位置指向 Web 文書のハイパーリンク表示位置を考慮した収集法

田中 秀彦　鈴木 徹也

芝浦工業大学システム工学部電子情報システム学科 〒337-8570 埼玉県さいたま市見沼区大字深作 307

E-mail: {p03070,tetsuya}@shibaura-it.ac.jp

あらまし　本研究では位置情報を含む Web 文書の収集法として, Web ブラウザ上でのハイパーリンク表示位置を考慮したクローリング手法を提案する．Web 上の大量の情報を対象にした位置指向検索を実現するためには，位置情報を含む Web 文書を効率的に収集するクローリング手法が重要となる．先行研究ではリンク文字列に位置情報が含まれるか否かに着目し，Web 文書の訪問優先順位を決定している．本研究では，ハイパーリンクの表示位置にも着目し，その優先順位を決定する．実験により，提案手法は従来手法よりも効率的に位置情報を含む Web 文書を収集できることを確認した．

キーワード　Web クローラ　地理情報システム　データマイニング

1.序論

1.1.研究の背景と目的

位置指向検索とは，住所情報や緯度経度などの位置情報を元にした検索方法である．既存の位置指向検索システムの例としては Google Maps[1]が挙げられる．位置指向検索では，地理的な距離を考慮した検索が可能になる．そのため，World Wide Web(WWW)上で一般的に使われているキーワード検索では困難な，行政区域をまたぐような検索が可能になるといった利点がある．

WWW 上の文書(Web 文書)を対象にした位置指向検索の実現を考えた場合, 位置情報を含む Web 文書を収集することが必要である．特に，より品質の高い検索を行うためには大量の Web 文書を収集する必要がある．その収集には，クローラを利用するのが現実的である．クローラとは，Web 文書内に存在するハイパーリンク(リンク)を辿りながら Web 文書を自動的に収集するソフトウェアである．また，その作業をクローリングという．

横路[2]らによると, クローラで収集した Web 文書をランダムに目視で確認したところ，位置情報を含むものの割合は全体の 3 割以下であった．したがって，位置情報を含む Web 文書を効率よく収集するには, クローリング戦略が重要となる．

上記の理由から, 本研究では位置情報を含む Web 文書を効率的に収集する方法を提案することを目的とする．

1.2.論文の構成

本論文の構成を以下に示す．

- 第 2 節では既存の研究について述べる．
- 第 3 節では提案するクローラの構成法について述べる．
- 第 4 節では実験について述べる．
- 第 5 節では実験結果の評価について述べる．
- 第 6 節では全体のまとめについて述べる．

2.準備

この節では既存の研究について述べる．

2.1.収集優先度付きクローラ

本研究で扱うクローラの基本形として，収集優先度付きクローラを説明する．図 1.1 にその擬似コードを示す．収集優先度付きクローラは，優先度付き URL キューを利用する．優先度付き URL キューとは, 未訪問の Web 文書の URL を記録するキューである．クローラはその優先度付き URL キューから，URL を 1 つ取り出す．ただしキューからは，優先度が最も高い URL のうち，最も先に記録された URL が取り出されるものとする．そして，その URL が指す Web 文書を取得する．次にその Web 文書内のリンクを取り出す．取り出したリンクのうち, 未訪問の Web 文書へのリンクを，そのキューに記録する．その際，リンクの優先度を計算する．クローラはこの動作を繰り返し行う．

リンクの優先度は関数 priority で計算する．関数 priority によってクローリング戦略が変わる. たとえば図 1.2 に示すように，常に同じ優先度を返す priority 関数を用いると幅優先探索となる．

2.2.横路らのクローリング戦略

先行研究として，横路らのクローリング戦略を説明する．横路らの方法では，クローリング戦略にリンク文字列を利用する．**リンク文字列**とは HTML のアンカータグに挟まれた文字列である．具体的には，リンク文字列からリンクを以下のような 3 つのケースに分類して収集優先度を決定する．

- case1 ：リンク文字列に位置情報が含まれている場合，リンク先の Web 文書を優先的に収集する．
- case2 ：そのリンク文字列には位置情報が含まれないが，同一 Web 文書内に case1 のリンクがある

```
手続き crawl
入力
  queue:優先度付きキュー. 収集開始 URL が記録されているとする.
  maxpages:収集ページ数
  priority:A 要素(アンカー要素)を引数に取り, その優先度を返す関数
出力 訪問したページの URL 集合
副作用 queue の更新
begin
  counter := 0
  visitied := {}
  while counter < maxpages and
    queue の要素数 > 0 begin
    url := getElement(queue)
    URL url のページを取得し, その内容を page とする.
    visited := visited ∪{url}
    cunter := counter + 1
    for page 中の各 A 要素 anchor について begin
      if not(anchor のリンク先 URL ∈ visited) begin
        putElement(queue,リンク先 URL,
          priority(anchor))
      end
    end
  end
  return visited
end
```

図 1.1 優先度付きクローラの擬似コード

```
関数 priority
入力 anchor:A 要素(アンカー要素)
出力 優先度
begin
  return 0
end
```

図 1.2 幅優先探索手法の擬似コード

場合, リンク先の Web 文書を中程度優先的に収集する.

- case3 : 上記以外のリンク先の Web 文書は優先しない.

このクローリング戦略は, 事前に収集した Web 文書を解析して決定された. 横路らの手法を関数 priority に適用したものを図 1.3 の擬似コードで示す.

横路らは 100,000 ページを収集するクローリング実験を行った. その結果, 幅優先探索手法では, 収集した Web 文書のおよそ 2 割が位置情報を含んでいた. また, 横路らの提案した手法では, およそ 6 割が位置情報を含んでいた. このことから適切なクローリング戦略の利用が収集の効率改善につながることが分かる.

```
関数 priority
入力 anchor : A 要素(アンカー要素)
出力 優先度
begin
  if anchor が case1 の場合 begin
    return 1.0 //最優先
  end
  if anchor が case2 の場合 begin
    return 0.5 //中程度に優先
  end
  return 0 //優先しない
end
```

図 1.3 横路らの手法を用いた擬似コード

位置情報を含む Web 文書の収集において横路らは, クローリング戦略としてリンク文字列に着目した. そこで, Web 文書内の他の情報も考慮してクローリング戦略を考えれば, より効率的に Web 文書の収集が可能になると考えられる.

3.提案するクローラの構成法

この節では, 位置情報を収集するクローラの構成法を提案する. 提案する構成手順は次のようになる.

(1) クローリング戦略を決定する目的で Web 文書を収集する.
(2) 収集した Web 文書中のリンクからその特徴(属性）を抽出する.
(3) 抽出したリンクの属性から機械学習により分類器を作成し, それを利用した関数 priority を構成する.

抽出するリンクの属性とは以下の 4 種類である.

- 属性 a1: そのリンク文字列に位置情報が含まれるなら真, そうでなければ偽.
- 属性 a2: 同一 Web 文書内に属性 a1 が真となるリンクがあれば真, そうでなければ偽.
- リンクの表示位置を反映した属性
- リンク先 Web 文書に位置情報があれば真, そうでなければ偽となる属性(教師信号)

関数 priority は次のように構成する. 学習によって得られる分類器は, リンクの属性（属性 a1, 属性 a2, リンクの表示位置を反映した属性）から, そのリンク先 Web 文書内に位置情報があるかどうかを予測するのに利用する. リンク先に位置情報があると分類器によって判断されれば, 関数 priority は優先度 1 を返し, そうでなければ優先度 0 を返す.

先行研究との大きな違いはリンクの表示位置を反

映した属性を利用することである．その属性を抽出する動機は次のような理由による．Web 上では，ナビゲーションを含む Web 文書をよく見かける．**ナビゲーション**とは，Web サイトにおいて，閲覧者が目的の Web 文書を見つけやすくするためのリンク集である．そしてナビゲーションは，Web 文書の上下左右など，閲覧者が直感的に分かりやすい位置に配置されることが多い．さらにこのナビゲーションには，その組織の所在地を記したページへのリンクを含む場合がよくある．

Web ブラウザ上での各リンクの正確な表示位置を計算するのは，計算コストが大きいと考えられる．そこで本研究では以下に示す疑似 X 座標と疑似 Y 座標を提案する．

3.1.疑似 X 座標

疑似 X 座標は, Web 文書のレイアウトに HTML の表構造が用いられていると仮定して計算される座標である． それを計算する疑似コードを図 2.1 と図 2.2 に示す．その概要を簡単に説明すると, 以下のようになる．

- Web 文書の左端の疑似 X 座標を 0，右端の疑似 X 座標を 1 とし，Web 文書の横幅を 1 とする．
- 表の横幅は，それが配置されている領域の横幅とする．
- 表のセルは，その表の横幅を等分する．
- リンクの疑似 X 座標は, それが配置されている領域の中点とする．

結果として疑似X座標は 0 から 1 までの値となる．

```
関数 countBlockElements
入力 node : DOM ツリーのノード
出力 node とその配下にあるブロック要素数
begin
  s:= 0
  for node の各子要素 child について begin
    s := s + countBlockElements(child)
  end
  if node が HR 以外のブロック要素 begin
    return s+2
  end
  if node が HR 要素 begin//HR 要素は子要素を持たない
    return s+1
  end
  return 0
end
```

図 2.1 擬似 X 座標計算の擬似コード 1

3.2.疑似 Y 座標

疑似 Y 座標は，HTML 文書中に現れるブロック要素のタグの個数を数えて，以下のように計算される．

```
関数 calcPseudoX
大域変数 table : アンカー要素とその擬似 X 座標を記憶するテーブル
入力
  node : DOM ツリーのノード
  L : 現在の左端の擬似 x 座標
  W: 現在の幅
出力 なし
副作用 大域変数 table の更新
begin
  if node がアンカー要素 begin
    px := L + W / 2.0
    table にアンカー要素とその優先度 px を記録する
    return
  end
  if node が TR 要素 begin
    n:= node 直下にある TD 要素の総数
    for j := 1 to n begin
      L' := L+ W * (∑_{i=1}^{j-1}i 番目の TD 要素のコラム幅)/(∑_{i=1}^{n}i 番目の TD 要素のコラム幅
      W' := W * (j 番目の TD 要素のコラム幅)/(∑_{i=1}^{n}i 番目の TD 要素のコラム幅)
      calcPseudoX(j 番目の TD 要素,L',W')
      end
    end
  end
ただし
  ∑_{i=a}^{b}f(i) = f(a) + f(a+1) +... + f(b)とする．
  TD 要素のコラム幅とは, TD 要素の colspan の値とする．colspan が指定されていない場合のコラム幅は 1 とする．
```

図 2.2 擬似 X 座標計算の擬似コード 2

- （そのリンクより前にあるブロック要素のタグ数）/（Web 文書中に現れる全てのブロック要素のタグ数）

これを計算する擬似コードを図 2.3 に示す．ブロック要素のタグ数を数えるのは，Web ブラウザ上では，そのタグの前後に縦方向の空白が挿入されるからである．疑似 Y 座標も，疑似X座標同様，0 から 1 までの値となる．

4.実験

次の 2 つのことを実験により確かめる． 1 つめは，提案した疑似座標と Web ブラウザ上のリンク表示位置との相関である． 2 つめは，提案手法で構成されたク

```
関数 calcPseudoY
大域変数
table :アンカー要素とその擬似 Y 座標を記録するテーブル
order :ブロック要素の出現順位．初期値 0
s :ブロック要素の総数．関数 countBlockElements で計算した値．
入力 node :DOM ツリーのノード
出力 なし
副作用 大域変数 table の更新
begin
  if node が A 要素 begin
    py := order / s
    table にアンカー要素 node とその疑似 Y 座標 py を記録する．
    return
  end
  if node がブロック要素 begin
    order := order + 1
  end
  if node の各要素 child について begin
    calcPseudoY(child)
  end
  if node が HR 以外のブロック要素 begin
    //HR 要素は子要素を持たない
      order := order + 1
  end
end
```

図 2.3 擬似 Y 座標計算の擬似コード

ローラーの収集効率である．

4.1.疑似座標の検証

この実験では，Web 文書中のリンクについて，本研究で定義した擬似座標と Web ブラウザによるレンダリング後の座標とを比較する．

本実験ではポータルサイト Yahoo! JAPAN のトップページ(http://www.yahoo.co.jp/)上のリンクを対象にした．レンダリング後の座標は 0 から 1 に収まるように正規化した．

実験の結果を図 3.1 と図 3.2 に示す． 図 3.1 は Web ブラウザ上での X 座標と疑似 X 座標との散布図である．図 3.2 は Web ブラウザ上での Y 座標と疑似 Y 座標の散布図である．実験結果を元に相関係数を求めたところ，X 座標に関しては 0.06，Y 座標に関しては 0.80 であった．また，Web ブラウザ上での表示位置の X 座標が 0.8 以上のものに限定して擬似 X 座標との相関係数を求めると 0.78 となった．

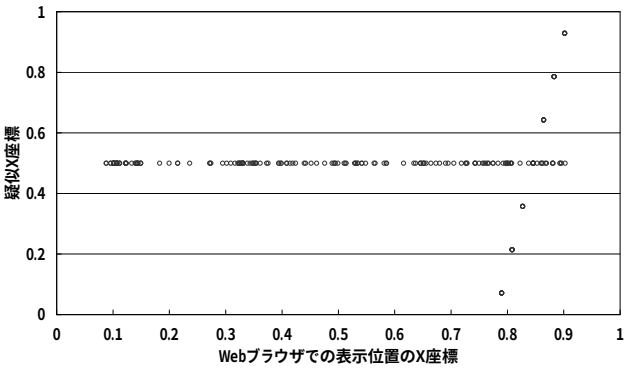

図 3.1 リンクの擬似 X 座標と実際の X 座標

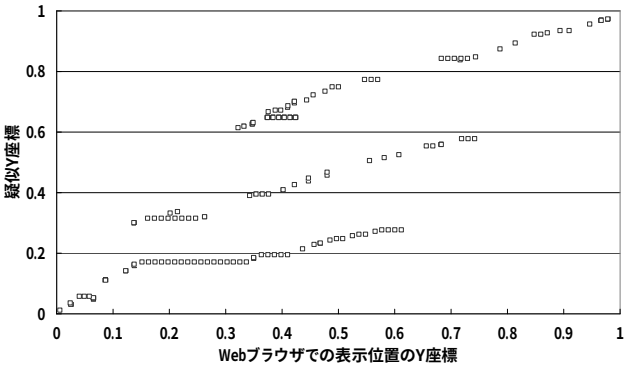

図 3.2 リンクの擬似 Y 座標と実際の Y 座標

4.2.クローリング実験

提案手法で構成するクローラが，既存のクローラと比べてどれだけ効率的に目的の情報を収集できるかを，実際にクローリングを行い検証する．

4.2.1.実験に使用するクローラ

実験には，幅優先探索に基づくクローラ，近似横路手法に基づくクローラ，提案手法により構成される 8 個のクローラの合計 10 種類を用いる．これらのクローラは目的の収集数まで Web 文書を収集したら動作を停止する．

近似横路手法に基づくクローラとは，横路らのクローリング戦略を模したクローラである．それらの異なる点は，位置を特定できる情報の定義である．横路らが行った実験では，位置情報として，緯度経度，住所，最寄り駅，ランドマーク名を利用している．それに対して我々が行う実験では，日本の都道府県名から町名までをつなげた文字列を位置情報として用いる．

4.2.2.クローリング開始 URL 群

提案手法で構築したクローラの汎用性を確認するために，以下の 2 種類の開始 URL 群を利用する．

- 開始 URL 群 A
  - http://www.yahoo.co.jp/
  - http://jp.msn.com/
  - http://www.ocn.ne.jp/
- 開始 URL 群 B

- http://www.goo.ne.jp/
- http://www.livedoor.com/
- http://www.nifty.com/

これらの URL はポータルサイトから選んだ. **ポータルサイト**とはインターネットへアクセスする際の玄関口として作られた Web サイトで, 検索エンジンやリンク集などを持つ. ポータルサイトは, 運営している企業の性質から, 検索エンジン系, Web ブラウザのメーカー系, ネットワークプロバイダ系などに分けることが出来る. 実験のための開始 URL は, このうち検索エンジン系, Web ブラウザ系, ネットワークプロバイダ系で, よく使われているものを選んだ.

4.2.3.分類器の構築

以下のようにして, 提案する手法でクローラに用いる分類器（決定木）を合計 8 個作成した.

まず開始 URL 群 A を利用し, 幅優先探索により 10,000 の Web 文書を収集した. そして収集した Web 文書中の各リンクに対して次の 5 種類の属性を抽出した.

- 属性 a1 : そのリンクのリンク文字列に位置情報があるなら真, そうでなければ偽
- 属性 a2 : Web 文書内のリンク文字列に位置情報があるなら真, そうでなければ偽
- 属性 a3 : 擬似 X 座標
- 属性 a4 : 擬似 Y 座標
- 属性 a5 : リンク先 Web 文書に位置情報があるなら真, そうでなければ偽(教師信号)

そして以下の組み合わせで, 機械学習により分類器 A1 から A4 を作成した.

- 分類器 A1:使用属性 a1, a2, a5
- 分類器 A2:使用属性 a1, a2, a3, a5
- 分類器 A3:使用属性 a1, a2, a4, a5
- 分類器 A4:使用属性 a1, a2, a3, a4, a5

機械学習アルゴリズムには J48 を選択した. **J48** アルゴリズムとは, Weka[3]における C4. 5 アルゴリズムの実装である. 決定木の学習効率が高い汎用的な学習アルゴリズムである ID3 の枝狩り基準を改良したものが C4. 5 アルゴリズムである. Weka はそのような学習アルゴリズムを利用できるデータマイニングツールである.

同様に, 開始 URL 群 B に対しても以下の組み合わせで分類器 B1 から B4 を作成した.

- 分類器 B1:使用属性 a1, a2, a5
- 分類器 B2:使用属性 a1, a2, a3, a5
- 分類器 B3:使用属性 a1, a2, a4, a5
- 分類器 B4:使用属性 a1, a2, a3, a4, a5

作成した分類器を抽出元の Web 文書に適用して分類性能を調べた結果を表 1.1 に示す. 表 1.1 中の割合は分類器が抽出元の Web 文書を正確に分類できた割合である.

表 1.1 分類器の分類性能

学習データ＼使用属性	a1 a2	a1 a2 a3	a1 a2 a4	a1 a2 a3 a4
URL群A	69%（分類器A1）	71%（分類器A2）	71%（分類器A3）	73%（分類器A4）
URL群B	75%（分類器B1）	75%（分類器B2）	76%（分類器B3）	77%（分類器B4）

4.2.4.Web 文書の収集

収集効率の比較のために, 開始 URL 群を変えてクローリングを行った. 実験 A では, 開始 URL 群 A を用いて, 10 種類のクローラで 20,000 ページを収集した. 実験 B では, 開始 URL 群 B を用いて, 同様に 10 種類のクローラで 20,000 ページを収集した.

4.2.5.Web 文書の収集結果

Web 文書の収集結果を, ページ収集効率と収集ページ数のグラフと表で示す. **ページ収集効率**とは以下の式で表される値である.

- （ページ収集効率） = （位置情報を含む Web 文書数）/（収集したすべての Web 文書数）

ページ収集効率が高いものほど効率良く位置情報を含む Web 文書を収集していると言える. 図 4.1, 図 4.2, 表 2.1 は実験 A の実験結果であり, 図 4.3, 図 4.4, 表 2.2 は実験 B の実験結果である.

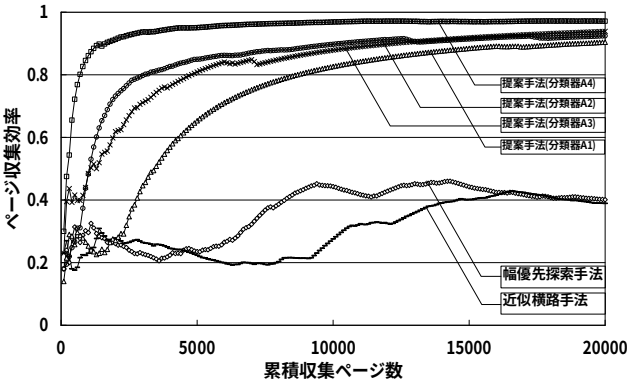

図 4.1 実験結果 A の収集効率のグラフ 1

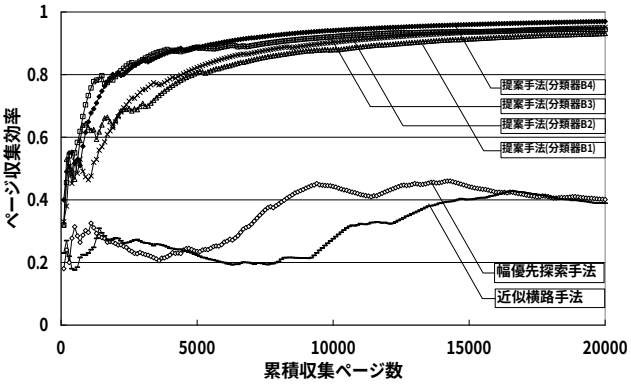

図 4.2 実験結果 A の収集効率のグラフ 2

表 2.1 実験 A のページ収集効率

累積収集数 / 収集手法	5000 ページ	10000 ページ	15000 ページ	20000 ページ
幅優先探索手法	0.235	0.443	0.442	0.401
近似横路手法	0.224	0.276	0.402	0.389
提案手法(分類器A1)	0.653	0.826	0.884	0.904
提案手法(分類器A2)	0.809	0.879	0.918	0.939
提案手法(分類器A3)	0.849	0.897	0.914	0.924
提案手法(分類器A4)	0.950	0.968	0.970	0.971
提案手法(分類器B1)	0.807	0.880	0.914	0.932
提案手法(分類器B2)	0.823	0.903	0.935	0.951
提案手法(分類器B3)	0.888	0.921	0.937	0.944
提案手法(分類器B4)	0.886	0.941	0.960	0.970

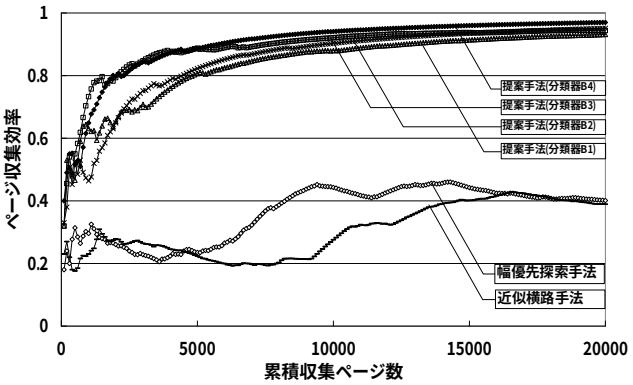

図 4.3 実験結果 B の収集効率のグラフ 1

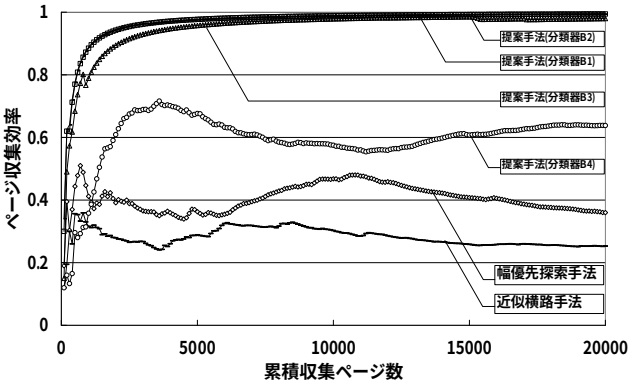

図 4.4 実験結果 B の収集効率のグラフ 2

表 2.2 実験 B のページ収集効率

累積収集数 / 収集手法	5000 ページ	10000 ページ	15000 ページ	20000 ページ
幅優先探索手法	0.363	0.468	0.408	0.360
近似横路手法	0.289	0.303	0.258	0.253
提案手法(分類器A1)	0.683	0.837	0.889	0.916
提案手法(分類器A2)	0.840	0.920	0.947	0.960
提案手法(分類器A3)	0.844	0.919	0.946	0.960
提案手法(分類器A4)	0.561	0.516	0.559	0.590
提案手法(分類器B1)	0.977	0.989	0.992	0.994
提案手法(分類器B2)	0.975	0.988	0.992	0.994
提案手法(分類器B3)	0.958	0.979	0.985	0.980
提案手法(分類器B4)	0.676	0.575	0.608	0.638

5.実験結果の評価

実験 A では，幅優先探索手法と近似横路手法の最終的な収集効率が約 4 割であるのに対し，提案手法で構築したクローラのそれはどれも 9 割を超えている．実験 B でも，幅優先探索手法と近似横路手法の最終的な収集効率はそれぞれ 4 割にも満たず，あまり高くない．A4 と B4 を除いて，提案手法で構築したクローラの収集効率は 9 割を超えている．A4 と B4 の収集効率がそれぞれ約 6 割と約 6 割 4 分なのは，表 1.1 の分類器の分類性能の値が高いことから，過学習による原因が考えられる．

実験 A，B の結果では，リンク文字列に位置情報があるか無いか（属性 a1，a2）に注目しただけでも，機械学習によって構築された分類器は高い収集効率を実現した．我々が提案した疑似座標のどちらか一方だけを用いた場合は，それよりもさらに高い収集効率をあげた．

ところで，疑似 Y 座標と実際の Y 座標との相関が高いのに対して，疑似X座標と実際の X 座標との相関は低い．そのような疑似 X 座標も収集効率が高いクローリング戦略の実現に貢献する理由は，Web 文書内で部分的に用いられているテーブル要素に対して分類器がうまく働いたためと考えられる．これは 4.1.節の検証結果の，相関係数を求める範囲を実際の X 座標の 0.8 以上とした場合で相関が高かったことからもそのように予想される．

6.まとめ

本研究では，位置情報を含む Web 文書を収集する場合，既存の研究で用いられていたリンク文字列に加えて Web 文書中のリンクの表示位置を考慮することを提案した．リンクの表示位置のために，本研究では擬似座標を定義した．また本研究では，機械学習を用いてクローリング戦略を構築することにした．実験により，リンク文字列と擬似座標とを使った統計的な判断は有効であることを確かめた．ただし，過学習が起こる場合があるので，擬似座標の属性をすべて適用したものが最適であるとは限らないことが分かった．

今後の研究課題を 2 つ述べる．1 つめは，近年の Web 文書レイアウトで多く利用されているカスケーディングスタイルシート(CSS)を考慮して，リンク表示位置を計算することである． 本研究で提案した疑似 X 座標と実際の X 座標との相関は高くなかった．CSS を考慮することでより相関の高い座標を計算でき，有効なクローリング戦略の構築に役立つ可能性がある．もう 1 つは，分類器の構築に URL 中のホスト名を利用することである． 我々が収集した Web 文書の URL と位置情報の有無とを調べたところ，ホスト名によって位置情報を含む量に偏りがみられた．位置情報を大量に持つ

ホスト名の特徴をうまく抽出できれば，より収集効率の高いクローリング戦略が構築できる可能性がある．

謝　辞

本研究は科研費 若手研究(B) 18700035 の助成をうけたものである．

参　考　文　献

[1] Google Maps http://maps.google.co.jp/

[2] 横路誠司 高橋克己 三浦信幸 島健一，”位置指向の情報の収集，構造化および検索手法”，情報処理学会論文誌，vol.41，No.7，pp.1987-1998，July 2000．

[3] 元田浩 津本周作 山口高平 沼尾正行，データマイニングの基礎, pp.247-252, (株)オーム社, 東京, 2006．