テンプレートの交叉とDOM構造の解析による情報抽出手法の提案

楠村　幸貴† 　土方　嘉徳† 　西田　正吾†

† 大阪大学大学院基礎工学研究科　〒 560–8531　豊中市待兼山町 1–3
E-mail: †{kusumura,hijikata,nishida}@nishilab.sys.es.osaka-u.ac.jp

あらまし　Web 上には大量の情報が存在しているが，これらは整形されておらず，多種多様な記述形式で記述されている．このため，これらの不均一な文書から情報を属性と属性値の形で整形して抜き出す情報抽出の研究が行われている．特に近年，その中でも教師信号を必要としないブートストラッピング・アルゴリズムを用いた情報抽出手法が注目を浴びている．しかし，これまで提案されている手法には二つの問題がある．一つ目の問題は，従来手法は前後の文字列をテンプレート化するため Web 上の表や箇条書きからは抽出できないという点である．二つ目の問題は，Web 上の文書は多様性が高いため決まったテンプレートでは微妙な差があるだけで抽出できない場合が多い点である．本研究ではこれに対し，Web ページの DOM 構造を解析するテンプレートを作成することで表や箇条書きからも抽出を行い，生成したテンプレート同士を交叉させてテンプレートを増やすことで微妙な記述の揺れに強い抽出手法を提案する．

キーワード　情報抽出，ブートストラッピング，情報統合，半構造データ

Unsupervised Information Extraction by Crossover and Structure Analysis

Yukitaka KUSUMURA†, Yoshinori HIJIKATA†, and Shogo NISHIDA†

† Graduate School of Engineering Science, Osaka University　1–3 Machikaneyama, Toyonaka, Osaka 560-8531, JAPAN
E-mail: †{kusumura,hijikata,nishida}@nishilab.sys.es.osaka-u.ac.jp

Abstract　The Web is a vast resource for information. At the same time it is extremely distributed and written in many different formats and descriptions. To integrate this information, information extraction system is needed. This paper describes an unsupervised approach for extracting information from the Web. We present an bootstrapping algorithm, which is called BootCross(Bootstrapping based crossover and structual analysis). BootCross has two unique features. First, BootCross creates the extracting templates about upper-level structure in document object model of web pages to extract information from not only sentences but also structured formats like tables and lists. Second, for adapting bootstrapping algorithm to the diversity of the Web, BootCross boosts number of templates by crossover operations of two similar templates. In this paper, we evaluate the proposed methods by experiments, showing the benefits of them.

Key words　information extraction, bootstrapping, information integration, semi-structured data

1. はじめに

Web 上には大量の半構造化文書があり，多種多様な情報が公開されている．これらはデータベースのように整形された形式で保存されていないため，計算機はこれらを正確に扱うことができない．この問題に対し，これらの文書から情報を属性と属性値という形にして取り出す情報抽出の研究が行われている [1]〜[3]．従来の情報抽出では，予め人手で作成した抽出ルールを用いる手法 [4], [5] や予め抽出位置にタグ付けされた正解付き文書（タグ付きコーパス）から機械学習により抽出ルールを作成する手法 [6]〜[8] が用いられてきた．しかし，これらの手法は抽出ルールを用意する作業やタグ付きコーパスを作成する作業が必要であり，抽出システムを新しいドメインに適用する際のコストが高いという問題があった．

この問題に対して，近年自然言語処理の分野でブートストラッピング・アルゴリズムを用いた情報抽出手法が提案されて

いる [9]〜[11]．これは抽出ルールと抽出する値を交互に繰り返し学習することにより，雪だるま的に大量の値を抽出する手法である．例えば，Web 上から映画の作品名を抽出する場合，ユーザは予め「タイタニック」などの属性値の例（シードと呼ばれる）を少数を用意する．この手法はそれを元に文書を調べ「映画『タイタニック』近日公開」といったパターンを獲得し，このパターンから属性値部分を抽出位置とみなして「映画『(抽出位置)』近日公開」というテンプレートを作成する．次にこの手法はこのテンプレートで文書集合を調べることで，別の記述「映画『マトリックス』近日公開」から新たな属性値「マトリックス」などを抽出する．さらに，得られた属性値をシードとして追加し同様のステップを繰り返すことで，少量の属性値から大量の属性値を抽出していく．

しかし，この方法を Web 上の文書に適用しようとすると，次の 2 つの問題がある．一つ目の問題は，前後の単語でテンプレートを作成すると，表や箇条書きなどの構造化された記述形式からは値を抽出できないという点である．例えば，図 1 に映画の情報を table タグを用いて記述した例を示す．この中の「セブン」という文字列を映画作品として抽出したい場合，その属性を表現した単語「作品名」は「セブン」と記述されている位置からは離れた位置に記述されている．そして「セブン」という記述の前後には表を構成する最低限のタグしか存在しないため，有効な抽出用のテンプレートを作成できない．二つ目の問題は，Web 上の文書は新聞記事などのようなプロが記述した文章に比べ多様性が高いため，事例から獲得したテンプレートだけではパターンの数が足りず，対象の記述に微小な差があるだけで抽出できないという点である．例えば「映画『(抽出位置)』公開」というテンプレートは「映画『マトリックス』発表」や「映画 $$ マトリックス $$ 公開」のように，一部が異なるだけのページに対しても抽出ができない．

本研究ではこれらの問題に対して次の解決方法を提案する．一つ目の問題に対する解決方法は，ある値からテンプレートを作成する際に前後の文字列 (HTML タグを含む) だけでなく，DOM 構造 [12] を解析し特定の位置にあるテキスト部分（例えば，表の中に記述された値に対しては，同じ列の一行目や同じ行の一列目にあるテキストなど）を取り出し，属性を表す語 (以下，属性名と呼ぶ．例えば,「作品名」や「主演」) を利用したテンプレートを作成することである．本研究ではある語に対して DOM 構造上でその語と意味的に関連する可能性が高いテキスト部分を USP:Upper-Level Semantic Text Portion と呼ぶ．本研究では，テンプレートを作成する際に属性値に対する USP を調べ，そこに属性名が含まれる場合，USP 中の単語を加えてテンプレートを作成する手法を提案する．

二つ目の問題に対する解決方法は，事例から獲得したテンプレート同士を交叉し，新しいテンプレートを生成することである．一般にテンプレートは，決まった数の属性値前後の単語またはタグで構成されるが，異なる 2 つのテンプレート中において同じ位置に存在する語が異なっていれば，それらを入れ替えた新しいテンプレートを作成する．例えば,「映画『(抽出位置)』公開」と「映画 $$(抽出位置)$$ 発表」いうテンプレートがある場合，これらの一部を入れ換え,「映画 $$(抽出位置)$$ 公開」や「映画『(抽出位置)』発表」といったテンプレートを新たに生成する．しかし，無闇にテンプレートを交叉させると誤ったテンプレートを作成してしまう可能性があるため，本研究ではテンプレートの類似度を計算し，交叉させるテンプレートを選択する手法を提案する．

本論文は，これらの解決方法を用いたブートストラッピングアルゴリズム BootCross(Bootstrapping based on crossover and structure analysis) を提案するものである．本論文では，まず 2. 章で関連研究について述べ，3. 章において BootCross アルゴリズムの処理の流れを説明する．次に 4. 章において USP を用いた抽出テンプレートの作成方法とその抽出方法を説明する．そして 5. 章において抽出テンプレートを交叉させる方法を述べ，6. 章においてテンプレートと抽出した属性値の評価を行う手法を述べる．その後，7. 章において提案手法の評価を述べ，最後に 8. 章においてまとめを述べる．

2. 関 連 研 究

本研究は，ブートストラッピングを用いた情報抽出において，USP を利用し構造化された記述からも抽出を行う方法と，交叉により記述の不均一さに強い抽出手法を提案するものである．そこで本研究の関連研究としては，ブートストラッピングを用いた情報抽出の研究，構造化された記述から情報抽出を行う研究，文書の不均一さに注目しある程度の記述の異なりを許容する抽出ルールを作成する研究が挙げられる．本節ではこれらの関連研究を順に述べる．

自然言語処理の分野では近年，ブートストラッピングによる情報抽出の研究（Riloff ら [10]，Yangarber ら [13]，Brin ら [9]，Etzioni ら [14]）が行われている．Riloff らと Yangarber らは，新聞記事から人名や地名を抽出するタスクに対してブートストラッピングを用いた．新聞記事には決まった言い回し（例：人名の後ろには「氏」という単語が書かれる．など）が多く存在するため，属性値の前後のテキストを単純にテンプレート化する手法（Yangarber らの手法 [13] や Riloff らの手法 [10]）が用いられている．彼らは精度を保つために，あるサイクルにおけるテンプレートの確信度と属性値の確信度を計算し，不確かなテンプレートと属性値を削除する方法を提案した．この確信度については本研究でも用いている．確信度の詳細については 6. 章で述べる．

また，Brin らは Web 上から書籍名とその著者を抽出するタスクに対してブートストラッピングを用いた．彼らは精度を保つために，抽出した値が正しいかどうかを判定する際にヒューリスティック（著者名は大文字で始まる二つの単語とするなど）をいくつか利用している．しかし，これらのヒューリスティックはドメインに依存するため，この手法を他のドメインに利用することはできない．Etzioni らは Web から地名を抽出するタスクや映画情報を抽出するタスクに対してブートストラッピングを用いた．彼らはより多くの値を抽出するために，ドメインに依存しない一般的な属性名と属性値の関係として「(属性名) such as (属性値)」や「(属性値) is a (属性名)」といったパ

```
< H2 > 作品一覧 < /H2 >
< TABLE >
< TR >< TD > 作品名 < /TD >< TD > 主演 < /TD >< /TR >
< TR >< TD > カクテル < /TD >< TD > トム・クルーズ < /TD >< /TR >
< TR >< TD > セブン < /TD >< TD > ブラッド・ピット < /TD >< /TR >
< /TABLE >
```

図 1　構造化された文書の例

ターンを用意しておき，これを元にテンプレートを学習する方法を用いた．しかし，この手法は予め用意されたパターン以外の記述には対応できないという問題がある．また，上記の研究はすべて，表や箇条書きなどの構造化された記述形式を対象としていない．これらの研究に対し本研究は，構造化された記述形式からも抽出を行うために，DOM 構造を用いたテンプレートを作成する．さらに，より多くの記述にマッチングできるよう，テンプレートの交叉を行う．

Web ページ中の構造化された記述に対して情報抽出を行うシステムとしては，Web ラッパー [3] がある．Web ラッパーは特定の Web サイトから情報を抽出するプログラムである．ある Web サイト内では，複数のページでも共通のレイアウトが用いられることが多く，Web ラッパーはこの特徴を利用して抽出ルールを作成する．特に，Web ラッパーの研究の中には本研究と同様 HTML ページの DOM 構造 [12] についてのルールを用いる Tree ラッパーの研究 [5], [16], [17] がある．Tree ラッパーはサイト共通のレイアウトを DOM 構造においてルートから抽出する位置へ下方向に移動するパスで表現し，抽出ルールとするものである．しかし，本研究は Web 全体を対象としており，共通のレイアウトが存在するという仮定は成り立たず，ルートからトップダウンに属性値が記述される位置をルール化することはできない．そこで本研究は属性値の付近の構造のみを用いた USP の抽出ルールを利用し，属性値の位置からボトムアップに抽出を行う．

文書の不均一さに注目し，ある程度の記述の異なりを許容する抽出ルールを作成する手法には，ベクトル空間モデル [18] を用いた手法 [11] と確率モデルを用いた手法 [19] がある．これらの手法ではテンプレートマッチングにより一致したかどうかを判定するのではなく，テンプレートと入力文の間で類似度を計算し，その閾値によって抽出を行うか否かを決定するものである．ただし，この方法だけでは，抽出数が増える分誤抽出が増える可能性も高くなる．このためこれらの研究では，抽出時に抽出対象の文字列が属性値（人名や地名）かどうかを確認する別の辞書（人名辞典や地名辞典）を利用したり，定義文を抽出するという誤抽出の少ない問題を対象としている．これらに対し本研究は，テンプレート同士を交叉し新しいテンプレートを作ることで，精度をうまく保ちつつテンプレートの不足を補う手法を提案するものである．

3. BootCross の概要

BootCross の処理の流れを以下に示す（図 2 にこの概要図を示す）．

（1） 初期シードとなる属性値の例を入力する．なおこの時，属性名も一緒に入力するものとする．この属性名は (2) と (3) の処理で用いられる．本システムの入力値の例を表 1 に示す．この表では属性を列で表現しており，各列の一行目に属性名を記述している．入力する属性名は類義語を含めて数個記述する．2 行目は属性値を表しており，これらの属性値を各属性に対し 10 個程度記述するものとする．

（2） シード内の属性値を元に検索エンジンで検索を行い，シード内の属性値を含むページを獲得する．検索エンジンから検索結果を収集するためには Google Web APIs [20] を用いる．なお，検索エンジンに送信する検索語には，属性値とそれに対応する属性名を合わせたものを用いる．

（3） シード内の属性値と属性名を用いて各ページ内を調べることでテンプレートを作成する．ページを DOM 構造に変換する際には CyberNeko HTML Parser [21] を使用する．詳細は 4. 2 節で述べる．

（4） 作成されたテンプレートを交叉し，新しいテンプレートを作成する．詳細は 5. 章で述べる．

（5） テンプレートを用いて抽出を行う．詳細は 4. 章で述べる．

（6） 抽出結果に基づき，抽出した属性値とテンプレートの確信度を調べ，属性値をシードに追加した後，処理 (2) に戻る．詳細は 6. 章で述べる．

4. USP を用いた抽出テンプレート

ある属性 f の属性値を抽出するテンプレート t_i は左側の単語列 t_i^{left}，右側の単語列 t_i^{right}，USP 中の単語列 t_i^{up} という 3 つの種類の単語列から成る．これらの単語列は次のように抽出に用いられる．(1) 入力文書を前方から調べ，t_i^{left} が一致する部分を探す．一致する部分があるならば，(2) に進む．(2)(1) で一致した部分から右方向に文書を調べていき，L 単語以内に

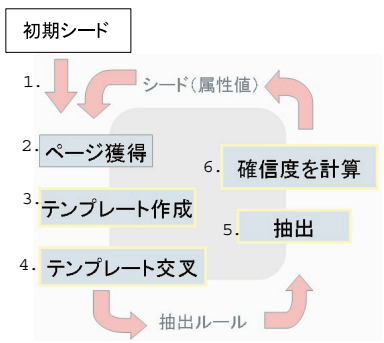

図 2　ブートストラッピングによる情報抽出

表 1　入力値の例

属性名	”作品”,”作品名”	”監督”,”ディレクター”	”出演”
属性値	タイタニック 宇宙戦争	ジェームズ・キャメロン スティーブン・スピルバーグ	レオナルド・ディカプリオ トム・クルーズ

t_i^{right} と一致する部分が存在するかどうかを調べる．一致する部分があるならば，t_i^{left} と t_i^{right} で囲まれる部分を抽出候補とし，(3) に進む．(3) 抽出候補に対する USP を取り出し，t^{up} 中の単語がすべて含まれるかどうかを調べる．もしそうならば，抽出候補を属性 f の属性値として抽出する．本章ではまず，4.1 節において，USP の位置を特定し USP を抽出する方法について述べる．次に 4.2 節において，このテンプレートの作成方法について述べる．

4.1　USP の抽出

Web 上には表や箇条書きなどの様々な記述形式が存在しており，USP の位置は記述形式によって異なる．図 3 に複数の記述形式における USP の位置の例を示す．この図では各記述形式に対して，項目（「A1」や「A2」など）に対する USP を灰色の部分（「属性名 A」など）として表現している．このようにある語に対する USP の位置はその語が含まれる記述形式によって異なるため，一般的な DOM 構造において USP を抽出することは容易ではない．

そこで本研究では，USP の位置が明確な特定の記述形式のみを対象とし，その他の記述形式からは USP の抽出を行わないこととする．そして，USP の抽出は対象の記述形式ごとに予め用意された抽出ルールを用いて行う．このために我々はある記述形式に対する USP の抽出ルールを，記述形式の判定条件と USP の位置によってモデル化した．記述形式の判定条件は記述形式を特定するための条件であり，DOM 構造上の特徴で記述される．USP の位置はある記述が記述形式の判定条件に当てはまる場合に，抽出される位置を表す．具体的な USP の抽出ルールの例として，本論文の実験で用いた 3 つの抽出ルールを表 2 に示す．これは図 3 で示した 3 つの記述形式に対する抽出ルールである．なお，USP の抽出方法については，今後高性能なものが考案される可能性がある上，Web ページの記述スタイルも今後変化していくことが考えられる．このため我々は表 2 の抽出ルールを本研究の新規性として捉えるのではなく，任意の抽出方式や抽出ルールが使えるものとして考えている．

4.2　テンプレートの作成

本研究では USP を抽出する際に特定の記述形式のみを対象としており，予め定義されていない記述形式に対しては USP を取り出さず，前後の文字列のみのテンプレートを作成する．

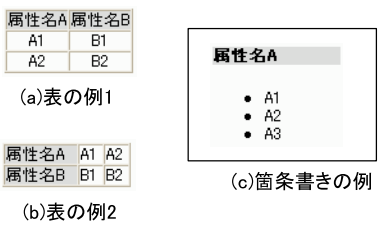

図 3　表と箇条書きの例

ここでは USP を含んだテンプレートと USP を含まないテンプレートを作成する手法を述べる．これらのテンプレートは次の手順で作成される．テンプレートは次の手順で作成される．

（1） シード中の属性値を用いて，取得したページ全体から属性値が記述される部分を探す．属性値が発見されると前後 N 単語と USP を取り出す．USP が抽出でき，その中に属性名が含まれる場合，USP 中の名詞(注1)と前後 N 単語，発見された属性値の属性を，一つのテンプレートの候補 (以下，テンプレート候補) として保存する．USP が抽出できない場合または USP 中に属性名が含まれない場合，前後の N 単語のみをテンプレート候補として保存する．これらの処理を収集したページ全体に行うことにより，テンプレート候補の集合 T が作成される．

（2） T 中に登場する単語について各属性に対する評価値を計算する．属性 f に対する単語 w_i についての評価値 $E_{word}(w_i, f)$ は次の式で計算される．

$$E_{word}(w_i, f) = log(tf_{all}(w_i)) \times P(f|w_i) \times sf(w_i) \times pos(w_i)$$

$tf_{all}(w_i)$ は単語 w_i の全文書中の登場頻度を表しており，サポート率の高い単語に重みを付けるために用いた．$P(f|w_i)$ は属性 f の属性値の付近 (前後 N 語以内か USP 内) に単語 w_i が登場する回数を $tf_{all}(w_i)$ で割ったものである．これは属性と関係の高い語に重みを付けるために用いた．$sf(w_i)$ は単語 w_i の付近に存在する属性値の種類数であり，特定の属性値とのみ関連のある語の重みを下げるために用いた(注2)．$pos(w_i)$ は語自体の特徴による重みを表している．実験では語が属性名や属性値である場合を 10，記号，HTML タグ，助詞の場合を 0 としている．この重み付けは助詞や HTML タグなど属性値とは無関係の一般的な語を無視し，属性と関係のある語を高くするために用いた．なお，これらの重みは予備実験により決定した．

（3） T 中の各テンプレート候補に対して，評価値を計算する．テンプレート候補 t_i の評価値 $E_{template}(t_i)$ は次の式で計算される．

$$E_{template}(t_i) = \alpha \cdot E_{left}(t_i) + \beta \cdot E_{right}(t_i) + \gamma \cdot E_{up}(t_i)$$

$E_{left}(t_i)$，$E_{right}(t_i)$，$E_{up}(t_i)$ はそれぞれ左側 N 語の単語から計算される評価値，右側 N 語の単語から計算される評価値，

(注1)：USP の単語は前後の単語とは異なり，属性値に対して助詞などによる文法上の係り受け関係が存在しない．このため名詞のみを取り出す．

(注2)：例えば，属性が作品名である語「タイタニック」の付近には「沈没」や「号」などの語が多い．これらの語は「タイタニック」という属性値のみと相関が強いが，作品名という属性から観ると無関係である．このような特定の属性値に依存するような語の重みを下げるために $sf(w_i)$ を用いた．

表 2　抽出ルールの例

図 3-(a) に対する抽出ルール	
記述形式の判定条件	(1). DOM 構造において，対象の語が存在する位置から上位 5 階層以内に table タグが存在する． (2). (1) の table タグ内に他の table タグや input タグ，form タグ，img タグが含まれない． (3). (1) の table の一行一列目以外の一行目のセルに属性名が含まれ，1 行目以外の位置には含まれない． (4). (1) の table の一行目のセルが持つテキストの長さが 22 文字を超えない．
USP の位置	同列の一行目のセルが持つテキストノード

図 3-(b) に対する抽出ルール	
記述形式の判定条件	(1). DOM 構造において，対象の語が存在する位置から上位 5 階層以内に table タグが存在する． (2). (1) の table タグ内に他の table タグや input タグ，form タグ，img タグが含まれない． (3). (1) の table の一行一列目以外の一列目のセルに属性名が含まれ，1 列目以外の位置には含まれない． (4). (1) の table の一列目のセルが持つテキストの長さが 22 文字を超えない．
USP の位置	同行の一列目のセルが持つテキストノード

図 3-(c) に対する抽出ルール	
記述形式の判定条件	(1). DOM 構造において，対象の語が存在する位置から上位 3 階層以内にリストタグ (ul タグもしくは ol タグ) が存在する． (2). (1) のリストタグ内に他のリストタグ，table タグ，input タグ，form タグ，img タグ，属性名が含まれない． (3). (1) の table のリストタグの上位 3 階層以内において，リストノードの先祖ノードの手前に文字数が 22 文字を超えないノードがある．
USP の位置	記述形式の条件 (3) で指定されるテキストノード

USP 中の単語から計算される評価値を表している．ただし，USP を含まないテンプレート候補に対しては $E_{up}(t_i)$ は 0 となる．α，β，γ はこの 3 つの評価値の重みを表しており，実験では $\alpha = \beta = \gamma = 1$ とした．

なお，$E_{left}(t_i)$ と $E_{right}(t_i)$ は含まれる語の $E(w, f)$ の最大値とし，$E_{up}(t_i)$ は含まれる語の $E(w)$ の和とする．この違いは次の違いによる．テンプレートのうち左右の単語は抽出語の前後と一致する必要があるため，HTML タグや助詞など本来 $E(w)$ が低い語が含まれる場合も多いが，この場合でも他に $E(w)$ の高い語があれば重要なテンプレートとする必要がある．これに対し $E_{up}(t_i)$ では USP 中の名詞のみで構成されているためすべての語が重要となる．

（4） 評価値 $E_{template}(t_i)$ が閾値 θ 以上のテンプレート候補をテンプレートとする．

5. テンプレートの交叉

本研究では，交叉によって間違ったテンプレートが生成されないように交叉するテンプレートを選択する必要がある．このために，本研究ではテンプレートの評価値 $E_{template}(t_i)$ が高く，似ているテンプレート同士を交叉させることとする．具体的にはテンプレート t_i と t_j 間の類似度 $S(t_i, t_j)$ を次のように定義し，この値が高いテンプレート同士のみを交叉させることとする．

$$S(t_i, t_j) = \alpha \cdot S_{left}(t_i, t_j) + \beta \cdot S_{right}(t_i, t_j) + \gamma \cdot S_{up}(t_i, t_j)$$

$S_{left}(t_i, t_j)$，$S_{right}(t_i, t_j)$，$S_{up}(t_i, t_j)$ はそれぞれ左側 N 語の単語から計算される類似度，右側 N 個の単語から計算される類似度，USP 中の単語から計算される類似度を表している．α，β，γ はそれらの重みであるが，4.2 節の値と同じものとする．$S_{left}(t_i, t_j)$ は t_i の左側 N 語の単語列と t_j の左側 N 語の単語列を順番に調べ，同じ位置に同じ単語がある場合にその単語の評価値 $E(w, f)$ を足していった合計である．$S_{right}(t_i, t_j)$ に関しても同様に計算する．$S_{up}(t_i, t_j)$ は t_i の USP 中の単語 M 個を順番に調べていき，t_j の USP 中に同じ単語が含まれていればその単語の評価値 $E(w, f)$ を足していった合計である．$S_{up}(t_i, t_j)$ については，USP 中の単語の順番は重要でないため，同じ単語が含まれていたら加点する．

テンプレートの選択を行う際には，全テンプレートのペアについて類似度 $S(t_i, t_j)$ を計算し，上位 C% のテンプレートを交叉させることとする．交叉は同じ位置にある同じ品詞の単語で異なる単語同士を入れ換えることで行う．入れ換えられる位置が複数ある場合は，それぞれの位置で入れ換えを行い，各入れ替えで生まれた新しいパターンのテンプレートをすべて作成することとする．

6. 確 信 度

ブートストラッピングでは誤った値が混入するすると，その値によって誤ったテンプレートが作成され，それによりさらに謝った値が抽出されてしまう．これを防ぐためには各サイクルごとに抽出後に属性値とテンプレートを再評価することが重要である．本研究では Yangarber ら [13] が用いた確信度によりテンプレートと属性値を再評価し，閾値より下回るものを次のサイクルでは使用しないことで精度を保つ．Yangarber らはテンプレート t_i に対する確信度 $C_{template}(t_i)$ と，属性値 w_e に対する確信度 $C_{value}(w_e)$ を次のように計算している．

$$C_{template}(t_i) = Prob(t_i) \cdot log|pos(t_i)| \quad (1)$$

$$C_{value}(w_e) = 1 - \prod_{t_k \subset M_t} (1 - Prob(t_k)) \quad (2)$$

ただし，$Prob(t_i)$ は t_i が抽出したすべての属性値のうち，正しい属性値として既にシードに含まれている属性値の割合とする．$pos(t_i)$ は t_i によって抽出された属性値の中で，正しい属性値として既にシードに含まれている属性値の数とする．M_t は w_e を抽出したテンプレートの集合とする．(1) 式はより正確により多くの正しい属性値を抽出できたテンプレートに高い値をつけるためにこの形になっている．(2) 式は次の考えに基づいている．属性値 w_e が M_t に含まれる複数のテンプレー

トによって抽出される場合，それら全てのテンプレートが間違って属性値を抽出していると，抽出した値は正しい属性値ではない．(2) 式では，この場合を除いた事象が起こる確率を確信度と定義している．ただし，Yangarger らは M_t に含まれるテンプレートが 1 つしか無い場合，$C_{value}(w_e) = 0$ としている．本研究でもこの方法を用い，テンプレートの確信度の閾値と属性値の確信度の閾値としてそれぞれ $C_{MinTemplate}$ と $C_{MinValue}$ を用意しておき，各サイクルにおいてこれらよりも確信度が小さいテンプレートと属性値を削除する．実験では $C_{MinTemplate} = C_{MinValue} = 0.5$ とした．

7. 評価実験

本研究ではブートストラッピング・アルゴリズムにおいて，より多くの属性値を抽出できるテンプレートを作成するため，テンプレートに USP を用いる手法と似たテンプレート同士を交叉させる手法を提案した．そこで本実験では，テンプレートに USP と交叉を用いない手法 (Baseline 法)，USP を用いる手法（USP 法），交叉を用いる手法 (Crossover 法)，USP と交叉を用いる手法 (BootCross 法) の 4 つを比較することで，提案手法の有効性を評価する．本章ではまず 7.1 節において実験で用いる評価指標と実験の方法について述べる．次に，7.2 節において Baseline 法，USP 法，Crossover 法，BootCross 法を比較する実験について述べる．

7.1 実験方法

本研究では各手法の抽出結果を比較する指標として，各抽出手法で得られた属性値の種類の数 (以下，語彙数)，各抽出手法で得られた属性値の種類の数に対する正しい属性値の種類の数の割合 (以下，語彙精度)，全ページ中から各抽出手法で抽出できた正しい属性値の総数 (以下，レコード数)，全ページ中から各抽出手法で抽出した属性値のうち正しい属性値の割合 (以下，レコード精度) を用いる．

実験は英語で記述されたページから映画情報（属性は作品名，監督名，出演者名の 3 つとする）を抽出することとし，初期シードとして各属性 10 個の属性値と属性名として計 10 個のキーワードを用意した．実験では収集した文書が 10 万件になった時点でページの収集を止め，そのサイクルを最後のサイクルとして抽出を行った後，その時点で得られた属性値が正しいかどうかを調べる．属性値が正しいかどうかを調べる際には，IMDb [22] という映画の情報を公開しているサイトを利用した(注3)．

ただし，USP や交叉を用いない場合においても，テンプレートを生成する際のパラメータ (ウインドウ幅 N とパターンの評価値 θ) を調節することで，精度 (語彙精度とレコード精度) と抽出数（語彙数とレコード数）は変化する．そこで我々はまず，Baseline 法においてこれらのパラメータを変化させ (N に対して 3 と 5，θ に対して 500,1000,1500)，これらのパラメータによって精度と抽出数がどのように変化するかを調べた．この結果から，最も精度が高かった N=5，θ=1000 の場合を Baseline1 法，最も抽出数が大きかった N=3，θ=500 の場合を Baseline2 法とし，これらと USP 法，Crossover 法，BootCross 法を比較することで，提案手法の有効性を調べる．なお，USP 法，Crossover 法，BootCross 法のテンプレート生成時のパラメータとしては，Baseline1 法と同じ N=5，θ=1000 を用いた．本提案手法は従来の手法に対して抽出数を増やすこと目指したものであるため，抽出数が多く精度の低いものを用いるよりも，精度が高く抽出数が少ないものを用いた方が有効だと考えられるからである．

表 3 実験条件

抽出する単語列の長さ L	5
確信度の閾値	$C_{MinTemplate} = C_{MinValue} = 0.5$
USP の抽出ルール(注4)	表 2 の抽出ルール
交叉の閾値 C (注5)	10%(ただし最大 500 個とする)

その他の実験条件を表 3 に示す．USP の抽出ルールに関しては，USP の記述位置を調べる予備調査を行った上で，誤った USP を抽出することが無いように限定的なルールとして作成した．これらのルールの再現率は，table タグを用いて記述された表 270 件に対して 59%(159 件)，list タグを用いて記述された箇条書き 126 件に対して 42%(54 件) であった．

7.2 実験結果

我々はまず，各手法の語彙数・語彙精度の関係とレコード数・レコード精度の関係をグラフにした．この結果について 7.2.1 節で述べる．次に我々は，各手法の特徴をつかむため，テンプレートの数，マッチング回数，抽出したレコード数，抽出した語彙数を比較する調査を行った．この調査について 7.2.2 節で述べる．

7.2.1 精度と抽出数の関係

実験結果を図 4 と図 5 に示す．まず図 4 に注目すると，USP 法が最も語彙精度が高い．これは USP 中の語をテンプレートに追加したことで，より正確に値を抽出できることを示している．最終的な語彙数については，Baseline2 法に劣るものの，提案手法の中では Crossover 法が最も大きい．これはテンプレートの交叉を行うことで，より多くの値を抽出できたことを示している．それらを組み合わせた BootCross 法はそれらのほぼ中間に位置しており，USP 法よりも精度が低いもののより多くの語彙を獲得しており，Crossover 法よりも獲得できた語彙は少ないものの精度が高い．BootCross 法と二つの Baseline を比較すると，BootCross 法は語彙精度を重視した Baseline1 よりも高い語彙精度で，最終的には約 1.5 倍の語彙を獲得できることがわかる．また語彙数を重視した Baseline2 に対しては，最終的な語彙数で 200 個程度劣ってはいるもの，約 15%高い精度で属性値できる．これらのことから本研究の提案手法は語彙を抽出する上で有効であると考えられる．

次に，図 5 に注目すると，USP 法と BootCross 法が高いレコード精度で多くのレコードを抽出していることがわかる．このことから USP を用いたテンプレートは，レコードを抽出す

(注3)：我々はこのサイトに対して抽出した値を送信しその値が正しいかどうかを調べるプログラムを作成し，これを用いて正しい属性値と判定されたものだけを正解としておき，残りの値については手作業で調べることで正解かどうかを判定した．

る上でも有効であると考えられる．これに対し，Crossover 法はレコード精度とレコード数の関係において Baseline1 法よりも下回っている．このことから，Crossover 法はレコードの抽出においてあまり有効でないと考えられる．また，これらの結果を得た原因については 7.2.2 節で詳しく述べる．BootCross 法と二つの Baseline を比較すると，Baseline1 法に対してはより高いレコード精度で，約 1.6 倍の属性値を獲得できる．また Baseline2 法に対してはほぼ同等の数の属性値を 16%ほど高い精度で抽出することができる．このことから BootCross はレコードを抽出する上でも有効であると考えられる．

7.2.2 テンプレートの分析

本節では各提案手法が実験において，生成したテンプレートの総数（テンプレート数），テンプレートがページにマッチングした回数（マッチング数），最終的に抽出できたレコード数と語彙数を比較する．図 6 にこれらの結果を示す．なお，図では比較のためにこれらの 3 つの手法と同じ N=5，θ=1000 でテンプレートを作成した Baseline1 法についての結果も示している．

まず，USP 法と Baseline1 法を比較する．USP 法のテンプレート数は Baseline1 法のテンプレート数とあまり変わらないのに対し，USP 法のマッチング数は Baseline1 法に対して増加している (約 1.3 倍)．これは，USP 法で用いるテンプレートは表や箇条書きにもマッチングすることができるためである．これらの記述形式では多くの属性値が列挙されることが多いため，一つのテンプレートで多くのレコードとマッチングを取ることができたと考えられる．また，USP 法は Baseline1 法に対して約 1.4 倍の数のレコードを抽出することができているのに対し，

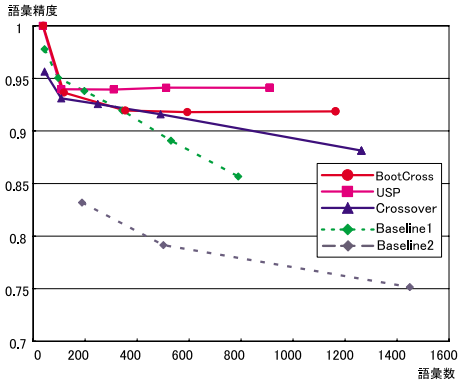

図 4　各手法の語彙数と語彙精度の関係

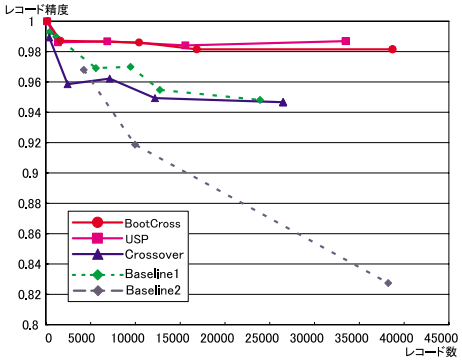

図 5　各手法のレコード数とレコード精度の関係

語彙数では Baseline1 法の約 1.2 倍程度とマッチング数の増加に比べて小さい値になっている．USP を用いたテンプレートは表や箇条書きから抽出を行うことができるが，Web 上の表や箇条書きには同じ情報が含まれていることも多い．例えば「アカデミー賞受賞作品一覧」や「トム・クルーズ過去の出演作」などがそうである．これらはページが異なっていても，同じ事実について述べているため含まれる属性値は同じである．USP を用いたテンプレートはこれらから抽出を行うため，語彙数あたりのレコード数が大きくなるという特徴がある．

次に，Crossover 法と Baseline1 法を比較すると，Crossover 法のテンプレート数は Baseline1 法の約 1.6 倍になり，マッチング数も Baseline1 法の約 1.7 倍に増加している．これは交叉によってテンプレートの数が増え，それにより多くのページとマッチングすることができたことを示している．レコード数と語彙数に注目すると，語彙数では Baseline1 法に比べて大きく増加している (約 1.6 倍) ものの，レコード数では Baseline1 法に比べてそれほど増加していない (約 1.1 倍)．これは，交叉によって抽出できるようになった属性値には，新しい語彙が多いことを示している．交叉によって生成されたテンプレートは事例としては登場しないパターンであるため，それによって抽出される語彙は新しい語彙であることが多かったものと推測される．

図 7 に実際に見られた交叉の例を示す．図では各行の t1 から t6 がそれぞれテンプレートを表現している．例えば t1 と t2 のテンプレートを持つとき，Baseline 法ではこれらのパターンに完全に一致するページからしか抽出を行うことができず，2002 年に発売された DVD の販売を行うページか，2001 年に発売された DVD のレビューを記述したページのみから抽出を行う．これに対し Crossover 法は矢印の部分を入れ換えることにより抽出できるページを増やす．これにより，2001 年に発売された DVD の販売を行うページ（t3 のテンプレート）や 2002 年に発売された DVD のレビューを記述したページに対しても抽出を行うことができる．さらに Crossover 法は次のサイクルにおいて t3 のテンプレートと t4 のテンプレートを交叉し t5 のテンプレートを作成し，さらに次のサイクルにおいて t6 のテンプレートと交叉を行っていた．これらのテンプレートは人間が見ても有効そうなテンプレートであることがわかる．このことが語彙数の増加につながったと推測される．

最後に，図 6 の BootCross 法に注目すると，語彙数では Crossover 法に劣るものの，マッチング数とレコード数は最も大きい．BootCross は，USP を用いることでマッチングの頻度が高いテンプレートを作成しつつ，それらを交叉させることでテンプレートの数を増やし多様性を増加させるというアプローチである．この結果はこのアプローチが正しいことを示していると考えている．

本実験の結果をまとめると次のようになる．USP を用いた手法は精度が高く，同じ属性値を抽出しやすいが多くのレコードを抽出するために有効である．交叉を用いた手法は精度はそれほど高くないものの，テンプレート同士を組み合わせることで多様性の高いテンプレートを作成でき，多くの語彙を抽出する

ために有効である．それらを組み合わせた手法は精度と抽出数のバランスが良く，多くの語彙のレコードを正確に抽出するために有効である．

8. ま と め

我々は Web 上の不均一な文書に対してブートストラッピングにより情報抽出を行うアルゴリズム BootCross を提案した．本手法は DOM 構造を用いたテンプレートを用いた点とテンプレート同士を交叉させる点に新規性がある．我々は Web 上の文書 10 万件に対して評価実験を行い，この手法が従来の手法よりも精度と抽出数の関係において効率が良く，より多くの語彙のレコードを抽出するために有効であることを示した．

本提案手法の問題点は，テンプレートの生成と抽出を行う処理に時間がかかることである．USP をテンプレートに用いるためには，従来行われていなかった DOM 構造を解析する処理時間が必要であり，テンプレート同士を交叉させると，生成されるテンプレートの数が増加するためページとマッチングを行うための処理時間が増える．今後の課題はまず，処理時間という観点から評価実験を行うと共に，この問題を解決する高速なアルゴリズムを開発することである．また，今回の実験では表と箇条書きの中でも単純で一般的な記述形式だけを対象として抽出ルールを作成した．しかし，Web 上にはそうでない記述形式も多い．従って多様な記述形式に対して自動的に USP を判別できる技術を開発することで，より多くの属性値を抽出できるようになるものと考えている．本稿で提案した技術が今後のテキストマイニング技術の発展に寄与できれば幸いである．

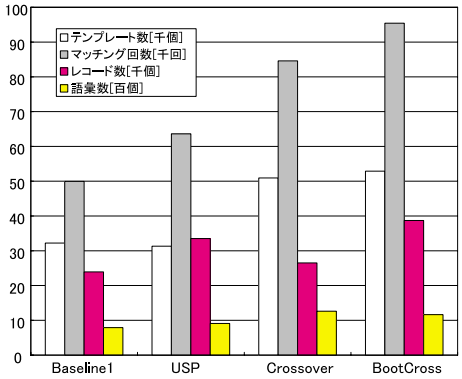

図 6　各手法のテンプレートの比較

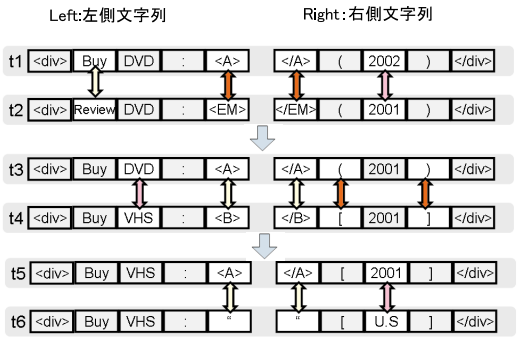

図 7　交 叉 の 例

文　献

[1] J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, pages 80–91, 1996.

[2] L. Eikvil. Information extraction from world wide web – a survey. *Technical Report No. 945, ISBN 82-539-0429-0*, 1999.

[3] 山田泰寛, 池田大輔, 坂本比呂志, 有村博紀. WWW からの情報抽出. **人工知能学会論文**, pages 302–310, 2004.

[4] D. Appelt, J. Hobbs, D. Israel, and M. Tyson. Fastus: A finite-state processor for information extraction from real-world text. *Proceedings of IJCAI-93*, pages 1172–1178, 1993.

[5] R.Baumgartner, S.Flesca, and G.Gottlob. Visual web information extraction with lixto. *Proceedings of VLDB*, pages 119 – 128, 2001.

[6] Nickolas Kushmerick, Dan Weld, and Robert Doorenbos. Wrapper induction for information extraction. *Proceedings of IJCAI*, pages 729–737, 1997.

[7] J.Ambite, N.Ashish, G.Barish, C.Knoblock, S.Minton, P.Modi, I.Muslea, A.Philpot, and S.Tejada. Ariadne: A system for constructing mediators for internet sources. *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 561–563, 1998.

[8] 山田 寛康, 工藤 拓, 松本裕治. Support vector machine を用いた日本語固有表現抽出. **情報処理学会論文誌**, 43(1):43–53, 2002.

[9] S.Brin. Extracting patterns and relations from the world wide web. *Proceedings of SIGMOD Workshop on Databases and the Web*, pages 172–183, 1998.

[10] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. *Proceedings of AAA-I-98*, pages 474–479, 1999.

[11] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. *Proceedings of ACM DL2000*, pages 85–94, 2000.

[12] *Document Object Model (DOM) Level 3 Core Specification, W3C Working Draft*, April 2002. http://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20020409/.

[13] R. Yangarber and L.R. Grishman. Unsupervised learning of generalized names. *Proceedings of COLING 2002*, pages 474–479, 2002.

[14] O.Etzioni, M.Cafarella, D.Downey, S.Kok, A.M.Popescu, T.Shaked, S.Soderland, D.S. Weld, and A.Yates. Web-scale information extraction in knowitall. *Proceedings of WWW2004*, pages 100–110, 2004.

[15] W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in html documents. *Proceedings of WWW2002*, pages 232–241, 2002.

[16] L.Liu, C.Pu, and W.Han. XWRAP: An XML-enabled wrapper construction system for web information sources. *Proceedings of ICDE*, pages 611–621, 2000.

[17] S.Gupta, G.Kaiser, D.Neistadt, and P.Grimm. Dom-based content extraction of html documents. *Proceedings of WWW2003*, pages 207 – 214, 2003.

[18] G. Salton, A. Wang, and C. Yang. A vector space model for information retrieval. *the American Society for Information Science*, 18:613–620, 1975.

[19] H. Cui, M. Y. Kan, and T. S. Chua. Unsupervised learning of soft patterns for generating definitions from online news. *Proceedings of WWW2004*, pages 90 – 99, 2004.

[20] Google Web APIs. http://www.google.com/apis/.

[21] CyberNeko HTML Parser. http://people.apache.org/andyc/neko/doc/html/.

[22] IMDb:The Internet Movie Database. http://www.imdb.com/