DEIM Forum 2009 B9-4

Webページを対象とした包含従属性の効率的な発見手法

高橋　公海† 　森嶋　厚行†,†† 　杉本　重雄†,†† 　北川　博之†††

† 筑波大学大学院 図書館情報メディア研究科　〒 305–8550 茨城県つくば市春日 1–2
†† 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　〒 305–8550 茨城県つくば市春日 1–2
††† 筑波大学大学院 システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †{mtaka,mori,sugimoto}@slis.tsukuba.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

あらまし　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．本論文では，Web コンテンツの管理を支援するために，データベース分野における重要なデータ一貫性制約の一つである包含従属性に着目し，Web ページ要素の包含関係を効率よく計算するための手法を提案する．本手法のアイデアは，厳密な包含関係の計算を行う前にフィルタリングを行うことで，計算を行う Web ページ要素の候補の数を減らすことである．

キーワード　Web サイト管理，コンテンツ一貫性，包含従属性

An Efficient Method to Find Inclusion Dependencies in Web Pages

Masami TAKAHASHI†, Atsuyuki MORISHIMA†,††, Shigeo SUGIMOTO†,††, and Hiroyuki KITAGAWA†††

† Grad. Sch. of Library, Information and Media Studies, Univ. of Tsukuba　1–2 Kasuga, Tsukuba, Ibaraki,Japan 305–8550 Japan
†† Grad. Sch. of Library, Information and Media Studies/Research Center for Knowledge Communities, Univ. of Tsukuba., Univ. of Tsukuba　1–2 Kasuga, Tsukuba, Ibaraki,Japan 305–8550 Japan
††† Grad. Sch. of Sys. and Info. Eng., Univ. of Tsukuba　1–1–1 Tennohdai, Tsukuba, Japan 305–8573
E-mail: †{mtaka,mori,sugimoto}@slis.tsukuba.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

Abstract　Today, publishing information fromWeb sites is common, and the size of the Web contents that need to be managed is increasing. To support the management of Web contents, we focus on inclusion dependencies, which are one of the most important data integrity constraints in the database area, and propose an efficient method to discover inclusion relationships existing in Web contents. The underlying idea is to filter out irrelevant combinations of Web-page elements before strict comparisons.

Key words　Web-site Management, Content Integrity, Inclusion Dependency

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 の特徴の一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．例えば，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等といった問題がある．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 等をも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．また，ある 2 つの Web サイトが同じ内容を含むにも関わらず，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もよく見られることである．例えば，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．

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きた [1] [2] [3] [4]．このシステムは，Web コンテンツ間に成立すべき制約 (例えば，研究室の構成員である学生の Web サイト上に掲載されている論文リストは，研究室の Web サイトに掲載されている論文リストのサブセットである，等) を与えることにより，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．本システムを利用することにより，既存の Web サイトを，DB をバックエンドにした Web サイトに再構築しなくとも，このような一貫性管理が可能になる．しかし，本システムを利用する際には，既存の Web コンテンツからそこに存在する制約を発見し，システムに与えなくてはならない．膨大な Web コンテンツを対象とした場合には手作業で一つずつ制約を発見していくことは現実的ではなく，既存の Web コンテンツに対する制約発見支援が必須である．

本論文では，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**と呼ぶ) の**包含従属性** (inclusion dependency) [5] に限定し，システムを用いてその発見を支援するための手法を提案する．包含従属性とは，コンテンツの内容間に常に包含関係があることを保証する制約である．例えば,「学生の論文リストを表すコンテンツ X が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」という制約を $X \subseteqq_{IND} Y$ と記述する．この制約の発見を支援するために，既存の Web コンテンツにおいて $X \subseteqq Y$ の包含関係が存在するか否かを決定する．包含関係の存在は包含従属性を示すための根拠となる．

本論文で扱う問題． Web ページ要素間の包含関係を発見する際のナイーブなアプローチは，全ての Web ページの組において，全ての Web ページ要素の組合せを対象に，包含関係の成立する Web ページ要素の組を算出することである．このとき，対象とする Web ページの集合を P とすると，その組合せの数は ${}_{|P|}C_2$ となり，各ページに含まれるページあたりの平均 Web ページ要素数を m とすると，1 つのページの組あたりの Web ページ要素の組合せ数は m^2 となる．よって，ナイーブなアプローチにおける組合せの数は ${}_{|P|}C_2 \times m^2$ となる．例えば，対象とする Web ページ集合に含まれるページ数を 100，各ページに含まれるページあたりの Web ページ要素数を 50 とすると，その組み合わせの数は ${}_{100}C_2 \times 50^2 = 12,375,000$ となる．したがって，大量の Web ページの処理は困難である．

そこで本論文では，大量の Web ページの処理に対応できるよう，Web ページ要素間の包含関係を効率よく発見する手法を提案する．詳細については 4 章で述べるが，基本的なアイデアは，包含関係の成立する Web ページ要素の組を実際に計算する前にフィルタリングを行うことによって，厳密に包含関係の有無を調べなければならない Web ページ要素の組の数を減らし，計算コストを削減することである．

関連研究． 論文 [6] では，効率よく Similarity Join を行うために，2 つの単語集合に対する類似度が与えられた閾値以上になるかどうかを検証する前に，検証する候補の組を減らすフィルタリング方法として，単語の位置を利用する positional filtering を提案している．これは，単語集合内の単語をソートした際の順序を利用したもので，実際に類似度の検証を行う前にその候補の数を減らすことで効率化を図っている．しかし，本研究は論文 [6] とは次の 2 つが異なる．

（1） 本研究の対象は包含関係であるが，論文 [6] では類似関係を対象としている．

（2） 論文 [6] の手法では再現率が低くなり，本研究のようにコンテンツ管理のための制約を発見することを目的とし，漏れなく制約を発見することが求められている場合には適さない．

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既に多く行われてきた．論文 [7] 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よ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インスタンスに対して，属性 X の値の集合と属性 Y の値の集合に包含関係が存在するとき，一度のディスクスキャンで $X \subseteqq Y$ を算出する (図 1(上))．

それに対し，我々は Web ページ要素を対象としているため，リレーションの属性を対象とした場合とは次の 2 点が異なる．

（1） Web ページのコンテンツには間違いも多く含まれるため，厳密な包含関係の判定ではうまくいかない．すなわち，一文字異なるだけでも包含関係の可能性を除去してしまうと，包含従属性の発見率が低下する．

（2） Web ページ要素はリレーション属性と異なり階層構造を成している (図 1(下))．したがって，3 章で詳述するように，出力される包含関係に重要なものとそうでないものが含まれる．例えば，$X \subseteqq Y$ が得られたとき，階層構造での Y の上位要素 Z(図 1(下)) に関して $X \subseteqq Z$ が得られることは自明であり，制約としての重要性は低いと考えられる．

そこで，論文 [8] では，(1) に対しては包含率という概念を導入し，どれだけの包含率でどの包含関係が存在するかを漏れなく計算すること．(2) に対しては，スコア付けルールによって，より重要と考えられる包含関係に上位スコアをつけるような手法を提案した．この手法を用いることで，Web ページ要素間の包含関係を発見することは可能である．

本論文の構成は次の通りである．2 章で，我々が開発中の Web コンテンツ一貫性制約を用いた Web サイト管理システムについて説明する．3 章で，これまで提案した包含関係の発見アルゴリズムについて説明する．4 章で，包含関係の厳密な計算を減らすための候補のフィルタリングを提案する．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. コンテンツ一貫性制約を用いた Web サイト管理手法

図 2 は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 [2] [3] [8] の概要である．これは，Web サイト間に成立すべき制約を与えることにより，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支

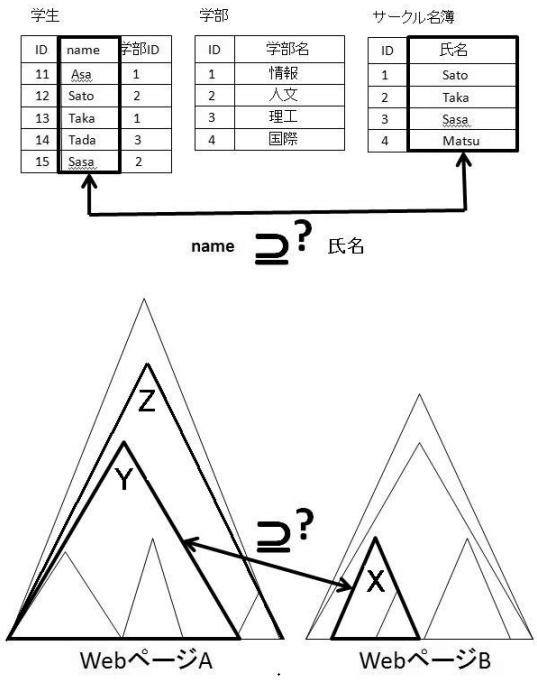

図 1　RDB における包含関係の発見 (上) と我々の扱う問題 (下)

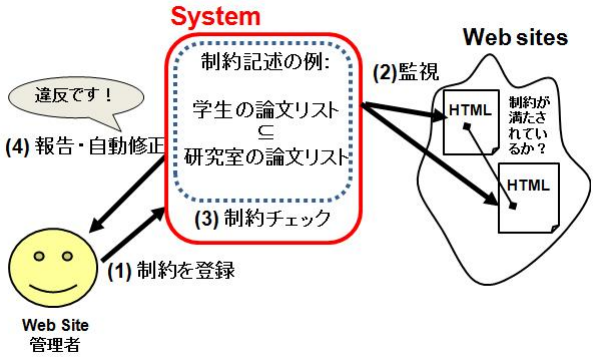

図 2　コンテンツ一貫性制約を用いた Web サイト管理

援を行うシステムである．

本システムの利用手順は次の通りである．

（1） 利用者がコンテンツ一貫性制約を登録する．

（2） システムは定期的もしくは Web サイトの更新が行われた際などに Web サイトのチェックを行う．

（3） 先に登録しておいた制約と照らし合わせ，制約が破られていないかどうかを調べる．

（4） 制約違反を発見した場合は，Web サイト管理者に報告もしくは自動修正といった対応を行う．

本システムを利用するためには，Web コンテンツに関する制約をあらかじめ与える必要がある．しかし，Web コンテンツを構成するページ数が増えると，手作業で一つずつ制約を発見していくことは現実的ではない．そこで，その制約発見支援のための仕組みが重要となる．次章以降では，制約の一つである包含従属性の発見をいかに効率よく支援するかについて議論する．

3. Web コンテンツ間の包含従属性発見支援

3.1 概　　要

本章では，論文 [8] で提案した，包含関係の「包含率」および「重要度スコア」の導入，および Web 要素間の包含関係を包含率とともに計算する BruteForce アルゴリズムについて説明する．

まず，対象となる各 Web ページを，Web ページ要素の階層構造としてモデル化する (実際の Web ページそのものでなく，ラッピング結果でもよい)．また，Web ページ内の Web ページ要素 x, y に含まれる単語集合 X,Y が存在し，それらには次の関係が成立すると仮定する．

Web ページ要素が表す集合が満たすべき性質． x が y の下位要素であれば，$X \subseteqq Y$ である．

この性質を満たすような集合としては，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．提案手法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．

3.2 包含率と重要度スコアの導入

論文 [8] では**包含率**と**重要度スコア**を導入することを提案している．具体的には，与えられた X と Y に対して，$x \subseteqq y$ がどれだけの包含率で成立するかを判定した際には，$[x \subseteqq y$, 包含率 c $(0 \leqq c \leqq 1)$, 重要度スコア s $(0 \leqq s \leqq 1)]$ を計算する．

まず，包含率について説明する．x に対する y の包含率とは，Web ページに含まれる Web ページ要素 x と y に対応する単語集合 X と Y において，X に含まれるどれだけの割合の単語が Y に含まれているかを示すものである．例えば，X が Y に完全に含まれていれば包含率は 1，全く含まれていなければ包含率は 0 なる．一般には次式が成立するとき「y が x を包含率 c で包含する」と言う．

$$|X| \leqq |Y| \wedge \frac{|X \cap Y|}{|X|} \geqq c$$

次に，スコアリングを行うための重要度スコアを説明する．スコアリングを行う動機は次の通りである．発見された包含関係には，自明な包含関係とそうでない包含関係が混在し，大量の包含関係が出力されると，価値の高い包含関係が埋もれてしまう場合がある．これは，1 章の関連研究で述べたように，主に Web ページに含まれる Web ページ要素間に階層関係が存在することに起因する (図 1(下))．そこで，重要な包含関係とそうでないものを区別するため，我々は出力される包含関係に対して重要度スコア $(0 \leqq s \leqq 1)$ を割り当てることを提案した．重要度スコアが大きな値であるほど重要な包含関係であることを表す．論文 [8] ではこの重要度スコアを計算するためのルールをいくつか定義したが，本稿では省略する．

3.3 包含率を計算する Brute Force アルゴリズム

ここでは，論文 [8] で提案した，1 組の要素 x, y とそれらに対応する単語集合 X と Y が与えられたとき，$X \subseteqq Y$ がどれだけの包含率で成立するかを判定する BruteForce アルゴリズムを説明する．

アルゴリズムを図 3 に示す．本アルゴリズムは入力として単語集合の組 (X,Y) をとり，包含率 c ($0 \leqq c \leqq 1$) を出力する．1 行目で，X 中の値のうち Y に含まれないものがあるたびに 1 ずつ減算するカウンタ $counter$ の初期値を計算する．$counter = 0$ になるということは，X のいずれの値も Y に含まれていなかったということである．それ以降では，X に含まれる各値が Y に含まれているかどうかを順に判定していき，含まれていない値を見つけるたびに $counter$ を 1 つ減じる．そして，$counter$ が 0 になれば，包含率 $c = 0$ で $x \subseteqq y$ が成立するため，0 を出力する (11 行目，18 行目)．

X が最後の値までテストされていないにも関わらず，Y の次の値が無くなってしまった場合 (5 行目，15 行目，20 行目) には，包含率 $c = ((counter - a)/X.size)$ を返す．ここで，a は X の残りの要素数である．

BruteForce アルゴリズムでは，一度の実行で入力の X と Y に対してそれぞれ一度ずつのディスクスキャンが必要となる．

```
Input: refValues, depValues
Output: depValues⊆refValues, 包含率 for depValues⊆refValues
 1  counter = depValues.size; // 含まれない値があるたびにカウントダウン
 2  while depValues has next value do
 3      currentDep := depValues.next();
 4      if refValues is empty then
 5          return (counter-a)/depValues.size;
 6      while true do
 7          currentRef := refValues.next() ;
 8          if currentDep = currentRef then break;
 9          else if currentDep < currentRef then
10              counter := counter - 1;
11              if counter == 0 then return 0;
12              if depValues has next then
13                  currentDep := depValues.next();
14                  currentRef := refValues.prev();
15              else return (counter-a)/depValues.size;
16          else if refValues has no next value then
17              counter := counter - 1;
18              if counter == 0 then return 0;
19              if depValues has next then
20                  return (counter-a)/depValues.size;
```

図 3　Brute Force アルゴリズム

4. 提 案 手 法

対象とする Web ページの集合を P とすると，その組合せの数は ${}_{|P|}C_2$ となり，各ページに含まれるページあたりの平均 Web ページ要素数を m とすると，1 つのページの組あたりの Web ページ要素の組合せの数は m^2 となる．したがって，Web 要素の組の数は ${}_{|P|}C_2 \times m^2$ となり，これらの全てに対して BruteForce アルゴリズムを適用すると，計算コストは多大なものになる．

そこで本論文では，BruteForce アルゴリズムを適用する要素の組を削減するために，事前に一定以上の包含率の可能性がないと考えられる要素の組を発見し，それらを除去する．

提案手法の概要を図 4 に示す．この図では例として，Web ページの集合 $P = \{p_1, \ldots, p_{|P|}\}$ を，Web サイト A，Web サイト B という 2 つの Web サイトに含まれるページの集合とする．また，同じ Web サイトのページでの Web 要素間の包含関係は計算しないものとする．

各 Web ページ p_i に含まれる Web ページ要素の集合を $E_i = \{e_1, \ldots, e_{|E_i|}\}$ と表記する．また，p_i と e_j に含まれる単語の集合から stop word を除いたものをそれぞれ $W(p_i)$, $W(e_j)$ とする．

提案手法は，次の 2 つのフェーズから構成される．

フェーズ 1: ページの組合せの削減． 可能性の無い Web ページの組を除去する．

フェーズ 2: Web ページ要素の組合せの削減． 可能性の無い Web ページ要素の組を除去する．

次に，これらを説明する．

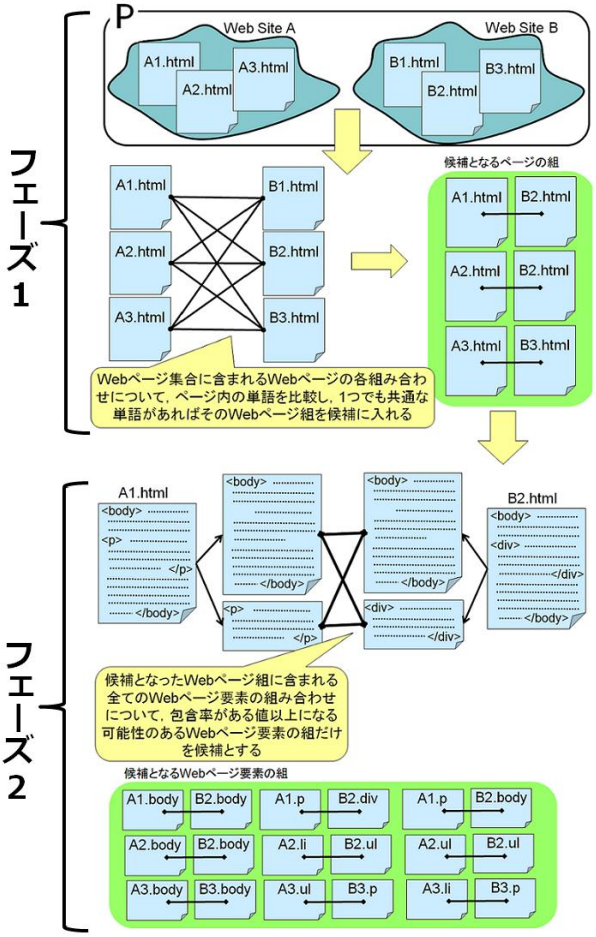

図 4　提案アーキテクチャ

フェーズ 1． ページ集合 P の全ての組合せから，全く関係のないページの組を候補から除く．全く関係のないページの組とは，それらのページに含まれる単語が 1 つも一致しないようなページの組である．具体的には，次式で定義する Web ページの組の集合 R_1 を計算する．

$$R_1 = \{(p_i, p_j) | p_i, p_j \in P \wedge W(p_i) \cap W(p_j) \neq \phi\}$$

フェーズ 2． R_1 に含まれる Web ページの組のそれぞれに対して，Web ページ要素の組の集合を作成し，そこに含まれる Web ページの要素の組の中から，包含率がある値 c 以上になる可能性のあるものだけを残した集合 R_2 を計算する．詳細は次節で述べるが，具体的には R_2 は次のようになる．

$$R_2 = \{(e_k, e_l) | e_k \in E_i \wedge e_l \in E_j \wedge (p_i, p_j) \in R_1 \wedge InclusionFilter(e_k \subseteqq e_l, c)\}$$

$InclusionFilter(e_k \subseteqq e_l, c)$ は，Web ページ要素の組 (e_k, e_l) が，包含率 c 以上になる可能性がある場合に，真となる述語である．

4.1 *InclusionFilter*

まず，$InclusionFilter(x \subseteqq y, c)$ の基本的なアイデア (図 5) について説明する．Web ページ要素 x，y に対応する (ストップワードを除いた) 単語集合 $W(x)$，$W(y)$ をあらかじめ辞書順でソートした単語列を X，Y とする．$InclusionFilter$ では，単語を辞書順に並べた際の範囲を利用し，Y が X 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 (図 5)．包含率は単語集合 X のうち，何%の単語が単語集合 Y に含まれているかで算出されるため，X，Y 中の単語を並べた際に範囲の共通部分が $c \cdot |X|$ 以上無ければ，包含率が c 以上になる可能性も無いと判断できる．具体的には，図 5(上)(下) のように X のサイズ (単語数) の $(1-c\%)$ 以上の範囲が，Y の単語列の範囲からはみ出している場合に，偽を返す．

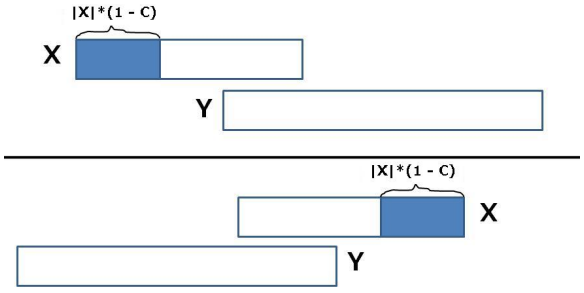

図 5　InclusionFilter の基本的なアイデア

具体的には，$InclusionFilter(x \subseteqq y, c)$ を計算するためには，まず x, y に対してそれぞれ 4 つ組 $T_x = [w_0^x, w_{(1-c)}^x, w_c^x, w_{last}^x]$ と $T_y = [w_0^y, w_{(1-c)}^y, w_c^y, w_{last}^y]$ を計算する．ここで，w_0^x は単語列 $X = W(x)$ における先頭の単語，$w_{(1-c)}^x$ は X の先頭から $1-c$ の割合の位置にある単語，w_c^x は X の先頭からから c の割合の位置にある単語，w_{last}^x は X の最後の位置にある単語 (T_y も同様) である．このとき，述語の値は次のようになる．

$$\begin{cases} false & \text{if } (w_{1-c}^x < w_0^y) \text{ or } (w_{last}^y < w_c^x) \\ true & otherwise \end{cases}$$

次に，$InclusionFilter$ の具体例 (図 6) について説明する．図 6 は $InclusionFilter(x \subseteqq y, 0.7)$ を計算する対象の Web ページ要素の組であり，Web ページ要素の組 (x, y) が，包含率 70% 以上になる可能性があるかどうかを計算する．対象とする Web ページ要素 x に対応する (ストップワードを除いた) 単語列には，$X = \{a, b, c, d, e, f, g, h, i, j\}$，Web ページ要素 y に対応する単語列には $Y = \{e, f, g, h, i, j, k, l, m, n, o\}$ という単語が含まれており，辞書順でソートされているものとする．図 6 では，xmiddle1 $(= w_{(1-c)}^x)$ < yfirst $(= w_0^y)$ となっているため，X 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無いことが分かる．したがって，偽を返す．

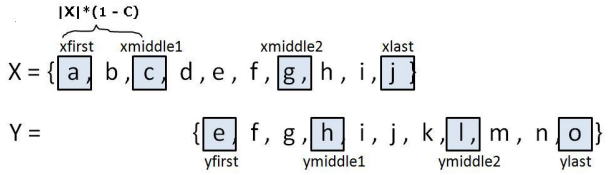

図 6　InclusionFilter の計算例

5. 実　　験

提案手法の効果を調査するための実験を行った．

5.1 実　　験

実験では，コンテンツが分散管理されている実際の Web サイトに対して提案手法を適用し，どの程度候補を減らすことができるか調査した．Web ページに含まれる単語集合からストップワードを除く際に用いる英語のストップワードのリストとして SMART stop list [9] を利用し，日本語については接続詞・助動詞・助詞・記号をストップワードとした．

対象としたのは筑波大学情報学群の Web サイト [10] 中のページ，および情報科学類の Web サイト中のページである．ただし，全てのページの組合せを候補とするのではなく，情報学群のページと情報科学類のページ間の組合せだけを考える．

実験結果. 提案手法の適用結果を図 7 に示す．

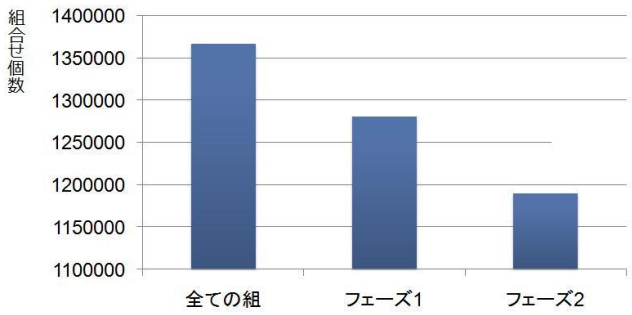

図 7　提案手法の効果

提案手法を適用することにより，フェーズ 1 では 8%，フェーズ 2 では 5%程度 Web ページ要素の組合せ数が減少した．この結果は，特にフェーズ 2 に関しては，予想より悪いものであった．その理由としては Web ページ要素をソートして並べた際に共通範囲が重なっていても，実際の単語集合に含まれる単語自体は一致していないような Web ページ要素の組が多数存在したものと考えられる．

6. まとめと今後の課題

本稿では，既存の Web コンテンツから，Web ページの要素間のコンテンツの包含関係を効率よく発見するための手法を提案した．具体的には，全ての Web ページの要素の組み合わせに対して厳密な包含関係を計算するのではなく，あらかじめ可能性の無いものを候補リストから削除する手法を提案した．実験の結果では，ページの組合せに関して 8%程度，要素の組合せに関して 5%程度の削減が可能であった．今後の課題として

は，より詳細な実験等を通じて，削減率を向上させるための仕組みを開発することがあげられる．

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 (#19024006)，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)(#20800076) による．

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 年 3 月．

[2] Natsumi Sawa，Atsuyuki Morishima，Shigeo Sugimoto，Hiroyuki Kitagawa．Wraplet: Wrapping Your Web Contents with a Lightweight Language．Proc. IEEE SITIS' 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, 2008 年 3 月．

[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 年 3 月．

[5] Serge Abiteboul, Richard Hull, Victor Vianu: Foundations of Databases. Addison-Wesley 1995.

[6] Chuan Xiao, Wei Wang, Xuemin Lin, Jeffrey Xu Yu. Efficient Similarity Joins for Near Duplicate Detection. www2008, 2008.

[7] J. Bauckmann, U. Leser, F. Naumann. Efficiently Computing Inclusion Dependencies for Schema Discovery. InterDB'06 (ICDE Workshop), 2006.

[8] 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, pp. 127-132, 福島，2008 年 9 月．

[9] SMART stop list, "http://members.unine.ch/jacques.savoy/clef/englishST.txt", (参照 2009-02-20)

[10] 筑波大学情報学群，"http://inf.tsukuba.ac.jp/"，(参照 2009-02-20)