

Webからのスキーマ抽出に関する基礎検討

中根 史敬[†] 大坪 正典[†] 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: †{nakane,hijikata,nishida}@nishilab.sys.es.osaka-u.ac.jp

あらまし Web上には大量の半構造化文書が存在するが、それらに書かれた情報はデータベースのように整形されていない。そこで、属性と属性値の形でそれらの情報を抽出する研究（情報抽出）が盛んに行われてきた。具体的な課題としてはスキーマ（属性名の組）の抽出と属性名に対応する属性値の抽出の2つがある。そのうち、属性値を抽出する手法として教師信号を必要としないブートストラッピングが注目を浴びている。しかし、スキーマ抽出における属性名の獲得にブートストラッピングをそのまま適用した研究はない。もし属性名の獲得にブートストラッピングを適用しようとする、属性名の前後のテキストを抽出用のテンプレートとする必要がある。しかし、このテンプレートは属性値に該当するテキストも含み、非常に多様となってしまう。そのため、他のページにおいて適合する確率が低くなる問題がある。そこで本研究では、多様性の低いテンプレートを作成し、作成したページにのみ適用することでこの問題を解決し、ブートストラッピングによるスキーマ抽出の実現を目指す。また、属性名の抽出に有用でないページのフィルタリングと多様なスキーマの抽出を行うためのアルゴリズムも提案する。

キーワード 情報抽出, スキーマ, ブートストラッピング, 半構造化データ

A Basic Study on Schema Extraction From the Web

Fumitaka NAKANE[†], Masanori OTSUBO[†], Yoshinori HIJIKATA[†], and Shogo NISHIDA[†]

[†] Graduate School of Engineering Science, Osaka University 1-3 Machikaneyama, Toyonaka, Osaka
560-8531, Japan

E-mail: †{nakane,hijikata,nishida}@nishilab.sys.es.osaka-u.ac.jp

Abstract There are a vast number of semi-structured documents on the web. To utilize information written on them, schemas (sets of attribute names) and attribute values have to be extracted and stored in a database. Recently, there are a lot of researches using bootstrapping algorithm to extract attribute values. However, the algorithm is not directly applicable for schema extraction. In being applied to extract new attribute names, it uses texts before and after an attribute name which is already extracted as a template. But because the template can include some attribute values, it is very diverse and has low probability of matching in other pages. We are aiming at extracting schemas using bootstrapping. To resolve above problem, our method makes a template which have low diversity and apply them only on a page where the template is made. Furthermore, we propose algorithm for extracting diverse schemas and filtering pages unuseful for extracting attribute names.

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

1. はじめに

Web上には大量の半構造化文書がある。しかし、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、計算機はこれらに記述された情報を正確に扱うことができない。そこで、属性と属性値の形でそれらの情報を抽出する研究（情報抽出）がこれまで盛んに行われてきた [1] ~ [3]。具体的な課題としては、半構造化文書から (1) データベースで必要となるスキーマを抽出すること (2) スキーマを構成する各属性に対す

る属性値を抽出すること、の2点がある。

スキーマ設計の際には、データベース中のデータの構造、形式、関連、整合性制約などを考慮しなければならない。しかし、最も単純には、あるキーとなる値（オブジェクトをある程度一意に特定可能な値。例えば、ID や人名、製品名など）を含むレコードを構成する属性の組が分かっていることが必要であろう。例えばパソコン「IBM ThinkPad X60」をキーとなる値と考え、そのレコードの属性を Web 上で調べてみる。kakaku.com などの価格の比較サイトでの属性は、「メーカー」、「型番」、「最安

のショップ名」「最安の価格」など価格やショップ名を中心とした属性となる。しかし、IBM(Lenovo)の公式サイトでは、「型番」「プロセッサ」「主記憶」などパソコンの仕様を中心とした属性となる。このように同じパソコンであっても、その情報を発信する Web サイトの目的により、その属性は異なることが分かる。このようなスキーマ(属性の組)が事前に分かっていると、より効率よく属性値も抽出できるものと思われる。

これまで、前述(2)の属性値抽出については、情報抽出の研究分野で多くの研究が行われてきた[1]~[3]。従来の情報抽出手法には、(i) 予め人手で作成した抽出ルール(またはテンプレート)を用いる手法[4],[5]や、(ii) 予め人手で文書中の抽出位置にタグ付けをしておき、機械学習により抽出ルールを学習する手法[6]~[8]、(iii) 少数の抽出する値を手で与えておき、抽出ルールと抽出値を交互に繰り返し学習していく手法(ブートストラッピング[9]~[12])がある。これらの手法は、ニュース記事などの均質な文書だけでなく、Web 文書などの多様な形式の文書においても、ある程度の精度で情報抽出できるようになってきている[13]。特にブートストラッピングは、人手を必要としない手法であるため、近年注目を集めている。

前述(1)の Web からのスキーマ抽出に関する既存の研究には、フォームや表などを対象としたものがある[14]~[18]。またこれらのうち[17],[18]は人手で表中のセルにラベル付けした事例から属性名を学習している。これらの手法は特定の記述形式を対象としており、人手をかけずに網羅的にスキーマを抽出することはできない。

そこで本研究では、人手必要としないアルゴリズムであるブートストラッピングを用い、Web から網羅的に属性名を獲得することによってスキーマを抽出を行う。ただし、スキーマ抽出における属性名の獲得にブートストラッピングをそのまま適用すると、多様性が高く、Web 全体に適用した場合に適合する確率が低いテンプレートが作成されてしまう。この問題には、2つの属性名を用いてページの検索と多様性の低いテンプレートの作成を行い、作成したページにのみテンプレートを適用することで対応する。また、属性名のみを用いて検索を行うと有用でないページ(用語解説など)がヒットすることがある。このようなページがヒットするのを防ぎ、抽出するスキーマを持つオブジェクトの情報が書かれたページを獲得するために、キーとなる値もクエリに含めて検索を行う。さらに、キーとなる値として決まったもののみを用いていると抽出できる属性名は限られたものになってしまうため、キーとなる値もブートストラッピングで学習する。これにより多様なスキーマの抽出を目指す。

2. ブートストラッピングによる情報抽出

ここで情報抽出におけるブートストラッピングの基本的な流れを、Web 上からパソコンの製品名を抽出する場合を例に説明する。この手法は予め用意された「ThinkPad X60」などの属性値を少数含む辞書(シード)を用意しておき、その属性値を含む文書を検索する。次に、それらの文書を調べ「最新 PC ThinkPad X60 レビュー」といったパターンを獲得する。続い

て、このパターンから属性値部分を抽出位置とみなすことで「最新 PC (抽出位置) レビュー」というテンプレートを作成する。このテンプレートで文書集合を調べることで、別の記述「最新 PC Latitude D620 レビュー」から新たな属性値「Latitude D620」を抽出することができる。さらに、得られた属性値をシードに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ことができる。

3. 関連研究

本研究の関連研究として、ブートストラッピングによる情報抽出の研究について 3.1 節で、スキーマの抽出を行う研究について 3.2 節で述べる。

3.1 ブートストラッピングによる情報抽出

Riloff ら、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[11],[19]。新聞記事には決まった言い回し(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そのままテンプレートとしている。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。本研究でもこの確信度を用いている。確信度の詳細については 5.3.1 節で述べる。

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[9]。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クス(著者名は大文字で始まる単語の連続とする、など)を利用している。

Ciravegna は、Web 上からコンピュータ科学の研究者の人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[10]。彼らは精度を保つために、抽出した人名が既存のデータベースやデジタルライブラリ中で有効であるかを判定し、不確かなものを削除している。

楠村らは、Web 文書にブートストラッピングアルゴリズムを適用した場合の2つの問題に対する解決方法を提案している[20]。一つ目の問題は、表や箇条書きなどの構造化された記述形式から抽出を行うことができないということである。これに対しては、DOM 構造^(注1)において上位にある語を利用したテンプレートを作成することで対応している。二つ目の問題は、Web 上の文書の記述形式は多様であり、微妙な差異があるだけで抽出できないため、事例から獲得したテンプレートだけでは数が足りないということである。これに対しては、事例から獲得したテンプレートを交差し、新しいテンプレートを生成することで対応している。また彼らは、テンプレートと抽出値の確信度の計算に Yangarber らの方法[19]を用いている。

本研究もブートストラッピングによる情報抽出を行うものであるが、以下の2点において上記の研究と大きく異なる。1つ目は、上記の研究が属性値の抽出を行うのに対し、本研究は属性名の抽出を行う点である。2つ目は、上記の研究では、ブー

(注1): <http://www.w3.org/DOM/>

トストラッピングの各サイクルにおいて、作成した抽出用テンプレートを Web などの文書集合に対して適用するのに対し、本研究の属性名抽出ではテンプレートを、作成したページにのみ適用する点である。これにより多様な属性名を高い精度で抽出することを目指す。

3.2 スキーマの抽出

Tokunaga らは、クラス語 C を入力とし、そのクラスの属性名を抽出する手法を提案している [21]。具体的には、C を含む文書を検索により Web から獲得し、それらからすべての名詞 A を属性名候補として抽出し、出現パターンと重要度によりスコアリングを行い上位のものを出力する。出現パターンとして、主に「C の A」というクラス語との共起関係を用いている。

Yoshida らは論理的構造（各行、各列が属性名を表すか属性値を表すか）に基づいて Web ページ中の表を 9 つの型に分類するアルゴリズムを提案している [22]。表の型とは、例えば「1 列目は属性名、その他は属性値を表す」といったものである。具体的には、表の各セルが属性名/属性値である確率と、分類結果を仮定しその時の型を適合した場合の表のスコアを EM アルゴリズムによって繰り返し計算することによって尤もらしい表の型を決定している。また [23] ではテキストの出現頻度を用いた方法も提案している。

Tengli らは、表中のセルに人手でラベル付けした事例から、表の論理的構造の特定を行っている [17]。具体的には、属性名あるいは属性値とラベル付けされた語を含むセルと隣接するセルについて、セルに含まれる文字列どうしの類似度を計算することで、表中において属性名あるいは属性値を表すセルを特定する。

Chen らは、内容が類似する（テキストが部分一致する、同じカテゴリの固有表現である、あるいは数値である）セルを多く含むものを表とみなすなどのヒューリスティクスを用いてページ中の表を認識した後、表の論理的構造を特定している [18]。論理的構造の特定の際には、内容の類似するセルを結合していき、結合の境界を属性名と属性値の境界とみなしている。

また [14] ~ [16] では、フォームのテキスト入力フィールドに隣接しているテキストを属性名として抽出したり、表の行に存在するテキストを属性名と属性値の組とみなしたり、入力フォームの Widget の種類（テキスト入力フィールドやラジオボタンなど）ごとに抽出するヒューリスティクスを変更したりしている。

上記の研究と本研究の違いについて述べる [21] では、クラス語との共起関係に基づいて属性名を抽出している。しかし、属性名はクラス語の近くではなく、むしろ図 2 のように属性値や他の属性名の近くに現れることが多い。さらに、特定のパターンに限らず、表やリストなど、多様な形式によって属性名は記述される。本研究では、2 つの属性名の前後テキストの完全一致部分をテンプレートとするので、抽出ルールをクラス語との共起関係や特定の記述パターンに限定することなく属性名の抽出を行うことができる。

[17], [18], [22], [23] は表の論理的構造の特定を行う手法である。これらの手法を用いれば、表中の属性名を抽出することが

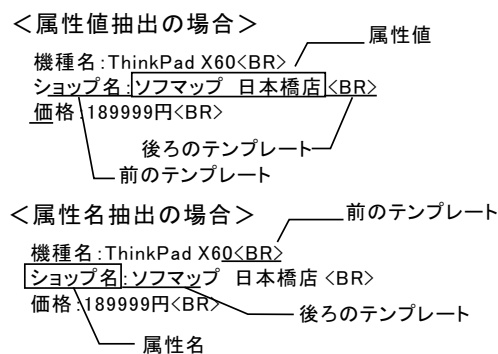


図 1 属性値抽出と属性名抽出

できる [14] ~ [16] でも、表中の属性名やフォームに隣接する属性名を抽出することができる。しかし、これらの手法では表やフォームといった特定の記述形式からしか属性名を抽出することができない。それに対して本研究では、表やフォーム以外にも、箇条書きやリスト、あるいは同じ形式で繰り返し属性名が記述されているプレーンテキストなどからも属性名の抽出が可能である。

4. 本手法の概要

これまでの情報抽出におけるブートストラッピングは、属性値の抽出を行ってきた。属性値の前後には、それに対応する属性名と、他の属性名が存在することが多いため、そのテキストの多様性は低かった。そのため、前後のテキストが属性値の抽出に有効であったと考えられる。

本研究では属性名を抽出しなければならないが、この手法をそのまま用いるには問題がある。属性名の前後（テンプレート）には属性値が含まれることが多くなる（図 1 参照）ため、その多様性は高くなる。多様性が高くなれば、各テンプレートが属性名に適合する確率が低くなり、テンプレートの有用性は低くなってしまふ。その結果、学習したテンプレートを用いて Web 全体から属性名を抽出することは困難となる。逆に、属性値を含めずにテンプレートを作成すると、それらはタグや記号などから構成されることが多くなり、属性名以外の箇所にも適合してしまうと考えられる。その結果、学習したテンプレートがそのページでは精度よく属性名を抽出できたとしても、Web 全体に適用するとノイズの抽出が増える可能性がある。

しかし、ブートストラッピングですでに抽出した値を持っており、それを次の抽出に利用できる。また、Web 全体で見れば各ページの記述形式の多様性は高いが、個々のページに着目するとレコード中のデータ要素は同じ形式で繰り返し記述されていることが多い（図 2 参照）。よって、あるレコードに含まれる 2 種類の属性名をすでに持っていれば、そのレコードを高い確率で特定できるだけでなく、現在のページで有効となる属性名抽出用のテンプレートを作成できると考えられる。

そこで本研究では、すでに獲得している 2 つの属性名をクエリに含めて Web 全体からページを検索する。検索で獲得した各ページにおいて、それぞれの属性名の前後にあるテキストで完全一致する部分をテンプレートとして取り出す。例えば、図

```

<DIV class = content id = product>
<UL>
<LI><B>メーカー型番:</B>1709571
<LI><B>タイプ:</B>B5ノート
<LI><B>CPU:</B>Core 2 Duo
<LI><B>メモリ:</B>1GB
<LI><B>HDD:</B>120GB
</UL>
</DIV>

```

図 2 属性名と属性値の記述領域の例

2において、属性名「CPU」と「メモリ」から、テンプレート「< LI > < B > (抽出位置) : </ B > 」を作成する。そして、そのテンプレートを、Web全体に適用する汎用的なテンプレートではなく、そのページにのみ有効なテンプレートとする。すなわち、作成したページにのみテンプレートを適用し、属性名を抽出することとする。この点が、汎用的なテンプレートを学習する属性値抽出用のブートストラッピングと大きく異なる。

しかし、この方法で作成したテンプレートは、その多くがタグや記号だけから構成され、短いものであると考えられる。そのため、たとえ1つのページにのみ適用したとしても、ページ中の様々な場所に適合し、目的とする属性名以外のテキストも抽出してしまう可能性がある。そこで、抽出範囲を、テンプレートを作成するのに用いた2つの属性名が記述された部分に限定する。この部分は、DOM構造を解析することによって特定する(詳しくは5.2節で述べる)。

また、本研究では2つの属性名をクエリに含めてWeb全体を検索するが、属性名だけではオブジェクトのレコードが記述されたページだけでなく、その他のページも適合してしまう。例えば「CPU」と「メモリ」で検索を行うと、あるパソコンの情報が書かれたページだけではなく、「CPU」や「メモリ」について用語解説したページなども適合する。そこで本研究では、2つの属性名に加えて、キーとなる属性値(例えば、パソコンなら製品名。以下「キー」と略す)もクエリに含めて検索を行う。また、決まったキーだけを用いていると、多様なスキーマを抽出することができない恐れがある。そこで、属性名をブートストラッピングで学習するだけでなく、キーもブートストラッピングで学習し、これらの学習を並行して行うこととする。

これらをまとめると、本手法の新規性は以下の3点にある。

- 属性名の抽出では、2つの属性名をクエリに含めた検索で適合した各ページから各属性名の前後にあるテキストで完全一致する部分を抽出し、それをそのページにのみ有効なテンプレートとして獲得する。これにより、レコードの記述形式の多様性に対応する。

- DOM構造を参照することで、上記テンプレートの適用範囲を限定する。これにより、レコード記述位置以外の部分にテンプレートを適用して、ノイズとなる値を抽出してしまうことを防ぐ。

- キーと属性名をそれぞれブートストラッピングにて並行

して抽出する。これにより、多様なスキーマの獲得を目指す。

5. スキーマ抽出手法

提案するスキーマ抽出手法の流れを図3に示す。本手法はキー抽出と属性名抽出を並行して実行することにより、多様なスキーマの獲得を目指すものである。以下、5.1節、5.2節でキー抽出の手順、属性名抽出の手順についてそれぞれ述べ、その実装について5.3節で述べる。

5.1 キー抽出の手順

キー抽出は以下の手順で行う(図3左側)。

Step1. キー・属性名リストからキー抽出において未使用であるキーを取り出し、それぞれをクエリとする。ただし、Yangarberらの方法[19]によりキーの評価値を求め、閾値を超えるもののみを用いる(Yangarberらの方法については5.3.1節で詳しく述べる)。

Step2. Web検索により上記キーを含むページを獲得し、ページリストに保存する。

Step3. キーの前後 n 単語をテンプレートとし、テンプレートリストに保存する。このとき、HTMLタグも1単語とみなす。

Step4. テンプレートリスト中のテンプレートを用いてページ集合から他のキーを抽出し、キーリストに追加する。ただし、Yangarberらの方法[19]によりテンプレートの評価値を求め、閾値を超えるもののみを用いる。

Step5. Step1.に戻る。

テンプレートの作成とマッチング(Step3.とStep4.)においてHTMLタグの属性名、属性値は無視する。それによってより多くの箇所にテンプレートが適合することになり、多くのキーを抽出することができると考えられる。本手法において属性名だけでなくキーもブートストラッピングで抽出するのは、抽出したキーを属性名の獲得に利用することによって多様なスキーマを抽出するためである。したがって、上記のようにして多くのキーを抽出することが有効であると考えられる。

5.2 属性名抽出の手順

属性名の抽出は以下の手順で行う(図3右側)。

Step1. キー・属性名リストにおいてキー1つ、属性名2つからなる組み合わせ(属性名抽出において未使用であるもの)を作成して取り出し、それぞれをクエリとする。ただし、Yangarberらの方法[19]により評価値をそれぞれ求め、閾値を超えるキー、属性値を用いるものとする。

Step2. Web検索により上記キーと2つの属性名を含むページを獲得する。

Step3. 2つの属性名の前後テキストの完全一致部分からテンプレートを作成し、抽出を行う範囲を特定する。

Step4. 上記テンプレート用いて上記範囲から属性名を抽出し、属性名リストに追加する。

Step5. Step1.に戻る。

Step3.の詳細な手順は以下のようにする。

Step3-1. 2つの属性名(属性名A, Bとする)に対して前後テキストの完全一致部分をテンプレートとする。完全一致の判

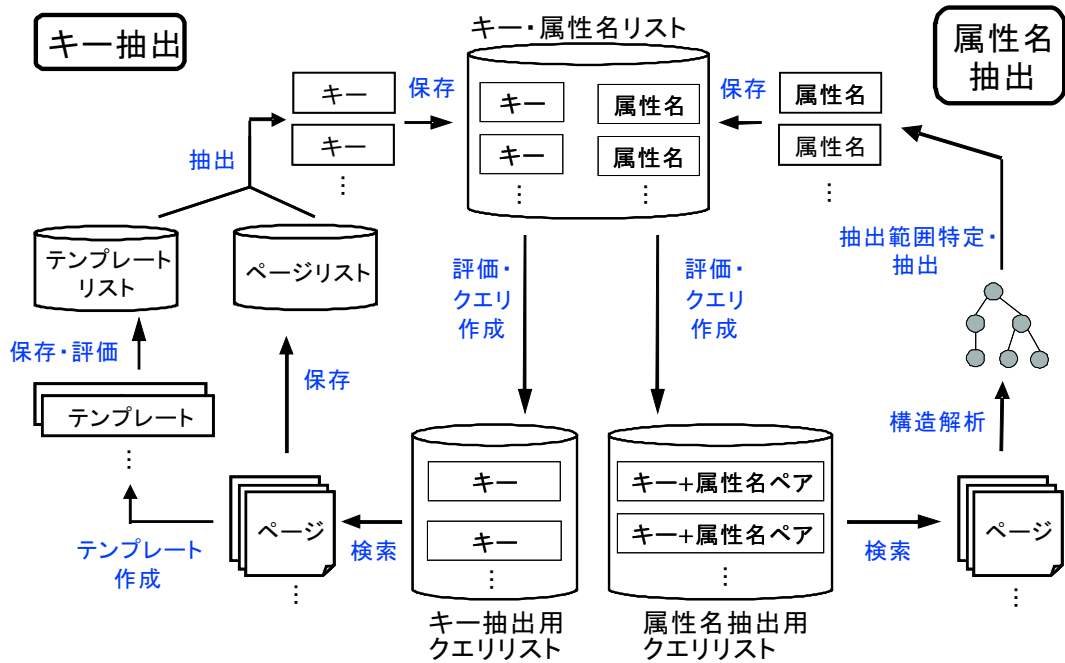


図3 スキーマ抽出手法の流れ

定は単語単位で行う (HTML タグは 1 単語とみなす)。ただし、HTML タグ中の属性値は DOM オブジェクトを一意に特定する傾向があるため、無視する。ここでテンプレートに該当する部分は 2 箇所 (2 つの属性名それぞれの前後) できる。

Step3-2. DOM 構造において、それ自身と全ての子孫からなる部分木 (部分木 A, B とする) が上記 2 箇所のテンプレートに該当する部分それぞれを全て含むような最下位のノード (ノード A, B とする) をそれぞれ特定する。

Step3-3. ノード A, B が一致すれば、それ自身と全ての子孫からなる部分木に該当するテキストから抽出を行う。一致しなければ Step3-4. へ進む。

Step3-4. ノード A, B の親ノードをそれぞれ特定する。2 つの親ノードが一致すれば、それ自身と全ての子孫からなる部分木に該当するテキストから抽出を行う。

以上の手順について、例として、図2に示したページ例 (以下、ページ例) から抽出を行う場合を説明する。図4に、ページ例の DOM 構造を示す。ページ例に対して、属性名「CPU」(属性名 A) と「メモリ」(属性名 B) からテンプレート「(抽出位置):」を作成する。図4において Step3-2. で述べた部分木 A, B を点線で、ノード A, B を丸で示す。ノード A, B は一致しないが、それぞれの親ノードは であり、一致する。したがって、ページ例で から までの部分に対してテンプレートを適用する。

5.3 実装

上記の処理について、テンプレートとキー、属性名の評価値の算出方法を 5.3.1 節で、属性名の抽出におけるクエリの利用について実装に用いたツールを 5.3.3 節で述べる。

5.3.1 テンプレートとキー、属性名の評価値

Yangarber らはテンプレート t_i に対する確信度 $C_{template}(t_i)$ と、抽出した語 w_e に対する確信度 C_{word} を次のように計算し

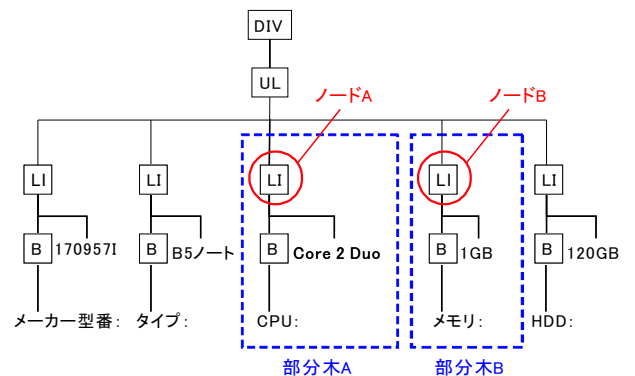


図4 図2に示したページ例の DOM 構造

ている。

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

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

ここで、

- $Prob(t_i)$: t_i が抽出した語について、すべての語の数に対する正しい語として既に辞書に含まれている語の数の割合
- $pos(t_i)$: t_i が抽出した語について、正しい語として既に辞書に含まれている語の数
- M_t : 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語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) 式の第 2 項は、語 w_e を抽出したテンプレート $t_k \in M_t$ についてそれぞれが誤って抽出する確率の積である。つまり (2) 式は語 w_e を抽出したテンプレートのうち少なくとも 1 つが正しい語を抽出する確率である。

本研究では、 $C_{word}(w_e)$ をキーと属性名それぞれの評価値、

$C_{template}(t_i)$ をテンプレートの評価値として用いる。

5.3.2 属性名抽出におけるクエリの利用

5.2 節の Step1. で述べたとおり、属性名の抽出におけるページ獲得には 1 つのキーと 2 つの属性名の組合せを検索クエリとして用いる。本手法ではキーと属性名をそれぞれブートストラッピングにて獲得するため、サイクルを重ねるごとに上記クエリは爆発的に増加する。そこで、各サイクルにおいて、全てのクエリを用いるのではなく、クエリの評価値を算出して上位 m 個のみを用いる。それは、始めのほうのサイクルで獲得されて評価値の低いキーと属性名よりも、後から獲得されていても評価値の高いキーと属性名から作成したクエリを優先的に利用したほうが効率的に高い精度で属性名を獲得できると考えられるためである。また、前述のとおり各サイクルで膨大な数のクエリが作成されるので、計算コスト、時間的コストの面からもこの方法は現実的であると考えられる。クエリの評価値は以下の式で求める。

$$Qvalue = Kvalue * \log_2\left(\frac{A1value + A2value}{2} + 1\right) \quad (3)$$

$Qvalue$: クエリの評価値

$Kvalue$: キーの評価値

$A1value$: 属性名 1 の評価値

$A2value$: 属性名 2 の評価値

5.3.3 ツール

上記の処理を実現するシステムを Java を用いて実装した。また、Web ページの検索には Google SOAP Search API^(注2)を、DOM 構造の解析には CyberNeko HTMLParser^(注3)を用いた。

6. 予備実験

これまでの述べた処理について、実装する際に決定すべきパラメータには以下のものがある。

<キー抽出> (5.1 節参照)

key_th : Step1. で使用するキーの評価値の閾値

key_pages : Step2. で獲得するページの数 (Google API で上位何位までのページを獲得するか)

temp_length : Step3. で作成するテンプレート長 (単語数) n

temp_th : Step4. で使用するテンプレートの評価値の閾値

<属性名抽出> (5.2 節参照)

attribute_th : Step1. で使用する属性名の評価値の閾値

attribute_keyth : Step1. で使用するキーの評価値の閾値

attribute_querynumth : Step1. で使用するクエリの数の閾値

attribute_pages : Step2. で獲得するページの数 (Google API で上位何位までのページを獲得するか)

これらのパラメータを変化させながら属性名抽出を実行することで、適切な値を決定する必要がある。本稿では、PC の属性名を獲得するタスクを想定し、システムを実際の Web に対して適用した場合の動作を観察する。

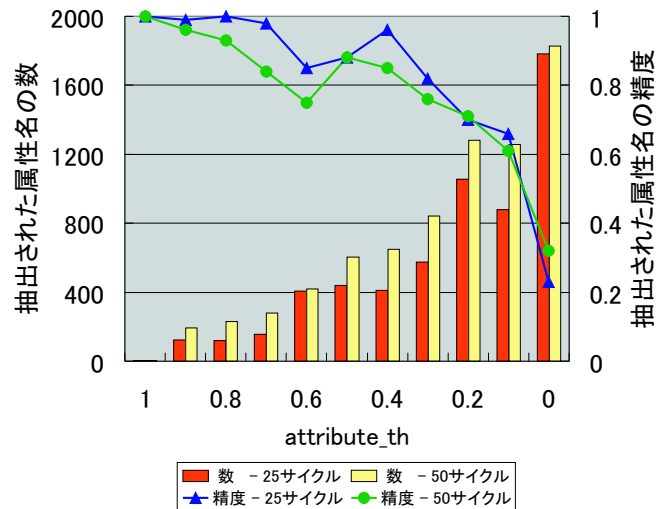


図 5 予備実験結果

6.1 設定

実験の設定について以下に述べる。シードにはキー 5 個、クエリ 5 個を手で与えておく。キーは PC の製品名、属性名は PC の属性として一般的であると考えられるもの (表 1) を用いた。これらの評価値は 5.3.1 の値に関わらず、常に 1.0 とする。

また、キーの学習は行わず、attribute_querynumth=5、attribute_pages=10 としている。

6.2 結果

attribute_th を 0.0 から 1.0 まで 0.1 ずつ変化させ、各サイクルで抽出された属性名の数と精度を求めた結果を図 5 に示す。精度を求めるにあたって、各サイクルで抽出された属性名から最大 100 個をランダムに抜き出し、実験タスクにおける属性名として妥当であるかどうかの評価を手で行った。この結果より、閾値によっては高精度で十分な数の属性名が獲得できていると言える。

6.3 今後の予備実験内容

6.2 節で述べた結果は、システムのパラメータを決定するには不十分なものであると言える。今後は前述したパラメータについて網羅的に実験を行う必要がある。また、パラメータ以外にも、結果を左右する要因として、ネットワークの状態や検索結果の変化、シードとして与えたキーや属性名が考えられる。これらの要因により正確な結果が求められないことを防ぐために、複数回試行し、求められた値を平均して結果として示すことが必要だと考えている。

表 1 シードとして用いたキーと属性名

キー	属性名
Latitude D620	Operating System
ThinkPad X60	Display
Amilo Pro V3205	Processor
Compaq nx7300	Memory
Travelmate 4202WLMi	Hard Disk Drive

(注2): <http://code.google.com/apis/soapsearch/>

(注3): <http://sourceforge.net/projects/nekhtml/>

7. まとめと今後の予定

ブートストラッピングを用いて Web から自動でスキーマを抽出する手法について述べた。本手法では、2つの属性名で Web を検索し、適合したページから各属性名の前後にあるテキストで完全一致する部分からそのページにのみ有効なテンプレートを作成することにより、レコードの記述形式の多様性に対応する。また、DOM 構造を参照することで、テンプレートの適用範囲を限定し、ノイズとなる値を抽出してしまうことを防ぐ。さらに、キーとなる属性値と属性名をそれぞれブートストラッピングにて学習し、多様なスキーマの獲得を目指す。

本稿では、システムの実装と予備実験について詳細に述べた。予備実験では、属性名抽出における性能を抽出数と精度の面から観察した。今後は、予備実験を進め各パラメータを決定した後、多手法との比較を行うことによって本手法の有効性を検証する予定である。

文 献

- [1] J.Cowie, and W.Lehner: Information extraction, Communications of the ACM, Vol. 39, No. 1, pp. 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 Web ラッパーの自動構築, 人工知能学会誌, Vol. 19, No. 3, pp. 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, pp. 1172-1178, 1993.
- [5] R.Baumgartner, S.Flesca, and G.Gottlob: Visual web information extraction with lixto, Proceedings of the 27th Very Large Databases Conference, pp. 119-128, 2001.
- [6] N.Kushmerick, D.Weld, and R.Doorenbos: Wrapper induction for information extraction, Proceedings of the 15th International Joint Conference on Artificial Intelligence, pp. 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 the ACM SIGMOD International Conference on Management of Data, pp. 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, Vol. 43, No. 1, pp. 43-53, 2002.
- [9] S.Brin: Extracting patterns and relations from the world wide web, Proceedings of SIGMOD Workshop on Databases and the Web, pp. 172-183, 1998.
- [10] F. Ciravegna, S. Chapman, A. Dingli, Y. Wilks: Learning to Harvest Information for the Semantic Web, Proceedings of the 1st European Semantic Web Symposium, 2004.
- [11] E.Riloff, and R.Jones: Learning dictionaries for information extraction by multi-level bootstrapping, Proceedings of the 16th National Conference on Artificial Intelligence, pp. 474-479, 1999.
- [12] E.Agichtein, and L.Gravano: Snowball: Extracting relations from large plain-text collections, Proceedings of the 5th ACM International Conference on Digital Libraries, pp.85-94, 2000.
- [13] C.H.Chang, and S.C.Lui: IEPAD: Information extraction based on pattern discovery, Proceedings of the 10th international conference on World Wide Web, pp.681-688, 2001.
- [14] S.Raghavan, and H.Garcia-Molina: Crawling the hidden web, Proceedings of the 27th Very Large Databases Conference, pp.129-138, 2001.
- [15] H.He, W.Meng, C.Yu, and Z.Wu: Automatic integration of web search interfaces with WISE-Integrator, VLDB Journal, Vol.13, No.3, pp.256-273, 2004.
- [16] Z.Zhang, B.He, and K.C.Chang: Understanding web query interfaces: Best-Effort parsing with hidden syntax, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 107-118, 2004.
- [17] A.Tengli, Y.Yang, and N.Ma: Learning table extraction from examples, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp.987-993, 2004.
- [18] H.Chen, S.Tsai, and J.Tsai: Mining tables from large scale html texts. Proceedings of the 18th International Conference on Computational Linguistics, pp. 166-172, 2000.
- [19] R.Yangarber, and L.R.Grishman: Unsupervised learning of generalized names, Proceedings of the 19th International Conference on Computational Linguistics, Vol. 1, pp. 474-479, 2002.
- [20] 楠村幸貴, 土方嘉徳, 西田正吾: テンプレートの交差と DOM 構造の解析による情報抽出手法の提案, 電子情報通信学会論文誌, Vol. J90-D, No. 9, pp.2495-2509, 2007.
- [21] K. Tokunaga: Automatic discovery of attribute words from web documents, Proceedings of International Joint Conference of Natural Language Processing, pp. 106-118, 2005.
- [22] M. Yoshida, K. Torisawa, and J. Tsuji: Extracting ontologies from world wide web via HTML tables, Proceedings of the Pacific Association for Computational Linguistics, pp. 332-341, 2001.
- [23] 吉田稔, 鳥澤健太郎, 辻井潤一: 表形式からの情報抽出手法, 言語処理学会第 6 回年次大会発表論文集, pp.252-255, 2000.