情報処理学会論文誌　Vol. 52　No. 4　1–13 (Apr. 2011)

エンティティペア間類似性を利用した潜在関係検索

グェン　トアンドゥク[†1]　　ボレガラ　ダヌシカ[†1]
石塚　満[†1]

エンティティ間の潜在的な関係を利用して検索を行う潜在関係検索は，新しい Web 検索のパラダイムとしての可能性を有する．潜在関係検索エンジンを利用すると，例えばクエリ {(Japan, Mt. Fuji), (Germany, ?)} に対して，ドイツで最も高い山である “Zugspitze” を “?” の答えとして出力することができる．つまり潜在関係検索エンジンは，日本で最も高い山が富士山であるという関係を 利用して，ドイツで最も高い山である Zugspitze を答えとして出力する．このような潜在関係検索を Web 上で高速かつ高精度に行うためには，いくつかの課題を解決する必要があり，本稿ではそれらを解決する方法を示す．まず，高速な検索を行うために，エンティティペアを Web から発見，抽出する手法と，抽出されたエンティティペアへのインデックス構築手法を提案する．次に，我々の考案による関係類似度計算アルゴリズムを利用し，インデックスを用いる検索結果のランキングを行うことで，高精度な潜在関係検索エンジンを実現する．また本研究では評価実験として，提案システムと既存の関係検索システムとの Web 上での性能比較を行う．その結果，提案手法は高い精度と平均逆順位 (MRR) を得ることができ，かつ高速にクエリを処理できることを確認している．この結果により，潜在関係検索エンジンが実用レベルで応用可能であることを示す．

Exploiting Relational Similarity between Entity Pairs for Latent Relational Search

Nguyen Tuan Duc,[†1] Danushka Bollegala[†1]
and Mitsuru Ishizuka[†1]

Latent relational search could be a new search paradigm based on the implicitly stated relation between two entities. A latent relational search engine is expected to return the entity “Zugspitze” as an answer to the question mark (?) in the query {(Japan, Mt. Fuji), (Germany, ?)}, because Mt. Fuji is the highest mountain in Japan, whereas Zugspitze is the highest mountain in Germany. To perform latent relational search on the Web, one must overcome several challenges: discovering entity pairs to build an index for high speed retrieval, exploring and representing entity pairs’ relation, and ranking the candidate answers according to the degree of relational similarity between the candidate entity pairs and the input pair. We propose a method for extracting entity pairs from a text corpus to build an index for a high speed latent relational search engine. We apply a state-of-the-art relational similarity measuring algorithm invented by us to correctly assess the degree of relational similarity between two entity pairs and accurately rank the result list. We evaluate the system using a Web corpus and compare the performance with an existing relational search engine. The results show that the proposed method achieves high precision and MRR while requiring small query processing time.

1. まえがき

膨大な WWW 情報空間中には，多数のエンティティとそれらのエンティティ間の関係が多数，潜在的に記述されている．従来のキーワードベースの Web 検索エンジンは，キーワードを受け取り，そのキーワードを含む文書を見つけ出すが，エンティティ間の関係を検索することは出来ない．一方で，エンティティ間の関係を利用し，その関係に基づいた検索を実現するために，潜在関係検索という新しい検索パラダイムが検討されてきた[1),2)]．潜在関係検索とは，与えられたエンティティペア (A, B) (以降，ソースペアという) と与えられたエンティティ C (以降，キーエンティティという) に対して，(A, B) と (C, D) が類似関係を持つようなエンティティD を検索することである．ここで，(A, B) ペア中の A と B との関係が (C, D) ペア中の C と D との関係と強く類似するときに，(A, B) と (C, D) の関係類似度が高いという．

潜在関係検索の概要を **図 1** に示す．潜在関係検索はクエリ {(Japan, Mt. Fuji), (Germany, ?)} に対して，“Zugspitze” を最初にランキングした結果リストを返す．なぜなら，エンティティペア (Japan, Mt. Fuji) と (Germany, Zugspitze) の関係が類似しているからである (日本で最も高い山が富士山で，ドイツで最も高い山が Zugspitze である)．

WWW 空間における情報爆発に伴い，単純なキーワードを含む Web ページ検索だけではなく，エンティティ間の関係に着目した関係検索も有効な方面が考えられるようになってきた．特に，自然言語処理，Web マイニングやレコメンダシステムなどの分野において関係検索が応用できる可能性があり，関係検索システム実現への期待が存在する[3)]．例え

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

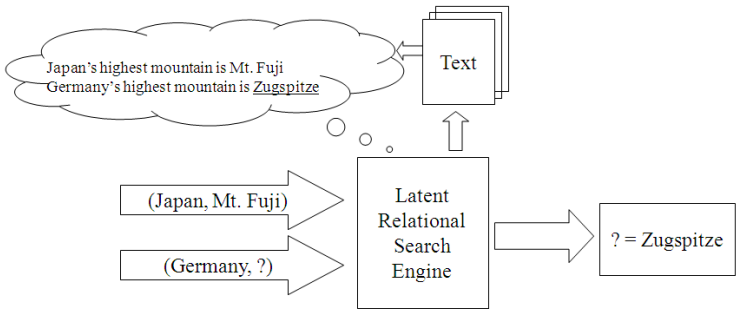


図 1　潜在関係検索の例
Fig. 1　An example of latent relational search

ば，Turney は単語ペア間の関係類似度を利用し，類義語，上位語，下位語，対義語を自動的に見つけるための統一的な手法を提案した[4] が，これを潜在関係検索で行うと，例えば，“animal” の下位語を見つけるために，{(fruits, orange), (animal, ?)} というクエリを用いて検索を行うことになる．また，製品名検索の例を挙げると，Apple の iPod ユーザが Microsoft の対応するミュージックプレイヤーを検索したい時，クエリ {(Apple, iPod), (Microsoft, ?)} で検索を行えば，“Zune” が答えとして得られる[3]．このように，キーワード (“Zune”, “music player” など) を知らずに情報を検索したい時に，まず潜在関係検索エンジンを使い，キーワードを取得し，取得したキーワードで既存のキーワードベース検索エンジンで検索すればよくなる．つまり，未知の領域での検索を行いたい時，潜在関係検索が有効であるといえる．これは，関係検索エンジンが異なる領域間の知識マッピング能力を持つからである．即ち，ユーザの既知の領域 (Apple の製品) における知識を，未知の領域 (Microsoft の製品) にマッピングすることで，未知の領域の知識を獲得する，という能力である．このマッピングは関係の類似度に基づくものであるが，実際に検索で使うソースペアの 2 つのエンティティ間の関係は顕在的にではなく，種々のテキスト文形として潜在的にしか与えられない．即ち，関係種別は述語名やラベルといったように明示的に与えられている訳ではない．故に，我々はこの検索方法を “潜在関係検索” と呼ぶ．本論文で扱うエンティティの種類は人名，組織名，地名などの固有表現であるが，本論文の手法を使い，他の品詞の種類 (例えば，時間，生年月日や一般の名詞など) を扱うこともできる．

本稿では，高速に潜在関係検索を実現するための，テキストコーパスからエンティティペアを自動抽出する手法と，検索の高速化のために必要なインデックス構築手法を提案する．また，エンティティ間の関係を，その周辺の文脈の語彙パターンで表現し，高精度な関係類似度計算を行う我々が考案した手法[2),5)] を応用することにより，高精度な潜在関係検索を実現する．更に，エンティティの類似度によるエンティティクラスタリング手法を用いて，同一エンティティの異なる表現 (surface forms) を 1 つのエンティティクラスタにまとめることで，検索結果の精度，再現率を向上させる．最後に，提案したシステムを Web 上のコーパスで評価し，既存の関係検索システムと比較することにより，提案手法が高精度，高速かつ高い平均逆順位 (mean reciprocal rank, MRR) を達成できることを示す．またこの結果により，潜在関係検索が実用的なレベルで応用できることを明らかにする．

本稿は以降，第 2 節で潜在関係検索の関連研究を紹介し，本研究との差異の概要を説明する．次に第 3 節で潜在関係検索エンジンの全体図を示し，第 4 節で高速な潜在関係検索のためのエンティティペア抽出手法とエンティティ間の関係表現手法，及びインデックス構築手法を説明する．また，第 5 節で関係類似度による検索結果フィルタリング手法とランキング手法について述べ，第 6 節でシステムの評価方法と評価結果や既存研究との比較結果を示す．最後に，第 7 節で結論と今後の課題について述べる．

2. 関 連 研 究

潜在関係検索というアイディアはアナロジーシソーラス構築に関する研究[1)] や，関係類似度計算に関する研究[2)] において検討されてきた．Veale[1)] は，クエリ “Muslim church” に対して，“mosque” を答えとして出力するような検索手法を提案した．この手法は，WordNet の中の概念階層を詳細に分割することにより，クエリ “Greek A” (あるいは {(English, A), (Greek, ?)}) に対して，“Alpha” (ギリシャの α 文字) を出力する，という手法である．しかしながらこの手法では，シソーラス (WordNet) 内に存在しない単語に対する答えは出力できない．従って，WWW 上では検索エンジンのユーザの興味の対象となる新しいエンティティが次々と生まれてくるにもかかわらず，WordNet は頻繁には更新されないため，そのようなシソーラスを使うことは実用的ではない．

TextRunner[6)] はテキストコーパスから自動的に関係のインスタンスを抽出できるシステムである．TextRunner は，既存の関係抽出システムとは異なり，関係の種類が予め与える必要がなく，どの関係の種類でも抽出できるので，Open Information Extraction を実現したという．つまり，例えば，“Google acquired YouTube for $1.65 billion” という文が与えられたとすると，*acquired(Google, YouTube)* という述語構造を抽出するのである．こ

のように，TextRunner は述語を関係として抽出できるが，*acquired(Google, YouTube)* のように関係を表現しても周囲の文脈は関係に取り込まれず，関係類似度を高精度で測定することはできない．即ち，この表現方法を利用すると，例えば “Microsoft acquired Powerset for $100M” や “Ebay acquired 33% interest in EachNet” という文があった時，(Google, YouTube) と (Microsoft, Powerset) との間の関係類似度は，(Google, YouTube) と (Ebay, EachNet) との間の関係類似度と同じと判断されてしまう．

Turney や Bollegala らは，単語間の関係を周辺文脈の語彙パターンで表現し，そのパターンの集合の類似度で関係類似度を定義することにより，高精度な関係類似度の計算法を実現した[2),7)]．例えば，ペア (lion, cat) の語彙パターンを抽出するために，ウェブ検索エンジンで，クエリ “lion ∗ ∗ ∗ cat” (“∗” はワイルドカード演算子で，多くの検索エンジンでは，0 かまたは 1 語にマッチする) を使い検索する．そして，検索結果のテキストスニペットから，例えば，語彙パターン “lion is a big cat”, “lion is the largest cat” などを抽出する．語彙パターンがペア (lion, cat) に依存しないように，“lion” を 変数 X に，“cat” を変数 Y に置き換え，(lion, cat) ペアの関係を特徴付ける語彙パターンとして “X is a big Y” や “X is the largest Y” が得られる．また，次にペア (ostrich, bird) が入力されたら，同様の手法で，(ostrich, bird) の関係を特徴付ける語彙パターンとして，例えば，“X is a big Y”, “X is a large Y”, “X is the largest Y in the world” などのパターンを抽出する．これで，ペア (lion, cat) と (ostrich, bird) の間の関係類似度を語彙パターン集合のコサイン類似度として定義できる．これらの研究は類似度の計算に関する研究であるため，2 つの単語ペアの 4 単語はあらかじめ与えてある，という仮定を置いている．本研究では上記の研究の成果を利用し，関係を周辺文脈の語彙パターンで表現することで，関係類似度に基づく検索結果のランキングを行う．また，我々が考案した語彙パターンのクラスタリング手法[2)] を使い，類似パターン (“X is a large Y” と “X is a big Y” など) を 1 つのクラスタにまとめることで，完全にマッチするパターンの低頻度問題 (data sparseness problem) を解決する．

WWW2REL[8)] システムでは，関係 R について，R($\arg_1$, ?) または R(?, $\arg_2$) のようなクエリに対して答えを出力することができる．このシステムではまず，関係 R を持つ 40 個の単語ペアをシードとして使い，関係 R を表現する語彙パターンを生成する．例えば，INDUCES という関係に対して，(carbon dioxide, headache) というシードペアを与えると，そこから関係 INDUCES を特徴づける語彙パターンとして “may cause”, “lead to” などを抽出する．次に，これらの語彙パターンを使い，クエリ INDUCES(aspirin, ?) に対して，“aspirin may cause ∗” (“∗” はワイルドカードのオペレータで，多くの Web 検索エンジンでは 1 つ以上の単語にマッチする) のようなクエリをキーワードベースの Web 検索エンジンに投げ，“apoptosis” という答えを出力する．上記のように，WWW2REL は関係検索を実現するが，潜在関係検索ではない．また，それぞれの関係について，40 個のシードペアを取得するためにシソーラスが必要であり，シソーラスに出現しない関係に対しては答えを出せない．

Kato ら[3)] は，既存のキーワードベースの検索エンジンを利用し，単語間の関係を bag-of-words モデルで表現し，潜在関係検索を実現した．Kato らの手法では，次の 2 つのステップの作業により，クエリ {(A, B), (C, ?)} の答え D を出力する．ステップ 1 ではペア (A, B) における A と B の関係を表す単語や語彙パターンの集合 T を抽出する．T は，キーワードベースの検索エンジンと χ^2 検定を利用して，A と B が含まれるページに偏って出現する単語や語彙パターンを抽出することで作成できる．ステップ 2 では，与えられたキーエンティティC と，抽出された単語または語彙パターン $t \in T$ を使い，C と t のみとよく共起する単語を検索エンジンを使って抽出する．この単語が D の候補であり，χ^2 値の高いものほど上位にランキングされる．Kato らの手法は，関係検索のためのインデックスを作成せずに，既存のキーワードベースの Web 検索エンジンのインデックスを利用できるので，実装のコストが小さい．また，bag-of-words モデルを用いるので，幅広い範囲の単語種類を検索できるという利点がある．しかし，上記の手法は正確に単語ペア間の関係類似度を測定できないため，検索の精度が低い．その理由は，関係を bag-of-words モデルで表現するので，C と $t \in T$ との共起頻度が高い単語 D_1 があった時，D_1 が C と t で結ばれる関係に無くても，D_1 が答え D の候補になる可能性が高くなるからである．例えば，C が “Microsoft”, t が “CEO” の場合，Windows は Microsoft, CEO とよく共起するが，Microsoft の CEO が Windows であるというのは誤りである．また，上記の手法では，単語間の関係を十分に表現できないため，精度や平均逆順位 (MRR) が下がってしまう．例えば，上記の手法では語彙パターンを抽出する時に， A と B との間の単語しか調べず， A と B の前後にある単語を調べないので，前後も含む文脈の情報を考慮することができないという欠点がある．更に，潜在関係検索のクエリを処理する時に，キーワードベースの検索エンジンに数 10 個のクエリを投げているので，速度の点で実用レベルの応用には厳しいと考えられる．

本研究では関係抽出の手法を使い，自動的にエンティティペアやエンティティ間の関係を特徴づける語彙パターンのインデックスを作成し，高精度な関係類似度計算[2),7)] の研究成果を利用し，高速かつ高精度の潜在関係検索を実現する．

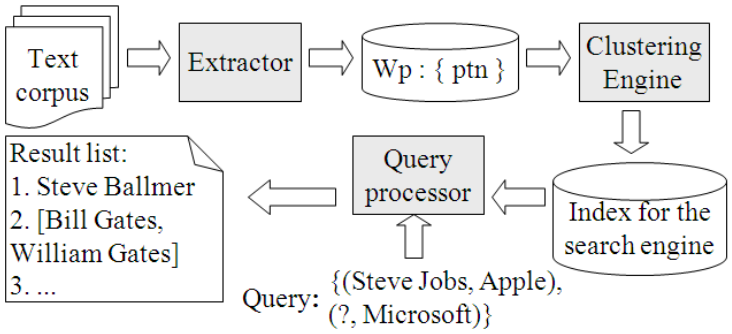


図 2　潜在関係検索エンジンの全体図
Fig. 2　Overview of the relational search engine

3. 潜在関係検索システムの概要

本システムの全体図を **図 2** に示す．本手法では，高速で検索を実現するために，エンティティペアの情報のインデックスを作成する．インデックスを作成するためにまず，Web ページなどのドキュメントをクロールし，テキストコーパスを作成する．Extractor は，テキストコーパス中の各ドキュメントからエンティティペアを自動的に発見し，エンティティペアのインデックスを作成する．エンティティペア間の関係類似度を計算するために，エンティティペアの 2 つのエンティティの関係を抽出し，表現する必要がある．そこで，あるエンティティペアの関係を表す特徴量として，そのエンティティペアにおける，2 つのエンティティ間の関係を表す語彙パターンを抽出する．例えば Extractor は，文 “Steve Jobs is the CEO of Apple” から，エンティティペア (Steve Jobs, Apple) を抽出し，“Steve Jobs” と “Apple” との関係を特徴づけるものとして，“X is the CEO of Y”,“X ∗ CEO ∗ Y” などの語彙パターンを抽出する (ここで，各特徴量の値がエンティティペアに依存しないように，文中の各エンティティを変数 (X, Y) で置き換えた．また，“∗” はワイルドカード記号で，0 個以上の単語を意味する)．この処理が終わると，エンティティペアと，それらと共起する語彙パターンのデータベースが得られる．次に，抽出されたエンティティや語彙パターンは Clustering Engine で処理され，意味的に類似する語彙パターンは 1 つの語彙パターンクラスタにまとめられる．これにより，類似するエンティティペアにおける類似語彙パターンの，異なる表現形式を吸収することができ，完全マッチング語彙パターンの低頻度問題を解決する．また，1 つのエンティティの異なる表現形式 (例えば，“United States” と “America” など) に対しても，1 つのエンティティクラスタにまとめる作業を行い，異なる表現形式を同一的に扱えるようにする．こうして，関係検索のインデックスが完成する．Query Processor は検索クエリ (例えば {(Steve Jobs, Apple), (?, Microsoft)}) に対して，検索インデックスを調べ，関係類似度の高いエンティティペアで，かつ片方のエンティティが入力されたキーエンティティとマッチングするペアを探す．最後に，Query Processor はエンティティクラスタの情報を用い，類似する検索結果エンティティを一個の結果クラスタにまとめ，クラスタの平均類似度を計算する．この平均類似度に基づいてソートされたリストが，検索結果リストとなる．このリストの各要素は 1 つエンティティクラスタであるので，それぞれに複数のエンティティ ([Bill Gates, William Gates] など) が入っていることがある．このように，本検索エンジンではエンティティの複数の表現形式を出力することができる．

4. エンティティペア抽出と関係表現

4.1 エンティティペアの抽出

本手法ではまず，テキストドキュメント (Web ページ) を文ごとに区切り，各文を形態素解析器や固有表現抽出器 ⋆1 に入れ，文を形態素に分け，また固有表現を抽出する．現在の実装では，エンティティは固有表現だけを検索対象にしているが，一般的にどの単語種類でもペアのインデックスを作成すれば検索できる．文の解析結果の中で 2 つ以上のエンティティがあれば，文中の前後順序を保ったすべてのエンティティのペアを抽出する．ここで注意すべき点は，エンティティペアの関係種類が事前に分かる必要はなく，すべてのペアを抽出するということである．例えば，“It is now official: Microsoft acquires San Francisco based company Powerset for $100M.” という文からは，3 つのエンティティペア (Microsoft, San Francisco), (Microsoft, Powerset), (San Francisco, Powerset) が抽出される．ここで，実際には有意な関係を持っていないにもかかわらず，偶然抽出されたペアをフィルタリングするために，以後出現頻度 5 以上のペアだけを検索対象として扱う．また，文中での距離が遠いエンティティ同士は明確な関係を持たない可能性が高いので，その間の距離がある閾値 M よりも大きいエンティティペアは検索対象とせず，抽出しない．

⋆1 Stanford POS Tagger と Stanford Named Entity Recognizer を使用している．
http://nlp.stanford.edu/software/CRF-NER.shtml

表 1　エンティティペア (Microsoft,Powerset) の語彙パターン抽出例
Table 1　The process of extracting lexical patterns for the entity pair (Microsoft, Powerset)

文 (NE tagged)	It is now official: Microsoft acquires San Francisco based company Powerset for $100M.
変数置き換え	It is now official: X acquires San Francisco based company Y for $100M.
サブシーケンス	now official: X acquires San Francisco based company Y for $100M
Stemming	now offici : X acquir San Francisco base compani Y for $100M
語彙パターン	X acquir * Y, X * San Francisco * Y, offici: X acquir * Y, now offici: X acquir San Francisco * Y, X * compani Y for $100M, X acquir San Francisco base compani Y, . . .

4.2　エンティティ間関係の表現

エンティティペア間の関係類似度を計算するためには，各エンティティペアの 2 つのエンティティ間の意味関係を，何らかの特徴ベクトルで表現する必要がある．本研究では先行研究[2),5),7)] に従い，エンティティ間の関係をそれらのエンティティが出現した文の周辺文脈を考慮して表現する．具体的には，エンティティペアにおける意味関係を，それぞれエンティティで挟まれる語彙パターン (lexical pattern) と，それらの前後に出現する語彙パターンの頻度で表現する．つまり，語彙パターンの抽出対象となるのは，各エンティティの前後の単語列を含む部分単語列である．ここで部分単語列とは，文の単語列のある位置から始まる，連続した単語列である． しかし，後で説明するように，ワイルドカード “* ” 記号を導入することで，文のサブシーケンス (連続しない部分単語列) も取れることになる．また，語彙パターンだけでなく，品詞パターン (つまり，語彙パターンの各語彙の品詞からなるパターン) や係り受けパターン (語彙パターンに対応する文の係り受け解析で得られた係り受け関係のパターン) などを利用し，特徴ベクトルを作ることもできる．語彙パターンの抽出対象を各エンティティの前後を含むエンティティペアの周辺の単語列 (文のサブシーケンス) だけにするのは，遠く離れた語彙パターンはエンティティと関係の薄いものである可能性が高いからである．また，文全体を抽出対象にすると， 語彙パターンの数は膨大になる傾向がある．そこで，形態素解析と固有表現抽出を行った文 S におけるエンティティC と D の関係を特徴づける語彙パターンを抽出するために，以下の S の部分単語列 Z を調べる:

$$Z = b_1 b_2 ... b_k C w_1 w_2 ... w_m D a_1 a_2 ... a_p$$

ここで，Z は C の前の k 個の単語，エンティティC 自身，C と D の間の単語列，エンティティD，及び D の後ろの p 個の単語からなる単語列である． **表 1** に $k = p = 3$ の時，文 “It is now official: Microsoft acquires San Francisco based company Powerset for $100M.” におけるエンティティペア (Microsoft, Powerset) について語彙パターンの抽出を行った例を示す． 表 1 から分かるように，この例では文の Z が以下のようになる (コロン記号 “:” や “$” 記号は 1 語とする):

$$Z = \textit{now official: X acquires San Francisco based company Y for \$100M}$$

この単語列では，エンティティC が変数 X に， D が変数 Y に置き換えられている．これは，語彙パターンを特定のエンティティペアに依存しないように (あらゆるエンティティペアで共有できるように) 表現する必要があるからである．

次に，単語の語形変化 (活用など) の違いを吸収するために，英語の word stemmer [⋆1] を使い，各単語を stemming する．この時 stemming された単語列 Z から，すべての n-grams ($n \leq (M+2)$; M は前節で説明した閾値) を生成する．ここで $n = (M+2)$ を許可するのは，X と Y とその間の単語列 ($Xw_1w_2...w_mY$) を語彙パターンとして抽出したいことによる．また，生成された n-gram の集合の中で，b_i だけを含む n-gram (例えば $b_1b_2b_3$) と a_i だけを含む n-gram (例えば $a_3a_4a_5$) を除去する．更に w_i だけを含む n-gram ($w_iw_{i+1}...w_j$) に対して，$X * w_iw_{i+1}...w_j * Y$ に変換する (“*” はワイルドカード記号で， 0 以上の語を意味する． “*” を入れることで パターンのマッチング確率が高くなるので， 検索の再現率を高めることができる)．Y を含まない n-grams (例えば，$b_kXw_1w_2$) に対しては，最後に “*Y” をつける ($b_kXw_1w_2 * Y$)．同様に，X を含まない n-gram(例えば，$w_iw_{i+1}...w_mYa_1$) について，最初に “X*” を付加する ($X * w_iw_{i+1}...w_mYa_1$)．最後の語彙パターンの集合は，上記のパターン集合から，content word (ストップワードではなく，変数 X, Y でもない単語) を一つ以上含んだ語彙パターンだけを選ぶことで作る．例えば上記の列 Z からは， 表 1 の最後の行で示した語彙パターンが生成される．

このように，本研究で用いた関係の表現手法は先行研究[2),5),7)] の手法と同じであるが，関係検索の再現率を上げるために，パターン抽出アルゴリズムに次に述べる工夫を加えている．まず第一の工夫として，語彙パターンの中の単語を stemming し活用形の違いを吸収する．これにより類似度計算の際，違う活用形を持った語彙パターンが同一に扱われ，再現率を高くすることができる．また第二の工夫として，n-gram は X と Y 両方を含むという条件を省く．その代わり，“X*” や “*Y” を付加することで，語彙パターンとエンティティとの相対位置を区別する．これにより，二つのエンティティペアが共通の語彙パターンを持つ確率が高くなるので，検索の再現率を向上できる．例えば，“Obama is the 44th and

⋆1 Python NLTK toolkit の PorterStemmer を使用

current president of the U.S.” と “Sarkozy is the current president of France” という文において，もし語彙パターン “current president of” を許可 (つまり語彙パターン “X* current president of * Y” を生成) すると，(Obama, U.S) と (Sarkozy, France) が共通の語彙パターンを持つようになる．

エンティティペアと語彙パターン抽出ステップでは，エンティティペアの出現頻度と語彙パターンの出現頻度を記録する．従って，エンティティペアの出現頻度とパターンの出現頻度，各パターンとエンティティペアとの共起頻度の情報がデータベースに記録される．ここで，エンティティペア wp と共起する語彙パターンの集合を $\mathbf{P}(wp)$ と定義する:

$$\mathbf{P}(wp) = \{p_1, p_2, \ldots, p_n\} \tag{1}$$

更に，ソースペア (入力されたペア) と少なくとも 1 個のパターンを共有するエンティティペアを高速に検索するために，各語彙パターンをキーとして持ち，その語彙パターンと共起したエンティティペアの集合を値として持つ転置インデックスも作成する．ここで語彙パターン p と共起したエンティティペアの集合を，$\mathbf{W}(p)$ と定義する:

$$\mathbf{W}(p) = \{wp_1, wp_2, \ldots, wp_m\} \tag{2}$$

また，エンティティペア wp_i が語彙パターン p_j と同一文内で共起する頻度を $\mathrm{f}(wp_i, p_j)$ とした時，語彙パターン p のエンティティペア頻度ベクトルを以下のように定義する:

$$\mathbf{\Phi}(p) = (\mathrm{f}(wp_1, p), \mathrm{f}(wp_2, p), \ldots, \mathrm{f}(wp_m, p))^{\mathrm{T}} \tag{3}$$

同様に，エンティティペア wp の語彙パターン頻度ベクトルを以下のように定義する:

$$\mathbf{\Psi}(wp) = (\mathrm{f}(wp, p_1), \mathrm{f}(wp, p_2), \ldots, \mathrm{f}(wp, p_n))^{\mathrm{T}} \tag{4}$$

4.3　語彙パターンクラスタリング

語彙パターンの中の各単語が stemming され，活用形の違いが吸収されたとしても，類似関係を持つ 2 つのエンティティペアが同一の語彙パターンを持つ確率は依然として低い．これは，自然言語における，様々な言い換え表現を持つ関係が存在するからである．例えば，“X acquired Y” と “X bought Y” は意味的に類似するが，文面上では全く異なる表現である．そこで，二つの異なるパターン p と q が類似する意味関係を持つかどうかを判断するために，p と q の意味的類似度を p と q の語彙パターン頻度ベクトル $\mathbf{\Phi}(p)$ と $\mathbf{\Phi}(q)$ のコサイン類似度として定義する．次に，Bollegala ら[2] が提案した語彙パターンの逐次的クラスタリングアルゴリズムを使い，意味的に類似する語彙パターンを 1 つのパターンクラスタにまとめる．このアルゴリズムではまず，語彙パターンの集合をパターンの出現頻度で降順にソートする．次に頻度の高いパターンの順にパターンを取り，そのパターンと最大の類似度を持つパターンクラスタを見つける．両者間の類似度が閾値 θ 以上の場合，パターンをそのパターンクラスタに追加する．そうでない場合，このパターン自身が新たなパターンクラスタとなる．パラメータ θ は Bollegala ら[9] が提案した手法で自動的に決めることもできるが，本研究では潜在関係検索の性能が最大になるように実験的に決める．ここでは，偶然抽出されたパターンやノイズパターン (“now offici: X * San Francisco * Y” など) をフィルタリングするのと，クラスタリングのアルゴリズムの実行時間を減らすために，出現頻度が 10 以上のパターンだけをクラスタリングの対象にする．

本研究における逐次的クラスタリングアルゴリズムの計算量は，$\mathrm{O}(n \log n + \kappa n)$ である (n は入力語彙パターンの数，κ は結果クラスタの数，通常 $\kappa \ll n$ である)．この計算量は k-means や擬似階層クラスタリングなどの他のクラスタリングアルゴリズムよりも遥かに小さい．$\mathrm{O}(n \log n)$ はソートに費やす計算量で，これには効率的なアルゴリズムが多数存在している (実際の逐次的クラスタリングの計算量は $\mathrm{O}(\kappa n)$ であり，$\kappa < \log(n)$ の時クラスタリングアルゴリズムの計算量は $\mathrm{O}(n \log n)$ となる)．

4.4　エンティティクラスタリング

同一のエンティティはしばしば複数の表現形式を持つ．例えば，“United States” というエンティティは “U.S.” などの形で略記されることが多いが，“America” という別名も持つ．そこで関係検索を実行する時に，同一のエンティティの表現形式の違いを吸収し，同一のエンティティとして扱うために，エンティティのクラスタリングを行う．本研究では Bollegala ら[2] が提案した語彙パターンのクラスタリング手法と同様の，分布仮説 (distributional hypothesis) に基づいたエンティティクラスタリングの手法を提案する．例えば，エンティティ “United States” はよくエンティティ “Obama” と “Clinton” などと一緒にペアをなす．同様に，エンティティ “U.S.” もエンティティ “Obama” と “Clinton” と一緒にペアをなす．2 つのエンティティが同じようなエンティティ集合と一緒にペアをなすということは，それらのエンティティが似ているコンテキストで出現することである．分布仮説により，その 2 つのエンティティが類似することが分かる．即ち，2 つのエンティティがそれぞれペアとなるエンティティ (以下，パートナーエンティティと呼ぶ) の集合を持つ時，その集合が類似していればエンティティ同士は類似した意味を持つと考える．それ故，2 つのエンティティ間の類似度は，それぞれのパートナーエンティティの頻度ベクトル同士のコサイン類似度で定義される．ここで，4.1 節の手順を実行した後、抽出されたエンティティの集合を $\mathbf{E}$ とし，エンティティペア (w_i, w_j) の出現頻度を $\mathrm{f}(w_i, w_j)$ とする．次に，エンティティ $w_j \in \mathbf{E}$ のパートナーエンティティの頻度ベクトルの i 番目の要素を以下のように定義する．

$$h(w_j, i) = \mathrm{f}(w_j, w_i) + \mathrm{f}(w_i, w_j), \forall w_i \in \mathbf{E} \tag{5}$$

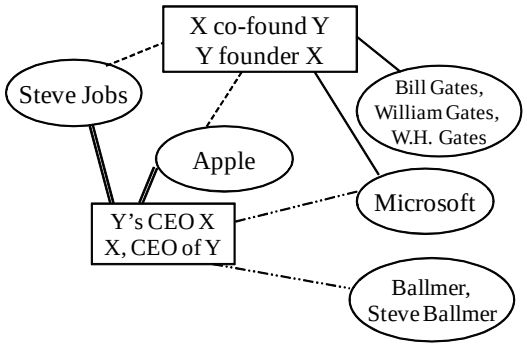


図 3　潜在関係検索エンジンのインデックスのモデル．(楕円はエンティティクラスタ，長方形は語彙パターンクラスタを表す)
Fig. 3　Model of the index for latent relational search

即ち，パートナーエンティティの頻度ベクトルでは，右側のパートナーエンティティと左側のパートナーエンティティの出現頻度両方を利用する．すると，エンティティC と D それぞれのパートナーエンティティの頻度ベクトル同士のコサイン類似度は，以下の式で計算できる．

$$\mathrm{simW}(C,D)=\frac{\sum_i h(C,i)\cdot h(D,i)}{\sqrt{\sum_i h^2(C,i)}\sqrt{\sum_i h^2(D,i)}} \tag{6}$$

この定義を用いて，上記のパターンクラスタリングのアルゴリズムと同様に，エンティティのクラスタリングを行う．ここで，エンティティクラスタリングにおける類似度の閾値を ξ とする．

語彙パターンのクラスタリングとエンティティのクラスタリングを実行すると，**図 3** のように，エンティティクラスタや語彙パターンクラスタ，エンティティクラスタの関係などがインデックスの情報から得られる．この図によると例えば，エンティティクラスタ “Steve Jobs” は語彙パターンクラスタ “X co-found Y” と “X, CEO of Y” を経由して，エンティティクラスタ “Apple” とリンクされている．

5. 候補検索とランキング

5.1 候補の検索

クエリ {(A, B), (C, ?)} が入力された時，潜在関係検索エンジンは (C, X) の形で，(A, B) と類似する関係を持つペアの集合 $\Re$ を検索する必要がある．ここで，ソースペア (A, B) を s とし ($s=(A,B)$)，また候補となるペアを $c=(C,X)$ とする．候補ペアの集合 $\Re$ を検索するためにまず，ソースペア s の語彙パターン集合 $\mathbf{P}(s)$ 中の各語彙パターンを調べる．それぞれの語彙パターン $p\in\mathbf{P}(s)$ について，p の頻度が 10 以上であれば，p のエンティティペア集合 $\mathbf{W}(p)$ を調べる．この時 $\mathbf{W}(p)$ 中で (C, X) の形を持つペアで，出現頻度 5 以上のものを候補集合 $\Re$ に追加する:

$$\Re=\bigcup_{p\in\mathbf{P}(s)\wedge\mathrm{freq}(p)\geq 10}\{wp\in\mathbf{W}(p)|(wp[0]=C)\wedge\mathrm{freq}(wp)\geq 5\} \tag{7}$$

これにより，$\Re$ 中のエンティティペアは，ソースペア s と少なくとも 1 つ以上の語彙パターンを共有する．また，この条件を利用することにより，候補数を限定することができ，候補の検索プロセスを高速化できる．

5.2 候補のランキング

前のステップでの処理により候補数はある程度限定されたが，候補集合 $\Re$ のサイズは依然として大きい場合がある．しかしながら，全ての候補が適切な候補であるとは限らない．なぜなら，“1 つ以上の語彙パターンを共有する” という条件を満たすペアは，必ずしもソースペアと類似の関係ではないからである．故に，候補ペアとソースペアとの関係類似度を利用することで，不適切な候補のフィルタリングと，候補リストのランキングを行う．ソースペアとの関係類似度の高いペアは，類似した関係を持っているので，正解になる可能性が高い．そこで先行研究[2),5),7)] に従い，s と c の語彙パターンの頻度ベクトル $\mathbf{\Psi}(s)$，$\mathbf{\Psi}(c)$ を利用することで，ソースペア s と候補ペア c との間の関係類似度を計算する．s と c の間の関係類似度 $\mathrm{relsim}(s,c)$ は，語彙パターンの頻度ベクトルの擬似コサイン類似度として定義され，この計算手順を **図 4** に示す．擬似内積 $\mathbf{\Psi}(s)\cdot\mathbf{\Psi}(c)$ は次のように計算する．まず，語彙パターン p が $\mathbf{P}(s)\cap\mathbf{P}(c)$ に属すならば，普通の内積と同様に，$\mathrm{f}(s,p)\cdot\mathrm{f}(c,p)$ を内積に加える．パターン p が $\mathbf{P}(c)$ に属しているが，$\mathbf{P}(s)$ には属していない場合，$\mathbf{P}(s)\backslash\mathbf{P}(c)$ (差集合) に属し，かつ p と同じ語彙パターンクラスタに属すパターン q を見つける．もし複数の q が存在していれば，出現頻度が最大のものを選ぶ．選んだパターン q は，p と同じクラスタに属するため p と意味的に類似するので，$\mathrm{f}(s,q)\cdot\mathrm{f}(c,p)$ を内積に加える．また，選んだ q を以降の内積計算プロセスから除外するため，マークしておく (図 4 では q を $\mathbf{T}$ に追加する)．ここで，適切な候補の集合 Γ を，$\Re$ の中で関係類似度 $\mathrm{relsim}(s,c)$ がある検索類似度の閾値 σ 以上であるような c の集合として定義する．

$$\Gamma=\{c\in\Re|\mathrm{relsim}(s,c)\geq\sigma\} \tag{8}$$

```
function relsim(s, c)
  //Initialize the inner product to 0
  ρ ← 0
  //Initialize the set of used patterns
  T ← {}
  for pattern p ∈ P(c)
    if p ∈ P(s) then
      ρ ← ρ + f(s, p)f(c, p)
      T ← T ∪ {p}
    else
      Ω ← the cluster that contains p
      max ← −1
      q ← null
      for pattern p_j ∈ (P(s)\P(c))\T
        if (p_j ∈ Ω)∧(f(s, p_j) > max) then
          max ← f(s, p_j)
          q ← p_j
        end if
      end for
      if max > 0 then
        ρ ← ρ + f(s, q)f(c, p)
        T ← T ∪ {q}
      end if
    end if
  end for
  return ρ/(|Ψ(s)| · |Ψ(c)|)
end function
```

図 4　エンティティペア s と c との関係類似度の計算アルゴリズム
Fig. 4　Algorithm for calculating the relational similarity between two entity pairs s and c

表 2　評価用の関係
Table 2　Relation types for evaluation

関係	ペアの数	エンティティペアの例
人-出身地	20	(Franz Kafka, Prague), (Andre Agassi, Las Vegas), (Charlie Chaplin, London)
会社-本部地	15	(Google, Mountain View), (Microsoft, Redmond), (Apple, Cupertino)
CEO-会社	16	(Eric Schmidt, Google), (Steve Ballmer, Microsoft), (Carol Bartz, Yahoo)
Acquirer - Acquiree	48	(Google, YouTube), (Microsoft, Powerset), (Yahoo, Kelkoo)

また，クエリ {(A, B), (C, ?)} に対し，上記の処理をその転置クエリ {(B, A), (?, C)} についても行い，転置クエリにおける候補で，かつ Γ に含まれる候補のスコア (関係類似度) を考慮して，最終的な候補スコアを計算する．転置クエリのスコアも利用するのは，関係類似度がエンティティペアの転置操作に対して不変であるという性質があるからである[10]．ここで，$s' = (B, A)$，$c' = (X, C)$ とすると，Γ 中の候補 $c = (C, X)$ のソースペア $s = (A, B)$ に対するスコアは下記の式で定義される:

$$\chi(s, c) = \text{relsim}(s, c) + \frac{1}{2}\text{relsim}(s', c') \tag{9}$$

(relsim(s', c') が σ 以上の時だけ，c' が転置クエリの候補集合に入る)．上記のスコアにおいて，転置クエリで得られた類似度の重みを 1/2 にしているのは，もともとのクエリで得られた類似度を優先させるためである．

最後に，エンティティクラスタの情報を使い，同一エンティティの複数の表現形式や意味的に類似するエンティティを，1 つの候補クラスタにまとめる．候補 $c_i = (C, X_i)$ と $c_j = (C, X_j)$ があった時に，X_i と X_j が同じエンティティクラスタに入れば，c_i と c_j は同じ候補クラスタに合併され，最終的な候補集合 Γ′ が

$$\Gamma' = \text{Merged}(\Gamma) \tag{10}$$

となる．また，各候補エンティティペア c_i が (C, X_i) であるとすると，候補クラスタ $K = \{X_1, X_2, \ldots, X_k\}$ のスコアは以下の式で定義される

$$\text{score}(s, K) = \frac{1}{k}\sum_{i=1}^{k}\chi(s, c_i) \tag{11}$$

結果リストは，候補クラスタのリストを，クラスタのスコアで降順にソートしたものである:

$$\mathbf{R} = \text{SORTED}(\Gamma') \tag{12}$$

6. 実験による評価

6.1 パラメータの決定

提案手法には次の 3 つのパラメータがある: 語彙パターンクラスタリングにおける類似度の閾値 θ とエンティティクラスタリングにおける類似度の閾値 ξ，検索候補選択のための類似度の閾値 σ．これらのパラメータの値を適切に決定するために，**表 2** に示した 4 種の関係を使い，パラメータの値を変化させながら検索エンジンの性能を測定した．これらの関係は，関係抽出や関係類似度を測定する研究でよく用いられるものである[2),6),11)]．

6.1.1 評価用のデータセット

システムの性能を評価するためには，上記の 4 種の関係のインスタンスを含んだテキストコーパス (図 2 の “Text corpus”) を用意する必要がある．一般的に，図 2 の Text corpus には評価対象になる関係 (上記の 4 種の関係) とは関連しないドキュメントも含まれている．例えば，Text corpus には 表 2 で示した関係の情報の他，Web ページ上の広告のテキストやノイズ的テキストがよく入っている．それだけでなく，評価対象の関係情報とは全く

関連しない記事などを含むことも多い．しかしこれらの場合にも，図 2 の Extractor は正しくエンティティペアやペアの語彙パターンを抽出できる．一方で，評価対象と関連しない記事の割合が大きくなると，評価に必要な関連記事を十分な数にするためには，膨大な量のコーパスが必要となる．またその時，エンティティペアや語彙パターンが膨大な数となり，パラメータを変化する度に長時間を費やすため，評価にかかる時間とコストが大きくなる．そこで，ドキュメントを評価対象の関係情報を含んだものに限定するために，表 2 にある関係の情報 (関係を表すキーワード) とそのインスタンス (エンティティペア) を使い，Google [⋆1]にクエリを投げ，各クエリの検索結果から上位 100 件の URL を集めた．これらの URL を Crawler に入れ，Web ページをダウンロードし，図 2 における Text corpus を作成する．このコーパスには，12000 個のドキュメント (Web ページ) が入っている．これらのドキュメントには，表 2 におけるエンティティペアや関係の関連情報が入っているが，関連エンティティの数や関係の数は正確には特定できない．なぜなら，例えば Microsoft による Powerset の買収の記事の中に，Apple による Emagic の買収の情報も入っている，といったことが起こり得るからである ((Apple, Emagic) というペアが 表 2 に入っていない場合にも (Apple, Emagic) ペアが抽出される)．次に集められたドキュメント集合から，図 2 の Extractor が HTML タグをすべて削除し，テキスト内容を取得し，エンティティペアと語彙パターンの抽出を行う．その解析結果として，113,742 個のエンティティペアと 2,069,121 個の語彙パターンが抽出された．この中で頻度 5 以上のエンティティペアの数は 4,103 個で，頻度 10 以上の語彙パターンの数は 27,568 個である．ここでノイズのパターンやエンティティペアを除去するため，システムは頻度 5 以上のエンティティペアだけを検索対象とする．また，頻度 10 以上のパターンだけを，パターンクラスタリングアルゴリズムの対象とする (ただし，エンティティペア間の類似度を計算する時には，頻度 10 以下のパターンも考慮する)．

6.1.2　エンティティクラスタリングアルゴリズムの性能

エンティティクラスタリングアルゴリズムのパラメータ決定を行うため，エンティティクラスタリングの類似度の閾値 ξ を変化させながら，その精度を測定した．この測定のために，複数のエンティティを含んだクラスタ (シングルトンでないクラスタ) を調べ，クラスタ内のすべての単語が実際に同じエンティティの異なる表現であるかを調べる．クラスタ内のすべての単語が同じエンティティを指していれば，このクラスタを正しいクラスタとし，

⋆1 http://www.google.com

表 3　エンティティクラスタリングの精度
Table 3　Precision of the entity clustering algorithm

ξ	シングルトンでないクラスタの数	正解のクラスタ数	精度 (%)
0.1	222	92	41.44
0.15	137	79	57.66
0.2	101	69	68.32
0.25	74	61	82.43
0.3	51	51	100.00

そうでなければ誤りだと判断する．例えば，[Steve Ballmer, Steven Ballmer, Ballmer] は同じエンティティを指すので，このクラスタは正しいクラスタとなる．エンティティクラスタリングアルゴリズムの精度を**表 3** に示す．エンティティクラスタリングの類似度の閾値 ξ が小さければ，シングルトンではないクラスタが多数生成され，その精度も低い．閾値 ξ を大きくすると，シングルトンではないクラスタの数が少しずつ減少し，クラスタリングの精度は向上する．ξ が 0.3 以上の場合，シングルトンではないクラスタの精度が 100%に達する．これまでに説明したように，コーパス内で類似するエンティティの数は特定できないので，コーパス内のシングルトンではないクラスタの数も特定できず，クラスタリングの再現率は計算できない．しかし，検索のタスクに対しては精度が重視されるので，クラスタリングの精度の方がより重要だと考えられる[12]．故に，シングルトンではないクラスタの精度を 100%に保ちつつ，できるだけシングルトンでないクラスタ数を大きくするため，以降の実験では ξ を 0.3 に設定した．

6.1.3　語彙パターンクラスタリングにおける類似度の閾値

語彙パターンクラスタリングにおける類似度の閾値 θ を適切な値に決定するために，閾値 θ を変化させながら，検索エンジンの精度，再現率，F 値を測定した．テストクエリの集合は 表 2 に示した 4 つの関係を含めた，$i \neq j$ であるような全ての $(\mathrm{A}_i, \mathrm{B}_i)$，$(\mathrm{A}_j, \mathrm{B}_j)$ から作った $\{(\mathrm{A}_i, \mathrm{B}_i), (\mathrm{A}_j, ?)\}$ のようなクエリで構成される．表 2 には，1 つしか正解数を持たないクエリと，複数の正解を持つクエリがある．例えば，{(人 $_i$， 出身地 $_i$)，(人 $_j$， ?)} のようなクエリでは，正解は人 $_j$ の出身地の 1 つだけである．或いは例えば $\{(\mathrm{Acquirer}_i, \mathrm{Acquiree}_i), (\mathrm{Acquirer}_j, ?)\}$ のクエリに対し，正解は $\mathrm{Acquirer}_j$ の買収した会社の集合となるため，複数存在する可能性がある．また，システムは候補を検索できない時，またはすべての候補ペアとソースペア間の類似度が低い時に，候補結果を出力せず “No answer” を出力する．1 つしか正解を持たないクエリに対して，最上位の結果だけを調べ，正解かどうか

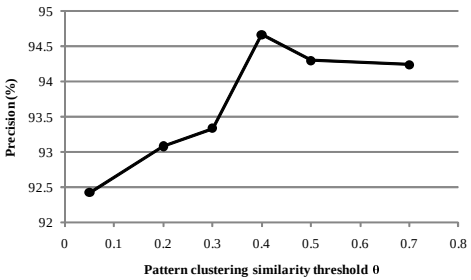


図 5　平均精度と閾値 θ の関係 (ξ=0.3, σ=0.05 の時)
Fig. 5　The relation between the average precision and the threshold θ (at ξ=0.3, σ=0.05)

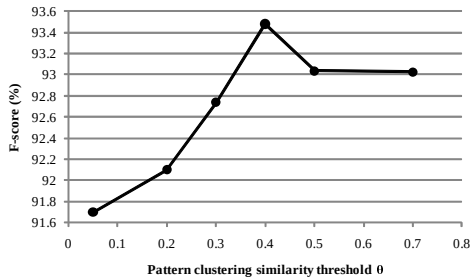


図 6　三種の関係 (人-出身地，会社-本部地，CEO-会社) の平均 F 値と閾値 θ の関係 (ξ=0.3, σ=0.05 の時)
Fig. 6　The relation between the average F-score of three relation types (Person-Birthplace, Company-Headquarters, CEO-Company) and the threshold θ (at ξ=0.3, σ=0.05)

を判断する．1 つしか正解を持たないテストクエリの数を Q，候補を 1 つ以上出力したクエリ (“No answer” でないクエリ) の数を a とし，その中で正解が最上位にあるクエリの数を c とする．ここで，精度を c/a，再現率を c/Q と定義する．正解を複数持つクエリに対しては，トップ 10 の検索結果を調べる．正解を複数持つ全てのクエリに対して，調べた結果の数を a とし，その中で正解の数を c とする．その時，正解を複数持つクエリセットの精度を c/a と定義する．また，このクエリセットについては，再現率を評価できない．これは以前説明したように，コーパスの中で関連エンティティがどの程度あるかは特定できず，評価の際，検索結果のトップ 10 しか考慮しないからである．

図 5 は θ を変化させた時の， 表 2 の 4 種の関係におけるクエリの平均精度の推移を示している (エンティティクラスタリングにおける類似度の閾値 ξ を 0.3，候補検索における類似度の閾値 σ を 0.05 としている)．同様に， **図 6** に各 θ における，再現率が計算可能な 3 種の関係 (人-出身地，会社-本部地，CEO-会社) を持つ，クエリの F 値の平均を示す．これらの結果から分かるように， θ が 0.4 の時，精度と F 値の両方が最大となっている． **表 4** にテストセットのクエリと検索結果のいくつかの例を示す．また，候補検索における類似度の閾値 σ を [0.03, 0.2] の間で変化させても，これらの図の形 (ピークの位置) は変化しない．ただし，σ が 0.05 の時に，ピークにおける精度と F 値の値が最大となり，σ を 0.2 よりも大きくすると，再現率が大きく減少し， F 値も大きく減少することが確認された．そこで

表 5　関係ごとの検索エンジンの性能 ($\theta = 0.4, \xi = 0.3, \sigma = 0.05$)．Acquirer - Acquiree 関係については 6.2 節で説明するように，再現率が計算できない．括弧内はベースラインのパターン抽出アルゴリズム使用時の値．
Table 5　Performance of the search engine for each relation type. The recall for the Acquirer - Acquiree relation type can not be evaluated, as explained in Section 6.2. The numbers in parentheses are the values while using the baseline lexical pattern extraction algorithm

関係の種類	精度	再現率	F 値
人-出身地	98.89 (94.37)	98.89 (74.44)	98.89 (83.23)
会社-本部地	90.59 (79.22)	85.56 (67.78)	88.00 (73.05)
CEO-会社	95.56 (86.67)	95.56 (86.67)	95.56 (86.67)
Acquirer - Acquiree	81.34 (53.72)	-	-
平均	91.60 (78.49)	93.34 (76.30)	94.15 (80.98)

以降の実験は，θ を 0.4 に，σ を 0.05 に設定して行う．

6.2　平均精度と F 値

選択されたパラメータ ($\theta = 0.4, \xi = 0.3, \sigma = 0.05$) におけるシステムの平均精度と F 値を評価するために，パラメータ調整用のコーパスとは別に，6,000 個のドキュメントを含んだコーパスを用意した．別のコーパスを使用するのは，選択されたパラメータの値の普遍性 (どのコーパスにも同程度の性能を出す) を検証するためである．精度と F 値を評価するコーパスは， 表 2 で示した関係を含むが，関係のインスタンス (エンティティペア) が，パラメータ決定用のコーパスとは完全に異なる．またコーパスの生成手法は，パラメータ調整用のコーパスの時と同じである．

表 5 にそれぞれの関係のクエリセットの平均精度，再現率及び F 値を示す．括弧内の数は 4.2 節で説明したパターン抽出アルゴリズムから，本研究での 2 つの工夫点 (stemming 及び X，Y を含まないパターンも許可すること) を除いた時の性能である．この 2 つの工夫を除いたパターン抽出アルゴリズムを “ベースラインのパターン抽出アルゴリズム” と呼ぶ．

表 5 にあるように， 正解を複数持つ Acquirer - Acquiree 関係に関するクエリセットの再現率と F 値は，以前説明した理由で計算することはできない．正解を 1 つしか持たないクエリに対しては，提案手法は高い精度 (最上位の結果の精度) と再現率を得た．また，正解を複数持つ関係のクエリに対しても，81.34%の精度 (トップ 10 結果の精度) を得た．平均すると，提案手法は 91.60% の精度を得た．

また，ベースラインのパターン抽出アルゴリズムと比較すると，本研究で提案したパターン抽出アルゴリズムでは再現率の平均が 22.3%向上した．従って，提案したアルゴリズムに

表 4　クエリと検索結果の例
Table 4　Some example queries and search results

クエリ	結果	正解?	共通語彙パターンの例
{(Franz Kafka, Prague), (Albert Einstein, ?)}	[Ulm]	Yes	X * born in Y; X wa born in Y; X wa born in Y in; X *, born * Y ...
{(Yahoo, SunnyVale), (Apple, ?)}	[Cupertino]	Yes	X * headquart in Y, calif.; X * locat in Y; at X headquart in Y ...
{(Michael Dell, Dell), (?, Google)}	[Eric Schmidt, Dr. Eric Schmidt]	Yes	X, CEO of Y; X *, chairman * Y; X, chairman and CEO * Y ...
{(Microsoft, Hotmail), (Google, ?)}	[Greenborder, Greenborder Technologies]	Yes	X acquir * Y; X bought Y; X's acquisit of Y; X announc * Y; X * plan to * Y ...
{(Google, YouTube), (Microsoft, ?)}	[Yahoo]	No	X * purchas * Y; X * buyout of Y; X * interest * Y; deal between X and Y; X * close to * Y ...

より再現率を大幅に向上できた．更に，提案したアルゴリズムを使用すると，共通する語彙パターン数が大きくなるので関係類似度の計算が正確になり，表 5 に示したように検索の精度も向上する．

6.3　既存研究との比較

Kato ら[3] の手法では，Yahoo のウェブ検索エンジンのインデックス [⋆1]を利用するので，膨大なインデックスデータが使える． 一方本手法では，Yahoo のインデックスと比べて 小さいインデックスを用いているが，これは潜在関係検索専用のインデックスである．そこで，提案したインデックス構築手法とランキング手法の効果を検証するために，提案手法と Kato らの手法の性能を比較する．

提案手法と Kato ら[3] の手法を，正確かつ公平に比較するのは難しい．なぜなら，実験で使った Web ページの言語 (英語-日本語) や関係 (本研究は典型的な関係を用いて評価したのに対し， Kato らは幅広い関係を用いて評価した) には差異があるからである．更に， Kato らが使ったいくつかの関係 ("Japanese local food","Japanese temple builder" など) の情報は日本語のページでしか書かれておらず，英語では同じ関係を評価できない．しかし，本研究と Kato らの研究では，いくつか共通の関係を用いて評価している (例えば，CEO-会社や買収関係) ので，それらを用いた比較は有意である．

表 6 に提案手法と Kato ら[3] の手法との性能の比較結果を示す．比較には提案手法における，正解を 1 つしか持たない 4 つ関係 (人-出身地，会社-本部地，CEO-会社，Acquirer*-Acquiree) の結果の平均と，Kato らの評価実験の結果[3] を用いる．Acquirer*-Acquiree 関係のクエリは前述した Acquirer-Acquiree クエリとは違い，{(Acquirer$_i$, Acquiree$_i$), (?, Acquiree$_j$)} のようなクエリで，正解を 1 つしか持たない関係である．Kato ら[3] は正解を 1 つしか持たない関係のクエリしか用いていないため，比較を行うためにこのクエリを導入した．表 6 における MRR は平均逆順位 (mean reciprocal rank) であり，最も上位にある正解について，その順位の逆数の平均値を計算したものである．クエリ q について，最初の正解の順位が r_q であるすると，クエリセット Q の平均逆順位は次のように定義される:

$$MRR = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{r_q} \tag{13}$$

MRR が大きければ正解の順位 (ランク) が平均的に小さい (トップにランクされる) ので，MRR が大きいほど検索エンジンの性能は良いと考えられる (MRR の最大値は 1 である)．表 6 から分かるように，提案手法の MRR は既存手法よりも 76.7%よくなっている (0.963 と 0.545)．また，表 6 における "@N" は，トップ N 個の結果の中で，正解を含むクエリの割合を表す．これらのすべての指標について，提案手法は既存手法よりも良い結果を達成している．特に，提案手法は正解を 1 つしか持たないクエリに対し，その正解の 95.0%を最上位 (最初の結果) にランクすることができている．それぞれの関係のクエリにおける具体的な MRR と@N の値を **表 7** で示している．

提案手法におけるクエリの 1 つあたりの処理時間は Intel Pentium 4, 3.0GHz のプロセッサで 10 秒以内であり， この処理速度は Kato らの手法では実現するのが難しいと考えら

表 6　単一正解クエリにおける既存手法との性能比較結果 (@N はトップ N 結果に正解があるクエリの割合).
Table 6　Comparison between the proposed method and the method in Kato et al.[3] for queries with only one correct answer.

手法	MRR	@1	@5	@10	@20
Kato らの手法[3]	0.545	43.3	68.3	72.3	76.0
提案手法	0.963	95.0	97.8	97.8	97.8

⋆1 http://developer.yahoo.com/search/

表 7　単一正解のクエリに対する提案検索エンジンの性能
Table 7　Performance of the proposed system for queries with only one correct answer

関係種類	MRR	@1	@5	@10	@20
人 - 出身地	0.994	98.9	100.0	100.0	100.0
会社 - 本部地	0.881	85.6	91.1	91.2	91.3
CEO - 会社	0.975	95.6	100.0	100.0	100.0
Acquirer*-Acquiree	1.000	100.0	100.0	100.0	100.0
平均	0.963	95.0	97.8	97.8	97.8

れる．

また提案手法が良い性能を得たのは，語彙パターンの表現方法が適切で，抽出アルゴリズムが良く働いたからであると考えられる．2 つのエンティティペアの関係類似度が高くても，各ペアのエンティティ間にあるテキストが正確にマッチすることはあまりない．例えば，“Obama is the 44th and current president of the U.S” と “Sarkozy is the current president of France” という文から，(Obama, U.S) と (Sarkozy, France) の関係類似度は高いと分かるが，それらの間にあるのテキストはマッチしない．このような場合，Kato ら[3] の語彙パターン抽出手法では，エンティティ間のテキストの正確なマッチングを要求するため，関係類似度を精度良く測定できない．一方，本研究における語彙パターン抽出アルゴリズムは上記のペアに対して，より多くの共通語彙パターンを生成する (例えば “X * president * Y”, “X * president of * Y”,“X * current president * Y” などの共通語彙パターンを生成する)．従って，これらのペアの関係類似度を高くすることができ，検索結果の精度を向上できている．

7. むすび

本稿では高速の潜在関係検索エンジンを実現するための，エンティティペア抽出とインデックス構築手法を提案した．また関係類似度計算の研究に従い，エンティティ間の関係を語彙パターンで表現し，エンティティペアの関係類似度を高精度に計算することで，ランキング付き検索結果を精度良く出力できる潜在関係検索エンジンを実現した．実際の Web コーパスを用いた評価により，本稿で提案した潜在関係検索エンジンは，高精度かつ高速で，高い平均逆順序 (MRR) を達成することが確認できた．また一方で，潜在関係検索の再現率を高めるため，本研究では従来の語彙パターン抽出アルゴリズムを改良し，検索エンジンの再現率も高めることができた．これにより，情報爆発時代に対応する潜在関係検索という新しい検索パラダイムが，実践レベルで実現可能であることを示した．なお本論文の技術の一部は特許出願を行っている[13]．

今後の課題は，潜在検索エンジンを実際に大規模な Web コーパス上で適用させることと，ユーザのフィードバックを取得し，ユーザによる検索結果の評価を解析することである．また，情報推薦システムなどに潜在関係検索を応用し，情報爆発時代においてこの新たな検索パラダイムを活かすことである．

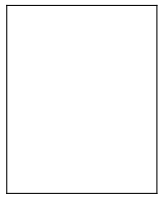
謝辞　本論文の執筆において，後藤友和氏と田中翔平氏から貴重なコメントや修正をいただきました．ここに感謝の意を表します．

参考文献

1) Veale, T.: The Analogical Thesaurus, *Proc. of IAAI'03*, AAAI Press, pp.137–142 (2003).
2) Bollegala, D., Matsuo, Y. and Ishizuka, M.: Measuring the Similarity between Implicit Semantic Relations from the Web, *Proc. of WWW'09*, ACM, pp.651–660 (2009).
3) Kato, M.P., Ohshima, H., Oyama, S. and Tanaka, K.: Query by Analogical Example: Relational Search using Web Search Engine Indices, *Proc. of CIKM'09*, pp. 27–36 (2009).
4) Turney, P.D.: A Uniform Approach to Analogies, Synonyms, Antonyms, and Associations, *Proc. of Coling'08*, pp.905–912 (2008).
5) Bollegala, D., Matsuo, Y. and Ishizuka, M.: Measuring the Similarity between Implicit Semantic Relations using Web Search Engines, *Proc. of WSDM'09*, ACM, pp. 104–113 (2009).
6) Banko, M. and Etzioni, O.: The Tradeoffs Between Open and Traditional Relation Extraction, *Proc. of ACL'08*, pp.28–36 (2008).
7) Turney, P.D.: Similarity of Semantic Relations, *Computational Linguistics*, Vol.32, No.3, pp.379–416 (2006).
8) Halskov, J. and Barriere, C.: Web-based Extraction of Semantic Relation Instances for Terminology Work, *Terminology*, Vol.14, No.1, pp.20–44 (2008).
9) Bollegala, D., Matsuo, Y. and Ishizuka, M.: Relational Duality: Unsupervised Extraction of Semantic Relations between Entities on the Web, *Proc of WWW'10*, ACM, pp.151–160 (2010).
10) Goto, T., Duc, N.T., Bollegala, D. and Ishizuka, M.: Exploiting Symmetry in Relational Similarity for Ranking Relational Search Results, *Proc. of PRICAI'10*, pp. 595–600 (2010).
11) Bunescu, R. and Mooney, R.: Learning to Extract Relations from the Web using

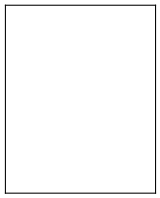
Minimal Supervision, *Proc. of ACL'07*, pp.576–583 (2007).
12) de Lima, E.F. and Pedersen, J.O.: Phrase Recognition and Expansion for Short, Precision-Biased Queries Based on a Query Log, *Proc of SIGIR'99*, ACM, pp.145–152 (1999).
13) ボッレーガラダヌシカ，　石塚満，グェントアンドゥク：検索方法及びシステム，特許出願 2009-275762, 2009 年 12 月 3 日 (2009).

(平成 22 年 7 月 1 日受付)
(平成 23 年 1 月 17 日採録)



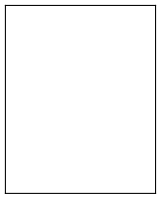
グェントアン ドゥク（学生会員）

2007 年東京大学工学部電子情報工学科卒，2009 年同大学院情報理工学系研究科創造情報学専攻修士課程修了，現在: 同専攻博士課程在学．
ウェブからの情報抽出， ウェブ情報検索， 並列分散プログラミングに興味を持つ．



ボレガラダヌシカ

2005 年 東京大学 工学部電子情報工学科卒，2007 年 同大学院 情報理工学系研究科修士課程修了，2009 年 同研究科博士課程修了 (短縮修了)．博士 (情報理工学)．現在: 同研究科・助教．
複数文書自動要約，Web 上で人物の曖昧性解消，単語間の属性類似性，単語ペア間の関係類似性，Web からの関係抽出などの研究に興味を持つ．WWW, ACL, ECAI などの会議を中心に研究成果を発表．



石塚　　満（正会員）

1971 年 東京大学 工学部卒， 1976 年 同大学院 工学系研究科博士課程修了．工学博士．同年 NTT 入社， 横須賀研究所勤務． 1978 年 東京大学生産技術研究所・助教授． (1980-81 年 Perdue 大学客員准教授)．1992 年 同大学工学部電子情報工学科・教授．現在: 同大学院情報理工学系研究科・教授．
研究分野は人工知能， Web インテリジェンス， 意味計算， 生命的エージェントによるマルチモーダルメディア．IEEE, AAAI， 人工知能学会 (元会長)， 電子情報通信学会等の会員．