XPathと包含パターンを利用したWebからの関連語抽出

伊藤　淳†　筧　捷彦††

† 早稲田大学大学院 基幹理工学研究科 情報理工学専攻　〒169–8555 東京都新宿区大久保 3–4–1
†† 早稲田大学理工学術院　〒169–8555 東京都新宿区大久保 3–4–1
E-mail: †ito-jun@kake.info.waseda.ac.jp, ††kakehi@waseda.jp

あらまし　本研究では，Web 上の HTML 文書から関連語の抽出を行う．ここで取り上げる関連語とは，システムに入力されたいくつかの単語と同じカテゴリに属する語のことを指す．我々は HTML 文書の DOM に着目し，入力された単語が含まれる DOM ノードと同じパスを指定する XPath と，XPath で得られた DOM ノードから関連語を抽出するための包含パターンを利用した抽出法を提案する．また，取得した HTML 文書中のどのタグに関連語が登場しやすいかといった知見も提示する．

キーワード　Web Data Mining，Set Expansion, Information Retrieval, XPath, DOM

Related Words Extraction from the Web using XPath and Inclusion Pattern

Jun ITO† and Katsuhiko KAKEHI††

† Department of Computer Science and Engineering,
Graduate School of Fundamental Science and Engineering, Waseda University
3–4–1 Okubo, Shinjuku, Tokyo, 169–8555 Japan
†† Faculty of Science and Engineering, Waseda University
3–4–1 Okubo, Shinjuku, Tokyo, 169–8555 Japan
E-mail: †ito-jun@kake.info.waseda.ac.jp, ††kakehi@waseda.jp

1. 序　論

本研究は，ユーザから与えられたいくつかの単語（シード）をもとに，それらの関連語を Web から自動的に抽出することを目的とする．ここで登場する関連語とは，シード（e.g., toyota, ford, benz）と同一のカテゴリ（e.g., car maker）に属する単語（e.g., honda, peugeot, chrysler）のことを指す．

コーパスから抽出パターンを学習し，固有表現や関係を抽出する研究は古くから行われてきた．クエリログをコーパスとした固有表現抽出では，小町ら [4]，伊藤ら [5] の研究がある．また，Web をコーパスとした関係抽出では，Cafarella ら [6] の KnowItNow，Pantel ら [7] の Espresso がある．

我々は，Web をコーパスとした関連語抽出を行う Wang ら [1, 2, 3] の研究に着目し，その抽出手法をより Web コーパスに適したものにするべく改良を行った．

2. Set Expander for Any Language

Wang らが提案した Set Expander for Any Language（SEAL）は，ユーザから与えられたシードをもとに，文字レベルのヒューリスティクスを用いてパターンを抽出する．is-a などの関係性や形態素解析を必要とせず，シードの左右に現れる文字列のみを用いてパターンを抽出するため，教師データを必要としないうえ，言語非依存であることが特徴となっている．また，コーパスとなる文書がマークアップ言語（e.g., html, xml, tex）であるか，そうでない（e.g., txt, doc）かにもよらない．

本章では SEAL におけるパターン抽出アルゴリズムとその問題点について述べる．

2.1 パターン抽出アルゴリズム

2.1.1 Web からの文書取得

コーパスとなる文書は，検索エンジン API を利用して Web から取得される．クエリはユーザから入力されたシード（e.g., toyota, ford, benz）をもとに構成される．その際，それぞれのシードがダブルクォートされたうえで半角スペースによって連結される（e.g., “toyota” “ford” “benz”）．すなわち，個々のシードでフレーズ検索が行われ，かつ AND 検索される．

問い合わせ結果の上位 N 文書を取得し，個々の文書においてパターンを抽出する．パターンはその文書においてのみ，抽出・適用される．

2.1.2 パターンの定義

パターンは，シード s を挟み込む左文字列 l と右文字列 r の組 $\langle l, r\rangle$ によって定義される．したがって，文書 D の中にすべてのシード s について，$lsr \in D$ を満たす文字列が存在するとき，パターンは抽出される．また，$\langle l, r\rangle$ で抽出される文字列集合を C，その要素を $c \in C$ とすると，次の制約を満たさなければならない．

（1） すべてのシード s において，s と等しい c が存在する．
（2） (1) を満たすような l の接尾辞文字列 l' と r の接頭辞文字列 r' が存在しない．

例えば，表 1 のサンプル HTML において，上の制約を満たすような左右文字列は共に “`</li>\r\n  <li>`” である．したがって，“`</li>\r\n  <li>[...]</li>\r\n  <li>`” がパターンとして定まる．なお，`[...]` は任意の文字列が入ることを示し，この部分にあてはまる単語が抽出されることになる．このパターンを文書に適用することによって，シードに含まれていない peugeout が新たに抽出される．

表 1 サンプル HTML

seed = { toyota, ford, benz }

```
...
<ul>
  <li>honda</li>
  <li>toyota</li>
  <li>ford</li>
  <li>benz</li>
  <li>peugeot</li>
  <li>chrysler</li>
</ul>
...
```

2.1.3 左右文字列の発見

制約を満たす左右文字列は，シード番号でラベル付けされたパトリシア木を構築することで発見される．

パトリシア木は左文字列と右文字列で別々に用意する．文書の開始からシードのすぐ左までの文字列を左文字列パトリシア木に，シードのすぐ右から文書の終了までの文字列を右文字列パトリシア木にそれぞれ追加する．このとき，左文字列は反転したうえで追加される．また，ノードを追加する際に，左右文字列と隣接するシードの番号がラベルとして付与される．

表 1 のサンプル HTML におけるパトリシア木の状態を図 1 に示す．この状態から，次の制約を満たす左文字列パトリシア木のノード N_l と右文字列のパトリシア木のノード N_r の組を発見する．ここで，N_l が持つラベル集合を $Label(N_l)$，N_r が持つラベル集合を $Label(N_r)$ とする．

(i) ラベル集合 $Label(N_l) \cap Label(N_r)$ がすべてのシードを少なくともひとつ含む．
(ii) (i) を満たすような親ノードが存在しない．

この制約を満たす N_l と N_r の文字列を取り出すことで，すべてのシードを少なくともひとつ囲む最大長の左右文字列の組が抽出される．なお，左文字列は反転して取り出される．したがって，2.1.2 で示した “`</li>\r\n  <li>[...]</li>\r\n  <li>`” がパターンとなる．

ただし，この説明は簡略化したものであり，実際は文字列長の最小値を指定できるようなアルゴリズムになっている．詳しくは Wang らの論文を参照されたい [1, 2].

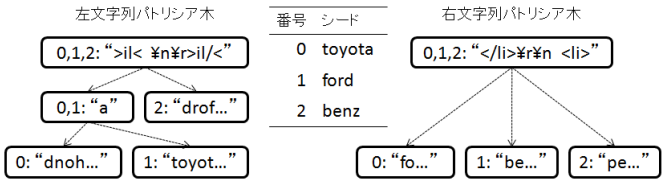

図 1 サンプル HTML におけるパトリシア木

2.2 問 題 点

2.2.1 パトリシア木の肥大化

2.1.3 で述べたとおり，シードが発見されると文書の開始からシードのすぐ左までの文字列と，シードのすぐ右から文書の終了までの文字列がパトリシア木に追加される．シードが発見されるたびにかなり長い文字列がパトリシア木に追加されるため，メモリ効率に不安がある．

実際にどのような左右文字列が最終的にパターンとして抽出されているのかを知るために実験を行った．その結果が表 2 および図 2 である．

表 2 パターンにおける文字列長の期待値と最大値

	文字列長期待値	最大文字列長
左文字列	8.978	6199
右文字列	17.472	1773

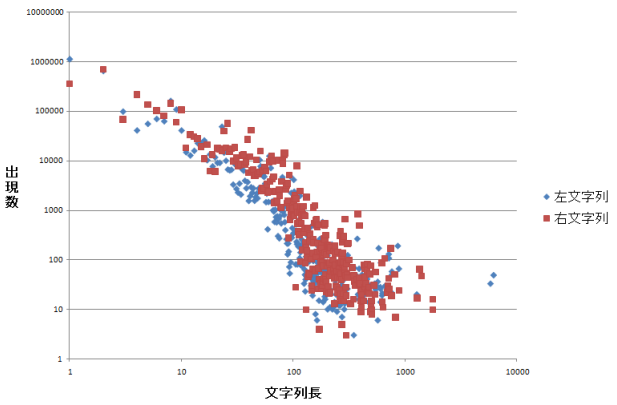

図 2 パターンの文字列長と出現数

表 2 を見ると分かる通り，文字列長の期待値はたかだか 20 文字程度である．パトリシア木に追加された文字列の大部分は

パターン抽出に関与していない．

図 2 は文字列長の分布を示している．文字列長は 100 文字を境に出現数が 100 分の 1 程度まで減少していることが分かる．ほとんどのパターンは 100 文字以下の左右文字列で構成されており，1000 文字以上になると出現数は非常に少なくなる．

これらの結果から，文書の先頭や末尾までの長い文字列をパトリシア木に追加する必要はなく，1000 文字程度で打ち切りを行っても多くの場合で支障がないことが分かる．文書の先頭や末尾までの長い文字列は冗長であると言える．

2.2.2 リスト形式の最前・最後列要素の抽出漏れ

表 1 のサンプル HTML を見ると，左文字列を “
”，右文字列を “” とし，“
[...]” をパターンとするのが自然である．しかし，これは SEAL のパターン制約 (2) に抵触するためパターンとはならず，honda や chrysler を抽出することはできない．

これから分かる通り，抽出したい単語がリストで出現している場合，最前・最後列の要素が漏れる可能性がある．これはシードが最前・最後列に出現していないときに起きる．つまり，シードの出現位置に依存してパターンの抽出が可能かどうかが左右される．

3. XPath と包含パターンによる抽出

SEAL のパターン抽出はマークアップ言語であるかによらないという特徴があるが，実際に抽出されたパターンを見るとHTML のタグを含むものが多い．Wang らの報告によれば，抽出されたパターンの 80〜90 %がタグを含むそうだ．

Web 上の多くの文書が HTML 文書であることと，パターンにタグが含まれることが多いことから，HTML 文書に対して効果的な方法で抽出を行うのが良いのではないかと考えた．そこで本研究では，HTML 文書の DOM に着目した関連語抽出手法を提案する．

3.1 XPath を利用した候補文字列の抽出

XML Path Language（XPath）とは，XML 文書の特定の部分を指定するための言語構文である．ただし，構文は XML 文書に特化したものではないため，DOM で表すことができれば HTML 文書にも適用することができる．

提案手法では，まず XPath を利用し，シードが現れた DOM ノードと同じパスに存在するノード群を抽出する．例えば，表 1 のサンプル HTML 文書における ul タグが body タグ直下に出現していたとすると，シードが現れた li タグのテキストノードを指定する XPath は，“/html/body/ul/li/text()” で表される．これを DOM に適用することで，honda，toyota，ford，benz，peugeot，chrysler を抽出することができる．

しかし，これにより取得された文字列はあくまで関連語を含んでいるかもしれない文字列にすぎない．関連語は，“list of honda cars” などのように部分文字列として現れることもあるし，そもそも含まれていない可能性もある．リンク URL を指定する XPath（e.g., “/html/body/div/a/@href”）が抽出された場合などは，シード文字列が URL の一部に含まれている（“http://example.com/car-maker/honda.html”）こともある．

したがって，XPath によって抽出される文字列は，関連語そのものでなく，あくまで関連語を含んでいるかもしれない候補文字列である．

3.2 包含パターンを利用した関連語の抽出

XPath により候補文字列が抽出されたら，そこから包含パターンを使用して関連語の抽出を行う．包含パターンとは，SEAL におけるパターンを文書全体でなく候補文字列に適するよう変更を加えたものである．具体的には以下の 2 点が異なる．

(a) 候補文字列は\0x02 および\0x03 でラップされる．
(b) XPath ごとに包含パターンは抽出・適用される．

(a) で候補文字列をアスキー文字の\0x02 および\0x03 で囲うのは，表 1 のように候補文字列と関連語が等しい場合に，挟み込む文字列がなくなってしまうことを防ぐためである．\0x02 および\0x03 でラップすることにより，どの場合においても同じ手法をあてはめることができる．

(b) によって 2.2.2 で指摘した抽出漏れが起きる可能性が低くなる．適用範囲を XPath ごとに閉じることで，2.2.1 のように長い文字列をパトリシア木に追加する必要がない．また，“\0x02[...]\0x03” のように候補文字列を関連語としてそのまま抽出する包含パターンが，別の XPath で抽出された候補文字列に適用されることで不必要な単語が抽出されることもない．

4. 評価実験

4.1 実験設定

SEAL と XPath による関連語の抽出精度を調べるために実験を行った．実験データセット（表 3）は Wang らが使用したものに最新データを反映して用いている．また，中国語データセットに関しては簡体字表記に変更した．

実験データセットは英語，日本語，中国語の各言語ごとに 12 カテゴリが用意され，各言語で共通のカテゴリが 9 つ存在する．各カテゴリではシードとして使用される単語と，その同義語が記述されている．クエリの生成はシードのみが利用されるが，関連語が正しく抽出できたかは同義語も含めて判断される．

実験は各カテゴリで 3 回行い，抽出した関連語が正解である割合（適合率），カテゴリで記述されている単語を網羅している割合（再現率），および F 値のそれぞれについて平均値を算出した．1 回の試行ごとにシードをランダムに 3 つ選択してクエリを構成し，1 つのクエリにつき最大で検索結果の上位 100 文書を取得した．また，検索エンジン API は Bing API(注1)を，HTML パーサには TagSoup(注2)を使用した．

4.2 実験結果

4.2.1 各カテゴリにおける関連語の抽出精度

表 5 を見ると分かる通り，XPath による抽出の適合率が高い．これは，XPath ごとに閉じた抽出によって不必要な単語の抽出を抑えられたためである．SEAL のパターンは文書全体に

(注1)：http://www.bing.com/developers
(注2)：http://home.ccil.org/~cowan/XML/tagsoup

表 3　実験データセット

	データ名	英語	日本語	中国語	説明
1	disney-movies	✓	✓	✓	ディズニー映画
2	constellations	✓	✓	✓	星座
3	countries	✓	✓	✓	世界の国
4	mlb-teams	✓	✓	✓	MLB チーム
5	nba-teams	✓	✓	✓	NBA チーム
6	nfl-teams	✓	✓	✓	NFL チーム
7	car-makers	✓	✓	✓	自動車製造業
8	us-presidents	✓	✓	✓	米国大統領
9	us-states	✓	✓	✓	米国州
10	cmu-buildings	✓			メロン大学の建物
11	diseases	✓			病気
12	periodic-comets	✓			彗星
13	japan-emperors		✓		天皇
14	japan-prime-mins		✓		総理大臣
15	japan-provinces		✓		都道府県
16	china-dynasties			✓	中国王朝
17	china-provinces			✓	中国行政区分
18	taiwan-cities			✓	台湾行政区分

表 4　XPath による抽出の妥当性検証

	適合率	再現率	F 値
兄弟	0.404	0.191	0.320
従兄弟	0.341	0.093	0.233
兄弟&従兄弟	0.456	0.284	0.344
XPath	0.475	0.475	0.490

適用されるので，“>[...]<” のようなパターンが抽出されると途端に適合率が低下する．一方で，再現率はどちらの手法も 1 に近く，ほぼカテゴリに含まれる単語を網羅していることが分かった．

適合率が XPath の場合においても 0.5 程度と高くないことから，関連語の精査が必要である．Wang らはこれを Random Walk with Restart [8] というアルゴリズムを適用することで行っている．我々も同様の実験を行っているが，本論文は抽出手法のみを紹介するものであるためここでは扱わない．

4. 2. 2　XPath による抽出の妥当性

表 4 はシードと兄弟，従兄弟，兄弟と従兄弟関係にあるノードと，XPath で取得されたノードについて，関連語を含んでいる割合（適合率）と，文書内に含まれる関連語がどれだけ網羅されているか（再現率）を調べた結果である．XPath によって同じパスに存在するノード群を抽出すると，関連語を含まないノードが取得される可能性が高まる．一方で，シードが現れたノードと兄弟関係にあるノードや，従兄弟関係にあるノードに限定すると，網羅性が落ちる可能性が高まる．したがって，どの範囲までのノードを取得するのが妥当であるのかについて検証した．

表 4 を見ると分かる通り，XPath による抽出法が最も精度が良い．XPath による抽出法を適用すると，1 文書中から取得した候補文字列の約半数が関連語を含み，またそのような候補文字列は文書内の約半数の関連語を含むことが分かる．ある文書では兄弟ノードが有効だが別の文書では従兄弟ノードが有効であるといったように文書ごとに特性が異なるため，全体として見てみるとノード取得範囲が広い方が安定した精度が出るようである．

表 5　実 験 結 果

	SEAL			XPath		
英語	適合率	再現率	F 値	適合率	再現率	F 値
disney-movies	0.216	1.000	0.355	0.338	1.000	0.505
constellations	0.243	1.000	0.391	0.656	1.000	0.792
countries	0.679	1.000	0.809	0.784	1.000	0.879
mlb-teams	0.085	1.000	0.156	0.601	1.000	0.751
nba-teams	0.132	1.000	0.233	0.388	1.000	0.559
nfl-teams	0.398	1.000	0.570	0.607	1.000	0.756
car-makers	0.358	1.000	0.527	0.539	1.000	0.700
us-presidents	0.198	1.000	0.331	0.466	1.000	0.636
us-states	0.222	1.000	0.363	0.638	1.000	0.779
cmu-buildings	0.078	0.833	0.142	0.069	0.900	0.128
diseases	0.123	1.000	0.219	0.169	1.000	0.289
periodic-comets	0.509	1.000	0.675	0.575	1.000	0.730
平均	0.270	0.986	0.398	0.486	0.992	0.625
日本語	適合率	再現率	F 値	適合率	再現率	F 値
disney-movies	0.170	1.000	0.290	0.435	1.000	0.606
constellations	0.561	1.000	0.718	0.731	1.000	0.845
countries	0.725	0.995	0.839	0.752	1.000	0.859
mlb-teams	0.234	1.000	0.380	0.390	1.000	0.561
nba-teams	0.203	1.000	0.337	0.468	1.000	0.638
nfl-teams	0.251	1.000	0.401	0.526	1.000	0.689
car-makers	0.413	1.000	0.584	0.526	0.982	0.685
us-presidents	0.150	1.000	0.261	0.230	1.000	0.375
us-states	0.539	1.000	0.700	0.532	1.000	0.695
japan-emperors	0.495	1.000	0.663	0.490	1.000	0.658
japan-prime-mins	0.155	1.000	0.269	0.497	1.000	0.664
japan-provinces	0.540	1.000	0.702	0.828	1.000	0.906
平均	0.370	1.000	0.512	0.534	0.999	0.682
中国語	適合率	再現率	F 値	適合率	再現率	F 値
disney-movies	0.055	1.000	0.105	0.053	1.000	0.101
constellations	0.282	1.000	0.440	0.277	1.000	0.434
countries	0.614	0.990	0.758	0.745	0.985	0.849
mlb-teams	0.338	1.000	0.505	0.353	1.000	0.522
nba-teams	0.197	1.000	0.329	0.777	1.000	0.875
nfl-teams	0.247	1.000	0.396	0.529	1.000	0.692
car-makers	0.348	0.964	0.512	0.375	0.964	0.540
us-presidents	0.257	1.000	0.409	0.500	1.000	0.666
us-states	0.427	1.000	0.598	0.559	1.000	0.717
china-dynasties	0.290	1.000	0.450	0.376	0.988	0.545
china-provinces	0.489	1.000	0.657	0.485	1.000	0.653
taiwan-cities	0.311	1.000	0.474	0.525	1.000	0.689
平均	0.321	0.996	0.469	0.463	0.995	0.607

4. 2. 3　XPath により抽出されるタグの傾向

XPath でどのようなタグが抽出されたのかを表 6，7 に示した．候補文字列が関連語を含む場合を正解として扱っている．

表 6 は正解総数が多い順に並べたものである．リンクを扱う A タグが正解総数の 1, 3, 4 位に登場しており抽出において

表 6　XPath で抽出したタグ（正解数順）

	タグ	正解数	適合率
1	a/text()	154674	0.364
2	option/text()	109641	0.597
3	a/@href	52644	0.482
4	a/@title	45858	0.409
5	td/text()	36355	0.318
6	div/text()	33061	0.125
7	option/@value	28883	0.562
8	p/text()	26810	0.318
9	font/text()	18467	0.194
10	span/text()	9794	0.185

表 7　XPath で抽出したタグ（適合率順）

	タグ	正解数	適合率
1	noscript/text()	252	1.000
2	span/@title	85	1.000
3	td/@alt	75	1.000
4	td/@title	75	1.000
5	b/@title	90	0.989
6	fieldset/text()	54	0.982
7	table/@summary	123	0.976
8	h1/text()	71	0.959
9	area/@alt	341	0.953
10	span/@onclick	176	0.931

表 8　抽出できなかった正解要素を含むタグ

	タグ	出現数	出現率
1	a/@href	76975	0.326
2	a/text()	41206	0.174
3	td/text()	24761	0.105
4	a/@title	16839	0.071
5	span/@style	15072	0.064
6	p/text()	8082	0.034
7	font/text()	6881	0.029
8	span/text()	4973	0.021
9	div/text()	4223	0.018
10	img/@alt	3580	0.015

重要なタグであることがうかがえる．また，2, 7 位に現れる OPTION タグは Web ページにおける入力フォームで用いられるタグであり，関連語がリストとして定義されやすいと言える．これはテーブル構造を扱う 5 位の TD タグについても同様のことが言える．6, 8 位の DIV, P タグは文章を記述するタグであるため関連語が含まれることが多かったと思われる．また，9, 10 位の FONT, SPAN タグは関連語を強調するために使用されたものと思われる．

表 7 は適合率が高い順に並べたものである．テキストノードではなく属性ノードが多いことが見て取れる．また，テキストノードも DIV, SPAN, P タグのような一般的なタグではなく，NOSCRIPT や FIELDSET などの一般的でないタグが上位に含まれている．H1 タグは見出しタグとして SEO（Search Engine Optimization）対策においても重要視されているため，関連語を含む確率が高いのかもしれない．

4. 2. 4　XPath により抽出されないタグの傾向

表 8 は関連語を含んでいたが，XPath による抽出から漏れたノードを出現数順に並べたものである．表 6 と同様に A タグが上位に位置しており，抽出漏れしたノード全体の約半数を占めている．したがって，XPath と同時に A タグをすべて抽出するといった手法も考えられ，検討の余地がある．また，全体的に表 6 と似たタグが良く登場しており，関連語が登場するタグには傾向があることが分かる．

5. 結　　論

本研究では，SEAL におけるパターン抽出アルゴリズムを HTML 文書に応用させたものを提案した．XPath と包含パターンを用いることにより，リスト形式で定義された関連語の抽出漏れを抑止している．また，XPath ごとに閉じた包含パターンを適用することで，“>[...]<” のようなタグにより不必要な単語が多く抽出されることも抑止している．

提案手法により，関連語の抽出精度は F 値で平均 0.178 向上した．ただし，適合率はまだ 0.5 前後の値であるため，SEAL と同様に，抽出された関連語をランキングアルゴリズムによって精査することが必要となる．

また，XPath による抽出が兄弟・従兄弟などの他の関連性を利用した抽出よりも精度が高いことも示した．さらに，XPath によって抽出されるタグとそれに含まれている関連語の割合や，XPath で抽出できない関連語がどのタグに含まれていたかなどを検証したところ，A タグに関連語が出現しやすいという結果となったことを示した．

これらの結果を受け，今後は XPath だけではなく関連語の抽出に有効なタグについても利用ができないか研究を進めたい．

文　　献

[1] Richard C. Wang, William W. Cohen, Robert E. Frederking, Tom M. Mitchell and Fernando C. N. Pereira, “Language-Independent Class Instance Extraction Using the Web”, In CMU SCS Technical Report Series (CMU-LTI-09-020), 2009.

[2] Richard C. Wang and William W. Cohen, “Character-Level Analysis of Semi-Structured Documents for Set Expansion”, In Proceedings of EMNLP, 2009.

[3] Richard C. Wang and William W. Cohen, “Language-Independent Set Expansion of Named Entities using the Web”, In Proceedings of ICDM, 2007.

[4] 小町守, 鈴木久美, “検索ログからの半教師あり意味知識獲得の改善”, 人工知能学会論文誌, 23 巻 3 号, pp. 217-225, 2008.

[5] 伊藤 淳, 戸田 浩之, 廣嶋 伸章, 望月 崇由, 鈴木 智也, 筧 捷彦, “クエリログをコーパスとした意味知識獲得法の改善”, In Proceedings of DEIM, 2010.

[6] M. J. Cafarella, D. Downey, S. Soderland, and O.Etzioni, “ KnowItNow: Fast, Scalable Information Extraction from the Web ”, In Proceedings of EMNLP, 2005.

[7] Patrick Pantel and Marco Pennacchiotti, “Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations”, Proceedings of ACL, pp. 113-120, 2006.

[8] Hanghang Tong, Christos Faloutsos and Jia-Yu Pan, “Fast Random Walk with Restart and Its Applications”, In Proceedings of ICDM, 2006.