正解語ペア漸増による関連語取得のための両方向構文パターン発見

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　本研究では，これまで我々が提案してきた Web からの関連語発見手法において用いる，様々な関係性に対応する両方向構文パターンを発見する手法を提案する．Web からの関連語発見手法は，1 語が与えられたときに，その語に対する上位語，下位語，同位語などの関連語を，Web 検索エンジンを利用することによって比較的高速に発見する手法である．そこでは，両方向構文パターンと呼ぶ，異なる構文パターンを利用する．この両方向構文パターンを変更することにより，様々な関係性を持った語を発見することが可能となる．ある関連語を発見するための両方向構文パターンは，人手で考えることも可能であるが，我々はこれまで，与えられる 1 語とそれに対して想定される正解の語で構成される，1 つの正解語ペアが与えられたときに，自動的に両方向構文パターンを発見する手法について提案してきた．しかし，これまでの手法は構文パターンの評価手法が簡易的なものであり，十分に有用な構文パターンを発見することは難しかった．そこで，本研究では，与えられた正解語ペアから別の正解語ペアを次々と発見していくことによって，関連語発見においてより有用な両方向構文パターンを発見する手法を提案する．

キーワード　知識発見, Web マイニング, オントロジー

1. はじめに

Web 上には，あらゆる分野についての文書が存在するようになり，それらを利用して様々な知識を発見する研究が広く行われるようになっている．我々はこれまで，一般的な Web 検索エンジンを利用した知識発見手法を提案してきた [1], [2]．提案手法では，両方向構文パターンと呼ぶ，方向性の異なる 2 種類の構文パターンを用いる．1 語が与えられたときに，それらの構文パターンを基に Web 検索を行い，得られた検索結果に含まれるタイトルやスニペットのみから，様々な知識を比較的高速に発見する．求めたい関連語に応じて，適切な構文パターンを用意することにより，与えられた語の上位語，下位語，同位語などの辞書的に定義されるような関連語を発見したり，会社名から CEO の氏名を，芸術家から作品名を発見したりすることなどを行うことができる．しかし，求めたい関係語の発見に役に立つ両方向構文パターンを考案することは難しい場合がある．そこで，ユーザが，1 語とそれに対して実際に求めたい関連語のペア，すなわち，正解語ペアを与えることによって，与えられた正解語ペアの関係にある語を求めることができるような両方向構文パターンを自動的に発見する手法についても提案を行ってきた．

本稿では，我々がこれまでに提案した両方向構文パターンを発見する手法をさらに発展させた手法として，ブートストラッピングを用いる手法を提案する．ブートストラッピングを用いるというのは，まず始めに与えられ正解語ペアから，両方向構文パターンを発見し，つぎに，その両方向構文パターンを用いて別の正解語ペアを発見する．そのようにして漸増させた正解語ペアを用いてさらに両方向構文パターンを発見することにより，より良い両方向構文パターンを発見する，といったものである．

以下，2. 節で関連研究について述べ，3. 節で両方向構文パターンを用いた知識発見手法について説明し，4. 節で正解語ペアを与えて両方向構文パターンを発見する手法について説明する．そして，5. 節でブートストラッピングを用いた両方向構文パターンの発見手法について説明し，6. 節で実験について述べる．最後に，7. 節でまとめと今後の課題について述べる．

2. 関連研究

ブートストラッピングを用いた情報抽出手法には様々なものが提案されている [3]〜[9]．これらの手法はブートストラッピングを用いることによって，俳優名や映画のタイトルのような単純なアイテム集合，または，本のタイトル，著者，出版社のようなオブジェクト集合を次々と発見することそのものが目的である．

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 [10](注1) や EDR 電子化辞書 [11]，言語工学研究所 デジタル類語辞典（シソーラス）[12] 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 [13] 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

(注1)：http://wordnet.princeton.edu/
(注2)：http://wikipedia.org/
(注3)：http://wiktionary.org/

上位下位関係を発見する手法を提案した．Ghahramani ら [14] 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 [15] 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 [16] 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 [17] 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 [18] 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 [19] 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 [20] 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 [21] 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 [22] 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 [23] は，Web 情報から人物の愛称を求める手法を提案している．

3. 両方向構文パターンを用いた知識発見

3.1 両方向構文パターンを用いた知識発見の概要

本節では，これまで我々が提案してきた両方向構文パターンを用いた Web からの知識発見手法について説明する [2]．本手法は，1 語が与えられた際に，何らかの関係にある別の語をいくつか取得するというものである．用いる構文パターンを変えることによって，様々な関係にある語を発見することができる．

以後，入力を語 s とし，出力を $T = \{t_1, t_2, \ldots, t_n\}$ と記述することとする．

本手法では，2 つの異なる方向性をもつ構文パターンを用いる．まず，適切なクエリを用いて Web 検索を行い，その検索結果として得られたタイトルやスニペットのテキスト情報から構文パターンを基にして知識発見を行う．

2 種の構文パターンは，以下のように表される．

$$Pattern^{Pre} := \text{prefix} \parallel \langle t \rangle$$
$$Pattern^{Post} := \langle t \rangle \parallel \text{suffix}$$

記号 $\parallel$ は文字列の連接を表す．$\langle t \rangle$ が，発見される語が出現する場所を表しており，prefix と suffix はそれぞれパターンとしての適切な文字列である．$Pattern^{Pre}$ では $\langle t \rangle$ が必ず最後部に現れ，$Pattern^{Post}$ では $\langle t \rangle$ が必ず最前部に現れる．このように発見される語が出現する位置が，最後部のパターンと最前部のパターンという，2 つの異なる構文パターンを両方向構文パターンと呼ぶこととする．これらの構文パターンを含む文章を，Web 検索によって取得することで，比較的高速に精度良く知識発見を行うことが可能となる．

prefix や suffix には，入力として与えられた語を含むこともできる．入力語 s は，与えられるまで何であるか分からないため，構文パターンなどの中では $\langle s \rangle$ で表すものとする．

知識発見を行うためのテキスト情報は Web 検索によって取得する．その際に用いるクエリは，両方向構文パターンのそれぞれに応じたものである．$Pattern^{Pre}$ を含む文章を取得するためのクエリを WQ^{Pre}，$Pattern^{Post}$ を含む文章を取得するためのクエリを WQ^{Post} と表すと，それぞれ，以下のように作成される．

$$WQ^{Pre} := \begin{cases} \text{prefix} & (\text{if prefix contains } \langle s \rangle) \\ \text{prefix} \wedge \langle s \rangle & (\text{if prefix does not contain } \langle s \rangle) \end{cases}$$

$$WQ^{Post} := \begin{cases} \text{suffix} & (\text{if suffix contains } \langle s \rangle) \\ \text{suffix} \wedge \langle s \rangle & (\text{if suffix does not contain } \langle s \rangle) \end{cases}$$

prefix や suffix が入力語 s を含む場合，prefix や suffix がそのまま Web 検索のクエリとなる．prefix や suffix が入力語 s を含まない場合，入力語を含むような文章を検索するために，prefix や suffix に加えて，s を AND 条件に加える．Web 検索ではこのように作成されたクエリを用いてフレーズ検索(注5)を行い，検索結果中に現れるタイトルとスニペットのテキスト情報を取得する．なお，取得する検索結果数は，本稿においては 100 件とする．

(注4)：http://labs.google.com/sets

(注5)：Yahoo!や Google など，多くの Web 検索エンジンではクエリをダブルクォーテーションで括ることによって，入力したクエリキーワードが連続して現われているページを検索することが可能であり，一般にフレーズ検索と呼ばれる．

ここで，WQ をクエリとした際の Web 検索結果のタイトルやスニペットに含まれる文章中に出現するあらゆる部分文字列の集合を $Parts(WQ)$ と表すこととする．なお，日本語の場合は $Parts(WQ)$ を部分文字列の集合とするが，英語の場合は $Parts(WQ)$ を部分フレーズの集合とする．これは，日本語では形態素解析を行わなければ語の区切りが不明であるが，英語の場合は単語の区切りには必ずスペースがあり，明確であるためである．

この時，知識発見手法の出力の語集合 T は以下の式で表すことができる．

$$\begin{aligned} T = \{\langle t\rangle \mid \quad & Pattern^{Pre} \in Parts(WQ^{Pre}) \\ & \wedge\ Pattern^{Post} \in Parts(WQ^{Post})\} \\ = \{\langle t\rangle \mid \quad & (\text{prefix} \parallel \langle t\rangle) \in Parts(WQ^{Pre}) \\ & \wedge\ (\langle t\rangle \parallel \text{suffix}) \in Parts(WQ^{Post})\} \end{aligned}$$

この式は，Web 検索によって得られた文書中において，両方向構文パターンの $Pattern^{Pre}$ と $Pattern^{Post}$ の両方に適合する文字列が，出力されることを示している．片方の構文パターンにのみ適合するような文字列のみを考えると，あらゆる文字列長のものが考えられ，非常に数多く存在しているといえる．しかし，両方向構文パターンの両方に適合する文字列を考えると，精度良く適切な語を発見することが可能となることが明らかになっている．

3.2 両方向構文パターンを用いた上位語発見

ここでは，実際に，上位語を発見する際にどのように行うかを示す．上位語を発見するために用いられる両方向構文パターンの一例には，以下のようなものが存在する．

$$\begin{aligned} Pattern^{Pre} &:= \langle s\rangle \parallel \text{are} \parallel \langle t\rangle \\ Pattern^{Post} &:= \langle t\rangle \parallel \text{such as} \parallel \langle s\rangle \end{aligned}$$

$\langle s\rangle$ の部分は入力として与えられた語が置き換えられる．例えば，入力された語 s が **whales** であった場合，その時の $prefix$ と $suffix$ はそれぞれ以下のようになる．

$$\begin{aligned} &\text{prefix} := \textbf{whales}\ \text{are} \\ &\text{suffix} := \text{such as}\ \textbf{whales} \end{aligned}$$

このとき，それぞれの構文パターンのために用意される Web 検索のクエリは以下のようになる．

$$\begin{aligned} WQ^{Pre} &:= \textbf{whales}\ \text{are} \\ WQ^{Post} &:= \text{such as}\ \textbf{whales} \end{aligned}$$

今回用いる構文パターンは，両方とも入力語 s を置き換える部分を含んでいるため，WQ^{Pre} として prefix が，WQ^{Post} として suffix がそのまま用いられる．これらのクエリを用いて Web 検索でフレーズ検索を行い，検索結果中のタイトルとスニペットから文章を収集する．

whales are を用いた検索結果のスニペットに，以下のような文章が含まれていたとする．

> whales are marine mammals known for their long migrations.

such as whales を用いた検索結果のスニペットに，以下のような文章が含まれていたとする．

> scientific advances in the study of marine mammals such as whales, seals, manatees, ...

出力の式は以下のように表される．

$$\begin{aligned} T = \{\langle t\rangle \mid\ & (\text{whales are} \parallel \langle t\rangle) \in Parts(\text{“whales are”}) \\ & \wedge(\langle t\rangle \parallel \text{such as whales}) \in Parts(\text{“such as whales”})\} \end{aligned}$$

上記に示した 2 つの例文から，T の要素としての条件に適合する文字列を発見すると以下のようになる．

$$T = \{\text{marine mammals}\}$$

これは，入力語に対する上位語として，適した回答の 1 つであるといえる．以上が，我々がこれまで提案してきた両方向構文パターンを用いた知識発見手法の最も基本的な場合についての説明である．

上記の場合，prefix と suffix には入力語が置換される部分 $\langle s\rangle$ が含まれていた．しかし，両方向構文パターンは必ずしも $\langle s\rangle$ を含む必要はない．例えば，先ほど用いた上位語発見のための $Pattern^{Pre}$ の代わりに，以下のように $\langle s\rangle$ を含まない構文パターンを用いることも可能である．

$$Pattern^{Pre} := \text{popular} \parallel \langle t\rangle$$

この場合，s としてどのような語が与えられても prefix は変わらずに以下のようになる．

$$\text{prefix} := \text{popular}$$

この場合，$prefix$ のみを Web 検索におけるクエリとして用いると入力された語 s と関連するテキストを取得することができない．そこで，Web 検索におけるクエリとしては，prefix と s の AND 条件を用いる．s が **whales** の時の Web 検索クエリ WQ^{Pre} は以下のようになる．

$$WQ^{Pre} := \text{popular} \wedge \textbf{whales}$$

3.3 両方向構文パターンを用いた同位語発見

以下の両方向構文パターンを用いることによって，与えた語の同位語，すなわち，共通の上位語を持つ語を発見することが可能である．

$$\begin{aligned} Pattern^{Pre} &:= \langle s\rangle \parallel \text{や} \parallel \langle t\rangle \\ Pattern^{Post} &:= \langle t\rangle \parallel \text{や} \parallel \langle s\rangle \end{aligned}$$

この同位語発見手法は，精度が良く，また，様々な分野において利用できることが分かっている [1], [24]．5. 節において利用することになる．

以上，本節では両方向構文パターンを用いた知識発見手法について説明した．実際の実装においては，検索結果から記号な

どを削除する，大文字と小文字を区別しない，出力からストップワードを削除するなどの処理を行う必要がある．また，本節の説明では，両方向構文パターンの prefix と suffix を決定すれば，Web 検索のクエリが自動的に決定するようになっているが，Web 検索のクエリは必ずしも両方向構文パターンから一意に決定される必要はない．詳細については，文献 [2] を参考のこと．

4. 正解語ペアからの両方向構文パターンの発見

知識発見に有用な両方向構文パターンを考えることは，場合によっては難しいことがある．そこで，入力語とそれに対する正解の出力を与えることで，両方向構文パターンを自動的に発見する手法を提案した．本節では，これまで提案してきた手法を少し改良した手法の概要について述べる．

まず，入力として与えられるのは，正解語ペアを (s_0, t_0) である．求めたいのは，s_0 を入力したときに t_0 が出力されるような両方向構文パターンである．実際には，prefix と suffix は独立に扱えるため，prefix の集合と suffix の集合が求められることになる．

両方向構文パターンの発見は Web 検索を利用して行う．まず，$s_0 \wedge t_0$ をクエリとして Web 検索を行い，最大 1000 件の検索結果を取得する．取得された検索結果に含まれるタイトルやスニペットから，t_0 の直前に頻出する文字列を prefix の候補として，t_0 の直後に頻出する文字列を suffix の候補として取得する．

これらの中には，有用でないものも数多く含まれている．そこで，これら prefix や suffix の候補に対して，以下を考慮してスコア付けを行う．

- 上記 Web 検索結果中で候補として取得された回数
- s_0 を入力として t_0 を実際に発見できる回数
- s_0 を入力として Web 検索を行った際のヒット件数

1000 件の検索結果中で，候補として取得された回数が多ければ，基本的には構文パターンとして有用であると考えて良い．しかし，あまりに多く発見される場合，s_0 や t_0 に特化した語で場合が多い．例えば，(香川, うどん) を入力した場合，prefix の候補として,「讃岐」が頻出する．よって，現在の実装では，100 回以上発見された候補は無条件で取り除いている．

候補の取得においては，形態素解析を用いておらず，一定の長さ以下である文字列を取得している．ある文字列が prefix や suffix の候補として n 回出現しているとき，その文字列を含むより長い文字列の中で，最も出現が多いものの出現回数を m とすると，$n-m$ をその文字列の初期スコアとする．また，prefix の先頭部分が s_0 である場合と，suffix の末尾部分が s_0 である場合は，単純に出現回数の 10 倍を初期スコアとする．このようにして付けられた初期スコアの大きい順に候補を順位付けし，上位 10 個の prefix の候補と suffix の候補について，s_0 を入力した際に t_0 を発見できるかどうかを調べる．

様々な正解語ペアを与えて，両方向構文パターン発見を行った結果，s_0 を入力した際に t_0 を 15 回程度取得できるようなものに良い構文パターンが多いことがわかった．また，s_0 を入力として検索を行った際のヒット件数が 1000 よりも小さい場合はあまり良い構文パターンではないことが分かった．そこで，実際に t_0 を発見できる構文パターンについて，ヒット件数が 1000 よりも大きく，t_0 を発見できた回数がなるべく 15 回に近いものをより上位に順位付けする．そのように順位付けされた prefix と suffix の候補を結果として返す.

表 1　同位語発見の実験結果のまとめ

prefix	t_0 発見回数	ヒット件数
⟨s⟩ オンタリオ州	10	246,000
⟨s⟩ の首都	55	70,800
⟨s⟩ 写真集	3	507
⟨s⟩ 産イソテルス	2	2
⟨s⟩ の首都である	56	174

実際に発見される構文パターンの例として，(カナダ, オタワ) を入力した際に得られる prefix を表 1 に示す.

これらのうち，有用な prefix は,「⟨s⟩ の首都」と「⟨s⟩ の首都である」の 2 つであり，それ以外は「カナダ」以外が入力されたときに役に立つとはほぼ考えられないものである．このように，1 つの正解語ペアからの構文パターンを発見する手法には精度の問題があるといえる．

5. ブートストラッピングを用いた両方向構文パターン発見

本節では，初期に与えられた正解語ペアからブートストラッピングによって正解と考えられる語ペアを増やしていくことによって，両方向構文パターンを発見する精度を向上させる手法について述べる．

5.1 正解語ペア数を増やすことによる両方向構文パターン発見の精度向上

ユーザの直接入力も含め，何らかの手法によって正解語ペアが多く得られれば，両方向構文パターンの発見の精度は高くなる．例えば，以下のような正解語ペア集合 P が与えられたとする．

$$P = \{(s_0, t_0), (s_1, t_1), \ldots, (s_n, t_n)\}$$

4. 節で述べた手法を利用して，このうちの 1 つの正解語ペア (s_i, t_i) に対して prefix と suffix の候補集合が得られる．いずれかの正解語ペアから発見される prefix の集合を $PreSet(P)$, suffix の集合を $SufSet(P)$ とする．

両方向構文パターンは，$PreSet(P)$ と $SufSet(P)$ の要素を組み合わせて作成することができる．正解語ペア P の要素数が多ければ多いほど，より多くの両方向構文パターンの候補が取得できることは明らかである．

発見された両方向構文パターンは，それぞれ有用かどうかを見分ける必要がある．直感的には，ある両方向構文パターンを用いた際に，正解語ペア P のなるべく多くの要素を正しく発見できるものがより良いものであるといえる．このような構文パターンの評価は，正解語ペア P の要素数が多ければ多いほどより正確に行うことが可能である．

5.2 正解語ペアの発見手法

与えられる正解語ペアの要素数が少ない場合は，ブートストラッピングにより，正解語ペアを次々と求めていくことによって，より良い両方向構文パターンを求めることができるようになると考えられる．

ここでは，始めに正解語ペア集合 P_0 が与えられたとする．なお，P_0 の要素数は 1 以上であれば良いため，ここでは，$P_0 = \{(s_0, t_0))\}$ とする．最終的な目的は，s_0 を与えれば t_0 を返すような，有用な両方向構文パターンを発見することである．

まず，P_0 を利用して可能な操作は 2 つある．

- $FindPattern(s \rightarrow t)$:

s_0 が与えられたときに t_0 を取得するための構文パターンを発見する．

- $FindPattern(t \rightarrow s)$:

t_0 が与えられたときに s_0 を取得するための構文パターンを発見する．

例えば，s_0 とは異なる語 s_1 が何らかの手法で取得された場合，$FindPattern(s \rightarrow t)$ で発見した構文パターンを用いて，s_1 のペアとなるべき t_1 を発見することが可能となる．逆に，t_1 のみが取得された場合，$FindPattern(t \rightarrow s)$ で発見した構文パターンを用いて正解語ペアを作成することが可能である．

いったん構文パターンが発見されると，正解語ペアを増やす操作が行えるようになる．

以下の操作は，既存の s_0 や t_0 を含む，(s_0, t_0) 以外の正解語ペアを取得しようとする操作である．

- $FindPairs(s \rightarrow t)$:

$FindPattern(s \rightarrow t)$ で発見された両方向構文パターンを用い，s_0 を与えて t_0 ではない t_1 を発見し，新たな正解語ペア (s_0, t_1) を取得する．

- $FindPairs(t \rightarrow s)$:

$FindPattern(t \rightarrow s)$ で発見された両方向構文パターンを用い，t_0 を与えて s_0 ではない s_1 を発見し，新たな正解語ペア (s_1, t_0) を取得する．

関連語発見手法の出力は，通常は 1 語ではなく数語である．そのため，例えば，s_0 を入力として関連語発見を行うと，既知の正解ペアである (s_0, t_0) 以外の正解語ペアを取得することが可能である．これらの操作では，必ず既知の正解語ペアのどちらかを含んだ正解語ペアしか得ることができない．$TargetToPairs(s \rightarrow t)$ と $TargetToPairs(t \rightarrow s)$ を両方行うことで全く新しい正解語ペアを取得することが可能であるが，それが可能であるかどうかは s_0 と t_0 の関係性にも依存する．s_0 と t_0 の関係性については後述する．

s_1 や t_1 を直接的に発見するためには，s_0 や t_0 を入力として，同位語発見を行えばよい．同位語発見には，3.3 節で述べた手法を用いることができる．

- $FindPeers(s \rightarrow s)$:

s_0 を与えてその同位語 s_1 を発見する．

- $FindPeers(t \rightarrow t)$:

t_0 を与えてその同位語 t_1 を発見する．

同位語は，一般に，複数取得することができる．$FindPeers$ と $FindPairs$ を組み合わせることで，新たな正解語ペア発見の操作を作成することができる．

- $FindPeersToPairs(s \rightarrow t)$:

既存の s_0 を基に $FindPeers(s \rightarrow s)$ を行い s_1 を取得し，$FindPattern(s \rightarrow t)$ で発見された両方向構文パターンを用い，s_1 を与えて t_1 を発見し，(s_1, t_1) を取得する．

- $FindPeersToPairs(t \rightarrow s)$:

既存の t_0 を基に $FindPeers(t \rightarrow t)$ を行い t_1 を取得し，$FindPattern(t \rightarrow s)$ で発見された両方向構文パターンを用い，t_1 を与えて s_1 を発見し，(s_1, t_1) を取得する．

以上のように，既存の s や t を含む正解語ペアを増やす $FindPairs(s \rightarrow t)$ と $FindPairs(t \rightarrow s)$，同位語発見を用いる $FindPeersToPairs(s \rightarrow t)$ と $FindPeersToPairs(t \rightarrow s)$ の 4 つの操作によって，つぎつぎと正解語ペアを増やしていくことが可能である．

5.3 正解語ペアの関係性

正解語ペアの間には何らかの関係があるのだが，その性質によっては，これらの 4 つの操作は制約を受ける場合がある．

正解語ペアの関係について分類する．1 つの正解語ペア (s_i, t_i) は，語 s_i が入力されたときに，語 t_i が得られることが期待されるということを表しているが，その s_i と t_i の間に成り立つ関係性には以下の 4 種が考えられる．

- 1 対 1
- 1 対多
- 多対 1
- 多対多

1 対 1 とは，s_i が決定されれば t_i が決定し，さらに，t_i が決定されれば s_i が決定するような関係である．例えば，国と首都のような関係は 1 対 1 関係である．1 対多とは，s_i に対して正解となる t_i は多数存在しうるが，ある t_i に対する s_i は一意に決まるような関係である．多対 1 とは，1 対多の逆の関係である．例えば，父親と子どものような関係は 1 対多関係である．多対多とは，どちらからも一意に決定するようなことがない関係のことであり，例えば，学会と研究者の関係は，1 つの学会には多くの研究者が所属し，1 人の研究者は多くの学会に所属できることから，多対多の関係であるといえる．

以下で，正解語ペアを増やす 4 つの操作に対する制約をまとめる．

- $FindPairs(s \rightarrow t)$:

1 つの s に対して，複数の t が対応しうることが前提となっている操作であるため，1 対 1 の場合と多対 1 の場合は，用いるべきではない．

- $FindPairs(t \rightarrow s)$:

1 つの t に対して，複数の s が対応しうることが前提となっている操作であるため，1 対 1 の場合と 1 対多の場合は，用いるべきではない．

- $FindPeersToPairs(s \rightarrow t)$:

既存の s に対する同位語発見の部分については制約を受けない．しかし，1 対 1 の場合と多対 1 の場合は，発見された s の同位語のそれぞれに対応させる t は 1 つに制限すべきであり，また，

発見された t が既知の正解語ペアに存在する場合にもそれを用いるべきではない．

- $FindPeersToPairs(t \rightarrow s)$:

既存の t に対する同位語発見の部分については制約を受けない．しかし，1 対 1 の場合と 1 対多の場合は，発見された t の同位語のそれぞれに対応させる s は 1 つに制限すべきであり，また，発見された s が既知の正解語ペアに存在する場合にもそれを用いるべきではない．

5.4 正解語ペア集合と構文パターン集合の評価

正解語ペアや両方向構文パターンを新たに取得した際，ノイズが混ざってしまうことがある．ブートストラッピングによってアイテム集合を増やす場合，ノイズが残ってしまうと，そのノイズによって次々と間違ったアイテムを取得することになってしまう．そこで，精度良く正解語ペアや両方向構文パターンを取得していくことが必要となる．本節では，正解語ペアや両方向構文パターンを取得した際に，それらを評価して，適切でない正解語ペアや両方向構文パターンを排除する手法について説明する．

提案する手法では，正解である語ペアを発見することができる両方向構文パターンはより有用であり，より有用であると考えられる両方向構文パターンが正しく機能するような語ペアはより正解である可能性が高い，という考えに基づく．これは，HITS アルゴリズムにおける hub と authority の間に成立する関係に類似するものである．そこで，HITS アルゴリズムを用いて両方向構文パターンと正解語ペアを同時に評価する手法を提案する．

P_k を現時点での正解語ペア集合とすると，それらを基にして発見できるすべての prefix の集合は $PreSet(P)$，すべての suffix の集合は $SufSet(P)$ と表される．このとき，これらの prefix と suffix を用いて作成できるすべての両方向構文パターンを考えるには，$PreSet(P)$ と $SufSet(P)$ の直積集合を考えればよい．ここでは，$PreSet(P)$ と $SufSet(P)$ のあらゆる組み合わせによる両方向構文パターンの集合を BP_k と書くことにする．

$$BP_k = \{b_1, b_2, \ldots\}$$

今，BP_k と P_k を頂点集合とする 2 部グラフを考える．$b_i \in BP_k$ が $(s_j, t_j) \in P_k$ の正解語ペアに対して正しく機能する，すなわち，b_i を両方向構文パターンとして関連語発見を行い，q_j を入力としたときに a_j を実際に発見することができる場合には，BP_k と P_k のアイテム間にリンク（エッジ）が存在するような 2 部グラフを生成する．2 部グラフに対して HITS アルゴリズムを適用する場合の，Hub 集合には両方向構文パターン集合が，Authority 集合には正解語ペア集合がそれぞれ対応する．

図 1 は 2 部グラフの一例を表している．HITS アルゴリズムによって計算される hub 値や authority 値が高い順に構文パターンと正解語ペアが並び替えられている．上位部分のグラフには，密なリンクが張られた構文パターン群と正解語ペア群があり，下位部分のリンクは疎になっている．このグラフの上位部分にある正解語ペアと構文パターンのみを利用することによって，ノイズを減らすことが可能であると考えられる．

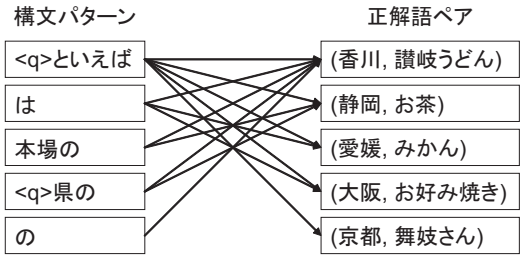

図 1 構文パターン集合と正解語集合に対する HITS アルゴリズムの適用

表 2 国名から首都の取得（P_0 における両方向構文パターン）

prefix	suffix
⟨s⟩ オンタリオ州	川
⟨s⟩ オンタリオ州	にある
⟨s⟩ の首都	川
⟨s⟩ の首都	にある

6. 実　　験

本節では，提案手法を実装して行った実験について述べる．今回の実験では，5.3 節で述べた正解語ペアの関係性については，「1 対多」であると仮定して，それの際に制約の少ない $FindPeersToPairs(s \rightarrow t)$ を利用して正解語ペアを増やすことを行った．

6.1 国名から首都の取得

まず，$P_0 = \{(\text{カナダ}, \text{オタワ})\}$ を与えて行った実験について述べる．

はじめに，P_0 に対する両方向構文パターン発見 $FindPattern(s \rightarrow t)$ を行い，その結果として，prefix と suffix のそれぞれ上位 2 件を取得する．この取得する件数は，パラメータとして変化させることができる．表 2 がそれらの組み合わせによる両方向構文パターンのすべてである．

次に，語 s「カナダ」の同位語を取得する．発見されたのは，「アメリカ」「米国」「オーストラリア」「日本」「北欧」「イギリス」であった．このうち，上位 2 語の「アメリカ」と「米国」を利用する．ここで利用する語の数もパラメータとして変化させることができる．

次に，それらの語に対する語 t を $FindPeersToPairs(s \rightarrow t)$ を利用して求めると，「アメリカ」に対しては「ナイアガラ」が，「米国」に対しては「ワシントン dc」が発見された．これにより，P_1 は以下のようになった

$$P_1 = \{(\text{カナダ}, \text{オタワ}), (\text{アメリカ}, \text{ナイアガラ}), (\text{米国}, \text{ワシントン dc})\}$$

ここで，HITS による評価を行う．表 3 がその結果である．HITS の Hub 値や Authority 値は，全体をベクトルと見なしたときに長さが 1 になるように正規化している．Hub 値や Authority 値がある程度よりも小さいときには，正解語ペアや両方向構文パターンから取り除く．実験では 0.2 以下の時に取

表 3　国名から首都の取得（P_1 における HITS による評価）

prefix	suffix	値
⟨s⟩ の首都	にある	0.56
⟨s⟩ オンタリオ州	川	0.56
⟨s⟩ オンタリオ州	にある	0.44
⟨s⟩ の首都	川	0.44

語 s	語 t	値
カナダ	オタワ	0.93
米国	ワシントン dc	0.26
アメリカ	ナイアガラ	0.26

表 4　国名から首都の取得（P_1 から取得された両方向構文パターンの HITS による評価）

prefix	suffix	値
⟨s⟩ の首都	にある	0.34
⟨s⟩ の首都	で開催された	0.34
⟨s⟩ 首都	にある	0.34
⟨s⟩ 首都	で開催された	0.34
首都	で開催された	0.34
首都	にある	0.34
⟨s⟩ オンタリオ州	川	0.22

語 s	語 t	値
カナダ	オタワ	0.79
米国	ワシントン dc	0.61

表 5　国名から首都の取得（P_3 から取得された両方向構文パターンの HITS による評価）

prefix	suffix	値
⟨s⟩ 首都	で開催された	0.19
⟨s⟩ の首都である	で開催された	0.19
⟨s⟩ の首都	で開催された	0.19

語 s	語 t	値
イギリス	ロンドン	0.82
米国	ワシントン dc	0.29
日本	東京	0.29
世界	ニューヨーク	0.26
カナダ	オタワ	0.24

り除くようにしており，ここでは何も取り除かれない．

次に，P_1 を基に両方向構文パターン発見 $FindPattern(s \rightarrow t)$ を行う．各正解語ペアから最大 2 種類の prefix と suffix を取得したところ，prefix として 6 種類，sufix として 5 種類得られた．よって，両方向構文パターンは 30 個取得された．

それらに対してやはり HITS を用いた評価を行い，Hub 値や Authority 値が 0.2 以下のものを削除した結果が表 4

正解語ペアからは (アメリカ, ナイアガラ) が削除されている．両方向構文パターンも明らかに良くなっている．このような操作を続けていった結果，3 段階目では，表 5 のような結果となった．正解語ペアや両方向構文パターンの HITS による値がすべて閾値以下になってしまったときは，最大値を持つもののみ残すようにしている．このように，正解語ペアが増え，両方向構文パターンが洗練されてきているのが分かる．

6.2　その他の例

その他いくつかの実験について述べる．

$P_0 = \{(\text{秋田}, \text{きりたんぽ})\}$ を与えたときは，$P_2 = \{(\text{秋田}, \text{きりたんぽ}), (\text{山形}, \text{芋煮})\}$ となり，これ以上新しい正解語ペアが取得されなかった．両方向構文パターンとしては，prefix として,「⟨s⟩ 名物の」や「郷土料理」などが，suffix として,「鍋」や「等 ⟨s⟩」などが得られた．鍋に特化したパターンになってしまっており，失敗であるといえる．

$P_0 = \{(\text{ヨーヨーマ}, \text{チェロ})\}$ を与えたときは，P_4 の段階で，6 名のチェロ奏者と，1 名のヴァイオリン奏者がそれぞれの演奏楽器名とともに正解語ペアとして得られた．明らかにチェロ奏者に偏ってしまっており，両方向構文パターンにもその傾向が見られている．prefix としては,「⟨s⟩ の」や「の無伴奏」などが，suffix としては,「奏者」や「ソナタ」などが得られた．prefix として現れている「の無伴奏」は，有名な楽曲名である「無伴奏チェロ組曲」や「無伴奏ヴァイオリンソナタ」などの一部として得られたものである．こちらも，比較的チェロに特化したパターンになってしまった．

$P_0 = \{(\text{ホンダ}, \text{本田宗一郎})\}$ を与えたときは，P_4 の段階で,「ホンダ」「トヨタ自動車」「トヨタ」「ソニー」「日産自動車」と，それぞれの創業者がペアになったものが正解語ペアとして得られていた．prefix としては,「⟨s⟩ 創業者」や「⟨s⟩ 創業者の」などが，suffix としては,「氏」が得られた．得られたパターンは，他の様々な企業名を入れたときにも創業者名を得られるものになっている．

験を通し，狭い範囲で構文パターンを取得してしまう場合があることが分かった．今後，なるべく広い分野に対応するパターンを発見する手法について検討しなくてはならない．

7.　まとめと今後の課題

本稿では，これまで我々が提案してきた関連語取得における両方向構文パターンを，正解語ペアをブートストラッピングによって増やしていくことによって精度良く発見する手法を提案した．構文パターンの発見と正解語ペアの発見を繰り返すことにより，多くの正解語ペアを発見することで，より良い構文パターンを発見する手法である．システムを実装し，いくつかの実験を行ったが，今後，さらに多くの実験を行い，様々なパラメータをどのように設定すれば，多くの関係語発見において提案手法が有効に働くかを明らかにしなくてはならない．

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，課題番号 18049041），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] 大島裕明, 田中克己：“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”, 情報処理学会研究報告, **Vol.2008**, No.88, 2008-DBS-146, pp. 37–42 (2008).
[3] S. Brin: “Extracting Patterns and Relations from the World Wide Web”, The World Wide Web and Databases, International Workshop (WebDB'98), pp. 172–183 (1998).
[4] E. Agichtein and L. Gravano: “*Snowball*: Extracting Relations from Large Plain-Text Collections”, Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 85–94 (2000).
[5] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld and A. Yates: “Web-Scale Information Extraction in KnowItAll: (Preliminary Results)”, Proceedings of the 13th international conference on World Wide Web, pp. 100–110 (2004).
[6] S. Soderland, O. Etzioni, T. Shaked and D. Weld: “The Use of Web-based Statistics to Validate Information Extraction”, Proceedings of AAAI 2004 Workshop on Adaptive Text Extraction and Mining (ATEM'04) (2004).
[7] E. Riloff and R. Jones: “Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping”, Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pp. 474–479 (1999).
[8] 水口弘紀, 河合英紀, 土田正明, 久寿居大：“Web 知識を利用したブートストラップによる辞書増殖手法”, 電子情報通信学会第 18 回データ工学ワークショップ論文集 (DEWS2007).
[9] 楠村幸貴, 土方嘉徳, 西田正吾：“テンプレートの交叉と DOM 構造の解析による情報抽出手法の提案”, 電子情報通信学会第 17 回データ工学ワークショップ論文集 (DEWS2006) (2006).
[10] 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).
[11] 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).
[12] 株式会社 言語工学研究所：“デジタル類語辞典”.
[13] 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).
[14] Z. Ghahramani and K. Heller: “Bayesian sets”, Proc. of the 19th Annual Conference on Neural Information Processing Systems (NIPS2005) (2005).
[15] D. Lin: “Automatic retrieval and clustering of similar words”, Proc. of the 36th annual meeting on Association for Computational Linguistics, pp. 768–774 (1998).
[16] 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).
[17] 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).
[18] 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).
[19] 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).
[20] 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).
[21] 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).
[22] 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).
[23] 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).
[24] H. Ohshima, S. Oyama and K. Tanaka: “Searching coordinate terms with their context from the Web”, Proc. of the 7th International Conference on Web Information Systems Engineering (WISE 2006), pp. 40–47 (2006).