情報処理学会論文誌　データベース　Vol. 3　No. 3　1–10 (Sep. 2010)

推薦論文

ビットシグネチャを用いたWebページの包含従属性発見の効率化

高橋公海[†1,*1]　森嶋厚行[†1,†2]　弓矢英梨佳[†1]
杉本重雄[†1,†2]　北川博之[†3]

近年，Webサイトを通じた情報発信が広く普及し，管理しなくてはならないWebコンテンツの量が増加している．我々は，Webコンテンツで成立する包含従属性の発見を支援するために，包含関係を効率良く計算する問題に取り組んでいる．我々は以前，厳密に包含関係を計算する前にあらかじめ低コストでのフィルタリングを行い，計算のためのコストを削減する手法を提案したが，その手法では大幅な削減に結び付かなかった．そこで本論文では，ビットシグネチャ法を用いたフィルタリング手法を検討する．提案手法により，以前の手法と比べてフィルタリングにおける削減率を大幅に向上可能なことを確認した．

Using Bit-signatures to Increase Efficiency in Finding Inclusion Dependencies in Web Pages

Masami Takahashi,[†1,*1] Atsuyuki Morishima,[†1,†2]
Erika Yumiya,[†1] Shigeo Sugimoto[†1,†2]
and Hiroyuki Kitagawa[†3]

Today, publishing information from Web sites is common, and the size of the Web contents that need to be managed is increasing. We have been tackling the problem of finding inclusion relationships among Web page elements, in order to help the administrators find inclusion dependencies in Web page contents. In our previous paper, we proposed a filtering method to apply a filter to reduce at a low cost the number of pairs of Web page elements to be examined, but it didn't show an expected performance. This paper proposes a more efficient filtering method, which uses bit signatures. The new method showed much improved performance compared to the previously proposed filter.

1. はじめに

近年，Webサイトを通じた情報発信が広く普及し，コンテンツの量も増加している．Webの特徴の1つは分散管理であるが，一方で，その特徴がコンテンツの一貫性維持を困難とする一因となっている．たとえば，ある2つのWebサイトが同じ内容を含むにもかかわらず，管理者が別であるために統一的に管理されていないということはよく見られることである．具体的には，大学の研究室のWebサイトでは，各構成員が自分のホームページ上で研究論文リストを公開することが多いが，これらの論文リストの間には矛盾が多く見られることがある．また，ある学科のWebサイトの入試説明会情報に変更があった際に，その学科に属する学部のWebサイトでも同様の変更を行わなければならないが，このような場合に確実な変更を保証することは多大な労力を要する．

一般に，コンテンツの一貫性を維持するためには，バックエンドにDBシステムを配置し，DBに格納されているデータからWebページを作成するアプローチがとられる．しかし，筑波大学のWebサイトを対象とした我々の予備調査[1)]では，バックエンドにDB等を持たずに手作業で管理されているWebサイトも数多く存在することが分かっている．

この問題に対し，我々は，DB等をバックエンドに持たないWebコンテンツの一貫性管理の支援を目的としたシステムの研究開発を行っている[1)–6)]．我々が提案している，明示的なコンテンツ一貫性制約を用いたWebサイト管理システムの概要を**図1**に示す．本システムは，Webコンテンツ間に成立すべき制約を与えることにより，DBをバックエンドにしたWebサイトに再構築せずとも，既存のWebコンテンツに対して**後付けで**コンテンツの一貫性管理が行えるようにするものである．

現システムでは，Webコンテンツ間の制約を，Webページの HTML要素やXML要素間（以降では，Webページの HTML要素やXML要素を総称して **Webページ要素**と呼

†1 筑波大学大学院図書館情報メディア研究科
Graduate School of Library, Information and Media Studies, University of Tsukuba
†2 筑波大学知的コミュニティ基盤研究センター
Research Center for Knowledge Communities, University of Tsukuba
†3 筑波大学大学院システム情報工学研究科
Graduate School of System and Information Engineering, University of Tsukuba
∗1 現在，日本電信電話（株）NTT 未来ねっと研究所
Presently with NTT Network Innovation Labs., Nippon Telegraph and Telephone Corp.

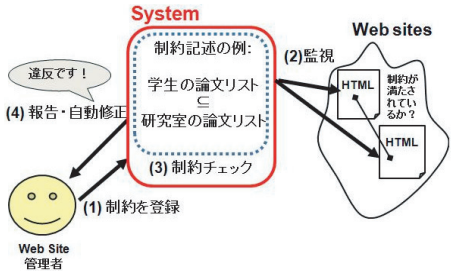


図 1　コンテンツ一貫性制約を用いた Web サイト管理
Fig. 1　Web-site management system using content integrity constraints.

ぶ）の**包含従属性**（inclusion dependency）[7] に限定している[4)–6)]．包含従属性とは，コンテンツの内容間につねに包含関係があることを保証する制約である．我々の文脈では，たとえば，「学生の論文リストを表すコンテンツ X がつねに研究室の論文リスト Y のサブセットでなければならない」といった制約のことである．そしてこの制約を $X \subseteq_{IND} Y$ と記述する．

Web コンテンツにおいて計算機による包含従属性の管理が必要がどうかの判断は最終的には人間がしなくてはならないが，そのようなカ所の候補を，計算機を用いて列挙することにより支援することは可能である．具体的には，包含従属性が成立するための必要条件として，特定の時点の Web コンテンツのインスタンスにおいて包含関係が存在することがあることから，包含従属性の発見を支援するためには，その根拠となる包含関係 $X \subseteq Y$ が Web コンテンツで成立していることを発見することが有効である．しかし，膨大な Web コンテンツを対象として手作業で 1 つずつ包含従属性を発見していくことは現実的ではない．また，伝統的に包含従属性が議論されてきたリレーショナルデータベースと異なり，Web コンテンツにおいては意図せぬ間違い等によって完全な包含関係が成立しているとは限らない．そこで，我々はこれまで，(1) **包含率** という概念を導入し，集合 X の要素のうち割合 c が集合 Y に含まれるとき，$X \subseteq_c Y$（X は Y に包含率 c で含まれる）と表記し，(2) c が与えられたとき，既存の Web コンテンツに存在する包含率 c 以上で成立する包含関係を計算機を用いてもれなく発見する手法を提案してきた．

特に論文 6) では，厳密な包含関係の計算を行う前にフィルタリングを行い，計算を行う Web ページ要素の組の候補数を減らす手法を提案した．その手法については 3 章で述べるが，Web ページ要素に対応する単語集合をあらかじめ辞書順にソートし，単語を辞書順に並べた際の範囲を表す値を各要素に対して用意し，それを用いてフィルタリングを行うものである．しかし，実データを用いた実験の結果，その手法では厳密な包含関係の計算を行う Web ページ要素の候補の数を十数%程度しか減らすことができず，大量の Web ページの処理に対応することは難しいことが分かった．

本論文で扱う問題と構成．そこで本論文では，フィルタリングを行うための別のアプローチとして，ビットシグネチャ法を用いる手法を提案する．ビットシグネチャは，テキスト検索等で索引手法として利用されてきたものであり，各テキストの特徴を表すビット列を作成し，偽陰性を持たない検索フィルタとして利用するものである．

詳細については 4 章で述べるが，本論文における基本的なアイデアは，各 Web ページ要素に含まれる単語集合を表すビットシグネチャをあらかじめ計算し，フィルタリングに利用することである．問題は，包含率 c が与えられたとき c 以上で包含関係が成立する Web ページ要素の組を効率良く求めるような，ビットシグネチャを用いたアルゴリズムを開発できるか，という点である．

本論文の構成は次のとおりである．2 章では関連研究について述べる．3 章で，論文 6) で提案した包含関係の発見アルゴリズムを説明する．4 章で，ビットシグネチャを用いたフィルタリング手法を説明する．5 章では提案手法の効果を調べるための実験と，その結果を示す．6 章はまとめと今後の課題である．

2. 関 連 研 究

情報統合の分野において，包含関係の発見に関する研究はすでに多く行われてきた．論文 9) では，リレーション（群）のインスタンスが与えられたとき，これらのリレーションの属性間に包含関係が成立するかどうかの判定を行う効率良いアルゴリズムを提案している．そのアルゴリズムでは，与えられたリレーション群の属性の集合 $ATTR = \{A_1, A_2, \ldots\}$ に対して，属性 A_i の値の集合と属性 A_j の値の集合に包含関係が存在する組 (A_i, A_j) のすべてを 1 度のディスクスキャンでもれなく算出する．また，論文 10) では，ハッシュ値を用いて 2 つのリレーションの属性間に包含関係が成立するかどうかを効率良く判定する Adaptive Pick-and-Sweep Join（APSJ），Adaptive Divide-and-Conquer Join（ADCJ）という 2 つのアルゴリズムを提案している．しかし，これらは本研究と異なり，厳密な包含関係だけを求めるアルゴリズムである．

論文 11) は，集合型オブジェクトの演算におけるクラス間の上位下位の関係の考慮について議論している．本論文での語の集合の包含関係の計算では，語の概念的な上下関係につ

いては考慮していないが，これらを考慮するとより広い意味での包含関係を発見できる可能性がある．これに関しては今後の課題である．

これまで，Web ページやテキストを対象とした類似検索に関する研究は行われてきた．論文 16) では，simhash と呼ばれる特殊なハッシュ値を計算することで類似する文書の検索を行う．また，テキスト類似検索の領域では，効率良く Similarity Join を行うための positional filtering[12] という手法が提案されている．しかし，包含関係と類似関係は同じものではない．すなわち，類似度は低いが包含関係にある文書の組や，類似度が高く包含関係にない文書の組が存在する．

数多くの領域において，判定の処理を効率化するために，あらかじめ低コストでフィルタリングを行うというアプローチがとられている．たとえば，上に述べた positional filtering[12] はそのようなフィルタリング手法の一種である．しかし，positional filtering は本研究とは異なり，100%の再現率（もれのないフィルタリング）を保証しない．

ビットシグネチャ法は，このようなフィルタリングの実現手法としてよく知られたアプローチであり，一般には再現率 100%となるように設計されることが多い．ビットシグネチャ法は元々テキスト検索の分野で用いられてきたが，近年ではそれ以外の分野で利用されることも多い．的野ら[13] は分散環境における RDF 問合せ処理の効率化のための索引手法に利用し，渡辺ら[14] はデータを暗号化した状態でデータベースに保存し，暗号化したまま問合せを施すプライバシ保護検索手法に利用している．また，石川ら[15] は OODB の文脈において，集合演算を効率化するために利用することを提案している．

これらに対する本研究の新規性は，包含率を考慮した包含関係の判定という処理を効率化するためのビットシグネチャの利用方法を提案することである．

3. Web コンテンツ間の包含従属性発見支援

3.1 概　　要

我々の提案している Web コンテンツ間の包含従属性の発見支援の概要を説明する．これは，すべての Web ページ要素の組合せのうち，各要素が持つ単語の（多重）集合間に包含関係がある組をすべて列挙するものである．

具体的には，まず，対象となる Web ページの集合 $P = \{p_1, p_2, \ldots\}$ を考える．p_i は実際の Web ページそのものでなく，XML データ等へのラッピング結果でもよい．ここで，各ページ $p_i \in P$ を Web ページ要素（HTML 要素や XML 要素）の木構造としてモデル化し，ページ p_i の Web ページ要素を $elem(p_i) = \{e_1, e_2, \ldots\}$ と表記する．各 e_i はそれぞれ単語の多重集合 $W(e_i)$（以下では断りのない限り $W(e_i)$ は多重集合であり，それに関する演算は多重集合の演算である）をその要素中に持つ．したがって，同一ページ p において木構造を構成する $elem(p_i)$ 中の Web ページ要素間では「e_i が e_j の下位要素であれば，$W(e_i) \subseteq W(e_j)$ である」という性質が満たされる．この性質を満たすような単語集合としては，各 Web ページ要素に含まれる文字列を n-gram 等で単純に分割したものや，形態素解析で分割したものが考えられる．我々の議論はその集合の作成方法とは独立しているため，この性質を満たしていればいずれも適用可能である．本論文では，形態素解析で分割した場合と，n-gram（$n = 2$，3）について検証を行っている．

我々は，ある Web ページ要素の組 e_i，e_j と値 $0 \leq c \leq 1$ に関して次が成立するとき，e_i **は** e_j **に包含率** c **で包含される**といい，$e_i \subseteq_c e_j$ と表記する[5]．

$$\frac{|W(e_i) \cap W(e_j)|}{|W(e_i)|} = c$$

これは通常の包含関係の一般化になっており，$c = 1$ のとき，$W(e_i) \subseteq W(e_j)$ と同じになる．これにより，間違い等が避けられない Web コンテンツにおいて包含関係の発見を扱うことができる．また，$e_i \subseteq_{\geq c} e_j$ を自然な拡張として定義する．すなわち，上式左辺の値が c より大きいとき，e_i **は** e_j **に包含率** c **以上で包含される**といい，$e_i \subseteq_{\geq c} e_j$ と表記する．他の不等号も同様である．

このとき，我々の問題は，対象となる Web ページに含まれるすべての Web ページ要素の集合 $E = \bigcup_{p_i \in P} elem(p_i)$ と与えられた $0 \leq c \leq 1$ に対し，すべての組の集合 $pairs = \{(e_i, e_j) | e_i, e_j \in E\}$ の中から，$e_i \subseteq_{\geq c} e_j$ が成立するすべての組 (e_i, e_j) を出力することである．

3.2 フィルタリングによる可能性のない組の除去

一般に，$pairs$ のサイズは膨大なものになるため，すべての可能な組 $(e_i, e_j) \in pairs$ に対して $e_i \subseteq_{\geq c} e_j$ を確認するのは大変な作業である．そこで，c が与えられたとき，低コストな手法であらかじめ可能性のない組を除去することができるフィルタ $filter(e_i, e_j, c)$ を考える．これは，$e_i \subseteq_{\geq c} e_j$ が成立する可能性がないときのみ偽を返す述語である．すなわち，

$$pairs' = \{(e_i, e_j) | (e_i, e_j) \in pairs \wedge filter(e_i, e_j, c)\}$$

としたとき，我々が対象とするフィルタ $filter(e_i, e_j, c)$ は単に $|pairs'| \leq |pairs|$ を満たす自明なフィルタではなく，(1) $|pairs'| < |pairs|$ かつ，(2) $pairs'$ 中の要素の組と包含率 c に対する結果が，元の $pairs$ を用いた結果と変わらない，という 2 つの条件を満たすものである．

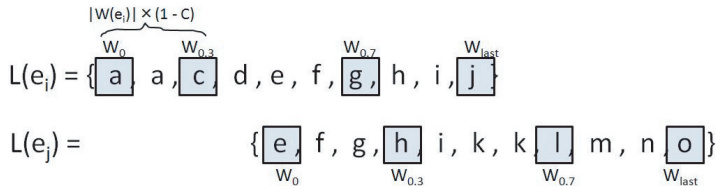


図 2　$L(e_i)$ と $L(e_j)$ の関係
Fig. 2　Relationship between $L(e_i)$ and $L(e_j)$.

3.3　従来の手法[6] とその課題

論文 6) では，$filter(e_i, e_j, c)$ を実現する一手法として，各 Web ページ要素 $e \in E$ に対して単語の 4 つ組 $d(e,c) = (w_0^e, w_{(1-c)}^e, w_c^e, w_{last}^e)$ をあらかじめ求めておき，$filter(e_i, e_j, c)$ の判定は $d(e_i, c)$ と $d(e_j, c)$ を用いて行う inclusion filter（以下，$i\text{-}filter$）を提案した．

$i\text{-}filter$ では，単語を辞書順に並べた際の範囲を利用し，e_i が e_j の範囲をどの程度含んでいるかを調べ，フィルタリングに利用する．包含率 c は単語の多重集合 $W(e_i)$ のうち，何%の単語が多重集合 $W(e_j)$ に含まれているかで算出されるため，e_i，e_j 中の単語を並べた際に範囲の共通部分が $c \cdot |e_i|$ 以上なければ，包含率が c 以上になる可能性もないと判断できる．

具体的には，まず，単語の多重集合 $W(e_i)$，$W(e_j)$ をあらかじめ辞書順でソートした単語列を $L(e_i)$，$L(e_j)$ とする．このとき，各 Web ページ要素の単語の 4 つ組はそれぞれ $d(e_i, c) = (w_0^{e_i}, w_{(1-c)}^{e_i}, w_c^{e_i}, w_{last}^{e_i})$ と $d(e_j, c) = (w_0^{e_j}, w_{(1-c)}^{e_j}, w_c^{e_j}, w_{last}^{e_j})$ となる．ここで，$w_0^{e_i}$ は単語列 $L(e_i)$ における先頭の単語，$w_{(1-c)}^{e_i}$ は $L(e_i)$ の先頭から $1-c$ の割合の位置にある単語，$w_c^{e_i}$ は $L(e_i)$ の先頭から c の割合の位置にある単語，$w_{last}^{e_i}$ は $L(e_i)$ の最後の位置にある単語（$d(e_j)$ も同様）である．

このとき，$i\text{-}filter$ の値は次のようになる．

$$i\text{-}filter(e_i, e_j, c) = \begin{cases} false & \text{if } (w_{1-c}^{e_i} < w_0^{e_j}) \text{ or } (w_{last}^{e_j} < w_c^{e_i}) \\ true & \text{otherwise} \end{cases}$$

たとえば，Web ページ要素組 (e_i, e_j) を対象とし，包含率 0.7 以上になる可能性があるかどうかを求める場合について考える．対象とする Web ページ要素 e_i に対応する単語列は $L(e_i) = [a, a, c, d, e, f, g, h, i, j]$，Web ページ要素 e_j に対応する単語列は $L(e_j) = [e, f, g, h, i, k, k, l, m, n, o]$ であり，辞書順でソートされているものとする（**図 2**）．

図 2 では，$w_{(1-c)}^{e_i} < w_0^{e_j}$ となっているため，$W(e_i)$ のサイズ（単語数）の 30%以上の範囲の単語が左にはみ出している．よって，これらの単語列の共通部分の範囲は必ず 70%より小さくなり，包含率が 0.7 以上になる可能性はないことが分かる．したがって，この場合には偽を返す．

$i\text{-}filter$ の問題点としては，単語列の範囲が共通であっても，単語自体がほとんど一致していない場合にはフィルタリングの精度が低くなってしまうことがあげられる．実データを用いた実験ではこのようなケースが多く，5 章で示すように，Web ページ要素の候補の数を十数%程度しか減らすことができなかった．

4. 提 案 手 法

本論文では，$filter(e_i, e_j, c)$ を実現するための別のアプローチとして，ビットシグネチャを用いる手法（以下，$b\text{-}filter$）を提案する．これは，各 Web ページ要素 $e \in E$ に対してビットシグネチャ $b(e)$ をあらかじめ求めておき，$filter(e_i, e_j, c)$ の判定を $b(e_i)$，$b(e_j)$，c を用いて行う手法である．本手法では，任意の包含率 c に対して $e_i \subseteq_{>c} e_j$ を判定するためのフィルタをどのようにして実現するかがポイントとなる．

4.1　ビットシグネチャの作成

本節では，ビットシグネチャ $b(e_i)$ の算出方法について説明する．ビットシグネチャのサイズを $sigsize$ とする．このとき，要素 $e_i \in E$ のビットシグネチャ $b(e_i)$ を次のように計算する．

- 各単語 $w \in W(e_i)$ ごとにハッシュ値 $0 \le h(w) \le sigsize - 1$ を計算する．
- 同じハッシュ値 $h(w)$ を持つ単語の出現回数が一定数 t 以上であれば，$b(e)$ の第 $h(w)$ ビットを 1 とする．それ以外のビットは 0 とする．ただし，t は次のように計算する．

(1)　$e_i \in E$ の中で最も $W(e_i)$ のサイズ（単語数）が小さい要素のサイズを $minsize$ とする．すなわち，

$$minsize = \min_{e_i \in E} \left| W(e_i) \right|.$$

(2)　t を次の式で決定する．

$$t = \begin{cases} 1 & (minsize \le sigsize) \\ \lfloor \frac{minsize}{sigsize} \rfloor & \text{otherwise} \end{cases}$$

t をこのように設定する理由は次のとおりである．t の値は，ビットシグネチャの値が 1 や 0 に偏らず，できるだけ散らばるように設定することが望ましい．上記で，$minsize$ をもとに t を設定している理由は，要素が木構造を構成しているため，$|W(e_i)|$ の分布が比較的小さい値に偏るためである．ここで，$|W(e_i)|$ の値が大きい要素をもとに t を設定してしまうと，$|W(e_i)|$ の値が小さな要素のビットシグネチャは 0 の割合が多くなるという問題がある．一方，$|W(e_i)|$ の値が小さな要素をもとに t を設定すると，$|W(e_i)|$ の値が大きな要素のビットシグネチャにおいて 1 の割合が多くなることが考えられるが，$|W(e_i)|$ の分布が比較的小さい値に偏るためさほど問題ではない．したがって，単語の分布が一様と仮定したときに，$minsize$ に近いサイズの要素のビットシグネチャにおいて，値の 1，0 のバランスがとれるように設定している．

例) 次の Web 要素群を考える．

$$E = \{e_1, e_2, e_3\}$$
$$W(e_1) = \{|a1, a2, b1, c1, d1|\}$$
$$W(e_2) = \{|a1, a2, b1, b2, c1, d1, d2, e1, e2, f1|\}$$
$$W(e_3) = \{|b1, b2, b3, b4, d1, d2, e1, f1, g1, h1|\}$$

また，この例では $sigsize = 8$ とし，$h(w) = (w$ の先頭の 1 文字の文字コード) %8，とする．このとき，

(1)　$W(e_i)$ に関するハッシュ値の多重集合 $H(e_i)$ は次のとおり：

$$H(e_1) = \{|0, 0, 1, 2, 3|\}$$
$$H(e_2) = \{|0, 0, 1, 1, 2, 3, 3, 4, 4, 5|\}$$
$$H(e_3) = \{|1, 1, 1, 1, 3, 3, 4, 5, 6, 7|\}$$

(2)　$minsize = |W(e_1)| = 5 \leq sigsize$ より，$t = 1$ である．したがって，各 e_i において $h(w)$ となる値が 1 つでもあれば $b(e_i)$ の第 $h(w)$ ビットを 1 とする．

(3)　よって，ビットシグネチャ$b(e_i)$ は次のとおり：

$$b(e_1) = 00001111$$
$$b(e_2) = 00111111$$
$$b(e_3) = 11111010$$

4.2　ビットシグネチャを用いた包含率付き包含関係の判定

本節では，$b(e_i)$，$b(e_j)$，包含率 c が与えられたとき，可能性のない組を除去するためのフィルタ $b\text{-}filter(e_i, e_j, c)$ を定義する．

基本的なアイデアは，与えられた $b(e_i)$ と $b(e_j)$ から $e_i \subseteq_c e_j$ について可能性のある最大の包含率 $\mathtt{c}_{\max}$ を求め，$\mathtt{c}_{\max} < c$ のとき，$b\text{-}filter(e_i, e_j, c) = false$ とする（候補からはずす）ことである．そのため，$b(e_i)$ と $b(e_j)$ に関する次の定理を利用する．

定理 1. e_i と e_j が与えられたとする．このとき，$e_i \subseteq_c e_j$ が成立する可能な最大の包含率 $\mathtt{c}_{\max}$ は次で計算される．

$$\mathtt{c}_{\max} = \frac{|W(e_i)| - X}{|W(e_i)|}$$

ただし，X は $b(e_i)$ で 1 が立っており $b(e_j)$ で 0 が立っているビットの数である．　□

証明. ビットシグネチャのある第 k ビットについて，$b(e_i)$ で 1 が立っており $b(e_j)$ で 0 が立っているものに着目する．このとき，$b(e_j)$ の第 k ビットに関しては，あと 1 つ $h(w) = k$ となるような単語があれば，1 が立っていた可能性がある．いいかえると，e_i に含まれており e_j に含まれない単語が少なくとも 1 つ必ず存在することが保証される．したがって，このようなすべての（X 個の）$b(e_j)$ のビットに対して単語が 1 つだけ足りなかった場合，すなわち，e_i の単語中 X 個の単語が e_j に含まれなかった場合が，$e_i \subseteq_c e_j$ の包含率が最大となる場合である．　□

定理 1 を利用し，偽陰性がないことが保証された $b\text{-}filter$ を次のように定義する．

$$b\text{-}filter(e_i, e_j, c) = \begin{cases} true & \mathtt{c}_{\max} \geq c \\ false & \text{otherwise} \end{cases}$$

例) 4.1 節の例において，$b\text{-}filter(e_1, e_2, 0.7)$ を考える．このとき，$\mathtt{c}_{\max} = \frac{5-0}{5} = 1$ となり，$1 > 0.7$ であるため包含率 0.7 以上の可能性がある．したがって真を返す．また，$b\text{-}filter(e_1, e_3, 0.7)$ については，$\mathtt{c}_{\max} = \frac{5-2}{5} = 0.6$ となり，$0.6 < 0.7$ であるため，偽を返す（フィルタリングされる）．

4.3　計算量の解析

包含関係発見の対象となる Web ページ要素の数を m としたとき，すべての組合せは ${}_mC_2$（すなわち $O(m^2)$）となる．また，要素に含まれる平均の単語数を n とすると包含関係発見のアルゴリズム[5] の計算量は $O(n)$ であるが，このアルゴリズムでは事前に単語のソートを行っていなければならない．したがって，何も工夫しない場合，すべての要素の単語のソー

トに $O(mn\log(n))$，包含関係の発見に $O(nm^2)$ の計算量がかかる．

i-filter を利用した場合，各 Web ページ要素内の単語のソートと 4 つ組 $d(e,c)$ の計算に $O(n\log(n))$ かかる．また，要素 1 組あたりのフィルタリングには $O(1)$ しかかからない．したがって，フィルタリングされた後に残った組に 1 度でも含まれる要素数を $m_{\mathtt{i}}$（$\ll m$）とおくと，まずすべての要素組のソートと 4 つ組の計算には $O(mn\log(n))$，次に，フィルタリングには $O(m^2)$，最後に，包含関係の検査に $O(nm_{\mathtt{i}}^2)$ かかる．したがって，何も工夫しない場合に比べて速い．

b-filter を利用した場合，各 Web ページ要素のビットシグネチャの計算量は $O(n)$ かかるが，包含関係の発見アルゴリムを適用する要素以外はソートをする必要がない．また，要素 1 組あたりのフィルタリングは $O(1)$ である．したがって，フィルタリングされた後に残った組に 1 度でも含まれる要素数を $m_{\mathtt{b}}$（$\ll m$）とおくと，まずすべての要素組のビットシグネチャの作成に $O(nm)$，次に，フィルタリングに $O(m^2)$ かかる．さらに，フィルタリング後の単語ソートに $O(m_{\mathtt{b}}n\log(n))$，最後に包含関係の検査に $O(nm_{\mathtt{b}}^2)$ かかる．次章の実験で示すように $m_{\mathtt{b}} \ll m_{\mathtt{i}}$ となるため，この中では最も高速なアルゴリムであることが分かる．

5. 実　　験

本章では，実データを用いた実験とその結果について述べる．実験 1 ではフィルタなし（all-pairs）に対して，*i-filter*，*b-filter* がどれだけ Web ページ要素の組合せを減らすことができるかを測定した．また，これらの実行時間の測定比較も行った．本実験においては *i-filter*，*b-filter* の実行時間はフィルタの作成とフィルタ適用後の処理（false positive のチェック）も含むすべての時間とした．実験 2 では，各 Web ページの要素に含まれる語の種類による影響を検証するため，語の作成に，(1) 形態素解析を利用した場合と，(2) n-gram（$n = 2$，3）で分割した場合，について比較した．実験 3 では *b-filter* の性質を検証するための実験を行った．

表 1　実験データ
Table 1　Experimental data.

Web サイト	全 Web ページ数	除去した Web ページ数	実験対象 Web ページ数
情報学群（inf）	61	12	49
情報科学類（coins）	175	4	171
情報メディア創成学類（mast）	39	0	39
知識情報・図書館学類（klis）	27	9	18
計	286	25	261

本実験で対象としたのは，コンテンツが分散管理されている実際の Web サイト群（**表 1**）である．具体的には，筑波大学情報学群[17]（以下 inf）と，情報学群に属する 3 つの学類（情報科学類[18]（以下 coins），情報メディア創成学類[19]（以下 mast），知識情報・図書館学類[20]（以下 klis））の Web サイトである．これら各 Web サイトのルートページからクローリングを行いアクセスできた同一 Web サイト内のページのうち，文字化けされていない HTML ページだけを対象として実験を行った．このページ収集処理で得られた Web ページ数を表 1 に示す．これらのページを XHTML に変換した結果の各 HTML 要素を，それらのページの Web ページ要素とした．

今回の実験では，情報学群と各学類間（inf-coins，inf-mast，inf-klis）の Web サイトの組，そして同じ Web サイト（inf-inf，klis-klis，mast-mast，coins-coins）の組をつくり，各サイト組のそれぞれのサイトから，Web ページ要素を取り出して組み合わせたものを対象とした．

各単語 w ごとのハッシュ値 $h(w)$ は，単語を表すビット列を 2 進の符号なし整数と見なし，シグネチャのサイズで割った余りとした．各 Web ページ要素に含まれる文字列を単語に分割する際には，形態素解析ツール Sen[21] を利用した．実験環境は，OS：CentOS 5，CPU：Intel Core2 DUO 2.13 GHz，メモリ：16GB であり，実験プログラムは Java 1.6.0 で実装した．

5.1 実　験　1

実験 1 では，フィルタなし（all-pairs）に対し，*i-filter*，*b-filter* がそれぞれどの程度組合せを減らすことができるかの検証，および実行時間の測定を行った．*i-filter*，*b-filter* についてはフィルタの作成とフィルタ適用後の処理の時間もすべて含めた合計を実行時間とした．なお，本実験で示す *i-filter*，*b-filter* の実行時間は，いずれも 5 回の計測結果の平均である．

実験のためのパラメータとしては，包含率を 0.7 とし，各 Web ページ要素に含まれる文字列は単語に分割した．*b-filter* の *sigsize* は，*i-filter* の $d(e_i, 0.7)$ に必要なメモリのサイズ（bit）よりも *b-filter* における $b(e_i)$ のサイズが小さくなるように（*b-filter* に有利にならないように）設定した場合（**表 2**）と，*sigsize* を $\{251, 509\}$ とした場合について行った．$d(e_i, 0.7)$ は各 Web ページ要素に含まれる文字列を形態素解析した際には単語の 4 つ組となり，サイズが個々に異なるため，平均をとっている．また，表 2 には *b-filter* のメモリサイズの平均を載せている．

表 2　実験 1 で用いた $d(e_i, 0.7)$ と $b(e_i)$ のサイズ（bit）
Table 2　Sizes of $d(e_i, 0.7)$ and $b(e_i)$ in Exp.1 (in bit).

i-filter							*b-filter*
inf-coins	inf-mast	inf-klis	inf-inf	coins-coins	mast-mast	klis-klis	
82.4	98.8	89.8	106.3	77.6	93.9	70.3	85.3

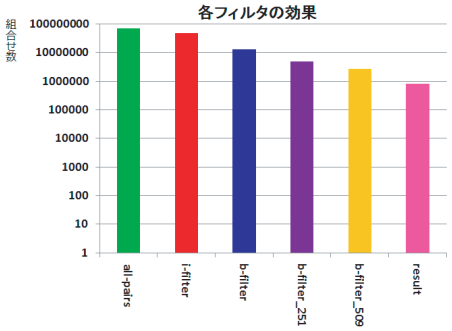


図 3　実験 1: 各フィルタの効果
Fig. 3　Exp.1: Effects of filters.

結果．まず，各フィルタの効果を**図 3** に示す．result はすべての組の包含率を厳密に計算した答えの分布を示している．*b-filter* に有利にならないようにシグネチャのサイズを設定した場合（すなわち，図中の “*b-filter*”）であっても *b-filter* の方が削減効果が高いことが分かった．また，*b-filter*_251 は 93%程度，*b-filter*_509 は 96%程度，組合せ数を削減することができた．

次に，実行時間の比較結果を**図 4** に示す．その結果，*b-filter* が最も高速であった．これは 4.3 節の計算量の解析結果とも矛盾しない結果である．

5.2　実　験　2

実験 2 では，各 Web ページの要素に含まれる語の種類による影響を検証するため，語の作成に，(1) 形態素解析を利用した場合（実験 1）と，(2) n-gram（$n = 2, 3$）で分割した場合，について比較を行った．実験のためのパラメータとしては，包含率を 0.7 とし，*sigsize* は $\{251, 509\}$ のいずれかとした．

結果．実験結果を**図 5**，**図 6** に示す．result はすべての組の包含率を厳密に計算した答えの分布を示している．語の作成に形態素解析器を用いて抽出した単語を用いた場合（図 3）と比較しても，n-gram（$n = 2, 3$）で分割した場合の傾向に特に大きな違いは見られなかった．

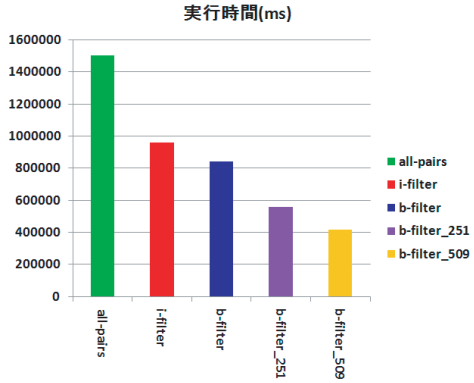


図 4　実験 1: 実行時間の比較
Fig. 4　Exp.1: Comparison in the elapsed time.

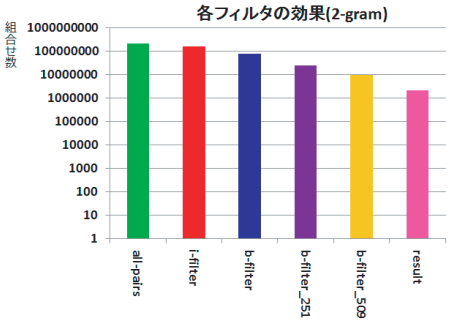


図 5　実験 2: 各フィルタの効果（2-gram の場合）
Fig. 5　Exp.2: Effects of filters (2-gram).

5.3　実　験　3

実験 3 では，包含率 c とビットシグネチャサイズを変更したときに，*b-filter*(e_i, e_j, c) がどのような振舞いをするか調査した．各 Web ページ要素に含まれる文字列は，形態素解析した場合について検証した．*sigsize* は $\{251, 509\}$ のいずれかであり，$h(w)$ は分割後の文字列 w のバイト列をサイズ $|w|$ の符号なし整数と見なして，*sigsize* で剰余をとったものを利用した．

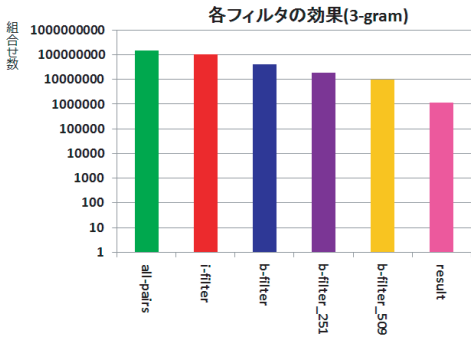


図 6　実験 2: 各フィルタの効果（3-gram の場合）
Fig. 6　Exp.2: Effects of filters (3-gram).

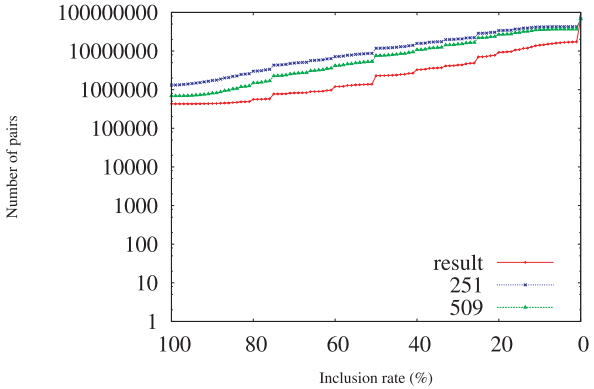


図 7　実験 3: *b-filter* におけるシグネチャサイズの影響
Fig. 7　Exp.3: Effects of the signature size in *b-filter*.

結果. 実験結果を**図 7** に示す．縦軸はフィルタリングで残った組合せの数，result はすべての組の包含率を厳密に計算した答えの分布を対数軸で示している．

結果から，*sigsize* が大きい方がフィルタリングの性能が高いことはもちろんだが，*b-filter*(e_i, e_j, c) の c の値が大きいほど，*sigsize* を大きくする効果が高く出ていることが分かった．実用的には，c の値がある程度以上であるときに意味があるため，望ましい性質であるといえる．たとえば，実験 1 の結果に対して実際に制約として利用できると考えられる Web ページ要素組の包含率を人手で確認したところ，c の値の下限は 0.6 であった．

6. まとめと今後の課題

本論文では，Web コンテンツに関する包含従属性の概念を導入し，既存の Web コンテンツにおける包含従属性の発見を支援するために，Web ページの要素間のコンテンツの包含関係を効率良く発見するための手法を提案した．具体的には，この問題を一定以上の包含率を持つ包含関係を求める問題として定義し，すべての Web ページ要素の組合せに対して厳密な包含関係を計算するのではなく，ビットシグネチャ法を用いてあらかじめ可能性のないものを候補リストから削除する手法を提案した．

今後の課題としては，まず，Web ページの階層関係やヒューリスティクスを利用した処理の効率化があげられる．また，本論文では単語の分布を一様と仮定してハッシュ関数の議論を行ったが，ここには工夫の余地があると考えられる．また，データと連動したシグネチャの更新についても議論が残されている．最後に，出力された Web ページ要素間の包含関係のランキング手法の開発があげられる．

謝辞　ゼミ等においてコメントいただきました，筑波大学大学院図書館情報メディア研究科阪口哲男准教授，永森光晴講師に感謝いたします．本研究の一部は科学研究費補助金特定領域研究（#21013004），科学研究費補助金基盤研究（B）（#19300081），科学研究費補助金若手研究（B）（#20700076）による．

参　考　文　献

1) 澤菜津美，森嶋厚行，飯田敏成，杉本重雄，北川博之：コンテンツ一貫性制約を用いた Web サイト管理手法の提案，*DEWS2007*, 7 pages (2007).
2) Sawa, N., Morishima, A., Sugimoto, S. and Kitagawa, H.: Wraplet: Wrapping Your Web Contents with a Lightweight Language, *Proc. IEEE SITIS 2007*, pp.387–394 (2007).
3) 澤菜津美，森嶋厚行，杉本重雄，北川博之：情報統合利用を目的とした HTML ページのラッピング支援，*DEWS2008*, 7 pages (2008).
4) 高橋公海，澤菜津美，森嶋厚行，杉本重雄，北川博之：Web コンテンツ一貫性管理支援ツールの開発，第 70 回情報処理学会全国大会講演論文集（第 5 分冊），pp.189–190 (2008).
5) 高橋公海，森嶋厚行，松本亜季子，杉本重雄，北川博之：Web コンテンツ一貫性管理のための制約発見支援，*iDB2008*, 福島，pp.127–132 (2008).
6) 高橋公海，森嶋厚行，杉本重雄，北川博之：Web ページを対象とした包含従属性の効

率的な発見手法，*DEIM2009*, 6 pages (2009).
7) Abiteboul, S., Hull, R., Vianu V.: *Foundations of Databases*, Addison-Wesley (1995).
8) Faloutsos, C. and Christodoulakis, S.: Signature files: An access method for documents and its analytical performance evaluation, *ACM Trans. Information Systems* (*TOIS*), Vol.2, No.4, pp.267–288 (1984).
9) Bauckmann, J., Leser, U. and Naumann F.: Efficiently Computing Inclusion Dependencies for Schema Discovery, *InterDB'06* (*ICDE Workshop*) (2006).
10) Melnik, S. and Garcia-Molina, H.: Adaptive Algorithms for Set Containment Joins, *ACM Trans. Database Systems*, Vol.28, No.1, pp.56–99 (2003).
11) Tanaka, K. and Yoshikawa, M.: Towards Abstracting Complex Database Objects: Generalization, Reduction and Unification of Set-type Objects, *ICDT1988*, pp.252–266 (1988).
12) Xiao, C., Wang, W., Lin, X. and Yu, J.X.: Efficient Similarity Joins for Near Duplicate Detection, *WWW2008*, pp.131–140 (2008).
13) 的野晃整，サイドミルザ パレビ，小島　功：ブルームフィルタを用いた分散 RDF 問合せ処理のための索引手法，*DEWS2008*, 8 pages (2008).
14) 渡辺知恵美，新井裕子，天笠俊之：ブルームフィルタを用いたプライバシ保護検索における攻撃モデルとデータ撹乱法の一検討，日本データベース学会論文誌，Vol.8, No.1, pp.113–118 (2009).
15) Ishikawa, Y., Kitagawa, H. and Ohbo, N.: Evaluation of Signature Files as Set Access Facilities in OODBs, *Proc. 1993 ACM SIGMOD International Conference on Management of Data* (*SIGMOD '93*), pp.247–256 (1993).
16) Manku, G.S., Jain, A. and Sarma, A.D.: Detecting near-duplicates for web crawling, *Proc. 16th International Conference on World Wide Web*, pp.141–150 (2007).
17) 筑波大学情報学群．http://inf.tsukuba.ac.jp/ (参照 2008-09-11)
18) 筑波大学情報学群情報科学類・情報学類．http://www.coins.tsukuba.ac.jp/ (参照 2008-09-11)
19) 筑波大学情報学群情報メディア創成学類．http://www.mast.tsukuba.ac.jp/ (参照 2008-09-11)
20) 筑波大学情報学群知識情報・図書館学類．http://www.klis.tsukuba.ac.jp/ (参照 2008-09-11)
21) 形態素解析システム Sen．http://www.mlab.im.dendai.ac.jp/~yamada/ir/MorphologicalAnalyzer/Sen.html (参照 2010-05-30)

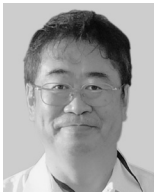
(平成 22 年 2 月 12 日受付)
(平成 22 年 5 月 8 日採録)

(担当編集委員　鬼塚 真)

高橋　公海（学生会員）
2008 年筑波大学図書館情報専門学群卒業．2010 年同大学大学院図書館情報メディア研究科博士前期課程修了．現在，日本電信電話（株）NTT 未来ねっと研究所勤務．WWW 応用に興味を持つ．

森嶋　厚行（正会員）
1993 年筑波大学第三学群情報学類卒業．1998 年同大学大学院工学研究科修了．博士（工学）．1998～2001 年日本学術振興会特別研究員．1999～2000 年 AT&T Labs-Research 客員研究員．2001 年芝浦工業大学工学部情報工学科講師．現在，筑波大学大学院図書館情報メディア研究科/知的コミュニティ基盤研究センター准教授．2009 年日本データベース学会上林奨励賞受賞．情報統合技術，XML/WWW データベース，メタデータデータベース等に興味を持つ．ACM，IEEE-CS，電子情報通信学会，日本データベース学会各正会員．

弓矢英梨佳（学生会員）
2010 年筑波大学図書館情報専門学群卒業．現在，同大学大学院図書館情報メディア研究科博士前期課程に在籍．WWW 応用に興味を持つ．日本データベース学会学生会員．



杉本　重雄（正会員）

1977 年京都大学工学部情報工学科卒業．1982 年同大学大学院工学研究科博士後期課程情報工学専攻単位取得退学．京都大学工学博士．1982 年京都大学工学部助手．1983 年図書館情報大学図書館情報学部助手，助教授，教授を経て 2002 年より筑波大学大学院図書館情報メディア研究科/知的コミュニティ基盤研究センター・教授，現在に至る．ディジタルライブラリ，メタデータに関心を持つ．Dublin Core Metadata Initiative の評議委員会委員．ACM，IEEE-CS，情報処理学会，電子情報通信学会，日本ソフトウェア科学会，日本データベース学会ほか正会員．

北川　博之（フェロー）

1978 年東京大学理学部物理学科卒業．1980 年同大学大学院理学系研究科修士課程修了．日本電気（株）勤務の後，1988 年筑波大学電子・情報工学系講師．同助教授を経て，現在，筑波大学大学院システム情報工学研究科教授，ならびに計算科学研究センター教授．理学博士（東京大学）．異種情報源統合，XML とデータベース，WWW の高度利用，データマイニング等の研究に従事．著書『データベースシステム』（昭晃堂），『The Unnormalized Relational Data Model』（共著，Springer-Verlag）等．日本データベース学会理事，電子情報通信学会フェロー，ACM，IEEE-CS，日本ソフトウェア科学会各正会員．