

Web データ抽出システム Ducky における ブラウザ GUI と抽出ルール生成機構の構築

金岡 慧^{1,a)} 遠山 元道^{2,b)}

概要: Web は巨大な情報資源であり、これらのデータを 2 次利用のために得たいというニーズがある。しかしその場合、手作業で書き出すかスクレイピングプログラムを記述するなど、ユーザにとって負荷であったり、高度な知識が要求される。

これらの問題を解決するべく、著者らは *Ducky* を構築した。*Ducky* は対象となる Web ページの URL や CSS セレクタから成る独自のデータ抽出ルールに基づいてデータ抽出を行い、XML, JSON, CSV 形式のファイルとして出力するシステムである。その独自のデータ抽出ルールの文法により、抽出されるデータのノイズ除去や、複数ページにまたがる大型サイトからのデータ抽出・統合なども可能なことが特徴である。

本論文では *Ducky* の新たなブラウザ GUI を提案し、ユーザが直感的に目的の要素を選択できるようにする。ブラウザ GUI を通してユーザが選択した要素はデータ抽出ルールに変換され、システムに登録される。これにより、ユーザがデータ抽出ルールの定義を行う負荷を軽減し、GUI における直感的な操作のみで Web データ抽出が可能となることを示す。

Browser GUI for Rule Generation in Web Data Extraction System Ducky

KEI KANAOKA^{1,a)} MOTOMICHI TOYAMA^{2,b)}

Abstract: To gain the benefit of invaluable data from World Wide Web, manual extraction or creation of web scraping programs may be necessary. But these processes can be tedious and complicated. To address these, we have proposed *Ducky*, which is a Web data extraction system including a web wrapper that extracts data from web sources and translates them into structured data based on a user-defined data extraction rules. *Ducky* is able to extract data flexibly from various structured web pages, remove noise from extracted data and integrate data distributed to multiple pages from different sites. In this paper, we propose the browser GUI of *Ducky* to help users to extract the data. It can operate intuitively by the actions such as clicking, pointing a cursor (mouse over) to an objective elements. These users' actions are converted into data extraction rules in a configuration file. We hereby help users to extract the data by intuitive operations and reduce users' burden to write the configuration file.

1. はじめに

Web 上には膨大な量のデータが存在し、これらを 2 次利

用のために得たいというニーズがある。その場合、手作業で書き出すかスクレイピングプログラムを記述する方法が考えられる。前者の手動での記述の場合、手間と時間のコストが大きくなり、誤りが生じる危険性もある。一方後者のスクレイピングプログラムを記述する方法は、知識のないユーザにとっては敷居の高い方法となってしまう。そのため、それらのデータを得るために Web ページやサイトを機械的に処理する様々な手法がこれまでに提案されてき

¹ 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University

² 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Keio University

a) kei@db.ics.keio.ac.jp

b) toyama@ics.keio.ac.jp

た. このように Web を巨大な知識ベースと捉え, Web から有用な知識を取り出す情報抽出技術の研究は現在までに数多く行われてきた [12].

研究として問題となるのは, Web ページやサイトはそれぞれ HTML の構造が異なる点である. それらはその作成者によって人が認識・理解しやすい形でデザインされ, ブラウザ上に描画され, 利用されるということが前提になっている. この問題に対し, 著者らは HTML 文書からデータを抽出し, 構造化する Web ラッパーに注目した. Web ラッパーとは, Web ページから特定の情報を抽出する抽出ルールおよび抽出プログラムのことである. Web ラッパーはその抽出ルールによって HTML 文書から特定の情報の位置を把握し, 抽出を行うことで, 構造を再構成する.

Web ラッパーは自動生成, 半自動生成など, 自動のレベルの異なるものがこれまでに提案されてきた. 膨大な量の Web ページやサイト群に対し, Web ラッパーを半自動で生成することは現実的でないという観点から, Web ラッパーの自動生成の研究は行われてきた. それらは Web ページを入力とし, ページレイアウトを解析し, ページ内における抽出したい情報の位置の把握, すなわち抽出ルールの導出を自動で行う必要がある. その際, 事前に教師データとしてユーザにトレーニングサンプルを入力されるものも提案されてきた. Web ラッパーの自動生成, 半自動生成とでは, 抽出されるデータの精度とユーザに対する負荷といった点で, トレードオフが生じる.

これらの問題に対し, 著者らは半自動の Web ラッパーを含む, Web ページからの情報抽出・管理を可能とする *Ducky*[6][7] システムの提案, 開発を行っている. *Ducky* は様々な構造の Web ページ, 複数のページから成る大型 Web サイトを対象にデータを抽出・統合することができるシステムである. 本論文では *Ducky* の新たなブラウザ GUI を提案し, ユーザが直感的に目的の要素を選択できるようにする. ブラウザ GUI を通してユーザが選択した要素はデータ抽出ルールに変換され, システムに登録される. これにより, ユーザがデータ抽出ルールの定義を行う負荷を軽減し, GUI における直感的な操作のみで Web データ抽出が可能となることを目的とする.

本論文の構成は以下の通りである. まず, 2 章で *Ducky* システムの概要を説明する. 3 章で今回提案するブラウザ GUI と抽出ルール生成機構の構築について説明し, 4 章で評価を行う. 5 章で関連研究について述べ, 6 章でまとめを行う.

2. システム概要

2.1 システム全体像

Ducky の全体像を図 1 に示す. ユーザは今回新たに構築したシステムのブラウザ GUI を通して, データ抽出を行

う Web ページを表示し, 目的の要素をマウスオーバーで選択することができる. Web ページの URL や, 目的の要素の位置情報などは全てデータ抽出ルールに変換され, 格納される. そのフォーマットを図 2 に示す. 目的の要素の位置情報は, CSS セレクタ (2.2.1) で表される. 抽出されるデータの形式は, xml, json, csv から選択する. またデータ抽出を行う頻度を設定し, 抽出データを自動でアップデートすることができる. これらの情報もデータ抽出ルールに格納される.

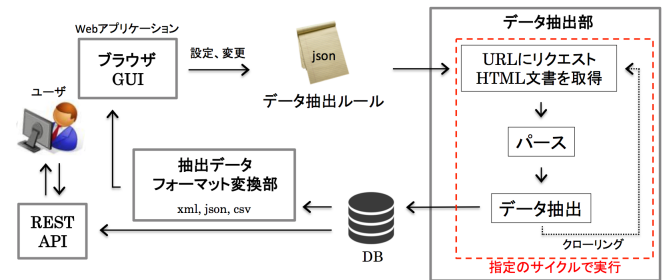


図 1 システム全体像

```
{
  "name" : "",          \\名前
  "author" : "",        \\作成者
  "frequency" : "",     \\抽出頻度
  "format" : "",        \\出力の形式
  "scraping" : [{...}]  \\データ抽出
}
```

図 2 データ抽出ルール フォーマット

2.2 データ抽出

2.2.1 CSS セレクタ

CSS セレクタは Xpath のように, マークアップ言語である HTML 文書の特定の部分を指定することができる言語構文である. たとえば図 3 の映画.com^{*1}を例にとると, 今回得たいデータが映画のタイトルやその公式 HP の URL である場合, それらは HTML の構造から全て同じ CSS セレクタ, “div.unit li > a”で指定することが出来る. このように CSS セレクタは HTML 要素をそれらの id 属性, class 属性, その他の属性などを用いて表すことができ, その記法はシンプルでわかりやすいといった特徴が挙げられる. *Ducky* において CSS セレクタを用いる理由として, 以下に 2 つの背景を挙げる.

CSS の発展によって Web ページやサイトのレイアウトは以前よりも複雑化しており, この傾向は一般的に見受けられる. 特に近年では, Web デザイナーは DB から引き出

^{*1} <http://eiga.com/link/>

したデータそれぞれに対して HTML タグの id 属性や class 属性を付与し、特定の意味を持たせる傾向にある。例えば同じカテゴリに属するデータは同じ class 属性が付与されるというようなものである。このため、そのようなデータは同じセレクトで指定することができる。先の例に挙げた映画.com においても、映画のタイトルを内包する div タグには unit というクラス属性が付与されているといったことが見受けられる。またフォント、色、視覚的パターンといったスタイルの指定は CSS セレクトを用いて CSS において定義され、ユーザがブラウジングしやすいように Web ページやサイトはデザインされる。

また近年、Web 開発者はソースコードの簡潔さや可読性向上のため、しばしば JavaScript ライブラリを用いる。中でも最も有名なものはドキュメントオブジェクトモデル (DOM) 操作ができる jQuery である。W3Techs^{*2}によると、2015 年 8 月時点で jQuery を利用している Web サイトは全体の 65.5 % と示されている。jQuery では要素の指定に CSS セレクトを用いることから、多くの Web 開発者は CSS セレクトに精通していると言える。

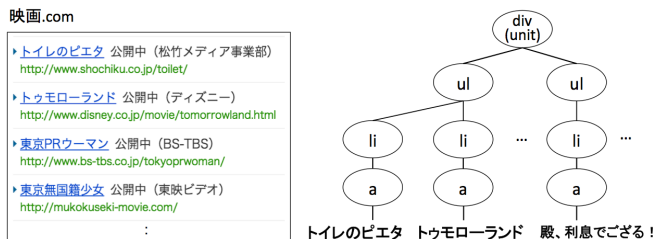


図 3 HTML 構造

2.2.2 データ抽出ルール

データ抽出に関する設定は、データ抽出ルールの *scraping* フィールドに定義される。そのフォーマットと定義されるフィールドを図 4 と表 1 に示す。ブラウザ GUI を通してユーザによって選択された Web ページ上の目的の要素の位置情報は、CSS セレクトとして *selector* フィールドに格納される。

scraping フィールドで記述されたパラメータを元に、データ抽出部においてデータの抽出を行う。処理の流れは次のようになる。

(1) “url”, “selector” の処理

指定された URL にリクエストを送り、HTML 文書を取得。指定された CSS セレクトを用いて取得した HTML 文書をパース。CSS セレクトで指定された要素を取得。

^{*2} Web 上で使われているテクノロジーのシェアや種類を調査・報告を行っている。
http://w3techs.com/

```
"scraping" : [{
  "url" : "...",
  "selector" : "...",
  "data" : [{
    "field" : "...",
    "attr" : "...",
    "find" : "...",
    "remove" : ["...", "...", ...],
    "replace" : ["...", "..."], ...
  ]
},
"next" : {...}
}]
```

図 4 データ抽出に関するルール定義

(2) “data” の処理

抽出した要素のテキストノードを抽出。“field”で抽出されたデータの項目名を定義。“attr”, “find”が記述されている場合、抽出した要素の属性もしくは子要素のテキストノードを取得。“remove”や“replace”が記述されている場合、取得したデータに対して指定の処理を行う。

(3) “next” フィールドの有無

“next フィールド”が記述されている場合、取得した URL を “url” フィールドの値として処理 (1) に戻る。記述されていない場合、取得したデータを DB へ渡す。“next” フィールドは同一テンプレートで生成されている遷移先 Web ページ群に対し、それぞれへリクエストを送り、情報抽出・統合を行う。(3.2.3 参照)

3. ブラウザ GUI

ブラウザ GUI を用いることで、ユーザはデータ抽出を直感的に行うことが出来る。ユーザがシステムに目的の Web ページの URL を入力すると、システムはその URL へリクエストを送り、HTML 文書を受け取る。そしてそのページが GUI に表示される。ユーザは目的のデータをマウスオーバーやクリックといった普段のブラウジングと同じ動作で選択することができる。ユーザがブラウザ GUI 上で行った動作は全てデータ抽出ルール化され (2.2.2), システムに登録される。

以下、ブラウザ GUI の外部仕様と内部仕様を、Ameba 芸能人・有名人ブログ^{*3}における具体的なデータ抽出例を用いて説明する (図 5)。50 音順のリンクがリストアップされたページ^{*4}をスタートとし、遷移先のページには芸能人・有名人のブログへのリンクが更にリストアップされて

^{*3} http://official.ameba.jp/

^{*4} http://official.ameba.jp/genrekana/kanatop.html

表 1 フィールド一覧

フィールド名	型	概要	
scraping	array	データを抽出するのに必要な以下のフィールドを定義	
url	string	起点となる URL を定義	
selector	string	現在のページから抽出したい要素の CSS セレクタを定義	
data	string	抽出されるデータをどのようなオプション (以下) で抽出するかを定義	
field	array	抽出データの項目名を定義	
attr	string	selector で指定したタグの属性を指定	
find	string	selector で抽出したタグの子要素を CSS セレクタで指定	
remove	array	“blank” 空白削除 “parentheses” 括弧と括弧内の文字列の削除 string 記述された正規表現, 文字列を削除	
replace	array	抽出されたデータの文字列置換	
next	object	起点のページからの遷移がある場合に使用	

いる。このサイトから芸能人の名前とそのブログのページへの URL を抽出対象とし、それぞれ “name”, “url” という項目名を付与するとする。最終的に生成されるデータ抽出ルールは図 6 に示す。

```
"scraping" : [{
  "url" : "http://official.ameba.jp/genrekana/kanatop.html",
  "selector" : "div.syllabaryMdl > table > tbody > tr > td > a",
  "next" : {
    "selector" : "dl.clr > dt > a",
    "data" : [{
      "field" : "name",
      "attr" : "href"
    }, {
      "field" : "url",
      "attr" : "alt",
      "find" : "img"
    }]
  }
}]
```

図 6 Ameba 芸能人・有名人ブログからのデータ抽出ルール

3.1 外部仕様

3.1.1 要素の選択

ユーザはまずスタートのページの URL を GUI に与える。表示されたページ上で任意の要素へマウスオーバーすると、シアンの色で色付けられる。今回の場合、50 音順のリンクへマウスオーバーすると、それらが色付けられる。これらのリンクは全て同様のスタイルが適用されており、同様の CSS セレクタで指定することができることから、ユーザへの視覚的フィードバックとしてこのように返される。クリックの動作によって、マウスオーバーの要素を指定することができる。

3.1.2 抽出対象要素をポップアップから選択

クリックでの要素の指定の後、指定の要素がテキストノード以外にも情報を保持していた場合、それらはポップ

アップ上に表示される (図 5 右図)。得たい項目にはチェックボックスに印を付けることで、抽出対象として選択することができる。

3.1.3 ページ遷移

指定された要素に URL が含まれていた場合、ツールバー上の矢印ボタンをクリックすることでページ遷移が可能となる (図 5 中央の図)。

3.2 内部仕様

3.2.1 CSS セレクタの生成

システムは GUI 上でのユーザのマウスオーバーの動作を監視し、マウスオーバーされているノードの CSS セレクタを生成する。CSS セレクタは現在マウスオーバーされているノードから、body ノードまでさかのぼって生成される。この際、それぞれのノードの class 属性も同時に取得する。生成された CSS セレクタで指定することができる全てのノードは、ユーザへのフィードバックとしてシアンの色で色付ける。ユーザのクリックの動作によって、ノードの指定がなされる。今回の例において、50 音順のリンクをユーザがクリックすることによって生成された CSS セレクタは、図 6 の “div.syllabaryMdl > table > tbody > tr > td > a” のようになる (一部抜粋)。

3.2.2 メタ情報・周辺要素の可視化

ユーザが指定した要素のテキストノードが抽出対象となるが、alt 属性などのメタ情報が存在した場合、それらもポップアップで表示される仕様となっている (図 5 右)。また近年の Web ページでは画像は画像でも、リンク付きの画像であることも多い。その HTML 構造は図 7 のように、a タグが img タグを内包する形となっている。そのような要素をユーザがクリックした場合、親ノードである a タグの属性も抽出対象として含み、ポップアップに表示する。つまり図 5 において田中将大の画像がユーザによっ

②リンクが選択された状態で矢印ボタンを押すことで、ページ遷移



①マウスオーバーすると、同様のパスに存在するものも色付けされる
クリックすることで要素を選択

抽出できる情報はポップアップで表示される

図 5 ブラウザ GUI を用いたデータ抽出例

てクリックされた場合、その HTML 構造は図 7 のようになっているため、img タグがもつ src 属性と alt 属性の値、親ノードである a タグの href 属性の値がポップアップに表示される。

ユーザがポップアップのチェックボックスで選択したものは、抽出対象として後にデータ抽出ルールに変換される。

```
<dl class="clr">
  <dt>
    <a href="http://ameblo.jp/tanaka-masahiro/">
      
    </a>
  </dt>
```

図 7 リンク付き画像の HTML 構造例

3.2.3 ページ遷移

ツールバー上の矢印ボタンがユーザによってクリックされた場合、選択されている要素の href 属性の値を取得し、そのリンク先へページ遷移を行う。これはデータ抽出ルールにおいて、“next”フィールドを用いて表現される(図 6)。図 6 において、3 行目に存在する“selector”フィールドの値“div.syllabaryMdl > table > tbody > tr > td > a”は図 5 における 50 音順のリンクの位置を示す CSS セレクタである。ここで選択されたのは a タグであり、href 属性を持つ。その値である URL が次の“next”フィールドにおける“url”フィールドの値として用いられる(2.2.2)。つまり“あ”から“わ”までの URL 全てにリクエストを送り、その遷移先のページにおいて芸能人の名前とそのブログの URL を取得するといった処理を行う。このように、“next”フィールドは遷移先の Web ページが同一のテンプレートで生成されている Web ページに対して、それらの情報を抽出・統合することを可能にする。

4. 評価

4.1 評価方法

今回提案したブラウザ GUI の有用性を評価するために 2 つの実験を行った。1 つ目の実験では、30 の Web サイトを対象にブラウザ GUI を用いてデータ抽出を行った。2 つ目の実験では、実験対象となる Web サイトを実験 1 の結果から系統別に 2 つ選定し、それぞれのサイトに対してブラウザ GUI を用いずにデータ抽出ルールを手書きで作成してもらう場合と、ブラウザ GUI を用いてデータ抽出を行う場合に分けてユーザによる評価実験を行った。なお、再現率と適合率は以下のように定義する。

$$\text{再現率} = \frac{\text{期待通り取得できたデータの総数}}{\text{取得したいデータの総数}} \times 100 \quad (1)$$

$$\text{適合率} = \frac{\text{期待通り取得できたデータの総数}}{\text{実際に取得されたデータの総数}} \times 100 \quad (2)$$

4.2 実験 1

4.2.1 結果および考察

今回対象とした 30 の Web サイトのうち、23 の Web サイトからのデータ抽出に関するデータを表 2 に示す。表 2 に記載したサイトのうち、SONY 商品カテゴリー一覧*5を除き、再現率と適合率が 100%のデータを得ることが出来た。以下、SONY 商品カテゴリー一覧に関して考察を行う。

SONY 商品カテゴリー一覧におけるデータ抽出の流れは次のようになる。商品カテゴリー一覧から計 52 のカテゴリーページへのリンクをクリックし、ページ遷移する。

*5 http://www.sony.jp/products_menu.html

遷移先のページ*6において“すべての商品を見る”というリンクをクリックし、更に遷移先のページで“商品情報”をクリック。その商品の情報を抽出するといったものであった。この動作が適用できるのは遷移先が同一テンプレートとなっているようなページ群に対してである。しかし実際にデータ抽出を行うと、52のうち36の遷移先ページはテンプレートが異なることがわかった。それらは更に同じ挙動を示す4つのカテゴリに分類することができるため、それぞれに対してルールを定義し、データを統合する必要がある。表2において、SONY商品カテゴリー一覧以外で“起点ページからの遷移先ページ数”が記載されているものは、遷移先のページが同一テンプレートで生成されていることが確認することが出来たため、今回期待したようなデータの抽出を行うことが出来た。しかしSONY商品カテゴリー一覧において今回上手くデータ抽出を行うことができなかった36の遷移先をリストアップするだけでもユーザにとって労力となることが予想されるため、今後この点に関しては対応策を考えていきたい。

表に記載しなかった残り7つのサイト*7に関しては、ページが読み込まれてからJavascriptを用いてデータを取得し、レンダリングするような動的に構造が変化するWebページであった。現在、DuckyではそのようなWebページへの対応ができておらず、今回のブラウザGUIでも対応することができなかった。近年のWebサイトはこのような動的コンテンツも多いことから、今後対応していくことが必要となると考える。

4.3 実験2

実験対象となるWebサイトは、実験1の結果を受け、表2のホームページ名に*の付いている2つを選定した。選定理由は、映画.com(以下、サイト1と呼ぶ)は1番シンプルな見開きページからのデータ抽出であること、Ameba芸能人・有名人ブログ(以下、サイト2)はページ遷移を伴う階層構造型のWebサイトであることである。同一のWebサイトに対して、ブラウザGUIを用いずにデータ抽出ルールを手書きで作成してもらう場合(以下、サイトA)と、ブラウザGUIを用いてデータ抽出を行う場合(以下、サイトB)に分けて実験を行った。それぞれのデータ抽出に至る前までの過程にかかった平均時間と平均タイピング数(クリック数を含む)、抽出されたデータの適合率の平均値に関して評価を行った。なおブラウザGUIを用いる場合において、スタートとなるURLが入力された直後やクロール時のWebページの読み込みにかかった時間は含まないものとする。実験はHTMLやCSS、Javascriptを記述した経験がある被験者(以下、経験者)と、HTML

やCSSを記述した経験のない被験者(以下、未経験者)に3名ずつ、計6名の協力のもとに行った。

実験の手順としては、まずブラウザGUIを用いてデータ抽出を行ってもらい、その後ブラウザGUIを用いずにデータ抽出ルールを手書きで作成してもらうこととした。特に未経験者に関してはデータ抽出ルールの手書きでの記述にあたり、事前にCSSセレクトの記述方法のレクチャーを行った。なお実験において、Google ChromeなどのWebブラウザで見られる、CSSセレクトを取得することが出来る機能は用いないこととした。

4.3.1 結果および考察

ブラウザGUIを用いない場合と用いる場合の、それぞれのデータ抽出に至る前までの過程にかかった時間とタイピング数の比較を図8に示す。図8から、ブラウザGUIを用いずにデータ抽出ルールをユーザに手書きしてもらう方が、ブラウザGUIを用いた場合に比べ、時間の面でもタイピング数の面でもコストが高いことが示された。また経験者と未経験者のブラウザGUIを用いた場合のデータ抽出に関して、時間においてもタイピング数においてもあまり差がつかない結果となった。HTMLに関する知識の有無に関わらずこのような結果が得られたのは、普段のブラウジングに近い動作でデータ抽出が可能なブラウザGUIの効果が存分に発揮されたと考えられる。

また表3に抽出されたデータの再現率と適合率の変遷を示す。表3の抽出されたデータの再現率と適合率の平均の変遷は、2つのWebページやサイトからのデータ抽出を経験者・未経験者の両者を対象に行い、算出した数値である。横軸の2項目はそれぞれ左が再現率、右が適合率であり、縦軸はデータ抽出を行った回数を示す。再現率が100%となった被験者は、次の回においては平均値の算出対象外とした。GUIを用いない場合の結果は、未経験者のユーザの知識不足もあり、CSSセレクトの記述が困難であったことから、再現率と適合率が100%に至るまでにかかる回数が多くなった。一方でGUIを用いた場合、どちらのサイトに対しても2回目までには経験者・未経験者ともに再現率・適合率100%のデータ抽出が行えたことがわかる。特にサイト2のように、取得データ数が1万件を超えるようなデータの抽出をクリックのみで行えるのは大きな利点であると考えられる。

表3 再現率と適合率の変遷

	サイト1				サイト2			
	A		B		A		B	
1回目	66.6	64.5	100	100	45	41.6	100	83.3
2回目	100	92.3	-	-	68.5	62.1	100	100
3回目	100	93	-	-	100	80.1	-	-
4回目	100	100	-	-	100	90.2	-	-
5回目	-	-	-	-	100	100	-	-

*6 例えば1つ目の遷移先は <http://www.sony.jp/bravia/>

*7 <http://www.pokemon.jp/zukan/> など

表 2 対象となる Web サイトからのデータ抽出に関するデータ

ホームページ名	起点ページ数	ページ遷移 (回)	起点ページからの遷移先ページ数	空白、括弧削除	正規表現	項目数	取得データ数
映画.com*	1	なし	-	-	-	2	549
FC Barcelona	1	なし	-	-	-	3	26
EXILE 公式 HP	1	なし	-	-	-	3	14
慶應義塾豆百科	1	なし	-	-	-	2	100
SKE48 公式 HP	1	なし	-	-	-	4	71
文部科学省 大学公式 HP リンク集	4	なし	-	○	-	2	1136
金融庁 リンク集	1	なし	-	○	-	2	94
中日ドラゴンズ	1	なし	-	○	-	4	73
広島東洋カープ	1	あり (1)	6	○	-	2	83
NMB48 公式 HP	1	なし	-	○	-	3	65
Ameba 芸能人・有名人ブログ*	1	あり (1)	44	○	-	3	11774
乃木坂 46 公式 HP	1	なし	-	○	-	3	32
SAMURAI JAPAN	1	なし	-	○	-	6	12
横浜 DeNA ベイスターズ	5	なし	-	○	-	4	90
読売ジャイアンツ	1	なし	-	○	-	5	104
阪神タイガース	2	なし	-	○	○	3	92
ソフトバンクホークス	1	なし	-	○	○	3	115
SAMURAI BLUE	1	なし	-	○	○	5	51
日本図書館協会 図書館公式 HP リンク集	1	あり	-	○	○	2	1634
楽天イーグルス	1	あり (1)	5	○	○	6	170
上場企業一覧リンク集「日本企業」	21	なし	-	○	○	2	3551
日本ハムファイターズ	1	あり (1)	5	○	○	6	85
SONY 商品カテゴリー一覧	1	あり (3)	52	-	-	3	238

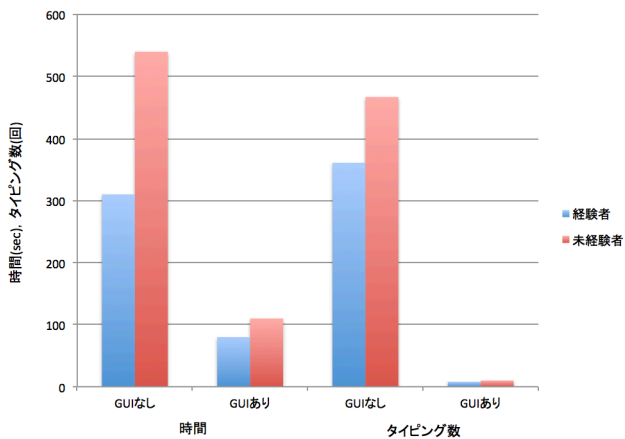


図 8 経験者、未経験者ごとのデータ抽出に至る前までの過程にかかった時間とタイピング数の比較

5. 関連研究

今日まで、Web から情報を取得する研究は数多くなされてきた [12]. その目的は様々であり、Web ドキュメントからメインコンテンツのみを認識して抽出する技術であったり、Web ラッパーを利用し、データ抽出をする技術などもある。抽出した情報は関係データベースのレコードや XML データなど必要な形式に変換され、新たなサービスなどで二次利用のために用いられる事が多い。提案システムにはこの Web ラッパーが含まれていることから、関連する技術を紹介する。ラッパーの作成には、二通りの方法がある。1 つは半自動 (semi-automatic)、2 つ目は自動 (automatic) な方法である。半自動によるラッパーの作成では、ユーザがルールを事前に指定することで抽出を行う。Zhang らが提案したラッパー [13] や Adelberg が提案した NoDoSE [1] ではルール定義に設定ファイルを用いることで実現する。設定ファイルは XML などのフォー

マットに、対象 URL や取得したい HTML 要素のパスなどを記述する。定義後はシステムが自動でデータ抽出を行う。ユーザはプログラミングを必要としないことが特徴である。XPath [4][11] では、XPath を拡張した独自の構文を開発し、データの抽出を行う。また、DeIXTo [8] では、ユーザによるルール定義を支援するために、GUI を用いた抽出システムを開発した。これらはクリックなどで要素の指定を行なうことができる。しかし、以上のような半自動のラッパー生成の手法では、抽出対象となるページそれぞれについてルール定義を行わなければならない、ユーザにかかる負担が高くなる。そこでユーザのルール定義におけるコストを削減するために、ラッパーの自動生成を行なう研究も多くなされてきた。

ラッパーの自動生成で最も有名で古いものは、Kushmerick[9] によるラッパーである。Kushmerick はトレーニングサンプルを与えて抽出すべきコンテンツの前後に現われる文字列を学習するラッパー帰納問題による方法で、ラッパーの自動生成を行った。つまりこれは事前に教師データを必要とする。一方、Chang ら [3] は教師データを用いない方法でラッパーの自動生成を行った。Chang らが提案した IEPAD は、入力 HTML の文字列の繰り返し、つまり規則性を探索とするものである。IEPAD は HTML をそのまま入力として利用できることから、完全自動のラッパーであると言える。しかし対象ページを単一レコードを含むページに限定しており、抽出不能なページに対しては何ら適用できないという問題点が存在する。

このように、これまでラッパーの自動生成に関する研究がされてきたが、これらのラッパーの抽出精度はいずれも完璧ではなかったり、現在の HTML には対応できないものや、IEPAD のように扱えないドキュメントが存在するなど、課題が残されていると言える。また、情報抽出の分

野全体として見たとき、取得したいデータは単一の URL に存在しているとは限らず、階層構造となっている Web サイトにまたがっていたり、別々の Web ページ上に点々と存在していることも考えられる。そこで、文献 [2] や [5] では GUI を用いて様々なページからデータを取得し統合可能な抽出システムを開発した。また、[10] ではクロールしデータ統合するためのページ URL を収集するアルゴリズムを提案している。OXPath[4] では独自の構文によって、階層構造の Web ページから構造化データを抽出し統合することができる。しかし OXPath において複雑な構造に対応する場合、非常に複雑なルール定義になってしまい、ユーザにとってルール定義やメンテナンスが困難である。

このようにさまざまな方法で Web ドキュメントからデータを取得する方法が研究されてきたが、自動とそうでない半自動の方法とではコストと精度がトレードオフの関係になり、依然課題として残っている問題点が数多く残されている。特に対応できる Web サイトの構造、複数ページからのデータ統合などの問題点を網羅的に解決する、実際に利用できるシステムが少ないといった問題点が大きいと考える。

6. まとめ

本論文では *Ducky* の新たなブラウザ GUI と抽出ルール生成機構を提案した。評価実験により、ユーザがデータ抽出ルールの定義を行う負荷を軽減し、GUI における直感的な操作のみで Web データ抽出が可能となることが示された。今後の課題としては 2 つ挙げられる。1 つ目はブラウザ GUI における CSS セレクタ生成アルゴリズムの改善、2 つ目はブラウザ GUI としてもシステム自体としても、近年みられる Web ページにおける動的な構造の変化に対応すべきであるということである。これらを通し、抽出可能な Web ページの構造を増やしていく必要があると考える。

また将来的には、ユーザによって抽出されたデータの共有をシステムを利用するユーザ間で行うことで、半自動ラッパーの Web ページごとにルール定義を行わなければならないという欠点を補っていきたいと考える。

参考文献

- [1] Brad Adelberg. NoDoSE - a tool for semi-automatically extracting structured and semistructured data from text documents. *SIGMOD Rec.*, 27(2):283–294, June 1998.
- [2] Sudhir Agarwal and Michael Genesereth. Extraction and integration of web data by end-users. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management, CIKM '13*, pages 2405–2410, New York, NY, USA, 2013. ACM.
- [3] Chia-Hui Chang and Shao-Chen Lui. Iepad: Information extraction based on pattern discovery. In *Proceedings of*

- the 10th International Conference on World Wide Web, WWW '01*, pages 681–688, New York, NY, USA, 2001. ACM.
- [4] Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart, and Andrew Sellers. Oxpath: A language for scalable data extraction, automation, and crawling on the deep web. *The VLDB Journal*, 22(1):47–72, February 2013.
- [5] Matthias Geel, Timothy Church, and Moira C. Norrie. Sift: An end-user tool for gathering web content on the go. In *Proceedings of the 2012 ACM Symposium on Document Engineering, DocEng '12*, pages 181–190, New York, NY, USA, 2012. ACM.
- [6] Kei Kanaoka, Yotaro Fujii, and Motomichi Toyama. Ducky: A data extraction system for various structured web documents. In *Proceedings of the 18th International Database Engineering & Applications Symposium, IDEAS '14*, pages 342–347, New York, NY, USA, 2014. ACM.
- [7] Kei Kanaoka and Motomichi Toyama. Effective web data extraction with ducky. In *Proceedings of the 19th International Database Engineering & Applications Symposium, IDEAS '15*, pages 212–213, New York, NY, USA, 2014. ACM.
- [8] Fotios Kokkoras, Konstantinos Ntonas, and Nick Bassiliades. Deixto: A web data extraction suite. In *Proceedings of the 6th Balkan Conference in Informatics, BCI '13*, pages 9–12, New York, NY, USA, 2013. ACM.
- [9] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artif. Intell.*, 118(1-2):15–68, April 2000.
- [10] Tiezheng Nie, Zhenhua Wang, Yue Kou, and Rui Zhang. Crawling result pages for data extraction based on url classification. In *Proceedings of the 2010 Seventh Web Information Systems and Applications Conference, WISA '10*, pages 79–84, Washington, DC, USA, 2010. IEEE Computer Society.
- [11] Andrew Jon Sellers, Tim Furche, Georg Gottlob, Giovanni Grasso, and Christian Schallhart. Oxpath: Little language, little memory, great value. In *Proceedings of the 20th International Conference Companion on World Wide Web, WWW '11*, pages 261–264, New York, NY, USA, 2011. ACM.
- [12] H.A. Sleiman and R. Corchuelo. A survey on region extractors from web documents. *Knowledge and Data Engineering, IEEE Transactions on*, 25(9):1960–1981, September 2013.
- [13] Suzhi Zhang and Peizhong Shi. An efficient wrapper for web data extraction and its application. In *Computer Science Education, 2009. ICCSE '09. 4th International Conference on*, pages 1245–1250, July 2009.