情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

Web ラッパのアグリゲーションサービスへの適用と評価

中野 雄介[†1]　寺西 裕一[†2]　西尾 章治郎[†2]

近年，複数の Web アプリケーションの情報をひとつのページにまとめる，アグリゲーションサービスが注目されている．ユーザは複数の Web アプリケーションのページを表示することなく，ひとつのページで Web アプリケーションが提供する情報を比較，集計することができる．しかし，アグリゲーションサービスには個々の Web アプリケーションの画面から必要な部分を抽出するためのルールが必要となる．このようなルールの作成とメンテナンスのためのコストを抑える必要がある．そこで我々は，これまで研究してきた Web ラッパを改良し，Web アプリケーションのデザイン変更に自動追従する Web ラッパを提案する．また，これをアグリゲーションサービスに適用し，その有効性を評価した．評価の結果，既存手法における月平均のルール修正回数が 4 回であったのに対し，提案手法は 2.7 回であり，修正回数を抑えることができた．また，既存手法で 1 つのルールの作成にかかる稼働は 3.2 人日，1 回のルールの修正にかかる稼働は 1.1 人日であったのに対し，提案手法はそれぞれ，1.2 人日，0.4 人日であり，大幅に管理コストを削減できることを確認できた．

Application of Web Wrappers to Aggregation Services and Evaluation

Yuusuke Nakano ,[†1] Yuuichi Teranishi [†2]
and Shojiro Nishio [†2]

Aggregation services which enable users to see multiple web applications on one web page are getting popular. The users can compare and count the information from multiple web applications without showing each web application on their PCs' screen. However, the aggregation services need rules which indicate the segments to be extracted from the web applications. To provide the aggregation services in a low-cost way, the service providers have to reduce the cost of describing and maintaining the rules. Hence, we improve our wrapper to track changes of the web applications' design automatically and made an experiment to evaluate the effectiveness of our web wrapper by applying the wrapper to aggregation services. As a result of the experiment, we found that the existing method needs 4 adjustments of the rule a month and the proposed method needs 2.7 adjustments. In addition, the existing method took 3.2 man-days to describe one rule and 1.1 man-days to adjust one rule and the proposed method took 1.2 man-days for description and 0.4 man-days for adjustment.

1. はじめに

近年，アグリゲーションサービスによる複数の Web アプリケーションの統合が注目を集めている．例えば，複数の銀行口座の Web アプリケーションをひとつの Web ページに集約するアカウントアグリゲーションサービスが有名である．このようなアグリゲーションサービスにより，ユーザは複数の Web ページを表示する事無く，1 つの Web ページ上で複数の Web アプリケーションが提供する情報を比較，集計することができる．しかし，このようなアグリゲーションサービスは個々の Web アプリケーションの画面から必要な部分を抽出する，スクリーンスクレイピング技術によって実現されることが多く，アグリゲーションサービス提供者はスクレイピングのためのルールの作成，メンテナンスのために多くの稼働を割く必要がある．

一方，我々は近年の Web サービスの需要の高まりに応えるために，既存の Web アプリケーションを Web サービスとして利用可能とするラッパをの研究を続けてきた．Web サービスはソフトウェアコンポーネントであり，SOAP や REST で記述されたメッセージをやり取りすることで，ネットワークを介した相互運用が可能となるよう作られる．これにより，Web サービスは通常の関数呼び出しと同様の手順で，他のシステムからの利用が可能でとなる．我々のラッパは Web サービスのプロトコルである SOAP と Web アプリケーションのプロトコルとを相互変換することで，既存の Web アプリケーションをラップし，Web サービス化する．多様な Web アプリケーションをラップするために，ラッパはラップ対象の Web アプリケーションごとに，Web アプリケーションのプロトコルと Web サービスのプロトコルとを相互変換するルールが書かれたコンフィグファイルを用いる．例えば，ホテル検索 Web アプリケーションをラップして，ホテル検索 Web サービスを作る場合，コンフィグファイルを記述する必要がある．従って，膨大な数の Web サービスを提供する場合，膨大な数のコンフィグファイルを記述する必要があり，これはラッパによる Web サービス

†1 日本電信電話株式会社 NTT ネットワークサービスシステム研究所
NTT Network Service Systems Laboratories, NTT Corporation
†2 大阪大学大学院 情報科学研究科
Graduate School of Information Science and Technology, Osaka University

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

作成の妨げとなる．この問題は一般的な Web ラッパ全体に言えることである．コンフィグファイルの記述において最も時間のかかる作業は，Web アプリケーションから出力された HTML ドキュメントの抽出部分を指定するルールの記述である．このルールにより，ラッパは Web アプリケーションの出力から Web サービスの出力に変換することが可能となる．このルールの作成を支援するために，Web アプリケーションが生成する HTML ドキュメントの重要な部分を自動抽出する手法が必要である．

そこで，我々は検索機能を提供する Web アプリケーションが生成する HTML ドキュメントから検索結果の部分を自動抽出することで，コンフィグファイルの作成を支援するラッパツールをこれまで開発してきた[1),2)]．ラッパツールは柔軟な手法を用いて，Web アプリケーションの検索結果の HTML ドキュメントから検索結果の部分を自動抽出する．本手法は周期的なパターンを持つ部分が検索結果であると判断し，その部分を自動抽出するが，多様な Web アプリケーションが存在し，必ずしもすべての Web アプリケーションが出力する HTML ドキュメント内の検索結果の部分が整った周期的なパターンを持つわけではない．このため，多少周期的ではなくても自動抽出可能な柔軟な手法を用いる．

このようなラッパは Web アプリケーションの検索結果のページから検索結果の部分のみを抽出するため，一種のスクリーンスクレイピングであると言える．このため，これまで研究してきたラッパをアグリゲーションサービスに適用することで，アグリゲーションサービスの作成，メンテナンスコストを抑えることができると考えられる．本稿では，アグリゲーションサービスの作成，メンテナンスにかかるコスト抑えるために，Web アプリケーションから生成された HTML ドキュメントの構造変化に追従するラッパを提案する．また，本ラッパをアグリゲーションサービスに適用することで，提案手法の有効性を評価結果についても述べる．

2. 関 連 研 究

Web ラッパの生成に関する様々な研究が行われている[3)]．

HTML ドキュメントからの重要部分の抽出のために，HTML ドキュメント内のパターンを記述するための言語を定義し，半自動的にラッパを作ることのできるシステムが 4) で提案されている．また，Word-based Heterogeneous Information Rpresentation Language (WHIRL) と呼ばれるロジックベースの言語を定義し，ヒューリスティックな手法によってラッパを作成するシステムが 5) で提案されている．

機械学習を応用したラッパ生成システムが 6) で提案されている．このシステムのユーザは学習データを入力すると，システムはラッパを生成する．本文献中では left-right (LR) と呼ばれるフォーマットも提案されている，LR ラッパは抽出するべき部分を囲む，対となる文字列を持つ．加えて，本文献では header-tailer LR (HLRT), open-close LR (OCLR), header-tailer open-close LR (HOCKRT) と呼ばれるラッパとこれらのラッパを生成するためのアルゴリズムが提案され，それぞれのラッパの比較に関して述べられている．

機械学習を応用した半自動的なラッパ生成手法が 7) で提案されている．本手法により，半構造化データのタグの構造を用いて抽出を行う，TreeWrapper を生成することができる．半構造化データはタグによる階層構造で表現されるため，本手法は有効である．このような構造によって，ルートのタグから抽出箇所のタグまでのパスを記述することで，抽出するべき箇所を指定することが可能となる．8)–10) ではラッパや学習データを生成するための GUI によるサポートに関して述べられている．

以上で紹介した関連研究はすべて，学習のための例を生成するなどの，ユーザによる手作業が必要である．一方，学習データの入力などを必要としない，自動的にラッパを生成するための研究も行われてきた．

HR，TD，TR，A，P，BR などのようなタグが抽出するべき部分を示す特別なタグであると仮定し，これらを用いて抽出する手法が 11) で提案されている．また，最も長い文字列の繰り返しパターンを見つけることで，抽出位置を発見する手法が 12) で提案されている．

13)，14) で提案されているラッパ生成システムのユーザは HTML ドキュメントといった半構造化データを複数入力し，本システムは入力された複数の HTML ドキュメントを比較することで，変化のある部分を発見し，その部分を抽出するためのラッパを生成する．

また，Web アプリケーションのソースコードからラッパを生成する手法が 15) で提案されているが，ソースコードを公開していない，多くの Web アプリケーションからラッパを生成することはできない．

我々が文献 1)，2) で提案した方法は，ひとつの HTML ドキュメントからユーザによる少ない作業でラッパを生成できるという意味で，11)，12) の手法と同様に分類できる．これらの関連研究は HTML ドキュメント内のタグの並びから，抽出するべき部分を発見している．このため，これらの手法は少しでも不規則なタグが含まれると，抽出に失敗する可能性がある．一方，我々の手法は HTML ドキュメント内のタグの情報を用いず，タグの深度パターンを用いるため，不規則なタグが含まれている場合であっても正しく抽出できることを期待できる．

このように，HTML ドキュメントから重要な部分を抽出するための様々な Web ラッパ

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

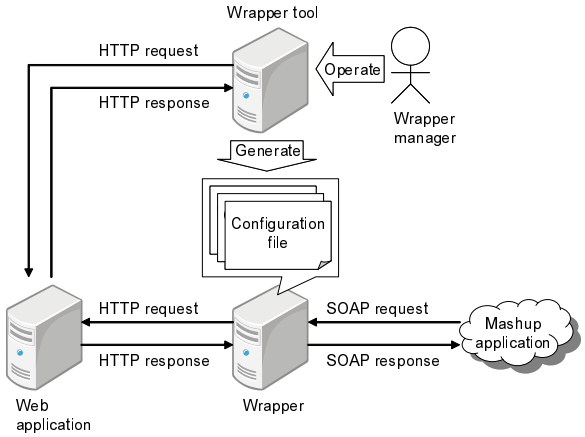

図 1 ラッパシステムの概要
Fig. 1 Overview of wrapper system

がこれまで提案されてきた．しかし，Web アプリケーションが生成する HTML ドキュメントの構造変化する場合には対応することは基本的にできなかった．

分類子の集合体を用い，抽出ルールを修復する Web ラッパが 16) で提案されている．それぞれの分類子はお互いに相関のない，プレフィックスやサフィックスのような，HTML ドキュメント内の抽出部分の特徴からなる．しかし，分類子を作成するために，対象の Web アプリケーションが生成する複数の HTML ドキュメントが必要であり，このような手間のかかる作業がラッパ生成に必要となる．また，金融系の Web アプリケーションなど，厳しいセキュリティーポリシーで運用されている Web アプリケーションからは，十分に学習データを収集できないため，ひとつの HTML ドキュメントから自身を修正できるラッパを生成する手法が必要である．

3. Web アプリケーションの構造変化に対応するラッパ

この章では Web アプリケーションから検索結果の HTML ドキュメントを取得し，取得された HTML ドキュメントの構造変化に追従する Web サービス化ラッパを提案する．提案手法は，我々が文献 1)，2) で提案したラッパ生成手法を基本的に拡張するものである．以下ではまず本稿で想定する文献 1)，2) のラッパについて概要を説明する．

3.1 想定するラッパシステムの概要

文献 1)，2) で提案したラッパ生成手法について説明する．ラッパが動作するためには対象の Web アプリケーションごとにコンフィグファイルが必要である．コンフィグファイルはラッパ管理者によって記述される．ラッパ管理者はラッパツールと呼ばれる，コンフィグファイルの作成を支援するシステムを用いてコンフィグファイルを記述することで，ラッパを作成する (図 1)．まず，ラッパ管理者は対象の Web アプリケーションの URL をラッパツールに入力する．その後，ラッパツールは Web アプリケーションから検索結果の HTML ドキュメントを取得する．ラッパ管理者はラッパツールを操作することで，取得された HTML ドキュメントからコンフィグファイルを生成する．生成されたコンフィグファイルを適用することで，ラッパは対象の Web アプリケーションをラップすることができる．このようにして，Web アプリケーションは Web サービスとして利用可能となる．

3.2 Web アプリケーションの構造変化に追従するラッパ

ラッパツールにより作成されたコンフィグファイルにより，ラッパは対象の Web アプリケーションから検索結果の部分を抽出するが，Web アプリケーションが生成する HTML ドキュメントの構造が変化すると，コンフィグファイルによる抽出に失敗してしまう．このような課題を解決するために，コンフィグファイルにより抽出に失敗した際に，コンフィグファイル内の抽出するべき部分を指し示す記述を再生成する手法を提案する．この再生成は下記のようにして行われる．

Step 1 コンフィグファイルに従って，対象の HTML ドキュメントから検索結果の部分を抽出する．

Step 2 抽出が失敗していることを判定する．

Step 3 HTML ドキュメントから重要な部分の候補を抽出する．

Step 4 抽出された候補から最も有望なものを選択する．

Step 5 Step 4 で選択された部分の位置をコンフィグファイルに書き出すことで，抽出対象の部分を指し示す記述を再生成する．

なお，Step3 の HTML ドキュメントからの重要な部分の候補の抽出は，HTML ドキュメントの各タグの入れ子の回数をタグの出現順に並べ，その数列を FFT することで実現する．このようにして，主要な周波数成分を見つけ，その周波数成分を持つ部分を抽出すべき部分と判断する．詳しくは文献 1)，2) を参照されたい．以下でそれぞれのステップに関して詳細に述べる．

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

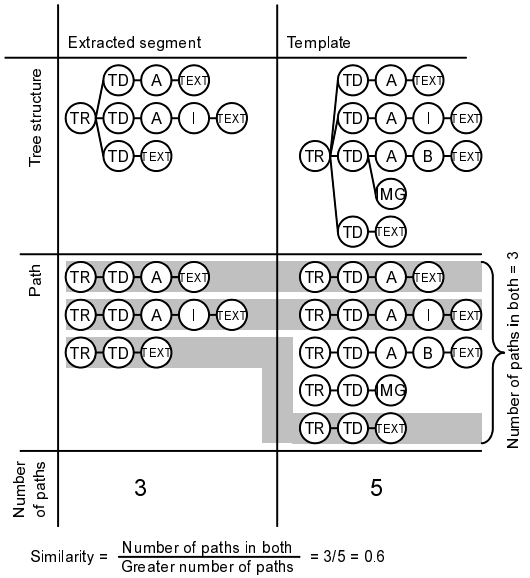

図 2 類似度の計算
Fig. 2 Similarity calculation

3.2.1 コンフィグファイルに従った抽出

ラッパはコンフィグファイルに従って HTML ドキュメントから対象の部分を抽出する．コンフィグファイルは XSLT で記述されており，その中には対象の部分の位置を指し示す記述がある．これは XPath で記述され，この場合は HTML タグから，例えば TABLE や UL といった検索結果の部分を囲むタグまでのパスを表す．このような記述を用いることで，ラッパは検索結果の部分を抽出することができる．

3.2.2 抽出失敗の判定

以上のようにして，コンフィグファイルに従った抽出の結果，抽出に失敗した場合はコンフィグファイルの記述を再生成する必要がある．以降は本論文の提案手法である，Web アプリケーションの構造変化への追従手法について説明する．追従するかどうかを決定するために，ラッパはコンフィグファイルに書かれたテンプレートを用いて抽出の失敗を判定する．このテンプレートは対象の部分を構成するタグを含む木構造で表される (図 2 の “Template” 列，“Tree structure” 行)．テンプレートはコンフィグファイルの生成時にラッパツールによって生成されるため，ラッパ管理者はテンプレートを手書きする必要はない．テンプレートの木構造と抽出された部分の木構造との類似度を算出することで，コンフィグファイルに予め記述されているしきい値より類似度が下回った場合，ラッパは抽出に失敗したと判定する．類似度を次のように算出する．各木構造のルートノードからリーフノードまでの部分木をパスと呼ぶ．ここでは，抽出された部分の木構造に含まれるパスの集合を $\mathcal{P}_e$，テンプレートの木構造に含まれるパスの集合を $\mathcal{P}_t$ とする．また，$I_{P1}(p)$ はパスの集合 $\mathcal{P}1$ にパス p が含まれている場合は 1，含まれていない場合は 0 を返す関数であるとする．このとき，パスの類似度 Similarity を以下のとおり定義する．

$$Similarity = \frac{\sum_{p \in \mathcal{P}_e} I_{P_t}(p)}{max\{|\mathcal{P}_e|, |\mathcal{P}_t|\}}$$

図 2 の例を用いて説明すると，抽出された部分には TR-TD-A-TEXT, TR-TD-A-I-TEXT, TR-TD-TEXT というパスが存在し，テンプレートにもこれらの 3 つのパスが存在するため，類似度の分子は 3 となる．また，テンプレートの方が抽出された部分よりも多い，5 つのパスを持つため，分母は 5 となり，類似度は 3 / 5 = 0.6 となる (図 2 の下の式)．

このようにして，各検索結果の抽出が成功しているかどうかを類似度と類似度のしきい値とを比較することで判定し，全検索結果内で抽出に成功している検索結果の数の割合を算出する．算出された割合がしきい値を下回った場合，その HTML ドキュメントから，検索結果の部分の抽出に失敗したと判定する．

3.2.3 再 生 成

抽出の失敗を判定した後，ラッパは XPath の記述の再生成を開始する．まず，ラッパは HTML ドキュメントから検索結果の部分の自動抽出を行う．この抽出はラッパツールで用いられる手法と同様の手法で行われ，タグの特徴的な深度変化のパターンのある複数の部分を候補として抽出する．その後，ラッパはテンプレートを用いることで最も有望な候補を選択する．このとき，ラッパはテンプレートの木構造とそれぞれの候補の木構造との類似度を算出し，最も類似度の高い候補を最も有望な候補，つまり，検索結果の部分として選択する．最終的に，ラッパは最も有望な候補があった部分を抽出するための XPath をコンフィグライルに追記する．

再生成の後，XSLT ベースの抽出に失敗した場合，ラッパは追記された XPath を用いて検索結果の部分を抽出する．もし，追記された XPath を用いても抽出に失敗した場合，ラッパは先で述べた手法を用いて別の XPath の記述を生成する．抽出の失敗毎に再生成を繰り返すことで，ラッパは自動的に多様な HTML ドキュメントからの検索結果部分の抽出が可

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

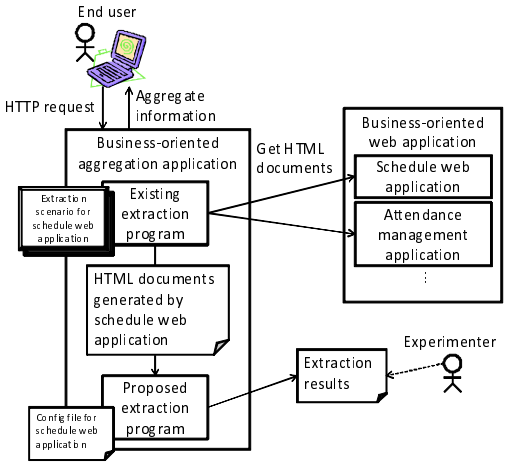

図 3 実証実験環境
Fig. 3 Condition of demonstration experiment

能となる．

4. 評　　価

我々は追従手法の有効性を評価するために提案手法を実装し，実証実験を行った．これにより，提案手法はラッパの管理コストを大幅に低減できることを検証した．

4.1 実証実験の手法

提案手法の有効性を評価するために，既存の業務用情報集約アプリケーションに提案手法を導入し，既存の抽出手法と提案手法との比較を行うための実証実験を行った．本実証実験においては，業務用情報集約アプリケーション内で稼働している既存の抽出手法による抽出の失敗回数と，提案手法による抽出の失敗回数とを，スケジュールアプリケーションからスケジュール部分を抽出において比較した．また，ラッパ作成時の稼働と抽出失敗の修正稼働に関しても比較した．

図 3 に実証実験環境の概要を示す．エンドユーザは業務用情報集約アプリケーションにブラウザを介して HTTP リクエストを送信する．すると，業務用情報集約アプリケーション内の既存の抽出プログラムは予め各 Web アプリケーション毎に設定された抽出シナリオ (Java のクラスで，抽出対象の HTML ドキュメントを取得するための対象 Web アプリケーションとのやり取りと，HTML ドキュメントからの抽出手順に関して記述されている) に従って，そのユーザに関する情報を複数の業務用 Web アプリケーション (スケジュールアプリケーションや勤怠管理アプリケーション等) から HTML ドキュメントの形で取得し，取得された各 HTML ドキュメントからユーザに関する情報を抽出する．その後，これらの抽出結果は業務用情報集約アプリケーションによってまとめてエンドユーザに提示される．同時に，提案手法と抽出シナリオベースの既存手法とを比較するために，スケジュールアプリケーションの HTML ドキュメントのみが提案手法のプログラムに入力され，提案手法のプログラムはコンフィグファイルに従ってスケジュールの部分を抽出する．このようにして抽出された結果を評価者が確認し，抽出に失敗していた場合はコンフィグファイルの修正を行い，このコンフィグファイルの修正回数を抽出失敗の回数としてカウントしたした．なお，既存の抽出手法による抽出の失敗回数に関しては，抽出シナリオの修正回数とした．

4.2 実証実験による評価結果

以上の実証実験を 2008 年 9 月から 2009 年 1 月までの間で行い，約 2600 回の抽出を繰り返した．表 1 に実証実験の結果を示す．提案手法における月平均のコンフィグファイル修正回数は 2.7 回であり，既存手法の 4 回よりも修正回数を抑えることができた．また，提案手法で 1 つのコンフィグファイルの作成にかかる稼働は 1.2 人日，1 回のコンフィグファイルの修正にかかる稼働は 0.4 人日であり，既存手法のそれぞれ 3.2 人日，1.1 人日よりも作成・修正にかかる稼働を抑えることができた．このことから，1 つの Web アプリケーションのラッパの管理にかかる稼働は，提案手法では月平均 1.08 人日である一方，既存の抽出手法は 4.4 人日となり，大幅な管理コストの削減を実現できることを確認できた．

抽出失敗回数が大幅に削減された理由としては，提案手法が HTML ドキュメントの構造変化に自動追従出来ていることが挙げられる．本実証実験において対象としたスケジュール Web アプリケーションはユーザによって見た目をカスタマイズできる機能を持っていた．このため，ユーザによって HTML ドキュメントの構造は多様に変化し，既存の抽出手法ではこのような構造変化に対応するためには，抽出に失敗する毎にシナリオを修正する必要が

表 1 実証実験結果
Table 1 Result of demonstration experiment

	月平均コンフィグファイル修正回数	コンフィグファイル作成稼働	コンフィグファイル修正稼働	月平均の管理稼働 (月平均回数 × 修正稼働)
提案手法	2.7 回	1.2 人日	0.4 人日	1.08 人日
既存手法	4 回	3.2 人日	1.1 人日	4.4 人日

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

あった．一方，提案手法では，自動的に抽出の失敗を検知し，XPath の再生成を行うため，失敗回数を大幅に減少させることができた．

また，コンフィグファイルの作成にかかる稼働が大幅に減少した理由としては，既存の抽出手法は Java のプログラムを作成する必要がある一方，提案手法は抽出対象部分の自動抽出によるコンフィグファイル作成支援を提供していたことが挙げられる．既存の抽出手法では HTML ドキュメントの中から特徴的な文字列を探し出し，その文字列を手がかりとして，対象の部分を抽出するためのプログラムをすべて人が作る必要がある．一方，提案手法は対象の部分までの XPath を自動生成する．このため，コンフィグファルの作成にかかる稼働が大幅に減少した．また，修正の稼働が減少した理由に関しても，既存手法は Java のプログラムの修正が必要であるが，提案手法は XSLT の修正のみであることが理由としてあげられる．

なお作業者からは，コンフィグファイルに含まれるしきい値のチューニングの自動化に関する課題が挙がった．提案手法は抽出の失敗を判断するために類似度としきい値との比較を行うが，このしきい値の設定に稼働がかかるため，しきい値のチューニングの自動化，または支援が必要という課題である．今回の実証実験でのコンフィグファイルの作成において，作業者ははじめ，自身の経験による根拠のない値でしきい値を設定する．その後，複数の HTML ドキュメントからの抽出を試み，正しく抽出でき，できるだけ高いしきい値を探る．このような作業が作業者にとって負担となることがわかった．

5. おわりに

本稿では，Web アプリケーションから生成される HTML ドキュメントの構造変化に追従することで，抽出の失敗と管理コストの低減するラッパを提案した本ラッパは抽出の失敗を判定し，抽出のための新たな XPath を自動的に再生成する．本ラッパの有効性を評価するために，実際に稼働している業務用情報集約アプリケーションに提案手法を導入し，既存の抽出手法との比較を行うことで，提案手法により大幅に管理コストを低減できることを確認した．

今後は，しきい値のチューニングの負担を抑える仕組みについて研究をすすめる．同時に，HTML ドキュメント内の意味情報も用いることで，HTML ドキュメントの構造を手がかりとした手法では追従できなかった Web アプリケーションへの対応を目指す．また，Flash や Javascript 等を用いた動的な HTML ドキュメントからの検索結果の部分の抽出手法に関しても研究を行う．

参 考 文 献

1) Nakano, Y., Yamato, Y., Takemoto, M. and Sunaga, H.: Implementation and Evaluation of Wrapper System that Creates Web Services from Web Applications, *IPSJ Journal*, Vol.49, No.2, pp.727–738 (2008).
2) Nakano, Y., Yamato, Y., Takemoto, M. and Sunaga, H.: Method of creating web services from web applications, *SOCA '07: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, Washington, DC, USA, IEEE Computer Society, pp.65–71 (2007).
3) Yamada, Y., Ikeda, D., Sakamoto, H. and Arimura, H.: Information Extraction from the Web : Automatic Generation of Web Wrappers(¡Special Issue¿Text Processing for Intelligently Accessing Information on the WWW), *Journal of Japanese Society for Artificial Intelligence*, Vol.19, No.3, pp.302–310 (2004-05-01).
4) Atzeni, P. and Mecca, G.: Cut and paste, *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, New York, NY, USA, ACM, pp.144–153 (1997).
5) Cohen, W.W.: WHIRL: A Word-based Information Representation Language, *Artificial Intelligence*, Vol.118, pp.163–196 (1999).
6) Kushmerick, N.: Wrapper induction: efficiency and expressiveness, *Artif. Intell.*, Vol.118, No.1-2, pp.15–68 (2000).
7) Murakami, Y., Sakamoto, H., Arimura, H. and Arikawa, S.: Extracting Text Data from HTML Documents, *IPSJ SIG Notes*, Vol.2001, No.27, pp.21–24 (2001).
8) Baumgartner, R., Flesca, S. and Gottlob, G.: Visual Web Information Extraction with Lixto, *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp.119–128 (2001).
9) Minton, S.N. and Ticrea, S.I.: Trainability: Developing a responsive learning system, *In Proceedings of the 2003 Workshop on Information Integration on the Web (IIWeb-03*, pp.27–32 (2003).
10) : Dapper, http://www.dappit.com/index.php.
11) Embley, D.W., Jiang, Y. and Ng, Y.-K.: Record-boundary discovery in Web documents, *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, New York, NY, USA, ACM, pp.467–478 (1999).
12) Chang, C.-H. and Lui, S.-C.: IEPAD: information extraction based on pattern discovery, *WWW '01: Proceedings of the 10th international conference on World Wide Web*, New York, NY, USA, ACM, pp.681–688 (2001).
13) Yamada, Y., Ikeda, D. and Hirokawa, S.: Automatic Wrapper Generation for Multilingual Web Resources, *DS '02: Proceedings of the 5th International Conference*

Vol.2010-DPS-145 No.38
Vol.2010-GN-77 No.38
Vol.2010-EIP-50 No.38
2010/11/26

on Discovery Science, London, UK, Springer-Verlag, pp.332–339 (2002).
14) Crescenzi, V., Mecca, G. and Merialdo, P.: RoadRunner: automatic data extraction from data-intensive web sites, *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, New York, NY, USA, ACM, pp.624–624 (2002).
15) Huy, H.P., Kawamura, T. and Hasegawa, T.: Web Service Gateway - A Step Forward to E-Business, *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, Washington, DC, USA, IEEE Computer Society, p.648 (2004).
16) Chidlovskii, B.: Automatic Repairing of Web Wrappers by Combining Redundant Views, *ICTAI '02: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC, USA, IEEE Computer Society, p.399 (2002).
17) : scrubyt, http://scrubyt.org/.