情報処理学会論文誌 Vol.53 No.8 2018–2027 (Aug. 2012)

推薦論文

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,a)] 寺西 裕一[2,3] 西尾 章治郎[4]

受付日 2011年11月25日, 採録日 2012年5月12日

概要：近年，複数のWebアプリケーションの情報を1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．アグリゲーションサービスによって，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，1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の実現には個々の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．このルールは，Webアプリケーションが出力するHTMLの構造に応じて変える必要があるため，メンテナンスのコストが大きくなってしまう．そこで本研究では，アグリゲーションサービスを実現するコンポーネントとなるWebラッパを改良し，Webアプリケーションの HTMLの構造変化に自動追従可能とする手法を提案する．また，本手法を用い，既存のWebアプリケーションが生成するHTMLの構造変化への追従成功率を測定した．この結果，構造が頻繁に変化する多様な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に追従できることを確認した．加えて，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．

キーワード：スクレイピング，Webラッパ，アグリゲーション

Application of Web Wrappers to Aggregation Services and Evaluation

Yuusuke Nakano[1,a)] Yuuichi Teranishi[2,3] Shojiro Nishio[4]

Received: November 25, 2011, Accepted: May 12, 2012

Abstract: 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 improved an web wrapper, a component to enable aggregation service which we proposed in the former work, to track changes of the web applications' design automatically. We made an experiment to evaluate the effectiveness of our wrapper by calculating the success rate of tracking and applying the wrapper to aggregation services. As a result of the experiment, we found that our wrapper tracks changes of HTML documents generated by a wide variety of web applications. In addtion, we found that the existing method needs 4 adjustments of the rule a month and the proposed method needs 2.7 adjustments in average.

Keywords: scraping, Web wrapper, aggrigation

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

[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, Musashino, Tokyo 180–8585, Japan
[2] 大阪大学大学院サイバーメディアセンター
Cybermedia Center, Osaka University, Ibaraki, Osaka 567–0047, Japan
[3] 情報通信研究機構
National Institute of Information and Communications Technology, Koganei, Tokyo 184–8795, Japan
[4] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] nakano.yuusuke@lab.ntt.co.jp

本稿の内容は2010年11月のマルチメディア通信と分散処理研究会にて報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

プリケーションの統合が注目を集めている．たとえば，複数の銀行口座の Web アプリケーションを 1 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 [1]．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 ページを表示することなく，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．

多くの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．たとえば，複数のホテル検索 Web アプリケーションから，ホテル検索アグリゲーションサービスを作る場合，検索結果表示ページから検索結果部分を抽出するといったルールを作成する必要がある．多くの Web アプリケーションを用いたアグリゲーションを行う場合，膨大な数のルールの作成が必要となる．

一方，近年多くの Web アプリケーションが WebAPI も提供している．このような WebAPI を用いることで，HTML ドキュメントからのスクレイピングをせずに，アグリゲーションサービスを提供することが可能となる．しかし，古くから用いられている CGI ベースのアプリケーションや，社内向けの情報共有システム等はこうした対応が難しいものもあり，すべての Web アプリケーションが WebAPI も提供するわけではない．したがって，依然，スクレイピングのためのルールの作成は必要であり，アグリゲーションサービス提供者はこのようなルールの作成のために多くの稼働を割く必要がある．

こうしたルールの作成を支援するためには，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できることが望ましい．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，ルールを含むコンフィグファイルの作成を支援する Web ラッパをこれまで開発してきた [15], [16]．Web ラッパは，Web アプリケーションを Web サービスとして扱えるようラップする機能を持ち，これにより We アプリケーションをアグリゲーションサービスのコンポーネントとして用いることができる．文献 [15], [16] で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し，自動抽出された部分を抽出するためのルールの作成を支援する方法を提案している．この方法では，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しない限り，1 度作成されたルールでの抽出に失敗することはない．

しかし，Web アプリケーションが生成する HTML ドキュメントはその構造が変化する．たとえば，Web アプリケーションの提供者による Web アプリケーションのデザインの変更がなされた場合や，Web アプリケーションがユーザによってカスタマイズ可能な場合には，HTML ドキュメントの構造が時間やユーザによって変化することになる．こうした場合，アグリゲーションサービスのメンテナンスを行わなければ，必要な部分の抽出に失敗してしまう．一方，既存の Web ラッパ作成支援システムでは，重要な部分を抽出する機能のみを提供しており，動的なドキュメントの構造の変化に対応することは基本的にできなかった．

そこで本研究では，アグリゲーションサービスのメンテナンスにかかるコストを抑えるために，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ことが可能な Web ラッパの構成手法を提案する．本手法では，HTML ドキュメントの構造の変化を，自動検知し，抽出ルールの再生成を行う．これによって，ドキュメントの構造の動的な変化にともなうメンテナンスの手間を削減できる．また，本稿では本 Web ラッパの追従性能の評価，および，実際に本 Web ラッパをアグリゲーションサービスに適用する実証実験による有効性評価の結果についても述べる．

2. 関連研究

これまでに，Web アプリケーション等が出力する HTML ドキュメントを解析し，別のサービスで利用可能なように「ラップ」する Web ラッパの生成に関する様々な研究が行われている [2]．

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文献 [3] で提案されている．また，Word-based Heterogeneous Information Representation 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文献 [4] 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文献 [5] 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．このフォーマットを用いて実現される 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文献 [6] 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．文献 [7], [8], [9] 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等の，ユーザによる手作業が必要である．一方，学習データの入力等を必要としない，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 等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文献 [10] で提案されている．また，最も長い文字列の繰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文献 [11] で提案されている．

文献 [12], [13] 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文献 [14] 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 [15], [16] で提案した方法は，1 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，文献 [10], [11] 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への対応は基本的にはできなかった．

分類子の集合体を用い，抽出ルールを修復する Web ラッパが文献 [17] 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 アプリケーション等，厳しいセキュリティポリシで運用されている 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，1 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 [15], [16] 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 [15], [16] 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 [15], [16] 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

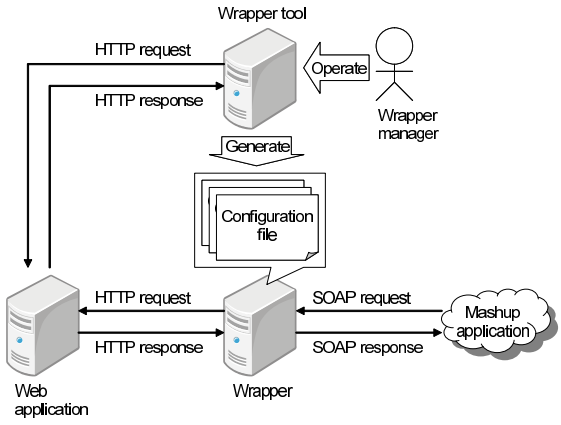

図 1 ラッパシステムの概要
Fig. 1 Overview of wrapper system.

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，Step 3 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 [15], [16] 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．

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，たとえば 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

以上のような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．以降は本稿の提案手法である，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し，コンフィグファイルにあらかじ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．

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

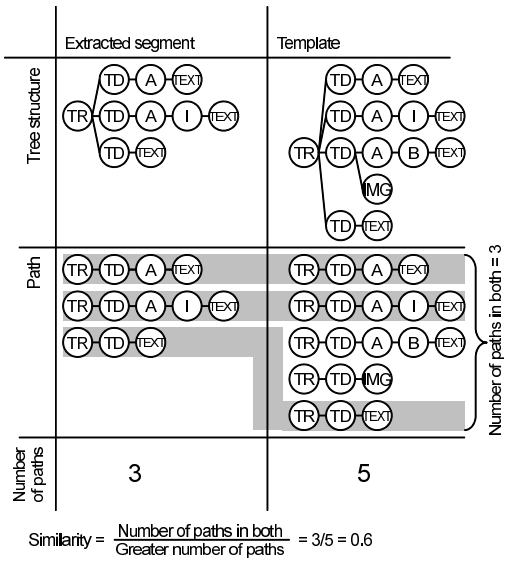

図 2 類似度の計算
Fig. 2 Similarity calculation.

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．

なお，しきい値として最適な値は，Web アプリケーションによって異なる．これは，Web アプリケーションによっては，完全に統一された HTML ドキュメントの構造で検索結果を返すものや，構造にある程度のバリエーションをもって検索結果を返すもの等の違いがあるためである．後者の Web アプリケーションについては，些細な HTML ドキュメントの構造の違い（フォントを変化させるためのタグの有無等）を抽出の失敗と判定しないよう，しきい値を必要以上に高く設定しないようにしなければならない．コンフィグファイルの作成者はしきい値設定用のサンプル HTML ドキュメントを用いて，正しく抽出していると判定できる，できるだけ高い値をしきい値として設定する．

このように，しきい値を設定することで，ラッパはサンプル HTML ドキュメントに含まれる構造変化であれば許容でき，無駄な追従を防止することができる．

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 をコンフィグファイルに追記する．

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．抽出の失敗ごと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能となる．

4. 評価

本稿では，HTML ドキュメントの構造変化に追従するための手法を提案している．本手法は大きく 2 つの機能から構成される．1 つが XSLT に従った HTML ドキュメントからの検索結果部分の抽出の失敗を判定する機能であり，もう 1 つが文献 [15], [16] の手法を用いて抽出された複数の検索結果部分の候補から，正しい検索結果部分を自動選定する機能である．本手法の有効性を示すために，HTML ドキュメントの構造変化への追従成功率と，アグリゲーションサービスの作成，メンテナンスコストの低減について評価した．

4.1 HTML ドキュメントの構造変化への追従の評価

追従手法の追従成功率を評価するために，提案手法を実装し，HTML ドキュメントの構造が変化した場合の抽出成功率の測定した．

4.1.1 追従の評価手法

頻繁に HTML ドキュメントの構造を変化させる既存の Web アプリケーションから検索結果の HTML ドキュメントを収集し，収集した HTML ドキュメントからコンフィグファイルを生成し，生成したコンフィグファイルをラッパに適用した．その後，提案した追従手法の有効性を検証するために，ラッパに収集した HTML ドキュメントを入力し，抽出成功率を算出した．

評価者の影響を避けるため，Web 関連の技術者が選択した Web アプリケーションから HTML ドキュメントを収集し，下記の手順で有効性を検証した．

Step 1 先の技術者が日頃使っている Web アプリケーションの中で，検索機能を備えるものそれぞれに，異なる 3 つのリクエストを別々の日に送信し，それらに対する HTML ドキュメントを収集する．

Step 2 Step 1 で収集した 3 つの HTML ドキュメントを比較し，リクエストの内容やリクエストする日によって，検索結果の HTML ドキュメントの構造が変化する Web アプリケーションを見つける．

Step 3 Step 2 で見つけた Web アプリケーションに 45 個のリクエストを送信することで，それぞれの Web アプリケーションから 45 個の HTML ドキュメントを収集する．リクエストはそれぞれの Web アプリケーションの検索ページのテキストフィールドに 45 個のキーワードを入力し，検索ボタンをクリックすることで行う．キーワードは Google Suggest [18] と呼ばれる，入力された文字列に対して，その文字列を頭文字とする，よく検索されるキーワードを出力する WebAPI に対して，50 音に含まれる 45 文字（あ行〜ら行）を入力することで生成する．

Step 4 Step 3 で収集した，各 Web アプリケーションが生成する HTML ドキュメントのうちの 1 つをラッパ生成ツールに入力し，コンフィグファイルを生成する．このときに入力する HTML ドキュメントは “あ” から始まるキーワードに対する検索結果の HTML ドキュメントである．

Step 5 Step 4 で生成したコンフィグファイルをラッパに適用し，Step 3 で収集した HTML ドキュメントをラッパに入力することで，HTML ドキュメントからの抽出を行い，抽出結果の適合率と再現率を計算する．適合率は抽出結果の中に正しい抽出結果が含まれる割合である．また，再現率は抽出するべき部分のうち，どれだけ抽出できているかを示す割合である．それぞれは，下記のように表される．

$$precision = \frac{|\{\text{extracted search results}\} \bigcap \{\text{targeted search results}\}|}{|\{\text{extracted search results}\}|}$$

$$recall = \frac{|\{\text{extracted search results}\} \bigcap \{\text{targeted search results}\}|}{|\{\text{targeted search results}\}|}$$

このようにして，収集された HTML ドキュメントから文献 [15], [16] で提案されているラッパ生成ツールを用いてラッパを生成するため，再生成の際に正しく検索結果部分を抽出できることは分かっている．このため，本評価は文献 [15], [16] の提案内容の評価を含まず，本稿で提案している失敗の判定と，抽出された複数の検索結果部分の候補か

表 1 追従の評価結果
Table 1 Result of tracking experiment.

Web アプリケーション名	Web アプリケーションの説明	適合率	再現率	構造変化のタイプ
allcinema	映画検索サービス	1.0	1.0	個人名が入力された場合，人物の検索結果のテーブルが映画の検索結果のテーブルの上に挿入される．
AmebaVision *	動画共有サイト	1.0	1.0	一定以上の検索結果数がある場合，次のページへのリンクが検索結果の上に挿入される．
Aucfan.com	オークションの横断検索サービス	0.47	0.42	検索キーによって，それぞれのオークションサイトの検索結果のテーブルが配置される位置が異なる．
Google	Web サーチエンジン	1.0	0.98	特定のキーワードが入力された場合，広告や関連するキーワードが検索結果の上に挿入される．
YouTube *	動画共有サイト	1.0	0.98	特定のキーワードが入力された場合，関連するキーワードが検索結果の上に挿入される．

* 本 Web アプリケーションは検索結果を複数のセグメントに分割するため，ラッパはこれらのセグメントのうちの 1 つを抽出する．

らの正しい検索結果部分の自動選定との 2 つの機能の有効性を示すものとなる．また，評価者以外が選択した Web アプリケーションに対して，Google Suggest が生成したキーワードを送信し，HTML ドキュメントを収集することで，評価者の影響を受けずに提案手法の有効性を評価した．

4.1.2 追従の評価結果

収集した HTML ドキュメントから対象部分を実際に抽出することで，提案手法を評価した．先の技術者が選定した Web アプリケーションの数は 45 個であり，それらのうちの 5 個は検索結果の HTML ドキュメントの構造を頻繁に変更していた．これらの Web アプリケーションは検索キーワードによって検索結果の HTML ドキュメントの構造を変化させており，検索するタイミングによって構造を変化させていなかった．このように選定された 5 個の Web アプリケーションに対して，Step 3〜Step 5 を実施した．

評価結果を**表 1** に示す．allcinema，AmebaVision，Google，YouTube の適合率は 1.0 であった．また，allcinema，AmebaVision の再現率は 1.0 であり，Google と YouTube の再現率は 0.98 であった．これはラッパは多様な Web アプリケーションが生成する HTML ドキュメントのほとんどすべてから検索結果を抽出することに成功したことを示す．しかしながら，Aucfan.com の適合率は 0.47 であり，再現率は 0.42 であることから，ラッパはある種の 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に追従できない場合もあることが分かる．

また，ラッパが HTML ドキュメントの構造変化に追従していることを確認するために，XPath の再生成回数と再生成された XPath の利用回数をカウントした．**表 2** に示すように，ラッパは HTML ドキュメントの構造変化に対して XPath を再生成し，再生成された XPath を利用できていることが分かる．たとえば，ラッパは allcinema の XPath を 3 回再生成し，各 XPath は複数回利用されている．このようにして，再生成された XPath を利用することで，ラッパは構造変化する HTML ドキュメントから検索結果部分を抽出することができる．

Aucfan.com における抽出の失敗の主な原因として，Aucfan.com は複数のオークション Web アプリケーションの検索結果をまとめて表示する Web アプリケーションであり，各オークション Web アプリケーションの検索結果の表示順が動的に変更されることがあげられる．各オークション Web アプリケーションの検索結果は類似したテーブルの構造を持っているため，抽出対象である特定のオークション Web アプリケーションの検索結果のテーブルの位置が変更されると，ラッパは間違ったオークション Web アプリケーションの検索結果部分を抽出してしまう．このようにして，ラッパは Aucfan.com の構造変化に追従することができなかった．

また，本手法は，YouTube の 1 つの HTML ドキュメントの構造変化に追従することができなかった．これは，ラッパが HTML ドキュメントからの検索結果部分の抽出結果から，正しい抽出結果の選定に失敗したためである．このような追従の失敗は，同じ Web アプリケーションによって生成された他の HTML ドキュメントからの XPath の再生成によって解決される．たとえば，検索結果の前に広告が

表 2 再生成の結果
Table 2 Result of XPath regeneration.

Web application name	Number of regenerations	Generated XPath	Generation point (n-th document)	Number of uses
allcinema	3	/html/body/table/tbody/tr/td[1]/table[2]/tbody/tr[1]/td[2]/div[2]/div[3]/table[2]/tbody	default	29
		/html/body/table/tbody/tr/td[1]/table[2]/tbody/tr[1]/td[2]/div[2]/div[2]/table[2]/tbody	6th	10
		/html/body/div[3]/table/tbody/tr/td[1]/div[2]/div[2]/div[3]/div[2]/div[4]/div/div[5]/div/div[1]/div[2]/table/tbody	7th	1
		/html/body/table/tbody/tr/td[1]/table[2]/tbody/tr[1]/td[2]/div[2]/div/table[2]/tbody	13th	5
AmebaVision	1	/html/body/div[2]/div[2]/div/div/div[4]	default	29
		/html/body/div[2]/div[2]/div[1]/div/div[2]	2nd	16
Aucfan.com	2	/html/body/div/div[3]/div[9]/div[1]/div[2]/table/tbody	default	36
		/html/body/div/div[3]/div[6]/div[1]/div[2]/table/tbody	3rd	6
		/html/body/div/div[3]/div[5]/div[1]/div[2]/table/tbody	32nd	3
Google	3	/html/body/div[3]/div[2]/ol	default	39
		/html/body/div[3]/div[3]/ol	4th	1
		/html/body/div[4]/div[2]/ol	18th	3
		/html/body/div[3]/div/ol	24th	2
YouTube	4	/html/body/div/div[5]/div/table/tbody/tr[1]/td	default	20
		/html/body/div/div[6]/div/table/tbody/tr[5]/td	3rd	5
		/html/body/div/div[6]/div/table/tbody/tr[2]/td	12th	5
		/html/body/div/div[5]/div[2]/table/tbody/tr[1]/td	26th	2
		/html/body/div/div[6]/div/table/tbody/tr[1]/td	28th	12

挿入され，既存の XPath での抽出に失敗したとする．さらに，正しい抽出結果を選定できず，XPath の再生成ができなかった場合，追従に失敗する．その後，同様に広告が検索結果の前に挿入された他の HTML ドキュメントから，既存の XPath での抽出に失敗する．しかし，今度は正しく XPath の再生成ができたとする．このようにして，1 度，他の HTML ドキュメントから XPath の再生成を行うと，先に追従に失敗した HTML ドキュメントからも，再生成された XPath を用いることで，正しく検索結果部分を抽出することができるようになる．

一方，ラッパは Google が生成する 6 つの HTML ドキュメントから，1〜2 個の検索結果を抽出するのに失敗した．これは，抽出に失敗した検索結果の前だけに img タグが挿入されていたためであった．XSLT ファイルにはこのような検索結果を抽出するための記述がなく，ラッパは抽出に失敗した．このような抽出の失敗を防ぐためには，ラッパ管理者が XSLT ファイルを修正する必要がある．

4.2 アグリゲーションサービスへの適用に関する評価

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．

4.2.1 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．なお，本実証実験では文献 [15], [16] で提案したラッパ生成手法の評価も行う．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のスケジュール部分の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．

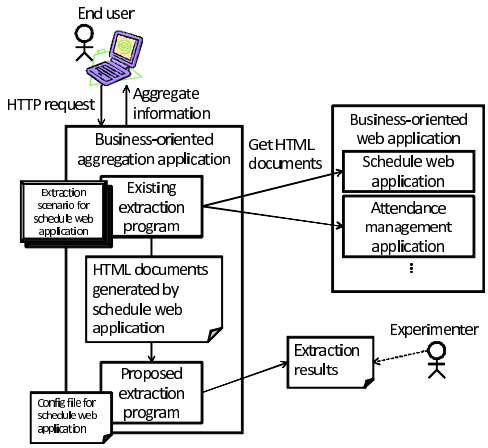

図 3 実証実験環境
Fig. 3 Condition of demonstration experiment.

図 3 に実証実験環境の概要を示す．エンドユーザは業

表 3 実証実験結果
Table 3 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

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あらかじめ各 Web アプリケーションごと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とりと，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．なお，本実証実験では業務用 Web アプリケーションとして，スケジュールアプリケーションを対象とし，スケジュールアプリケーションの HTML ドキュメントに含まれる日付情報，スケジュールの内容，ユーザ名等を抽出した．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．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．

4.2.2 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 2,600 回の抽出を繰り返した．**表 3** 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．

本実証実験はトライアルの位置づけであったため，数名程度のユーザ数であり，期間中に 2,600 回の抽出がなされた．これは 1 日あたり 30 回程度の抽出がなされた計算となる．なお，抽出回数の増加に比例し，既存手法，提案手法ともに修正回数が増大すると思われる．抽出回数の増大はユーザ数の増大以外にも，対象の Web アプリケーションの数，1 ユーザあたりの利用回数等の増大も原因となり，これらいずれの増大でも HTML ドキュメントの構造のバリエーションは増大する．このような構造のバリエーションの増大は，自動追従で対応できず，手作業によるコンフィグファイルの修正が必要な回数を増大させる．

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できていることがあげられる．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，抽出に失敗するごとにシナリオを修正する必要があった．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．

一方，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，文献 [15], [16] のラッパ生成手法が大きく貢献している．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，ラッパ生成手法で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あげられる．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，ラッパ生成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，ラッパ生成手法は XSLT の修正のみであることが理由としてあげられる．

なお，提案手法ではコンフィグファイルに，一定以上の類似度であれば抽出が成功していると判定するためのしきい値を設定可能としている．作業者からは，このしきい値のチューニングの自動化に関する課題があがった．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．このような作業がコンフィグファイルの作成にかかる稼働の大半を占めていたため，しきい値のチューニングの自動化，または支援が必要である．

また，依然，月 2.7 回のコンフィグファイルの修正が発生しており，既存手法の修正回数が 4 回であることから，自動追従が成功したのは 3 割程度であることが読み取れる．このため，さらなる自動追従の改良が必要であることも分かった．

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を低減させ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，HTML ドキュメントの構造変化に対する追従成功率を測定した．この結果，構造が頻繁に変化する多様な 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抽出できることを確認した．また，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等，さらなる追従の高性能化を目指す．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検討を進めたい．

参考文献

[1] goo: goo ワンビリング，入手先 ⟨http://billing.goo.ne.jp/⟩.

[2] 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).

[3] Atzeni, P. and Mecca, G.: Cut and paste, *PODS '97: Proc. 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp.144–153, ACM, New York, NY, USA (1997).

[4] Cohen, W.W.: WHIRL: A Word-based Information Representation Language, *Artificial Intelligence*, Vol.118, pp.163–196 (1999).

[5] Kushmerick, N.: Wrapper induction: Efficiency and expressiveness, *Artificial Intelligence*, Vol.118, No.1-2, pp.15–68 (2000).

[6] 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).

[7] Baumgartner, R., Flesca, S. and Gottlob, G.: Visual Web Information Extraction with Lixto, *VLDB '01: Proc. 27th International Conference on Very Large Data Bases*, pp.119–128, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA (2001).

[8] Minton, S.N. and Ticrea, S.I.: Trainability: Developing a responsive learning system, *Proc. 2003 Workshop on Information Integration on the Web* (*IIWeb-03*), pp.27–32 (2003).

[9] Dapper, available from ⟨http://www.dappit.com/index.php⟩.

[10] Embley, D.W., Jiang, Y. and Ng, Y.-K.: Record-boundary discovery in Web documents, *SIGMOD '99: Proc. 1999 ACM SIGMOD International Conference on Management of Data*, pp.467–478, ACM, New York, NY, USA (1999).

[11] Chang, C.-H. and Lui, S.-C.: IEPAD: Information extraction based on pattern discovery, *WWW '01: Proc. 10th International Conference on World Wide Web*, pp.681–688, ACM, New York, NY, USA (2001).

[12] Yamada, Y., Ikeda, D. and Hirokawa, S.: Automatic Wrapper Generation for Multilingual Web Resources, *DS '02: Proc. 5th International Conference on Discovery Science*, pp.332–339, London, UK, Springer-Verlag (2002).

[13] Crescenzi, V., Mecca, G. and Merialdo, P.: RoadRunner: Automatic data extraction from data-intensive web sites, *SIGMOD '02: Proc. 2002 ACM SIGMOD International Conference on Management of Data*, pp.624–624, ACM, New York, NY, USA (2002).

[14] Huy, H.P., Kawamura, T. and Hasegawa, T.: Web Service Gateway - A Step Forward to E-Business, *ICWS '04: Proc. IEEE International Conference on Web Services*, p.648, IEEE Computer Society, Washington, DC, USA (2004).

[15] 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).

[16] Nakano, Y., Yamato, Y., Takemoto, M. and Sunaga, H.: Method of creating web services from web applications, *SOCA '07: Proc. IEEE International Conference on Service-Oriented Computing and Applications*, pp.65–71, IEEE Computer Society, Washington, DC, USA (2007).

[17] Chidlovskii, B.: Automatic Repairing of Web Wrappers by Combining Redundant Views, *ICTAI '02: Proc. 14th IEEE International Conference on Tools with Artificial Intelligence*, p.399, IEEE Computer Society, Washington, DC, USA (2002).

[18] Google Suggest, available from ⟨http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=106230⟩.

推薦文

本稿では，アグリゲーションサービスの作成，保守にかかる費用を削減するために，Web アプリケーションから生成された HTML ファイルの構造変化に自動追従する Web ラッパを提案している．有用性の高い自動化手法が提案されており，コンフィグファイルにおける設定要素である

Xpath を自動再生成するアイディアも興味深い．以上より本稿は推薦論文に値する．

（マルチメディア通信と分散処理研究会主査　勝本道哲）

中野 雄介 （正会員）

2005 年和歌山大学大学院システム工学研究科修了．同年日本電信電話株式会社入社．同社，NTT ネットワークサービスシステム研究所にて Web マイニング，ユビキタスコンピューティング，サービスデリバリープラットフォーム，分散データベース研究に従事．2011 年大阪大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．2004 年情報処理学会第 66 回全国大会学生奨励賞受賞．2007 年電子情報通信学会学術奨励賞受賞．

寺西 裕一 （正会員）

平成 5 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学院基礎工学研究科博士前期課程修了．同年日本電信電話株式会社入社．平成 17 年大阪大学サイバーメディアセンター講師，平成 19 年同大学院情報科学研究科准教授，平成 20 年より情報通信研究機構専攻研究員，招へい専門員を兼任，平成 23 年より情報通信研究機構研究マネージャおよび大阪大学サイバーメディアセンター招へい准教授，現在に至る．博士（工学）（平成 16 年 3 月，大阪大学）．マルチメディア情報システム，ユビキタス応用システム等の研究開発に従事．本会論文賞を受賞．IEEE 会員．

西尾 章治郎 （正会員）

昭和 50 年京都大学工学部数理工学科卒業．昭和 55 年同大学院工学研究科博士後期課程修了．工学博士．京都大学工学部助手，大阪大学基礎工学部および情報処理教育センター助教授を経て，平成 4 年大阪大学工学部教授，平成 14 年大学院情報科学研究科教授となり，現在に至る．その間，大阪大学サイバーメディアセンター長，大学院情報科学研究科長，理事・副学長を歴任．データベースシステムにおけるデータおよび知識管理に関する研究に従事し，紫綬褒章，立石賞功績賞等を授与される．日本学術会議会員．本会では理事を歴任し，論文賞，功績賞を受賞．IEEE，電子情報通信学会フェロー．