平成 24 年度茨城大学工学部情報工学科卒業研究論文

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

平成 25 年 2 月 8 日

茨城大学工学部情報工学科
國井慎也 (09T4026L)
指導教員 新納浩幸 准教授

平成 24 年度 茨城大学工学部情報工学科 卒業研究

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

著者：國井慎也 (09T4026L)

指導教員：新納浩幸 准教授

論文要旨

本論文ではトピックモデルを語義曖昧性解消 (Word Sense Disambiguation, WSD) に利用する手法を提案する．

WSD は文中の多義語である単語に対して，その単語がどの語義で使われているかを識別するタスクである．このタスクは自然言語処理の中心課題であり，従来より多数の手法が提案されているが，近年は教師付き学習を用いる手法が中心である．そこではまず少量の対象単語の用例を用意し，その用例中の対象単語に語義を付与する．次に用例中の対象単語の周辺の文脈を素性リストで表し，先に付与した語義とのペアを作る．このペアの集合が訓練データとなり，この訓練データから様々な機械学習手法を利用して語義の分類器を学習することで WSD を解決する．

本論文では,WSD の精度向上のためにためにトピックモデルを利用する．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．トピックモデルである Latent Dirichlet Allocation を用いた場合，単語 w に対して $p(w \mid z_i)$ が得られることから，ある文脈 s を単語の集合 $\{w_1, w_2, \cdots, w_m\}$ で表し，$p(s \mid z_i)$ を $\prod_{j=1}^{m} p(w_j \mid z_i)$ と近似することで WSD の対象単語の用例に対してトピック z_i への関連度が得られる．

トピック素性を WSD に直接利用する方法は，基本素性にトピック素性を結合させた素性ベクトルを作成し，その素性ベクトルに対して学習手法を適用することである．ここでトピック素性の表現方法として，ハードタグとソフトタグがある．ハードタグとはトピック素性のなかで最も関連度の高いトピックの次元のみを 1 にし，他は 0 としたベクトルである．またソフトタグとは各トピック z_i との関連度 $p(s \mid z_i)$ を第 i 次元の値としたベクトルである. トピック素性はシソーラスの情報であることを考えると，関連度の低いトピックの値は小さい値ではなく 0 である方が好ましいはずである．そこで本論文ではハードタグとソフトタグの中間のタイプであるミドルソフトタグを提案する．ミドルソフトタグではトピック z_i との関連度 $p(s \mid z_i)$ がある閾値よりも小さい場合に，第 i 次元の値を 0 にし，そうでなければ $p(s \mid z_i)$ とする. そして最後に得られたベクトルの大きさを 1 に正規化したものである．

実験では SemEval-2 の日本語 WSD タスクのデータを利用して, 提案手法の効果が確認できた. 今後の課題としては, トピック数や閾値のより適切な設定法があげられる. これにより, さらなる精度の向上ができると考えている．

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

本論文ではトピックモデルを語義曖昧性解消 (Word Sense Disambiguation, WSD) に利用する手法を提案する．

WSD は文中の多義語である単語に対して，その単語がどの語義で使われているかを識別するタスクである．このタスクは自然言語処理の中心課題であり，従来より多数の手法が提案されているが，近年は教師付き学習を用いる手法が中心である．そこではまず少量の対象単語の用例を用意し，その用例中の対象単語に語義を付与する．次に用例中の対象単語の周辺の文脈を素性リストで表し，先に付与した語義とのペアを作る．このペアの集合が訓練データとなり，この訓練データから様々な機械学習手法を利用して語義の分類器を学習することで WSD を解決する．

このアプローチで問題となるのは，素性リストのスパース性 (Feature Sparseness) である．例えば素性として周辺の単語表記だけを利用する場合，訓練データから得られる素性の集合は，取り得る素性全体に対してスパースとなる．そのためテストデータでは訓練データには全く出現しない単語のみで素性リストが構成される場合も多く，その場合，正しく語義識別ができない可能性が高い．

スパース性を解消するためにシソーラスを利用することが一般に行われているが，トピックモデルを利用することも 1 つの方法である．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．トピックモデルである Latent Dirichlet Allocation(LDA) [1] を用いた場合，単語 w に対して $p(w \mid z_i)$ が得られることから，ある文脈 s を単語の集合 $\{w_1, w_2, \cdots, w_m\}$ で表し，$p(s \mid z_i)$ を $\prod_{j=1}^{m} p(w_j \mid z_i)$ と近似することで WSD の対象単語の用例に対してトピック z_i への関連度が得られる．

これを利用して素性のスパース性に対処できる可能性がある．ここではトピックへの関連度によって作られた素性ベクトルをトピック素性と呼ぶことにし，ピック素性以外の従来使われていた WSD の素性ベクトルを基本素性と呼ぶことにする．

トピック素性を WSD に直接利用する方法は，基本素性にトピック素性を結合させた素性ベクトルを作成し，その素性ベクトルに対して学習手法を適用することである．ここでトピック素性の表現方法として，ハードタグとソフトタグがある．ハードタグとはトピック素性のなかで最も関連度の高いト

ピックの次元のみを 1 にし，他は 0 としたベクトルである．またソフトタグとは各トピック z_i との関連度 $p(s \mid z_i)$ を第 i 次元の値としたベクトルである．Cai [2] は WSD においてハードタグよりもソフトタグの方が効果があることを示した．ただしトピック素性はシソーラスの情報であることを考えると，関連度の低いトピックの値は小さい値ではなく 0 である方が好ましいはずである．そこで本論文ではハードタグとソフトタグの中間のタイプであるミドルソフトタグを提案する．

1.2 本論文の構成

本論文では, その手法と理論について紹介し, トピック数や閾値の設定方法についても考察する. 2 章では, トピックモデルである LDA について説明する.3 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. 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.4 章では, 提案手法であるミドルソフトタグのトピック素性を作成する方法を説明する.5 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.6 章では, 実験結果から考察を行う.7 章では, 本論文の結論を記す.

第2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. トピックモデルでは, 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)[1] や (Probabilistic Latent Semantic Indexing（PLSI）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.

2.1　Latent Dirichlet Allocation

Latent Dirichlet Allocation（LDA)[1] 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ことが特徴である. 基本的には, 文書が単語の分布によって特徴付けられる各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 のモデルでは, 幾つかの単純な仮定によってつくられている.

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 β $_{ij}=p(w^j=1|z^i=1)$ 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.

$$p(\theta \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1} \tag{2.1}$$

ただし, パラメータ α は,k 次元のベクトルであり書く成分は $\alpha_i > 0$ 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.

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \mathbf{p}(\theta \mid \alpha) \prod_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{p}(\mathbf{z_n} \mid \theta) \mathbf{p}(\mathbf{w_n} \mid \mathbf{z_n}, \beta) \tag{2.2}$$

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.

$$p(\mathbf{w} \mid \alpha, \beta) = \int \mathbf{p}(\theta \mid \alpha) (\prod_{\mathbf{n=1}}^{\mathbf{N}} \sum_{\mathbf{z_n}} \mathbf{p}(\mathbf{z_n} \mid \theta) \mathbf{p}(\mathbf{w_n} \mid \mathbf{z_n}, \beta)) \mathbf{d}\theta \tag{2.3}$$

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.

$$p(\mathbf{D} \mid \alpha, \beta) = \prod_{\mathbf{d=1}}^{\mathbf{M}} \int \mathbf{p}(\theta_\mathbf{d} \mid \alpha) (\prod_{\mathbf{n=1}}^{\mathbf{N_d}} \sum_{\mathbf{z_{dn}}} \mathbf{p}(\mathbf{z_{dn}} \mid \theta) \mathbf{P}(\mathbf{w_{dn}} \mid \mathbf{z_{dn}}, \beta)) \mathbf{d}\theta_\mathbf{d} \tag{2.4}$$

LDA のグラフィカルモデルを図 2.1 に示す. この図が示している通り LDA

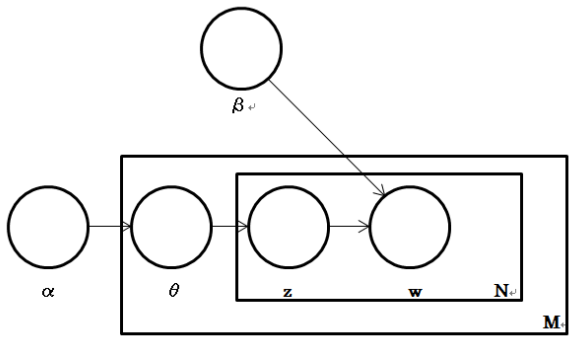

図 2.1: LDA のグラフィカルモデル

は, 3 つの階層からなっている. 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.

2.2 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.

2.2.1 推論

LDA を使用するために解く必要がある主な推論の問題は, 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.

$$p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{\mathbf{p}(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{\mathbf{p}(\mathbf{z} \mid \alpha, \beta)} \tag{2.5}$$

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.

$$p(\mathbf{w} \mid \alpha, \beta) = \frac{\mathbf{\Gamma}(\sum_{\mathbf{i}} \alpha_{\mathbf{i}})}{\prod_{\mathbf{i}} \mathbf{\Gamma}(\alpha_{\mathbf{i}})} \int (\prod_{\mathbf{i=1}}^{\mathbf{k}} \theta_{\mathbf{i}}^{\alpha_{\mathbf{i}}-\mathbf{1}})(\prod_{\mathbf{n=1}}^{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \prod_{\mathbf{j=1}}^{\mathbf{V}} (\theta_{\mathbf{i}} \beta_{\mathbf{ij}})^{\mathbf{w_n^j}}) \mathbf{d}\theta \tag{2.6}$$

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.

2.2.2 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.

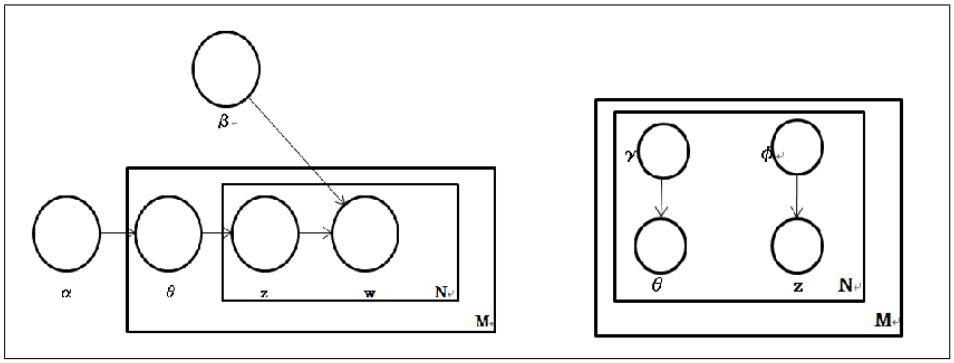

図 2.2: (左図)LDAのグラフィカルモデル.(右図) 変分ベイズを使用したグラフィカルモデル)

問題のある θ と z の組は, 図2.2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.

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = \mathbf{q}(\theta \mid \gamma) \prod_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{q}(\mathbf{z_n} \mid \phi_\mathbf{n}) \tag{2.7}$$

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.

$$(\gamma^*, \phi^*) = argmin_{(\gamma,\phi)} D(q(\theta, \mathbf{z} \mid \gamma, \phi) \parallel \mathbf{p}(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)) \tag{2.8}$$

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 (2.8) の γ と ϕ の更新式は以下で表される.

$$\theta_{ni} \propto \beta_{iw_n} exp(E_q[log(\theta_i) \mid \gamma]) \tag{2.9}$$

$$\gamma_i = \alpha_i + \sum_{n-1}^{N} \phi_{ni} \tag{2.10}$$

式 (2.9) の期待値は以下で定義される.

$$E_q[log(\theta_i) \mid \gamma] = \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \tag{2.11}$$

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 (2.9) と式 (2.10) 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 $\mathbf{w}$ 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 $\phi_{ni}^0 := 1/k$ 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) $\phi_{ni}^{t+1} := \beta_{iw_n} \exp(\Psi(\gamma_i^t))$
(7) normalize ϕ_n^{t+1} to sum to 1.
(8) $\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi_n^{t+1}$
(9) **until** convergence

図 2.3: 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 2.3 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.

2.2.3 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.

$$l(\alpha, \beta) = \sum_{d-1}^{M} logp(\mathbf{w_d} \mid \alpha, \beta) \tag{2.12}$$

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.

$$\beta_{ij} = \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni}^* w_{dn\cdot}^j \tag{2.13}$$

第3章　トピックモデルを利用した WSD

この章では, 従来手法のトピックモデルを利用した WSD について述べる. トピックモデルを利用した WSD を説明するために, まず WSD と WSD の一般的な手法について説明を行い, 次に WSD に一般的に使用されるサポートベクトルマシン [4] について説明する. そして最後に, トピックモデルを利用した WSD について述べる.

3.1　WSD

3.1.1　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 3.1 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 3.1: 「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

3.1.2 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 3.1 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.

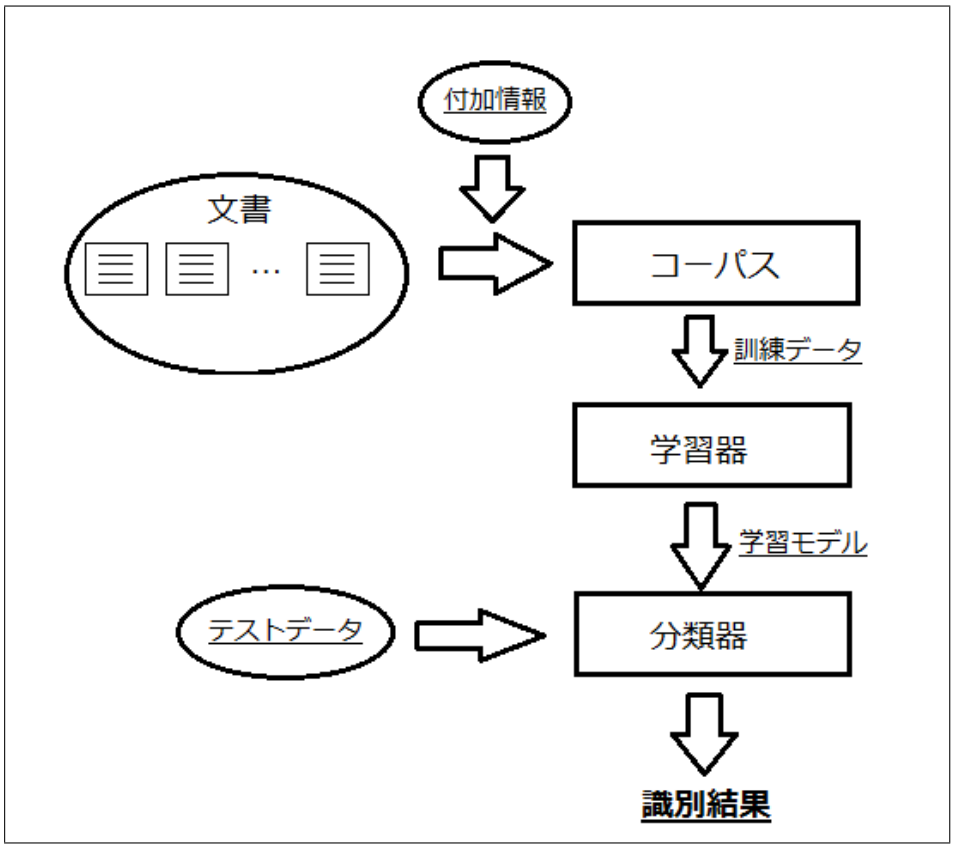

図 3.1: WSD における教師あり学習の流れ

3.2 サポートベクトルマシン

3.2.1 概要

サポートベクトルマシン (Support Vector Machine,SVM) 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よばれ, 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.

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(|D|)}, y^{(|D|)})\} \tag{3.1}$$

$x^{(1)}, x^{(2)}, \cdots, x^{(|D|)}$ 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せられる.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - \mathbf{b} \tag{3.2}$$

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.

3.2.2 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 3.2 のように訓練データが分布していると仮定する.

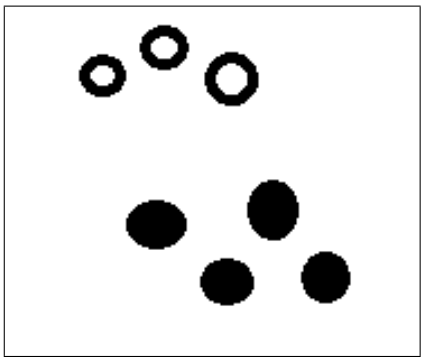

図 3.2: 訓練データ

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 3.2 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 $|\ \mathbf{x}_+ - \mathbf{x}_*\ |$ と表せられる.$\mathbf{w}$ と $\mathbf{x}_+ - \mathbf{x}_*$ は同じ方向を向いているので,$\mathbf{w} \cdot (\mathbf{x}_+ - \mathbf{x}_*) = |\ \mathbf{w}\ ||\ \mathbf{x}_+ - \mathbf{x}_*\ |$ 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,

$$\mid \mathbf{x}_+ - \mathbf{x}_* \mid = \frac{\mathbf{1}}{\mid \mathbf{w} \mid} \tag{3.3}$$

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, k の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.

3.2.3 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.

$$y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}} - \mathbf{b}) \geq \mathbf{1} \tag{3.4}$$

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.

$$(min.)\frac{1}{2}\mathbf{w^2} \tag{3.5}$$

$$(s.t.)y^{(i)}(\mathbf{w}\cdot\mathbf{x^{(i)}} - \mathbf{b}) - \mathbf{1} \geq \mathbf{0}; \forall \mathbf{i}. \tag{3.6}$$

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.

$$L(\mathbf{w}, \mathbf{b}, \alpha) = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{w^2} - \sum_{\mathbf{i}} \alpha_{\mathbf{i}}(\mathbf{y^{(i)}}(\mathbf{w}\cdot\mathbf{x^{(x)}} - \mathbf{b})) \tag{3.7}$$

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.

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \mathbf{b}, \alpha) = \mathbf{w} - \sum_{\mathbf{i}} \alpha_{\mathbf{i}}\mathbf{y^{(i)}}\mathbf{x^{(i)}}, \frac{\partial \mathbf{L}(\mathbf{w}, \mathbf{b}, \alpha)}{\partial \mathbf{b}} = \sum_{\mathbf{i}} \alpha_{\mathbf{i}}\mathbf{y^{(i)}} \tag{3.8}$$

これらを 0 と置くと以下が得られる.

$$\mathbf{w}^* = \sum_{\mathbf{i}} \alpha_{\mathbf{(i)}}\mathbf{y^{(i)}}\mathbf{x^{(i)}} \tag{3.9}$$

$$\sum_i \alpha_i y^{(i)} = 0 \tag{3.10}$$

一つ目の等式（3.10）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 $\mathbf{w}^* = \sum_{\mathbf{i}} \alpha_{(\mathbf{i})} \mathbf{y}^{(\mathbf{i})} \mathbf{x}^{(\mathbf{i})}$ を分離平面の式 $\mathbf{w} \cdot \mathbf{x} = \mathbf{b}$ に代入すると

$$f(\mathbf{x}) = \sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{y}^{(\mathbf{i})} \mathbf{x}^{(\mathbf{i})} \cdot \mathbf{x} - \mathbf{b} \tag{3.11}$$

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,

$$L(\mathbf{w}^*, \mathbf{b}, \alpha) = \frac{\mathbf{1}}{\mathbf{2}} \sum_{\mathbf{i,j}} \alpha_{\mathbf{i}} \alpha_{\mathbf{j}} \mathbf{y}^{(\mathbf{i})} \mathbf{y}^{(\mathbf{j})} \mathbf{x}^{(\mathbf{i})} \cdot \mathbf{x}^{(\mathbf{j})} - \sum_{\mathbf{i}} \alpha_{\mathbf{i}} \tag{3.12}$$

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.

3.2.4 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 $y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(\mathbf{i})} - \mathbf{b}) - \mathbf{1} \geq \mathbf{0}$ 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

$$y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(\mathbf{i})} - \mathbf{b}) - \mathbf{1} \geq -\xi_{\mathbf{i}} \tag{3.13}$$

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.

$$(\min.) \frac{1}{2} \mathbf{w^2} + \mathbf{C} \sum_{\mathbf{i}} \xi_{\mathbf{i}} \tag{3.14}$$

$$\mathrm{s.t} y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(\mathbf{i})} - \mathbf{b}) \geq \mathbf{1} - \xi_{\mathbf{i}}; \forall \mathbf{i}., \xi_{\mathbf{i}} \geq \mathbf{0}; \forall \mathbf{i}. \tag{3.15}$$

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.

$$L(\mathbf{w},\mathbf{b},\xi,\alpha,\beta)=\frac{\mathbf{1}}{\mathbf{2}}\mathbf{w^2}+\mathbf{C}\sum_{\mathbf{i}}\xi_{\mathbf{i}}-\sum_{\mathbf{i}}\alpha_{\mathbf{i}}(\mathbf{y^{(i)}}(\mathbf{w}\cdot\mathbf{x^{(x)}}-\mathbf{b})-\mathbf{1}+\xi_{\mathbf{i}})-\sum_{\mathbf{i}}\beta_{\mathbf{i}}\xi_{\mathbf{i}} \tag{3.16}$$

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,

$$\frac{\partial L(\mathbf{w},\mathbf{b},\xi,\alpha,\beta)}{\partial\xi_i}=C-\alpha_i-\beta_i \tag{3.17}$$

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.

$$L(\mathbf{w}^*,\mathbf{b},\alpha,\beta)=-\frac{\mathbf{1}}{\mathbf{2}}\sum_{\mathbf{i,j}}\alpha_{\mathbf{i}}\alpha_{\mathbf{j}}\mathbf{y^{(i)}}\mathbf{y^{(j)}}\mathbf{x^{(i)}}\cdot\mathbf{x^{(j)}}-\sum_{\mathbf{i}}\alpha_{\mathbf{i}} \tag{3.18}$$

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.

3.3 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

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4章　提案手法

4.1　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

4.2　ソフトミドルタグの生成方法

対象単語 w の用例 s の基本素性を b, トピック素性を t で表す.

$$\mathbf{b} = (\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_N})$$

$$\mathbf{t} = (\mathbf{t_1}, \mathbf{t_2}, \cdots, \mathbf{t_N})$$

$\mathbf{b}$ のみを使った識別が通常の WSD となる．また基本素性にトピック素性を結合した素性 $\mathrm{append}(\mathbf{b}; \mathbf{t})$ による識別がトピックモデルを直接的に利用した WSD である.

$$\mathrm{append}(\mathbf{b}, \mathbf{t}) = (\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_N}, \mathbf{t_1}, \mathbf{t_2}, \cdots, \mathbf{t_N})$$

ここで t_i はトピック z_i との関連度であり, LDA を利用して得られる $p(s \mid z_i)$ に対応する．t_i の値として $p(s \mid z_i)$ を直接用いるのがソフトタグである．そして $\hat{i} = \arg\max_i p(s \mid z_i)$ のとき

$$t_i = \begin{cases} 1 & (i = \hat{i}) \\ 0 & (i \neq \hat{i}) \end{cases}$$

としたものがハードタグである.

LDA におけるトピックとは単語のクラスタであるため，トピックとは概念に相当する．そう考えた場合，単語は通常複数の意味があるので，1 つの概念に対応させるハードタグは現実的ではない．しかし単語をすべての概念と何らかの関連があると考えるのも妥当ではない．そこで本論文ではハードタグとソフトタグの中間にあたるミドルソフトタグを提案する．ミドルソフトタグとは t_i の値として $p(s \mid z_i)$ がある閾値よりも小さい場合に 0 にし，そうでなければ $p(s \mid z_i)$ とする, そして最後に得られたベクトルの大きさを 1 に正規化するとするものである．閾値の設定が問題であるが，ここでは 0.02 として実験を行った.

第5章　実験

5.1　実験に使用するデータ

データセットにはSemEval-2のJapanese WSD[3]タスクを使用する. SemEval-2:JapaneseWSDタスクは, 対象コーパスの分野が多岐に渡る特徴がある. 訓練データには, 白書, 新聞, 本や雑誌の分野からなり, 評価データは, 更に,Web上のサイトであるYahoo知恵袋のデータも含んでいる.

本データには, 岩波国語辞典の語義をもとに, 語義IDが付与されている. 岩波国語辞典に定義されていない新語義も付与されている.WSDに設定されている対象単語は50単語であり, 辞典に定義されいる語義数は219である. また, 各々の単語に対して50件のテストデータと約50件の訓練データが存在する.

表5.1は, 本タスクで配布された岩波国語辞典の例である.

表 5.1: 単語「あげる」の岩波国語辞典の例

Headword とりあげる【取（り）上げる】
37562-0-0-0-0 ((下一他))
37562-0-0-1-0 < 1 > 手より下にある物を手に取る.「受話器を—」
37562-0-0-2-0 < 2 > 産婆などが, 手助けをして子を生ませる.
37562-0-0-3-0 < 3 >（下の者からの）申請・意見・具申などを受け付ける, 採用する. また, 無視せずに, 問題として取り扱う.「—に足りない事」
37562-0-0-4-0 < 4 > 相手の持っている物をうばい取る.「刃物を—」. 財産などを没収する. 税金などを徴収する.「領地を—」

図5.1は訓練データの例である. ここで,sense=””で示されている部分が, 付与されている語義IDを示している. 語義IDが”X”なのは新語義を表している.

5.2　実験の流れと使用するツール

まず,SVMにおける識別では, 各対象単語の訓練データとテストデータにおける素性ベクトルを使用する.SVMの実行時には, カーネルとして線形カーネ

```
<sentence>
<mor pos="名詞-固有名詞-組織名" rd="デンツー">電通</mor>
<mor pos="補助記号-読点" rd="," ">、</mor>
<mor pos="名詞-固有名詞-組織名" rd="ハクホー">博報</mor>
<mor pos="接尾辞-名詞的-一般" rd="ドー">堂</mor>
<mor pos="助詞-格助詞" rd="オ">を</mor>
<mor pos="名詞-普通名詞-副詞可能" rd="ハジメ">はじめ</mor>
<mor pos="名詞-普通名詞-一般" rd="ジョーイ">上位</mor>
<mor pos="名詞-数詞" rd="ゴ">五</mor>
<mor pos="接尾辞-名詞的-助数詞" rd="シャ">社</mor>
<mor pos="助詞-副助詞" rd="クライ">くらい</mor>
<mor pos="助動詞" rd="ナラ" bfm="ダ">なら</mor>
<mor pos="名詞-普通名詞-一般" rd="エイチピー">ＨＰ</mor>
<mor pos="助詞-格助詞" rd="オ">を</mor>
<mor pos="動詞-一般" rd="ツクル" bfm="ツクル" >作る</mor>
<mor pos="形状詞-一般" rd="ジンテキ">人的</mor>
<mor pos="名詞-普通名詞-一般" rd="ケーザイ" sense="X">経済</mor>
<mor pos="接尾辞-形状詞的" rd="テキ">的</mor>
<mor pos="名詞-普通名詞-一般" rd="ヨユー">余裕</mor>
<mor pos="助詞-係助詞" rd="モ">も</mor>
<mor pos="動詞-非自立可能" rd="アル" bfm="アル" >ある</mor>
<mor pos="助動詞" rd="デショー" bfm="デス">でしょう</mor>
<mor pos="助詞-接続助詞" rd="ガ">が</mor>
<mor pos="補助記号-読点" rd="," ">、</mor>
<mor pos="名詞-普通名詞-一般" rd="チューショー">中小</mor>
<mor pos="助詞-格助詞" rd="ノ">の</mor>
<mor pos="名詞-普通名詞-サ変可能" rd="ダイリ">代理</mor>
<mor pos="接尾辞-名詞的-一般" rd="テン">店</mor>
<mor pos="助詞-格助詞" rd="デ">で</mor>
<mor pos="助詞-係助詞" rd="ワ">は</mor>
<mor pos="連体詞" rd="ソンナ">そんな</mor>
<mor pos="名詞-普通名詞-一般" rd="ヨユー">余裕</mor>
<mor pos="助詞-係助詞" rd="ワ">は</mor>
<mor pos="動詞-非自立可能" rd="アリ" bfm="アル" >あり</mor>
<mor pos="助動詞" rd="マセ" bfm="マス">ませ</mor>
<mor pos="助動詞" rd="ン" bfm="ヌ">ん</mor>
<mor pos="補助記号-句点" rd="." ">。</mor>
</sentence>
```

図 5.1: 訓練データの例

ルを用いた.SVM は,LIVSVM[1]を使用した.

次に, トピックモデルである LDA は,WSD で与えられた文書, 合計 1421 文書からモデルが作成される. 実行オプションとして, 潜在クラス数 K として,100 を指定した. そして, この学習したモデルから, 各対象単語におけるテストデータと訓練データの文 s に対してトピックベクトル $p(s \mid z_i)$ を求める. また,LDA を学習するために持橋大地氏が開発したツール[2]を使用する.

5.3 実験結果

表 5.2: 実験結果

	正解率
基本素性のみ	76.76
基本素性にハードタグのトピック素性を結合した素性	76.88
基本素性にソフトタグのトピック素性を結合した素性	76.96
提案手法 (閾値 0.02)	**77.00**

実験結果を 5.2 に示す. この実験によりトピック素性を利用することで WSD の精度を改善できることが示された. また, 提案したミドルソフトタグの効果が微少ながら確認できる. しかし, その正解率の上昇は大きくはなかった.

[1]http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[2]http://chasen.org/ daiti-m/dist/lda/

第6章　考察

本論文ではハードタグとソフトタグの中間にあたるミドルソフトタグを提案した．一般にソフトタグが最も情報量が多いために，理論的には，これを使うのが最良であるが，現実的には値の低い部分は誤差と見なせるものであり，逆に悪影響があると考えた．またトピックが概念に相当するものだとすれば，関連度の低いトピックには 0 を与えた方が適切だと考える．これらの考えをもとに実験を行い微少ながら効果を確認できた. しかし, 正解率の上昇は,0.04% のみに留まった.

6.1　閾値

実験では閾値を 0.02 に固定してミドルソフトタグを作成している. 閾値に応じて素性の効果は変化するので, トピック数を変化させた実験も行った. 実験結果を以下に示す.

表 6.1: 閾値とトピック数

閾値	0.01	0.02	0.03
正解率	76.96	77.00	76.88

閾値を変化すると正解率に影響をあたえることがわかる. 単純に大きければよいということでもない. 閾値が大きすぎると正解率が下がり, 小さすぎると基本素性のみで実験の結果と変わらないことがわかる．また, この閾値は, 各対象単語の各用例に対して一律である. そのため, 各対象単語毎, 各用例ごとに閾値を設定出来れば, さらなる精度の上昇が期待できると考える. 今後は, 適切な閾値を設定法を考察する予定である.

6.2　トピック数

トピックモデルを利用した WSD では利用するトピック数が精度に大きく影響する．一般にトピック数は大きい方がよいと考えられるが，一概に大きければ良いというものでもない．参考として，トピック数 K を 50 と 150 にした実験も行った．トピック数が 50 の場合，ソフトタグを利用した正解率は

76.88%，ハードタグを利用した正解率は 77.28%，そしてミドルソフトタグ (閾値 0.02) を利用した正解率は 76.92% となった．ソフトタグによるトピック素性を利用した場合とミドルソフトタグを利用した場合は, トピック数 100 のモデルよりも悪い値が出ているが, ハードタグによるトピック素性を利用した場合の正解率は 77.28%であり，これは SemEval-2 の日本語 WSD タスクではかなりの高い値である．また, トピック数 50 ではハードタグの方がソフトタグよりもかなり正解率が高く，Cai [2] の実験とは逆の結果が得られている．つまりトピック数は有効なタグのタイプにも影響していると考えられる．

トピック数が 150 の場合，ソフトタグを利用した正解率は 76.80%，ハードタグを利用した正解率は 76.84%，そしてミドルソフトタグ (閾値 0.02) を利用した正解率は 76.92% となった．

以上の結果より, ミドルソフトタグを利用した正解率は, どのトピック数でもソフトタグを利用した正解率を上回っている事がわかる．

6.3 素性分離型アンサンブル学習

最後にトピック素性を直接利用するアプローチとしては，単純ではあるが, 基本素性にトピック素性を結合するアプローチが有効であることを述べておきたい．トピック素性を直接利用するアプローチとしては，トピック素性を分離し，トピック素性のみからの識別の結果と基本素性のみからの識別の結果をアンサンブルするアプローチも考えられる [4]．実験結果は割愛するが, トピック素性のみからの識別の結果 (68.92%) が最大頻度を出力する識別の結果 (68.96%) よりも悪く，どのようにアンサンブルさせても良い正解率は得られなかった．この点からトピック素性を直接利用するのは，単純ではあるが，基本素性にトピック素性を結合するというアプローチが有効である．その場合，本論文で提案したミドルソフトタグにより更なる精度向上が可能だと考えている．

第7章　おわりに

本論文ではWSDにトピックモデルを利用する手法を提案した．トピックモデルを直接WSDに利用するには，基本素性にトピック素性を結合させる方式が，単純ではあるが，効果が高い．ただしその際にトピック素性をハードタグで表現するか，ソフトタグで表現するかのオプションがある．ここではそれらの中間に位置するミドルソフトタグを提案した．SemEval-2の日本語WSDタスクを用いた実験では，提案手法の効果は微少ながら確認できた．そしてミドルソフトタグは閾値，割り当てる数値およびトピック数が精度に関連しており，今後適切な設定を行い, さらなる効果を示していきたい.

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] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. Machine Learning Reseach, Vol. 3, pp. 993-1022, 2003.

[2] Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. Improving Word Sense Disambiguation using Topic Features. In the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLPCoNLL), pp. 1015-1023, 2007.

[3] Manabu Okumura, Kiyoaki Shirai, Kanako Komiya, and Hikaru Yokono. SemEval-2010 Task: Japanese WSD. In The 5th International Workshop on Semantic Evaluation, pp. 69-74, 2010.

[4] JMing wei Chang, Wen tau Yih, and Christopher Meek. Partitioned logistic regression for spam filtering. In the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD ' 08, pp. 97-105, 2008.

[5] 高村大地・奥村学（2010）『言語処理のための機械学習入門』コロナ社

付録A ソースリスト

ソースコード 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 my $a;
18
19 while (<IN>) {#読み込みファイルから一行読み込み
20     # 改行コードの除去
21     chomp ($_);
22     my $line=$_;
23
24     # 各行を半角空欄区切りで分割
25       $line=~s/　/ /g;
26     @data = split(/ /, $line);
27     my $j=0;
28     for ($j=0; $j<@data; $j++) {
29             $data[$j] =~ s/( )+//g;
```

```
30              chomp ($data[$j]);
31      }
32
33
34
35
36
37      for(my $k=0;$k<@data;$k++){
38          my $b=@data -1;
39              if($data[$k]=~m/e0=(.+)/){
40                  $hash{$data[$k]}++;
41
42              }
43
44
45      elsif ($data[$k]=~m/e1=(.+)/){
46                  $a=$k+1;
47
48                      $hash{$data[$k]}++;
49
50
51
52          }
53          elsif ($data[$k]=~m/e2=(.+)/){
54                  $a=$k+1;
55
56                      $hash{$data[$k]}++;
57
58
59
60          }
61  elsif ($data[$k]=~m/e3=(.+)/){
62                  $a=$k+1;
63
64                      $hash{$data[$k]}++;
65
66
67
```

```
68          }
69          elsif ($data[$k]=~m/e4=(.+)/){
70              $a = $k+1;
71
72                  $hash{$data[$k]}++;
73
74
75
76          }
77  elsif ($data[$k]=~m/e5=(.+)/){
78                  $a=$k+1;
79
80                  $hash{$data[$k]}++;
81
82
83
84          }
85          elsif ($data[$k]=~m/e6=(.+)/){
86              $hash{$data[$k]}++;
87
88
89          }
90  elsif ($data[$k]=~m/e7=(.+)/){
91              $hash{$data[$k]}++;
92
93
94          }
95
96
97
98      }
99
100 }
101 foreach $key (keys %hash) {
102     print "$key $hash{$key}\n";
103 }
104
105
```

```
106
107     ##終了処理
108 close (IN);
```

ソースコード 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";
11 my $file_output="seikai_label.txt";
12 open (IN, $file_input) or die "$!";
13 open (OUT, ">$file_output") or die "$!";
14
15
16 my @data;
17 my $a;
18
19 while (<IN>) {#読み込みファイルから一行読み込み
20     # 改行コードの除去
21     chomp ($_);
22     my $line=$_;
23
24     # 各行を半角空欄区切りで分割
25      $line=~s/　/ /g;
26     @data = split(/ /, $line);
27     my $j=0;
28     for ($j=0; $j<@data; $j++) {
29             $data[$j] =~ s/( )+//g;
30             chomp ($data[$j]);
31     }
32
```

```
33
34
35
36      my $k=@data-1;
37      if($data[$k]=~m/(\d+)-(\d)-(\d)-(\d)/){
38 my $s1=$1; my $s2=$2; my $s3=$3; my $s4=$4;
39 my $s="$s3"."$s4";
40         print OUT "$s\n";
41
42         }else {print OUT "999\n";}
43
44
45 }
46
47
48
49      ##終了処理
50 close (IN);
51 close (OUT);
```

ソースコード 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 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 $N=2;#クラスタ数
10
11  my $file1="seikai_label.txt";
12  my $file2="train.dat";
13
14  open (IN1,"$file1") or die "$!";
15  open (OUT,">$file2") or die "$!";
```

```
16
17 my @data1;
18 my $i=1;
19 while(<IN1>){
20       chomp($_);
21       $data1[$i]=$_;
22       $i++;
23 }
24
25
26 for(my $j=1;$j<=50;$j++){
27
28       my $file3=$j.".tf";
29       open(IN2,"$file3") or die "$!";
30
31       my $x=0;
32       my @DATA1=();
33       my @DATA2=();
34       my @DATA=(\@DATA1,\@DATA2);
35
36       while(<IN2>){
37            chomp($_);
38            my @dat = split(/\t/, $_);
39
40            for (my $j=0; $j<$#dat+1; $j++) {
41                        $dat[$j] =~ s/(  | )+//g;
42            }
43
44            $DATA[$x][0]=$dat[0];
45            $DATA[$x][1]=$dat[1];
46            $x++;
47
48       }
49
50       for(my $a=0;$a<$#DATA+1;$a++){##データを次元でソート
51            for(my $b=$a+1;$b<$#DATA+1;$b++){
52                 if($DATA[$a][0] > $DATA[$b][0]){
53                      my $temp=$DATA[$a][0];
```

```
54                      $DATA[$a][0]=$DATA[$b][0];
55                      $DATA[$b][0]=$temp;
56
57                      $temp=$DATA[$a][1];
58                      $DATA[$a][1]=$DATA[$b][1];
59                      $DATA[$b][1]=$temp;
60
61                  }
62              }
63          }
64          print OUT "$data1[$j] ";
65          for(my $i=0;$i<$#DATA+1;$i++){
66              print OUT "$DATA[$i][0]:1 ";
67          }
68          print OUT "\n";
69
70
71
72  }
```

ソースコード 5: 素性から e6 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
58  }
59  foreach $key (keys %hash) {
60      print "$key $hash{$key}\n";
61  }
62
63
64
65      ##終了処理
66  close (IN);
```

ソースコード 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;
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 7: トピックベクトルを正規化するプログラム

```
1   #!/usr/bin/perl
2
3 use strict;
4 use warnings;
5 use Encode;
6 use encoding "shift-jis";
7
8 my $file_input="$ARGV[0]";
9 my $topic=100;
10 my @kakuritu;
11 my $all;
12 open (IN, $file_input) or die "$!";
13
14 while(<IN>){
15     chomp($_);
16     my @data = split(/ /, $_);
17     for (my $j=0; $j<@data; $j++) {
18             $data[$j] =~ s/( | )+//g;
19     chomp ($data[$j]);
20     }
21     for(my $m=0;$m<$topic;$m++){
22
23         $kakuritu[$m]+=$data[$m];
24
25     }
```

```
26
27 }
28
29
30
31
32
33
34 for(my $n=0;$n<$topic;$n++){
35
36     #print sprintf("%e\n",$kakuritu[$n]);
37     $all+=$kakuritu[$n];
38 }
39
40
41 for(my $n=0;$n<$topic;$n++){
42
43     #print sprintf("%e\n",$kakuritu[$n]);
44 if($all!=0.0){
45     $kakuritu[$n]=$kakuritu[$n]/$all;
46 }else{$kakuritu[$n]=0.0}
47 }
48
49 for(my $n=0;$n<$topic;$n++){
50
51     print sprintf("%e ",$kakuritu[$n]);
52
53 }
54
55 print "\n";
```

ソースコード 8: 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";
10 my $file2="kihon_topic_label";
11 my $file3="attend_train.dat";
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++){
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  }
```

ソースコード 9: 提案手法の素性を作成するプログラム

```
1   #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5   use Encode;
6   use encoding "shift-jis";
7
8   my $file1="seikai_label.txt";
9   my $INDEX_line="$ARGV[0]";
10  $INDEX_line=$INDEX_line+1;
11  my @data1;
12  my $topic=100;
13  my $si=0.02;
14  open (IN, $file1) or die "$!";
15
16  my $x=0;
17  while(<IN>){
18
19      chomp($_);
20      $data1[$x]=$_;
21      $x++;
22  }
23
24
25  close(IN);
26
27  my $out="kihon_topic_label";
28  my @w;
29  open (OUT, ">$out") or die "$!";
30
31  for(my $i =1;$i <= 50 ;$i++){
32
33      my $inputfile=$i.".seikika";
```

```
34      open (IN, $inputfile) or die "$!";
35
36      while (<IN>) {
37       my  @words = split(" ");
38 my $ind=-1;
39 my $max=0;
40       for (my $k=0;$k<$topic;$k++){
41 if($words[$k]>$max){$ind=$k;$max=$words[$k];}
42
43
44       }
45 for (my $k=0;$k<$topic;$k++){
46 if($words[$k]>=$si){
47           $w[$k]=$words[$k];
48 }else{$w[$k]=0;}
49       }
50 my $all=0;
51 for (my $k=0;$k<$topic;$k++){
52 $all=$all+$w[$k];
53       }
54
55 for (my $k=0;$k<$topic;$k++){
56 if($all>0){
57 $w[$k]=$w[$k]/$all;
58 }
59       }
60
61
62
63
64      }
65      close (IN);
66
67
68      my $a=-1+$i;
69      print OUT "$data1[$a] ";
70
71      for(my $m=0;$m<$topic;$m++){
```

```
72 my $asd=$m+$INDEX_line+1;
73 my $ppp=0;
74
75       print OUT "$asd:$w[$m] ";
76
77       }
78
79       print OUT "\n";
80
81 }
```