.ac.uk/idom/ir_resources/
18) Ulrich Pfeifer and Tung Huynh: FreeWAIS-sf (1994). ftp://ls6-www.infomatik.uni-dortmund.de/pub/wais/freeWAIS-sf-1.0.tgz
19) Porter,M.F.: An Algorithm For Suffix Stripping, *Journal of the Society for Information Science*, Vol.14, No.3, pp.130–137 (1980).
20) Salton, G. and Buckley, C.: trec_eval. ftp.cs.cornell.edu/pub/smart
21) Singhal, A.: AT&T at TREC-6, *TREC-6*, pp.215–226 (1998).
22) Nakajima, H., Takaki, T., Hirao, T. and Kitauchi, A.: NTTDATA at TREC-7: system approach for ad-hoc and filtering, *TREC-7* (1999).

付 録

本稿での提案手法により推定される非サンプル文書数が，実際の非サンプル文書数とどの程度乖離しているかを **Add1R** の決定木作成の各集合分割において評価したものを**表 9** に示す．表中の “倍率”（表横軸）は推定される非サンプル文書数が実際の文書数の何倍となっているかを，また “深さ”（表縦軸）は集合 Set_0 から何回目の集合分割であるかを示す．表中の数はある深さで推定される非サンプル文書数が実際の文書数の何倍となるか，その推定された回数を示す．分割にあたってはすべての検索語について式 (6)，(7)，(8) により非サンプル文書数の推定が行われるが，ここでは実際に行われた集合分割についてのみ回数を計測した．なおサンプル文書数は 30 である．

深さ 1（最初の分割）では正しく推定できるためすべての推定で倍率 1 となる．また検索語を含む集合と含まない集合のそれぞれで非サンプル文書数を推定するため，より浅い深さでの推定回数の和よりもより深い集合での推定回数の和の方が大きくなることがあり（たとえば深さ 1 では 176 回，深さ 2 では 272 回），また深さ 1 での推定回数が質問数（=93）より大きくなっている．

表 9 から深さが浅い場合には比較的良好に推定できているが，深くなるにつれて実際の文書数より推定される文書数が小さくなる傾向が分かる．このため決定木が深くなるにつれ，非サンプル文書数が決定木作成に与える影響は小さくなり，サンプル文書のみを学習例とする **ID3** に近くなる．

(平成 12 年 2 月 2 日受付)
(平成 13 年 9 月 12 日採録)

中島　浩之（正会員）

1970 年生．1994 年東京工業大学大学院理工学研究科情報工学専攻修士課程修了．同年 NTT データ通信（株）（現（株）NTT データ）入社．キーワード抽出，情報検索に関する研究開発に従事．現在，日本電信電話株式会社に出向中．人工知能学会会員．

木谷　　強（正会員）

1960 年生．1983 年慶應大学工学部電気工学科卒業．同年日本電信電話公社（現　NTT）入社．1988 年 NTT データ通信（株）（現（株）NTT データ）．形態素解析，情報抽出，情報検索に関する研究開発に従事．現在，米国 NTT DATA AgileNet L.L.C. に出向中．博士（工学）．平成 10 年度坂井記念特別賞受賞．

岩城　　修（正会員）

1955 年生．1980 年東京工業大学大学院総合理工学研究科修士課程修了．同年日本電信電話公社（現 NTT）入社．1988 年 NTT データ通信（株）（現（株）NTT データ）．ドキュメント処理，コンテンツ管理に関する研究開発に従事．現在，同社開発本部ビジネスモデル開発部部長．博士（工学）．電子情報通信学会，画像電子学会各会員．