Web ページ構成の共通性を利用した連載コンテンツの抽出統合方法の研究

松岡　巨樹† 　中田　佳宏‡ 　井上　潮‡

†‡東京電機大学工学部 〒101-8457 東京都千代田区神田錦町 2-2

E-mail: †{05kc595,05kc081}@ed.cck.dendai.ac.jp, ‡inoue@c.dendai.ac.jp

あらまし インターネットの普及により，Web 上に様々なコンテンツが掲載されるようになった．中でも，連載記事はよく目にするものであり，また利用者が多いものである．しかし，連載記事は複数のページに分割されていたり，記事内に広告などの記事により，そのコンテンツ本来の可用性を妨げている．

本研究では，Web ページ構造の共通性を利用することにより，分割されたページを 1 つのページに統合すると共に，余分な広告情報を自動的に削除する方法を提案する．

キーワード Web ページ，連載記事，情報統合，不要情報除去

Extraction and Integration of Serial Contents using Common Features of Web Pages

Naoki MATSUOKA† 　Yoshihiro NAKATA‡ 　and 　Ushio INOUE‡

†‡Department of Information and Communication Engineering ,Tokyo Denki University

2-2 Kanda-Nishikicho,Chiyoda-ku, Tokyo, 101-8457 Japan

E-mail: †{05kc595,05kc081}@ed.cck.dendai.ac.jp, ‡inoue@c.dendai.ac.jp

Keyword Web page，Serial contens，Information integration，Spam removal

1. はじめに

近年，インターネットの普及により，ニュースや解説，報告などの記事が Web ページにて配信されることが非常に多くなった．これらの記事の中でも特に連載記事には有用な情報が多く，内容が有効な期間も他のコンテンツに比べて比較的長いため，利用価値が非常に大きい．

しかし，Web 上の連載記事は必ずしも利用者にとって利用し易くはない．例えば，記事が複数のページに細かく分割されて掲載されていたり，記事の内容とは無関係な情報が，記事本体のまわり，またはその内部に含まれていたりすることが一般的だからである．

このため，連載記事を閲覧したり，保存したりする場合に，かなりの手間暇がかかり，利便性が損なわれていると考えられる．

この問題を解決するため，我々は Web ページの構造と特定の文字列を手がかかりとして，複数の Web ページから連載記事を自動的に収集し，記事とは無関係な情報を削除し，読み易い一つの Web ページにまとめるシステムを提案する．

2. 関連技術

Autopager[1]は Firefox アドオンである．連続したページを一つのページであるかのように閲覧できる．しかし，これはあらかじめ登録されたサイトのみ使用可能であり，それ以外のサイトで使用するためには，サイトを登録しなければならず，利便性は低い．

ScrapBook[2]は Firefox アドオンである．その目的は Web ページをローカルに保存することにあり，保存前にページの余分な部分を消去するなどの編集処理が可能である．しかし，この編集処理は手動で行わなくてはならないため，この点において利便性が高いとはいえない．

3. 研究目標

本研究の目的は，連載記事を自動的に収集し，余剰情報を削除し，ひとつの記事に統合するシステムを実現することである．ここで，連載記事とは Web 上で複数のページにわたって存在する記事で，すべてのページをもってひとつのまとまった内容になっているもの

を指す．また，余剰情報とは本来の記事とは直接関係のない広告リンクなどである．

このシステムでは，次の項目を満足すること目標とする．

- 連載記事の URL をひとつ利用者が与えれば，その記事が掲載されているすべての Web ページを自動的に収集できる．
- Web ページ中で記事本文の一部分の文字列を利用者が指定すれば，記事本文と余剰情報を自動的に判別し，余剰情報のみを削除できる．
- 処理が完了した Web ページは標準的なブラウザで閲覧可能な形式になるようにする．
- 元の Web ページと違和感が生じないようにするため，余剰情報を削除してもページレイアウトをできるだけ保つようにする．
-

4. 提案手法

本研究では，上記目標の実現にあたって，連載記事ページの URL および HTML 構造の共通性を利用する．

連載記事の URL のほとんどは図 1 のようなすべての URL の終わりが数字，アルファベットの連番構成，あるいは，図 2 の様な 1 ページ目が「index.html」などの省略可な URL，残りの URL の終わりが図 2 と同様な連番の構成となっている．そして，それらはいずれも同一ディレクトリに存在している．また，ほとんどの連載記事ページは，そのページ内に前記のようなその記事のほかのページへのリンクが存在する．このリンクは，そのほとんどが相対パスであるが，絶対パスに置きなおすと，比較可能になる．

```
http://..... .com/.....01.html
http://..... .com/.....02.html
http://..... .com/.....03.html
```

図 1.一般的な記事の URL の規則性（パターン 1）

```
http://..... .com/
http://..... .com/.....01.html
http://..... .com/.....02.html
```

図 2.一般的な記事の URL の規則性（パターン 2）

このような規則性を持っている場合，その記事の他のページへのリンクであると判断することができる．

HTML 構造においては，ほとんどの連載記事ページにおいて記事の本体は，ID 属性をもつタグに囲われていて，また記事本体の直上の ID 属性をもつタグが記事本体全体を囲っている場合がほとんどである．図 2 に例を示す．

```
<body>
<div id=...>
 <div id=...>
 <p>
 <br>
 </p>
 </div>
 <div id=...>
  <div id=...>
  本体
  </div>
 </div>
</div>
</body>
```

図 2.記事ページの HTML 構造例

この ID 属性をもつタグで囲まれている部分を抜き出せば，記事本体部分を正確に抽出できる．また，同じ記事サイトであるならば，CSS と HTML 構造はほとんど全て共通であるので，同じ階層にあるタグによりすべての連載記事ページからの抽出が可能である．

また，CSS が共通であるということは，そのサイト内の連載記事すべてに一つの CSS が適用可能であるということである．このため，レイアウトを保つため CSS を収集する際にはこの一つの CSS のみでよい．

前記手法にて必要部分を抽出したものを統合し，この CSS を適用すれば，元のレイアウトを保った統合された記事ページが出力される．この手法にて生成された統合記事文書は通常の HTML であるため，標準的な Web ブラウザで閲覧可能である．

次に大まかなシステムの流れを説明する．

ユーザが URL と本文文字列を与える．まずこれらをもとにページ取得，余剰情報削除がおこなわれ，その後ページを修正し，ページを一つに統合する．また，同時に CSS を収集し，統合を行う．処理が終了したページは，サーバに保存される．処理の流れを図 3 に示す．

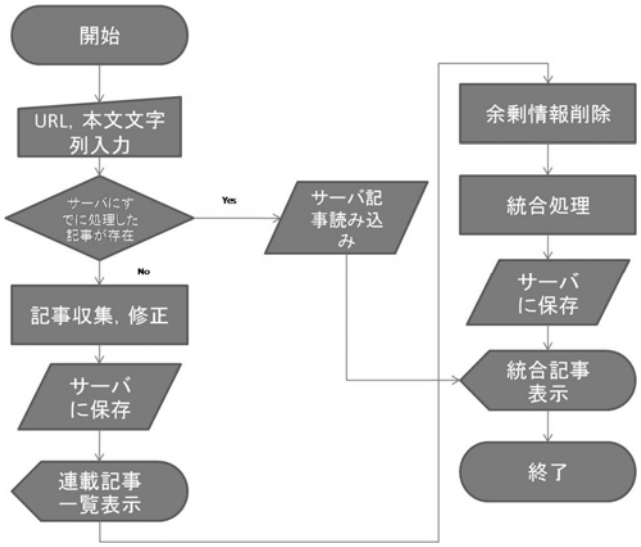

図 3. システムのフローチャート

また以下に具体的な手法を示す.

4.1. 記事 URL, ページの収集

連載記事の共通性を利用し，図 4 に示す流れで同一記事 URL を収集する.

まず入力した URL のソース内の href 属性よりリンクを取得し，これを絶対パスに変換する．次に連載記事は同一のフォルダに存在するので,入力 URL と同一のフォルダに存在するリンクを収集する．この際，同一ページへのリンク（#がついた URL）を除外し収集する．そして連載記事をページ順に並べるため拡張子を削除し昇順に並び替える.最後に入力 URL を基準とし上下の URL を調べていきタイトルが入力 URL と一致するか，また次ページ(前ページ)のリンクが存在するかを調べ，条件を満たすものを同一連載記事の URL としている.

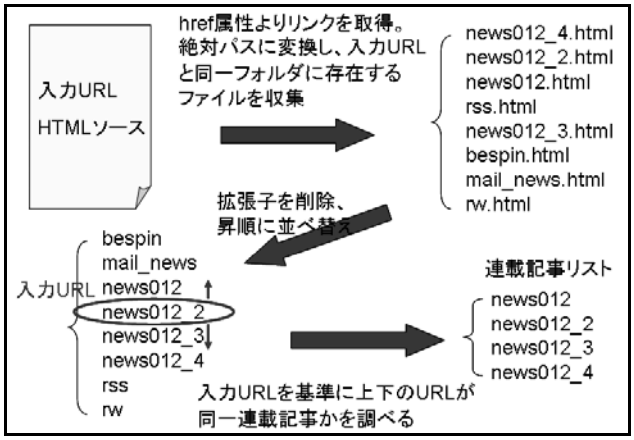

図 4.ページ収集

4.2. 記事 CSS の収集, 統合

統合した新しいページのスタイルを保つため，記事ページで定義している CSS を取得，統合する．統合の必要性があるのは，ほとんどの連載記事サイトの CSS は複数に分けて記述されている CSS をインポートして使用しているからである．CSS ファイルを読み込み，記されているすべてのインポート先のアドレスを正規表現にて抽出し,そこの記述を結合する.収集した CSS はシステムサーバに保存される.

4.3. 余剰情報の削除

削除するにあたって，余剰情報と，記事の内容などの記事本体であるか判別する必要がある.

本研究では，記事本体を判別するためにユーザに入力された記事本体の一部分を判別の指標とする．余剰情報の削除は，コンテンツ本体を含んでいると思われる ID 属性を持つタグにくくられている部分をスクレイプし，ほかの不要部分を削除という方針で行う．また，抜き出した部分のレイアウトを保つため，HTML,CSS の構造が崩れないように抜き出した必要部分をもとと同じように構成しなおす．具体的な方法は以下のとおりである.

4.3.1. <body>部分取得

HTML 文書の中でも，本体部分である<body>部分を取得する．連載記事ページにおいて<body>部分以外の部分は，変更点はほとんどないため，ヘッダや<meta>はほとんど共通している．そのため，変更部分である<body>タグのみを取得する

4.3.2. 記事一部分の文字列取得

ユーザにより入力された記事本体判別のための一部分の文字列を取得する．この文字列は記事本体の中で一意であるならば，記事本体を指定できる．なぜならば，この文字列は記事本体内にあるから，HTML 構造にて，同じ場所にあると断定できるからである．この文字列を指標として本文を括るタグの ID 属性を特定する.

4.3.3. HTML 文書操作

実際に余剰情報を削除するための操作を行う．前述のとおり，連載記事ページの特徴として，本文を括る直上の ID 属性を持つタグの部分を抜き出せば，効率よく本文を抽出できる．また，余剰情報削除後に元々のレイアウトを保つために CSS との整合性を考慮するため，抜き出した本体を必要なタグでくくる．具体的な操作は以下のとおりである.

(ア)　HTML 文書上下分割

入力された記事本体の一部の文字列を基準に取得した html 文書を上下に分割する．これにより，終了タグと対になっていない開始タグができる．この開始タグが本体を直接くくっているタグである．

(イ)　タグ取得

分割されたHTML文書の上の部分のタグを正規表現にて，すべて取得する．この時点で終了タグと対になっていない開始タグが出現するはずである．

(ウ)　開始タグ抽出

取得したタグから終了タグと対になっていない開始タグのみを抽出する．この開始タグは本体を囲う開始タグである．HTML のタグは，終了タグの直前にあるタグは開始タグである場合，同種のタグであるという性質がある．抽出方法ではこれを利用して，取得したタグを順に調べていき，開始タグであると判断された場合はすべて記憶していく．終了タグであると判断された場合，直前に記憶した開始タグを調べ，終了タグと同種のタグであったら,この開始タグを削除する．別種の開始タグであったら，この終了タグはノイズであると判断し，処理を行わない．HTML のタグの中には単独で使う単独タグがある．この単独タグは，形式だけでは開始タグと分別できないため，このタグに関しては個々に登録して判断する．単独タグであると判断されたならば，ノイズとして処理する．

(エ)　ID 抽出

取得した開始タグから ID を持つタグを上から順に抽出する．その最後に取得したタグは本体の直上の ID を持つタグである．この ID を用いて Web::Scraper[3] によって HTML 文書からの必要部分抽出を行う．

(オ)　HTML 構造整形

抽出した部分をそのまま<body>タグで囲って HTML 文書とすると，CSS との不整合を起こし，スタイルが保たれないので,タグを元と同じように構成しなおす．その後，ウで抽出した開始タグと，新たに抽出した終了タグで本体を囲う．終了タグの抽出方法は，アで分割された HTML 文書の下の部分をイ，ウと同様にタグを抽出していく．ここでは，開始タグの直後に来る終了タグは必ず同種のタグであるという HTML タグの性質を利用して,終了タグであればすべて記憶していき，開始タグと判断された場合，直後の終了タグが同種であったなら，その終了タグを記憶しない．

以上のエの処理を取得したページ分繰り返す．これは，CSS と HTML 構造がほとんどの連載サイト内では共通になっているためである．

4.4. 記事の統合

記事ページの<head>タグ内などの<body>タグ外は連載記事ページの共通性から，ほとんどが全く同じである．そのため，この部分は変更を加えない．記事本文は<body>タグの中を順に下に接続していく．統合イメージを図 5 に示す．

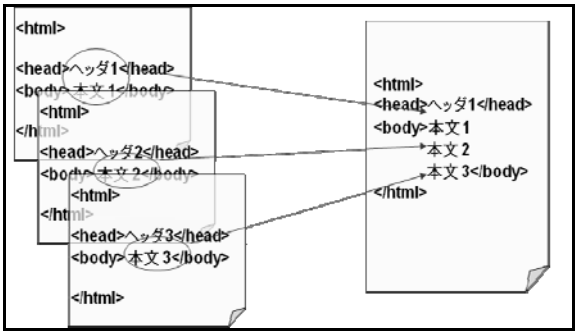

図 5.統合イメージ

この操作により，複数ページを一つにまとめる．

5. プロトタイプの実装

現在開発中のプロトタイプの構成図を図 6 に示す．

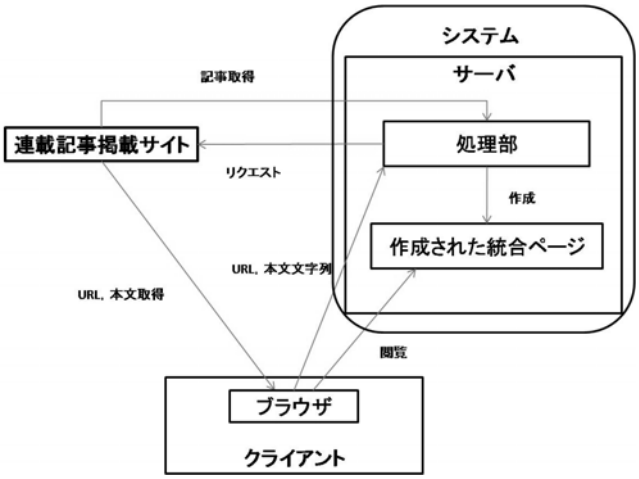

図 6.プロトタイプ構成図

ユーザはブラウザでシステムにアクセスし，初期画面にて連載記事の URL，記事の一部など必要事項を入力する．システム画面を図 7 に示す．

図 7.システム画面

これらの事項の入力後，取得結果が表示される．

その後，確認ページの show ボタンを押すと，処理作成されたページが表示される．ここまでの画面の遷移を図 8.に示す．

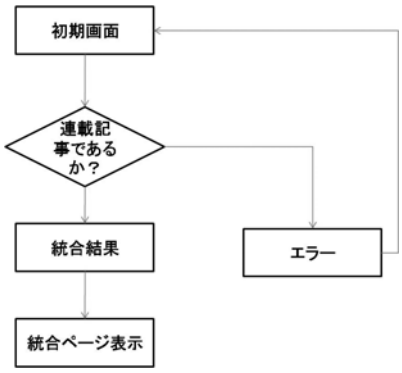

図 8.画面の遷移

統合されたページはサーバに保存される．保存される際のディレクトリ構成は保存ディレクトリを作成し，その下に記事ごとのディレクトリを作成する．記事の HTML，CSS はここに置かれる．

ディレクトリ構成図を図 9 に示す．

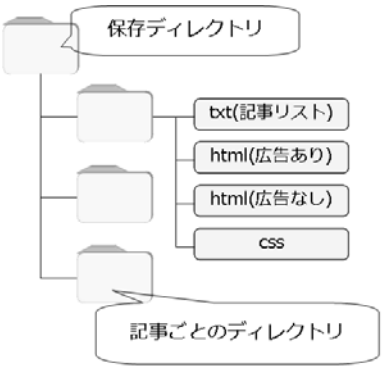

図 9.ディレクトリ構成図

すべての処理が終了すると，処理された連載記事が表示される．その例を図 10 に示す．

図 10.実行例

6. 検証

開発したシステムの部分ごとの検証を行った検証項目は以下の 4 つである.

① Web ページの自動収集精度
② 標準的なブラウザで閲覧可能な形式かどうか
③ 余剰情報削除の適用率，精度，削除された記事の再現率
④ 記事ページのレイアウト保存性

6.1. Web 連載記事の自動収集精度

作成したプロトタイプがサイト毎に正しくページを収集できているかの評価を行う．今回の検証では，50 サイトを対象に行った.

その結果，46 サイトが正しく収集でき，収集精度は 92%であることが確認できた. 円グラフを図 11 に示す.

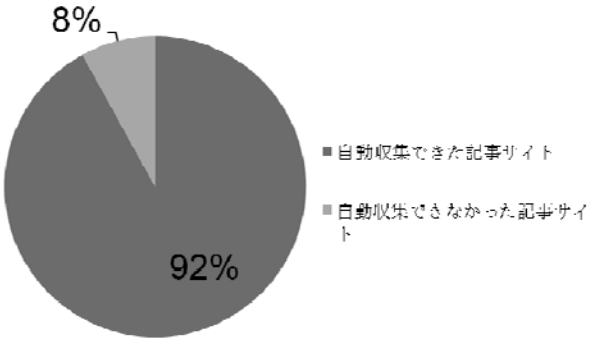

図 11.Web 連載記事の自動収集精度

6.2. 標準的なブラウザで閲覧性の評価

現在，標準的なレンダリングエンジンを搭載している Microsoft Internet Explorer，Mozila FireFox，Google Chrome の 3 種のブラウザを使用して評価を行う. 結果は表 1 のようになった.

表 1. ブラウザ閲覧可能結果

ブラウザ	閲覧可能であるか
Microsoft Internet Explorer	可
Mozilla FireFox	可
Google Chrome	可

結果として，3 種類のブラウザすべてで閲覧することができた. これは, 統合されたページは通常の HTML であるので，ウェブブラウザであるならほぼ間違いなく表示することが可能である.

6.3. 余剰情報削除の評価

余剰情報削除機能をどの程度サイトに適用できているか，また記事本文と余剰情報を正確に判別し削除しているかを評価した．それぞれ，適用可能なサイトの確率，削除精度，記事の再現率に分けて評価した.

6.3.1. 適用可能率

余剰情報削除機能の連載記事サイトへの適用可能確率を調査した. 50 の連載記事サイトを対象として検証した結果，38 サイト，76%の確率で余剰情報削除機能を適用することができた．円グラフを図 12 に示す.

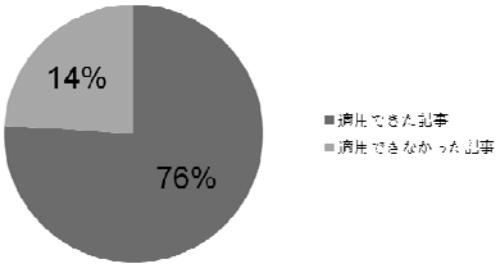

図 12.適用可能率

6.3.2. 余剰情報削除精度

余剰情報削除機能がどの程度記事内部の余剰情報を削除することが可能かを検証した. 6.3.1 項で適用できた 38 サイトを対象として調査した. 調査方法として，余剰情報の画面占有に対して，削除できた余剰情報の画面占有の割合を調べた．その結果，91%の精度で削除することができた.

6.3.3. 余剰情報を削除した記事の再現率

余剰情報を削除した記事がどれだけ元記事の余剰情報出ない部分の情報を失っていないかを調査する 6.3.1 項で適用できた 38 サイトを対象として調査し，記事本体の画面占有に対する，損失した記事本体の画面占有の割合を調べた．その結果，92%の再現率が確認できた.

また，余剰情報削除精度，余剰情報削除記事再現率の散布図を図 13 に示す.

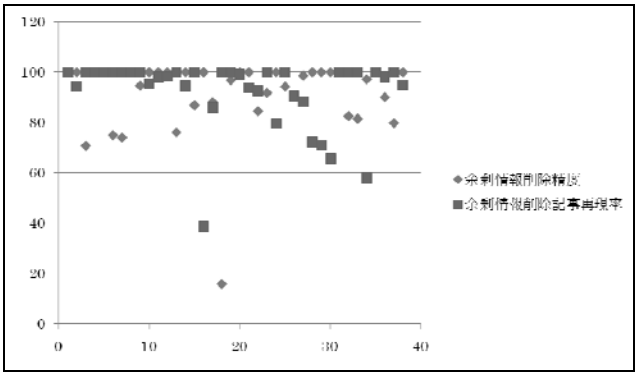

図 13.散布図

6.3.4. 記事ページのレイアウトの保存率

Web ページを閲覧し，元のページと比べてレイアウトが崩れているかを検証した．結果，38 の連載記事の中，33 の連載記事のレイアウトが保たれていることを確認した．図 14 に示す．

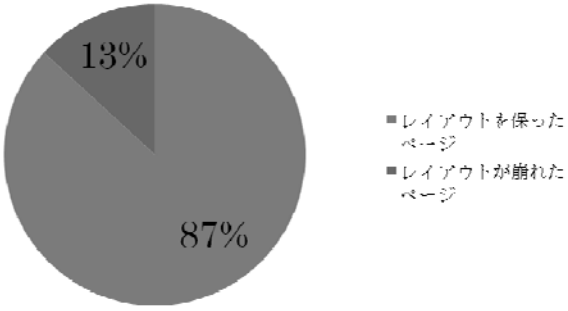

図 14.レイアウトの保存性

7. まとめ

本研究では，Web ページの構造の共通性を利用した連載記事の抽出統合のシステムを提案した．連載記事 URL 取得においては，記事の URL，タイトルの共通性を利用した収集手法を検討し，92%の自動収集精度を持つプロトタイプを開発できた．余剰情報削除においては，記事の HTML 構造の共通性を用いた余剰情報削除方法を検討，プロトタイプを実装した結果，適用率 76%，その適用できた連載記事に対する精度が 91%，削除適用した記事の再現率が 92%のシステムを実現した．記事統合においては，これらの方法を用いて作成したプロトタイプとその有効性を示した．記事統合においては，<body>タグ外の共通性を利用し，<body>タグ内のみを複数のページを一つにまとめる機能を実装した．これにより，ブラウザでの閲覧性，レイアウトを保った統合ページを作成できた．

8. 今後の課題

それぞれの機能ごとに述べる．記事 URL 自動収集機能はタイトルの比較によって記事の URL を収集しているが，一部にタイトルの文字列の切り分けができていない部分があるため，精度向上のためには修正が必要である．余剰情報削除に関しては JavaScript によるなんらかの操作をしている連載記事に適用することができなかった．今後は JavaScript の対処法を検討する．

文　献

[1] Auto pager
https://addons.mozilla.org/ja/firefox/addon/4925
[2] Scrap Book
http://amb.vis.ne.jp/mozilla/scrapbook/
[3] Web::Scraper
http://search.cpan.org/~miyagawa/Web-Scraper-0.24/lib/Web/Scraper.pm/.
[4] PHP:Tidy 関数　– Manual
http://jp.php.net/manual/ja/ref.tidy.php
[5] LWP::Simple
http://search.cpan.org/dist/libwww-perl/lib/LWP/Simple.pm