DEIM Forum 2011 D2-5

Webページ中のノード間の論理的関係の発見

真鍋　知博†　　田島　敬史††

† 京都大学工学部情報学科　〒606–8501 京都府京都市左京区吉田本町
†† 京都大学大学院情報学研究科　〒606–8501 京都府京都市左京区吉田本町
E-mail: †manabe@dl.kuis.kyoto-u.ac.jp, ††tajima@i.kyoto-u.ac.jp

あらまし　Webページ内のノード間には様々な論理的関係があり，その代表的なものとして，等位関係がある．等位関係とは，同じクラスに属するインスタンスの間に成り立つ関係である．例えば，ブログの各記事のタイトルを表すノード同士や，商品検索結果ページの各商品の価格を表すノード同士は等位関係にある．本論文では，Webページ内のノード間の様々な論理的関係の発見の第一歩として，ページ内のノード間の等位関係を発見する手法を提案する．本手法では，ページ中の各ノードについて，ルートからのパス上の要素名を並べたタグパス，同じ要素名を持つ兄弟の数，内部テキストの共通するprefixなどの情報を用いることで，広汎なページからの等位関係の発見を実現する．また，発見したノード間の論理的関係の利用方法や，本研究で提案する手法の，複数ページ間の論理的関係の発見への応用などの展望についても述べる．

キーワード　情報抽出，リスト抽出，表抽出，自動集約

1. はじめに

Web上の情報量の拡大にともない，これらの大量の情報の人間による閲覧作業を支援する技術や，Web上の情報を人手をかけずに自動利用する技術が重要となりつつある．そのような技術を実現するには，Webページ中の情報の自動理解の技術が重要となる．Webページの多くは，木構造を持った構造化文書として記述されている．よって，Webページ中の情報を理解するには，ページ中の各ノード内のテキストの意味の理解と，これらのノード間の構造の意味の理解が必要である．例えば，ある同じ種類のものの一覧を掲載しているページなどで，並び順は重要でなく，必ずしも先頭から順に読む必要はないようなページが存在するが，このようなページでは，そのような論理的構造を理解できれば，閲覧の効率化などの支援につながる．

このような背景から，Webページ中のテキストや構造の意味の自動理解に関する研究が，数多く行われている．特に，テキストに関しては，自然言語処理の技術を用いて，その深い意味まで理解しようという研究も行われている．一方，ページ構造の理解に関するこれまでの研究は，画像とキャプションの対応関係の発見や，ページの本文にあたる部分の抽出など，やや表層的な特定の構造の抽出のみに特化したものが中心で，ページ構造全体の，より包括的な，より深い意味の理解を目的とするような研究は，ほとんど行われていない．

しかし，Web情報のより高度な閲覧支援や自動利用を実現するには，ページ構造の，より包括的な深い意味の自動理解の技術が重要となると考えられる．そこで，本研究では，そのような技術の実現への第一歩として，われわれが，ノード間の様々な論理的関係の中でもっとも基本となると考える「等位関係」を発見する技術を開発する．ここで,「等位関係」とは，同じクラスに属するインスタンスの間に成立する関係を指す．例えば，図1は，各種のパスタを紹介するWebページの例であるが，このページ中に現れるテキストは，ページ名，ページの説明，パスタの分類名，パスタの各分類の説明，パスタ名，各パスタの説明というクラスに分別でき，同じクラスに属するテキスト同士は等位の関係にある．同様に，blogページ内の各記事のタイトルを表すテキスト同士は等位であり，また，商品検索結果のページ中の，各商品の価格を表すテキスト同士は等位である．

テキスト間のより複雑な関係を理解するためには，まず，各テキストをこのようにクラス分けすることが基礎となると考えられる．また，このようなクラスへの分別自身にも，例えば，見出しを表すテキストのみを集めて目次を生成したり，強調部分のみを集めて索引を生成したりといった応用が考えられる．そこで，本研究では，まず等位関係の発見，すなわちページ内の全テキストノードの，等位なノードの集合への分別を行う．

構造化文書において，このようなテキストの分別にもっとも有用と考えられるのは，ルートから各ノードまでのパス上の要素名を並べたタグパスの情報である．本研究では，この情報だけでは分別し切れない構造に対処するため，さらに，同じ要素名を持った兄弟の数や，複数のテキストに共通して現れるprefixの情報，HTML特有の特徴などを利用して分別を行う．

2. 関連研究

Webからの構造を用いた情報抽出には，テキストの文構造を用いるもの，ページ間のリンク構造を用いるもの，ページ内の木構造を用いるもの，そして，これらを組み合わせたものなどがあるが，本研究が対象とするのは，ページ内の木構造である．

ページ内木構造を対象とする研究で近年盛んなのが，DBから動的に生成されるページのリスト構造から，RDBの関係の形で情報を抽出するリスト抽出の研究である．現在，Web上には，膨大な数のDBが，Webページという人間向けインタフェースで覆われた状態で存在し，この状況を，ChangらはDeep Webと呼んだ[1]．リスト抽出の研究は，この膨大なDeep Webの

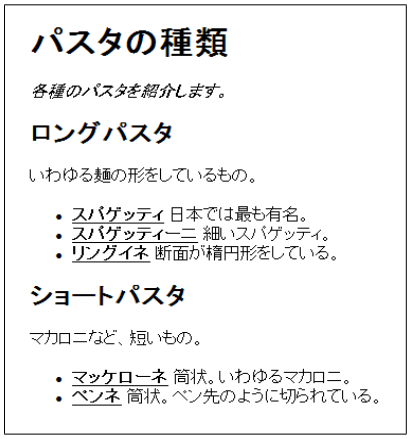


図 1　等位関係にある様々なテキストを含む Web ページの例

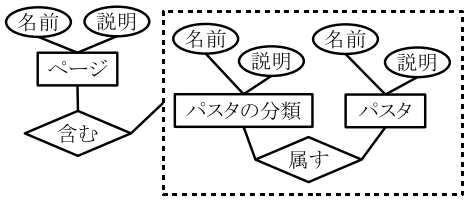


図 2　図 1 のページ中の情報の実体関連図による表現

情報を抽出し利用可能にすることを目的としている．

しかし，リスト抽出の研究は，基本的に，DB から生成された，RDB でいう第一正規形で表せるようなデータを主要コンテンツとするページを対象としている．一方，一般の Web ページは木構造，すなわち，任意の深さの入れ子構造を持つため，階層的分類を表す構造や，一つの実体が複数の属性値を持ち得る多値属性を含む構造など，第一正規形で表せないようなデータを含むものも多い．本研究では，このような，単一の第一正規形の関係では表せないような構造の抽出をも目指すという点が，従来のリスト抽出の研究とは異なる．

例えば，図 1 に示したページは，実体関連図で表せば，図 2 のような情報を含む．図 2 で破線の囲みは，実体集合と関連をまとめて一つの実体集合のように扱うための集約を表す．ここで最上位の実体はページであるが，リスト構造のみを扱う既存手法では，そのような実体は考慮しない．また，第一正規形で表せるようなフラットな表構造のみを考える場合，分類（ロングパスタ，ショートパスタ）とパスタ（スパゲッティ，ペンネ，...）のいずれをレコードにあたる単位とすべきかは明確ではない．前者をレコードとすると，それぞれに属するパスタの種類の数が異なり，後者をレコードとすると，パスタの分類が間に挟まり，いずれの場合も不完全な表構造になるためである．

一方，図 1 のテキストは，人間には容易に，ページ名，ページの説明，パスタの分類名，パスタの各分類の説明，パスタ名，各パスタの説明に分別でき，そのような分別ができれば，これらの間の階層構造等の，より複雑な関係の抽出も容易になる．そこで，本研究では，まず，このような分別を行う．

また，Lerman ら [2] は，Deep Web のサイトの多くが，以下の 3 種のページからなる共通の構造を持つことを指摘した．

- ユーザからのクエリの入力を受け付けるページ
- クエリに応じ，ヒットしたレコードをソートし，リストに整形して表示する**リストページ**
- リストページからリンクされ，個別のレコードの詳細を表示する**ディティールページ**

一部のリスト抽出の研究は，この Deep Web に典型的な構造を利用しているが，本研究では，必ずしも周辺に似た構造を持つページが存在しないような一般の Web ページをも対象とする．

また，既存のリスト抽出の手法は，まず似た構造の繰り返しを元にページをレコード単位に分割し，次に属性値を分別するというトップダウンな手法のものが多い．しかし，前述のように，階層構造を含む Web ページでは，どの単位をレコードとすべきかが曖昧なため，トップダウンな処理は不向きである．一方，本研究の手法は，まず，テキストを等位関係にある集合に分別するというボトムアップな手法になっている．

リスト抽出に関する研究の中で，本研究に近い手法を用いているものに，Miao らの，タグパスをクラスタリングすることによりリスト構造を発見する手法 [3] が挙げられる．**タグパス**とは，注目するノードについて，ルートからのパス上の各要素の名前を並べたもので，例えば次のように表される．

```
/html/body/table/tbody/tr/td/li/b/a
```

Miao らは，タグパス間の距離を定義し，この距離に基づいてクラスタリングを行うことで，リストを構成するタグパスを検出している．この距離は，注目する二つのタグパスを持つノード以外は無視するとして，互いがどれだけ規則的に出現するかにより定義されるため，間に他のタグパスが出現しても大きな影響を受けず，無関係な構造の挿入に対して頑健である．

このような方法でリスト抽出が可能なのは，ページ中のノードが，等位関係にあるグループ毎に異なるタグパスを持つことが多いためである．本研究でも，同様の仮定に基づき，ノード分別のための主要な情報としてタグパスを用いている．一方，論文 [3] と本研究の最大の相違点は，発見する構造にある．前者がレコードにあたる単位の発見を行っているのに対し，本論文では，属性の発見までを対象とし，レコードにあたる単位の発見は行わない．また，本研究では，同じタグパスを持つが等位ではないようなノードへの対応に重点を置いている．

3.　ノード間の論理的関係

本章では，本研究が扱う Web ページ中のノード間の論理的関係とは，どのようなものかについて述べる．

3.1　等 位 関 係

初めに，等位関係について述べる．本論文では，特定のクラスに属するインスタンスの任意のペアの間に成り立つ関係を，**等位関係**と定義する．等位な関係は，Web ページ中のノード間では，それを実体関連図に書き下した際に，同じ実体集合に属する実体を表すノードの間や，同じ属性に属する属性値を表すテキストノードの間などに成り立つ．例えば前述の通り，図 1 のページに含まれるテキストノードは，ページ名，ページの説明，パスタの分類名，などに人手で容易に分別できるが，この分別は，実体関連図 2 における属性と一致する．あるページ中の情報の実体関連図への書き下し方は必ずしも一意ではないが，

どのような書き下し方でも，得られる属性の集合と，それぞれに属するノードは，おおむね等しくなると考えられる．よって，本研究では，ページ中のノードを，それぞれ等位なノードからなる複数の集合へ分別し，そのような集合の各々を**属性**，ある属性に属する各要素を**属性値**と呼ぶ．本研究が考慮する他の論理的関係は，いずれも属性を最小単位として成り立つ．

3.2 実体集合を表す属性集合

RDB では，属性集合という用語は，属性の任意の集合を指す．しかし，Web ページからの情報抽出においては，任意の属性集合を考えることには意味がない．本研究では，一つの実体集合を過不足なく表す属性の集合のことを，その**実体集合を表す属性集合**と呼ぶ．例えば図 2 において，パスタの名前属性とパスタの説明属性の集合は，パスタという実体集合を過不足なく表すので，これらの属性の集合は，パスタという実体集合の属性集合である．また，ある実体が，属性集合中の各属性について持つ属性値の集合を**レコード**と呼ぶ．このように定義されるレコードは，RDB やリスト抽出の既存研究におけるレコードと，ほぼ同様のものになる．

3.3 関　　連

実体関連モデルにおいて，実体集合の間に関連が成り立つことがあるのと同様，Web ページにおいても，実体集合を表す属性集合の間に**関連**が成り立つことがある．どのような関連が存在するかは，対象とするドメインに強く依存する．例えば，レシピページであれば，料理と食材の間に「材料とする」という関連が，映画の紹介ページであれば，映画と人物の間に「出演する」という関連がそれぞれ存在する可能性がある．このように，特定のドメインにしか出現しない関連は無数に存在する．

一方，ドメインに関わらず出現する関連として，ISA 関連や HASA 関連が挙げられる．ISA 関連は，一方の実体集合が他方を汎化する場合に成立し，例えば図 1 の例では，パスタの分類はパスタを汎化する．HASA 関連は，一方の実体集合がもう一方を含んでいる場合に成立し，例では，ページが，パスタの分類や種類，その間の ISA 関連についての記述を含んでいる．

成り立つ関連の意味を機械的に推定するのは困難であるが，何らかの関連があるか否かを推定することや，ISA 関連や HASA 関連のような上下のある関連について，出現順序や木構造での上下関係の情報をもとに，実体集合間の上下関係を推定することは比較的容易であると思われる．

4. ノード分別の手法

本章では，Web ページ内のノードを属性に分別する手法を述べる．入力は，Web ページ記述言語として最も普及している HTML によって記述された，単一の Web ページとする．その対象を単一のページとするのは，一般の Web サイトの解析のためには，単一の Web ページの解析が必要になることも多いためである．例えば，記述量の少ないサイトの中には，ディティールページを設けず，単一のページに全ての実体の情報を収めているものも存在する．

以下では，まず，属性毎の分別の前にあらかじめ行っておくべき，いくつかの前処理について述べる．次に，本研究の手法の基本となる，タグパスと HTML の class 属性を用いた分別について述べ，最後に，それでは分別しきれない構造への対処法として，同じ要素名を持つ兄弟の数や，内部のテキストの prefix や，TABLE 要素を利用した更なる分別について述べる．

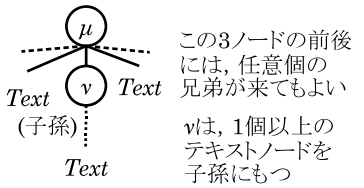


図 3　テキストを切らないと判断できるノード

4.1 前　処　理

まず，属性毎の分別の前に行う前処理について述べる．

4.1.1 DOM の構築

リスト抽出の既存手法には，ページ内の木構造を表す DOM 木の情報のみを用いるものと，それに加え，レンダリング後の位置情報を利用するものがあるが，本研究では前者のみを利用する．その理由としては，以下の三つが挙げられる．

- 計算コストが低い．
- HTML に特化せず，様々なデータ形式へ応用可能．
- HTML の要素名と class 属性を用いれば，レンダリング上のスタイルに関する情報も十分含められることが多い．

また，HTML パーザとしては，Nokogiri(注1) を使用した．

4.1.2 空白の扱い

テキストノードには，ソースのインデントなどに使われる，空白文字のみからなるものも含まれるが，それらは後述するテキストを切らないタグの検出に不都合なため，予め取り除く．ここでは，Ruby における空白文字，全角スペース(注2)，HTML で実体参照によって利用できる空白文字(注3) を空白文字とする．

4.1.3 テキストを切らないタグの扱い

HTML 文書中には，意味的に繋がっているが，挿入されたタグによりテキストノードとしては分断されている文が含まれることがある．例えば，文中リンクを含む文から，アンカーテキストの部分だけを切り出して属性とするのは，応用によっては有用だが，元の文の意味が不明になる．そのため，本研究では，このような場合には文全体を一つの属性値とすべきと考える．

そこで，本研究では図 3 に示すような構造があった場合，すなわち，あるノード μ の子らに注目した際に，テキストノード二つの間に丁度一つの非テキストノード ν があり，ν が子孫にテキストノードを一つ以上含む場合で，前後二つのテキストノードの内容が異なっている場合，ν の内部のテキストは全て，前後のテキストと意味的に繋がっていると判断する．また，ν と同じタグパスをもつ全てのノードについて同様に判断する．

対象を，前後のテキストノードの内容が異なる構造に限定するのは，図 4 に示すように，現在のページのサイト内での位置づけの表示や，記号を挟んで列挙されたリンクなど，前後のテ

(注1)：http://nokogiri.org/
(注2)：Unicode 0x3000
(注3)：同 0x00A0, 0x00AD, 0x2002, 0x2003, 0x2009, 0x200C, 0x200D

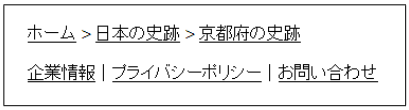
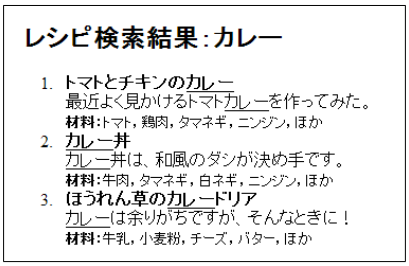


図 4　テキストのセパレータとしての利用の例

所在	店名	開店	閉店	内線
中央キャンパス	中央食堂	8:00	21:00	1234
中央キャンパス	中央喫茶	8:00	15:00	1235
北部キャンパス	北部食堂	8:00	21:00	2346
南部キャンパス	南部購買	10:00	18:00	4567

図 5　TABLE タグによる表



レシピ検索結果：カレー

1. トマトとチキンのカレー
   最近よく見かけるトマトカレーを作ってみた。
   材料：トマト，鶏肉，タマネギ，ニンジン，ほか
2. カレー丼
   カレー丼は、和風のダシが決め手です。
   材料：牛肉，タマネギ，白ネギ，ニンジン，ほか
3. ほうれん草のカレードリア
   カレーは余りがちですが、そんなときに！
   材料：牛乳，小麦粉，チーズ，バター，ほか

図 6　テキストの連続性が検出できる例

氏名	大学名	所属
郵便番号	住所	
メールアドレス		
真鍋 知博	京都大学	工学部情報学科
606-8501	京都府京都市左京区吉田本町	
manabe@dl.kuis.kyoto-u.ac.jp		
田島 敬史	京都大学	大学院情報学研究科
606-8501	京都府京都市左京区吉田本町	
tajima@i.kyoto-u.ac.jp		

図 7　TABLE タグによるリスト

```
<H1>パスタの種類</H1>
<P><I>各種のパスタを紹介します。</I></P>
<H2>ロングパスタ</H2>
<P>いわゆる麺の形をしているもの。</P>
<UL>
 <LI><B>スパゲッティ</B> 日本では最も有名。</LI>
 <LI><B>スパゲッティーニ</B> 細いスパゲッティ。</LI>
 <LI><B>リングイネ</B> 断面が楕円形をしている。</LI>
</UL>
<H2>ショートパスタ</H2>
<P>マカロニなど、短いもの。</P>
<UL>
 <LI><B>マッケローネ</B> いわゆるマカロニ。</LI>
 <LI><B>ペンネ</B> ペン先のように切られた筒状。</LI>
</UL>
```

図 8　図 1 の Web ページのソース

表 1　テキストノードのタグパスによる分別

タグパス	属性
/H1	ページ名
/P/I	ページの説明
/H2	パスタの分類名
/P	パスタの各分類の説明
/UL/LI/B	パスタの名前
/UL/LI	各パスタの説明

キストがセパレータであり，文としては切れている場合，前後のテキストノードの内容が一致することが多いためである．

図 6 に具体例を示す．例では，タイトルや説明文中の，クエリと一致した箇所が，下線タグで強調されている．ここで，三件目のレコードのタイトルや一件目のレコードの説明文において，下線タグが文の途中にあり，前述の条件を満たす．よって，同じタグパスを持つ他の下線タグも全て，文の意味を切らないと正しく判定される．一方，各レコードの材料属性値の見出しは，太字タグで強調されているが，常に文頭にあるので，テキストを切らないタグであるとは判断されない．実際，このような出現をするタグは，例のように，文の意味も切る場合が多い．

4.2　タグパスと class 属性による分別

ノードを互いに等位な集合に分別するための情報として，まず考えられるのは，タグパスである．図 8 の例では，表 1 のように，各テキストノードは，属性毎に異なったタグパスを持っている．これは恣意的な例であるが，多くの Web ページで同様の仮定が成り立つ．また，CSS のセレクタとして多用される，HTML における class 属性を，タグ名と併せて利用すれば，CSS を多用したページにも対応可能である．

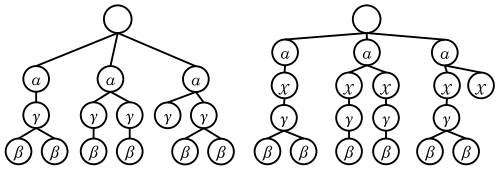


図 9　序数による分別が不適当な木構造の例

しかし，より一般には，同じ属性に属するテキストノードが，必ず同じタグパスを持つわけではない．そのような例外的な属性値を正しい属性へ分類するには，まず，どのような属性，属性集合が存在するかを知ることが重要なため，本研究では，まず，大半を占める，タグパスで分別可能な属性値を分別して，暫定的な属性や属性集合を構築し，それから例外的な属性値の分別を行うという方針を取る．この例外的な属性値の後処理による分別の部分は，今後の課題であり，本論文では扱わない．

4.3　兄弟の数と序数の利用

前節で述べた例外的な属性値とは逆に，異なる属性に属するテキストノードが同じタグパスを持つ場合もあり，こちらの方がはるかに出現頻度は高い．特に，図 5，図 7 に例を示すような，TABLE 要素を用いて記述された表やリストにおいては，複数の属性が同じタグパスを持つ例が頻出する．TABLE 要素に限らずこのような場合，人間はノードの序数を用いて属性を識別している．ここで**序数**とは，あるノードについて，現段階で同じ属性に分別されている兄弟ノードがいくつあり，その中で何番目か，という情報であると定義する．

このような場合は，TABLE 要素の例で言えば，各 TR 要素がそれぞれ同数の TD 要素を子として持つというような，特有の構造を持つ．これを一般化し，あるノードの子に，タグパス α を持つノード（以下単に α）が複数ある際，全ての α が，タグパス β を持つノード（同 β）を同数ずつ子孫に持つ場合，各 α の子孫の各 β を，序数ごとに分別する手法を考える．

ただし，ここで，各 α ごとに，各 β の親ノード γ はある一つのノード（α 自体でもよい）であり，各 α から γ までのタ

トピックス ニュース, 話題	**海外** 北米, アジア, 諸外国	**政治・経済** 政治, 経済
教養 動物, 料理, 科学, 歴史	**エンターテイメント** 芸能, 音楽, スポーツ	**ウェブ** 検索, ポータル, 文化
やってみた 歌う, 演奏, 踊る, 描く	**サブカルチャー** アニメ, 漫画, ゲーム	**その他** その他

図 10　レコードが格子状に並んだ表の例

ニンジン 色:オレンジ 栄養:カロチン	**ダイコン** 色:白	**シイタケ** 栄養:ビタミン	**サツマイモ** 色:紅 栄養:ビタミン	**ネギ** 栄養:カリウム
トマト 色:赤 栄養:カロチン	**キュウリ** 色:緑	**ゴボウ** 色:土色	**ジャガイモ** 色:土色 栄養:カリウム	**タマネギ** 色:オレンジ 栄養:アリル

図 11　明らかに prefix を持つ表の例

グパスの出現はユニークでなくてはならない．つまり，各 α から γ までのパス上では，γ 自体を除くどのノードの子にも，同じ要素名を持つ兄弟，つまり分岐が存在してはならない．この条件は，図 9 に示すような不適当な分別を防ぐために必要となる．図 9 の例では左右とも，α は二つずつの β を持つが，その木構造上の出現位置がまちまちであり，一番目と二番目の β がそれぞれ異なる属性に属すると考えるのは，不適当である．

また，上位のノードが分別されて初めて下位のノードが分別できるケースもある．図 7 に示した例では，まず，テーブルを α 兼 γ，行を β とすることで，各テーブルの行を序数毎に分別することができる．すると，各テーブルの一行目は三列ずつ，二行目は二列ずつを子孫に持つので，行を γ，列を β として，手法を再度適用することで，各列を分別することができる．

しかし，この手法を単純に適用すると，図 10 のような格子状に並んだレコードに対しても，各列毎の分別をしてしまう．これを防ぐため，各 α 全ての子孫に一度づつだけ現れる，特定のタグパスを持つテキストノードが存在するかどうかを見ることにする．例えば，図 5 の例では，各行が一つずつ強調されたテキストを含むが，図 10 の例では，そのような条件を満たすテキストが存在しない．しかし，それでも十分ではなく，そのようなタグパスがないような表も存在する．このような表については，TABLE タグの特別な扱いとして，4.5 節で述べる．

序数によって分別される構造には，最初にヘッダや凡例が現れることが多いという特徴もある．図 5 では，第一の行は列のヘッダであり，図 7 では，第一のテーブルは凡例である．これらを検出できれば，属性に**属性名**を付加するのに有用である．

4.4　prefix の利用

4.3 節にて，属性名に言及した．属性名は，ヘッダとして現れることもあるが，図 11 のように，属性値に前置され，冗長な形で現れることもある．この例では，色属性と栄養属性が同じタグパスを持ち，しかも，レコードによってそのタグパスを持つ数が異なるので，序数によって分別することも不可能である．しかし，属性名が各属性値に前置されているので，このような，共通に現れる「prefix」を検出すれば分別が可能である．

prefix の検出にあたっては，一つの実体は，同じ属性に関する記述を複数は持たないという仮定を置く．また，自然言語による説明文の先頭が偶然一致した場合などを区別するため，全ての実体のうちで，その prefix を持つ属性値を持つものの割合がある閾値を超える場合のみを考えることとする．

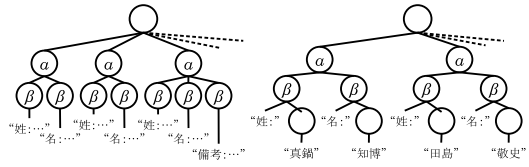


図 12　prefix による分別が可能な木構造の例

以下，タグパス α を持つノード（以下単に α ）が複数あり，各 α がタグパス β を持つノード（同 β）を子に持つが，その数が等しくない場合に，β を prefix を用いて分別する手法を説明する．具体的なアルゴリズムの概要は以下の通りである．

（1） 各 β について，含むテキストを全て繋げた文字列を作る．

（2） 各 β の文字列の先頭から一文字を切り出し，各 β を取り出した文字の一致する集合に分別する．

（3） 各集合について，各 α が，その集合に含まれる β を一つ以下しか子にもたなくなるまで，β の文字列の先頭から一文字を切り出し，集合の分割を繰り返す（いずれかの β の文字列の終端に達しても条件を満たさない場合，その集合は破棄）．

（4） 各集合について，含まれる β を子にもつ α の割合が閾値を超えるなら，その集合を独立した属性とする．

図 12 に具体例を示す．prefix による分別においては，序数を利用した分別の場合のような，全ての α に一度づつ現れるタグパスは必要ではない．なぜなら，prefix によって分別される場合，その属性そのものを，閾値を上回る割合の α が持つためである．図 12 左の例では，それぞれの β を，内部のテキストの一文字目で分別すると，姓，名，備，の三種類が得られ，この分別は属性毎の分別と一致する．内部のテキストを全て結合したものを特徴量として利用するため，右の例でも，β に対して分別が行われ，直下にある属性名と属性値の対を表すノードの各々が属性として分別される．

文書全体に対し，この手法をどのような順で適用するかという問題もある．しかし，この手法によって得られるのは，属性名が前置されたノードの集合，すなわち本論文が発見を目指す属性であり，この手法による分別が成されれば，それによって成立した新たな構造をさらに分別する必要はない．そのため，現時点で得られている全ての互いに等位な β の集合を，それぞれ親ノード α 毎の集合に分割し，それぞれの「集合の集合」に対して，上記のアルゴリズムを適用すればよい．

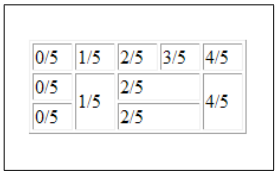
4.5　TABLE タグの利用

Web ページの中には，特定のタグパスや prefix を持たないような見出しを持つ表も多く見られるため，これまでの手法だけでは，そのような表から属性を発見することができない．一方，そのような表は，TABLE タグによって記述されていることが極めて多いという手掛かりもある．本研究では，一行（一つの TR 要素）が一件のレコード，一列が一つの属性，一セル（TR 要素の一つの子要素）が一つの属性値に対応する，TABLE タ

年始の営業時間

店舗	1日	2日	3日	4日より
今出川店	全休	12:00-18:00		通常営業
東大路店		13:00-18:00		

図 13　同内容をまとめるスパニング



0/5	1/5	2/5	3/5	4/5
0/5	1/5	2/5		4/5
0/5		2/5		

図 14　スパニングと序数

グを用いたテーブルを**正規の表**と呼ぶ．本節では，正規の表を識別して，表中の各列を互いに等位なセルの集合と見なすための，基本的な手法について述べる．

正規の表とその他の表の間を区別する有用なてがかりの一つに，一つのセル中の属性の数がある．正規の表では，一つのセルは一つの属性値だけを含むが，レイアウトのための表や，セルがレコードに対応する表では，一つのセルが複数の属性を含むか，ディティールページへのリンクを含むことが多い．また画像は，文字情報に比べ大きな情報量を持つ．これらの仮定に従い，次のいずれかの条件を満たすセルは，複数の属性に関する記述を含む**データリッチ**なセルであると判定することにする．

- 現時点までの分別によれば，二種類以上の属性値を含む
- 画像（IMG 要素）を内部に持つ
- リンク（A 要素）を内部に持つ

そして，ある TABLE 要素について，閾値を超える割合のセルがデータリッチなら，その TABLE 要素は正規の表ではないと考える．この手法によって，実際には正規の表であるものが，そうではないと判断されたとしても，そのような表のセル内には，タグの情報が十分に含まれていて，タグパスによる分別によって補完できると考えられる．一方，正規の表であると識別された TABLE 要素に対しては，各セルを表すノードに関して，4.3 節と同様に序数による分別を行う．また，内部に入力フォーム（INPUT 要素）や別の TABLE 要素を含む TABLE 要素は，明らかにレイアウト構造であるとして考慮しない．このため，TABLE 要素同士の入れ子は考慮せず，一つのドキュメント内の各 TABLE 要素に対する手法の適用順序は定めない．

ただし，TABLE 要素内のセルを序数で分別する場合，スパニングを考慮する必要がある．スパニングとは，HTML の TABLE において，複数のセルを結合し，一つの大きなセルとする機能である．スパニングが，ヘッダではなく内容のために使われる場合は主に二つあり，一つはレイアウトのため（図 7），もう一つは図 13 のように，同内容のセルをまとめるために利用される．前者については，またぐセルの数も，分別のための有効な手掛かりである．一方後者については，後の利用を考慮し，またぐセルの数だけノードをコピーし，それぞれに序数を付加するべきである．しかし，スパニングがどちらを意図しているかを識別するのは難しい．そこで，図 14 のように，結合されたスペースに対して左端のセルの序数を適用し，より右のセルへの序数の付加においてスパニングを考慮すれば，どちらの用途に対しても大きな問題を生じないと考えられる．

一方，単に序数によって正規の表の内容を分別することにも問題がある．ページ内に，同じ数の列を持つ正規の表が複数存在しても，序数が同じ列が等位であるとは限らず，列のヘッダ

Algorithm 1 $Main(N_{doc})$

```
// 入力: N_doc: DOM ツリーのルートノード
// 返値: A_result: N_doc 内のノードの属性別の子配列の配列
for all T_blank such that N_doc の子孫 and 空白文字のみ do
    T_blank を削除
end for
A_deco ← SearchDecotative(N_doc)
JoinDecotative(N_doc, A_deco)
for all N_TABLE such that N_doc の子孫の TABLE または TBODY 要素 and 子孫に TABLE，TBODY，または INPUT 要素をもたない do
    CutTABLE(N_TABLE)
end for
for all N such that N_doc の子孫 and class 属性値をもつ do
    N のタグ名に，class 属性値を含める
end for
PartbyOrdinalNo(N_doc)
for all S_tagpath such that N_doc の子孫に現れるタグパス do
    A_eqtp ← (N_doc の子孫 and S_tagpath を持つ) ノードの配列
    A_eqtp の要素を，親ノードごとの子配列に分別
    S_prefix ← 空の文字列
    PartbyPrefix(A_eqtp, S_prefix)
end for
for all N such that N_doc の子孫のノード do
    A_result << N
end for
return A_result の要素を，タグパスごとの子配列に分別した配列
```

によって等位を判断すべきである．しかし，ヘッダの検出も，それ自体が研究対象となるような問題である [4] ため，本論文では，単に序数によって分別する．この手法は，同じ列数を持つ表は全て同じ構成の列から成り，等位な列は常に，同じ列数を持つ表内の同じ序数に配される場合にのみ正しく分別を行う．

4.6　各手法の適用順序と実装

これまで，各ノードを等位な集合に分別するための手掛かりと，それを利用した分別の手法を列挙してきた．本節では，各手法の適用順序を議論する．手法の一覧は以下のようになる．

- タグパスの利用（4.2 節）
- HTML における class 属性の利用（4.2 節）
- 兄弟の数と序数の利用（4.3 節）
- prefix の利用（4.4 節）
- HTML における TABLE 要素の利用（4.5 節）

このうちタグパスの利用は，本研究における分別の基本であり，タグパスが異なるノードは等位ではないとして扱うことは，4.2 節で述べた．そこで，本研究では，タグパスに対し，他の手法によって得られた分別のためのラベルを付加していく形で実装を行った．その場合，他の各手法の適用順序によって，異なる結果が得られる場合があるため，等位な関係をより強く示唆する手掛かりを利用するものを，優先して適用するべきである．

タグパスを除けば，正規の表を表す TABLE 要素による等位関係の発見が，最も強力であると予想される．正規の表の定義により，その各列は一つの属性を表していると考えられるから

表 2　ノードのペアに対して成り立つ関係

出力＼正解	等位である	等位でない
等位である	関係 A	関係 B
等位でない	関係 C	関係 D

である．次いで強力であると考えられるのは，HTML における class 属性であり，CSS に強く依存したページにおいては，タグ名と同等に働く．残る二者については，prefix の利用が，兄弟の数と序数の利用における例外を補完する働きをするため，兄弟の数と序数の利用を優先するべきと考える．以上から，アルゴリズム全体の概要は，Algorithm1 に示すものになる．

また，これまでに述べた手法の中で，いくつかの閾値を用いている．そこで，これらの閾値の定め方について以下で述べる．

まず，序数による分別では，α にあたるノードが最低 5 つ，各 α ノードが子孫に持つ β ノードが最低 2 つあることとする．prefix による分別と正規の表による分別の際も同様とし，行数が 5 以上，列数が 2 以上の場合のみそれぞれの分別を適用する．これらの閾値は，リスト構造ではないが偶然似通った構造を持っているものを誤って分別するのを避けるため必要となる．

prefix による分別において，特定の prefix を持つレコードの割合は，その prefix を採用するかどうかを決める閾値である．統計的手法を用いることも考えられるが，文字の出現確率を加味する必要があるため，今回は単純に，実データの観察に基づき，全レコード数を A，特定の prefix を含むレコード数を B として，$B \geqq 3 + A/3$ を条件とした．

正規の表は，データリッチなセルの割合が 1/2 を超えないものとする．これは，左列をアンカー，右列を説明文とするリンク集など，一般に見られる正規の表構造に対する考察に基づく．

5. 実　　験

本章では，これまでに述べてきたアルゴリズムを実装して実際に多くの Web ページに適用した場合の，精度を測定する．

5.1 手　　法

本論文で述べてきたアルゴリズムは，Web ページを入力とし，ノードの集合の集合を出力する一種のクラスタリングだと考えられる．そこで本論文では，出力の評価に Rand Index (RI) を使用する．これは，無作為に選んだノードのペアの間の関係が，われわれの手法で正しく判別される確率にあたる．

各ノードのペアに対して，表 2 に挙げた四種類の関係のうちのいずれか一つが成り立つ．ここで，関係 X にあるペアの数を，単に X と表すと，以下の三つの値が定義できる．

$$RandIndex = \frac{A + D}{A + B + C + D}$$

$$Precision = \frac{A}{A + B} \qquad Recall = \frac{A}{A + C}$$

これらを評価の尺度とする．総合的な結果は，ページ毎の RI，Precision，Racall を，それぞれ平均して算出する．

今回の実験では，対象をテキストノードに限定した．また，空白やテキストを切らないタグの処理の部分の精度は考慮せず，それらの処理を行った後のテキストノードの集合を，人手で分別したものを正解とし，これをシステムの出力と比較した．

5.2 データセット

データセットの構築のために，Wikipedia 日本語版の「一覧の一覧」ページ(注4)を利用した．このページには，Wikipedia 内の，何らかのクラスに属するインスタンスを一覧にしているページへのリンクが，そのクラス名をアンカーとして列挙されており，それを基に，以下の手順に従ってページ集合を得た．

（1） クラス名を一つ，重複を避けて無作為に取り出す．

（2） そのクラス名からリンクされている一覧ページを閲覧し，そこに挙げられているインスタンス達の列挙を主な内容とするようなページを検索するというタスクを設定する．

（3） 上記のタスクに対応するクエリを手動で生成する．

（4） クエリを一般の Web 検索エンジン(注5)に投入し，検索結果の上位 200 件を得る．

（5） 200 件の検索結果のページの内容を上位のものから順にチェックし，上で設定したタスクに適合するページで，Wikipedia 外にあり，かつ，Wikipedia を直接の出典としないものを，最大 10 件まで抽出する．

（6） 十分なページ数が得られるまで，(1)〜(5) を繰り返す．

例えば，クラス名として「将棋類」が選ばれたとする．そのリンク先のページには，チャトランガ，チェスなど，将棋に類するゲームが列挙されており，それらを列挙するようなページの検索をタスクとする．そのようなタスクのためのクエリとしては，クラス名「将棋類」だけでは，将棋類の例の「一覧」ではないようなページが検索結果の大半を占めてしまうので，一覧，種類，などの語を加える．他にも，目録，図鑑，なども考えられるが，このクラスには不向きであると考えられる．これらの付加語は一つ含まれれば十分なので，OR 演算子で結合することにし，その結果，クラス「将棋類」に関するタスクのためのクエリとしては「将棋類 &(一覧 | 種類)」が生成される．

今回のデータセット生成においては，上記の「一覧の一覧」のページから 12 のクラスを選び，各クラス名について，検索結果の上位 200 件から最大 10 件の適合ページを抽出した結果，計 108 のページを得た．

5.3 結　　果

これら 108 ページについて，ドメイン毎の，および総合的な RI，適合率，再現率の平均を表 3 に示す．

5.4 考　　察

全体的に良好な結果を得た．特に RI について高い値を得ている．適合率と再現率については，ほとんどのドメインにおいて，適合率が上回り，集合を分割しすぎる傾向がみられた．この傾向は，等位であるが異なるタグパスを持つ例外への対処を行っていないことと，単一ページに複数の正規の表が含まれる場合，その間の属性の対応を取っていないことが挙げられる．

平均には表れていないが，ページによっては精度の著しい低下がみられた．それらのページを精査し，いくつかの共通する

(注4)：一覧の一覧-Wikipedia http://ja.wikipedia.org/wiki/一覧の一覧
(注5)：Google http://www.google.com/

表 3　ドメイン毎の結果および総合的な結果

クラス	ページ数	*RandIndex*	*Precision*	*Recall*
リーグ優勝チーム	7	0.86	0.83	0.66
史跡	10	0.96	0.91	0.84
貴族院議長	2	1.00	1.00	0.98
新書	10	0.90	0.81	0.82
野球場	10	0.94	0.82	0.78
ホテル	10	0.88	0.91	0.69
日本の色	10	0.95	0.99	0.86
オートバイの車種	10	0.88	0.82	0.75
スーパー	10	0.95	0.84	0.75
超高層ビル	10	0.93	0.94	0.75
新聞	10	0.91	0.98	0.75
岩石	9	0.84	0.66	0.67
総合	108	0.91	0.87	0.76

構造を観察した．それらについて，以下で個別に述べる．

まず，正規の表の判別に際して問題がみられた．一行が一つのレコードを表さず，複数行にまたがって単一のレコードに関する記述が行われている場合，データリッチなセルの割合が閾値を下回ることがあるが，正規の表ではない．このような場合，特定のタグパスをもつノードが，特定の行に偏って出現する．この情報も含めた，より強力な判別を考えるべきであろう．

また，prefix を用いた分別に関して，prefix を持つ属性が列挙されているが，属性名と属性値のペアを表すノードが存在しないので，属性名だけが prefix による分別の対象になり，属性値は分別されないケースがみられた．例えば，HTML において定義する用語を記述する DT 要素を使用して属性名を記述し，用語の説明を記述する DD 要素を使用して属性値を記述した場合である．何らかの prefix，すなわち属性名が発見された場合，次の prefix が現れるまでは，その属性名が指す属性に関する記述であると考えれば，改善できると考えられる．

prefix に関する別の問題として，prefix を持つ属性値が列挙されているのだが，全てのレコードの全ての属性値が，一つのノードの下に列挙されているために，分別の対象にならないページも存在する．このような場合，名前や画像など，特有のタグパスを持つ属性が存在し，各レコードに関する記述の，初めに配されていることが多い．一つのレコードについて，prefix を持つ属性値が列挙されている場合，それらの間に無関係なノードが挟まることは稀なので，異なるタグパスの出現をセパレータとしてノードの集合を分割し，その結果得られた集合の集合に対して，prefix による分割を適用することが考えられる．

6. おわりに

本論文では，単一の Web ページからの，互いに等位なノードからなる集合の発見について述べた．章 1. で述べたように，ノード間の論理的関係の発見には，様々な応用が考えられるが，多くの応用では，属性への分別に加え，実体集合を表す属性集合を特定する必要がある．そのような属性集合を発見する簡単な手法としては，ユーザに実体の名前を一つ与えてもらう方法がある．その場合，その名前を含むようなノードが存在すれば，そのノードはその実体の名前を表している可能性が高い．そして，通常，名前属性値は，実体に関する記述の最初に見出しとして現れるので，名前を表すタグパスが特定できれば，それを区切りとして，実体集合を表す属性集合が判定できる．

このような手法による属性集合の判定の簡単な応用例としては，Web サイトからの図鑑の自動生成があげられる．HTML において画像は特有の要素名を持ち，容易に識別できる．また，画像に対する自然言語による説明文は，通常，他の属性値よりも長い文字列であるという特徴がある．そこで，画像を含むページにおいて，名前の出現をセパレータとして属性集合を判定し，属性集合の中から，各画像と対応する説明文を抜き出せば，画像と説明文からなる，図鑑の一項目が得られる．

また，応用によらず，その入力として，リスト構造を持つページを得る必要がある．しかし，通常の検索エンジンにクエリを投入しても，リスト構造を持つページばかりがヒットするとは限らず非効率である．リスト構造を持つページの効率的な収集方法として，検索隠し味 [5] の利用が考えられる．検索隠し味とは，検索したいドメインに関するキーワードを自動的に推定してクエリに追加してくれることで，汎用検索エンジンを専門検索エンジンとして利用する手法である．「リスト構造を持つページ」をドメインとして，この手法を適用した場合，「一覧」「種類」などの隠し味が得られることが予想される．

他に，本研究の手法を，ページ間の論理的関係へ拡張することも考えられる．アンカーが等位関係にあるリンクの先のページは，また等位関係にあり，それぞれのページ全体を一つのノードであると考えれば，それらも等位関係にあるといえる．よって，等位なページ全てのソースを単純に繋げ，全体に対して本アルゴリズムを適用することができる．このように本研究は，Web サイト内のナビゲーションにも応用できる．

最後に，現在策定作業の進められている HTML5 について述べる．HTML5 には，論理的関係を表すタグが多く追加される．例えば，サイト内で共通のヘッダや，本文ではないコラムを明示することが可能となり，普及が進めば，情報抽出が容易になると考えられる．しかし，本研究が抽出の目標とするような実体関連を直接記述するには向かないと思われ，過去の資源の活用の観点からも，本研究と HTML5 は相補的な関係にある．

文　献

[1] K. C.-C. Chang, B. He, C. Li, M. Patel and Z. Zhang: “Structured databases on the web: Observations and implications”, SIGMOD Record, **33**, 3, pp. 61–70 (2004).

[2] K. Lerman, L. Getoor, S. Minton and C. A. Knoblock: “Using the structure of web sites for automatic segmentation of tables”, SIGMOD Conference, pp. 119–130 (2004).

[3] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires and L. E. Moser: “Extracting data records from the web using tag path clustering”, WWW, pp. 981–990 (2009).

[4] K. Tajima and K. Ohnishi: “Browsing large html tables on small screens”, UIST, pp. 259–268 (2008).

[5] 小久保, 小山, 山田, 北村, 石田：“検索隠し味を用いた専門検索エンジンの構築”, 情報処理学会論文誌, **43**, 6, pp. 1804–1813 (2002-06-15).