Web ページの表形式データからの情報抽出

執筆者：結城隆

指導教官：新納 浩幸

平成16年3月2日

目 次

図目次

表目次

第1章　はじめに

現在Web上の情報をいかにより有効に使うかが検討されている。メーカーなどの企業ではWeb上から自社や他社の製品の評価などを抽出して消費者のニーズに応え、より売れる製品を作るために利用している[3]。

ここではWeb上の表形式のデータから情報抽出を行う。表形式のデータから情報を抽出する際の最も大きな問題は、Web上の表は様々な形式で記述されており、注目する情報を抽出するには通常ページごとの規則を作らなくてはならないことである。ここでは抽出規則をWrapper Inductionを用いて自動作成することを試みる。またWeb上にある表はTABLEタグを利用して書かれていることが多いので表がTABLEタグを使った形式で書かれていることを前提とした抽出法も試みる。

第2章　処理手順

2.1　クローリング

Web上の情報量は増加の一途をたどっており、従来の汎用サーチエンジンが利用しているデータ収集手法(以降、Crawlerと呼ぶ)のように、Web上の情報をできる限り網羅的に集めて索引づけするという方式には、データ容量および収集速度の両面において限界が見え始めている。 Crawlerの収集方式を改良し、より重要なページを優先的に収集する手法の研究もされているが、これらは汎用サーチエンジンでの利用を前提とした網羅的収集を目的としており、問題の根本的解決には至っていない。

一方で、特定分野に限定した情報の収集を目的とするFocused Crawling技術が近年注目を集めている。この技術は、本年度の主な調査対象である専門分野Web検索サイト構築に必須の要素技術となりうるばかりでなく、ユーザ端末でのWeb情報巡回ソフトウェアなどにも応用可能であると考えられる。

汎用Crawlerが、ページに含まれるすべてのリンクをたどって収集を繰り返すのに対し、Focused Crawlerは、あらかじめ与えられたトピックに基づき、関連するリンク先URLを優先的に収集することが特徴である。リンクの探索優先度を決定するにあたり、トピックに対する関連度を確率モデル等に基づいて計算する手法が主流となっている(以降、トピックに対する関連度計算を行うエンジンのことをClassifierと呼ぶ)。また、優先度決定の方法には、大きく分けてContext Independentなモデルと Context Dependentなモデルが知られている。本節では、まずContext Independentなモデルを用いた例として、トピックカテゴリツリーに基づく言語生成モデルを利用したChakrabartiらの手法について、次にContext Dependentなモデルを用いた例として、 Context Graphを利用したDiligentiらの手法について紹介する。

(1) トピックカテゴリツリーを用いたFocused Crawling

(a) アルゴリズム

初期設定

Chakrabartiの手法では、 Classifierを設定するために、基準となるカテゴリーツリーの具体例を示す。図2.1の左側にフォルダ階層上に示されたものが、トピッ

クの階層を表したものである。

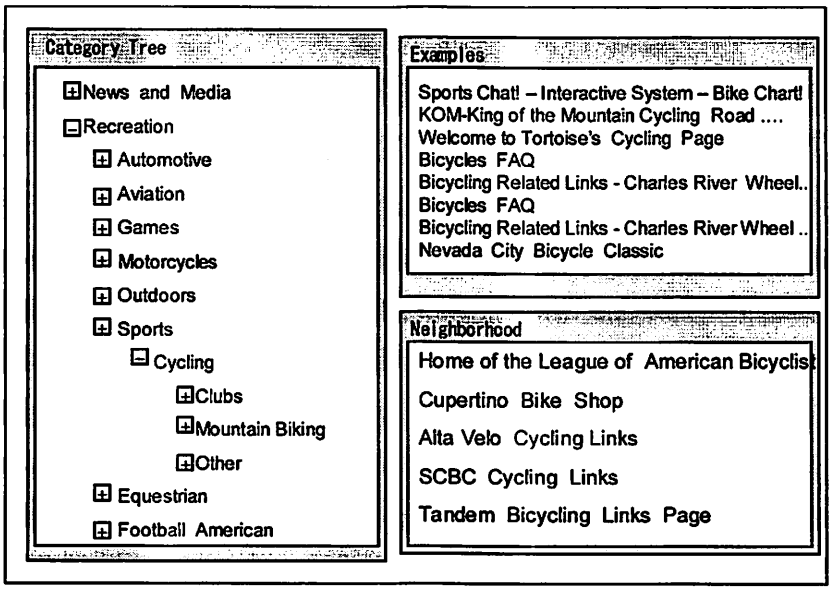

図 2.1: 基準となるトピック分類例

カテゴリーツリーとは、様々な粒度のトピックに対応するカテゴリを、Yahoo! のディレクトリ構造のように木造向上に表現したものである。各カテゴリにはそのカテゴリに分類される URL が割り付けられており、図2.1 右上の"Examples"の部分に示されている。Examples の各 URL をクリックすると、右下の"Neighborhood"の部分に、関連すると思われる URL が提示される。関連するかどうかは、例えば内容を表す単語を共通に含むかどうかで判断する。

次に、検索対象とするトピックを決める。これは、クエリの形で与えるのではなく、関連すると思われる URL を 20 ～ 30 程度用意する。以降、探索対象トピックを決めるために最初に与える URL の集合を、シーズ URL と呼ぶことにする。

カテゴリの選択と精緻化

システムは、与えられたシーズ URL を関連度の尺度の従い、既出のカテゴリー上に配置する。ユーザは、割り付けられたカテゴリの良し悪しを判定し、よいと判断したカテゴリを good としてマークする。マークされたカテゴリは図2.1 のカテゴリツリー上でハイライト表示される。

カテゴリツリーは必要に応じて詳細化することが可能であり、また、割り付けられている具体例を別のカテゴリに移動することも可能である。

以上の操作を繰り返し、カテゴリツリーを精緻化することで、Focused Crawling のための初期設定が終了する。

Focused Crawling　プロセス

処理プロセスは、各URLの関連度計算のため毎回実行されるClassifyプロセスと、有効なリンクをもつページを評価するために断続的に実行されるDistillationプロセスからなる。

Classifierは、先に設定したカテゴリーツリーに基づく言語生成確率に従い、各URLの関連度を計算する。即ち、カテゴリC^*に関する関連度をRc^*で表すと、任意のURL dについて、関連度Rは、$R_{root}(d) = 1$、および、$R_{parent}(d) = \sum_{ci} R_{ci}(d)$ {ci：c0の子カテゴリ}という条件を満たす。Focused Crawlingにおける関連度は、$R_{topic}(d)$で表される。ここでtopicとは、先の初期設定においてユーザがgoodと判定したカテゴリのことである。関連度の尺度に従いリンクを辿る方法として、以下の2通りの方法がある。

1)Hard focus rule：探索候補のURLに対して、最も関連度の高い(生成確率の最も高い)カテゴリツリー上のリーフノードを選択し、ルートノードからのパス上にgoodとマークされたカテゴリがあれば、探索候補として残す。

2)Soft focus rule：探索候補のURLに対して、goodノードに関する関連度$R_{topic}(d)$を計算し、関連度の高いものから優先的に探索する。次のリンクを辿る優先度は、現在のページの関連度とし、当該URLへのパスが複数ある場合は、最大の関連度を優先度とする。

Distillerは、集められてきたURL群の中から、各URLとURL間のリンク構造に基づき、探索対象となるURLへのリンクを多く持つHubページを見つけ出してくる。その他の優先度を決める要因として、特定のURLから同じURLに対して張られているリンクの回数や、URLを訪れる回数などを用いることができる。

(b) 紹介手法の特徴

Chakrabartiの手法のメリットをまとめると以下のようになる。

1) 負例に対するよりよい確率モデル
話題集合に含まれるか含まれないかという設定では、負例の確率分布がうまく推定できない

2) 負例を多く集めるより、正例を分類階層上に集めて必要に応じて編集するほうが利便性が高い

3) 親ノードや兄弟ノードの関連クラスの事例をもとに、必要に応じて検索候補を拡張できる。

(c) 手法の評価

評価の方法として、(i) 取得したURLの平均関連度による評価、(ii) シーズURLの別々の部分集合から出発して得たURLの重なり割合、(iii) 実際に結果を人手

でみた評価、の3つについて行っている。比較した手法は、Hard Focus rule, Soft Focus rule, Unfocus rule である。Unfocus rule とは、ランダムな順序でリンクを辿る方法である。対象分野として、ガーデニング、投資信託、サイクリング、エイズ等の各トピックについて crawl を行った。その結果、(i) では Hard Focus, Soft Focus 共に、Unfocus に対して顕著な違いが見られたが、Hard Focus と Soft Focus の差はそれほど見られなかった。(ii) についてもサイクリングについては8割以上の一致率、投資信託では6割以上の一致率を達成している。(iii) については、上位 100 URL を人手でチェックし、関連するページが得られたことを確認している。

(d) 手法の限界

紹介手法では、ターゲットの URL の探索途中に関連度の低いページが必ず含まれると、効率よくターゲットに到達できないという限界がある。

(2)Context Graph を用いた Focused Crawling

Diligenti らは、探索途中に関連度の低いページがある場合の悪影響を低減する方法として、Context Graph を用いた Focused Crawling 手法を提案している。

(a) アルゴリズム

Context Graph の構築

最初にトピックを表すドキュメントを与える。このドキュメントが含まれる層を Layer 0 とする。次に、このドキュメントにリンクをはっているドキュメントを求める (backward-crawling)。これにはいくつかの手法があるが、例えば Google のリンク逆引き機能を利用して求めることができる。こうして求められたドキュメントを Layer1 のノードとし、それぞれから Layer 0 のドキュメント (トピックを示すドキュメント) に対して一方向リンクが張られたグラフを構成する。さらに、Layer 1 の各ドキュメントから同様にしてリンク元ドキュメントを求め、Layer 2 を構成する。これを N 回繰り返すことにより、 Context Graph が構成される。図 2.2 に、N=2 の場合の Context Graph 構成例を示す。

なお、トピックを示すドキュメントを複数準備する場合には、各ページ個別に Context Graph を構成した後に単純に合成することで、 Merged Context Graph を構成する。 Merged Context Graph においてに Layer 0 に位置するドキュメント群が、Chakrabarti らの手法におけるシーズ URL に相当する。

Classufuer の学習

本手法における Classifier は、 Naive Base Classifier の考え方に基づいている。まず、あるドキュメントは、その中に出現する語または句 w_t を次元とするベクトルとして表現される。ベクトルの値は情報検索でよく用いられる TF-IDF により計算される。いま、Context Graph が Layer 0 ～ N の N+1 個のレイヤから構成さ

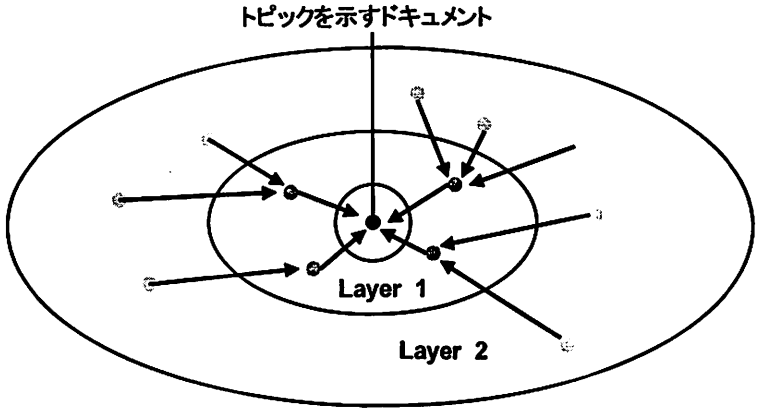

図 2.2: Context Graph の構成例

れるとすると、Classifier の処理は、ベクトルで表現されたドキュメント、N+1 個のいずれのレイヤに属するか、またはどのレイヤにも属さないかを判別することとなる。

あるドキュメント d_i がLayer jに属する確率 $P(c_j|d_i)$ は、下式により求められる。

$$P(c_j|d_i) \propto P(C_j)P(d_i|c_j) \propto P(c_j)X_{k=1}^{N_{dj}}P(w_{di,k}|c_j) \tag{2.1}$$

$w_{di,k}$ ドキュメント d_i に出現する語または句である。上式により、Context Graph 中のすべてのレイヤに対してドキュメント d_i が属する確率を求め、値が最大となるレイヤ j^* を決定する。ここで、$P(cj^*|d_i)$ があらかじめ定められたしきい値より小さい場合、ドキュメント d_i はどのレイヤにも属さないと判定され、そうでない場合にはドキュメント d_i はレイヤ j^* に属すると判定される。

以上のような動作をする Classifier を構築するためには、レイヤごとに、そこに属するドキュメント中の語 wt の、レイヤにおける出現確率 P(wt | cj) をあらかじめ計算しておく必要がある。これは Context Graph を構成するすべてのドキュメント中に出現するすべての語について、それがどのレイヤに何回出現したかを求めて正規化することで計算できる。

Crawling プロセス

Context Graph が N+1 個のレイヤから構成されるとき、Crawler は N+2 個の queue を持つ。Crawler は、まず空でない最も番号の小さい queue からドキュメントを取り出し、その中に含まれるリンクをたどって新たなドキュメントを取得する。ドキュメント取得後、Crawler は Classifier によって当該ドキュメントがどの

レイヤに属するか(あるいはいずれにも属さないか)を判定し、これにしたがって対応するqueueにドキュメントを格納する。ここで、あるドキュメントがLayer 0に属すると判定され、queue 0に格納された場合、Crawlerはその直接の親ドキュメントをqueue 1に格納するように動作させることもできる。これにより、Context Graphの再構築を行うことなしに、するように動作させることもできる。これにより、Context Graphの再構築を行うことなしに、backward-crawlingを行ってContext Graphを更新したのと同様な効果を得ることができる。

(b) 手法の評価

一般的な幅優先探索手法と、従来のFocused Crawler、およびContext Graphを用いた本手法のそれぞれで、"Call for Paper"に関連するドキュメントを収集する実験を行ったところ、本手法により収集されたドキュメントのうち、関連ドキュメントの占める割合は、幅優先探索はもとより、従来のFocused Crawlerと比較しても平均で50～60%高い値となった（ただし、ここでいう従来のFocused CrawlerとはContext GraphをLayer 0のみに制限し、収集順をNaive Baseにより計算される確率の高い順としているものである)。

(c) 手法の特徴と限界

以上のように、本手法ではContext Graphを用いることで、ある程度広い範囲のドキュメントを収集することができる。

本手法においては、以下の改良すべき点があり、今後の検討が必要である。

- トピックとなるドキュメントに対するリンク元ドキュメントが探索できない(トピックドキュメントがサーチエンジンに登録されていない、など）場合にContext Graphを構築する方式
- Classifierにおいて「どのレイヤにも属さない」と判定するしきい値の決定方法
- 収集中にContext Graphを動的に更新する仕組みの実装(現手法では擬似的に実施)

2.2 情報抽出

2.2.1 情報抽出

ここでは情報抽出(IE)の概要として、定義、歴史、一般的な処理手順及び現状での課題などをまとめる。

情報抽出(IE)はテキストから情報を選別するフィルタリング技術のひとつである抽出したい情報を予め表形式の形で用意しておき、与えられたテキストから

その表のスロットに埋めるべき情報を抽出する。

表形式のスロットに埋める情報をテキストから抽出することそれ自体はとくに新しいアイデアではない。これは対象文書を非常に限定した要約システムや文脈理解システムのひとつの実現方法となっており、過去にIEの雛形となるようなシステムは数多くあると思われる。例えば、[Cowie,1983]は植物と動物のfield-guideの記述からcanonicalな構造を抽出している。また[Dejong,1982]はAPニュースからSchankのスクリプトのスロットを埋めるFRUMPというシステムを開発している。[Hayes el al,1992]は商用的な最初のIEシステムと言われるJAS PERを開発した。日本でも[小松ほか,1987]らは要約システムとうたってはいるが、設定しているタスク(新聞記事から新製品情報の抽出)が現在のIEと同じである。ただしこれらの研究はtoy systemの域を出ていない。

また分野を限定しなくとも制限言語に対して、表形式のスロットに埋める情報をテキストから抽出することの可能性は1950年代にZelling Harrisによって既に提案されている。九工大の「法律文制限言語モデルとそれに基づく法的文章の計算機処理」の一連の研究もIEに通じているだろう。

IEについての共通の認識や手法は、 Mesage Understanding Conference(MUC)と呼ばれる一連のコンテストを通してここ10年程度で確率されてきたといえる。IEに対する共通の過程は、概略、分野と抽出する項目(表のスロットに入れる情報のタイプ)を限定し、表層的な解析結果と表層的な知識を用いて表形式のスロットを埋めることであろう。この観点現在のIEと同じ手法を用いた最も初期の研究は[Sager,1981][Sager et al,1987]による医学領域のテキストからの情報抽出であり、そこでは表層的な解析とテンプレートが利用されている。

IEは文書理解技術としても捉えることができる。テキスト中のすべての情報を理解しようとする完全な文書理解は現在でも不可能であるが、IEは文書理解を制限することで現在の技術でも達成できそうなタスクを設定しているといえる。

IEの一般的処理は2つのステップに分けられる。第1ステップではテキストから個々の事実を集め、第2ステップでは第1のステップで集めた事実を利用して、より大きな事実や新しい事実を生成し、最終的に表のスロットに入る事実を得る。第1ステップで集められる事実はパターンマッチの処理によって得られる。ただし自然言語の複雑性から単語の列を直接利用したパターンは現実的ではなく、まず入力テキストを解析して構造化する。この解析には形態素解析や構文解析やparatial parsingや固有名詞抽出処理(NE)が利用される。これらの処理が終わった後に構造化された表現に対してパターンを適用し、その処理によって事実を集める。第2のステップでは、まず固有名詞などの参照を解消し、その後に事実間の関係を推

論する。そして最終的に表のスロットに入る事実を得る。MUC では、抽出すべきイベントやその関係をシナリオ (scenario) と予備、目的の表のことをテンプレート (template) と呼んでいる。

IE の特徴は通常の文書理解にあたる部分を正規表現のパターンマッチの処理によって行っている点である。前述したシナリオのパターンマッチの処理がこれに当たる。例えば、

<Oraganization> hired <Person>

というパターンから、「<Organization>が<Person>を雇った」という情報を得る。通常は、上記の個々の単語の部分は構造化されているので、文字列のパターンマッチではなく、名詞句の次に hire をヘッドとする動詞句が現れる等のパターンマッチである。

名詞句などの部分的な構造を得るために構文解析は行われているが、この構文解析をどの程度行うかが IE の差別化になっている。上記の例で言えば、構文解析によって "hired" の主語と目的語が分かれば、「<Oraganization>が<Person>を雇った」という情報が得られるために、パターンは必要ないとも言える。事実 MUC での初期の研究では構文解析を行うシステムが主流であった。ただし現在の IE の有力なシステムはほぼすべてパターンを利用した手法をとっている。これは現在の構文解析の制度が悪い点、抽出に不必要な部分も解析する無駄がある点の２点の理由からである。

ただし構文解析を行わない欠点もある。構文解析を行うメリットはその出力が正規化されている点である。例えば、以下の分の部分はすべて同じ構造を出力するだろう。

IBM hired Harry.
Harry was hired by IBM.
IBM, which hired Harry, ...
Harry, who was hired by IBM, ...
Harry, hired by HBM, ...

パターン処理では上記の文に対するそれぞれのパターンを記述する必要がある。この問題に対して、 SRI の FASTUS やニューヨーク大学の Proteus は、metarule や schemata という手法をとっている。これは基本となるパターンからそのバリエーションに当たるパターンを自動生成するものである。これは GPSG のメタルールに似ている。

現状の IE は高精度の構文解析が不可能という仮定のもとで、パターンを利用している。構文解析の制度がよくなれば当然、パターン処理の負荷が減り、結果的

に精度向上に繋がる。近年、括弧付けされたコーパスから文法を得る研究が盛んであり、そこでは人間が作成した文法と同等以上の性能を出すようになっている。しかも年々精度は良くなっているこの分野での発展も IE の精度向上に貢献するはずである。

最後に IE の課題を述べる。 IE の問題はその移植性のコストの高さにある。新しい分野に IE を移植しようとすると、その対象分野や目的にあったパターンを新たに作成しなければならず、そのコストが大きい。

一つの解決策が学習手法を利用したパターンの自動獲得である。この技術動向については次節で述べる。ただし学習手法を利用する場合、十分な学習データを用意できないという現実的な問題がある。このために対話的ツールが別解として提案されている。ユーザは例文と抽出すべき事実をシステムに与える。システムは保持しているパターンを利用して例文を解析する。次にシステムはユーザにそのパターンを構造的にあるいは意味的に一般化したパターンを提示する。拡張するにはメタルールを用い、一般化するにはシソーラスを使う。ユーザは過度の一般化がなされていないかをチェックする。また学習データが不十分という問題に対して、学習からのアプローチではタグなしコーパスからの学習が提案されている [Riloff, 1996]。

2.2.2 情報抽出のための抽出規則の学習

IE を新たなドメインに移植する場合に最も困難な部分はパターンの作成と更新であり、 IE ではこの点が重要な問題となっている。この問題の 1 つの解決方法はパターンの自動獲得である。ほんせつではパターンの学習手法に関してまとめる。

パターンの学習を考える場合、情報抽出の対象文書がどれくらい構造化されているかによってその難しさは異なる。例えば、新聞の 2,3 行のアパートの賃貸広告は、文章などなく、部屋の間取り、設備、最寄り駅、家賃などが単に羅列されているだけである。あるいは医療情報センターからの退院サマリや新聞のテレビ欄などもこの類だろう。このような文書 (構造化文書とここでは呼ぶ) はある種の構造が認められ、パターンを作成するのは比較的容易である。対極に位置するのが、自由文書である。これは MUC で対象としたような新聞記事や Web 上のページである。このような文書から有効なパターンを見つけることは難しい。また構造化文書としては、ネットニュースで流れる会議の案内や求人情報などである。これらはゆるいある種の構造をもっている。

パターンの学習を論じる場合、対象文書がどの程度構造化されているかという観点の他に、取り出そうとするパターンにスロットが 1 つしかないのか、複数存

在するのかという観点も重要である。文書の中には注目しているイベントが1つしかない場合には、スロットは1つで間にあう場合が多い。例えば、ネットニュースで流れる会議の案内はあるひとつの会議の案内であり、時間表現は目的の会議の開催時間と推測できる。[佐藤 and 佐藤, 1996] のネットニュース fj.wanted からの要約作成もこの分類だろう (ただしこの研究は取り出す情報のタイプが多岐にわたるので、IE よりも要約の方に近い。) 一方、新製品とその発売会社を抽出するシステムにおいて「A社とB社は商品αと商品βをそれぞれ同時に発表した」という文から、会社は「A社」と「B社」、商品は「商品α」、「B社」が「商品β」という具合に、スロットは複数あり、しかも「A社」が「商品α」、「B社」が「商品β」という対応関係まで作るようなパターンを作成しなくてはならない。

以上、2つの観点からパターンの学習を分類すると理屈上は6つの場合が考えられるが、パターンの自動獲得に関する過去の研究では以下の4つのパターンが行われている。

(a) 構造化文書×複数スロット

インターネット上には多くの relationl なデータが埋まっている。例えば、名前とその人の email アドレスや、国名とその国の電話の国番号などである。これらデータのソースは HTML で記述されている。また、それらはここでいうところの構造化文書の範疇に入る。この Web 文書から relational な関係を抽出する、つまり、この Web 文書を relational な形式に変換するプログラムを一般に Wrapper と呼んでいる。[Kushmerick et al, 1997] はこの wrapper を学習するシステムを作成した。対象とする Web 文書は HLRT という規則を満たしている必要がある。基本的に学習するのは抽出する部分を区切る記号 (主に HTML のタグ) である。例を一般化することで wrapper を学習し、PAC モデルによって作成された Wrapper で十分かどうかを評価することで学習を終了させる。ここでの学習システムはMUC の NE の学習システムと似ている点を注記している。

[Ashish and Knoblock, 1997] の研究は [Kushmerick et al,1997] と同様のアプローチをとっている。ただし上記の HLRT という条件よりもよりゆるい条件の Web 文書を対象にできる。その反面、HTML に特化したヒューリスティクスを使っている点、人間の介在を必要とする点を欠点としてもつ。

(b) 準構造化文書×単数スロット

「準構造化文書×単数スロット」のパターンはWeb上の特定の文書の類がこれに当たる。[Soderland, 1997] は天気予報を記した文書から地名と天気の relational な関係を抽出する IE を作成している。そこでは後述する CRYSTAL が利用されている。

[Califf and Mooney, 1997] は「準構造体×単数スロット」を対象にRAPIERという抽出規則の学習システムを作成した。RAPIERが獲得する規則は、スロットに入る部分のパターン、そのパターンの直前のパターンと直後のパターンの三つ組である。パターンの部分は、単語自身や品詞、意味マーカなどの条件である。学習はbottom-upであり、例から最も特定的なパターンを作りそれを一般化することによって行われる。実験はコンピュータ関連の求人情報を行っており、100文書からの学習で適合率83.7%、再現率で53.1%であり、他の学習システムと同程度である。

[Freitag, 1998] ではここでいうところの「準構造化文書」を“messy text”と呼んでいる。そして“messy text”に対してパターンを学習するアプローチを考えた場合、あらゆるタイプの“messy text”からのIEに対して有効であるような単一の学習アプローチは存在しないと推測している。そこで [Freitag, 1998] は、rote learning, term space learning, relationallearning　という3つ学習アプローチを紹介し、それらを組み合わせたmultistrategyのアプローチを提案している。組み合わせにはmeta-learningの手法を用いている。実験では単一の各学習手法と組み合わせたここでの学習手法から得られたシステムのF-値によって比較している。

(c) 自由文書×単数スロット

[Lehnert et al, 1992] は　MUC-4の課題に対してCIRCUSというシステムを開発した。そこではselective concept extractionと呼ばれるアプローチをとっている。このアプローチはパターンマッチの一種であり、関係するテキスト部分を選択し、それ以外の部分を読み飛ばすようなアプローチである。CIRCUSの核になるのはconcept nodeと呼ばれる分野依存の辞書である。concept nodeはある単語(実質的には句)によってトリガされるcase frameである。さらにconcept nodeには抽出すべき情報のスロットも持っている。このスロットは1つである。概略、ある規定の単語が現れたら、ある条件を調べて、条件が満たされたら、スロットに入る情報を出力するという形である。CIRCUSは与えられた文からconcept nodeの集合を出力し、concept nodeにより抽出すべき情報を出力する。concept nodeは一種の抽出規則やパターンと見なすことができる。[Riloff,1993] はこのconcept nodeを自動獲得するシステムAutoSlogを開発した。例からの学習であるが、いくつかのヒューリスティクスを使ってトリガの対象となる単語を見つけている。ただし帰納学習を利用せずに、生成されたconcept nodeの採否は人手により決めている。実験では手作業で作成したconcept nodeとの比較を行っている。結果は同程度のパフォーマンスを出している。

(d) 自由文書×複数スロット

前述した AutoSlong を改良したのが [Soderland et al,1995] の CRYSTAL である。 AutoSlong の母体システムは CIRCUS であったが、 CRYSTAL の母体システムは BADGER と名づけられている。CRYSTAL も AutoSlog 同様、 concept node を学習する。しかし CRYSTAL の concept node は複数のスロットを持つ。このために「複数スロット」に対応している。また帰納学習も採り入れられている。基本的には bottom-up の学習であり、特化した concept node から類似のものを一般化する。

[Soderland, 1999] は様々な IE のタイプに対するという意味で汎用的なパターンの学習システム WHISK を開発した。WHISK 「自由文書×複数スロット」以外にも様々なタイプの IE に対応することができる。WHISK は教師付き学習の一種である。まず最初にタグ付けされていないデータとタグ付けされているデータを分離する。WHISK はタグつけされていないデータからタグを付けるべきデータを選ぶ。ユーザはそれらにタグ付けし、先のタグ付けされているデータと合わせて学習データとする。この学習データが現在もっているパターン集によってカバーされない場合に、その学習データからパターンを作成し、パターン集に追加する。タグを付けるべきデータを選ぶ際に selective sampling の手法を用いる。学習データの追加の終了は PAC 学習の手法を用いる。

[Kim and Moldovan, 1995] は MUC-4 へのアプローチのために、パターンを学習する PALKA という名のシステムを開発した。PALKA のパターン表現形式のために FP-structure と呼ばれる形式を提案している。学習データから FP-structure を作成し、得られたパターンは意味的な制約から一般化される。この一般化のステップに帰納学習手法が使われている。 PALKA は AutoSlog と似ている。競合する仮説が手作業でつけられた回答やユーザへの問い合わせで解決されている。

[Riloff, 1996] はパターンを学習する場合にコーパスにタグを付ける作業の負荷が高いことを考慮し、プレーンなコーパスからパターンを獲得する AutoSlog-TS というシステムを開発した。これは [Riloff, 1993] で作成した Autoslog の改良版であり、 AutoSlog-TS はタグつきコーパスの代わりに、対象ドメインと関係ある文書かない文書かを予め分類した文書集合を利用する。AutoSlog-TS は 2 段階の処理を行う。第 1 段階では、文を解析し、１５個のヒューリスティクスから名詞句を抽出するパターン (concept node) を作成する。ここで AutoSlogu は 1 つのパターンしか作らないが、AutoSlog-TS では複数のパターンを作成する。第 2 段階では取り出したパターンをトレーニングコーパスに施し、それがどれくらい関連するかを以下の式 (relevance rate と呼んでいる) により測る。

Pr(relevant text—text contains pattern_{i})

$$= \frac{rel - freq\_\{i\}}{pattern\_\{i\}} \tag{2.2}$$

rel-freq_{i} は pattern_{i} が関連ある文書でアクティブになった回数、total-freq_{i} はpattern_{i} がすべての文書でアクティブになった回数である。次にrelevance rate に log(freq) (logの底は2) をかけた値によって、パターンを重要度順にソートし、上位のパターンから人間の判断によりその採否を決定する。実験では最終的にパターンを得るまでの時間が AutoSlog より短縮されていることを示した。また作成できたパターンのパフォーマンスは AutoSlog よりわずかによかった。

2.3 html形式への変換

抽出した情報をブラウザで一括して表示する。図2.3のように情報を表形式で一括して表示し判断しやすいようにする。

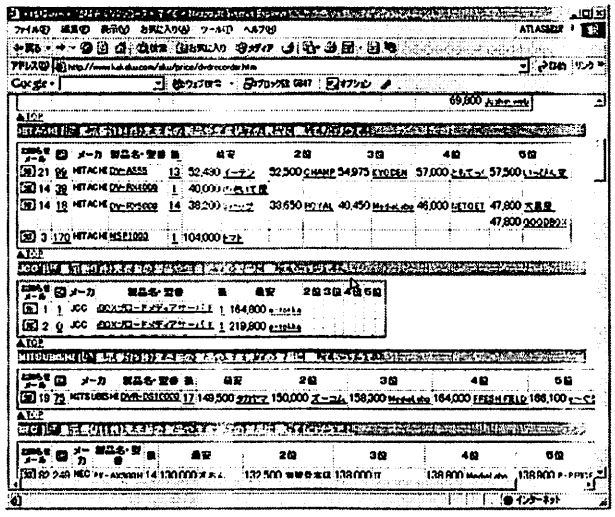

図 2.3: 情報比較サイトの例

第3章　Wrapper Induction

3.1　wrapper Induction

Wrapperとは、Webのページからある種の情報を抽出するプログラムに対応するものである。そのWrapperを帰納学習する技術がWrapper Inductionであり、概略述べれば、ページとそのページ内の抽出対象の対の集合を与えて、ページから目的の情報を抽出する規則である。

対象となるのは主に表タイプのWebページである。そこから表の項目を抽出する規則を自動構築する。例えば、以下はチーム名とそのチームの1STステージの順位が書かれているWebページのソース例である。

表3.1: Webページのソース例1

```
<HTML><TITLE> チーム名と順位</TITLE><BODY>
<B>横浜FM</B><I>1</I><BR>
<B>磐田</B><I>2</I><BR>
<B>市原</B><I>3</I><BR>
<B>鹿島</B><I>8</I><BR>
</BODY></HTML>
```

このページから以下のようなチーム名とチーム番号のペアを抽出することを考える。

{ <横浜FM,1>,<磐田,2>,<市原,3>,<鹿島,8> }

抽出するプログラムとしては表2のようなものがあればよい。このプログラムのポイントはl_kとr_kの組部分、すなわち{(<B>,</B>),(<I>,</I>)}である。直感的には、抽出項目(チーム名や順位)を挟む左右のタグがl_kとr_kである。表2のプログラムを自動構築するとは、この組の集合を学習することに他ならない。このようなWrapperをLR Wrapper、そしてLR Wrapperにより目的とする抽出項目を抽出できるページの集合をLR Wrapper classと呼んでいる。

LR Wrapper classのページPとPから抽出すべき項目Lの対<P,L>の集合を入

表 3.2: 抽出するプログラム

```
function LR wrapper(pageP){
  while there are more occurences in P of l_1{
   for each (l_k,r_k)in{(<B>,</B>),(<I>,</I>)}{
     scan in P to next occurence of l_k
     save position as starat of kth attribute
     scan in P to next occurence of r_k
     save position as end of kth attribute
    }
   }
   return extracted {···,<team,ranking>,···}pairs
}
```

力として<l_k,r_k>の組を学習するアルゴリズムが Wrapper Induction である。この学習アルゴリズムは l_k と r_k に関するある制約充足問題に変形でき、この問題を解くことで目的の l_k と r_k、つまり LR Wrapper が得られる。

ここで、現実に対象とする Web ページは都合よく LR Wrapper class に属しているのかどうかという疑問が当然起こる。後述する実験では、実際のサイトの53%がこのクラスに属していると予想している。このカバー率を上げるために、LR Wrapper class 以外に、HLRT Wrapper class 、 OCLR Wrapper class、HOCLRT Wrapper class、N-LR Wrapper class 及び N-HLRT Wrapper class の 5 つのクラスとそのクラスに対する Wrapper の学習手法を提案している。ここでは学習手法の説明は省略し、それぞれがどのようなクラスかだけを示す。

(a)HLRT Wrapper class

最初に例で出したチーム名と順位の HTML のソースが以下のようにないっていたとする。

表 3.3: Web ページのソース例 2

```
<HTML><TITLE> チーム名と順位</TITLE><BODY>
<B>チーム名と順位<B/><P>
<B>横浜 FM</B><I>1</I><BR>
<B>磐田</B><I>2</I><BR>
<B>市原</B><I>3</I><BR>
<B>鹿島</B><I>8</I><BR>
<HR><B>終わり</B></BODY></HTML>
```

この場合、LR Wrapperではうまく抽出できない。それは国名を取り出す左右の文字列 (<B>,</B>) が繰り返しの始まり (Head) にある「チーム名と順位」や繰り返しの終わり (Tail) にある「終わり」を国名として取り出してしまうからである。この不具合に対処するために、LR の規則の他に Head と Tail の部分を認識するように拡張したのが HLRT Wrapper class である。上記の例の場合、学習される規則は、Head の終わりの文字列 <P>と Tail の始まりの文字列 <HR>が LR に追加された形になり、{<P>,<B>,</B>,<I>,</I>,<HR>}となる。

(b)OCLR Wrapper class

最初に例に出したチーム名と順位番号の HTML のソースが以下のようになっていたとする。

表 3.4: Web ページのソース例 3

```
<HTML><TITLE> チーム名と順位</TITLE><BODY>
<B>チーム名と順位<B/><P><UL>
<B>横浜 FM</B><I>1</I><BR>
<B>磐田</B><I>2</I><BR>
<B>市原</B><I>3</I><BR>
<B>鹿島</B><I>8</I><BR>
</UL><HR><B>終わり</B></BODY></HTML>
```

この場合も、HLRT のときと同様、LR Wrapper ではうまくいかない。 ここでは、 LR の規則を含む始まり (OPEN) の文字列と終わり (Close) の文字列を取り出すことで対処する。具体的には、<LI>と
 で LR の部分がはさまれている。上記例の場合、学習される規則は、{<LI>,
,<B>,</B>,<I>,</I>} となる。

(c)HOCLRT Wrapper class 最初に例に出したチーム名と順位番号の HTML のソースが以下のようになっていたとする。

表 3.5: Web ページのソース例 4

```
<HTML><TITLE> チーム名と順位</TITLE><BODY>
<B>チーム名と順位<B/><P><UL>
<B>横浜 FM</B><I>1</I><BR>
<B>磐田</B><I>2</I><BR>
<B>市原</B><I>3</I><BR>
<B>鹿島</B><I>8</I><BR>
</UL><HR><B>終わり</B></BODY></HTML>
```

この例は HLRT と OCLR の両方のクラスに属している。このようなタイプ文書に対しては、LR の他に Head、 Open、 Close、 End を学習する。上記例の場合、学習される規則は、 {<P>,<LI>,
,<B>,</B>,<I>,</I>,<HR>} となる。

(d)N-LR Wrapper class

LR、 HLRT、 OCLR、 HOCLRT は対象として HTML 文書を想定している一方、N-LR は通常のテキストを想定している。例えば以下のようなテキストを考える。

表 3.6: Web ページのソース例 5

```
name : 鈴木
        address : 東京都
            phone :  123-4567
            phone :  444-5555
name : 田中
        address : 千葉県
            phone :  666-7777
name : 佐藤
name : 山田
        address : 埼玉県
        address : 栃木県
            phone :  888-9999
            phone :  222-1111
```

これは通常のテキストであるが、改行やインデントが視覚的に表を構成しており、表が埋め込まれていると考えられる。この表の項目の値を抽出するのも LR で実現できる。例えば住所に対応する値は、‘ address : ’と ‘ \n ’ (改行) で囲まれている。上記のような文書のタイプを N-LR Wrapper class と呼んでいる。LR Wrapper class はほぼ N-LR Wrapper class に含まれている。

(e)N-HLR Wrapper class

LR のクラスに対して、 HLRT のクラスがあったように、 N-LR のクラスに対しても N-HLRT クラスが存在する。 これは N-LR のクラスの例に、 Head と Tail が付いたものである。

以上この論文では 6 つの文書のクラスとそのクラスに対する Wrapper の学習手法を提案しているが、さらにこれらクラスが現実にカバーできる文書の割合と各クラスの学習効率についても論じている。

カバー率に対しては、以下のような実験を行っている。www.serch.com からラ

ンダムに30サイトの検索エンジンを選ぶ。これらはある分野専用の検索エンジンとなっている。各検索エンジンに対して、適当なクエリを与え、検索されたページを集める。それらのページのうち10ページに抽出項目のラベルをつき各クラスのWrapperの学習を行い、100%の正解率で目的の情報を抽出できる Wrapperが学習可能かどうかを調べる。もし学習可能であれば、最初の検索エンジンが対象とする分野に関してはWrapperの学習が可能だとする。この学習可能な分野の割合を調べる。結果、LRは53%、OCLRは53%、HOCLRTは57%、N-LRは13%、N-HLRTは50%のカバー率になった。6つのクラスが学習可能な分野の和集合をとると70%となった。N-LRやN-HLRTの成績はよくないが、他の4つではカバーできない文書の25%がこれらでカバーされている。また理論的に6つの文書クラスの包含関係も与えているが、単純な包含関係になるものはほとんどなく、各文書クラスの関係は複雑である。

次に学習の効率に対しては、各クラスの学習に必要とされる事例の数や学習時間について、実際の実験と理論的解析の二面から述べている。ここでは理論的解析の結果だけを紹介する。そこでは、PAC学習理論を用いて必要とされる事例数について以下の結果を得ている。

・LR と N-LR については $1/\varepsilon \cdot (2KlnR - ln\delta)$

・HLRT,OCLR,N-HLRTについては $1/\varepsilon \cdot ((2K+2)lnR + ln(R^2-2R+1) - ln(4\delta))$

・HOCLRTについては $1/\varepsilon \cdot ((2K+4)lnR + ln(R^4-4R^3+6R^2-4R+1) - ln(16\delta))$

ここでKは抽出すべき `l_i` と `r_i` の組の数(最初のチーム名とチーム順位の例では2)、Rは訓練データのページの大きさの中で最小のものの値を示す。そしてエラー率εを確率$1-\delta$で達成するのに必要な最低事例数が示した値である。また学習時間に関しては以下の結果を得ている。

・LRについては $O(KM^2|\varepsilon|^2V^2)$

・HLRTについては $O(KM\ 2|\varepsilon|^4V^6)$

・OCLRについては $O(KM\ 4|\varepsilon|^2V^6)$

・HOCLRTについては $O(KM\ 4|\varepsilon|^4V^{10})$

・N-LRについては $O(KM\ 2K|\varepsilon|^{2K+1}V^{2K+2})$

・N-HLRについては $O(KM\ 2K+2|\varepsilon|^{2K+3}V^{2K+4})$

ここでVは訓練データのページの大きさの最大のものの値を示す。Mは訓練データのラベルの大きさの総数を示す。

Wrapperの帰納学習で重要な点は、できるだけカバー率をあげられるクラスとその上での学習手法を考案することであろう。ここでの学習では、あるページで

項目が欠損するような場合や、他のタグで表現される場合などに対応できない問題が指摘されている。また訓練データを作成するためにラベルを付けるコストが高いことも問題としてある。また Wrapper は情報抽出の抽出パターンの一種とも捉えられる。そのため、 Wrapper Induction は情報抽出における抽出パターンの自動獲得に関する研究とも関連がある。

3.2 Wrapper Induction の比較

複数の Web サイトから情報を取得してデータベースに格納する処理には、当初サイトごとに人手で作成された Wrapper が用いられていた。このような最初の Wrapper 構築フレームワークが TSIMMIS である。これは、複数の情報源にアクセスして情報を取得し、それらに矛盾がないかを検証するためのツールを提供するものである。

一方、情報量が多く、構造が動的に変化する Web サイトにおいては、より効率的な Wrapper の構築が求められている。 Database　分野では異種情報の統合方法が中心議題であり、AI 分野では機械学習を Web サイトの学習に利用する方法が中心となった。ここでは論文 [2] を元に、後者について述べる。

1) では、構造化された Web ページの Wrapper とその生成について述べる。これらのシステムはおもに Wrapper 生成分野において提唱されたものである。2) では、構造化されていない Web ページにかんする技術について述べる。こちらのシステムは、主に IE 分野から提案されたものである。

1)Structured and semistructured Web pages

この説では、Wrapper 生成システムとして、ShopBot, WIEN, SoftMealy, STALKER について説明する。これらは、サーバーに送ったクエリへの応答として生成された Web ページからの情報抽出が主な応用である。重荷、区切り文字を利用した抽出パタン (delimiter-based extraction patterns) を利用し、文法解析は行っていない。また、構造化されたデータのみに対して動作することができる。

(a)ShopBot

ShopBot は、比較ショッピングエージェントであり、フォームに入力したクエリに基づいて生成された商品一覧画面からの情報抽出を目的としている。抽出処理は、ヒューリスティック検索とパタンマッチ、そして、帰納学習の組み合わせである。

ShopBot の動作は、オフラインの学習フェーズと、オンラインの比較ショッピングフェーズからなる。学習フェーズではヒューリスティックを用いて、クエリ入力

フォームとクエリの与え方を推定し、次に検索結果として得られたページのフォーマットを推定する。また、検索結果のWebページはheader、body、tailから構成され、 header部分とtail部分は異なるWebページの間で共通であると仮定する。検索結果Webページは、以下のステップにより推定される。

・存在してない製品キーワードを与えて、検索できなかった場合のテンプレートを学習する。

・次に、いくつか既存製品のキーワードを与えて、headとtailを推定する。

・最後に、body部分のフォーマットを解析する。タグとテキストの例として定義されたフォーマット仮説をいくつか用意する。ボディ部分を理論的に区切り(
等で区切る)、各仮説と比較して、最も一致するものを選択する。

以上の手順でフォーマットを解析する。ただし、「価格」といった情報スロットのラベルは導けないので「価格」だけは特別にハンドコードされたアルゴリズムで抽出する。

(b)WIEN

WIEN(Wrapper Induction Environment)は、Wrapper構築支援ツールである。WIENはShopBotに強く影響されたものであるが、 term wrapper inductionが初めて導入したものであり、このアルゴリズムでは、Webページの構成として、Head区切り文字(delimiter)と、抽出情報両側にあるRight区切り文字及びLeft区切り文字、そしてTail区切り文字からなるものを対象としている(この構成を論文ではHLRTと呼んでいる)。まず、スロットの前後を決める区切り文字を探し、次に表部分と周りのテキスト部分とを分ける区切り文字を探す。

WIENは抽出譲歩の前と後ろを区切る区切り文字のみを用いる。またいくつかの項目が抜けていたり、項目の出現順序が変則的な文書には適用できない。

一方、マルチスロットルールを用いることができるので、複数の項目を用いて特定しなければならない情報も抽出できる(例:住所録の表において、特定の個人の住所を抽出する際)。

(c)SoftMealy

WIENの登場後に、 Wrapper Inductionを改良したいくつかのシステムが提案された。SoftMealyは、半構造化されたWebページから情報を抽出するWrapperを、非決定性有限オートマトンとして学習する。

機能生成アルゴリズムは、訓練例から、文法ルールを生成するのに用いられる。訓練例は抽出情報とそれらを区切るセパレータからなるリストである。Wrapper inductionは、セパレータの場所と抽出情報とそれらを区切るセパレータからなるリストである。 Wrapper inductionは、セパレータの場所と抽出情報の出現順序

を得るための手がかりとなるラベルが付けられた行を入力として受け取る。このラベルによる場所情報を用いて文脈ルールを生成する。

生成されたWrapperは非決定性オートマトンであり、各状態は抽出する情報を表し、状態遷移は抽出情報の間にあるセパレータを定義した文法ルールを表す。情報抽出処理時には、Wrapperは抽出情報の周りにあるセパレータを利用する。

SoftMealyは、ワイルドカードが利用可能であり、抜けている項目(Missing items)にも対応可能である。また、抽出項目の出現順序に依存しない抽出が可能であるが、可能な出現順序全てをカバーする訓練例が必要である。SoftMealyの抽出パタンは、WIENより表現力が高い。

(d)STALKER

STALKERは情報抽出ルールの導出のための学習アルゴリズムである。ユーザは幾つかのサンプルページとそれに関連したデータの組を訓練例としてシステムに与える。システムはページからトークン列とそのインデックス列を作成する。これらのトークン、およびワイルドカードの列は、抽出情報の位置を特定するためのランドマークとして使用される。

STALKERは順列変換アルゴリズムを使用する。始めに可能な限り多くの静の訓練例を生成することができるリニア・ランドマーク・オートマトンを生成する。次に残りの訓練例生成できるオートマトンを生成する。すべての正の訓練例がカバーされたら、獲得したオートマトンに相当する単純なランドマーク文法を出力する。

STALKERは文書中の情報をそれぞれ独立に抽出するので、文書中における情報の出現順序に依存しない抽出が可能である。このことによりWIENよりも柔軟性の高い抽出が可能である。またSoftMealyとは対照的に、STALKERは抽出項目のすべての出現順を含んだ訓練例を用いなくてよい。

2)Semistructured and unstructured Web pages

この節では、より広範囲のテキストに適用可能やシステムとして、RAPIER,SRV, WHISKについて述べる。これらのシステムでは、意味論や文法を利用していないが、今後利用することも可能である。

これらのシステムはIEとWG(Wrapper Generation)の中間にある。手法は帰納論理プログラミングや関係学習に基づいており、SRVとWHISKは　帰納アルゴリズムFOIL　に、RAPIERは帰納アルゴリズムGOLEMと関係がある。

(a)RAPIER

RAPIER(Robust Automated Production of Information Extraction Rules)は、文書と情報の抽出部分に記述が埋め込まれたテンプレートを受け取り、抽出ルー

ルを学習する。

この学習アルゴリズムは帰納学習であり、はじめにターゲットのスロットと一致する最も特殊(汎用の逆の意味)なルールが適用される。次に2つのルールをランダムに選択して、2つのルールに対して汎化されたルールを生成する。そしてルールに変化がなくなるまで制約を追加する。

抽出パタンは、区切り文字とコンテンツ記述からなる。Pre-filler(ターゲットテキストの直前と一致するパタン)、 filler(ターゲットテキストと一致するパタン)、post-filler(ターゲットテキストの直後に一致するパタン)

各パタンは pattern items の列であり、一つの単語に一致するものか、もしくは複数の語に一致する pattern lists である。パタンとテキストの比較においては、(i) 単語、(ii) 品詞、(iii) 意味クラス、の3点が同じ場合に一致とみなす。

このシステムは、シングルスロットとして動作するが、テキストを3対上の部分に分割することで、マルチスロットとしての動作の可能性もある。

(b)SRV

SRV(Sequence Rules with Validation) は、トップダウンの関係学習アルゴリズムである。入力は抽出するフィールドに対してラベル付けされた文書集合と、トークンの特徴情報である。出力は、抽出ルールである。

SRV では IE を分類問題として扱う文書中の全ての抽出フレーズ候補に対して、ターゲット・スロットを満たす文字列の候補としての確信度を表す基準を割り当てる。オリジナルの SRV では、FOIL に類似したトップダウン処理の関係ルール学習を行う分類期を用いている。

SRV に与えるトークンの特徴情報には、文字列の長さ、文字列タイプ、つづり、品詞、語彙意味などを使うことができる。

処理は、まず全ての正例、負例のセットから開始し、以前に学習したルールがカバーしてない範囲の例と比較される。そして、ルールが正例のみをカバーする状態や、これ以上の汎化が無意味であるとなったときに、ルールと一致する全ての正例が取り除かれる。

SRV のルール表現力は高い。SRV は特定のアイテムを他の関連したアイテムとは独立に抽出できる点が STALKER や RPIER に似ている。関係学習ではシングルスロットの抽出のみ可能である。一方、WIEN ではマルチスロット抽出のルールを学習できる。

(c)WHISK

WHISK は、構造化されたテキストや半構造化されたテキスト、そして、ニュース記事などのフリーテキストからの抽出など、全てのタイプの抽出に対応するシ

ステムであり、文法情報と意味情報を用いる。構造化・半構造化テキストからの抽出では、文法情報は必要ない。一方、フリーテキストからの抽出では、入力テキストは文法解析と意味タグが付与されている状態で最も効率的に動作する。

WHISK の処理は、タグ付けされていないインスタンスと、空のタグ付けされたインスタンスがある状態から開始する。インスタンスとは、文書中の小さな単位のことである。半構造化された文書では、HTML などのタグを用いて文書がインスタンスに分割される。フリーテキストでは、文解析により文書を文に分割する。ユーザーにいくつかのタグ付けされてないインスタンスを提示し、ユーザが抽出部分を囲むようにタグを付与する。文書に埋め込まれたタグは、ルールの生成と評価に用いられる。

WHISK はトップダウンの分類ルール学習に属している。最初に最も一般的なルールを生成し、エラーがゼロになるか、または事前枝狩り (pre-pruning) 基準を満たすまでルールにタームが追加される。追加されるタームの選択では、ラプラシアン期待エラー (laplacian expected error)(e+1)/(n+1) が使われる。e はエラーの発生数であり、n は訓練例から得られた抽出情報の数である。学習は全ての正の抽出が行えるルールが生成されるまで行われる。また、事後枝狩り (post-pruning) が、オーバーフィッティングを防ぐために行われる。

3) Summary

本節では、Wrapper の自動学習システムについて述べた。表 refkakusisu は、各システムの特徴をまとめたものである。 X 印が、そのシステムが処理できる項目を表している。

上の 4 つのシステムは Wrapper Generation をベースとしたシステムであり、構造化された Web ページからデリミタを用いた抽出規則により抽出処理を行うものである。一方、下の 3 つのシステムは、IE をベースとしたものであり、関係学習を用いている。SRV と WHISK はトップダウン探索を行う。RAPIER は基本的にはボトムアップ探索である [1]。

表 3.7: 各システムの特徴

	struct	Semi	free	Multislot	Missing items	Permutations
ShopBot	X			X		
WIEN	X				X	X
SoftMeary	X	X			X	X
RAPIER	X	X			X	X
SRV	X	X			X	X
WHISK	X	X	X	X	X	X

struct：構造化された文書からの情報抽出

semi：半構造化された文書からの情報抽出

free：フリーテキストからの情報抽出

Multislot：マルチスロットの抽出ルールの生成

Missing items：文書中に抽出情報の一部が含まれていない場合の抽出処理

Permutations:文書中に出現する抽出情報の順番が不定の場合の抽出処理

第4章　TABLEタグ形式の表からのデータ抽出

秋田, 茨城の県番号と人口が書かれた表4.1がある。秋田, 茨城の県番号を抽出するとする。

表 4.1: 表の例

	秋田	茨城
県番号	05	23
人口（万人）	118	292

表4.1のTABLEタグ形式の表は次のように書かれている。

表 4.2: TABLE形式の表の例

```
<HTML><TITLE> html形式の表の例</TITLE><BODY>
<TABLE>
  <TR><B><TD> </B></TD><TD><I>秋田</I><TD><I>茨城</I></TD></TD></TR>
  <TR><B><TD>県番号</B></TD><TD>05</TD><TD>23</TD></TR>
  <TR><B><TD>人口（万人）</B></TD><TD>118</TD><TD>292</TD></TR>
</TABLE>
</BODY></HTML>
```

この表4.1と表4.2を見ると表のタグが<TABLE>, 改行が<TR>, 列の変更が<TD>であることが分かる。データを抽出するには、まず<TABLE>タグを認識し<TABLE>タグ以降を表として扱う。次に<TR>のタグを認識し、行として扱い</TR>のタグまでは同行となる。<TD>タグは列となりデータはこのタグと</TD>に挟まれた部分となる。

タグ形式の4.2からデータを取りだ出すには(項目名,名称,名称の項目名のデータ)の三つ組のデータを取り出す。表4.2から三つ組を取り出す。

表 4.3: 表の例から抽出した三つ組

(県番号, 秋田,05)
(県番号, 茨城,08)
(人口 (万人), 秋田,118)
(人口 (万人), 茨城,292)

この三つ組のデータと抽出するデータの項目名を与えて情報抽出をする。

第5章　実験

5.1　LR Wrapperによる情報抽出

情報抽出にはノートパソコンのスペックであるCPU名とインストールされているOS名と消費電力を取り出すことにした。

情報抽出に用いるページはメーカーのホームページのノートパソコンのスペック表の書かれているページを用いることにした。次にこのページに文字の位置情報のタグをつける。

これを対象にCPU,OS,消費電力が書かれている部分の左右の文字列を5個のページから取り出すと以下のようになる。

表 5.1: CPU の文字列

CPU	l1	>ffffff#=roloCgb "%32"=htdiw elddim=ngila DT< __>DT/<>TNOF/
		<1*>666666#=roloc __ TNOF<ーサッセロプ>2=napSlocffffff#=
	l2	>ffffff#=roloCgb "%32"=htdiw elddim=ngila DT< __>DT/<) 載
		搭ージロノクテ>RB<petSdeepS>RB<ルテンイ版張拡 (>RB<zHG06.1ーサッセロ
	l3	>ffffff#=roloCgb "%32"=htdiw elddim=ngila DT< __>DT/<) 載
		搭ージロノクテ>RB<petSdeepS>RB<ルテンイ版張拡 (>RB<zHG03.1ーサッセロ
	l4	>6e6e6e#=roloCgb __ "%17"=htdiwelddim=
		ngila DT< __>DT/<ーサッセロプ>cccccc#=rol
	l5	>6e6e6e#=roloCgb elddim=ngila DT< __>DT/<ーサッセロプ
		>cccccc#=roloCgb__ elddim=ngila parWon retnec=ngilAv DT<
	l6	>ffffff#=roloCgb elddim=ngila DT< __>DT/<ーサッセロプ>fffff
		f#=roloCgb "%22"=htdiw elddim=ngila DT< __>RT<__>RT
	l7	>ffffff#=roloCgb elddim=ngila DT< __>DT/<ーサッセロプ>fffff
		f#=roloCgb "%22"=htdiw elddim=ngila DT< __>RT<__>RT
	r1	</TD>__ <TD align=middle width="23%"bgColor=#ffffff>インテ
		ル Pentium __ M プロセッサー 1.30GHz (拡張版インテ
	r2	</TD>__ <TD align=middle width="23%" bgColor=#ffffff>インテ
		ル Pentium __ M プロセッサー 1.70GHz (拡張版インテ
	r3	</TD></TR>__ <TR>__ <TD align=middlewidth="31%" bgC
		olor=#ffffff colSpan=2>キャッシュメモリー</TD>__ <TDalign=
	r4	</TD></TR>__ <TR>__
		<TD vAlign=center noWrap align=middle width="29%" __
	r5	</TD></TR>__ <TR>__ <TDvAlign=center no
		Wrap align=middle __ bgColor=#cccccc>キャッシュメモ
	r6	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" bgC
		olor=#ffffff>キャッシュ メモリー (1 次／2 次) </TD>__<TD
	r7	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" bg—
		olor=#ffffff>キャッシュ メモリー (1 次／2 次) </TD>__<TD—

表 5.2: OS の文字列

CPU	l1	>ffffff#=roloCgb "%33"=htdiw elddim=ngila DT< __>DT/<SO>2=napSloc ffffff#=roloCgb "%13"=htdiw elddim=ngila DT< __>RT<
	l2	>ffffff#=roloCgb "%33"=htdiw elddim=ngila DT< __>DT/<lanoiseforP __ PX>RB<swodniW tfosorciM>ffffff#=roloCgb "%33"=h
	l3	>ffffff#=roloCgb "%33"=htdiw elddim=ngila DT< __>DT/<noitidE emoH __ PX>RB<swodniW tfosorciM>ffffff#=roloCgb "%33"=h
	l4	>6e6e6e#=roloCgb __ "%17"=htdiwelddim=ngila DT< __>DT/<SO>cccccc#=roloCgb
	l5	>6e6e6e#=roloCgb elddim=ngila DT< __>DT/<SO>cccccc#=roloCgb elddim=ngila parWon retnec=ngilAv DT< __>RT<
	l6	>ffffff#=roloCgb "%87"=htdiw elddim=ngila DT< __>DT/<SO>ffffff#=roloCgb "%22"=htdiw elddim=ngila DT< __>RT<__>
	l7	>ffffff#=roloCgb "%87"=htdiw elddim=ngila DT< __>DT/<SO>ffffff#=roloCgb "%22"=htdiw elddim=ngila DT< __>RT<__>
	r1	</TD>__ <TD align=middle width="33%"bgColor=#ffffff>Microsoft Windows XP __ Home Edition</TD>__<TD ali
	r2	</TD></TR>__ <TR>__ <TD align=middlewidth="31%" bgColor=#ffffff colSpan=2>プロセッサー<FONT __color=#666666
	r3	</TD></TR>__ <TR>__ <TD align=middlewidth="31%" bgColor=#ffffff colSpan=2>プロセッサー<FONT __color=#666666
	r4	</TD></TR>__ <TR>__ <TD vAlign=center noWrap align=middle width="29%" __
	r5	</TD></TR>__ <TR>__ <TDvAlign=center noWrap align=middle __bgColor=#cccccc>プロセッサー</TD>__
	r6	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" bgColor=#ffffff>プロセッサー</TD>__ <TD align=middlebgColor=#
	r7	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" bgColor=#ffffff>プロセッサー</TD>__ <TD align=middlebgColor=#

表 5.3: 消費電力の文字列

消費電力	l1	>ffffff#=roloCgb "%33"=htdiw elddim=ngila DT<__>DT/<力電費
		消>2=napSloc ffffff#=roloCgb "%13"=htdiw elddim=ngila DT<
	l2	>ffffff#=roloCgb "%33"=htdiw elddim=ngila DT< __>DT/<) ずま
		含電充ーリテッバ (>RB<W15 約>ffffff#=roloCgb "%33"=htdiwelddim=ngila
	l3	>ffffff#=roloCgb __ "%33"=htdiw elddim=ngila DT<__>
		DT/<) ずま含電充ーリテッバ (>RB<W04 約>ffffff#=roloCgb"%33"=htdiw el
	l4	>6e6e6e#=roloCgb __ "%17"=htdiwelddim=ng
		ila retnec=ngilAv DT< __>DT/<力電費消>ccc
	l5	>6e6e6e#=roloCgb __ elddim=ngilaretnec=ngilAv DT<
		__>DT/<力電費消>cccccc#=roloCgb elddim=ngilaparWon r
	l6	>ffffff#=roloCgb elddim=ngila DT< __>DT/<力電費消>ffffff#=r
		oloCgb "%22"=htdiw elddim=ngila DT< __>RT< __>RT/<>D
	l7	>ffffff#=roloCgb elddim=ngila DT< __>DT/<力電費消>ffffff#=r
		oloCgb "%22"=htdiw elddim=ngila DT< __>RT< __>RT/<>D
	r1	</TD>__ <TD align=middle width="33%"bgColor=#ffffff>約 40W<
		BR> (バッテリー充電含まず) </TD>__ <TD align=middlewidth="
	r2	</TD>__ <TD align=middle width="33%" __bgColor=#fff
		fff>約 56W (バッテリー充電含まず) </TD></TR>__ <TR>__
	r3	</TD>__ <TD align=middle width="33%" __bgColor=#fff
		fff>約 56W (バッテリー充電含まず) </TD></TR>__ <TR>__
	r4	</TD></TR>__ <TR>__
		<TD vAlign=center noWrap align=middle width="29%" __
	r5	</TD></TR>__ <TR>__ <TD vAlign=center no
		Wrap align=middle __ bgColor=#cccccc>省エネ法に基づ
	r6	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" bgC
		olor=#ffffff>省エネ法に基づくエネルギー消費効率<FONT __c
	r7	</TD></TR>__ <TR>__ <TD align=middlewidth="22%" __
		bgColor=#ffffff>省エネ法に基づく エネルギー消費効率<FO

この抽出した文字列からCPU,OS,重量でそれぞれ最長一致の文字列を取り出し抽出規則を帰納学習する。

表 5.4: 学習できた LR Wrapper

CPU	l	>
	r	</TD>
OS	l	>
	r	</TD>
消費電力	l	>
	r	</TD>

これを用いて情報抽出を行ったが学習できた文字列が大変みじかいため目的の情報以外の物も情報抽出してしまうためうまくいかない。

5.2 TABLEタグ形式の情報抽出

次にTABLEタグ形式の表の抽出を行った。TABLEタグ形式の表の抽出にもメーカーのホームページのノートパソコンのスペック表のページを用いた。そこからCPU,OS, ノートパソコンの重量のデータを取り出す。

TABLEタグ形式の表からのデータの抽出には(項目名, モデル名, 項目名のデータ)の三つ組を取り出す。

表 5.5: TABLEタグ形式の表から抽出したデータ

(FLORA 270Wサイレントモデルの主なスペック,FLORA 270Wサイレントモデル,)
(製品名,FLORA 270Wサイレントモデル,FLORA 270Wサイレントモデル)
(CPU,FLORA 270Wサイレントモデル,Pentium 4-M-1.8GHz)
(チップセット,FLORA 270Wサイレントモデル,Intel 845MP)
(メモリ(最大),FLORA 270Wサイレントモデル,128MB(PC2100 DDR SDRAM／最大512MB カスタムメイドで容量変更可))
(ビデオチップ,FLORA 270Wサイレントモデル,ATI MOBILITY RADEON7500(16MB))
(HDD,FLORA 270Wサイレントモデル,20GB(カスタムメイドで40GBに変更、40GBの2nd HDD追加可))
(FDD,FLORA 270Wサイレントモデル,3.5インチ(内蔵))
(光メディアドライブ,FLORA 270Wサイレントモデル,24倍速CD-ROM(カスタムメイドでCD-RW、CD-RW & DVD対応コンボドライブに変更可))
(スロット,FLORA 270Wサイレントモデル,PCカード(TypeII × 2またはTypeIII × 1、CardBus対応))
(通信,FLORA 270Wサイレントモデル,10BASE-T／100BASE-TX(カスタムメイドで56kbpsモデム、無線LAN内蔵可能))
(I/O,FLORA 270Wサイレントモデル,USB × 2、外部CRT、パラレル、シリアル、PS/2、オーディオ)
(サイズ(W × D × H),FLORA 270Wサイレントモデル,326 × 275 × 46mm)
(重量,FLORA 270Wサイレントモデル,約3.75kg)
(OS,FLORA 270Wサイレントモデル,Windows XP Home Edition(Windows2000モデルも用意))
(LaVie J LJ700/5Eの主なスペック,LaVie J LJ700/5E,)
(製品名,LaVie J LJ700/5E,LaVie J LJ700/5E)
...

表5.5の三つ組を取り出したら、その中から抽出する項目CPU,インストールOS,ノートパソコンの重量をCPU,OS,重量を表5.5に与えて抽出する項目名だけの三つ組を取り出す。

表5.6: TABLEタグ形式の表から抽出した項目名のみのデータ

(CPU,Endeavor NT330,Pentium M-1.40／1.50／1.60／1.70GHz、CeleronM-1.20GHzから選択)
(CPU,FLORA 270Wサイレントモデル,Pentium 4-M-1.8GHz)
(CPU,LaVie J LJ700/5E,超低電圧版MobilePentiumIII-M 933MHz)
(CPU,PC-MM2-NE6,TransmetaTM8600-1.0GHz)
(CPU,PowerBook G4 17インチ,PowerPC G4-1GHz(2次キャッシュ256KB、3次キャッシュ1MB))
(CPU,ThinkPad X312672-CBJ,Pentium M-1.40GHz)
(CPU,バイオノート505エクストリーム PCG-X505/SP,超低電圧版PentiumM-1GHz(拡張版インテルSpeedStepテクノロジー搭載))
(CPU,Afina AP5140CL,Intel Pentium M プロセッサ 1.40 GHz)
(CPU,WV2150,モバイル インテルCeleron プロセッサ 1.50GHz)
(CPU,PC STATION PA7240AVR,AMD Athlon(TM)XP プロセッサ 2400+ ※1)
(CPU,dynabook C8／213LDDW,Pentium M-1.30GHz)
...

この抽出された項目名のみの三つ組のデータのデータをhtml形式の表にプログラムで一括に表示できるように変換する。

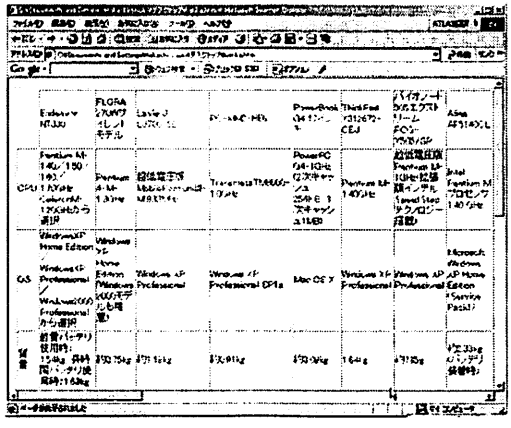

図5.1: 抽出した情報のブラウザでの一括表示

第6章　考察

今回の実験ではLR Wrapperによる学習は注目する情報の左右の文字列の取り出せる範囲が非常に短くなってしまいうまくいかなかった。このうまくいかなった原因としてはカラーやフォント、文字の大きさなどのタグにより注目する情報の左右の文字列が取れなかったためである。また注目している情報の文字にリンクが貼られていることも原因であった。

TABLEタグ形式の表からの注目する情報の抽出はある程度うまくいった。これはTABLEタグ形式の表は記述のフォーマットが厳密であるためである。しかしページのレイアウトとしてTABLEタグが使われている場合もレイアウトを表と誤認識してしまうためにうまくいかない。配置されている題名や文章を表のデータとして抽出してしまうためである。またマルチカラムが使われたTABLEタグには今回対応していない。これらの点は今後の課題である。

レイアウトの例としては図6.1のようなものがある。これは題名や表以外の文字や画像の部分もTABLEタグ形式を用いている。題名や文字などをきれいに配置をするのため、表のような形式となっている。

図 6.1: レイアウトの表の例

マルチカラムは図 refmulch のようなものがある。OS のデータが書かれているところを見ると、ここでは型番が違うが同じデータとなっている部分がある。これは型番が違うが同じデータを扱っているということで一つのデータが列をまたがっている。このように行や列をまたがっているものがマルチカラムである。

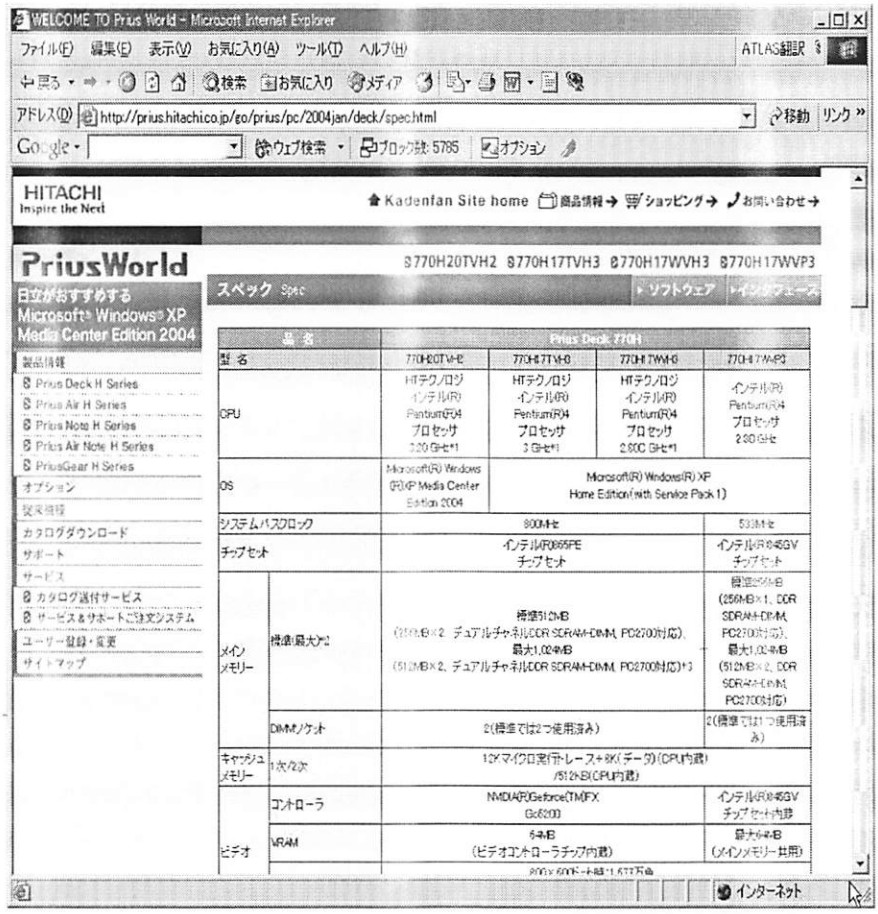

図 6.2: マルチカラムの表の例

第7章　おわりに

今回Webから表形式のデータ情報抽出する２つの手法を実装した。

一つはWrapper Inductionを用いて注目する情報を抽出するための規則学習を行った。しかし実験では適切な結果は得られなかった。考察で述べた問題点などがある。

もう一つはTABLEタグ形式の表からの情報抽出である。今回の実験ではこれを用いて注目する情報のデータを抽出することがある程度できた。しかしページのレイアウトとしてTABLEタグが使われている場合とマルチカラムが使われたTABLEタグへの対応は今回行っておらず、これらは今後の課題である。

謝辞

本研究の遂行及び論文の作成において多大な御助言及び指導を賜わった新納 浩幸教官 (茨城大学工学部システム工学科)、に深い感謝の意を表します。

また、御指導を頂きましたシステム工学科計算機応用学講座の教官の方々にも深く感謝します。

最後に、本研究を進めるにあたり助言、強力を頂きました岩崎 唯史教官 (茨城大学工学部システム工学科)、同研究室の紺野 憲一氏 (茨城大学大学院理工学研究科システム工学専攻、大城 亜里沙氏 (茨城大学システム工学科 4 回生)、時田 陽一氏 (茨城大学システム工学科 4 回生)、藤井 丈明氏 (茨城大学システム工学科 4 回生)、脇坂 恭志朗氏 (茨城大学システム工学科 4 回生) に深く感謝致します。

関連図書

[1] Kushmerick,N : "Wrapper induction:Efficiency and expressiveness", Artificial Intelligence,Vol.118, pp.15-68 (2000).

[2] Line Eikvil : "Information Extraction from World Wide Web- A Survey", Norweigan Computing Center, No.945(1999).

[3] 電子情報技術産業協会 : "ヒューマンインターフェイス技術に関する調査報告書", 平成15年度年次報告書 (2003).

[4] Cowie,J : "Automatic analysis of descriptive texts", Applied Natural Language Processing (1983).

[5] DeJong,G : "An overview of the FRUMP system", In W. Lehnert and M. Ringle, editors, "Strategies for Natural Language Processing", Lawrence Erlbaum Associates (1982).

[6] Hayes,P.J,Anderson, P.M., Huettner, A.K., Birenburg, I.B. and Schmandt, L.M. : "Automatic extraction of facts from press releases to generate news stories ", 3rd conference on Applied Natural Language Processing, pp.170-177 (1983).

[7] 小松英二, 加藤安彦, 安原宏, 椎野努 : "要約支援システム COGITO : 文書の構造解析", ICOT Technical report, TM-0415 (1987).

[8] Sager,N. : "Natural Language Information Processing", Addison wesley, (1981).

[9] Sager, N., Friedman, C. and Lyman, M. : "Medical Language Processing:Computer management of Narrative Date", Addison Wesley, (1987).

[10] 佐藤理史, 佐藤円 : "ネットニュースグループ fj.wanted のダイジェスト自動生成 ", 自然言語処理,Vol.3,No.2, pp.19-32 (1996).

[11] Kushmerick,N., Weld, D.S. And Doorenbos,R. : "Wrapper Induction for Information Extraction ", IJICAI-97, pp.729-735 (1997).

[12] Ashish and Knblock, C. : "Wrapper generation forsemi-structured Internet sources ", SIGMOD record,Vol.26, No.4, pp.8-15 (1997).

[13] Soderland, S. : "Learning to extract text-based information from the world wide web ", 3rd International Conference on Knowledge Discovery and Data Mining(KDD-97), pp.251-254 (1997).

[14] Califf, M. E. and Mooney, R.J. : "Relational Learning of Pattern-Match rules for Information Extraction ", ACL-97 Workshop on Natural Language Learning, pp.9-15 (1997).

[15] Freitag, D. : "Multistarategy Learning for Information Extraction ", 5th International Machine Learning Confernce, pp.161-169 (1998).

[16] Lehnert, W., Cardie, C., fisher, D., McCarthy, J., Riloff, E. and Soderland, S. : "University of Massachusetts: Description of the CIRCUS System as Used for MUC-4", 4th Message Understanding Conference (MUC-4),pp.282-288 (1992).

[17] Soderland, S., Fisher, D., Aseltine, J. and Lengert, W. : "CRYSTAL: Inducing aConceptual Dictionary ", IJICAI-95, pp.1314-1319 (1995).

[18] Soderland, S., : "Lerning Information Extraction Rules for Semi-structured and Free Text", Machine Learning,Vol.34,No.1 (1999).

[19] Kim, J.T. and Moldovan, D.I. : "Acquisition of Linguistic Patters for Knowledge-Based Infomation Extraction", IEEE Transactions on Knowledge and Data Engineering.Vol.7,No.5 pp.713-724 (1995).

[20] Riloff, E. : : "Automatically Generating Extraction Patterns from Untagged Text ", 13th National Conference on Artificial Intelligence(AAA-96), pp.1044-1049 (1996).

[21] Riloff, E. : : "Automatically Constructing a Dictionary for Information Retrieval", 11th National Confernce on Artificial Intelligence, pp.811-816 (1993).

[22] ネットワークアクセス技術委員会 : : "ネットワークアクセス技術専門委員会活動報告", 自然言語処理システムに関する調査報告者, pp.152-164 (1999).

付録 プログラムリスト

```
/*-------------------------------------------------

抽出する情報の左側をの文字列を取り出すプログラム

-------------------------------------------------*/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 100

void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],
     data[50000][3];
  int r,i=1,a,b,d,e,suu1,suu2,suu3,suu4,suu5,suu6,suu7,suu8;

  if(argc!=4) quit("引数の数が違う prog datafile ");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       sscanf(buf,"%s %s",word1,word2);
```

```
            if(strcmp(word2,"") == 0 )
        {    d=0;
              while(buf[d] != '\n')
                 d++;  e=d-1;
              if(buf[e] == ' ')
               {strcpy(data[i]," ");}
                 else
               strcpy(data[i],"__");
           }
              else
              strcpy(data[i],word2);

          i++;
      strcpy(word2,"");
   }

if((f1=fopen(argv[2],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       sscanf(buf,"%u %u %u %u %u %u %u %u",&suu1,&suu2,
                  &suu3,&suu4,&suu5,&suu6,&suu7,&suu8);
                   a = atoi(argv[3]);
              strcpy(word3,"");
         if(a == 1){b=suu1;for(i=1;i<=200;i++){
              suu1--;
                 strcat(word3,data[suu1]);}
               }

         if(a == 3){
            b=suu3;for(i=1;i<=200;i++){
              suu3--;
                 strcat(word3,data[suu3]);}
                    }

         if(a == 5){b=suu5;for(i=1;i<=200;i++){
```

```
            suu5--;
               strcat(word3,data[suu5]);}
              }

        if(a == 7){b=suu7;for(i=1;i<=200;i++){
            suu7--;
               strcat(word3,data[suu7]);}
              }
             printf("%u %s\n",b,word3);
}


    if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
  }
  void quit(char *s)
{
  puts(s); exit(1);
}

/*-------------------------------------------------

抽出する情報の右側をの文字列を取り出すプログラム

-------------------------------------------------*/


#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 100
```

```
void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],
     data[50000][3];
  int r,i=1,d,e,a,b,suu1,suu2,suu3,suu4,suu5,suu6,suu7,suu8;

  if(argc!=4) quit("引数の数が違う prog datafile ");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       sscanf(buf,"%s %s",word1,word2);

              if(strcmp(word2,"") == 0 )
        {     d=0;
              while(buf[d] != '\n')
                 d++;  e=d-1;
             if(buf[e] == ' ')
              {strcpy(data[i]," ");}
                else
              strcpy(data[i],"__");
           }
              else
              strcpy(data[i],word2);
              i++;
      strcpy(word2,"");
   }

if((f1=fopen(argv[2],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       sscanf(buf,"%u %u %u %u %u %u %u %u",&suu1,&suu2,
      &suu3,&suu4,&suu5,&suu6,&suu7,&suu8);
                    a = atoi(argv[3]);
```

```
            strcpy(word3,"");
          if(a == 2){b=suu2;for(i=1;i<=200;i++){
            suu2++;
               strcat(word3,data[suu2]);}
              }

          if(a == 4){
             b=suu4;for(i=1;i<=200;i++){
            suu4++;
               strcat(word3,data[suu4]);}
                   }

          if(a == 6){b=suu6;for(i=1;i<=200;i++){
            suu6++;
               strcat(word3,data[suu6]);}
              }

         if(a == 8){b=suu8;for(i=1;i<=200;i++){
            suu8++;
               strcat(word3,data[suu8]);}
              }
          printf("%u %s\n",b,word3);
}


     if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
  }
  void quit(char *s)
{
  puts(s); exit(1);
}

/*-----------------------------------------------
```

左右の文字列の最長一致を取り出すプログラム

```
----------------------------------------------------*/


#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 250


void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],
       word4[WORDSIZE];
  int r,i,a=0,b,flag;

  if(argc!=2) quit("引数の数が違う prog datafile ");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       i=0;flag=0;b=0;

       if(flag == 0){
          while(buf[i] != ' '){
          flag++;i++;}   }

       if(flag != 0){
          while(buf[i] != '\n'){i++;
           word2[b] = buf[i];
```

```
            b++;
          }
          }
           i=0;

           if(a == 0)strcpy(word3,word2);

           strcpy(word4,word2);

            while(word3[i] == word4[i]){i++;}
               word3[i] = '\0';    a++;
}
    printf("%s\n",word3);
     if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
  }
  void quit(char *s)
{
  puts(s); exit(1);
}

/*-------------------------------------------------
```

注目する情報を取り出すプログラム

```
-------------------------------------------------*/


#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 250
```

```
void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],
       home[WORDSIZE],kaisetsu[WORDSIZE],
     away[WORDSIZE],data[50000][3],l1[200][3],
       l3[200][3],l5[200][3],r2[200][3],
     r4[200][3],r6[200][3];
  int r,a,b,c,d,e,f,h,kazu,l11=1,r22=1,l33=1,r44=1,l55=1,r66=1;
  long i=1;
  strcpy(home,"");strcpy(away,"");strcpy(kaisetsu,"");
  if(argc!=8) quit("引数の数が違う prog datafile ");

 if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){
       sscanf(buf,"%s %s",word1,word2);
              if(strcmp(word2,"") == 0 )
        {    d=0;
             while(buf[d] != '\n')
                d++;  e=d-1;
            if(buf[e] == ' ')
             {strcpy(data[i]," ");}
               else
             strcpy(data[i],"__");
          }
             else
             strcpy(data[i],word2);
            i++;
     strcpy(word2,"");
   }
       strcpy(data[i],"");
```

```
 if((f1=fopen(argv[2],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){

          i=0;b=0;
          while(buf[i] != '\n'){
           l1[l11][b] = buf[i];
           b++;i++;
         }

         l11++;

  }

if((f1=fopen(argv[3],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){
     i=0;b=0;
     -    while(buf[i] != '\n'){                    -
          -  r2[r22][b] = buf[i];                    -
           b++;i++;
         }
          r22++;

  }

  if((f1=fopen(argv[4],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){

          i=0;b=0;
          while(buf[i] != '\n'){
           l3[133][b] = buf[i];
           b++;i++;
         }
```

```
        l33++;

    }

if((f1=fopen(argv[5],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){
     i=0;b=0;
         while(buf[i] != '\n'){
          r4[r44][b] = buf[i];
          b++;i++;

        }
         r44++;

    }

 if((f1=fopen(argv[6],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){

          i=0;b=0;
          while(buf[i] != '\n'){
           l5[l55][b] = buf[i];
           b++;i++;
        }

        l55++;

    }

if((f1=fopen(argv[7],"r"))==NULL) quit("ファイルが開けない");
    while(fgets(buf,LINESIZE,f1)!=NULL){
     i=0;b=0;
         while(buf[i] != '\n'){
          r6[r66][b] = buf[i];
```

```
            b++;i++;

          }
           r66++;

    }

                i=1;
//CPU の部分


while(strcmp(data[i],"") != 0) {
          a=l11-1;
      if( strcmp(data[i],l1[a]) == 0){  kazu=0;

      while( strcmp(data[i],l1[a]) == 0){a--;i++;kazu++;}
           if (a != 0)i=i-kazu;          } -

      if(a == 0){a=1;b=1;


        while(strcmp(data[i],r2[a]) != 0){strcat(home,data[i]);i++;kazu++;}

            while(strcmp(data[i],r2[a]) == 0 ){a++;i++;kazu++;}

            i=i-kazu;
    if(a == r22){
         printf("CPU %s\n",home);
          }strcpy(home,"");
  }
   // インストール OS の部分

    c=l33-1;
```

```
 if( strcmp(data[i],l3[c]) == 0){  kazu=0;

 while( strcmp(data[i],l3[c]) == 0)
  {c--;i++;kazu++;// printf("%d   ",i);printf("%d\n",c);
}
     if (c != 0)i=i-kazu;          }

 if(c == 0){c=1;d=1;


   while(strcmp(data[i],r4[c]) != 0)
      {strcat(away,data[i]);i++;kazu++;}

       while(strcmp(data[i],r4[c]) == 0 ){c++;i++;kazu++;}

  i=i-kazu;
if(c == r44 ){
     printf("OS %s\n",away);
             }  strcpy(away,"");  }


//  ノートパソコンの重量の部分


e=155-1;
  if( strcmp(data[i],15[e]) == 0){ kazu=0;

  while( strcmp(data[i],15[e]) == 0){e--;i++;kazu++;}
      if (e != 0)i=i-kazu;          }

  if(e == 0){e=1;f=1;

    while(strcmp(data[i],r6[e]) != 0){strcat(kaisetsu,data[i]);
```

```
                    i++;kazu++;}

          while(strcmp(data[i],r6[e]) == 0 ){e++;i++;kazu++;}
                  i=i-kazu;
    if(e == r66){
         printf("重量 %s\n\n",kaisetsu);
                          }strcpy(kaisetsu,"");


}    i++;

}


   if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
  }
  void quit(char *s)
{
  puts(s); exit(1);
}

/*-------------------------------------------------

３つ組を取り出すプログラム

-------------------------------------------------*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 100

void quit(char *);
```

```
int main(int argc,char *argv[])
 {
  FILE *f1;
 char buf[LINESIZE],table[10][20][20][500],word1[10],
 word2[10],word3[10],word4[10],word5[10],
 font[10],bd[10],font2[10];
 int b,a1,a2,a3,a4,a5,a6,a7,r,c,flug1=0,flug2=0,flug3=0,flug4=0,
 i=-1,tab=1,verse=0,line=1,value=0;

if(argc!=4) quit("引数の数が違う prog datafile");
if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
  while(fgets(buf,LINESIZE,f1)!=NULL){

while(buf[i]!='\n'){

  i++;
 while(buf[i] == ' '){i++;}
  a3=i-5;a4=i-4;a5=i-3;a6=i-2;a7=i-1;
  word1[0]=buf[a3];word1[1]=buf[a4];word1[2]=buf[a5];
  word1[3]=buf[a6];word1[4]=buf[a7];word1[5]=buf[i];

 if(strcmp(word1,"<TABLE") == 0 ||
      strcmp(word1,"<table") == 0 ||flug1==2){
  i--;
    while(buf[i] != '\n' && flug1 != 1){
     i++;
  a1=i-7;a2=i-6;a3=i-5;a4=i-4;a5=i-3;a6=i-2;a7=i-1;
 word2[0]=buf[a1];word2[1]=buf[a2];word2[2]=buf[a3];word2[3]=buf[a4];
 word2[4]=buf[a5];word2[5]=buf[a6];word2[6]=buf[a7];word2[7]=buf[i];

 word5[0]=buf[a6];word5[1]=buf[a7];word5[2]=buf[i];

 flug1=2;
```

```
   if(strcmp(word2,"</table>") == 0 || strcmp(word2,"</TABLE>") ==0)
{tab++;flug1=1;}
   if(strcmp(word5,"<TR") == 0 || strcmp(word5,"<tr") ==0)
{verse++;line=1;}
   if(strcmp(word5,"<TH") == 0 || strcmp(word5,"<TD") == 0 ||
      strcmp(word5,"<th") == 0 || strcmp(word5,"<td") == 0 || flug2==2){
      i--;

        flug2=3;
        while(buf[i] != '\n' && flug2 !=1){
          i++;
    a2=i-6;a3=i-5;a4=i-4;a5=i-3;a6=i-2;a7=i-1;
   word4[0]=buf[a4]; word4[1]=buf[a5]; word4[2]=buf[a6];
   word4[3]=buf[a7]; word4[4]=buf[i];

    a4=i+1;a5=i+2;a6=i+3;a7=i+4;
   word3[0]=buf[i];word3[1]=buf[a4];word3[2]=buf[a5];word3[3]=buf[a6];

font[0]=buf[i];font[1]=buf[a4];font[2]=buf[a5];
font[3]=buf[a6];font[4]=buf[a7];

font2[0]=buf[i];font2[1]=buf[a4];font2[2]=buf[a5];
font2[3]=buf[a6];font2[4]=buf[a7];
   bd[0]=buf[i];bd[1]=buf[a4];
bd[2]=buf[a5];bd[3]=buf[a6];
   c=i+1;
  if(((buf[i]=='>' && buf[c] != '<') || flug3 == 1) &&
(strcmp(word4,"</TD>") != 0 || strcmp(word4,"</TH>") != 0 ||
 strcmp(word4,"</td>") != 0 || strcmp(word4,"</th>") != 0 )) {
    flug3=1;
   if(strcmp(font,"<FONT") ==0 || strcmp(font,"<font") ==0 || flug4==1){

    while(buf[i] != '>' && buf[i] != '\n'){
   i++;
```

```
    if(buf[i] == '\n')flug4=1;
    }

    }
   if(buf[i] == '>'){i++;flug4=0;}

   if(strcmp(word3,"<BR>") ==0 || strcmp(word3,"<br>") ==0 )i=i+4;
   if(strcmp(font2,"</FON") ==0 || strcmp(font2,"</fon") ==0 )i=i+7;

     if(strcmp(bd,"</B>") ==0 || strcmp(bd,"</b>") ==0 )i=i+4;

    if(buf[i] != '\n'){
 table[tab][verse][line][value]=buf[i];value++; } /* 行列に値を代入す
る */

     }
  flug2=2;
   if(strcmp(word4,"</TD>") ==0 || strcmp(word4,"</TH>") ==0 ||
      strcmp(word4,"</td>") ==0 || strcmp(word4,"</th>") ==0 ) {

     flug3=2;if(buf[i]=='\n')value++;
  b=value-5; table[tab][verse][line][b] = '\0';flug2=1;line++;value=0;
       }
     }
    }
  }
}
}
     i=-1;
   }

 printf("要素は------>( %s, %s, %s)\n",table[1][atoi(argv[2])][1],
table[1][1][atoi(argv[3])],table[1][atoi(argv[2])][atoi(argv[3])]);
  if((r=fclose(f1)) == -1) quit("ファイルが閉じれない");
```

```
  }
void quit(char *s)
{
  puts(s);exit(1);
 }

/*-------------------------------------------------

３つ組から注目する３つ組を取り出すプログラム

-------------------------------------------------*/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 255

void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],
     tyuumoku[WORDSIZE],word3[WORDSIZE];


  int r,i,a,b;


  if(argc!=3) quit("引数の数が違う prog datafile ");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
```

```
    while(fgets(buf,LINESIZE,f1)!=NULL){
      sscanf(buf,"%s",tyuumoku);

   }

  if((f1=fopen(argv[2],"r"))==NULL) quit("ファイルが開けない");
     while(fgets(buf,LINESIZE,f1)!=NULL){b=1;
   while(buf[i]!= '\n') {
        i++;
    if(buf[i]!='('){ a=0;
         while(buf[i] != ',' && buf[i] !=')')
       if(b==1)word1[a]=buf[i];
       if(b==2)word2[a]=buf[i];
       if(b==3)word3[a]=buf[i];
        }
        b++;

}
   if(strcmp(word3,tyuumoku) == 0)printf("(%s,%s,%s)\n",word1,word2,word3);
     if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
  }

}
  void quit(char *s)
{
  puts(s); exit(1);
}

/*-------------------------------------------------

3つ組をhtml形式の表に変換するプログラム

-------------------------------------------------*/
```

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 255
#define WORDSIZE 255


void quit(char *);

int main(int argc,char *argv[])
{
   FILE *f1;
  char buf[LINESIZE],word[30][30][200],
     table[30][30][200];


  int r,i=0,a,b=0,c,d=0,e,flug,h,j,k,l;


  if(argc!=2) quit("引数の数が違う prog datafile ");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
  while(fgets(buf,LINESIZE,f1)!=NULL){
       i=-1;
   while(buf[i] != '\n') {
        i++;
   if(buf[i]!='(' && buf[i] != '\n'){ a=0;
        while(buf[i] != ',' && buf[i] != '\n'){
      word[d][b][a]=buf[i];

       i++;a++;
        }
         c=i-1;
```

```
      if(buf[c]==')')a--;

        word[d][b][a]='\0';

       b++;
         }


  }      d++;     b=0;

}

if((r=fclose(f1))==-1) quit("ファイルが閉じれない");
    h=1;
for(i=0;i<d;i++){flug=0;
    if(i==0)strcpy(table[0][0],"   ");
  for(e=0; e<h; e++){
       if(strcmp(table[0][e],word[i][1]) == 0)flug=1;
     }
  if(flug ==0){strcpy(table[0][h],word[i][1]);

   h++;
   }


}
 h--;
 j=1;
for(i=0;i<d;i++){
   flug=0;
  for(e=0; e<j; e++){
       if(strcmp(table[e][0],word[i][0]) == 0)flug=1;
     }
  if(flug ==0){strcpy(table[j][0],word[i][0]);   j++;
```

```
  }
 }
 j--;
   printf("%s\n",table[4][0]);

   for(i=0;i<d;i++){k=0;l=0;
      while(strcmp(table[k][0],word[i][0]) != 0)
             k++;
      while(strcmp(table[0][l],word[i][1]) != 0)
             l++;
    strcpy(table[k][l],word[i][2]);
}

  printf("<table>\n\n");

for(a=0;a<=j;a++){
 printf("<tr>");
 for(i=0;i<=k;i++)
    printf("<td>%s</td>",table[a][i]);

   printf("</tr>\n");

   }
 printf("</table>\n\n");
   }
  void quit(char *s)
{
  puts(s); exit(1);
}
```