情報処理学会論文誌　Vol. 52　No. 12　3349–3362 (Dec. 2011)

大規模分布類似度計算のための ベイズ手法を用いた新しい類似尺度

風間　淳一[†1]　ステイン　デ・サーガ[†1]
黒田　航[†2,†3,†4]
村田　真樹[†5]　鳥澤　健太郎[†1]

これまで提案されている語の意味的類似度尺度は，文脈プロファイルを限られた量のデータから点推定で求めて利用していることから，データスパースネスに対して頑健ではない．本論文は，ベイズ推定の手法を用いた頑健な意味的類似度計算方法を提案する．提案手法は，ベイズ推定により得られた文脈プロファイルの分布の下で元となる類似度の期待値をとることにより類似度を計算する．文脈プロファイルが多項分布で表現され，ベイズ推定における事前分布が Dirichlet 分布であり，元となる類似度が Bhattacharyya 係数である場合，この方法は解析解を持ち，効率的に計算できる．日本語の大規模語彙に対する類似度計算において，提案手法が既存のよく知られた意味的類似度尺度よりも優れていることを実験で示す．

A Bayesian Similarity Measure for Large-scale Calculation of Distributional Similarities

Jun'ichi Kazama,[†1] Stijn De Saeger,[†1]
Kow Kuroda,[†2] Masaki Murata[†5]
and Kentaro Torisawa[†1]

Existing word similarity measures are not robust to data sparseness since they rely only on the point estimation of words' context profiles obtained from a limited amount of data. This paper proposes a Bayesian method for robust distributional word similarities. The method uses a distribution of context profiles obtained by Bayesian estimation and takes the expectation of a base similarity measure under that distribution. When the context profiles are multinomial distributions, the priors are Dirichlet, and the base measure is the Bhattacharyya coefficient, we can derive an analytical form that allows efficient calculation. For the task of word similarity estimation for a large-scale vocabulary in Japanese, we show that the proposed measure gives better accuracies than other well-known similarity measures.

1. はじめに

語の間の意味的な類似度は，理論的にも興味深く，多くの応用が存在することから，計算言語学や自然言語処理分野での長年の研究対象となっている．**表 1** は，約 100 万語の名詞に対して意味的に類似している語 500 個を本研究で提案する手法で計算したものの一部である．このようなデータは，意味的に類似する語での結果が，目的の語にもあてはまるような場合，そうした類似語やその集合に対する結果によって，目的の語に対する結果を補完するという形で利用することができ，その適用範囲は広い．これまで，古くは語義曖昧性解消[6]，最近では，オントロジ拡張[28]，意味関係拡張[26] など多くの自然言語処理で利用され，その有用性が示されている．

意味的類似度に関する多くの研究は，「似た文脈に出現する語は似た意味を持ちやすい」という分布仮説[12] に基づいており，多くの意味的類似尺度がこの仮説に基づき提案されている[7)–10),13),17)]．

一般的に，多くの意味的類似度尺度は次のような形式をしている．

$$sim(w_1, w_2) = g(v(w_1), v(w_2)). \quad (1)$$

ここで，$v(w_i)$ は語 w_i が出現する文脈のパターンを表すベクトルであり，本論文では w_i の文脈プロファイルと呼ぶ．関数 g は，2 つの文脈プロファイルに対する関数であり，我々の直感にあった良い類似度を返すことが期待されるものである．文脈プロファイルベクトルの各次元は，1 つの文脈 f_k に対応し，文脈としては，典型的には，w_i とデータ中で隣接する語や係り受け関係にある語などが利用される．各次元の値 $v_k(w_i)$ は，典型的には，共起

†1 情報通信研究機構
National Institute of Information and Communications Technology
†2 京都工芸繊維大学
Kyoto Institute of Technology
†3 京都大学
Kyoto University
†4 早稲田大学総合研究機構
Comprehensive Research Organization, Waseda University
†5 鳥取大学大学院工学研究科
Graduate School of Engineering, Tottori University

表 1　提案手法による類似度上位の語の例
Table 1　Examples of top similar words by the proposed method.

新橋	参政権	尿酸値	瑠璃光寺	水彩画教室	天法
池袋	選挙権	コレステロール値	南禅寺	絵手紙教室	浦霞
新宿	投票権	血糖値	喜多院	押し花教室	鶴齢
六本木	自治権	眼圧	高台寺	写真教室	麒麟山
吉祥寺	市民権	中性脂肪値	長谷寺	ビーズ教室	上喜元
有楽町	被選挙権	血中濃度	清水寺	着付け教室	吉乃川
渋谷	地方参政権	血糖	室生寺	スケート教室	菊姫
恵比寿	選択権	血圧	浄瑠璃寺	木工教室	奥播磨

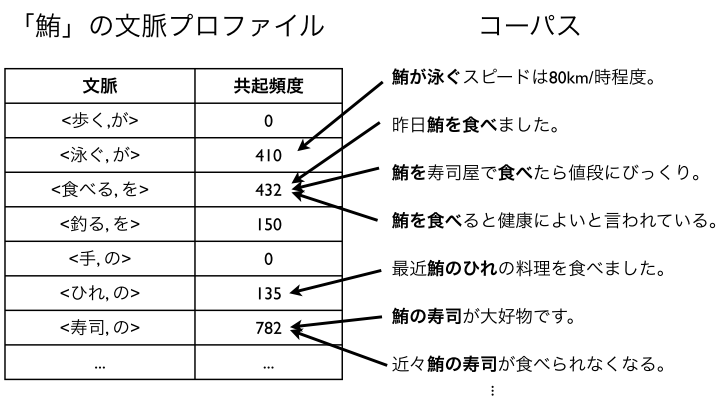


図 1　文脈プロファイルの例
Fig. 1　An example of context profiles.

頻度 $c(w_i, f_k)$，条件付き確率 $p(f_k|w_i)$，あるいは，w_i と f_k の自己相互情報量（Pointwise Mutual Information; PMI）などで，これらはコーパスデータ[⋆1]から計算される．**図 1** に，係り受け関係を文脈とし，共起頻度を値としたときの文脈プロファイルの例を示す．関数 g としては，多くの研究では，コサイン，Jaccard 係数，Jensen-Shannon ダイバージェンスなどが利用されている．

これまでの研究は，主に，いかにして意味的類似度の適切な関数 g を求めるかということに注目して行われてきた．一方，本論文での我々のアプローチは文脈プロファイル（$v(w_i)$）を曖昧さを残した形で頑健に求めておき，それを利用して類似度を頑健に求めるというものである．ここで我々が問題とするのは，$v(w_i)$ は限られた大きさのコーパスから推定されるため，データスパースネスに起因する不確実性を必ず含むということである．信頼できない文脈プロファイルを用いて類似度を計算した場合には，類似度にも相応の不確かさが残るはずである．提案手法を貫く直感は，「すべての他の条件が同じであれば，文脈プロファイルの信頼度がより高い，高頻度の語との類似度がより大きくなるべきである」というものである．たとえば，条件付き確率を文脈プロファイルとする場合，$p(f_k|w_1)$，$p(f_k|w_2)$ が確率分布としてまったく同一だとしても，もし w_1 が w_2 より高頻度であれば，ある他の語 w_0 との類似度は，$sim(w_0, w_1) > sim(w_0, w_2)$ となってほしい．本研究では，そのような性質を持つ類似度を提案する．

自然言語処理分野では，データスパースネスは深刻な問題であると認識され，言語モデル，あるいは，教師あり学習の文脈で，多くの研究が行われてきた．しかし，我々の知る限り，文脈類似度計算においてデータスパースネスを真剣に取り扱った研究はこれまでない．我々の目標は，100 万種類を超える非常に大規模な語彙について類似度を計算することであり，そのような場合には，たとえ大規模なコーパスを用いたとしてもデータスパースネスの問題が顕著となる．データスパースネス問題は通常，スムージング，正則化（regularization），マージン最大化などによって対処することが多い[2)–4)]．また，近年では，ベイズ法を用いた方法が，理論的にも実験的にも有望な手法として注目されている[18),24)]．

本論文では，文脈類似度の計算にこのベイズ推定の手法を取り入れることを試みる．提案手法は，下の式で表されるように，$v(w_i)$ を点推定する代わりに，その分布 $p(v(w_i))$ をベイズ推定し，その分布の下で元々の類似度の期待値を計算するというものである

$$\begin{aligned} sim_b(w_1, w_2) &= E[sim(w_1, w_2)]_{\{p(v(w_1)), p(v(w_2))\}} \\ &= E[g(v(w_1), v(w_2))]_{\{p(v(w_1)), p(v(w_2))\}}. \end{aligned} \tag{2}$$

データスパースネスに起因する不確実性は $p(v(w_i))$ により表現され，期待値計算操作によりそれを考慮した類似度が得られる．ベイズ推定は，通常，低頻度の観測に対しては分散の大きな分布を与え，したがって，期待値計算の結果得られる値は小さくなる．

ベイズ推定および式 (2) に示される期待値計算は，一般的には，計算量の大きな処理を必要とする．我々の本研究における動機は先に述べたとおり，非常に大規模な語彙に対して良い文脈類似度を計算し，それを幅広い自然言語処理タスクに応用することであるので，そのような計算コストは最小に抑えたい．

⋆1 大量の文書を収集したもので，形態素解析や係り受け解析などの必要な言語解析処理などを行った上で用いられる．

本論文の技術的な貢献は，文脈プロファイルが多項分布であり，事前分布がディリクレ分布であり，元の類似度が Bhattacharyya 係数[1] である場合に，式 (2) の計算に対して，解析解を求めることができ，したがって，効率的な計算ができることを示したことである[⋆1]．得られた解析解は，ベイズ手法を取り入れた新しい意味的類似尺度とも考えることができる．

実験では，日本語の大規模な Web コーパスを用いて 100 万語という大規模な語彙に対して意味的類似度を計算し，提案手法が，非ベイズ的な通常の Bhattacharyya 係数や，Jensen-Shannon ダイバージェンス，相互情報量ベクトルのコサインなどのよく知られた類似度尺度よりも優れていることを示す．

本論文は次のような構成となる．2 章では，ベイズ推定と Bhattacharyya 係数について簡単に説明し，3 章で，提案手法であるベイズ的 Bhattacharyya 係数について説明する．4 章では，いくつかの実装上の問題とその解決法を述べる．5 章で実験結果を報告し，最後に 6 章で，提案手法のさらなる拡張，関連研究などの議論を行う．

2. 背　　景

2.1 ディリクレ事前分布を用いたベイズ推定

観測データ D に対する $\boldsymbol{\phi}$ をパラメータとする確率モデル $p(D|\boldsymbol{\phi})$ を推定することを考える．最尤推定の場合，パラメータは $\boldsymbol{\phi}^* = \mathrm{argmax}_{\boldsymbol{\phi}} p(D|\boldsymbol{\phi})$ のように点推定される．たとえば，語 w_i が固定されたときの文脈 f_k の条件付き生成モデル $p(f_k|w_i)$ は，最尤推定では以下のように推定される．

$$p(f_k|w_i) = c(w_i, f_k) / \sum_k c(w_i, f_k). \tag{3}$$

一方，ベイズ推定での目標は，データ D が与えられたときのパラメータ $\boldsymbol{\phi}$ の分布 $p(\boldsymbol{\phi}|D)$ を求め，その後の処理で利用することである．ベイズ則を用いれば，これは，以下のようにとらえることができる．

$$p(\boldsymbol{\phi}|D) = \frac{p(D|\boldsymbol{\phi})p(\boldsymbol{\phi})}{p(D)}. \tag{4}$$

$p(\boldsymbol{\phi})$ は，パラメータ $\boldsymbol{\phi}$ の設定の尤もらしさを事前の知識により与える事前分布である．本論文では，$\boldsymbol{\phi}$ が多項分布である場合，つまり $\sum_k \phi_k = 1$ の場合を考える．これは，K 種類の選択肢から 1 つを選ぶ過程をモデル化したものと考えられる．それぞれの語 w_i の文脈プロファイルとして条件付き確率 $\phi_k = p(f_k|w_i)$ を推定する場合もここに含まれる．本論文では，さらに，事前分布がディリクレ分布 $\mathrm{Dir}(\boldsymbol{\phi}|\boldsymbol{\alpha})$ であると仮定する．ディリクレ分布は，多項分布に対する事前分布で，以下の式で表される．

$$\mathrm{Dir}(\boldsymbol{\phi}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \phi_k^{\alpha_k - 1}. \tag{5}$$

$\Gamma(\cdot)$ はガンマ関数であり，ディリクレ分布は，ハイパーパラメータ $\boldsymbol{\alpha} = \{\alpha_k\}$（$\alpha_k > 0$）でパラメータ化されている．

以上のような条件では，$p(\boldsymbol{\phi}|D)$ 自体もディリクレ分布となることが知られており，以下のように解析的に求めることができる．

$$p(\boldsymbol{\phi}|D) = \mathrm{Dir}(\boldsymbol{\phi}|\{\alpha_k + c(k)\}). \tag{6}$$

ここで，$c(k)$ はデータ D 中での選択肢 k の頻度であり，たとえば，$p(f_k|w_i)$ の推定では，$c(k) = c(w_i, f_k)$ である．つまり，元々の事前分布のハイパーパラメータに観測された頻度を加えるだけの非常に簡単な処理で，事後分布を求めることができる．

2.2 Bhattacharyya 係数による類似尺度

文脈プロファイルが確率分布である場合には，多くの場合，Jensen-Shannon（JS）ダイバージェンスなどの確率分布に対する類似尺度を用いる[6),9)]．JS ダイバージェンスは，以下の式で定義される．

$$JS(p_1||p_2) = \frac{1}{2}(KL(p_1||p_{avg}) + KL(p_2||p_{avg})).$$

ここで，$p_{avg} = \frac{p_1+p_2}{2}$ は，確率分布 p_1，p_2 をベクトルと見たときの平均ベクトルであり，$KL(\cdot)$ は，以下で定義される Kullback-Leibler（KL）ダイバージェンスである．

$$KL(p_1||p_2) = \sum_{k=1}^K p_{1k} \log \frac{p_{1k}}{p_{2k}}. \tag{7}$$

JS ダイバージェンスは，KL ダイバージェンスが持つゼロ確率問題を緩和することができる尺度である[⋆2]．JS ダイバージェンスは，既存研究でも示されているとおり，有効な尺度であるが，ディリクレ事前分布を仮定したとしても，式 (2) に対応する解析解などの効率

⋆1 4 章で述べるいくつかの実装上の工夫を行うことで，他の意味的類似尺度と比べてもそれほど大きくない計算量で計算が可能となる．

⋆2 $KL(p_1||p_2)$ では，ある次元 k に対して，p_{1k} がゼロでなく，p_{2k} がゼロである場合，距離は一律に無限大（類似度 0）となってしまい，類似度の比較に使用できない．

的な計算方法を求めることは難しい[⋆1]．

本研究では，以下の式で定義される Bhattacharyya 係数[1]（以降 BC と呼ぶ）を類似尺度として用いる．

$$BC(p_1, p_2) = \sum_{k=1}^{K} \sqrt{p_{1k} \times p_{2k}}. \tag{8}$$

BC も確率分布に対する類似尺度であり，次章で述べるように我々の目的に適している．また，式から分かるとおり，JS ダイバージェンス同様，ゼロ確率問題はない．この BC は，我々の知る限り語の意味的類似度の研究では用いられてこなかったが，実験で示すように，JS ダイバージェンス同様，良い類似尺度である．

3. 提案手法

元となる類似尺度が上で述べた BC であるとき，ディリクレ分布の下での式 (2) の期待値計算は解析解を持ち，効率的な計算が可能となる．ここで，我々が計算したいのは，2 つのディリクレ分布 $\mathrm{Dir}(p_1|\boldsymbol{\alpha})$，$\mathrm{Dir}(p_2|\boldsymbol{\beta})$ が与えられたときの，以下の値である．

$$\begin{aligned} BC_b(p_1, p_2) &= E[BC(p_1, p_2)]_{\{\mathrm{Dir}(p_1|\boldsymbol{\alpha}), \mathrm{Dir}(p_2|\boldsymbol{\beta})\}} \\ &= \iint_{\triangle \times \triangle} \mathrm{Dir}(p_1|\boldsymbol{\alpha}) \mathrm{Dir}(p_2|\boldsymbol{\beta}) BC(p_1, p_2) dp_1 dp_2. \end{aligned} \tag{9}$$

$\triangle$ は単体上での積分を表す．何段階かの導出を経て（付録を参照），以下の形式の解析解を得ることができる．

$$BC_b(p_1, p_2) = \frac{\Gamma(\alpha_0)\Gamma(\beta_0)}{\Gamma(\alpha_0 + \frac{1}{2})\Gamma(\beta_0 + \frac{1}{2})} \sum_{k=1}^{K} \frac{\Gamma(\alpha_k + \frac{1}{2})\Gamma(\beta_k + \frac{1}{2})}{\Gamma(\alpha_k)\Gamma(\beta_k)}. \tag{10}$$

ここで，$\alpha_0 = \sum_k \alpha_k$，$\beta_0 = \sum_k \beta_k$ である．ディリクレ事前分布を用いる本論文の設定では，式 (6) から，$\mathrm{Dir}(p_1|\{\alpha_k + c(w_1, f_k)\})$ と $\mathrm{Dir}(p_1|\{\beta_k + c(w_2, f_k)\})$ の 2 つのディリクレ分布に関して上記の期待値を計算すればよく，α_k を $\alpha_k + c(w_1, f_k)$ で，β_k を $\beta_k + c(w_2, f_k)$ で読み替えればよい．

最終的に，ディリクレ事前分布のハイパーパラメータ α_k，β_k と，観測された頻度 $c(w_i, f_k)$ だけから計算される以下の形式の新しいベイズ的類似尺度を得る．

$$BC_b(w_1, w_2) = \frac{\Gamma(\alpha_0 + a_0)\Gamma(\beta_0 + b_0)}{\Gamma(\alpha_0 + a_0 + \frac{1}{2})\Gamma(\beta_0 + b_0 + \frac{1}{2})} \sum_{k=1}^{K} \frac{\Gamma(\alpha_k + c(w_1, f_k) + \frac{1}{2})\Gamma(\beta_k + c(w_2, f_k) + \frac{1}{2})}{\Gamma(\alpha_k + c(w_1, f_k))\Gamma(\beta_k + c(w_2, f_k))} \tag{11}$$

ここで，$\alpha_0 = \sum_k \alpha_k$，$\beta_0 = \sum_k \beta_k$ であり，$a_0 = \sum_k c(w_1, f_k)$，$b_0 = \sum_k c(w_2, f_k)$ である．この類似尺度をベイズ的 Bhattacharyya 係数（BC_b）と呼ぶことにする．本論文の実験では，簡単のため，すべての語のすべての文脈に対して同じハイパーパラメータの値 $\alpha_k = \alpha$ を用いることとし，また，$\alpha_k = \beta_k$ として w_1，w_2 に関する対称性を仮定する．

上記のように定義された BC_b が冒頭で述べた我々の直感に従った性質を有していることを，実際の例により見ることができる．語 w_0，w_1，w_2 があるとして，文脈数 $K = 2$ とし，$c(w_0, f_1) = 10$，$c(w_0, f_2) = 20$，$c(w_1, f_1) = 1$，$c(w_1, f_1) = 2$，$c(w_2, f_1) = 100$，$c(w_2, f_2) = 200$ とする．この場合，最尤推定によれば，この 3 語の文脈プロファイルは $(1/3, 2/3)$ という同一のものとなる．このとき，$\alpha = 1.0$ とおくと，これらの語の間の BC_b は以下のように計算される．

$BC_b(w_0, w_1) = 0.970663$

$BC_b(w_0, w_2) = 0.995669$

期待したとおり，$BC_b(w_0, w_1) < BC_b(w_0, w_2)$ となり，w_0 を固定した場合に，観測頻度が大きい語との間の類似度のほうが大きくなっていることが分かる．非ベイズ的類似尺度では，これらはすべて 1 となる．また，通常の類似尺度とは異なり，$BC_b(w_0, w_0) = 0.992234$ となり，同じ語の間の類似度が最大値 1 をとらないことも分かる．これは，概念的には多少奇妙に思えるが，実際には，$sim(w_i, w_i)$ の値を利用することはそれほど多くないため，深刻な問題ではない．しかし，この状態を回避したい場合には，$BC_b(w_i, w_i) \equiv 1$ という定義を用いることもできる．これは，同じ語の間の類似度を $sim_b(w_i, w_i) = E[sim(w_i, w_i)]_{\{p(v(w_i))\}} = 1$ で定義しているととらえることもできる．

4. 実装上の課題

前章では，解析解（式 (11)）の導出について述べたが，これを実際に問題なく効率的に計算するためにはいくつかの問題を解決しなければならない．

まず，式 (11) 中のガンマ関数は引数が 170 を超えたあたりでオーバフローを起こす．そ

⋆1 すぐに思いつく方法としては，$p(v(w_i))$ から $v(w_i)$ をサンプリングして期待値を近似計算することであるが，解析解による直接的な計算に比べると非効率である．

のような場合によく使われる手法は，log 関数の空間で計算を行うことである．本研究では，$ln\Gamma(x)$ をオーバフローの問題なしに直接返す「log ガンマ関数」を利用する．本研究では，この関数を実装している GNU Scientific Library（GSL）（www.gnu.org/software/gsl/）を利用した．

次に，log ガンマ関数の計算は，既存の類似尺度で使用されるような単純な積操作よりも，重い．実際，GSL では，log ガンマ関数は反復法である Lanczos 法を使用して実装されている．加えて，式 (11) によれば，たとえ $c(w_i, f_k)$ がゼロでも和の中の項の値はゼロにはならないため，一見するとすべての k に対しての和をとらなければならないように見える．既存の類似尺度では，$c(w_i, f_k) > 0$ であるような k に対してだけ和をとることで計算ができることが多く，実際，$c(w_i, f_k)$ の多くはゼロ（スパース）であるため，この技法は既存の類似尺度の計算を大幅に高速化し実用にたえるものとしている．

本研究では，我々の目的は固定した大規模な語のセット中の語の間の類似度を計算することであると仮定し，その場合に必要となる log ガンマ関数の値と，$c(w_i, f_k) = 0$ の場合のデフォルトの値をあらかじめ最初に計算しておくことで，この問題を解決する．以下の値が前もって 1 度だけ計算される[⋆1,⋆2]．

すべての語 w_i に対して：

(A) $A[w_i] = ln\Gamma(\alpha_0 + a_0) - ln\Gamma(\alpha_0 + a_0 + \frac{1}{2})$

(B) $B[w_i][k] = ln\Gamma(\alpha_k + c(w_i, f_k) + \frac{1}{2}) - ln\Gamma(\alpha_k + c(w_i, f_k))$（$c(w_i, f_k) > 0$ であるようなすべての k に対して）

(C) $C[w_i][k] = -\exp(2(ln\Gamma(\alpha_k + \frac{1}{2}) - ln\Gamma(\alpha_k)))) + \exp(ln\Gamma(\alpha_k + c(w_i, f_k) + \frac{1}{2}) - ln\Gamma(\alpha_k + c(w_i, f_k)) + ln\Gamma(\alpha_k + \frac{1}{2}) - ln\Gamma(\alpha_k))$（$c(w_i, f_k) > 0$ であるようなすべての k に対して）

すべての k に対して：

(D) $D[k] = \exp(2(ln\Gamma(\alpha_k + \frac{1}{2}) - ln\Gamma(\alpha_k)))$.

$BC_b(w_1, w_2)$ を計算する際には，まず $c(w_i, f_k)$ がすべてゼロと仮定した場合の和の部分の値（$\sum_k D[k]$）を計算し，出力変数 V に入れる．その後，$c(w_1, f_k)$ と $c(w_2, f_k)$ を要素とするスパースなベクトルのゼロでない次元を走査し[⋆3]，もし，$c(w_1, f_k) > 0$ かつ $c(w_2, f_k) = 0$（あるいはその逆）の場合には，出力変数にゼロでない方の (C) の値を V に足す．

もし，$c(w_1, f_k) > 0$ かつ $c(w_2, f_k) > 0$ である場合には，$V = V - D[k] + \exp(B[w_1][k] + B[w_2][k])$ のようにして V の値を更新する．最後に，(A) を利用して，$\exp(A[w_1] + A[w_2] + ln(V))$ を計算して最終結果を得ることができる．このような実装により，BC_b を（固定した大規模な語のセット中の語の間の類似度を計算する場合には）既存の類似尺度と同様に実用的な時間で計算することができるようになる．

また，提案手法と比較手法に共通して，すべての語（本論文の場合 100 万語）の間の類似度を計算することは計算量が大きいため，$c(w_i, f_k)$ のスパースネスを利用した近似計算を行う．似たような方法は，類似度計算の既存研究[14),20),21)] でも利用されている．語 w_i に対する類似度を計算するときには，まず，$c(w_i, f_k)$ の降順でソートされた文脈 k の上位 L 個を取り出す[⋆4]．取り出された文脈それぞれについて，$c(w_i, f_k)$ の降順でソートされた語の上位 M 個を取り出す．ソート自体は，類似度を計算する前に 1 度だけ行っておけばよい．上記のそれぞれの文脈に対して取り出された語の和集合をとって類似度計算の候補集合とし，この候補語との間の類似度だけを計算する．最終的には，類似度が大きい上位 N 個を出力とする．

これらを組み合わせることで，16CPU コアを用いて並列に計算し，100 万語すべてに対して $L = M = 1{,}600$，$N = 500$ として類似度上位の 500 語を計算した場合，Jensen-Shannon ダイバージェンスの場合には 57 時間，提案手法の BC_b では，およそ 100 時間で計算が終わった．この程度の追加時間は，提案手法の概念的な複雑さを考えれば，許容できるものであると考える．なお，必要メモリサイズは Jensen-Shannon ダイバージェンスの場合には，約 26 GB，BC_b の場合には，約 34 GB であった．

5. 実　　験

5.1 実験設定

本章では，提案手法を日本語の名詞間の類似度計算において評価する．

語の間の意味的類似度の人手による評価は定義も難しくコストも大きいため，本研究で

⋆1 各語が式 (11) の第 1 番目の語 w_1 とした場合の記号を用いるが，計算時には適宜 w_2 の場合に置き換えて使用する．

⋆2 また，(C)，(D) の計算においては，exp の中の差分を計算してから exp をとるという計算の順番が，オーバフローを起こさずに安定して計算するための鍵となる．

⋆3 各ベクトルは，ゼロでない次元についての次元番号と値の組のリストで保持するとし，リスト中の組は次元番号の昇順でソートされているものとすると，このベクトル走査は，2 つのベクトルのゼロでない要素数の和のオーダの計算量で行うことができる．

⋆4 取り出される文脈の数が L より小さい場合には，頻度がゼロでない文脈すべてが用いられる．

は，既存研究[20] で行われている「集合拡張」の設定での自動評価を行う．語の意味的な集合が与えられたとして，良い類似尺度は，集合中の各語に対して集合中の他の語を類似語として出力することが期待される．語の集合が与えられたときに，入力が各集合中の語であり出力がその集合の他の語であるような入力–正解集合のペアを作成することができる．たとえば，集合 { 犬, 猫, 狸 } からは，犬 → { 猫, 狸 }，猫 → { 犬, 狸 }，狸 → { 犬, 猫 } という入力–正解集合ペアが得られる．各入力語に対して最も類似した 500 語を各類似尺度で計算し[⋆1]，それが正解集合とどれだけ一致しているかを測る．この設定は，文書検索と同じ設定となっており，上位 T での適合率の平均（MP@T）や，平均適合率の平均（Mean Average Precision; MAP）などのよく利用される性能尺度を用いることができる．それぞれの入力語に対して，P@T（上位 T での適合率）と AP（平均適合率）は以下のように定義される．

$$\mathrm{P@}T = \frac{1}{T}\sum_{i=1}^{T}\delta(w_i \in ans),$$

$$\mathrm{AP} = \frac{1}{R}\sum_{i=1}^{N}\delta(w_i \in ans)\mathrm{P@}i.$$

関数 $\delta(w_i \in ans)$ は，w_i が正解集合に含まれている場合に 1 を返し，それ以外の場合には 0 を返す関数である．N は出力の数であり，R は正解集合中の語数である．MP@T と MAP は，それぞれ，すべての入力語でのこれらの値の平均である．評価に用いた語の意味的集合については，5.3 節で具体的に説明する．

5.2　文脈プロファイルの抽出

本研究では，文脈として，係り受け関係を用いた[14),15)]．情報源としては，日本語の大規模 Web 文書コーパス[23)]（約 1 億ページ）を用いた．このコーパスでは，各々の文に係り受け解析結果が付与されている．そこから，「名詞–動詞」「名詞–名詞」の係り受けを，係り受けの関係のタイプとともに抽出し，その頻度を求めた．名詞 n が語 w（名詞あるいは動詞）に関係 r で係るとき，$(n, \langle w, r\rangle)$ という係り受けを表す組が抽出される．そして，$\langle w, r\rangle$ が n に対するある文脈 f_k に対応することとなる．

「名詞–動詞」の係り受けの場合には，日本語の助詞が係り受けの関係タイプを表すものとする．たとえば，「ワインを買う」という文からは，(ワイン, ⟨ 買う, を ⟩) という係り受け関係が取り出される．ここで，「れる」などの助動詞は，n のタイプに大きく影響するため，w の一部として残す．たとえば，「ワインが醸造される」からは (ワイン, ⟨ 醸造される, が ⟩) が取り出される．

「名詞–名詞」の係り受けに関しては，「n_1 の n_2」という関係のみを考え，ここから，$(n_1, \langle n_2, \text{の}\rangle)$ を取り出す．

最終的には，頻度も付与された $(n, \langle w, r\rangle, c)$ という組の集合が得られる．上記の Web コーパスから，約 4.7 億種類の係り受けが抽出され，そこには約 3,100 万種類の名詞（複合名詞も含む）と，約 2,200 万種類の文脈 f_k が含まれていた．我々は，名詞をそれと共起する文脈の種類数の降順でソートし，上位の 100 万語を選択した．文脈に関しては共起する名詞の種類数で降順にソートし，上位の 10 万個を選択した．最後に，抽出された係り受けのうち，上記の名詞 100 万語，文脈 10 万個のみから構成される係り受けだけを取り出した約 3.7 億組を学習データとして利用した．

5.3　評価セット

本研究では，以下の 3 つの評価セットを用意して用いた．

セット A，B：シソーラス兄弟語　ここでは，人手で作成されたシソーラス中で共通の上位語を持つ語（兄弟語）は類似語集合となりうると考え，本研究では，概念間の階層関係と語から概念へのマッピングが定義されている日本語辞書 EDR（V3.0）[5)] を用いた．この辞書には，304,884 の名詞が含まれ，平均サイズ 45.96 語の 6,703 個の兄弟語集合が得られた．200 集合をそれぞれランダムで選択し，セット A，セット B とした．ここで，セット A とセット B には重なりがないように選択をした．セット A は，事前分布のハイパーパラメータを調整するための開発セットとして用いる．セット B は，ハイパーパラメータ調整の有効性を確認するための最終評価セットとして用いる．

セット C：閉語集合　ここでは，Murata ら[19)] が作成した，閉語集合のデータを用いる．これは，国名，河川名，力士名など，ある時点で切り取れば数え上げられるような意味クラスの語をすべて列挙したようなデータである．このデータのうち，網羅度が「完全」となっている集合のみを用い，12,827 語を含む 45 集合を得た．

本研究では，語の曖昧性についてはこれらの評価セットの作成の際にも，また，類似度の計算の際にも扱わない．つまり，ある語が複数の集合に含まれる場合にはそれら複数の集合から得られる出力正解の和集合を正解出力とする[⋆2]．

⋆1 自分自身は取り除く．また，4 章で述べた類似計算のため 500 語出力されない場合もある．

⋆2 たとえば，集合 1 $\{a, b, c\}$，集合 2 $\{a, d, e\}$ を考えた場合，語 a は，集合 1 にも集合 2 にも含まれる．この場合（a の意味に曖昧性がある場合），a に対する正解出力は，集合 1 から得られる正解出力 $\{b, c\}$ と，集合 2 から得られる正解出力 $\{d, e\}$ の和集合である $\{b, c, d, e\}$ となる．

また，これらの評価セットに含まれる語彙は我々の 100 万語の語彙とは異なる．したがって，我々の 100 万語に含まれない語は削除することとし，入力そのものが含まれない場合や削除の結果正解集合のサイズが 1 語以下になった入力–正解集合ペアは取り除く．逆に，100 万語に含まれるが評価セットには含まれない語は，類似度計算の結果の評価に用いる上位 500 語の中に含まれる可能性があるが，そのままとすることにした．したがって，計測される性能は語彙を評価セットとそろえた場合に比べて低めの値となるが，手法の比較に用いることはできる．

最終的には，セット A では，平均で 115 語の正解集合を持つ 3,740 個の入力–正解集合ペアが，セット B では，平均で 65 語の正解集合を持つ 3,675 個の入力–正解集合ペアが得られた．セット C では，平均で約 1,700 語の正解集合を持つ 8,853 個の入力–正解集合ペアが得られた．

5.4 比較手法

実験では，提案するベイズ的 Bhattacharyya 係数 BC_b を，以下の既存の類似尺度と比較した．

JS　$p(f_k|w_1)$ と $p(f_k|w_2)$ の間の Jensen-Shannon ダイバージェンスである[7),9)]．

PMI-cos　$PMI(w_i, f_k) = \log \frac{p(w_i, f_k)}{p(w_i)p(f_k)}$ で定義される w_i と f_k の間の自己相互情報量（PMI）を k 番目の次元とするベクトルの間のコサインである[20),21),⋆1]．

Cls-JS　Hagiwara ら[11)]，Kazama ら[14)] は，$p(w_i, f_k) = \sum_c p(w_i|c)p(f_k|c)p(c)$ という隠れクラスを持つ係り受け生成モデルに対する EM 学習（隠れクラスによるソフトクラスタリング）の結果として得られる隠れクラスの分布 $p(c|w_1)$，$p(c|w_2)$ の間の Jensen-Shannon ダイバージェンスで語の類似度を計算することを提案している．その際，EM 学習における局所最適の問題を軽減するために，複数の異なる初期値を用いた EM 学習の結果得られる類似度を平均することを行っており，効果が示されている．本研究では，s1，s2 と記される 2 つの学習結果を組み合わせて用いた．これを実験では s1+s2 と記す．Kazama ら[14)] の研究に従って，s1，s2 はそれぞれ隠れクラスの数を 2,000，文脈の種類数は 100 万とし，さらに，100 万語彙という大規模な EM 学習を可能にするため並列化された EM アルゴリズムを用いて学習した．この手法は，クラスタリングというデータスパースネスに対処する異なるアプローチであり，比較方法の 1 つとした．

Cls-BC　上記手法において Jensen-Shannon ダイバージェンスの代わりに Bhattacharyya 係数を用いた手法である．

BC　$p(f_k|w_1)$ と $p(f_k|w_2)$ の間の Bhattacharyya 係数[1)] であり，BC_b に対するベースラインとなる．

BC_a　絶対ディスカウンティングにより求めた $p(f_k|w_1)$ と $p(f_k|w_2)$ の間の Bhattacharyya 係数である．絶対ディスカウンティングでは，$p(f_k|w_i)$ を求める際に $c(w_i, f_k)$ から定数 α（$0 \leq \alpha \leq 1$）を一律に引き，引いた分の確率値を頻度がゼロである文脈に均等に割り振る．提案手法は，式 (11) のとおり，観測された頻度に値を足すという一種のスムージングを行っていると考えられる．BC_a はこのスムージングのなかでも特に単純な方法として，比較手法として取り上げた．

また，本研究では，観測された頻度 $c(w_i, f_k)$ ではなく $\log(c(w_i, f_k)) + 1$ のように補正した頻度を用いる．このような頻度の補正は既存研究でも用いられている[14),25)]．本研究では，Web コーパスでみられた異常に高い頻度の係り受けの影響を取り除くためにこの補正を行った．予備的な実験で，補正した頻度を用いることにより，既存研究[14),25)] で報告されているように上位 500 語の品質が改善されることが分かった．

提案手法 BC_b や比較手法 BC_a のハイパーパラメータ α は開発セットを利用するなどして最適な値に調整する必要がある[⋆2]．

5.5 結　　果

表 2 は，セット A に対する結果である．提案手法と比較手法に対して MAP および上位 1，5，10，20 に対する MP を載せている．BC_b と BC_a に関しては，いくつかのハイパーパラメータの値と最適な値に対しての結果を載せている．ハイパーパラメータの最適化は，**図 2** のように行った．X 軸が log スケールでの α の値であり，Y 軸が MAP の値となっている．図 2 には，対応する MP の値の様子を示したグラフを α の範囲が揃うようにして下に載せてある．MAP と各 MP はおおむね相関しており，MAP に対して α を最適化することで MP に対してもおおよその最適な値が求まることが分かる．これらの結果から，最適なハイパーパラメータ $\alpha = 0.0016$ では，BC_b が確かにベースラインの BC より性能が良く，MAP では 6.6%の改善，MP@1 では，14.7%の改善となっていることが分かる．絶対ディスカウンティング BC_a も，最適なハイパーパラメータの値で BC より性能が良いが，改善幅は BC_b より小さいことが分かる．また，表 2 からは Cls-JS，Cls-BC は，複数のク

⋆1 ただし，Pantel ら[21)] が述べている PMI の値のディスカウンティングは使用していない．

⋆2 前述したように，提案手法のハイパーパラメータはベクトルで，語ごとに決まる $\{\alpha_k\}$ であるが，この実験では，$\alpha_k = \alpha$ とし，さらに，それがすべての語で共通であるとした．

表 2　セット A に対する性能
Table 2　Performance on siblings (Set A).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.0299	0.197	0.122	0.0990	0.0792
PMI-cos	0.0332	0.195	0.124	0.0993	0.0798
Cls-JS (s1)	0.0306	0.191	0.118	0.0944	0.0756
Cls-JS (s2)	0.0285	0.194	0.117	0.0941	0.0752
Cls-JS (s1+s2)	0.0333	0.206	0.129	0.103	0.0841
Cls-BC (s1)	0.0319	0.195	0.122	0.0988	0.0796
Cls-BC (s2)	0.0295	0.198	0.122	0.0981	0.0786
Cls-BC (s1+s2)	0.0344	0.214	0.134	0.107	0.0867
BC	0.0334	0.211	0.131	0.106	0.0854
BC_b (0.0002)	0.0345	0.223	0.138	0.109	0.0873
BC_b (0.0016)	**0.0356**	**0.242**	**0.148**	**0.119**	**0.0955**
BC_b (0.0032)	0.0325	0.223	0.137	0.111	0.0895
BC_a (0.0016)	0.0337	0.212	0.133	0.107	0.0863
BC_a (0.0362)	0.0345	0.221	0.136	0.110	0.0890
BC_a (0.1)	0.0324	0.214	0.128	0.101	0.0825
without $\log(c(w_i, f_k)) + 1$ modification					
JS	0.0294	0.197	0.116	0.0912	0.0712
PMI-cos	0.0342	0.197	0.125	0.0987	0.0793
BC	0.0296	0.201	0.118	0.0915	0.0721

ラスタリング結果を組み合わせることにより精度が向上すること，BC や Cls-BC が，既存手法の JS，PMI-cos，Cls-JS などと比べても性能が良く，Bhattacharyya 係数がベースラインの類似尺度として劣っているということがないことが分かる．表 2 には，前述した頻度の補正を行っていない場合の精度をいくつか参考のため載せた．PMI-cos ではその効果が微妙であるが，JS，BC では頻度の補正に効果があることが分かる．

ハイパーパラメータの最適化には，過学習の危険性がある[⋆1]ため，その安定性を検証する必要がある．ここでは，上でセット A に対して求めた最適な α の値が，セット B に対してもうまく働くかを調べた．結果は，**表 3** である．ここから，セット A に対する最適な値 α (= 0.0016) が，セット B に対してもうまく働くことが分かる．つまり，大規模な語彙のうちのごく小さな部分集合を用いるだけで，α をうまく最適化できることを示しており，提案手法は，全体として，現実的な手法となっている．

⋆1 この場合，セット A に対してのみ高い精度を示す値になっている可能性がある．

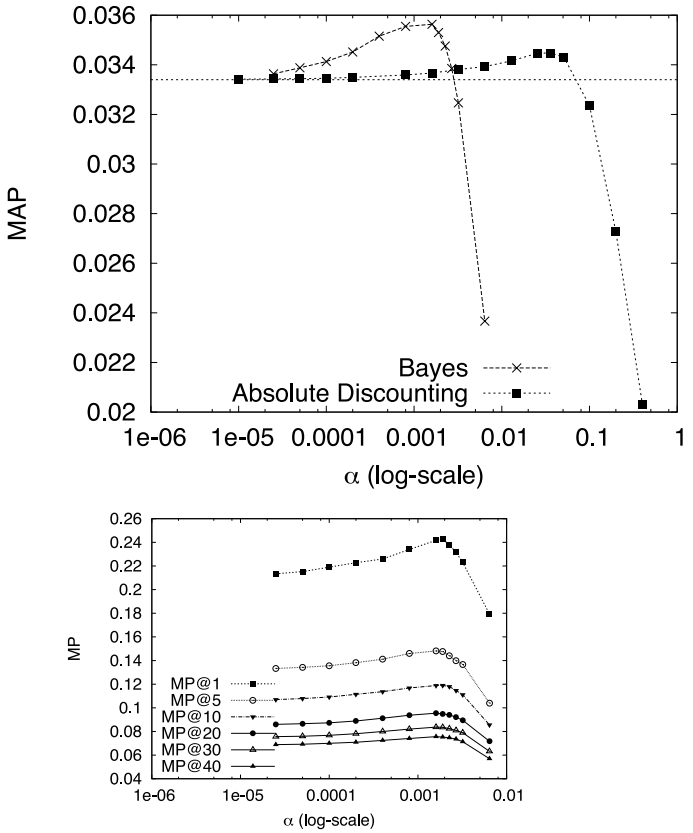


図 2　（上）MAP を目的とした α の最適化．水平の波線は BC の値を示す．「Bayes」が BC_b，「Absolute Discounting」が BC_a に対応する．（下）対応する MP の値
Fig. 2　(Upper) Tuning of α for MAP. The dashed horizontal line indicates the score of BC. "Bayes" corresponds to BC_b and "Absolute Discounting" to BC_a. (Bottom) Corresponding MPs.

次に，セット C に対する性能評価を行った．結果は，**表 4** である．このセットに対しては，セット A，セット B とは違った傾向が見られた．Cls-JS，Cls-BC が特に良い性能を示しており，BC_b は確かに BC から改善されている（たとえば，MP@1 で，7.5%の改善）が，MAP での差は見られなかった．また，MAP は MP とよく相関していないように見える．1 つの理由として，出力の数が 500 という設定が，平均正解数が約 1,700 というこの

表 3　セット B に対する性能
Table 3　Performance on siblings (Set B).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.0265	0.208	0.116	0.0855	0.0627
PMI-cos	0.0283	0.203	0.116	0.0871	0.0660
Cls-JS (s1+s2)	0.0274	0.194	0.115	0.0859	0.0643
Cls-BC (s1+s2)	0.0283	0.201	0.116	0.0879	0.0661
BC	0.0295	0.223	0.124	0.0922	0.0693
BC_b (0.0002)	0.0301	0.225	0.128	0.0958	0.0718
BC_b (0.0016)	**0.0313**	**0.246**	**0.135**	**0.103**	**0.0758**
BC_b (0.0032)	0.0279	0.228	0.127	0.0938	0.0698
BC_a (0.0016)	0.0297	0.223	0.125	0.0934	0.0700
BC_a (0.0362)	0.0298	0.223	0.125	0.0934	0.0705
BC_a (0.01)	0.0300	0.224	0.126	0.0949	0.0710

セットに対しては小さすぎ MAP の値がうまく調べられていないということが考えられた．計算コストと出力のファイル容量を余計に使ってよいならば，500 以上の語を出力することも可能である．そこで，表 4 には，$L = M = 3{,}600$，$N = 2{,}000$ の場合の結果を調べた結果も載せた．しかし，この設定でも MAP と MP はよく相関していない．

セット C に対しては Cls-JS，Cls-BC が特に良い性能を示しているが，前準備として必要な EM による係り受けのクラスタリングは，非常に計算量とメモリ量がかかる[15]．たとえば，100 万語，2,000 クラスでの学習は，我々が保有する 24CPU コアの計算機で並列で行った場合でも，約 8 日かかった．また，そのときの必要メモリサイズは，約 130 GB であった．さらに，複数の結果を組み合わせるとすると，その分だけ余計に計算量がかかる．一方，提案手法では，前準備は 1CPU コアで約 1 時間で終わる．

5.6 分　　析

提案手法により性能の改善が見られたわけであるが，これは，他の自然言語処理のタスクと同じように単に「平均における改善」であることに注意しなければならない．したがって，たとえば個々の語に対する出力類似語を見るだけでは，明らかな質的な差を観察することは難しい．量的な 1 つの分析として，**表 5** に BC_b を BC と比較したときに，MP@20 が上昇した語，変化のない語，低下した語の数を示した．ここから分かるように，MP@20 が低下した語も少なからず存在するが，その数は MP@20 が上昇した語の数より少ない．

前述したように，名詞は，共起する係り受けの種類の数の降順でソートされ，上位の 100 万語が選択されている．係り受けの種類数は，おおむね語の頻度と相関するため，順位が高い（ソートの上位にある）語は頻度が大きい語ということもできる．

図 3 は，このソートの順位に従って名詞を 4 万語ごとに分割して，各評価セットでの BC_b と BC の MP@20 の差を各々の領域で平均したものを表示したものである．ここから，BC_b の優位性は高順位の領域では小さくなり，順位が 40,000 以下の領域では，どの評価セット

表 4　セット C に対する性能
Table 4　Performance on closed-sets (Set C).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.127	0.607	0.582	0.566	0.544
PMI-cos	0.124	0.531	0.519	0.508	0.493
Cls-JS (s1)	0.125	0.591	0.565	0.547	0.524
Cls-JS (s2)	0.135	0.607	0.590	0.576	0.553
Cls-JS (s1+s2)	**0.152**	0.638	**0.618**	0.603	0.583
Cls-BC (s1)	0.125	0.589	0.566	0.548	0.525
Cls-BC (s2)	0.137	0.608	0.592	0.576	0.554
Cls-BC (s1+s2)	0.152	0.636	0.617	**0.604**	**0.585**
BC	0.131	0.602	0.579	0.565	0.545
BC_b (0.0004)	0.133	0.636	0.605	0.587	0.563
BC_b (0.0008)	0.131	**0.647**	0.615	0.594	0.568
BC_b (0.0016)	0.126	0.644	0.615	0.593	0.564
BC_b (0.0032)	0.107	0.573	0.556	0.529	0.496
$L = M = 3{,}200$ and $N = 2{,}000$					
JS	0.165	0.605	0.580	0.564	0.543
PMI-cos	0.165	0.530	0.517	0.507	0.492
Cls-JS (s1+s2)	0.209	0.639	0.618	0.603	0.584
BC	0.168	0.600	0.577	0.562	0.542
BC_b (0.0004)	0.170	0.635	0.604	0.586	0.562
BC_b (0.0008)	0.168	0.647	0.615	0.594	0.568
BC_b (0.0016)	0.161	0.644	0.615	0.593	0.564
BC_b (0.0032)	0.140	0.573	0.556	0.529	0.496

表 5　それぞれの評価セットにおいて，MP@20 が上昇した語，変化のない語，低下した語の数
Table 5　The numbers of improved, unchanged, and degraded words in terms of MP@20 for each evaluation set.

	# improved	# unchanged	# degraded
Set A	755	2,585	400
Set B	643	2,610	404
Set C	3,153	3,962	1,738

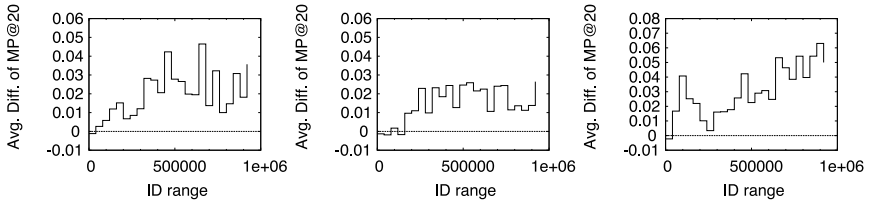


図 3　BC_b (0.0016) と BC の MP@20 の差を 40,000 語ごとに分割した領域で平均したもの（左：セット A. 中：セット B. 右：セット C）．ここでは，順位を最上位を 0 とした整数値（ID）で表している
Fig. 3　Averaged Differences of MP@20 between BC_b (0.0016) and BC within each 40,000-word rank range (Left: Set A. Middle: Set B. Right: Set C).

表 6　語の順位に関する統計．(A)：正解の平均順位．(B)：システム出力の平均順位．(C)：正しいシステム出力の平均順位．ここでは，順位を最上位を 0 とした整数値で表して平均をとっている
Table 6　Statistics on word ranks. (A): Avg. word ranks of answers. (B): Avg. word ranks of system outputs. (C): Avg. word ranks of correct system outputs.

	Set A			Set C		
	(A) 238,483			(A) 255,248		
	(B)	(C)	(B)/(C)	(B)	(C)	(B)/(C)
Cls-JS (s1+s2)	282,098	176,706	1.60	273,768	232,796	1.18
JS	183,054	113,442	1.61	211,671	201,214	1.05
BC	162,758	98,433	1.65	193,508	189,345	1.02
BC_b(0.0016)	55,915	54,786	1.02	90,472	127,877	0.707

でも BC の方が平均的に性能が良いことが分かる．これは，提案手法がスムージングの一種であると考えれば，高順位領域（高頻度語）ではその必要性が小さくなるということであるから，期待された傾向である．また，この結果は，異なる順位の領域に対しては異なるハイパーパラメータ（高順位領域に対しては，より小さな値）を用いる必要があることを示唆しており，今後の課題としたい．

次に，我々の扱っている語彙数は 100 万と評価セットの語彙数よりも大きいため，評価セット中の正解語集合は単に高順位の語をより含む傾向にあるだけではないかという懸念がある．提案手法が高順位の語を平均的により多く出力するだけならば，ここまでの実験結果はそれほど興味深いものではなくなる．この点を分析するため，**表 6** に順位に関するいくつかの統計を示した．ここでは，順位を最上位を 0 とした整数値で表して平均をとっているため，数値が小さいほど高順位であることを表す．確かに，提案手法 BC_b はベースライン BC よりも高順位語を類似語として出力する傾向にあることが分かる．しかし，これは手法の動機からすれば期待された傾向である．また，BC は JS よりも，JS は，Cls-JS よりも高順位語を出力する傾向にあるといえる．Cls-JS の出力の平均順位（表 6 中 (B)）は正解の平均順位（表 6 中 (A)）とほぼ同じであり，順位空間でよく分散された出力をしていることが分かる．ここで，(B) と出力のうち正解であった語の平均順位（表 6 中 (C)）の比（表 6 中 (B)/(C)）を調べてみると，Cls-JS，JS，BC はほぼ同じ値であるのに対し，BC_b はより小さい値となっている．これは，BC_b には出力のうちの低順位語に関する正解率が高いという他の手法には見られない顕著な傾向があるということであり，得られた改善は興味深いものであるといえる．しかし，提案手法の性能改善が実際どのようになされているのか，それが，それぞれの応用場面でどのような影響を持つかについては，より詳細な分析が必要である．

表 7　近似計算における L，M の影響
Table 7　Effect of L and M in approximation.

$L(=M)$	400	800	1,600	3,200	6,400	全候補
1 語あたりの平均候補数	34,523.3	86,892.2	183,847	316,624	447,004	931,843
全候補に対する比率	0.0370	0.0932	0.197	0.340	0.480	1.0
近似の良さ	0.836	0.915	0.960	0.984	0.995	1.0
性能（MAP）	0.0350	0.03559	0.0356	0.0356	0.0356	0.0356

4 章では，計算コストを抑えるため，L，M のパラメータを用いて，類似度を計算する候補の数を減らすという近似手法を用いていることを述べた．本論文では，主に $L = M = 1{,}600$ という設定を用いた．この設定は，予備的実験から経験的に決めたものであるが，これによる近似がすべての候補を対象として計算した場合をどの程度よく近似しているかは興味のあるところである．そこで，L（$= M$）を変化させて，計算量，近似の良さ，類似度の性能をセット A に対する BC_b の計算において調べた．

結果が，**表 7** である．ここで，すべての候補としては，共起する文脈を共有しない語の類似度はそもそも計算する必要はないと仮定し，語 w_i と共起する文脈 f_k と共起するような語すべてを考える．これを「全候補」と呼ぶ．計算量は，L，M による絞り込みによって，候補数が全候補の数と比べてどのくらいの比率にまで減少するかで測る．近似の良さは，絞り込み候補を用いた計算の上位 500 位までの語が，全候補を用いて計算した場合の上位 500 位までの語のどれかと一致する割合で測る．

この結果を見ると，$L = M = 1{,}600$ という設定は，全候補の場合の約 20%の計算コストで，高い近似の良さと全候補の場合と同等の性能を示しており，バランスのとれた設定であったことが分かる．

6. 議　　論

6.1 スムージング手法としての解釈

実験より，提案手法のベイズ的類似尺度は，ベースラインの Bhattacharyya 係数や他の既存類似尺度より性能が高いことが分かった．スムージング法としては，単純な絶対ディスカウンティングよりも優れていることが示された．現時点では，他のより洗練されたスムージング法に比べて優れているということはできないが，前述したように，語の意味的類似度計算においてスムージングの効果を詳しく調査した研究はこれまでない．最近の研究では，ベイズ手法の枠組みが，Kneser-Ney スムージング[16] などの洗練されたスムージング法を特殊な場合として含むことが分かっている[18),24)]．したがって，ベイズ手法を類似度計算に取り入れることは，自然な流れといえる．形式上は，我々の手法は，$p(f_k|w_i)$ を，最尤推定の $p(f_k|w_i) = \frac{c(w_i,f_k)}{a_0}$ ではなく，$p(f_k|w_i) = \left\{\frac{\Gamma(\alpha_0+a_0)\Gamma(\alpha_k+c(w_i,f_k)+\frac{1}{2})}{\Gamma(\alpha_0+a_0+\frac{1}{2})\Gamma(\alpha_k+c(w_i,f_k))}\right\}^2$ のように一種のスムージングをしたうえで，Bhattacharyya 係数をとっていることと同等である．しかし，この解釈の意味するところは，まだ明らかではない．

6.2 ベイズ法としての特徴

我々の手法は，ベイズ法としては，最も単純な部類に入る．より徹底したベイズ手法[18),24)]で行われているような，数値計算やサンプリングなどは行っていない．代わりに，本研究では，得られた解析解を直接用い，さらに，$\alpha_k = \alpha$ という単純化をして，語彙の小さな部分集合を開発セットとして単純なグリッドサーチを行い，この最適な値を求めた．しかし，さらに計算量をかけることが許される場合には，α_k 自体もベイズ手法を用いて調整し，語ごとに異なる最適値を求めることも考えられる．これについては，今後の研究課題としたい．

6.3 提案手法の欠点と今後の拡張

提案手法の欠点は，Bhattacharyya 係数以外の類似尺度を元としたときに効率的な計算方法を簡単には求められない点である．たとえば，Jensen-Shannon ダイバージェンスを元の尺度とした場合に，同じように解析解を求めることは難しそうに見える．我々は，この欠点を補うため以下のような方法で提案手法に柔軟性を持たせることができると考えている．1 つめは，ハイパーパラメータの α_k を PMI などの文脈の重要性といったような外部知識に従って設定するというものである．2 つめは，$\sum_k \mu(w_1,f_k)\mu(w_2,f_k)\sqrt{p_{1k}\times p_{2k}}$ で定義されるような「重み付き」Bhattacharyya 係数を元の尺度として用いるというもので，$\mu(w_i,f_k)$ は p_{ik} には依存しない重みである．解析解は，和の中で重み項 $\mu(w_1,f_k)\times\mu(w_2,f_k)$ が単純に掛けられる形の BC_b になる．さらに，BC_b は，元の尺度が，$BC^d(p_1,p_2) = \sum_{k=1}^K p_{1k}^d \times p_{2k}^d$ $(d>0)$ で定義される場合にも簡単に拡張できる．この場合，解析解は以下のようになる．

$$BC_b^d(w_1,w_2) = \frac{\Gamma(\alpha_0+a_0)\Gamma(\beta_0+b_0)}{\Gamma(\alpha_0+a_0+d)\Gamma(\beta_0+b_0+d)} \times \sum_{k=1}^K \frac{\Gamma(\alpha_k+c(w_1,f_k)+d)\Gamma(\beta_k+c(w_2,f_k)+d)}{\Gamma(\alpha_k+c(w_1,f_k))\Gamma(\beta_k+c(w_2,f_k))}.$$

導出を付録に示す．この拡張の解釈と効果の検証についても，今後の課題となる．

6.4 関 連 研 究

本研究に最も関連する研究としては，次に述べる 2 つの研究がある．

Rauber ら[22)] は，ディリクレ分布の間の Bhattacharyya 係数に対する解析解について述べている．これは，次の式によって表される．

$$BC(\mathrm{Dir}(\phi|\boldsymbol{\alpha}),\mathrm{Dir}(\phi|\boldsymbol{\beta})) = \frac{\sqrt{\Gamma(\alpha_0)\Gamma(\beta_0)}}{\sqrt{\prod_k\Gamma(\alpha_k)}\sqrt{\prod_k\Gamma(\beta_k)}} \times \frac{\prod_k\Gamma((\alpha_k+\beta_k)/2)}{\Gamma(\frac{1}{2}\sum_k^K(\alpha_k+\beta_k))}. \quad (12)$$

しかし，これはその動機，式の形式ともに我々の BC_b とは異なる．実験的，および，理論的な比較は今後の課題であるが，式 (6) によって得られたディリクレ分布の間の類似度を式 (12) に従って求める予備的な実験では，式 (12) は語の類似度を求めるという観点では意味のある類似度を計算できず，精度が非常に低いことが分かった．

Tsuda ら[27)] は，観測される確率変数 $x\in\mathcal{X}$ と観測できない確率変数 $h\in\mathcal{H}$ を考えたときに，$z=(x,h)$ 上に定義されるカーネル関数 $K_z(z,z')$ があったとき，h のすべての可能性に関して事後確率に従って元のカーネル関数の値を足し合わせるという「周辺化カーネル」を提案した．これは，以下の式で表される．

$$K(z,z') = \sum_{h\in\mathcal{H}}\sum_{h'\in\mathcal{H}} p(h|x)p(h'|x')K_z(z,z') \quad (13)$$

これは，元の類似尺度の期待値を計算するという我々の考え方と，積分と和という定義の差はあるが，共通するものである．しかし，Tsuda らの興味は，x が入力列，h が HMM の隠れ状態列のように構造を持つ場合に，これを効率的に計算することであり，本研究のように，学習データの不確かさをベイズ推定の枠組みで類似度計算に取り入れるというものではない．

7. 結　　論

本研究では，語の意味的類似度を頑健に計算するためのベイズ手法を提案した．提案手法は，ベイズ推定により得られた文脈プロファイルの確率分布の下で，元のベースラインの類似尺度の期待値を計算するというものである．文脈プロファイルが多項分布であり，事前分布がディリクレ分布であり，元の類似尺度が Bhattacharyya 係数の場合には，効率的な計算を可能にする解析解を得ることができることを示した．実験では，提案手法が通常の Bhattacharyya 係数や，Jensen-Shannon ダイバージェンス，PMI ベクトルのコサインなどのよく知られた既存手法，絶対ディスカウンティングを用いた Bhattacharyya 係数に比べて優れていることを示した．

参 考 文 献

1) Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions, *Bull. Calcutta Math. Soc.*, Vol.49, pp.214–224 (1943).
2) Chen, S.F. and Goodman, J.: An empirical study of smoothing techniques for language modeling (1998). TR-10-98, Computer Science Group, Harvard University.
3) Chen, S.F. and Rosenfeld, R.: A Survey of Smoothing Techniques for ME Models, *IEEE Trans. Speech and Audio Processing*, Vol.8, No.1, pp.37–50 (2000).
4) Cortes, C. and Vapnik, V.: Support Vector Networks, *Machine Learning*, Vol.20, pp.273–297 (1995).
5) CRL: EDR Electronic Dictionary Version 2.0 Technical GUIDE (2002). Communications Research Laboratory (CRL).
6) Dagan, I., Lee, L. and Pereira, F.: Similarity-based Methods for Word Sense Disambiguation, *Proc. ACL 97* (1997).
7) Dagan, I., Lee, L. and Pereira, F.: Similarity-Based Models of Word Cooccurrence Probabilities, *Machine Learning*, Vol.34, No.1-3, pp.43–69 (1999).
8) Dagan, I., Marcus, S. and Markovitch, S.: Contextual Word Similarity and Estimation from Sparse Data, *Computer, Speech and Language*, Vol.9, pp.123–152 (1995).
9) Dagan, I., Pereira, F. and Lee, L.: Similarity-based Estimation of Word Cooccurrence Probabilities, *Proc. ACL 94* (1994).
10) Grefenstette, G.: *Explorations In Automatic Thesaurus Discovery*, Kluwer Academic Publishers (1994).
11) Hagiwara, M., Ogawa, Y. and Toyama, K.: PLSI Utilization for Automatic Thesaurus Construction, *Proc. IJCNLP 2005* (2005).
12) Harris, Z.: Distributional Structure, *Word*, pp.146–142 (1954).
13) Hindle, D.: NOUN CLASSIFICATION FROM PREDICATE-ARGUMENT STRUCTURES, *Proc. ACL-90*, pp.268–275 (1990).
14) Kazama, J., De Saeger, S., Torisawa, K. and Murata, M.: Generating a large-scale analogy list using a probabilistic clustering based on noun-verb dependency profiles, *Proc. 15th Annual Meeting of The Association for Natural Language Processing* (*in Japanese*) (2009).
15) Kazama, J. and Torisawa, K.: Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations, *Proc. ACL-08: HLT* (2008).
16) Kneser, R. and Ney, H.: Improved backing-off for m-gram language modeling, *Proc. ICASSP95* (1995).
17) Lin, D.: Automatic retrieval and clustering of similar words, *Proc. COLING/ACL-98*, pp.768–774 (1998).
18) Mochihashi, D., Yamada, T. and Ueda, N.: Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling, *Proc. ACL-IJCNLP 2009*, pp.100–108 (2009).
19) Murata, M., Ma, Q., Shirado, T. and Isahara, H.: Database for Evaluating Extracted Terms and Tool for Visualizing the Terms, *Proc. LREC 2004 Workshop: Computational and Computer-Assisted Terminology*, pp.6–9 (2004).
20) Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.-M. and Vyas, V.: Web-Scale Distributional Similarity and Entity Set Expansion, *Proc. EMNLP 2009*, pp.938–947 (2009).
21) Pantel, P. and Lin, D.: Discovering Word Senses from Text, *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.613–619 (2002).
22) Rauber, T.W., Braun, T. and Berns, K.: Probabilistic distance measures of the Dirichlet and Beta distributions, *Pattern Recognition*, Vol.41, pp.637–645 (2008).
23) Shinzato, K., Shibata, T., Kawahara, D., Hashimoto, C. and Kurohashi, S.: Tsubaki: An open search engine infrastructure for developing new information access, *Proc. IJCNLP 2008* (2008).
24) Teh, Y.W.: A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes, *Proc. COLING-ACL 2006*, pp.985–992 (2006).
25) Terada, A., Yoshida, M. and Nakagawa, H.: A Tool for Constructing a Synonym Dictionary using Context Information, IPSJ SIG Technical Report (in Japanese), pp.87–94 (2004).
26) Tsuchida, M., De Saeger, S., Torisawa, K., Murata, M., Kazama, J., Kuroda, K. and Ohwada, H.: Large Scale Similarity-based Relation Expansion, *Proc. IUCS*

2010 (2010).
27) Tsuda, K., Kin, T. and Asai, K.: Marginalized kernels for biological sequences, *Bioinfomatics*, Vol.18, No.suppl 1, pp.S268–S275 (2002).
28) Yamada, I., Torisawa, K., Kazama, J., Kuroda, K., Murata, M., De Saeger, S., Bond, F. and Sumida, A.: Hypernym Discovery Based on Distributional Similarity and Hierarchical Structures, *Proc. EMNLP 2009* (2009).

付　　録

ここでは，6 章で述べた一般的な場合（BC_b^d）について解析解の導出を行う．まず，ディリクレ分布の正規化項を求めるためにも使われる以下の知られた関係がある．

$$\int_{\triangle} \prod_k \phi_k^{\alpha_k - 1} d\phi = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\alpha_0)} = Z(\alpha)^{-1}. \tag{14}$$

まず，定義から，$BC_b^d(p_1, p_2)$ は以下のようになる．

$$\begin{aligned} BC_b^d(p_1,p_2) &= \iint_{\triangle\times\triangle} \mathrm{Dir}(\phi_1|\alpha)\mathrm{Dir}(\phi_2|\beta) \sum_k \phi_{1k}^d \phi_{2k}^d \, d\phi_1 \, d\phi_2 \\ &= Z(\alpha)Z(\beta) \times \\ &\quad \underbrace{\iint_{\triangle\times\triangle} \prod_l \phi_{1l}^{\alpha_l - 1} \prod_m \phi_{2m}^{\beta_m - 1} \sum_k \phi_{1k}^d \phi_{2k}^d \, d\phi_1 \, d\phi_2}_{A} . \end{aligned}$$

式 (14) を応用すれば，A の部分は以下のように計算できる．

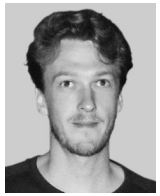
$$\begin{aligned} A &= \int_{\triangle} \prod_m \phi_{2m}^{\beta_m - 1} \left[\sum_k \phi_{2k}^d \int_{\triangle} \phi_{1k}^{\alpha_k + d - 1} \prod_{l\neq k} \phi_{1l}^{\alpha_l - 1} \, d\phi_1 \right] d\phi_2 \\ &= \int_{\triangle} \prod_m \phi_{2m}^{\beta_m - 1} \left[\sum_k \phi_{2k}^d \frac{\Gamma(\alpha_k + d) \prod_{l\neq k} \Gamma(\alpha_l)}{\Gamma(\alpha_0 + d)} \right] d\phi_2 \\ &= \sum_k \frac{\Gamma(\alpha_k + d) \prod_{l\neq k} \Gamma(\alpha_l)}{\Gamma(\alpha_0 + d)} \int_{\triangle} \phi_{2k}^{\beta_k + d - 1} \prod_{m\neq k} \phi_{2m}^{\beta_m - 1} \, d\phi_2 \\ &= \sum_k \frac{\Gamma(\alpha_k + d) \prod_{l\neq k} \Gamma(\alpha_l)}{\Gamma(\alpha_0 + d)} \frac{\Gamma(\beta_k + d) \prod_{m\neq k} \Gamma(\beta_m)}{\Gamma(\beta_0 + d)} \\ &= \frac{\prod \Gamma(\alpha_l) \prod \Gamma(\beta_m)}{\Gamma(\alpha_0 + d)\Gamma(\beta_0 + d)} \sum_k \frac{\Gamma(\alpha_k + d)}{\Gamma(\alpha_k)} \frac{\Gamma(\beta_k + d)}{\Gamma(\beta_k)}. \end{aligned}$$

したがって，

$$BC_b^d(p_1,p_2) = \frac{\Gamma(\alpha_0)\Gamma(\beta_0)}{\Gamma(\alpha_0 + d)\Gamma(\beta_0 + d)} \sum_{k=1}^{K} \frac{\Gamma(\alpha_k + d)\Gamma(\beta_k + d)}{\Gamma(\alpha_k)\Gamma(\beta_k)}.$$

(平成 23 年 4 月 11 日受付)
(平成 23 年 9 月 12 日採録)

風間　淳一（正会員）
独立行政法人情報通信研究機構ユニバーサルコミュニケーション研究所情報分析研究室主任研究員．2004 年東京大学大学院情報理工学系研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．同年北陸先端科学技術大学院大学情報科学研究科助教．2008 年より情報通信研究機構．自然言語処理の研究に従事．



ステイン　デ・サーガ
2006 年北陸先端科学技術大学院大学知識科学研究科博士課程修了．博士（知識科学）．北陸先端科学技術大学院大学研究員を経て，2007 年に情報通信研究機構に入所．2008 年に NICT MASTAR プロジェクト言語基盤グループに専攻研究員として着任．自然言語処理を用いた知識獲得の研究に従事．

黒田　　航（正会員）

京都工芸繊維大学（非常勤講師），京都大学（非常勤講師），早稲田大学総合研究機構（招聘研究員）．元独立行政法人情報通信研究機構知識創成コミュニケーション研究センター MASTAR プロジェクト言語基盤グループ短時間研究員．京都大学人間・環境学博士．言語学と自然言語処理を融合する研究に従事．

村田　真樹（正会員）

1993 年京都大学工学部電気工学第二学科卒業．1997 年同大学院工学研究科電子通信工学専攻博士課程修了．博士（工学）．同年京都大学にて日本学術振興会リサーチ・アソシエイト．1998 年郵政省通信総合研究所入所．独立行政法人情報通信研究機構主任研究員を経て，現在，鳥取大学大学院工学研究科情報エレクトロニクス専攻教授．自然言語処理，情報抽出の研究に従事．2005 年 FIT2005 論文賞受賞．共著書に『事例で学ぶテキストマイニング』（共立出版）等がある．言語処理学会，人工知能学会，電子情報通信学会，計量国語学会，ACL 等の会員．

鳥澤健太郎（正会員）

1992 年東京大学理学部卒業．1994 年同大学院修士課程修了．1995 年同大学院博士課程中退．同年同大学院助手．1998 年科学技術振興事業団さきがけ研究 21 研究員兼任（2002 年まで）．北陸先端科学技術大学院大学助教授を経て，2008 年より独立行政法人情報通信研究機構言語基盤グループ，グループリーダー．2011 年より同機構情報分析研究室室長，現在に至る．博士（理学）．自然言語処理の研究に従事．日本学術振興会賞等受賞．言語処理学会，人工知能学会，ACL 各会員．