修　士　学　位　論　文

トピックモデルをシソーラスとして利用した語義曖昧性解消の領域適応

平成26年度

茨城大学大学院理工学研究科

情報工学専攻

國 井　慎 也

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

本論文では語義曖昧性解消 (Word Sense Disambiguation，WSD) をタスクとした領域適応の問題に対して, トピックモデルとを利用することで WSD の精度を向上させる教師なしの領域適応手法を提案する.

自然言語処理の多くのタスクにおいて帰納学習手法が利用される. そこではコーパス A からタスクに応じた訓練データを作成し, その訓練データから分類器を学習する. そしてこの分類器を利用することで当初のタスクを解決する. このとき実際のタスクとなるデータはコーパス A とは領域が異なるコーパス B 内のものであることがしばしば起こる. この場合, コーパス A （ソース領域）から学習された分類器では, コーパス B （ターゲット領域）のデータを精度良く解析することができない問題が生じる. これが領域適応の問題であり[1], 近年活発に研究が行われている. 本論文では WSD のタスクでの領域適応を行う.

領域適応の手法はターゲット領域のラベル付きデータを利用するかしないかという観点で分類できる．利用する場合を教師付き手法，利用しない場合を教師なし手法と呼ぶ．教師付き手法の研究は多いが, 教師なし手法はパフォーマンスに問題があり, それほど活発には研究されていない. ただしラベル付けを必要としないことは大きな長所であるため, ここでは教師なし手法を扱う.

WSD の領域適応において教師なしの研究としては新納の研究 [2] がある. そこではターゲット領域のコーパスからトピックモデルを構築し, そのトピックモデルから得られるトピック素性をソース領域の訓練データとターゲット領域のテストデータに追加することで, ソース領域の訓練データから学習された分類器の精度を向上させる. ただしこの研究ではターゲット領域のトピックモデルだけを利用し, ソース領域のトピックモデルを利用していない. 上記論文でも指摘されているが,WSD にどのようにトピックモデルを利用できるかは明かではない. さらに WSD の領域適応においては, ソース領域のコーパスから作られるトピックモデル, ターゲット領域のコーパスから作られるトピックモデルおよびソース領域のコーパスとターゲット領域のコーパスを合わせたコーパスから作られるトピックモデルの 3 種類のトピックモデルが利用可能であり, これらをどのように組み合わせて利用すれば効果があるのかも明かではない. 本論文では後者の問題に対する解明を目的とする.

[1] 領域適応は機械学習の分野では転移学習 [1] の一種と見なされている.

本論文でのトピックモデルの利用法としては, 上記新納の研究 [2] と同様の手法を取る. つまりトピックモデルからトピック素性を作り,WSD に利用される通常の素性（ここではこれを**基本素性**と呼ぶ）にトピック素性を連結し, その連結した素性に対して学習手法を適用する. また上記した 3 種類のトピックモデルの利用法については, 基本的に各トピックモデルから得られる 3 つのトピック素性を連結したトピック素性を作り, それを基本素性に連結する. また 3 つのトピック素性を連結する際, ソース領域から得られるトピック素性については重み r を乗じる. これはソース領域から得られるトピック素性が必ずしも WSD の精度を向上させるとは限らず, 逆に悪化させる場合もあるためである. ソース領域から得られるトピック素性が精度向上に寄与すると判断できるときに r の値を 1 に近づけ, 逆に悪化させそうな場合は 0 に近づける.

重み r は以下の式によって設定する.

$$r = \frac{KL(T, S+T)}{KL(T, S+T) + KL(S, S+T)}$$

上記式中の S はソース領域のコーパス,T はターゲット領域のコーパス, $S+T$ はソース領域のコーパスとターゲット領域のコーパスを合わせたコーパスである. また $KL(A, B)$ は B から見た A の KL 情報量である.

実験では BCCWJ [3] における 3 つ領域 PB (書籍),OC (Yahoo! 知恵袋) および PN (新聞) の 3 領域を用いた. 対象単語は各領域にある程度の頻度が存在する多義語 17 単語を対象にした. また領域適応としては,PB から OC,OC から PB,PB から PN,PN から PB,OC から PN,PN から OC の 6 通りが存在する. 各領域適応において, トピック素性の使い方を変化させた実験を行い, 提案手法の有効性を示す.

1.2 本論文の構成

本論文では提案手法とそこに使用されている理論について紹介し, 既存のシソーラス利用した場合との比較実験も行う. 2 章では,WSD の概要と一般的流れを説明し,WSD でよく使用されるサポートベクトルマシンについて述べる 3 章では, 一般的な領域適応や先行研究について説明する. 4 章では, トピックモデルの代表的な手法である LDA について説明する. そして, 従来のトピックモデルを利用した WSD についての説明を行う. 5 章では, 提案手法やトピック素性を作成する方法を説明する. 6 章では, 提案手法を評価する実験方法と実験結果を説明する. 7 章では, 実験結果から考察を行う. 8 章では, 本論文の結論を記す.

第2章　語義曖昧性の解消

2.1　Word Sense disambiguation　(WSD)

2.1.1　概要

語義曖昧性解消 (WSD) は多義語の語義を識別し, 正しい語義を割り当てる処理のことである. 例えば「与える」という単語には, 表 2.1 のような語義が存在する.WSD では, このような多義語を対象として語義曖昧性解消のための処理を行う.

表 2.1:「与える」の語義

単語	語義
与える	(1) 自分の物を他人に渡し, その人のものとする. やる. ▽現在では上の者が恩恵的な意味で授ける場合に使う. (2) あてがう (1).「課題を―」「ヒントを―」 (3) こうむらせる.「損害を―」「不安を―」▽「当たる」「値」と同語源. 原義は, 合うように物をやる意. 関連遣る・授ける・施す・遣わす・恵む・くれる・くださる・賜る・給う・給する・贈る・上げる・差し上げる・貢ぐ・捧げる・供する・奉る・献ずる・渡す・譲る・払い下げ・下付・交付・支給・授与・進呈・贈呈・贈与・寄贈・恵贈・呈上・進上・献上・献呈・謹呈

2.1.2　一般的な流れ

現在の WSD は教師あり学習の手法を用いたものが一般的である. その流れを図 2.1 に示す.

教師あり学習は入力データと解答となるラベルデータのペアを学習させて, 識別したいデータが入力された際に学習内容を模範として出力データを推定する. 対して教師なし学習は, 正解となるラベルデータを含めずに入力データのみを学習して, その性質などから出力を推定する.

学習には大量に集めたコーパスから得られるデータを用いる. このコーパスには, 単に文書を集めたコーパスの他に, 文書中の単語に語義や構文構造などの付加情報を含む自然言語処理に特化した注釈付きコーパスがある. この注釈付きコーパスを使用することでより高い語義識別の精度が得られるが, 大量の文書に付加情報を手動で付けていくことは相当の時間とコスト

が必要である. 学習データになるものには, 文書中の単語の品詞をつけた POS タグや, 対象語の文脈に含まれる単語を集めたもの（bag-of-words）などがある. これらをベクトル化した素性ベクトルを含む教師データから作成した学習モデルを用いて, 語義の識別を行う.

本研究では学習や識別にサポートベクトルマシンを用いるが, その説明は次節で行う.

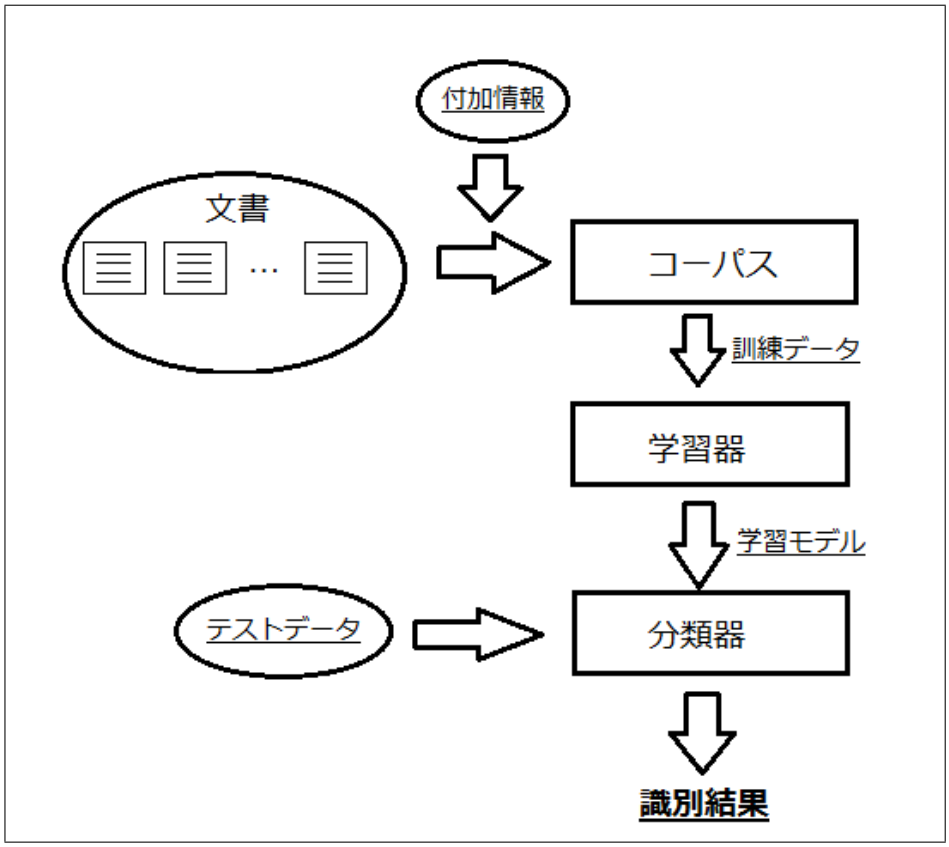

図 2.1: WSD における教師あり学習の流れ

2.2 サポートベクトルマシン

2.2.1 概要

サポートベクトルマシン (Support Vector Machine,SVM)[4] は, 自然言語処理の分野でよく使用される線形二値分類器であり, 非常に高い分類性能を持つことが知られている. また, カーネル法を用いれば SVM は非線形な分類も可能である.

SVM は線形二値分類器であり, クラス数が 2 つであるような問題に用いられてきた. 二つのクラスは, それぞれ正クラス及び負クラスを呼ばれ, 正クラスに属する事例は正例と呼ばれ, 負クラスに属する事例は負例と呼ばれる.

ここで, 訓練データ D が以下で与えられるとする.

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(|D|)}, y^{(|D|)})\} \tag{2.1}$$

$x^{(1)}, x^{(2)}, \cdots, x^{(|D|)}$ は, 事例の素性ベクトルであり,$y^{(1)}, y^{(2)}, \cdots, y^{(|D|)}$ はそれぞれの事例のクラスラベルである. 正例のクラスラベルは $+1$ であり, 負例の事例のクラスラベルは -1 である.SVM は線形分類器であるので, 分離平面の方向ベクトル w と切片 b をパラメータとして以下の関数で表わされる.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - \mathbf{b} \tag{2.2}$$

事例 $\mathbf{x}$ を,$f(\mathbf{x}) \geq \mathbf{0}$ ならば正クラス,$f(\mathbf{x}) < \mathbf{0}$ ならば負クラスに分類される.

2.2.2 マージン最大化

ここでは, パラメータ $\mathbf{w}$ と b を求めるための基本的な考え方を説明する. 図 2.2 のように訓練データが分布していると仮定する.

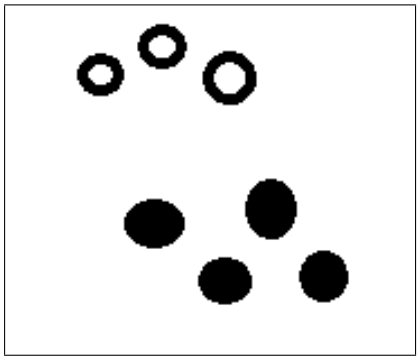

図 2.2: 訓練データ

これを分類する平面を分類平面とよぶ.SVM の目的は, 良い分類平面を構築することである. 方向ベクトルを $\mathbf{w}$ であり, 切片を b で表すと分類平面は $\mathbf{w} \cdot \mathbf{x} = \mathbf{b}$ を満たす点 $\mathbf{x}$ の集合となる.

図 2.2 をみると, この訓練データを分類できる分類平面は多数存在することがわかる.SVM では, この分類平面を「どちらのクラスからもなるべく遠い位置で分ける」という戦略が, 最大マージン最大化とよばれる戦略である. 分離平面のマージンとは, 最も近い訓練事例への距離として定義される.

分離平面の最も近くにある正例を $\mathbf{x}_+$ で表し,$\mathbf{x}_+$ と分離平面を結ぶ垂線の足を $\mathbf{x}_*$ で表すとする. ここで, マージンは $\mid \mathbf{x}_+ - \mathbf{x}_* \mid$ と表わされる. $\mathbf{w}$ と $\mathbf{x}_+ - \mathbf{x}_*$ は同じ方向を向いているので,$\mathbf{w} \cdot (\mathbf{x}_+ - \mathbf{x}_*) = \mid \mathbf{w} \mid\mid \mathbf{x}_+ - \mathbf{x}_* \mid$ が成り立つ.

分離平面 $\mathbf{w} \cdot \mathbf{x} = \mathbf{b}$ は, 式全体を定数倍しても変わらないので, うまく定数倍すれば $\mathbf{w} \cdot \mathbf{x} - \mathbf{b} = \mathbf{1}$ とすることができる. このようにパラメータが調整されているとする. また,$\mathbf{x}_*$ は分離平面上にあるので, 当然ながら $\mathbf{w} \cdot \mathbf{x}_* = \mathbf{b}$ である. よって,

$$\begin{aligned} \mathbf{x}_+ - \mathbf{x}_* &= \mathbf{w}\cdot\mathbf{x}_+ - \mathbf{w}\cdot\mathbf{x}_* \\ &= (b+1) - b \\ &= 1 \end{aligned}$$

となる.$\mathbf{w} \cdot (\mathbf{x}_+ - \mathbf{x}_*) = \mid \mathbf{w} \mid\mid \mathbf{x}_+ - \mathbf{x}_* \mid$ と合わせると,$\mid \mathbf{w} \mid\mid \mathbf{x}_+ - \mathbf{x}_* \mid = \mathbf{1}$ であることがわかるので,

$$\mid \mathbf{x}_+ - \mathbf{x}_* \mid = \frac{\mathbf{1}}{\mid \mathbf{w} \mid} \tag{2.3}$$

が導ける.$\mid \mathbf{x}_+ - \mathbf{x}_* \mid$ はマージンを表しているので, 結局, 分離平面のマージンは,$\frac{1}{|\mathbf{w}|}$ で表される. このままだと扱いにくいので二乗して,$\frac{1}{\mathbf{w^2}}$ を最大化する. さらに, 分数は扱いにくいので, この逆数をとってそれを最小化することにする. つまり $\mathbf{w^2}$ を最小化することになる.

2.2.3 厳密制約下の SVM のモデル

ここまではマージンのことを考えてきたが, 当然ながら訓練事例が正しく分類できる必要がある. $y^{(i)} = +1$ であるような訓練事例については,$\mathbf{w}\cdot\mathbf{x^{(i)}} - \mathbf{b} \geq \mathbf{1}$ であればよく,$y^{(i)} = -1$ であるような訓練事例については,$\mathbf{w}\cdot\mathbf{x^{(i)}} - \mathbf{b} < \mathbf{-1}$ であればよい. この条件は, 次のようにまとめることができる.

$$y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}} - \mathbf{b}) \geq \mathbf{1} \tag{2.4}$$

したがって, これを制約としたつぎのような最適化問題を解けばよい.

$$(min.)\frac{1}{2}\mathbf{w^2} \tag{2.5}$$

$$(s.t.)y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}}-\mathbf{b})-\mathbf{1}\geq\mathbf{0};\forall\mathbf{i}. \tag{2.6}$$

この不等式付き最適化問題は, 凸計画問題であることが知られている. そのため, ラグランジェ法で解ける. ラグランジェ乗数 α_i を導入すると, ラグランジェ関数は以下のようになる.

$$L(\mathbf{w},\mathbf{b},\alpha)=\mathbf{\frac{1}{2}w^2}-\sum_{\mathbf{i}}\alpha_{\mathbf{i}}(\mathbf{y^{(i)}}(\mathbf{w}\cdot\mathbf{x^{(x)}}-\mathbf{b})) \tag{2.7}$$

これを, それぞれのパラメータで偏微分すると, 以下になる.

$$\nabla_{\mathbf{w}}L(\mathbf{w},\mathbf{b},\alpha)=\mathbf{w}-\sum_{\mathbf{i}}\alpha_{\mathbf{i}}\mathbf{y^{(i)}}\mathbf{x^{(i)}},\frac{\partial\mathbf{L}(\mathbf{w},\mathbf{b},\alpha)}{\partial\mathbf{b}}=\sum_{\mathbf{i}}\alpha_{\mathbf{i}}\mathbf{y^{(i)}} \tag{2.8}$$

これらを 0 と置くと以下が得られる.

$$\mathbf{w}^*=\sum_{\mathbf{i}}\alpha_{\mathbf{(i)}}\mathbf{y^{(i)}}\mathbf{x^{(i)}} \tag{2.9}$$

$$\sum_i\alpha_i y^{(i)}=0 \tag{2.10}$$

一つ目の等式（2.9）は, 分離平面の方向ベクトル $\mathbf{w}^*$ は訓練事例ベクトルの線形和で表されることを示している. この等式 $\mathbf{w}^*=\sum_{\mathbf{i}}\alpha_{\mathbf{(i)}}\mathbf{y^{(i)}}\mathbf{x^{(i)}}$ を分離平面の式 $\mathbf{w}\cdot\mathbf{x}=\mathbf{b}$ に代入すると

$$f(\mathbf{x})=\sum_{\mathbf{i}}\alpha_{\mathbf{i}}\mathbf{y^{(i)}}\mathbf{x^{(i)}}\cdot\mathbf{x}-\mathbf{b} \tag{2.11}$$

となる. あとは,α_i と b を求めることが出来れば, 分離平面の式を得ることができる.

そこで, これらの等式をもとのラグランジェ関数はに代入すると,

$$L(\mathbf{w}^*,\mathbf{b},\alpha)=\mathbf{\frac{1}{2}}\sum_{\mathbf{i,j}}\alpha_{\mathbf{i}}\alpha_{\mathbf{j}}\mathbf{y^{(i)}}\mathbf{y^{(j)}}\mathbf{x^{(i)}}\cdot\mathbf{x^{(j)}}-\sum_{\mathbf{i}}\alpha_{\mathbf{i}} \tag{2.12}$$

が得られる. これで, もともとの変数 $\mathbf{w}$ と b がラグランジェ関数はから消去された.

ラグランジェ鞍点理論によると, じつはこのラグランジェ関数を最大化する α を求めればよいことがわかる. ただし,$\sum_i\alpha_i y^{(i)}=0,\alpha_i\geq 0$ という制約のもとでの最大化である. このように最大化問題として双対問題が求められる. この最大化問題は 2 次計画問題として知られている.

最適解 α_i^* が求まれば,$\mathbf{w}^*$ が求まり,$\mathbf{x}_+$ を用いて,$b=\mathbf{x}^*\cdot\mathbf{x}_+-\mathbf{1}$ として切片も求まる.

2.2.4 緩和制約下の SVM モデル

上記で導出した SVM であるが, これは実際のデータに対してはなかなかうまく動かない. それはすべての訓練事例が正確に分類されなくてはならないという制約 $y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}}-\mathbf{b})-\mathbf{1}\geq\mathbf{0}$ があるためである. 訓練データの中には例外的な事例が存在することが多く, こういった事例によって分類平面が大きく影響を受けてしまうことがよくあるからである.

そこで, ここではこの制約を緩めることを考える. 新たな変数 $\xi_i \geq 0$ を導入し

$$y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}}-\mathbf{b})-\mathbf{1}\geq-\xi_\mathbf{i} \tag{2.13}$$

という制約に書き換える.ξ_i は i 番目の訓練事例がうまく分けられていない度合いを示している. つまり,ξ_i は小さいほうがよい. そこで, これを目的関数に加え, 新しい最適化問題は次のようになる.

$$(\text{min.})\frac{1}{2}\mathbf{w^2}+\mathbf{C}\sum_\mathbf{i}\xi_\mathbf{i} \tag{2.14}$$

$$\text{s.t}y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}}-\mathbf{b})\geq\mathbf{1}-\xi_\mathbf{i};\forall\mathbf{i}.,\xi_\mathbf{i}\geq\mathbf{0};\forall\mathbf{i}. \tag{2.15}$$

ここで,C は正の定数であり, この値が大きいほどしっかり分類するようになる.

この最大化問題のラグランジェ関数は以下のようになる.

$$L(\mathbf{w},\mathbf{b},\xi,\alpha,\beta)=\mathbf{\frac{1}{2}w^2}+\mathbf{C}\sum_\mathbf{i}\xi_\mathbf{i}-\sum_\mathbf{i}\alpha_\mathbf{i}(\mathbf{y^{(i)}}(\mathbf{w}\cdot\mathbf{x^{(x)}}-\mathbf{b})-\mathbf{1}+\xi_\mathbf{i})-\sum_\mathbf{i}\beta_\mathbf{i}\xi_\mathbf{i} \tag{2.16}$$

それぞれのパラメータを偏微分する.$\mathbf{w}$ と b は厳密制約下の場合と同じで, $\mathbf{w}^*=\sum_\mathbf{i}\alpha_{(\mathbf{i})}\mathbf{y^{(i)}}\mathbf{x^{(i)}},\sum_\mathbf{i}\alpha_\mathbf{i}\mathbf{y^{(i)}}=\mathbf{0}$ が得られる. ξ_i に関しては,

$$\frac{\partial L(\mathbf{w},\mathbf{b},\xi,\alpha,\beta)}{\partial\xi_i}=C-\alpha_i-\beta_i \tag{2.17}$$

となるので,$C=\alpha_i+\beta_i$ が得られる.

これらの等式を合わせて整理すると, 双対ラグランジェ関数は次のようになる.

$$L(\mathbf{w}^*,\mathbf{b},\alpha,\beta)=-\mathbf{\frac{1}{2}}\sum_\mathbf{i,j}\alpha_\mathbf{i}\alpha_\mathbf{j}\mathbf{y^{(i)}}\mathbf{y^{(j)}}\mathbf{x^{(i)}}\cdot\mathbf{x^{(j)}}-\sum_\mathbf{i}\alpha_\mathbf{i} \tag{2.18}$$

ただし,$0\leq\alpha\leq C$ の制約のもとで最大化する.

b は $0\leq\alpha_i\leq C$ となる事例を一つ持ってきて $b=\mathbf{w}\cdot\mathbf{x^{(i)}}-\mathbf{y^{(i)}}$ とすることで計算できる.

第3章　領域適応

自然言語処理の多くのタスクで教師付きの機械学習手法が利用されるが, そこには領域適応の問題が存在する [5] [6]. 領域適応の問題とは, 学習の際に訓練データとして利用するソースデータの領域と, 学習により得られた分類器を適用する先のターゲットデータの領域が異なる問題であり, 近年活発に研究が行われている.

一般に, 領域適応の手法は事例ベースの手法と素性ベースの手法に分けられる [7]. 事例ベースの手法とは訓練事例に重みをつけて学習する手法であり, 共変量シフト下の学習が代表的研究である [8]. 共変量シフトとは $P_S(\boldsymbol{x}) \neq P_T(\boldsymbol{x})$ であるが, $P_S(y|\boldsymbol{x}) = P_T(y|\boldsymbol{x})$ という仮定である. 共変量シフト下の学習は期待損失最小化から確率密度比 $P_T(\boldsymbol{x})/P_S(\boldsymbol{x})$ を重みとした損失ベースの学習に帰着される. 一方, 素性ベースの手法とはソース領域の素性空間とターゲット領域の素性空間を共通の素性空間に写影する手法である. 概略, 領域間の違いを減少させ, 本来の領域の重要な性質を保持するような次元縮約法となる. Blitzer はソース領域とターゲット領域の両方に頻出する素性を Pivot Features と呼び, それを使って Structural Correspondence Learning (SCL) と呼ばれる次元縮約法を提案した [9]. Pan はソース領域を写影した空間とターゲット領域を写影した空間との距離を Maximum Mean Discrepancy (MMD) によって評価し, これを最小にするような MMD Embedding (MMDE) と呼ばれる変換法を提案した [10]. さらに Sinno は MMDE を改良した Transfer Component Analysis (TCA) と呼ばれる手法を提案した [11]. また素性に重みをつけて学習させる手法も, 素性ベースの手法の一種である. Daumé は簡易な素性の重み付け手法 [12] を提案した. そこではソース領域の訓練データのベクトル $\boldsymbol{x}_s$ を $(\boldsymbol{x}_s, \boldsymbol{x}_s, \boldsymbol{0})$ と連結した 3 倍の長さのベクトルに直し，ターゲット領域の訓練データのベクトル $\boldsymbol{x}_t$ を $(\boldsymbol{0}, \boldsymbol{x}_t, \boldsymbol{x}_t)$ と連結した 3 倍の長さのベクトルに直す．この 3 倍にしたベクトルを用いて，通常の分類問題として解く．この手法はソース領域とターゲット領域に共通している素性が重なることで，共通している素性に重みをつけている形になる.

一方, 領域適応の問題は, 訓練データのスパース性の問題とも見なせる. このため,Self-Training, 半教師付き学習 [13] あるいは能動学習も領域適応 [14][15] に利用できる. 特に Self-Training はターゲット領域のデータにラベル付けを必要としない教師なし領域適応が可能であるため有用性が高い.

Chen は, 領域適応に Self-Training を利用する際に, 領域間の違いを考慮して素性に重みをつけることで学習の効果を高める手法を提案した [16].

3.1 Self-training

Self-Training は, ターゲット領域のデータにラベル付けを必要としない教師なし領域適応の手法として利用できる. 今, ラベル付きのデータを L, ラベルなしデータを U とする. 初期の状態では,L はソース領域のラベル付きデータであり, U はターゲット領域のラベルなしデータである.

Self-Training の各ステップでは, 現在の訓練データ L から分類器 h が学習され, それをラベルなしデータ U に適用する. このとき U の各データにラベルに対する確信度を与える. 確信度が高い上位 c 個のデータに,h によって識別されたラベルを付与し, c 個のデータを L に追加する. このステップを U が空になるか, 全てのデータの確信度がある閾値以下になったときに終了する.

Self-Training は確信度の与え方と c の設定が必要である. ここでは学習アルゴリズムとして SVM を利用する. SVM により得られた分離超平面までの距離によって確信度を与える.

3.2 重み付き self-training

素性への重みとして,Chen は素性とクラスラベルとの相関係数を利用した [16].

データ $\boldsymbol{x}$ の素性 f の値を x_f, データ $\boldsymbol{x}$ のクラスを y_x とする. ソース領域のラベル付きデータに対して x_f と y_x の相関係数を $\rho_S(x_f, y_x)$ とおく. ターゲット領域のデータ $\boldsymbol{x}$ については, そのクラスが不明であるが, その時点の分類器で識別されたクラス y'_x を代用することで, x_f と y'_x の相関係数 $\rho_T(x_f, y'_x)$ が得られる.

そして素性 f の重み係数 $w(f)$ を以下で定義する.

$$w(f) = \frac{1 + \rho_S(x_f, y_x)\rho_T(x_f, y'_x)}{2} \tag{3.1}$$

$w(f)$ を利用して,x_f は以下のように更新される.

$$x_f \leftarrow x_f + \gamma_n w(f)$$

ここで n は Self-Training のステップ数を表し, γ_n は $\gamma_n \to 0$ を満たす. 具体的にここでは γ_n を以下で定義した.

$$\gamma_n = 0.01 \cdot \frac{N - n}{N}$$

ここで N は Self-Training のステップ数の最大値である.

また各素性 f に対して x_f を更新した後に $\boldsymbol{x}$ の大きさを 1 に正規化する.

3.3 重み付き self-training の多値分類への拡張

クラス分布に対する素性の重みの分布を利用した新たな重み付け法を説明する [17]. この重みは, ソース領域とターゲット領域の両領域において, ある素性が同じような使われ方をしているなら, 大きくし, 逆に異なる使われ方をしている場合は小さくするように設定される. 従来手法は多クラス分類のタスクにおいては拡張が出来ないが, この手法では多クラス分類においても適応できる.

本論文では以下の手続きによって $w(f)$ を定義する.

$$\hat{i} = \arg\max_i P_S(x, y_i)$$

$$\hat{j} = \arg\max_j P_T(x, y_j)$$

もし,$\hat{i}$ と $\hat{j}$ が等しいならば,

$$w(f) = \max(P_S(x, y_{\hat{i}}), P_T(x, y_{\hat{j}})$$

等しくないならば,

$$w(f) = 0$$

と定義する.

ここでソース領域 S のデータ $\boldsymbol{x}$ の中で, クラスが i となっているデータ $\boldsymbol{x}$ の集合を S_i と書くことにし,P_S を以下で定義する.

$$P_S(x, y_i) = \frac{\sum_{\boldsymbol{x} in S_i} x_f}{\sum_{\boldsymbol{x} in S} x_f}$$

ターゲット領域 T のデータ $\boldsymbol{x}$ に関しては, クラスが未知である. そのためその時点での分類器によって与えられたクラスを $\boldsymbol{x}$ のクラスと考え, T の中でクラスが i となっているデータ $\boldsymbol{x}$ の集合を T_i と書くことにし,P_T を以下で定義する.

$$P_T(x, y_i) = \frac{\sum_{\boldsymbol{x} in T_i} x_f}{\sum_{\boldsymbol{x} in T} x_f}$$

3.4 スパースネス問題

領域適応に対して外部の資源を利用し精度を上昇させる方法について述べる. 領域適応の問題の他, 自然処理言語の多くのタスクにおいて, 訓練事例数が学習データの次元数より小さいというスパースネス問題が存在し, この問題により正確なモデルを構築できないことが多い. スパースネス問題を解決するために, 一般的にシソーラスが利用される. シソーラスとしては分類語彙表などの手作業で構築されたものとコーパスから自動構築されたものがある. 前者

は質が高いが分野依存の問題がある．後者は質はそれほど高くないが，分野毎に構築できるという利点がある．トピックモデルをシソーラスとして利用した研究としては [18][19] がある. また, トピックモデルを利用して概念辞書を作成した研究としては [20] がある. 本論文では, 語彙分類表などの手作業で構築されたものと自動で分野ごとに利用したものとの比較実験を行う.

第4章　トピックモデル

トピックモデルは, 離散データを解析する手法として,bag-of-wrods 表現された文書の生成過程を確率的にモデル化したものである. トピックモデルにおいてある文書に含まれる各単語は, 文書固有のトピック (話題) 比率に従ってあるトピックを選択した後, そのトピックに固有の単語出現確率分布に従って生成されると仮定する. 代表例として,Latent Dirichlet Allocation（LDA)[21] や (Probabilistic Latent Semantic Indexing（PLSI）[22] があり, 情報検索, 文書クラスタリング, 協調フィルタリングなど, さまざまな分野に応用されている.

LDA のモデルのパラメータの推定方法には, 大きく分けて変分ベイズを使用する方法とギブスサンプリングを使用する方法がある. 今回は, 変分ベイズを使用した LDA の説明を行う.

4.1　Latent Dirichlet Allocation

Latent Dirichlet Allocation（LDA)[21] は, 文書集合の確率生成モデルである.LDA の混合比はディリクレ事前分布より生成するため, 訓練データにないような語を扱えることができるのがが特徴である. 基本的には, 文書が単語の分布によって特徴付けられる潜在的なトピックをランダムに含むものとして表現される.

LDA によってコーパスの各文書は, 以下のプロセスによって生成される.

1. ポアソン分布 ξ より, 文書の語数 N を選択する.
2. ディリクレ分布 α より, トピックの混合比 θ を選択する.
3.N 個の各単語 w_n について
   (a) 多項分布 θ より, トピック z_n を選択する.
   (b) トピック z_n で条件付けられた多項確率 $p\left(w_n|z_n,\beta\right)$ より, 単語 w_n を選択する.

LDA のモデルは幾つかの単純な仮定によって形成されている.

はじめに,k 次元のディリクレ分布は既知であり固定されることを仮定する. 次に, 単語の確率 $p(w|z_n,\beta)$ は, 固定量として推定される $K\times V$ の行列 $\beta_{ij}=p(w^j=1|z^i=1)$ によってパラメータ化される. 最後に, ポアソン分布によって決められる文章の長さ N の存在は重要ではなく, また他のパラメータ θ ,z と独立である.

k 次元のディリクレ分布のランダム変数 θ は,(k-1) シンプレックス (simplex, 単体) 上の値をとり, このシンプレックス上では以下の確率密度を持つ.

$$p(\theta \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1} \tag{4.1}$$

ただし, パラメータ α は k 次元のベクトルであり, 各成分は $\alpha_i > 0$ である. また, $\Gamma(x)$ は Gamma 関数である. このディリクレ分布は, 有限な次元の十分統計量を持ち, 多項分布の共役事前分布になっている指数分布族である.

次に, パラメータ α と β, トピックの混合比 θ, トピック z_n, 単語 w_n の条件付き同時確率は, 以下で与えられる.

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \mathbf{p}(\theta \mid \alpha) \prod_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{p}(\mathbf{z_n} \mid \theta) \mathbf{p}(\mathbf{w_n} \mid \mathbf{z_n}, \beta) \tag{4.2}$$

さらに, θ について積分を行い,z に関して足し合わせると, 文書の周辺確率を得ることができる.

$$p(\mathbf{w} \mid \alpha, \beta) = \int \mathbf{p}(\theta \mid \alpha) (\prod_{\mathbf{n=1}}^{\mathbf{N}} \sum_{\mathbf{z_n}} \mathbf{p}(\mathbf{z_n} \mid \theta) \mathbf{p}(\mathbf{w_n} \mid \mathbf{z_n}, \beta)) \mathbf{d}\theta \tag{4.3}$$

最後に, 1 つの文書の周辺確率を積分すると, コーパス D の確率を得る.

$$p(\mathbf{D} \mid \alpha, \beta) = \prod_{\mathbf{d=1}}^{\mathbf{M}} \int \mathbf{p}(\theta_\mathbf{d} \mid \alpha) (\prod_{\mathbf{n=1}}^{\mathbf{N_d}} \sum_{\mathbf{z_{dn}}} \mathbf{p}(\mathbf{z_{dn}} \mid \theta) \mathbf{P}(\mathbf{w_{dn}} \mid \mathbf{z_{dn}}, \beta)) \mathbf{d}\theta_\mathbf{d} \tag{4.4}$$

LDA のグラフィカルモデルを図 4.1 に示す. この図が示している通り LDA は, 3 つの階層か

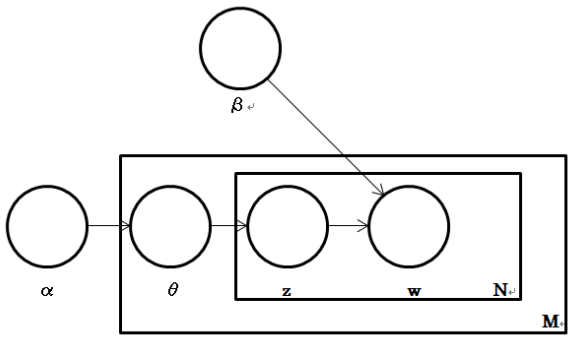

図 4.1: LDA のグラフィカルモデル

らなっている. パラメータ α と β はコーパスレベルのパラメータであり, コーパスを生成する過程で一度サンプリングされる. 変数 θ_d は文書レベルのパラメータであり, 文書ごとにサ

ンプリングされる. 最後に, 変数 z_n と w_{dn} は単語レベルのパラメータであり, 各文書の各単語に対してサンプリングされる.

ここで,LDA のモデルと他の単純なディリクレ多項モデルを違いを説明する. 古典的なクラスタリングのモデルは 2 つの階層からなっており, 一つのコーパスに対してディリクレ分布から一度サンプリングされる. 多項クラスタリング変数は, コーパスの各文書に対して一度サンプリングされる単語の集合のクラスタ変数によって選択される. そのため, 古典的なクラスタリングモデルでは文書が単一なトピックを関連付けられることを制限する. 一方,LDA のでは 3 つの階層に分かれており, 特にトピックのノードが文書内で繰り返しサンプリングされる. そのため LDA のモデルでは, 文書が複数のトピックに関連付けられる.

4.2 推論とパラメータの推定

ここまで LDA のモデルについて述べてきたがこのままでは LDA を使用できない. つまり, 与えられる文書等のデータから LDA のパラメータを求める必要がある. そのためここでは,LDA における推論の方法とパラメータの推定方法について述べる.

4.2.1 推論

LDA を使用するために解く必要がある主要なな推論の問題は, 文書の与えられたときの潜在変数の事後分布を求めることである.

$$p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{\mathbf{p}(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{\mathbf{p}(\mathbf{z} \mid \alpha, \beta)} \tag{4.5}$$

しかし, この分布は一般的に扱いにくいため潜在変数を周辺化する. すると以下の式が得られる.

$$p(\mathbf{w} \mid \alpha, \beta) = \frac{\mathbf{\Gamma}(\sum_{\mathbf{i}} \alpha_{\mathbf{i}})}{\prod_{\mathbf{i}} \mathbf{\Gamma}(\alpha_{\mathbf{i}})} \int (\prod_{\mathbf{i=1}}^{\mathbf{k}} \theta_{\mathbf{i}}^{\alpha_{\mathbf{i}}-\mathbf{1}})(\prod_{\mathbf{n=1}}^{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \prod_{\mathbf{j=1}}^{\mathbf{V}} (\theta_{\mathbf{i}} \beta_{\mathbf{ij}})^{\mathbf{w_n^j}}) \mathbf{d}\theta \tag{4.6}$$

この関数は, θ と β が積の形になっているのが扱いにくい. しかしながら, 正確な近似をするためにラプラス近似や変分近似など多くの種類の近似推論アルゴリズムが考案されている. 次の節では,LDA での変分ベイズでの推論の方法について述べる.

4.2.2 変分ベイズ推論

変分ベイズの基本的な考え方は, 対数尤度の適切な下限を得るために Jensen の不等式を利用することである. 本質的には, 下限求めるために変分パラメータを導入することである. この変分パラメータは, 可能な限り正確な下限を見つけるための最適化手法によって選択される.

扱いやすい下限を求める単純な方法は, いくつかの矢印やノードを削除した元の LDA のグラフィカルモデルを修正することを考えることである.

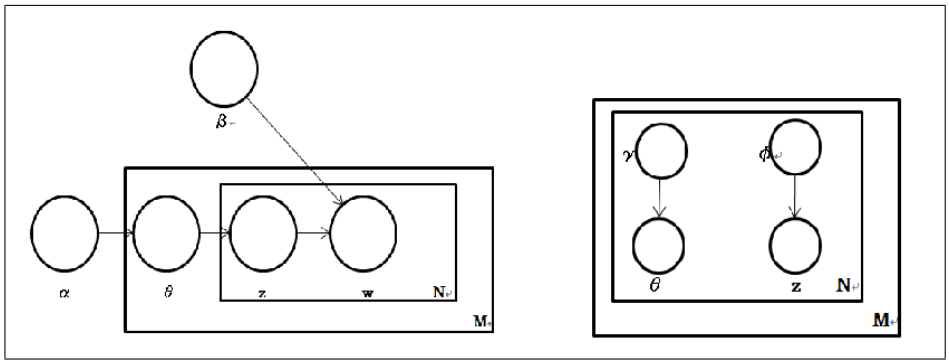

図 4.2: (左図)LDA のグラフィカルモデル.(右図) 変分ベイズを使用したグラフィカルモデル)

問題のある θ と z の組は, 図 4.2 の θ ,**z** 及び **w** の矢印において生じる. それらの矢印と **w** のノードを削除し, 潜在変数の分布を得るために自由変分パラメータを導入する. この分布は, 以下の変分分布に特徴づけられる.

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = \mathbf{q}(\theta \mid \gamma) \prod_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{q}(\mathbf{z_n} \mid \phi_\mathbf{n}) \tag{4.7}$$

ただし, ディリクレパラメータ γ と多項パラメータ $(\phi_1, \cdots, \phi_N)$ は, 自由変分パラメータである.

簡略された確率分布を明示したので, 次は変分パラメータ γ と ϕ の値を決定する最適化問題を解くことになる. 変分パラメータの値を決定することは, 以下に示す対数尤度の下限を求める最適化問題を解くこと同値である.

$$(\gamma^*, \phi^*) = argmin_{(\gamma,\phi)} D(q(\theta, \mathbf{z} \mid \gamma, \phi) \parallel \mathbf{p}(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)) \tag{4.8}$$

つまり, 変分パラメータの値の最適化は, 変分分布と真の分布 $p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)$ のカルバックライブラーダイバージェンス (Kullback Leibler(KL)divergence) を最小化することである. この最小化は, 反復定点法によって実行される. また, 式 (4.8) の γ と ϕ の更新式は以下で表される.

$$\theta_{ni} \propto \beta_{iw_n} exp(E_q[log(\theta_i) \mid \gamma]) \tag{4.9}$$

$$\gamma_i = \alpha_i + \sum_{n-1}^{N} \phi_{ni} \tag{4.10}$$

式 (2.9) の期待値は以下で定義される.

$$E_q[log(\theta_i) \mid \gamma] = \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \tag{4.11}$$

ただし,Ψ は対数ガンマ関数を 1 次の項までテイラー展開したものである.

式 (4.9) と式 (4.10) には, 魅力的な直感的解釈が存在する. ディリクレパラメータの更新は, 変分分布 $E[z_n \mid \phi_n]$ のもとでの観測変数が与えられた時の事後分布である. また, 多項パラメータの更新は,$p(z_n \mid w_n) \propto p(w_n \mid z_n)p(z_n)$ で使用されるベイズの定理に似ている.

変分分布は実際 **w** によって変化する条件付き分布である. これは式 (2.8) の最適化問題が w が固定されて,w の関数である最適化パラメータ (γ^*, ϕ^*) が与えらて実行されるためである. そして, 変分分布は w の陽関数を従属した $q(\theta, \mathbf{z} \mid \gamma^*(\mathbf{w}), \phi^*(\mathbf{w}))$ として書くことができる. つまり, 変分分布は事後分布 $p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)$ の近似値としてみなすことができる.

文章においては, 最適化パラメータ $(\gamma^*(\mathbf{w}), \phi^*(\mathbf{w}))$ は文章固有のものである. 特に, ディリクレパラメータ $\gamma^*(\mathbf{w})$ は, トピックシンプレックスにおける文書を表現するものとして考えれる.

(1) initialize $\phi^0_{ni} := 1/k$ for all i and n
(2) initialize $\gamma_i := \alpha_i + N/k$ for all i
(3) **repeat**
(4) **for** $n = 1$ **to** N
(5) **for** $i = 1$ **to** k
(6) $\phi^{t+1}_{ni} := \beta_{iw_n} \exp(\Psi(\gamma^t_i))$
(7) normalize ϕ^{t+1}_n to sum to 1.
(8) $\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi^{t+1}_n$
(9) **until** convergence

図 4.3: 変分ベイズ推論アルゴリズム

変分ベイズアルゴリズムまとめたものを図 4.3 に示す. この擬似コードから,LDA のための変分ベイズの反復には,$O((N+1)k)$ の操作が要求される. 経験的には, 一つの文書に必要な繰り返し回数は文書内の単語数のオーダーであることがわかる. つまり, おおよその繰り返し総数は,N^2k のオーダーで必要である.

4.2.3 パラメータの推定

この節では,LDA モデルにおけるパラメータ推定のための経験ベイズ法を説明する. 特に, 文書 $D = \{\mathbf{w_1}, \mathbf{w_2}, \cdots, \mathbf{w_n}\}$ が与えられたときのデータの以下の対数尤度を最大にする α と β を求めることを中心に説明する.

$$l(\alpha, \beta) = \sum_{d-1}^{M} logp(\mathbf{w_d} \mid \alpha, \beta) \tag{4.12}$$

上記で説明したように,$p(\mathbf{w} \mid \alpha, \beta)$ は簡単に求めることができない. しかし, α と β に関しての最大化する対数尤度の下限を変分推論によって求めることができる. また, 変分パラメータ γ と θ を関しての下限を最大化し, そして, 変分パラメータの値を固定して LDA モデルのパラメータ α と β に関して下限を最大化する変分 EM アルゴリズムを介して LDA モデルの経験ベイズの推定値を求めることができる.

以下にパラメータを求めるための変分 EM アルゴリズムの手順を示す.

1.(E-step) 各文書に対しパラメータ α と β を固定して, 変分パラメータ (γ_d^*, ϕ_d^*) を求める.

2.(M-step) E ステップで求めた対数尤度の下限を最大化するパラメータ α と β を求める.

これらの 2 つのステップを対数尤度の下限が収束するまで行う.

以下に,M ステップで行われる β の更新式を示す.

$$\beta_{ij} = \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni}^* w_{dn\cdot}^j \tag{4.13}$$

4.3 トピックモデルを利用した WSD

この節では, 従来手法のトピックモデルを利用した WSD について説明するトピックモデルを利用した WSD に研究として, ミドルソフトタグを利用したものなどがある [18][19]. まず, 従来手法のトピックベクトルを利用した WSD の流れを以下に示す.

１. 大量の文書を収集し, それぞれを形態素解析を行い, 索引語文書列を作成する. 索引語文書列は以下のように, 各行が文書 d, 各列は単語 ID と文書内での単語の出現回数のペアである.

1:1 2:1
2:1 3:1 4:2
1:2 3:1 4:1

２. LDA のトピック数を指定し, 作成した索引語行列からトピックモデルを構築する. トピック $z_i (i = 1, 2, \cdots, K)$ の下で名詞である単語 w の分布 $p(w \mid z_i)$ を求められる.

３. 対象単語の用例 s に含まれる対象単語以外の名詞リストを $w_1, w_2, \cdots, w_m$ とする. 文 s のトピック素性は K 次元ベクトルとなり, 第 i 次元の値はトピック z_i への関連度に対応する. 関連度は基本的には LDA から求まる以下の $p(s \mid z_i)$ を利用する.

$$p(s \mid z_i) = \frac{1}{Z} \prod_{j=1}^{m} p(w_j \mid z_i)$$

ここでZはトピック素性の大きさを 1 にする正規化定数である.

$$Z = \sum_{i=1}^{K} \prod_{j=1}^{m} p(w_j \mid z_i)$$

以上の行程で算出された $p(s \mid z_i)$ を用いて, 文 s に対するトピック素性を以下のように定義する.

$$(p(s \mid z_i), p(s \mid z_i), \cdots, p(s \mid z_K))$$

こうして作成したトピック素性を従来の基本素性ベクトルに加えて学習識別に用いる.

第5章　提案手法

5.1　素性ベクトルの作成

本実験で使用する素性は, 以下の 8 種類である. なお対象単語の直前の単語を w_{-1}, 直後の単語を w_1 とする.

e0 w の表記

e1 w の品詞

e2 w_{-1} の表記

e3 w_{-1} の品詞

e4 w_1 の表記

e5 w_1 品詞

e6 w の周辺自立語

e7 e6 の分類語彙の番号 4 桁と 5 桁

5.2　WSD へのトピックモデルの利用

近年 WSD に対しては教師付き学習手法が大きな成功を収めているが, そのアプローチにはデータスパースネスの問題が生じる. 一般にデータスパースネスに対してはシソーラスが利用される. シソーラスとしては，分類語彙表などの手作業で構築されたものとコーパスから自動構築されたものがある．前者は質が高いが分野依存の問題がある．後者は質はそれほど高くないが，分野毎に構築できるという利点がある．ここでは領域適応の問題を扱うので，後者を利用する．

トピックモデルとは文書 d の生起に K 個の潜在的なトピック z_i を導入した確率モデルである．

$$p(d) = \sum_{i=1}^{K} p(z_i)p(d|z_i)$$

トピックモデルの 1 つである Latent Dirichlet Allocation (LDA) [21] を用いた場合，単語 w に対して $p(w|z_i)$ が得られる．つまりトピック z_i をひとつのクラスタと見なすことで，LDA を利用して単語のソフトクラスタリングが可能となる．

ある領域のコーパスと LDA を利用して，その領域に適した $p(w|z_i)$ が得られる．$p(w|z_i)$ の情報を WSD に利用するいくつかの研究 [23][24][25] があるが，ここでは基本的にハードタグ

[26] を利用する．ハードタグとは w に対して最も関連度の高いトピック $z_{\hat{i}}$ を付与する方法である．

$$\hat{i} = \arg\max_i p(w|z_i)$$

まずトピック数を K としたとき，K 次元のベクトル t を用意し，入力事例 x 中の単語 $w_j(j = 1 \sim n)$ に対して最も関連度の高いトピック $z_{\hat{i}}$ を求め，t の $\hat{i}$ 次元の値を 1 にする．これを w_1 から w_n まで行い t を完成させる．作成できた t をここでは**トピック素性**と呼ぶ．トピック素性を基本素性に結合することで，新たな素性ベクトルを作成し，その素性ベクトルを対象に学習と識別を行う．

5.3 ３つのトピック素性

領域適応ではソース領域のコーパス, ターゲット領域のコーパス, それらを合わせたコーパスの 3 種類のコーパスが存在するため, それぞれからトピックモデルを学習することで, データに対して 3 種類のトピック素性を作ることができる. 今, ソース領域に対するトピック素性を tp(S), ターゲット領域に対するトピック素性を tp(T), ソース領域のコーパスとターゲット領域のコーパスを合わせたコーパスから学習できたトピックモデルから作られるトピック素性を tp(S+T), そして WSD で通常利用される基本素性を B と表記することにする.

トピック素性を WSD に利用する方法としては以下のケースが考えられる.

1. B + tp(T)
2. B + tp(S+T)
3. B + tp(T) + tp(S+T)
4. B + tp(T) + tp(S)
5. B + tp(T) + tp(S+T) + tp(S)
6. B + tp(T) + tp(S+T) + r * tp(S)

ターゲット領域の知識を反映するために,tp(T) や tp(S+T) を含めるのは当然なので, 単純な利用法としては (1) や (2) である. (3) はターゲット領域の知識に重みを付けた形であり, この手法も有望である. 問題は tp(S) の利用である.

領域適応の問題に対して, 一般に言えることであるが, ソース領域の知識をどのように利用するかが問題解決の鍵である. ソース領域の知識（この研究の場合は tp(S) にあたる）を利用すると,WSD の精度が向上する場合もあるし, 逆に悪化する場合もある. このため (4) の形が (1),(2),(3) よりも良くなる保証はない.

ただし (5) の形は tp(S) を利用しているが, 有望な手法である. これは Daumé の手法 [12] と類似の考え方である. Daumé の手法では, ソース領域の訓練データのベクトル x_s を $(x_s, x_s, 0)$ と連結した 3 倍の長さのベクトルに直し, ターゲット領域の訓練データのベクトル x_t を $(0, x_t, x_t)$ と連結した 3 倍の長さのベクトルに直す. この 3 倍にしたベクトルを用いて, 通常の分類問題として解く. この手法は非常に簡易でありながら, 効果が高い手法として知られている. この手法はソース領域とターゲット領域に共通してる特徴が重なることで, 結果として共通してる特徴の重みがつくことで領域適応に効果が出ると考えられる. (5) の形はソース領域の知識 tp(S) とターゲット領域の知識 tp(T) に共通した知識 tp(S+T) を追加している形と見なせるからである.

本論文の提案は (6) の形である. これは (5) の形の改良である. 前述したようにソース領域の知識 tp(S) が悪影響を及ぼすケースもあるので, tp(S) に重み r を乗じるというものである.

5.4 ソース領域の重み付き利用

本論文では以下の形でトピック素性を WSD に利用する.

B + tp(T) + tp(S+T) + r * tp(S)

ここで r の設定法が問題となる. ここでは r を「ソース領域が持つ一般知識の程度」と考える.

一般に領域適応の問題においては, ソース領域の知識をどのように利用するかが問題解決の鍵である. この問題はソース領域とターゲット領域との類似性と密接に関連している.

5.4.1 領域間の類似性

領域適応は本質的にソース領域とターゲット領域がある程度「似ている」ことが前提としてある. 全く異なっていた場合は, ソース領域のデータはターゲット領域において, 全く役立たないことが明白だからである. この「似ている」度合いを形式的に定義することは難しく, 領域適応の黎明期から領域適応の最も重要な問題の 1 つとして認識されている.

神嶌はこの「似ている」という概念に普遍的な定義を与えるのではなく, どのような観点で類似していて, その上でソース領域のどのような知識がターゲット領域で利用できるかを仮定し, その仮定をどのように数学的にモデル化するかが重要だと指摘している [1]. この観点から見ると, 領域適応の問題は対象とするタスクに応じて領域間の類似性を測り, その類似の度合いを学習に利用するという形は自然である.

Asch は品詞タグ付けをタスクとして領域間の類似性を測り, その類似度から領域適応を行った際に精度がどの程度悪くなるかを予測できることを示した [27]. 張本は構文解析をタスクとしてターゲット領域を変化させたときの精度低下の要因を調査し, そこから新たな領域間の類似性の尺度を提案している [28]. Plank は構文解析をタスクとして領域間の類似性を測ること

で, ターゲット領域を解析するのに最も適したソース領域を選んでいる [29]. Ponomareva[30] や Remus[31] は感情極性分類をタスクとして領域間の類似度を学習中のパラメータに利用した. これらの研究はタスク毎に類似性を測るが,WSD がタスクの場合, 領域間の類似性は WSD の対象単語に依存していると考えられる. 古宮は対象単語毎に領域間の距離を含めた性質[1]によって適用する学習手法を変化させている [32][33][34].

5.4.2 重み r の設定

ソース領域とターゲット領域の類似性を測るということは, ソース領域とターゲット領域に共通している知識（一般知識）とターゲット領域に特有の知識（特殊知識）を分離することを意味する. 類似性は, 本質的に, 共通している知識（一般知識）と特有の知識（特殊知識）との比較によって測られるからである.

ここでは重み r は「ソース領域が持つ一般知識の程度」と考えられる. そのため r の算出には一般知識をどのように設定するかがポイントである. ここではソース領域のコーパス S とターゲット領域のコーパス T を合わせたコーパス $S+T$ によって一般知識を表すことにする. これは 2 つのコーパスを合わせると共通する部分の重みがまし, 全体的に共通する部分に近づくと考えられるからである. これを利用して, コーパス S や T の一般知識との距離を KL 情報量を利用して $KL(S, S+T)$ と $KL(T, S+T)$ とし, 以下の関係を仮定する.

$$1-r : r = KL(S, S+T) : KL(T, S+T)$$

これにより r を以下の式で算出する.

$$r = \frac{KL(T, S+T)}{KL(T, S+T) + KL(S, S+T)}$$

$KL(S, S+T)$ の測り方について述べておく. コーパス $S+T$ 内の名詞 w について,w のコーパス S 内の頻度 $f_s(w)$ とコーパス $S+T$ 内の頻度 $f_{s+t}(w)$ を調べる. $KL(S, S+T)$ の定義は以下である.

$$KL(S, S+T) = \sum_w p_s(w) \log \frac{p_s(w)}{p_{s+t}(w)}$$

ここで $p_{s+t}(w)$ はコーパス $S+T$ における単語 w の出現確率であり, 以下となる.

$$p_{s+t}(w) = \frac{f_{s+t}(w)}{N_{s+t}}$$

ここで $N_{s+t} = \sum_w f_{s+t}(w)$ である. また $p_s(w)$ はコーパス S における単語 w の出現確率であり, 以下で定義する.

$$p_s(w) = \frac{f_s(w)+1}{N_s+V}$$

ここで $N_s = \sum_w f_s(w)$ であり, V はコーパス $S+T$ における名詞の種類数である.

[1]それら性質を全て含めて, 領域間の類似性と呼べる.

第6章　実験

現代日本語書き言葉均衡コーパス（BCCWJ [3]）の PB(書籍),OC(Yahoo!　知恵袋) および PN(新聞) を異なった領域として実験を行う．各領域に対してある程度の頻度をもつ多義語 17 単語を WSD の対象単語とする[1]．これら単語と辞書上での語義数及び各コーパスでの頻度と語彙数を表 6.1 に示す[2]．また PB と OC のコーパスは BCCWJ のものを利用したが, PN のコーパスについては 95 年度版の毎日新聞を利用した.

領域適応としては領域が 3 つあるので, それらの組み合わせで 6 通りの領域適応を行う.

実験は各領域適応において, トピック素性を利用せずに基本素性 B だけを用いた場合と 5.3 章で示した 6 通りのトピック素性の利用法を行った場合の合計 7 種類の手法を試した. 各手法に対して, 上記 17 単語の WSD を行い, それらの平均正解率 (%) を求めた. なお学習手法としては SVM [3]を用いた. またトピックモデルは LDA を用いて学習し[4], その際, トピック数は 100 で固定した. 結果を表 6.2 に示す.

トピックモデルを全く利用しない手法 (1) はその他の手法と比べて正解率が低く, WSD にトピックモデルを利用する効果が示せている. また提案手法 (6) が最も平均正解率が高く, 提案手法の有効性が示せた.

[1] PB と OC に対して頻度が 50 以上という条件をつけた.

[2] 語義は岩波国語辞書がもとになっている．そこでの中分類までを対象にした．また「入る」は辞書上の語義が 3 つだが，PB や OC では 4 つの語義がある．これは BCCWJ では新語義のタグも許しているからである．

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4] http://chasen.org/~daiti-m/dist/lda/

表 6.1: 対象単語

単語	辞書の語義数	PB 頻度	PB 語義数	OC 頻度	OC 語義数	PN 頻度	PN 語義数
言う	3	1114	2	666	2	363	2
入れる	3	56	3	73	2	32	2
書く	2	62	2	99	2	27	2
聞く	3	123	2	124	2	52	2
来る	2	104	2	189	2	19	1
子供	2	93	2	77	2	29	2
時間	4	74	2	53	2	59	2
自分	2	308	2	128	2	71	2
出る	3	152	3	131	3	89	3
取る	8	81	7	61	7	43	7
場合	2	137	2	126	2	73	2
入る	3	118	4	68	4	65	3
前	3	160	2	105	3	106	4
見る	6	273	6	262	5	87	3
持つ	4	153	3	62	4	59	3
やる	5	156	4	117	3	27	2
ゆく	2	133	2	219	2	27	2
平均	3.35	193.9	2.94	150.6	2.88	72.2	2.59

表 6.2: 実験結果 (平均正解率 %)

	OC→PB	OC→PN	PB→OC	PB→PN	PN→OC	PN→PB	平均
(1) B	74.18	70.18	70.38	76.94	69.25	74.88	72.64
(2) B + tp(T)	74.58	68.40	70.89	77.78	70.13	75.80	72.93
(3) B + tp(S+T)	73.48	70.46	72.70	78.50	70.25	76.24	73.61
(4) B + tp(T) + tp(S+T)	73.61	69.88	72.45	78.90	70.36	76.86	73.68
(5) B + tp(T) + tp(S)	73.61	68.79	72.09	78.91	70.17	76.48	73.34
(6) B + tp(T) + tp(S+T) + tp(S)	73.92	68.70	72.18	79.41	70.53	76.71	73.58
(7) B + tp(T) + tp(S+T) + r *tp(S) (提案手法)	73.63	69.89	72.14	79.08	70.58	77.17	**73.75**
重み r	0.0174	0.01139	0.9825	0.35655	0.98861	0.6434	

第7章　考察

7.1　トピックモデルの利用方法

本論文ではトピックモデルからトピック素性を作成し, そのトピック素性を基本素性に結合するという形で, トピックモデルを利用している. ただしトピックモデルを WSD に利用する方法はいくつか提案されている.

WSD にトピックモデルを利用する場合，間接的な利用と直接的な利用に分けられる.

間接的な利用とは WSD で必要とするリソースをトピックモデルで補強するようなタイプである．Cai は WSD に対してベイジアンネットワークを利用しているが，そのベイジアンネットワークに LDA から求めたトピック素性を組み込むことでもとのベイジアンネットワークを改良している [26]．Boyd-Graber は追加の潜在変数として WordNet の語義を LDA に組み込んだ．これによって WordNet から同義語グループ（synset）を探す処理の中に，トピックモデルを利用している [24]．Li はコーパスから語義の事前分布が得られる場合，そうでない場合，そしてコーパスの言い換えのリソースが不足していた場合の 3 つの状況に合わせた WSD のための確率モデルを構築する手法を提案している. この構築のためにトピックモデルを利用している [23].

直接的な利用とはトピックモデルから得られるトピック素性を，直接 WSD に利用するタイプである．本手法もこのタイプに属する．Boyd-Graber は LDA によって単語と単語の周辺分布を求め，周辺分布の類似性から単語の語義を推定した [25]．ただし，これは教師なしの枠組みであるため，本論文での基本素性を WSD に利用しておらず，通常の教師あり学習により得られる分類器をトピックモデルを利用して改善する形ではない．前述した Cai の研究 [26] では，提案手法との比較手法として基本素性にトピック素性を結合した素性を利用した手法も実装している．そこではトピック素性を最大の関連度をもつトピックだけ 1 にし，残りを 0 にしたハードタグと，全てのトピックについてその関連度を使ったソフトタグを試しており，ソフトタグの方が優れていることを報告している.

実装の容易性から考えると直接的な利用の方が優れている. ただしこの場合, トピックモデルを構築するコーパスの領域の他に, そのコーパスのサイズあるいはトピックモデルのトピック数などが精度に大きな影響を与える. これらの適切な値の推定も必要である. 特に, 本実験で用いたコーパスは素のテキストで PB では 16 MB,OC では 10 MB,PN では 15MB であり, OC が他と比べるとかなり小さい. そのため OC と他の領域間の類似性が小さくなり, 結果的にソース領域が OC の場合, 重み r の値が小さくなっている.

7.2　既存シソーラスとの比較

本論文ではトピックモデルをシソーラスとして利用した. 既存のシソーラスを使った場合との比較を行った. なお既存のシソーラスとしては分類語彙表を利用した. 結果を表 7.1 に示す.

既存シソーラスを利用するよりも, 領域に応じたトピックモデルを使う方が正解率が高い.

この結果は一般の WSD を解く際にも一般のシソーラスを使うだけでなく, タスクの対象となる領域のコーパスから得たシソーラスを利用する方が良いことを示唆している. さらに本研究の結果を考えると, 既存のシソーラスにトピックモデルを組み合わせて利用することも効果がある可能性がある. この点は今後の課題である.

7.3 シソーラスの領域依存性

領域適応の問題を考える場合, 全ての領域の知識を集めた知識を作れば, その知識がすべての領域で共通して利用できるという考え方がある. 実際にそのようなタスクも存在する. 例えば森は単語分割のタスクにおいて, 各々の領域のタグ付きデータを使うことで精度を上げることができたが, 全ての領域のタグ付きデータを使えば更に精度を上げることができたことを報告している [35].

本論文で対象としたタスクに対しても, 全ての領域のコーパスを合わせたコーパスを作り, そこからトピックモデルを作成すれば, そのトピックモデルは両領域で利用できると考えられる. これは実験で示した手法 (3) の B + tp(S+T) に対応するものであり, その実験結果でもよい評価値を出している. さらにターゲット領域の知識がターゲット領域において有効であるのは明かであり, 手法 (4) の B + tp(T) + tp(S+T) が手法 (3) よりも効果があることは予想できる. 実験の結果もそれを示している.

問題は tp(S) の利用である. 基本的には tp(S) は利用しなくても良い. ただしソース領域のコーパスとターゲット領域のコーパスを合わせたコーパス S+T と類似している場合に tp(S) の利用効果がある. 具体的には $KL(S, S+T)$ が $KL(T, S+T)$ と比べて極端に大きくなる場合のみ,tp(S) の利用効果がある. 本論文では重み r をつけて tp(S) を利用しているが, 実際は上記の場合のみ $r=1$ として, それ以外は tp(S) を利用しないという形でも可能である.

7.4 対象単語ごとのシソーラスの領域依存性

本論文の提案手法における tp(S) への重み r は領域適応毎に設定しているが, WSD では対象単語に応じて最適な領域適応の手法は異なるという考え方もある. トピックモデルの利用法が単語毎に異なるのかどうかを調べた.

表 7.2 は各単語の各領域適応において, 最も正解率が高かった手法を示している. なお表中の番号は, 表 6.2 に示した手法の番号に対応する.

表 7.2 を見ると, いくつかの単語は領域の組み合わせに依らずに有効な手法が存在している. 例えば「ゆく」は手法 4 ,「書く」は手法 5 ,「自分」は手法 4 がよい. また「やる」や「来る」は手法にほぼ依存していない. これら以外に対しては有効手法は単語に依存していない. また表 7.2 は有効手法が対象領域に依存していないことも示している. つまり WSD の領域適応ではトピックモデルの有効利用手法が対象単語と対象領域の 2 つから決まると考えて良い.

表 7.1: 既存シソーラスとの比較

	提案手法	B + シソーラス
OC→PB	73.63	72.85
OC→PN	69.89	70.64
PB→OC	72.14	70.68
PB→PN	79.08	78.13
PN→OC	70.58	69.72
PN→PB	77.17	75.87
平均	**73.75**	**72.98**

表 7.2: 単語毎の最適手法

単語	OC→PB	OC→PN	PB→OC	PB→PN	PN→OC	PN→PB
言う	1	2	3	1	6 7	3 5
入れる	2	5	4	6	3	7
書く	5	3	1 2 3 4 5 6 7	1 2 3 4 5 7	2	3 5 6 7
聞く	6	4 7	3	2	2 4	3
来る	3 4	1 2 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
子供	5	1 2 3 5 6	4	4 7	4	3
時間	2 6	6	1 2 3 4 5 6 7	6	2 4 5 6 7	3
自分	4	1	4	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
出る	2	3 4 7	6	2 3 4	5	4
取る	1 2 4 5 6	4 7	3	6	5	2
場合	1 3 4 6	1	2	1	3 6 7	3
入る	4	1	5	6	3 5 6 7	7
前	4	1 3	1	5 6	6 7	6
見る	1	1	1 3	1	3	2
持つ	1	2 6	3	3	2 3 4	1 2 3 4 5 6 7
やる	1 2 3 5	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
ゆく	4	1 2 3 4 5 6 7	4 6 7	1 3 4 5 6 7	1 2 3 4 5 6 7	2 4

第8章　おわりに

本論文では領域適応の WSD タスクに対して, トピックモデルを利用する手法を提案した. 具体的にはソース領域のコーパス, ターゲット領域のコーパスおよび両者を合わせたコーパスの各々からトピックモデルを学習し, 各々のトピックモデルからトピック素性を作成する. 結果,3 つのトピック素性が得られるが, それらをすべてを基本素性に結合して, その素性を WSD の学習に利用する. ただしソース領域から得られるトピック素性に関しては,WSD の精度を悪化させるケースもあるため, 重みをつけて素性を作成する. この重みは領域適応の問題の本質に関わる領域間の類似性から得られるが, ここではその類似性をコーパス間の KL 情報量で測った. 実験は BCCWJ コーパスの 3 領域を対象にして,6 タイプの領域適応を設定した. 17 単語を対象単語にし, トピック素性の様々な使い方を用いて WSD の正解率を測り, 平均正解率で評価した. 結果, 提案手法の有効性を示せた. WSD でのトピックモデルの効果的な利用方法を検討するのが今後の課題である.

謝辞

本研究を進めるにあたり、ご指導を頂いた指導教員の新納浩幸准教授に感謝致します。また、日常の議論を通じて多くの知識や示唆を頂いた新納研究室の皆様に感謝します。

参考文献

[1] Toshihiro Kamishima. Transfer learning (in japanese). *The Japanese Society for Artificial Intelligence*, Vol. 25, No. 4, pp. 572–580, 2010.

[2] Hiroyuki Shinnou and Minoru Sasaki. Domain Adaptation for Word Sense Disambiguation using k-Nearest Neighbors Method and Topic Model (In Japanese). pp. NL–211, 2013.

[3] Kikuo Maekawa. Design of a Balanced Corpus of Contemporary Written Japanese. In *Symposium on Large-Scale Knowledge Resources (LKR2007)*, pp. 55–58, 2007.

[4] 高村大也. 言語処理のための機械学習入門. コロナ社, 2010.

[5] Shinsuke Mori. Domain adaptation in natural language processing (in japanese). *The Japanese Society for Artificial Intelligence*, Vol. 27, No. 4, pp. 365–372, 2012.

[6] Anders Sogaard. *Semi-Supervised Learning and Domain Adaptation in Natural Language Processing*. Morgan & Claypool, 2013.

[7] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 22, No. 10, pp. 1345–1359, 2010.

[8] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation* . MIT Press, 2011.

[9] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128, 2006.

[10] Sinno Jialin Pan, James T Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *AAAI*, Vol. 8, pp. 677–682, 2008.

[11] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, Vol. 22, No. 2, pp. 199–210, 2011.

[12] Daumé III, Hal. Frustratingly Easy Domain Adaptation. In *ACL-2007*, pp. 256–263, 2007.

[13] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, Vol. 2. MIT press Cambridge, 2006.

[14] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[15] Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. Domain adaptation meets active learning. In *NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pp. 27–32, 2010.

[16] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *NIPS*, pp. 2456–2464, 2011.

[17] 國井慎也, 新納浩幸, 佐々木稔, 古宮佳那子. 素性に重みを付ける Self-training 手法を用いた文書分類の領域適応. 言語処理学会第 21 回年次大会, (to appear),2015.

[18] 國井慎也, 新納浩幸, 佐々木稔. ミドルソフトタグのトピック素性を利用した語義曖昧性解消. 言語処理学会第 19 回年次大会, pp. 3–9, 2013.

[19] Shinya Kunii and Hiroyuki Shinnou. Use of Combined Topic Models on Unsupervised Domain Adaptation for Word Sense Disambiguation. In *PACLIC-27*, pp. 415–422, 2013.

[20] 新納浩幸, 國井慎也, 佐々木稔. 語義曖昧性解消を対象とした領域固有のシソーラスの構築. 第 5 回日本語学ワークショップ, pp. 199–206, 2014.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Machine Learning Reseach*, Vol. 3, pp. 993–1022, 2003.

[22] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pp. 50–57, 1999.

[23] Linlin Li, Benjamin Roth, and Caroline Sporleder. Topic Models for Word Sense Disambiguation and Token-based Idiom Detection. In *ACL-2010*, pp. 1138–1147, 2010.

[24] Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. A Topic Model for Word Sense Disambiguation. In *EMNLP-CoNLL-2007*, pp. 1024–1033, 2007.

[25] Jordan Boyd-Graber and David Blei. Putop: Turning Predominant Senses into a Topic Model for Word Sense Disambiguation. In *SemEval-2007*, pp. 1024–1033, 2007.

[26] Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. Improving Word Sense Disambiguation using Topic Features. In *EMNLP-CoNLL-2007*, pp. 1015–1023, 2007.

[27] Vincent Van Asch and Walter Daelemans. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pp. 31–36, 2010.

[28] Keiko Harimoto, Yusuke Miyao, and Junichi Tsujii. Kobunkaiseki no bunyatekiou ni okeru seido teika youin no bunseki oyobi bunyakan kyori no sokutei syuhou (in japanese). In *The 16th Annual Meeting on Journal of Natural Language Processing*, pp. 27–30, 2010.

[29] Barbara Plank and Gertjan van Noord. Effective measures of domain similarity for parsing. In *ACL-2011*, pp. 1566–1576, 2011.

[30] Natalia Ponomareva and Mike Thelwall. Which resource is best for cross-domain sentiment analysis? In *CICLing-2012*, 2012.

[31] Robert Remus. Domain adaptation using domain similarity- and domain complexity-based instance selection for cross-domain sentiment analysis. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW 2012) Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction (SENTIRE)*, pp. 717–723, 2012.

[32] Kanako Komiya and Manabu Okumura. Automatic Domain Adaptation for Word Sense Disambiguation Based on Comparison of Multiple Classifiers. In *PACLIC-2012*, pp. 75–85, 2012.

[33] Kanako Komiya and Manabu Okumura. Automatic Determination of a Domain Adaptation Method for Word Sense Disambiguation using Decision Tree Learning. In *IJCNLP-2011*, pp. 1107–1115, 2011.

[34] 古宮嘉那子, 奥村学. 語義曖昧性解消のための領域適応手法の決定木学習による自動選択. 自然言語処理, Vol. 19, No. 3, pp. 143–166, 2012.

[35] 森信介. 自然言語処理における分野適応. 人工知能学会誌, Vol. 27, No. 4, pp. 365–372, 2012.

ソースコード 8.1: ファイルから自立語とその頻度を抜き出して、ファイル出力するプログラム

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use Encode;
6  use encoding "shift-jis";
7
8
9      #ファイルオープン
10 my $file_input="$ARGV[0]";
11
12 open (IN, $file_input) or die "$!";
13
14 my @data;
15 my %hash;
16 my $key;
17
18 while (<IN>) {#読み込みファイルから一行読み込み
19     # 改行コードの除去
20     chomp ($_);
21     my $line=$_;
22
23     # 各行を半角空欄区切りで分割
24      $line=~s/　/ /g;
25     @data = split(/ /, $line);
26     my $j=0;
27     for ($j=0; $j<@data; $j++) {
28             $data[$j] =~ s/( )+//g;
29             chomp ($data[$j]);
30     }
31
32
33
34
35
36     for(my $k=0;$k<@data;$k++){
37         my $b=@data -1;
38             if($data[$k]=~m/e0=(.+)/){
39                 $hash{$data[$k]}++;
40
```

```
41              }
42
43
44      elsif ($data[$k]=~m/e1=(.+)/){
45
46                      $hash{$data[$k]}++;
47
48
49
50          }
51          elsif ($data[$k]=~m/e2=(.+)/){
52
53                      $hash{$data[$k]}++;
54
55          }
56  elsif ($data[$k]=~m/e3=(.+)/){
57
58                      $hash{$data[$k]}++;
59
60          }
61          elsif ($data[$k]=~m/e4=(.+)/){
62
63                      $hash{$data[$k]}++;
64
65          }
66  elsif ($data[$k]=~m/e5=(.+)/){
67
68                      $hash{$data[$k]}++;
69
70          }
71          elsif ($data[$k]=~m/e6=(.+)/){
72
73                  $hash{$data[$k]}++;
74
75          }
76  elsif ($data[$k]=~m/e7=(.+)/){
77
78                  $hash{$data[$k]}++;
79
80          }
81
82
```

```
83
84      }
85
86  }
87
88  foreach $key (keys %hash) {##結果出力
89
90      print "$key $hash{$key}\n";
91
92  }
93
94
95
96  ##終了処理
97  close (IN);
```

ソースコード 8.2: ファイルから対象単語のラベル付け、ファイル出力するプログラム

```
1    #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5   use Encode;
6   use encoding "shift-jis";
7
8
9       #ファイルオープン
10  my $file_input="label_sjis";  ##元データファイル
11  my $file_output="seikai_label.txt";##出力先ファイル
12  open (IN, $file_input) or die "$!";
13  open (OUT, ">$file_output") or die "$!";
14
15
16  my @data;
17
18  while (<IN>) {#読み込みファイルから一行読み込み
19      # 改行コードの除去
20      chomp ($_);
21      my $line=$_;
22
23      # 各行を半角空欄区切りで分割
24        $line=~s/　/ /g;
25      @data = split(/ /, $line);
```

```
26      my $j=0;
27      for ($j=0; $j<@data; $j++) {
28              $data[$j] =~ s/( )+//g;
29              chomp ($data[$j]);
30      }
31
32
33
34
35      my $k=@data-1;
36      if($data[$k]=~m/(\d+)-(\d)-(\d)-(\d)/){##有効な形式であるかどうか
37  my $s1=$1; my $s2=$2; my $s3=$3; my $s4=$4;
38  my $s="$s3"."$s4";
39         print OUT "$s\n";
40
41         }else {print OUT "999\n";}
42
43
44  }
45
46
47
48      ##終了処理
49  close (IN);
50  close (OUT);
```

ソースコード 8.3: tf ファイルを作成するプログラム

```
1     #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5   my $N_line=$ARGV[0];
6
7   my $INDEXfile="INDEX";
8   my $N=100;#ファイルの数
9
10  my @array1=();
11  my @array2=();
12  my @INDEX=(\@array1,\@array2);#INDEXファイルの値を読み込むための配列
13
14  open (IND, "<$INDEXfile") or die "$!";#INDEXファイルオープン
15
```

```
16 my $ii=0;
17 my $i;
18
19 while (<IND>) {
20
21     # 改行コードの除去
22     chomp ($_);
23
24
25     # 各行をカンマ区切りで分割
26     my @data1 = split(/,/, $_);
27
28     # 全角、半角空白の除去
29     for ($i=0; $i<@data1; $i++) {
30         $data1[$i] =~ s/(　|　)+//g;
31     }
32
33     $INDEX[$ii][0]=$data1[0];#INDEXの数字を格納
34     $INDEX[$ii][1]=$data1[1];#INDEXの名詞を格納
35     $ii++;
36 }
37
38 close(IND);
39
40 my $file_input;
41 my $file_output;
42 my @data2;
43
44 for($i=1;$i<=$N;$i++){#読み込みと書き込み用のファイル名を作成
45     $file_input=$i."hindo";
46     $file_output=$i."tf";
47
48     #ファイルオープン
49     open (IN, $file_input) or die "$!";
50     open (OUT,">$file_output") or die "$!";
51
52
53     while (<IN>) {#読み込みファイルから一行読み込み
54     # 改行コードの除去
55         chomp ($_);
56
57     # 各行を半角空欄区切りで分割
```

```
58          @data2 = split(/ /, $_);
59          my $j=0;
60          for ($j=0; $j<@data2; $j++) {
61                  $data2[$j] =~ s/(  | )+//g;
62  }
63
64
65      #*.tfに書き込み
66          my $k=0;
67          for($k=0;$k<$N_line;$k++){
68              if(($INDEX[$k][1]) eq $data2[0]){
69                  print  OUT "$INDEX[$k][0] $data2[1] \n";
70              }
71          }
72      }
73      ##終了処理
74      close (IN);
75      close (OUT);
76  }
```

ソースコード 8.4: 訓練データを作成するプログラム

```
1    #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5
6   my @array1=(1,51);
7   my @array2=(50,100);
8
9   my $file1="seikai_label.txt";
10  my $file2="train.dat";
11
12  open (IN1,"$file1") or die "$!";
13  open (OUT,">$file2") or die "$!";
14
15  my @data1;
16  my $i=1;
17  while(<IN1>){
18      chomp($_);
19      $data1[$i]=$_;
20      $i++;
21  }
```

```
22
23
24  for(my $j=1;$j<=50;$j++){
25
26      my $file3=$j.".tf";
27      open(IN2,"$file3") or die "$!";
28
29      my $x=0;
30      my @DATA1=();
31      my @DATA2=();
32      my @DATA=(\@DATA1,\@DATA2);
33
34      while(<IN2>){
35          chomp($_);
36          my @dat = split(/\t/, $_);
37
38          for (my $j=0; $j<$#dat+1; $j++) {
39                      $dat[$j] =~ s/( | )+//g;
40          }
41
42          $DATA[$x][0]=$dat[0];
43          $DATA[$x][1]=$dat[1];
44          $x++;
45
46      }
47
48      for(my $a=0;$a<$#DATA+1;$a++){##データを次元でソート
49          for(my $b=$a+1;$b<$#DATA+1;$b++){
50              if($DATA[$a][0] > $DATA[$b][0]){
51                  my $temp=$DATA[$a][0];
52                  $DATA[$a][0]=$DATA[$b][0];
53                  $DATA[$b][0]=$temp;
54
55                  $temp=$DATA[$a][1];
56                  $DATA[$a][1]=$DATA[$b][1];
57                  $DATA[$b][1]=$temp;
58
59              }
60          }
61      }
62      print OUT "$data1[$j] ";
63      for(my $i=0;$i<$#DATA+1;$i++){
```

```
64              print OUT "$DATA[$i][0]:1 ";
65          }
66          print OUT "\n";
67
68
69
70  }
```

ソースコード 8.5: 素性ファイルから自立語を取り出すプログラム

```
1   #!/usr/bin/perl
2
3
4   ##ファイルから自立語とその頻度を抜き出して、ファイル出力するプログラム
5   use strict;
6   use warnings;
7   use Encode;
8   use encoding "shift-jis";
9
10
11      #ファイルオープン
12  my $file_input="$ARGV[0]";
13
14  open (IN, $file_input) or die "$!";
15
16  my @data;
17  my %hash;
18  my $key;
19  my $a;
20
21  while (<IN>) {#読み込みファイルから一行読み込み
22      # 改行コードの除去
23      chomp ($_);
24      my $line=$_;
25
26      # 各行を半角空欄区切りで分割
27      $line=~s/　/ /g;
28      @data = split(/ /, $line);
29      my $j=0;
30      for ($j=0; $j<@data; $j++) {
31              $data[$j] =~ s/( )+//g;
32              chomp ($data[$j]);
33      }
```

```
34
35
36
37
38
39      for(my $k=0;$k<@data;$k++){
40          my $b=@data -1;
41
42          if ($data[$k]=~m/e6=(.+)/){
43              $hash{$1}++;
44          }
45
46                  if ($data[$k]=~m/e2=(.+)/){
47              $hash{$1}++;
48          }
49
50          if ($data[$k]=~m/e4=(.+)/){
51              $hash{$1}++;
52          }
53
54      }
55
56  }
57  foreach $key (keys %hash) {##結果出力
58      print "$key $hash{$key}\n";
59  }
60
61      ##終了処理
62  close (IN);
```

ソースコード 8.6: トピックモデルを取り出すプログラム

```
1    #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5   use Encode;
6   use encoding "shift-jis";
7
8   my $file1="INDEX123"; # INDEX
9   my $INDEX_line=$ARGV[0];
10  my $file3="model.beta";
11
```

```
12 my $topic=100;  ##トピック数
13
14 my @DATA1=();
15 my @DATA2=();
16 my @DATA=(\@DATA1,\@DATA2);
17
18 my @DATA3=();
19 my @DATA4=();
20 my @DATA_beta=(\@DATA3,\@DATA4);
21
22 my @file111 = glob "*.e6";
23
24
25 open (IN, $file1) or die "$!";
26
27 my $x=0;
28 while (<IN>) { #INDEX123読み込み
29     chomp($_);
30     my @words = split(",");
31     $words[0]=~s/(  | )+//g;
32     $words[1]=~s/(  | )+//g;
33
34     $DATA[$x][0]=$words[0];
35     $DATA[$x][1]=$words[1];
36
37     $x++;
38
39 }
40
41 close (IN);
42
43 open (IN, $file3) or die "$!";
44 $x=0;
45
46 while(<IN>){
47     chomp($_);
48     my @words = split(" ");
49
50     for(my $i=0;$i<@words;$i++){
51     $words[$i]=~s/(  | )+//g;
52
53
```

```
54      $DATA_beta[$x][$i]=$words[$i];
55
56      }
57
58
59      $x++;
60
61  }
62
63  close (IN);
64
65  for(my $q =1;$q <= 100 ;$q++){
66
67      my $inputfile=$q.".e6";
68      my $out=$q.".kakuritu";
69  open (IN, $inputfile) or die "$!";
70      open (OUT, ">$out") or die "$!";
71
72  while(<IN>){
73
74      chomp($_);
75      my @words = split(" ");
76      $words[0]=~s/(  |  )+//g;
77      my $count;
78      my $flag=0;
79      for(my $j=0;$j<$INDEX_line;$j++){
80
81          if(($DATA[$j][1]) eq $words[0]){
82              $count = $DATA[$j][0]-1;
83              $flag=1;
84              last;
85          }
86
87      }
88
89      if($flag==1){
90          for(my $k=0;$k<$topic;$k++){
91             print OUT sprintf("%e ",$DATA_beta[$count][$k]);
92          }
93          print OUT "\n";
94
95          $flag=0;
```

```
96
97      }else{
98
99          for (my $i=0; $i<$topic;$i++){
100
101             print OUT sprintf("%e ",0);
102         }
103
104         print OUT "\n";
105
106     }
107
108
109
110 }
111     close(OUT);
112 }
```

ソースコード 8.7: 基本素性とトピック素性を結合するプログラム

```
1   #!/usr/bin/perl
2
3
4  use strict;
5  use warnings;
6  use Encode;
7  use encoding "shift-jis";
8
9  my $file1="train.dat";##元のトレーニングデータ
10 my $file2="kihon_topic_label";##Topic素性のみのデータ
11 my $file3="attend_train.dat";##出力先のファイル
12
13 my @DATA1=();
14 my @DATA2=();
15 my @DATA=(\@DATA1,\@DATA2);
16
17
18 my @DATA3=();
19 my @DATA4=();
20 my @DATA5=(\@DATA3,\@DATA4);
21
22 my $topic=100;
23
```

```
24 open (IN, $file1) or die "$!";
25
26 my @count;
27 my $j=0;
28 my $q=0;
29 while(<IN>){
30
31      chomp($_);
32      my @data = split(/ /, $_);
33      my $i;
34      for( $i=0;$i<@data;$i++){
35
36           $DATA[$j][$i]=$data[$i];
37
38      }
39      $count[$j]=$i;
40      $j++;
41
42 }
43
44 close(IN);
45
46 open (IN, $file2) or die "$!";
47
48 my $k=0;
49 while(<IN>){
50
51      chomp($_);
52      my @data = split(/ /, $_);
53
54      for(my $m=1;$m<@data;$m++){
55
56           $DATA5[$k][$m]=$data[$m];
57
58      }
59
60      $k++;
61
62 }
63
64 close(IN);
65
```

```
66
67 open (OUT, ">$file3") or die "$!";
68
69
70 for(my $a=0;$a<50;$a++){##結果出力
71
72     for(my $b=0;$b<$count[$a];$b++){
73
74         print OUT "$DATA[$a][$b] ";
75
76     }
77
78     for(my $c=1;$c<=$topic;$c++){
79
80         print OUT "$DATA5[$a][$c] ";
81
82     }
83     print OUT "\n";
84
85 }
```