社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DBS－146 (7)
2008／9／21

両方向構文パターンを用いた Web検索エンジンからの高速関連語発見手法

大島 裕明† 田中 克己†

† 〒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社会情報学専攻
E-mail: †{ohshima,tanaka}@dl.kuis.kyoto-u.ac.jp

あらまし 本稿では，1語が与えられたときに，Web検索を利用して，与えられた語とある特定の関係にある語（以後，関連語と呼ぶ）を発見する手法について提案を行う．関連語とは，例えば，上位語，下位語，類義語，同位語などのことであり，話題語や詳細語のように定義が曖昧な語や，略語なども含むものとする．提案手法では，Web検索エンジンの結果として得られるテキストを情報源として関連語を発見する．検索クエリにおいて，発見したい関連語が特徴的に現れる構文パターンを複数用いることにより，Web検索結果という少ない量のテキストからでも比較的精度良く，高速に語の発見を行うことが可能となる．
キーワード Webマイニング，知識発見

High-speed Extraction of Related Terms by Bi-directional Syntax Patterns from Web Search Engines

Hiroaki OHSHIMA† and Katsumi TANAKA†

† Department of Social Informatics, Graduate School of Informatics, Kyoto University Yoshida-honmachi, Sakyo, Kyoto 606-8501, Japan
E-mail: †{ohshima,tanaka}@dl.kuis.kyoto-u.ac.jp

Abstract We propose a method for discovering related terms using Web search engines. Related terms are, for example, hypernyms, hyponyms, synonyms, coordinate terms, topic terms, and abbreviations. The proposed method uses text resources from Web search results only. Bi-directional syntax patterns are used for Web search queries and for extracting related terms from Web search results. A system using our method has a relatively high speed and good precision.
Key words Web Mining, Knowledge Discovery

1. はじめに

現在，Webを情報源として様々な知識を発見する技術について，広く研究が行われている．そのような技術の中に，1語が与えられたときにある特定の関係にある他の語を発見するというものがある．例えば，与えられた語の上位語，下位語，類義語，同義語，同位語（共通の上位概念を持つ，兄弟概念にあたる語）を発見するようなものである．また，これらのように辞書的に定義される関係にある語ばかりでなく，話題語，詳細語，略語や，人物の呼称や愛称など，様々な関係にある語（以後，関連語と呼ぶ）を発見することが研究されている．

Webを情報源として関連語を発見する手法にも，Webをクロールして大量に収集したテキスト情報を用いるもの，Web検索結果で得られたページ群をダウンロードして用いるもの，Web検索結果に含まれるタイトルやスニペットのテキストだけを用いるものなど，様々なものが存在している．どの程度の量の情報を用いるかは，結果が得られるまでの速度と，精度や得られる語数に影響を与える．一般に，多くの情報を用いるほど，速度はより遅くなるが，精度はより良くなり，得られる語数はより多くなると考えられる．

我々がこれまで行ってきた研究の1つは，Web検索を利用して，与えられた語の同位語を発見するものである[1]．同位語が並列助詞「や」で接続されることに着目し，与えられた語に「や」を接続してWeb検索を行い，得られたタイトルやスニペットのみから同位語を発見する．この手法では，Web検索結果のみを用いることによって，比較的高速に結果を得ることができる．例えば，1語を与えたときに，6語程度の同位語を，3〜5秒程度で返すことができる．精度は70〜80%程度であり，様々なアプリケーションでの応用が可能である．

本稿では，これまで提案してきたWeb検索エンジンからの

同位語発見手法の一般化を行い，比較的高速に Web から様々な関連語を発見する手法について提案を行う．

以降，2 章では関連研究について，3 章で両方向構文パターンを用いた高速関連語発見手法について，4 章で両方向構文パターンの自動的な発見手法について，5 章でまとめと今後の課題について述べる．

2. 関 連 研 究

語の関係性を求めるためには，まず，電子化された辞書や，Web 上の辞書を利用することが考えられる．人手によって作られた電子化辞書には，WordNet [2](注1) や EDR 電子化辞書 [3]，言語工学研究所 デジタル類語辞典（シソーラス）[4] などがある．また，Wikipedia(注2)や Wiktionary(注3)も有名である．これらによって，上位語や下位語などを求めることが可能である．しかし，これらがあらゆる語を網羅することは不可能であり，また，多様な語の関係性には対応しないため，オンデマンドに関連語を発見する技術は必要である．

大規模なテキストコーパスやデータセットを利用する手法には様々なものが存在する．Hearst ら [5] は “such as” のような構文パターンに着目することで，大規模テキストコーパスから上位下位関係を発見する手法を提案した．Ghahramani ら [6] による Bayesian Sets は語の共起テーブルのような大規模なデータから，ベイズ推定を用いて同位語のクラスタを発見することによって，同位語を取得するものである．アルゴリズムはシンプルで高速であるが，EachMovie や Grolier encyclopedia のような大規模なデータセットが必要となる．Lin ら [7] は係り受け解析された大規模テキストコーパスから，係り受け関係を基にして語の類似度を計算することで，類義語のクラスタを作成する手法を提案した．Shinzato ら [8] は，Web 検索エンジンを利用して大量の HTML 文書をクロールし，それらから同位語を発見する手法を提案した．まず，HTML の構造において同レベルに列挙されている語が同位語の候補として取得される．それらの中でも特に語どうしの相互情報量や共起度が高いものを，同位語とする評価手法を提案した．彼らはまた，HTML の構造を利用して上位語や下位語を取得する研究 [9] も行っている．Google Sets (注4) は，Google が提供するサービスの 1 つであり，いくつかの同位語を与えると，それらが属する同位語の一群を結果として返すものである．Google Sets のアルゴリズムは非公開であるが，Google が収集した Web ページに含まれる語に対して大規模なクラスタリングを行い，同位語のクラスタをあらかじめ生成しているようである．山口ら [10] は，大規模なクエリログデータを利用して，同位語を取得する手法を提案した．

他にも様々な関係にある語を発見する研究が行われている．Oyama ら [11] は，与えられた語の詳細語を発見する手法を提案した．ここでの詳細語とは，典型的には part-of 関係にある語であると考えられる．文書の構造として，タイトルと本文という構造に着目し，ある語がタイトルとして使われている文書の本文に詳細語が現れると仮定して，詳細語としての度合いを計算する手法を提案している．野田ら [12] は，与えられた語の話題語を発見する手法を提案した．まず，Web 検索で与えられた語と「の」で接続される語を取得する．そこで得られた語を，Web 検索のヒット件数を利用して，話題語として適切かどうか判定する手法を提案している．中山ら [13] は，Wikipedia のデータを利用して，連想シソーラスを構築する手法を提案した．Wikipedia の各記事間でのリンク関係を考慮して語の関連度を計算することで，ある語から連想される他の語を求めている．外間ら [14] は，Web 検索を用いて，人物の呼称を求める手法を提案した．例えば，「松井秀喜」に対して「ゴジラ」といった呼称を求めるもので，この場合，「こと松井秀喜」というフレーズの直前に現れる語を求めて，その後，呼称としての評価を行う．若木ら [15] は，Web 情報から人物の愛称を求める手法を提案している．

語の関連性についての研究では，2 語の関係性の度合いを計算する研究も多く行われている．Church ら [16] は，2 語がどの程度意味的に関連があるかどうかを，相互情報量を用いて計算する手法について提案した．Turney [17] や，Baroni ら [18] は，Web 検索のヒット件数を用いて，類義語としての度合いを計算する手法を提案した．彼らの手法は，語の共起性や相互情報量を計算するものである．また，Google Similarity Distance [19] も語の類似性を計算する手法である．また，Turney ら [20], [21] や Bollegala ら [22] は，2 語が与えられたときに，それがある特定の関係性を満たすかどうかという，関係性の度合いを計算する手法について提案を行っている．取り扱われる関係性は固定されておらず，2 語のペアを与えることによって，様々な関係性を扱うことが可能となっている．

3. 両方向構文パターンを用いた関連語発見

3.1 Web 検索エンジンからの高速な同位語発見手法

本節では，我々が提案してきた同位語発見手法 [1] の一般化について述べる．まず，高速な関連語発見手法の一例として，提案してきた同位語発見手法について述べ，どのような特徴を持っているかを明らかにする．

同手法では，まず，同位語が並列助詞「や」で接続されることに着目し，与えられた語に「や」を接続して Web 検索を行う．例えば，与えられた語が「バッハ」であった場合には，「”やバッハ”」というクエリと，「”バッハや”」というクエリで Web 検索を行い，それぞれ 100 件の検索結果を取得する．ダブルクォーテーション（”）で括っているのは，Web 検索エンジンのフレーズ検索機能を用いるためである．ここで，検索結果のタイトルやスニペットに，「彼女はベートーベンやバッハのような…」という文と「バッハやベートーベンって生前から…」という文が含まれていたとする．この場合，「ベートーベン」という語が，「やバッハ」という構文パターンの直前に出現しており，また，「バッハや」という構文パターンの直後にも出現していることがわかる．このような場合に，「ベートーベン」という語を

(注1)：http://wordnet.princeton.edu/
(注2)：http://wikipedia.org/
(注3)：http://wiktionary.org/
(注4)：http://labs.google.com/sets

「バッハ」の同位語であると判定するのが提案手法であった．

この手法では，与えられた語を含む構文パターンを 2 つ利用している．与えられた語を $< query >$ と表し，発見したい語が現れる部分を $< target >$ と表すことにする．本手法で用いている構文パターンは以下の 2 つである．

（1） や $< query >$

（2） $< query >$ や

ともに「や」という語が用いられているため，同一視したくなるが，これらは明確に別のものとして考える必要がある．

これらの構文パターンを観察すると，(1) では $< target >$ がこの構文パターンの前に出現することが期待され，(2) では $< target >$ がこの構文パターンの後に出現することが期待されるのがわかる．このように，$< target >$ が前後に出現することが期待される 2 種類の構文パターンを用いることによって，発見したい語を正しく切り出すことが可能となる．すなわち，先ほどの例としてあげた 2 つの文において，この 2 つの構文パターンにあてはまる文字列は，「ベートーベン」だけであり，その一部である，「ベート」や「ベン」は，どちらか片方の構文パターンにはあてはまるが，もう片方にはあてはまらないことがわかる．このように，形態素解析を行うこともなく，正しい語の切り出しを行うことができるのが，このような 2 種類の構文パターンを用いる利点である．これらの構文パターンを両方向構文パターンと呼ぶこととする．

両方向構文パターンのうち，$< target >$ が前に出現する構文パターンを Pre パターン（$\mathbf{Patterns}^{Pre}$ と表記），$< target >$ が後に出現する構文パターンを Post パターン（$\mathbf{Patterns}^{Post}$ と表記）と呼ぶこととする．

3.2 両方向構文パターンを用いた関連語発見手法の流れ

ここでは，両方向構文パターンを用いた高速な関連語発見手法の流れについて述べる．

（1） $< query >$ が与えられる

（2） $< query >$ に対するいくつかの Pre パターン $\mathbf{Patterns}^{Pre}$ といくつかの Post パターン $\mathbf{Patterns}^{Post}$ を作成する．

（3） Pre パターンのための Web 検索クエリ $webQuery^{Pre}$ と，Post パターンのための Web 検索クエリ $webQuery^{Post}$ を作成する．

（4） $webQuery^{Pre}$ によって Web 検索を行い，検索結果のタイトルとスニペットから文の集合 $\mathbf{Sentences}^{Pre}$ を取得する．同様に，$webQuery^{Post}$ から $\mathbf{Sentences}^{Post}$ を取得する．

（5） $\mathbf{Sentences}^{Pre}$ の中で，各 $\mathbf{Patterns}^{Pre}$ の直前に出現するあらゆる文字列を取得する．$\mathbf{Sentences}^{Pre}$ のうち，ある文字列 $\mathbf{t}_i$ が Pre パターンのいずれかにあてはまるようなものの数を，$Count^{Pre}(\mathbf{t}_i)$ とする．

（6） $\mathbf{Sentences}^{Post}$ の中で，各 $\mathbf{Patterns}^{Post}$ の直後に出現するあらゆる文字列を取得する．$\mathbf{Sentences}^{Post}$ のうち，ある文字列 $\mathbf{t}_i$ が Post パターンのいずれかにあてはまるようなものの数を，$Count^{Post}(\mathbf{t}_i)$ とする．

（7） $Count^{Pre}(\mathbf{t}_i)$ と $Count^{Post}(\mathbf{t}_i)$ をもとに，$\mathbf{t}_i$ の評価値 $Value(\mathbf{t}_i)$ を決定する．

（8） ストップワードや $Value(\mathbf{t}_i)$ が閾値以下の語を取り除き，残りを結果として出力する．

まず，(1)$< query >$ が与えられ，それをもとにして，(2) 両方向構文パターンを作成する．構文パターンには必ずしも $< query >$ を含む必要はない．また，Pre パターン，Post パターンともに，複数であってもかまわない．次に，(3)Web 検索クエリを作成する．これらの Web 検索クエリには，必ずどこかに $< query >$ が含まれる．それらを基に，(4)Web 検索を行い，タイトルとスニペットのテキスト情報を収集する．通常，それぞれの Web 検索クエリに対して 100 件の検索結果を取得し，その場合，本手法によって関連語を取得するのにかかる時間は，3〜5 秒となる．なお，実装では，Yahoo!検索 Web サービス(注5)を利用している．得られたテキストは，スペースや記号を取り除き，英数字を半角にするなどのテキストクリーニングを行う．次に，(5)(6) 各構文パターンの直前や直後に現れる文字列を取得する．ただし，あらかじめ取得する文字列の字数の最大値を決めておく．日本語の場合，15 語程度としておくと十分である．Web 検索結果には同じ文章が複数回出現することがあったり，同じ文で複数の構文パターンにあてはまるような文字列があったりする可能性があるため，各構文パターンにおいて発見された文字列のスコアは，何種類の文において発見されたか数える．このスコアを利用して，(7) 発見された語の評価を行うが，たいていは相乗平均を取ることが十分に良い評価尺度となる．相乗平均を取ることで，どちらか片方のパターンにしかあてはまらない文字列の評価値は全て 0 となる．最後に，(8) ストップワードとして，1 字のひらがなやカタカナ，その他，よく使われるひらがなの組み合わせを取り除き，さらに，評価値がある閾値よりも大きな語を結果として出力する．

以上が，両方向構文パターンを用いた高速な関連語発見手法の一般的な流れである．この流れの中で，求めたい関連語に応じて変更しなくてはいけない設定項目は以下の 4 つである．

- $\mathbf{Patterns}^{Pre}$
- $\mathbf{Patterns}^{Post}$
- $webQuery^{Pre}$
- $webQuery^{Post}$

これらを変更することにより，様々な関係性を持つ語を高速に発見する手法を作成することが可能である．

3.3 両方向構文パターンと Web 検索クエリ

本手法において上位語を発見するための 1 つの設定として，以下のような設定がある．ただし，構文パターンは $< query >$ を除き，正規表現で表されている．

$$\mathbf{Patterns}^{Pre} := \text{である} < query >$$
$$\mathbf{Patterns}^{Post} := (\text{代表する} \mid \text{有名な}),?$$
$$webQuery^{Pre} := \text{"である} < query > \text{"}$$
$$webQuery^{Post} := \text{"である} < query > \text{"}$$
$$(\text{"代表する"}\ OR\ \text{"有名な"})$$

(注5)：http://developer.yahoo.co.jp/search/

表 1 「トヨタ」「清水寺」の上位語を求めた結果例

トヨタ	
発見された語	評価値
企業	8.5
メーカー	3.7
自動車メーカー	2.4
ブランド	2.0
産業	1.4
大企業	1.4
トップ	1.0

清水寺	
発見された語	評価値
観光地	3.0
観光スポット	1.0
建造物	1.0

まず，両方向構文パターンだが，この場合，Post パターンが実質的に複数のパターンからなっている．このように，Pre パターンや Post パターンは，複数の構文パターンを用いることが可能である．また，Pre パターンでは，パターン内に $< query >$ を含むのに対して，Post パターンでは含んでいない．このように，構文パターンは必ずしも $< query >$ を含む必要がない．

次に，Web 検索クエリでは，ともに $< query >$ の直前に「である」を付加している．また，Post パターンのためのクエリでは，その部分とは別に,「("代表する"OR"有名な")」という文字列をクエリに追加している．この設定を本手法に適用し,「トヨタ」や「清水寺」を与えた場合に出力される結果を表 1 に示す．これらの語の場合には，良い結果が得られていると言える．

3.4 両方向構文パターンを用いた関連語発見のまとめ

ここまで，両方向構文パターンを用いた関連語発見の手法について述べた．求めたい関連語に応じて設定を変更することにより，同位語の場合と同様の，高速な関連語発見手法を作成することが可能である．設定する項目は，$\mathbf{Patterns}^{Pre}$，$\mathbf{Patterns}^{Post}$，$webQuery^{Pre}$，$webQuery^{Post}$ の 4 つの項目である．また，それ以外に，Web 検索で取得する結果件数，評価手法などを変更することが可能である．

ある関係語を求める手法を作成したいときに，最も適切な設定を見つけることは困難である．そこで，次節では，自動的に適切な設定を発見する手法について述べる．

4. 両方向構文パターンの自動発見

4.1 設定の自動発見における制約

本節では，両方向構文パターンを用いた関連語発見において，1 つの $< query >$ と $< target >$ のペアを与えることで，自動的に適切な設定を発見する手法について述べる．例えば,「松井秀喜」と「ゴジラ」という語のペアを与えた場合，人物名が与えられたときにその呼称を求める手法を作成するための設定を発見する．

ただし，全ての設定項目を完全に自由にして，最適な設定を自動的に求めることは困難である．そこで，構文パターンが得られた際に，おのずと Web 検索クエリが決定する制約を設ける．すなわち，$\mathbf{Patterns}^{Pre}$ を求めればおのずと $webQuery^{Pre}$ が決定され，$\mathbf{Patterns}^{Post}$ を求めればおのずと $webQuery^{Post}$ が決定されるようにする．

例えば,「松井秀喜」と「ゴジラ」という語のペアが与えられたときに，以下のような $\mathbf{Patterns}^{Pre}$ と $\mathbf{Patterns}^{Post}$ を設定したとする．

$$\mathbf{Patterns}^{Pre} := \text{こと} < query >$$
$$\mathbf{Patterns}^{Post} := \text{の}$$

このときに，おのずと $webQuery^{Pre}$ と $webQuery^{Post}$ を以下のように決定する．

$$webQuery^{Pre} := \text{"こと} < query > \text{"}$$
$$webQuery^{Post} := \text{"} < query > \text{" の}$$

$\mathbf{Patterns}^{Pre}$ には，$< query >$ が含まれている．このような場合には，構文パターンをそのまま Web 検索クエリとして用いる．一方，$\mathbf{Patterns}^{Post}$ には $< query >$ が含まれていない．このような場合には，構文パターンと $< query >$ の AND 検索となる Web 検索クエリを生成する．このような制約を設けることで，設定の自動発見の問題は，構文パターンの自動発見というより容易な問題となる．

4.2 構文パターンの候補取得

1 つの $< query >$ と $< target >$ のペアが与えられたとき，適切な両方向構文パターンを発見することが課題である．ここでは，Web 検索を用いる手法を提案する．以下では，構文パターンの候補を取得する手法について述べる．

まず,「$< query >< target >$」をクエリとして Web 検索を行い，1000 件の検索結果を取得する．それらから $\mathbf{Patterns}^{Pre}$ を求める．まず，得られた文章群から $< target >$ の直後に現れるあらゆる文字列を取得し，それらの出現回数を数える．この時に取得する文字列の最大字数はあらかじめ決めておく．日本語の場合，15 語程度としておくと十分である．なお，この時点で，ストップワード，句読点を含む文字列，$< query >$ を文字列の中ほどに含むものを取り除く．次に，出現回数に応じて得られた文字列のスコア付けを行う．例えば,「松井秀喜」と「ゴジラ」が与えられていたとすると,「ゴジラ」の直後に現れる文字列には，例えば,「こ」「こと」「こと $< query >$」などがある．それぞれの文字列のスコアは，その文字列の出現回数から，その文字列自身を含むより長い文字列の中での最多の出現回数を引いたものとする．例えば,「こ」が 55 回,「こと」が 55 回,「こと $< query >$」が 35 回出現していたとすると,「こ」のスコアは，自身の出現回数である 55 から,「こ」を含む文字列のうち，最多出現回数をもつ「こと」の出現回数 55 を引いた 0 になる.「こと」のスコアは，55 から 35 を引いた 20 となる.「こと $< query >$」のスコアは，これ以上長い文字列が発見されていないため（$< query >$ を文字列の中ほどに含むものはスコア付けの前に取り除かれるため），そのまま 35 がスコアとなる．さらに，$< query >$ を含む文字列のスコアは 10 倍する．これは，$< query >$ を含む文字列を構文パターンとして利用できる場合は，含まない構文パターンよりもより良い結果を出すことが多かったためである．最終的には,「こと $< query >$」のスコアは 350 としている．このようにして，スコア付けされた文字列が得られ，それらが $\mathbf{Patterns}^{Pre}$ の候補となる．

$\mathbf{Patterns}^{Post}$ の候補も同様にして求められる．表 2 は「松井秀喜」と「ゴジラ」が与えられたときに，実際に取得された

表 2 「松井秀喜」と「ゴジラ」に対する両方向構文パターンの候補

Pre パターンの候補	スコア	Post パターンの候補	スコア
こと < *query* >	350	の	95
松井	212	は	46
の	152	メカ	33
が	70	< *query* > 選手が	30
松井こと < *query* >	60	< *query* > メジャー物語	30
と	50	た	28
は	31	< *query* > さんが	20
に	27	< *query* > 野球の館より	20

表 3 「松井秀喜」と「ゴジラ」に対する両方向構文パターンの評価

Pre パターンの候補	< *target* > 発見数	ヒット件数
こと < *query* >	55	8,500
松井	4	5,210,000
の	2	5,120,000
が	2	4,990,000
松井こと < *query* >	27	92
は	1	5,020,000

Post パターンの候補	< *target* > 発見数	ヒット件数
の	4	5,120,000
は	2	5,020,000
メカ	91	30,800
< *query* > メジャー物語	4	584
< *query* > さんが	2	1,020
< *query* > 野球の館より	4	11

両方向構文パターンの候補のうち，スコアが大きいものである．

4.3 構文パターンの候補の評価

得られた構文パターンの候補は，与えられた < *query* > と < *target* > の間の関係性を表すものになっている．しかし，関連語発見における構文パターンとして適しているかどうかは分からない．そこで，スコアが上位の候補に対して評価を行う．

評価で考慮すべき点はいくつか存在する．まず，構文パターンを用いて関連語発見を行った際に，実際に < *target* > を発見できることが最も重要な点である．しかし，< *target* > を異常に多く発見できる構文パターンは，< *query* > や < *target* > にのみ特有の構文パターンであり，他の語が与えられた場合には，適切な構文パターンとして機能しないことが多い．また，構文パターンを用いて Web 検索を行った際に，ある程度のヒット件数が得られることも重要な点である．少なくとも 100 件程度の検索結果が得られないと，利用できるテキストの量が少なくなり，関連語の発見に支障をきたす場合がある．

表 3 は，「松井秀喜」と「ゴジラ」が与えられたときに，構文パターンのそれぞれの候補において < *target* > を実際に発見できた数と，Web 検索でのヒット件数を示している．例えば，Pre パターンの「こと < *query* >」は，十分に < *target* > を発見することができ，さらにある程度のヒット件数を確保しているといえる．Post パターンの「メカ」は < *target* > が異常に多く発見されており，あまり良いパターンとは言えない．

Pre パターンの「松井こと < *query* >」や，Post パターンの「< *query* > メジャー物語」などは，「松井秀喜」に非常に関連が深い構文パターンであり，他の語が与えられたときには有効に機能しない．このような構文パターンを効率よく取り除く手法については今後の課題とする．

検討の余地はあるが，ここでは自動取得される構文パターンとして，< *target* > 発見数が 70 未満のもののうちなるべく大きいもので，ヒット件数が 1,000 件以上の構文パターンを利用することとする．

表 4 「松井秀喜」に対する「ゴジラ」のような語の発見手法

小泉純一郎	
語	評価値
宰相	3.7
変人	1.7
した	1.0

荒川静香	
語	評価値
イナバウアー	4.5
選手の	1.7

中田英寿	
語	評価値
ヒデ	7.5

坂本龍一	
語	評価値
教授	7.2
サカモト	1.7

4.4 両方向構文パターンの自動発見の実例

最後に，実際に両方向構文パターンの自動発見を行い，それを用いた実験結果の例を紹介する．

まず，上記で述べた「松井秀喜」と「ゴジラ」のペアより，以下の構文パターンが取得された．

$$\mathbf{Patterns}^{Pre} := \text{こと} < query >$$
$$\mathbf{Patterns}^{Post} := \text{の}$$

この設定を用いた手法において，「小泉純一郎」「荒川静香」「中田英寿」「坂本龍一」を与えた場合の結果を表 4 に示す．「堀江貴文」でも行ったが，結果が得られなかった．数は，多く得ることができなかったが，評価値が大きい語はそれなりに良い呼称を得ることができている．

「香川」と「讃岐うどん」のペアからは，以下の構文パターンが取得された．

$$\mathbf{Patterns}^{Pre} := \text{で有名な} < query >$$
$$\mathbf{Patterns}^{Post} := \text{本場の}$$

この設定を用いた手法において，「福島」「長崎」「千葉」「大阪」「静岡」「京都」を与えた場合の結果を表 5 に示す．評価値が 1.0 よりも大きい語は，おおむね良い結果である．

「田中克己」と「京都大学」のペアからは，以下の構文パターンが取得された．

$$\mathbf{Patterns}^{Pre} := \text{大学院}$$
$$\mathbf{Patterns}^{Post} := < query > \text{氏}$$

この設定を用いた手法において，「喜連川優」「村井純」「吉村作治」を与えた場合の結果を表 6 に示す．求められない人名は多いのだが，ある程度以上有名である場合は，所属の大学名を求めることができた．

5. まとめと今後の課題

本稿では，両方向構文パターンを用いた Web 検索エンジンか

表 5 「香川」に対する「讃岐うどん」のような語の発見手法

福島	
語	評価値
喜多方ラーメン	1.7
イン	1.4

長崎	
語	評価値
ちゃんぽん	4.2
チャンポン	2.8
佐世保バーガー	1.4

千葉	
語	評価値
落花生	2.0
研究	1.0

大阪	
語	評価値
お好み焼き	2.6
たこ焼き	1.4
料理	1.0
ビー	1.0
研究	1.0

静岡	
語	評価値
お茶	5.2

京都	
語	評価値
宇治茶	2.0
料理	1.4
京野菜	1.0
お茶	1.0

表 6 「田中克己」に対する「京都大学」のような語の発見手法

喜連川優		村井純		吉村作治	
語	評価値	語	評価値	語	評価値
東京大学	13.0	慶應義塾大学	11.5	早稲田大学	8.8
東大	1.0	慶應大学	1.4	早稲田大	2.6

らの高速関連語発見手法を提案した．提案手法を用いて，様々な種類の関連語を高速に発見する手法を作成することが可能である．手法の作成には，両方向構文パターンなどを考える必要があり，あまり容易ではないため，自動的に構文パターンを発見する手法についても提案を行った．いくつか実例を示して，提案手法が有効に機能することを示した．今後は，本手法が適応可能な範囲を明らかにすると共に，自動的な構文パターン発見の改善を行っていく予定である．

謝　　辞

本研究の一部は，NICT 委託研究「電気通信サービスにおける情報信憑性検証技術に関する研究開発」，および，京都大学グローバル COE プログラム「知識循環社会のための情報学教育研究拠点」，および，文部科学省科学研究費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術の研究」，計画研究「情報爆発時代に対応するコンテンツ融合と操作環境融合に関する研究」（研究代表者：田中克己，A01-00-02，課題番号 18049041），ならびに，計画研究「情報爆発に対応する新 IT 基盤研究支援プラットフォームの構築」（研究代表者：安達淳，Y00-01，課題番号：18049073），および，文部科学省研究委託事業「知的資産の電子的な保存・活用を支援するソフトウェア技術基盤の構築」，異メディア・アーカイブの横断的検索・統合ソフトウェア開発（研究代表者：田中克己）によるものです．ここに記して謝意を表します．

文　　献

[1] 大島裕明, 小山聡, 田中克己：“Web 検索エンジンのインデックスを用いた同位語とそのコンテキストの発見”, 情報処理学会論文誌（トランザクション）データベース, **Vol.47**, No.SIG19,TOD32, pp. 98–112 (2006).

[2] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K. J. Miller: “Introduction to WordNet: An on-line lexical database”, International Journal of Lexicography 3(4), pp. 235–312 (1990).

[3] 独立行政法人 情報通信研究機構：“EDR 電子化辞書 2.0 版仕様説明書”, 株式会社日本電子化辞書研究所 (2001).

[4] 株式会社 言語工学研究所：“デジタル類語辞典”.

[5] M. A. Hearst: “Automatic acquisition of hyponyms from large text corpora”, Proc. of the 14th International Conference on Computational Linguistics (COLING 1992), pp. 539–545 (1992).

[6] Z. Ghahramani and K. Heller: “Bayesian sets”, Proc. of the 19th Annual Conference on Neural Information Processing Systems (NIPS2005) (2005).

[7] D. Lin: “Automatic retrieval and clustering of similar words”, Proc. of the 36th annual meeting on Association for Computational Linguistics, pp. 768–774 (1998).

[8] K. Shinzato and K. Torisawa: “A simple www-based method for semantic word class acquisition”, Proc. of the Recent Advances in Natural Language Processing (RANLP05), pp. 493–500 (2005).

[9] K. Shinzato and K. Torisawa: “Acquiring hyponymy relations from Web documents”, Proc. of Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL04), pp. 73–80 (2004).

[10] 山口雅史, 大島裕明, 小山聡, 田中克己：“サーチエンジンのクエリログを利用した同位語の発見”, 日本データベース学会 Letters, **Vol.5**, No.2, pp. 17–20 (2006).

[11] S. Oyama and K. Tanaka: “Query modification by discovering topics from Web page structures”, Proc. of the 6th Asia Pacific Web Conference (APWeb 2004), pp. 553–564 (2004).

[12] 野田武史, 大島裕明, 小山聡, 田島敬史, 田中克己：“主題語からの話題語自動抽出とこれに基づく web 情報検索”, 日本データベース学会 Letters, **Vol.5**, No.2, pp. 69–72 (2006).

[13] 中山浩太郎, 原隆浩, 西尾章治郎：“Web 事典からのシソーラス辞書構築手法”, 情報処理学会論文誌（トランザクション）データベース, **Vol.48**, No.SIG19,TOD34, pp. 27–37 (2007).

[14] 外間智子, 北川博之：“Web データを用いた人物の呼称抽出”, 日本データベース学会 Letters, **Vol.5**, No.2, pp. 49–52 (2006).

[15] 若木裕美, 藤井寛子, 福井美佳, 住田一男：“Web 情報を用いた人物の愛称抽出”, 日本データベース学会論文誌, **Vol.7**, No.1, pp. 169–174 (2008).

[16] K. W. Church and P. Hanks: “Word association norms, mutual information, and lexicography”, Proc. of the 27th Annual Meeting of the Association for Computational Linguistics, pp. 76–83 (1998).

[17] P. D. Turney: “Mining the Web for synonyms: PMI-IR versus LSA on TOEFL”, Proc. of the 12th European Conference on Machine Lerning (ECML 2001), pp. 491–502 (2001).

[18] M. Baroni and S. Bisi: “Using cooccurrence statistics and the Web to discover synonyms in a technical language”, Proc. of the 4th International Conference on Language Resources and Evaluation (LREC 2004), pp. 1725–1728 (2004).

[19] R. L. Cilibras and P. M. Vitanyi: “The Google similarity distance”, IEEE Transactions on Knowledge and Data Engineering, **19**, 3, pp. 370–383 (2007).

[20] P. D. Turney: “Measuring semantic similarity by latent relational analysis”, pp. 1136–1141 (2005).

[21] P. D. Turney and M. L. Littman: “Corpus-based learning of analogies and semantic relations”, Machine Learning, **60**, pp. 251–278 (2005).

[22] D. Bollegala, Y. Matsuo and M. Ishizuka: “WWW sits the SAT-Measuring relational similarity on the Web”, Proc. 18th European Conference on Artificial Intelligence (ECAI 2008), pp. 333–337 (2008).