トピックモデルを用いた語義曖昧性解消

情報工学科

平成24年2月10日

執筆者：西野太樹 (08T4082A)
指導教員：新納浩幸 准教授

目 次

第 1 章　はじめに **4**
1.1　概要 4
1.2　本論の構成 4

第 2 章　語義曖昧性解消 **5**
2.1　概要 5
2.2　一般的な手法 5

第 3 章　サポートベクトルマシン **8**

第 4 章　トピックモデル **11**
4.1　pLSI 11
4.2　LDA 15

第 5 章　提案手法 **17**
5.1　素性ベクトルの作成 17
5.2　トピックベクトルを用いた手法 19

第 6 章　実験 **20**

第 7 章　考察 **22**

第 8 章　おわりに **23**

謝辞 **24**

参考文献 **25**

付 録 A　ソースリスト **26**

表 目 次

2.1 「与える」の語義 5
6.1 各手法による正解率 21

図 目 次

2.1 教師あり学習による WSD の流れ 7

3.1 SVM のマージン最大化 10

4.1 pLSI のアスペクトモデル 16
4.2 LDA のグラフィカルモデル 16

5.1 対象単語「経済」の例文 18

第1章　はじめに

1.1　概要

語義曖昧性解消 (WSD: Word Sense Disambiguation) は、文章中の複数の語義を持つ多義語の正しい語義を機械的に判断して割り当てる、自然言語処理の作業の１つである。機械翻訳や情報抽出などのアプリケーションにとって、WSD は欠かすことのできない前処理となる。

現在の WSD は素性ベクトルを利用した手法が一般的である。過去の新聞記事や書籍などを大量に集めた文書群（コーパス）から学習し、その規則をもとに対象単語の語義を識別する。だが、その曖昧性解消の精度は向上の余地がある。

本研究では、WSD の精度向上のためにトピックモデルを用いた手法を提案する。従来より用いられる対象語を含む文の対象語の周辺から得ることができる素性ベクトルに加えて、文書生成の確率モデルであるトピックモデルより導きだされた対象語を含む文書のトピックベクトルを加える。トピックモデルを WSD に用いることによって対象単語を含む文書の特徴を捉え、語義の識別にも効果をもたらす。本稿ではその手法と理論について紹介し、従来法との比較を行い考察する。

1.2　本論の構成

第 2 章では語義曖昧性解消について一般的な手法と共に述べる。第 3 章では学習、識別に使用するサポートベクトルマシンについて述べる。第 4 章では本手法の核となるトピックモデルの代表的な方法として、pLSI と LDA について述べる。第 5 章では提案する手法について紹介する。第 6 章では実験の設定方法、使用するツールとその結果について述べる。第 7 章では実験結果から関連研究を絡めて考察を行う。第 8 章では結論と本実験の今後の展望について述べる。

第2章　語義曖昧性解消

2.1　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[1]。WSD はこのような多義語を対象として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)　自分の所有物を他の人に渡して、その人の物とする。現在ではやや改まった言い方で、恩恵的な意味で目下の者に授ける場合に多く用いる。「子供におやつを―・える」「賞を―・える」 (2)　相手のためになるものを提供する。「援助を―・える」「注意を―・える」 (3)　ある人の判断で人に何かをさせる。 ア）相手に何かができるようにしてやる。配慮して利用することを認める。「発言の自由を―・える」「口実を―・える」 イ）割り当てる。課する。「宿題を―・える」「役割を―・える」 (4)　影響を及ぼす。 ア）相手に、ある気持ち・感じなどをもたせる。「感銘を―・える」「いい印象を―・える」「苦痛を―・える」 イ）こうむらせる。「損害を―・える」

2.2　一般的な手法

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出力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、正解となる出力データを含めず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。

[1]デジタル大辞泉

学習には文書を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に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を要する。学習データとなるものには、文章中の単語に品詞を付けた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。これらをパラメータ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。本研究では学習や分類にサポートベクトルマシンを用いるが、その説明は次章で行う。

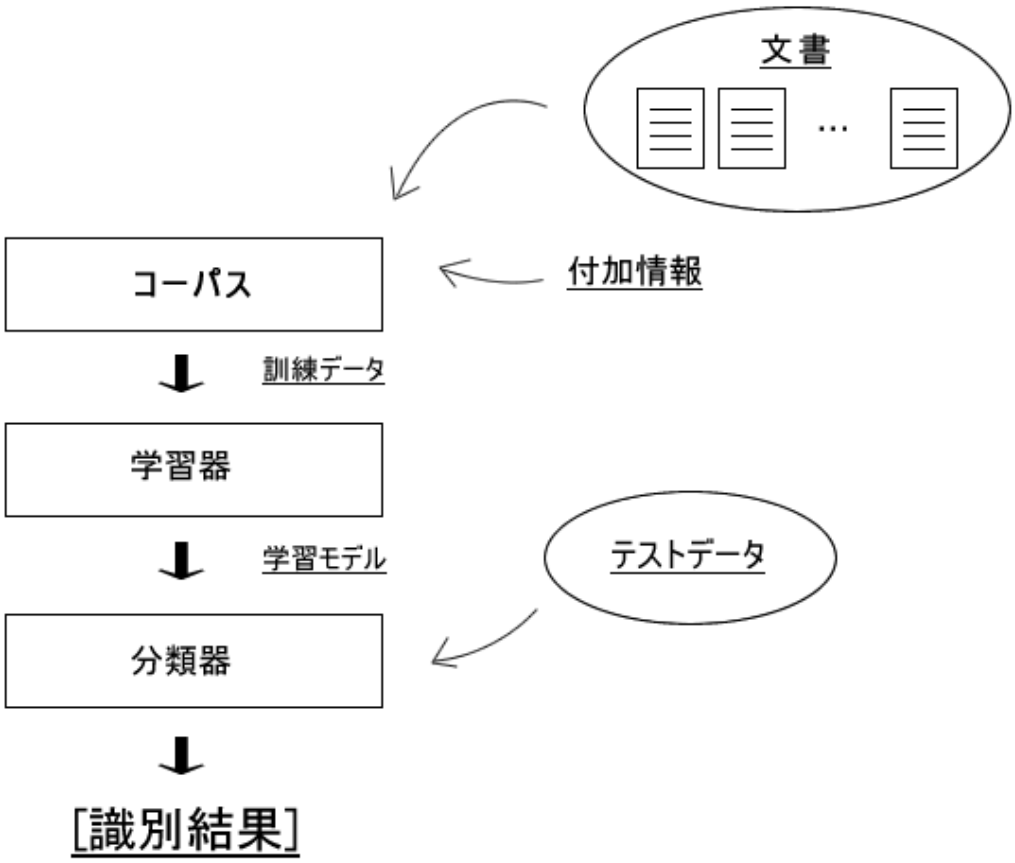

図 2.1: 教師あり学習による WSD の流れ

第3章　サポートベクトルマシン

サポートベクトルマシン (SVM) はカーネル法の一つで、WSD においてモデルの学習や分類器のために使用される。分類のために識別平面を最も近い各クラスのデータ間の距離を最大化するという基準で選択する。

まず 2 値の線形識別問題を考える。入力するデータを $X \in \mathbb{R}^m$、$Y \in \{+1, -1\}$ として、線形識別器

$$f(x) = \mathrm{sgn}(w^T x + b)$$

により与えられた x のクラス y を推定する。

このとき入力するデータが線形識別可能であるとして、任意の $i = 1, \cdots, N$ に対し

$$\begin{cases} Y_i = 1 \text{ ならば} & w^T X_i + b \geq 0, \\ Y_i = -1 \text{ ならば} & w^T X_i + b < 0 \end{cases}$$

を仮定する。このとき、パラメータ (w,b) が上の条件を満足するため、基準が必要となる。そのために+1 クラスと-1 クラスの互いの最近点であるサポートベクトルの間の距離（マージン）を最大化する基準を選択する。(図 (3.1) 参照)

それぞれのクラスのサポートベクトルを

$$\begin{cases} \min(w^T X_i + b) = 1, & i : Y_i = +1, \\ \max(w^T X_i + b) = -1, & i : Y_i = -1 \end{cases}$$

のように仮定する。このデータ点を X_*^+、X_*^- とすると、$w^T(X_*^+ - X_*^-) = 2$ であるから 2 点間の距離は $\frac{2}{\|w\|}$ で与えられる。したがって、このマージンを最大化する問題を解くことになる。

これを、条件をまとめて最小化に変換することで以下の主問題から解く形にする。

$$\min_{w_i, b, \xi_i} \frac{1}{2} \sum_{i,j=1}^{N} w_i w_j K_{ij} + C \sum_{i=1}^{N} \xi_i \quad \text{subject to} \begin{cases} Y_i(\sum_{j=1}^{N} K_{ij} w_j + b) \geq 1 - \xi_i & (\forall_i) \\ \xi_i \geq 0 & (\forall_i) \end{cases}$$

これに対する双対問題を以下に示す

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j Y_i Y_j K_{ij} \quad \text{subject to} \begin{cases} 0 \leq \alpha_i \leq C & (\forall_i), \\ \sum_{i=1}^{N} \alpha_i Y_i = 0 \end{cases} \tag{3.1}$$

双対問題は Lagrange 双対関数の最大化問題として与えられる。主問題に対する双対関数は

$$g(\alpha, \beta) = \min_{w,b,\xi} L(w, b, \xi, \alpha, \beta) \tag{3.2}$$

このとき $\alpha_i \geq 0$、$\beta_i \geq 0$、

$$\begin{aligned} L(w,b,\xi,\alpha,\beta) &= \frac{1}{2}\sum_{i,j=1}^{N} w_i w_j K_{ij} + C\sum_{i=1}^{N}\xi_i \\ &\quad + \sum_{i=1}^{N}\alpha_i\left\{1 - Y_i\left(\sum_{j=1}^{N} w_j K_{ij} + b\right) - \xi_i\right\} + \sum_{i=1}^{N}\beta_i(-\xi_i) \end{aligned}$$

により定義される。ここで式 (3.2) は制約なしの最小化問題であるので、その最適解 (w^*, b^*, ξ^*) は微分を実行し

$$\begin{aligned} \nabla_w &: \sum_{j=1}^{N} K_{ij} w_j^* - \sum_{j=1}^{N}\alpha_j Y_j K_{ij} = 0 \quad (\forall_i), \\ \nabla_b &: \sum_{j=1}^{N}\alpha_j Y_j = 0, \\ \nabla_\xi &: C - \alpha_i - \beta_i = 0 \quad (\forall_i) \end{aligned}$$

を満たす。これらの関係式を用いると

$$\frac{1}{2}\sum_{i,j=1}^{N} w_i^* w_j^* K_{ij} = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j Y_i Y_j K_{ij},$$

$$\sum_{i=1}^{N}\alpha_i\left\{1 - Y_i\left(\sum_{j=1}^{N} K_{ij} w_j^* + b^*\right)\right\} = \sum_{i,j=1}^{N}\alpha_i - \sum_{i,j=1}^{N}\alpha_i\alpha_j Y_i Y_j K_{ij}$$

が得られ、$C - \alpha_i - \beta_i = 0$ と合わせると、Lagrange 相対関数は

$$g(\alpha,\beta) = L(w^*, \beta^*, \xi^*, \alpha, \beta) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j Y_i Y_j K_{ij}$$

と書きなおされる。また β_i を消去すると $\alpha_i \geq 0$、$\beta_i \geq 0$ の条件は $0 \leq \alpha_i \leq C$ $(\forall_i)$ と表される。以上により式 (3.1) を得る。

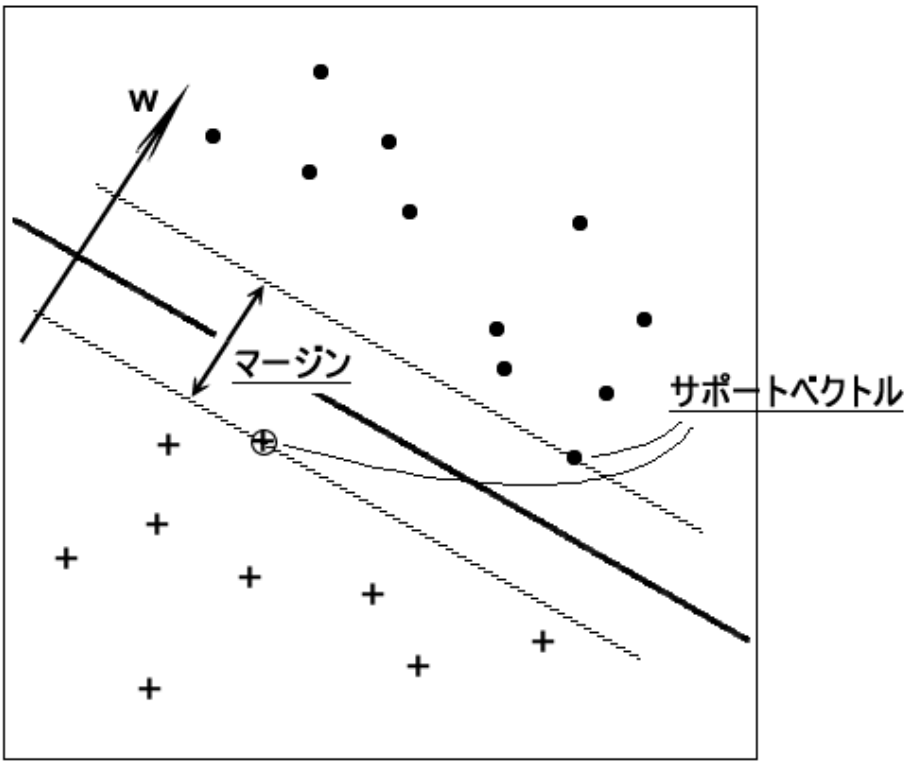

図 3.1: SVM のマージン最大化

第4章　トピックモデル

トピックモデルは、文書がいくつかの話題（トピック）によって生成されるものとして、その過程を確率的にモデル化したものである。代表的なものには、Probabilistic Latent Semantic Indexing(pLSI) や、Latent Dirichlet allocation(LDA) がある。

4.1　pLSI

pLSI は確率的なアプローチで行う、特異値分解による次元圧縮の手法である。処理を行う文書ベクトルは重み付けの必要がなく、単純に頻度を要素とするベクトルでよい。

pLSI では、アスペクトモデルと呼ばれる統計的なモデルを用いる。モデルとは文書と単語を結びつける潜在的なクラスを想定したモデルである。このアスペクトモデルは潜在的なクラス変数 z のための潜在変数モデルである。文書 d、単語 w、そして潜在クラス z を用いた同時確率モデルは以下で定義される。

$$p(w|d) = \sum_{z \in Z} p(w|z)p(z|d)$$

このとき、d と w は図 (4.1(a)) に示すとおり独立条件である。ベイズの定理から、

$$p(z|d) = \frac{p(z)p(d|z)}{p(d)}$$

以上から、$p(d, w) = p(d)p(w|d)$ なので、

$$P(d, w) = \sum_{z \in Z} P(z)P(d|z)P(w|z) \tag{4.1}$$

この時の各パラメータによるグラフィカルモデルは図 (4.1(b)) のようになる。

求めるものは $p(z)$、$p(w|z)$、$p(d|z)$ となる。

K 次元に次元縮約する場合、潜在的なクラスを K 個設定する。

$$z \in Z = \{z_1, \cdots, z_K\}$$

データ d に対して、

$$(p(z_1, d), p(z_2, d), \cdots, p(z_K, d))$$

が縮約されたベクトルとなる。

与えられた文書集合が $D = d_1, d_{2,}..., d_N$ であり、D で使われている単語の集合を $W = w_1, w_2, \cdots, w_M$ とすると、各 $k \in 1, 2, \cdots, K$ に対する $p(z_K)$、各 kk と $m \in 1, 2, \cdots, M$

に対する $p(w_m|z_k)$、各 k と $n \in 1, 2, \cdots, N$ に対する $p(d_n|z_k)$ が求まれば、$p(z)$、$p(w|z)$、$p(d|z)$ を求めることができる。

これらのパラメータは最尤法を用いて求める。今、文書 d に含まれている単語 w の個数を $n(d, w)$ で表すと、対数尤度関数 L は以下のようになる。

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log p(d_{i,} w_j)$$

(4.1) を使い、以下を得ることができる。

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log \left(\sum_{k=1}^{K} p(z_k) p(w_j|z_k) p(d_i|z_k) \right) \tag{4.2}$$

L を最大化するパラメータを求めるためには、EM アルゴリズムを用いる。ただし、尤度の関数に隠れ変数 z がすでに埋め込まれているので、Q 関数は直接求めることができる。

E-step では、z 以外のパラメータを固定した時の z の分布、$p(z|d, w)$ を求める。アスペクトモデルでは、

$$p(p, d, w) = p(z)p(w|z)p(d|z)$$

が仮定されているので、従って以下のようになる。

$$\begin{aligned} p(z_k|d, w) &= \frac{p(d, w, z_k)}{p(d, w)} \\ &= \frac{p(z_k)p(w|z_k)p(d|z_k)}{\sum_{k=1}^{K} p(z_k)p(w|z_k)p(d|z_k)} \end{aligned}$$

簡単のために、$p(z_k|d_i, w_j) = Q_{ijk}$ と置く。E-step ではこの Q_{ijk} を求めることになる。更新式の形で書くと以下のとおりになる。

$$Q_{ijk}^{(t+1)} = \frac{p(z_k)^{(t)} p(w_j|z_k)^{(t)} p(d_i|z_k)^{(t)}}{\sum_{k=1}^{K} p(z_k)^{(t)} p(w_j|z_k)^{(t)} p(d_i|z_k)^{(t)}}$$

次に M-step では z を固定した場合、つまり Q_{ijk} を固定した場合のその他のパラメータを求める。

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log \left(\sum_{k=1}^{K} p(z_k) p(w_j|z_k) p(d_i|z_k) \right) \tag{4.3}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log \left(\sum_{k=1}^{K} Q_{ijk} \frac{p(z_k) p(w_j|z_k) p(d_i|z_k)}{Q_{ijk}} \right) \tag{4.4}$$

$$\leq \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log \frac{p(z_k) p(w_j|z_k) p(d_i|z_k)}{Q_{ijk}} \tag{4.5}$$

最後の式の変形は、Jansen の不等式を用いる。

簡単のために、$a_k = p(z_k)p(w_j|z_k)p(d_i|z_k)$、$A = \sum_{k=1}^{K} a_k$ と置く。すると、$Q_{ijk} = a_k/A$ となる。また、確率 a_i/A で、値 a_iA/a_i をとるような確率変数 X を考える。今、関数 log が凸であることから Jensen の不等式を利用すると以下が成立する。

$$E[\log(X) \geq \log(E(X))] \tag{4.6}$$

式 (4.5) の左辺は以下のように変形できる。

$$\begin{aligned} E[\log(X)] &= \sum_{k=1}^{K} \frac{a_k}{A} \log\left(a_k \frac{A}{a_k}\right) \\ &= \sum_{k=1}^{K} Q_{ijk} \log\left(\frac{p(z_k)p(w_j|z_k)p(d_i|z_k)}{Q_{ijk}}\right) \end{aligned} \tag{4.7}$$

また式 (4.6) の右辺は以下のように変形できる。

$$\begin{aligned} \log E[(X) &= \log\left(\sum_{k=1}^{K} \frac{a_k}{A}\left(a_k \frac{A}{a_k}\right)\right) \\ &= \log\left(\sum_{k=1}^{K} Q_{ijk} \frac{p(z_k)p(w_j|z_k)p(d_i|z_k)}{Q_{ijk}}\right) \end{aligned}$$

以上より、式 (4.5) の不等号が成立する。また、式 (4.7) を変形すると、

$$\begin{aligned} E(\log(X)) &= \sum_{k=1}^{K} \frac{a_k}{A} \log\left(a_k \frac{A}{a_k}\right) \\ &= \sum_{k=1}^{K} \frac{a_k}{A} \log A \\ &= \log A \\ &= \log \sum_{k=1}^{K} a_k \\ &= \log\left(\sum_{k=1}^{K} p(z_k)p(w_j|z_k)p(d_i|z_k)\right) \end{aligned}$$

となるので、式 (4.5) 等号も成立する。

この式 (4.5) をさらに、変形して以下が成立する。

$$\begin{aligned} L &= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_i jk \log p(z_k)p(w_j|z_k)p(d_i|z_k) \\ &\quad - \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log Q_{ijk} \end{aligned}$$

上式の第 2 項は M-step では定数なので、L の最大化には関係ない。これを省いて以下を L とする。

$$L = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log p(z_k)p(w_j|z_k)p(d_i|z_k)$$

ここから、L の最大化は $\sum_{i=1}^{N} p(d_i|z_k) = 1$、$\sum_{j=1}^{M} p(w_j|z_k) = 1$、$\sum_{k=1}^{K} p(z_k) = 1$ の関係があるから、ラグランジュの未定乗数法を利用して解く。

$$L + \sum_{k=1}^{K} \alpha_k \left(1 - \sum_{i=1}^{N} p(d_i|z_k)\right) + \sum_{k=1}^{K} \beta_k \left(1 - \sum_{j=1}^{M} p(w_j|z_k)\right) + \gamma \left(1 - \sum_{k=1}^{K} p(z_k)\right)$$

$p(d_i|z_k) = u_i k$、$p(w_j|z_k) = v_{jk}$、$p(z_k) = w_k$ と置き、上記の式を $u_i k$、v_{jk}、w_k でそれぞれ偏微分し極値問題を解く。ここで Q_{ijk} は正確に書くと、$Q_{ijk}^{(t)}$ であり、M-step の時点で定数になる。

まず、u_{ik} の極値問題を解く。

$$\frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{u_{ik}} - \alpha_k = 0$$

よって、

$$u_{ik} = \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\alpha_k}$$

両辺を i に関して和をとると、

$$\sum_{i=1}^{N} u_{ik} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\alpha_k}$$

左辺は 1 なので、

$$\alpha_k = \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}$$

以上より、

$$\begin{aligned} u_{ik} &= p(d_i|z_k) \\ &= \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}} \end{aligned}$$

更新式の形で書くと、

$$p(d_i|z_k)^{(t)} = \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

となる。v_{jk}、w_k に関しても同様に以下で得ることができる。

$$p(w_j|z_k)^{(t)} = \frac{\sum_{i=1}^{N} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

$$p(z_k)^{(t)} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

実際に、EM アルゴリズムでパラメータを求める際には、初期値が必要となる。適当に設定する必要があるが、$p(z_k)^{(0)} = 1/K$ として、その他はランダムな値を与えることが多い。

4.2 LDA

LDA は pLSI を改良した、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大きな特徴である。基本的には、文書が単語分布によって特徴付けられる各潜在トピックをランダムに含むものとして表現されている。

コーパスの各文書は以下のプロセスで生成される。

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、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 $p(w_n|z_n,\beta$ より、単語 w_n を選択する。

文書集合を生成するグラフィカルモデルを図 (4.2) に示す。

はじめに、トピックの数 k は事前に既知であるとする。また単語確率 $p(w|z_n|\beta)$ は $k \times V$ の行列の $\beta_{ij} = p(w^j = 1|z^i = 1)$ によって、パラメータ化される。

k 次元のディリクレランダム変数 θ は、(k-1) シンプレックス内で値を取られ、このシンプレックス上で以下の確率密度を持つ。

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

このパラメータ α は $\alpha_i > 0$ の k 次元ベクトルであり、また $\Gamma(x)$ はガンマ関数である。

次にパラメータ α と β、ディリクレ事前分布 θ、z_n、w_n の生成確率は以下で与えられる。

$$p(\theta, \vec{z}, \vec{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

z を生成する θ を積分すると、文書の周辺分布を得ることができる。

$$p(\vec{w}|\alpha, \beta) = \int p(\theta|\alpha) \Big(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \Big) d\theta$$

最後に、一つの文書の周辺確率の積分を取り、コーパスの生成確率を得る。

$$p(D|\alpha, \beta) = \Gamma_{d=1}^{M} \int p(\theta_d|\alpha) \Big(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \Big) d\theta_d$$

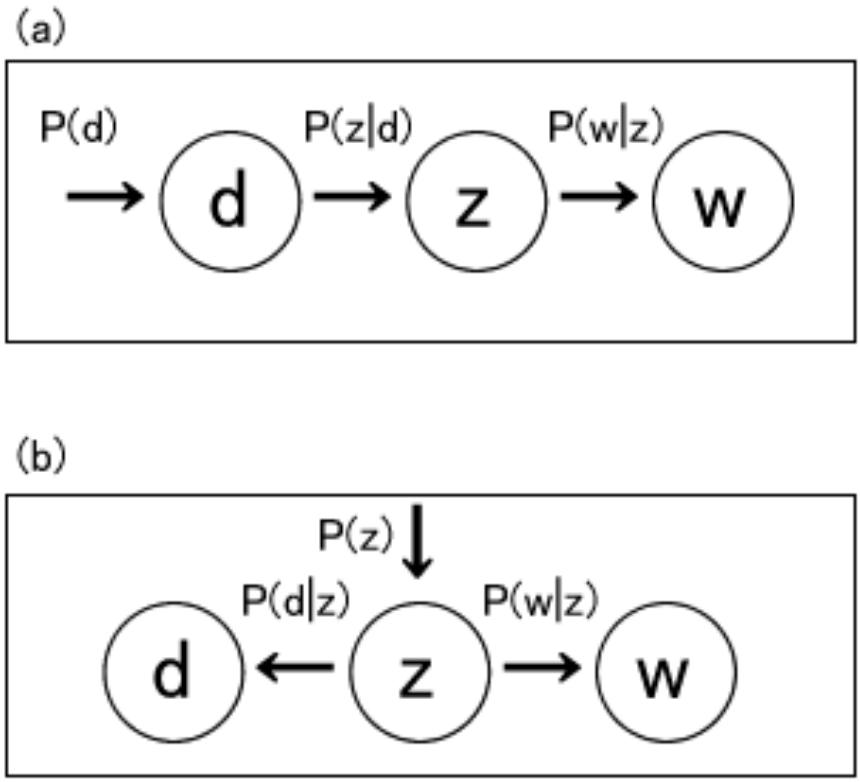

図 4.1: pLSI のアスペクトモデル

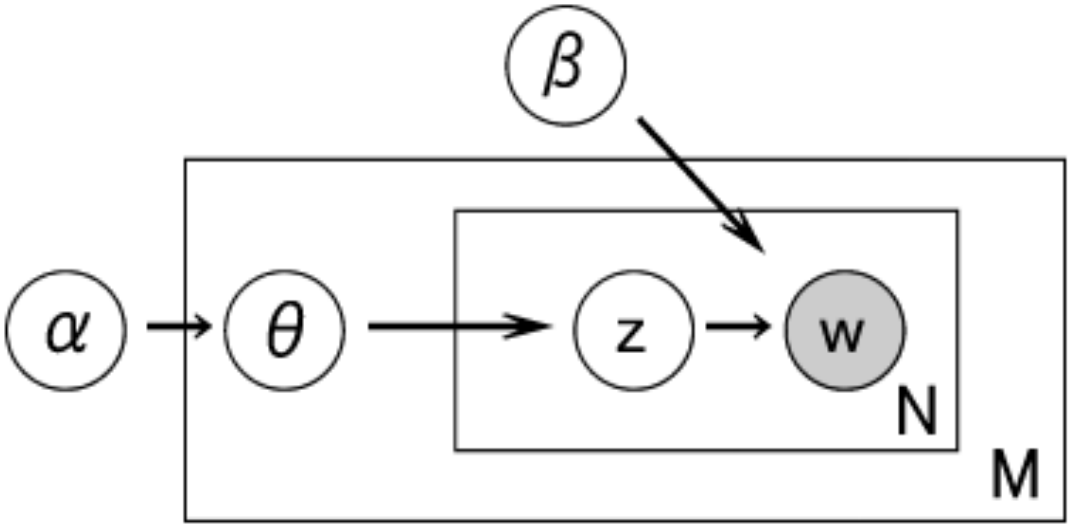

図 4.2: LDA のグラフィカルモデル

第5章　提案手法

本研究で提案する手法は従来法に加えてトピックモデルを用いる。まずは素性ベクトルを作成する方法を紹介し、次にトピックモデルを用いたベクトルの作成法を述べる。

5.1　素性ベクトルの作成

本実験で利用する素性は、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 としている。

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 の品詞

e6 w の周辺自立語の表記

e7 e6 の分類語彙表の番号の 4 桁と 5 桁

例えば図 (5.1) は WSD の対象単語が 16 単語目の “経済” である文の形態素解析結果である。ここからは以下の組成リストが作成される。

```
e0=経済，e1=名詞-普通名詞-一般，
e2=人的，e3=形状詞，e4=的，e5=接尾辞，
e6=人的，e6=作る，e6=ＨＰ，e6=余裕，
e6=ある，e6=中小，e7=2386，e7=1197，
e7=11972
```

```
<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: 対象単語「経済」の例文

5.2 トピックベクトルを用いた手法

従来法ではWSDの対象単語wを含む文sから、素性ベクトルfを作成し、それをインスタンスとして学習・識別を行う。ここではwを含むsだけでなく、文sを含む文書dが利用できるという設定のもとでwの曖昧性解消を行う。つまり、訓練データやテストデータは対象単語wを含む文書セットとなる。トピックモデルを構築するにあたり、この文書セットを用いる。

前処理として対象単語を含む文書セットの各文書を形態素解析し、索引語文書行列を作成する。索引語文書行列は以下のように、各行が文書dを、各列は単語IDと文書内での単語の出現回数のペアとなる。

```
0:1 1:1
2:1 3:2 4:1
1:2 4:1 5:1
```

そして文書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。ここではwを含む文s以外に文sを含む文書dが与えられるので、トピックモデル構築後はwに対して、以下のトピックベクトルを作成することができる。

$$(P(z_1|d), P(z_2|d), \cdots, P(z_K|d))$$

この作成したトピックベクトルを従来の素性ベクトルに加えて学習・識別に用いる。

第6章　実験

本手法の効果を確認するために、素性ベクトルをそのまま用いる場合と、素性ベクトルにトピックベクトルを追加した場合との各々の WSD の制度を比較する。

データセットには、SemEval-2 の Japanese WSD のタスクを使用する。そこでは、WSD の対象単語が 50 単語設定されている。各々の単語には、50 件のテストデータと、約 50 件の訓練データが存在する。更に各データを含む文書も提供されており、本実験の設定も満たされている。

pLSI の学習は、テストデータと訓練データの各データに対する文書、合計約 100 文書が対象である。この学習を各単語に対して行う。また、pLSI を学習するために工藤拓氏のツール[1]を使用する。実行時のオプションで、潜在クラスの数 K として対象単語の語義数を指定した。出力されたファイル群より、確率 $P(d|z_i)$ をモデル化した pzd ファイルをトピックベクトルとして用いる。

SVM による識別には、LIBSVM[2]を使用する。用いたカーネルは線形カーネルである。

実験結果を表 (6.1) に示す。素性ベクトルだけを用いた場合と比べ、トピックベクトルを加えた場合は (+0.36%) の改善となった。また、トピックベクトルだけを用いた場合は素性ベクトルだけを用いた場合に比べ、(-9.0%) と精度が大幅に下がった。最後に、トピックベクトルの重み付けを変化させたところ、0.5 倍にした場合は (+1.12%) 上昇し、2 倍にした場合は (-0.64%) と減少した。

[1] http://www.chasen.org/~taku/software/plsi/
[2] http://www.csie.edu.tw/~cjlin/libsvm/

表 6.1: 各手法による正解率

対象単語	素性ベクトルのみ	トピックベクトル追加（重み 0.5）	トピックベクトル追加（重み 1.0）	トピックベクトル追加（重み 2.0）	トピックベクトルのみ
相手	84.00	84.00	84.00	84.00	82.00
あう	90.00	90.00	92.00	92.00	66.00
あげる	36.00	36.00	36.00	38.00	12.00
与える	76.00	78.00	76.00	74.00	58.00
生きる	94.00	94.00	94.00	94.00	94.00
意味	52.00	56.00	56.00	60.00	42.00
入れる	70.00	70.00	70.00	68.00	72.00
大きい	94.00	94.00	94.00	94.00	94.00
教える	42.00	42.00	44.00	44.00	18.00
可能	60.00	60.00	60.00	58.00	56.00
考える	98.00	98.00	98.00	98.00	98.00
関係	98.00	98.00	98.00	96.00	78.00
技術	84.00	84.00	84.00	84.00	84.00
経済	98.00	98.00	98.00	98.00	98.00
現場	76.00	78.00	76.00	76.00	78.00
子供	68.00	68.00	66.00	66.00	42.00
時間	86.00	86.00	86.00	88.00	88.00
市場	66.00	66.00	60.00	48.00	32.00
社会	86.00	86.00	86.00	86.00	86.00
情報	82.00	82.00	82.00	82.00	84.00
すすめる	82.00	86.00	86.00	86.00	32.00
する	60.00	62.00	62.00	54.00	38.00
高い	86.00	86.00	86.00	86.00	86.00
出す	44.00	44.00	42.00	40.00	28.00
立つ	54.00	54.00	54.00	52.00	52.00
強い	90.00	90.00	90.00	90.00	92.00
手	78.00	78.00	78.00	78.00	78.00
出る	60.00	58.00	58.00	58.00	58.00
電話	80.00	80.00	80.00	80.00	84.00
とる	38.00	40.00	40.00	30.00	28.00
のる	84.00	84.00	80.00	74.00	48.00
場合	86.00	88.00	88.00	88.00	86.00
入る	58.00	60.00	62.00	62.00	46.00
はじめ	96.00	96.00	96.00	94.00	66.00
はじめる	82.00	82.00	82.00	82.00	78.00
場所	96.00	96.00	96.00	96.00	96.00
早い	70.00	72.00	72.00	72.00	52.00
一	92.00	92.00	92.00	92.00	92.00
開く	92.00	92.00	92.00	90.00	90.00
文化	96.00	96.00	96.00	96.00	98.00
他	100.0	100.0	100.0	100.0	100.0
前	76.00	74.00	74.00	74.00	62.00
見える	76.00	78.00	78.00	74.00	40.00
認める	72.00	72.00	78.00	76.00	76.00
見る	80.00	80.00	80.00	80.00	80.00
持つ	66.00	68.00	68.00	68.00	68.00
求める	78.00	78.00	78.00	78.00	78.00
もの	88.00	88.00	88.00	88.00	88.00
やる	94.00	94.00	94.00	94.00	94.00
良い	38.00	40.00	40.00	40.00	24.00
平均	76.64	77.12	77.00	76.00	68.00

第7章　考察

トピックベクトルの重みについて考察する。今回の実験では、トピックベクトルの重みを大きくするほど全体の平均の精度は下がった。しかし、個別の単語について見ていくと、重みを大きくするほど精度が下がるものと、精度が上がるものがある。つまり、単語によってトピックベクトルの有効性の度合いが異なるということである。度合いの有効性はトピックベクトルの重みに対応するので、どのように適切な重みを求めていくかは今後の課題である。

次にトピックモデルを用いた語義曖昧性解消の関連研究について概説する。

Cai ら (2007) は、トピックモデルの一つである LDA(Latent Dirichlet allocation) によって、ラベルなしコーパスからトピック特性を抽出した、教師ありシステムの手法を提案している [3]。単語の品詞や対象語が含まれる文脈の構成語、bag-of-words、構文関係といった素性特徴と、ラベルなしコーパスから LDA によって得られたトピックによる条件付き単語確率を用いている。そして SenseVal-2 や SenseVal-3 の実験を通して、LDA のために改良したベイジアンネットワークを用いた分類が効果的であることを示した。

Boyd-Graber ら (2007) は、追加の潜在変数として WordNet の語義を LDA に組み込んだ手法を提案している [4]。これは WordNet-WALK と呼ばれる、WordNet の同義語グループ（synset）を辿る確率的プロセスの考え方と、LDA とを組み合わせて利用する。

Li, Roth ら (2010) は、LDA を基に、コーパスから語義の事前分布が得られる場合とそうでない場合、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モデルを構築する手法を提案している [5]。第 1 のモデルではトピックと文書による語義の条件付き確率を最大化する。第 2 のモデルでは、第 1 モデルの条件の余弦値から語義の条件付き確率を間接的に最大化する。第 3 のモデルでは、語義の言い換えを構成する単語を使って語義の条件付き確率を最大化する。粒度の違う WSD の実験、リテラルと非リテラルの語義検出の実験のそれぞれにおいて、素性特徴を用いたシステムの精度を上回った。特に言い換えの精度は約 (+10%) の向上となった。

上記 3 つの研究と本研究の大きな違いとして、本研究では対象単語に対して文書が与えられるという仮定を置いている。そのために、トピックモデルを単語毎に学習している。この利点としては、トピック数をいくつに設定するかという問題を避けることができる。

また、上記 3 つの研究はトピックモデルを学習するために LDA を用いている。本研究ではトピックモデルを利用する効果を示すことが目的であるので、簡易な pLSI を用いたが、LDA を利用することももちろん可能である。LDA を用いた場合は、更に精度が高まるものと予想する。

第8章　おわりに

本研究では、WSD にトピックモデルを利用する手法を提案した。ただし対象単語 *w* を含む文、*s* を含む文書 *d* が利用できることを前提としている。この前提のもとで対象単語に対する文書群からトピックモデルを学習し、インスタンスのトピックベクトルを作成する。このトピックベクトルを通常の素性ベクトルに追加する形で WSD を行う。実験では SemEval-2 の Japanese WSD アスクのデータを用いて、提案手法の有効性を示した。今後はトピックベクトルへの適切な重み付けの割り出しと、LDA といった pLSI 以外のトピックモデルを用いた場合についても研究をしていく必要がある。

謝辞

卒業研究にあたり、熱心にご指導いただいた情報工学科の新納准教授に深い感謝の意を表します。また、多くのご意見ご指摘を頂きました自然言語処理研究室の皆様に感謝します。

参考文献

[1] 福水健次. カーネル法入門-正定値カーネルによるデータ解析-. 朝倉書店, 2010.

[2] 新納浩幸. R で学ぶクラスタ解析. オーム社, 2007.

[3] J. Boyd-Graber, D. Blei, and X. Zhu. A topic model for word sense disambiguation. In *the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 1024-1033, 2007.

[4] J. Cai, W. S. Lee, and Y. W. 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 (EMNLP-CoNLL)*, pp. 1015-1023, 2007.

[5] M. I. Jordan. D. M. Blei, A. Y. Ng. Latent dirichlet allocation. *Machine Learning Research*, Vol. 3, pp. 993-1022, 2003.

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

[7] Linlin Li, Benjamin Roth, and Caroline Sporleder. Topic Models for Words Sense Disamiguation and Token-based Idiom Detection. In *ACL-2010*, pp. 1138-1147.

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

付 録A　ソースリスト

// 文書より単語を抜き出すプログラム「getnoun.pl」

```
#! perl -w

use encoding "shiftjis";
binmode STDERR, ":encoding(shiftjis)";

use warnings;
use XML::XPath;
use File::Basename;
use Fcntl;

if(@ARGV == 2) {

  #入力／出力ディレクトリ名の設定
  my $in_dir = $ARGV[0];
  my $out_dir = $ARGV[1];

  # ディレクトリオープン
  opendir(DIRHANDLE, $in_dir);

  foreach(readdir(DIRHANDLE)){

    # ベース名、拡張子の取得
    ($base, $ext) = &split_path($_);

    # 入力ファイル名
    my $in_file = $_;

    # 入力ファイルパス
    my $in_path = $in_dir . "/" . $in_file;

    if($ext eq "xml") { #拡張子「.xml」の確認

      # 出力ファイル名
      my $out_file = $base . ".noun";

      # 出力ファイルパス
      my $out_path = $out_dir . "/" . $out_file;

      my $xp = XML::XPath->new(filename=>$in_path);
      my $nodeset = $xp->find('//mor[starts-with(@pos, "名詞-")]');

      open(OUTFILE, ">:encoding(cp932)", $out_path) or die("Error: $_");
      foreach my $node ($nodeset->get_nodelist) {
        if($node->string_value){
            print OUTFILE $node->string_value, "\n";
          }
      }
      if(close(OUTFILE)){
        print  "$in_file -> $out_file \n";
      }
    }
  }

  # ディレクトリクローズ
  closedir(DIRHANDLE);

} else {
  print "引数を2つ指定して下さい\n";
```

```
}

sub split_path {
    local($path) = @_;
    local($dir, $file, $base, $ext);

    # ディレクトリ名の取得
    if ($path =~ /(.*)\/(.*)/) {
        $dir  = $1.'/';
        $file = $2;
    } else {
        $dir  = './';
        $file = $path;
    }

    # 拡張子の取得
    if ($file =~ /(.*)\.(.*)/) {
        $base = $1;
        $ext  = $2;
    } else { # 拡張子なし
        $base = $file;
        $ext  = "";
    }

    return($base, $ext);
}
```

// 索引語行列を作成するプログラム「noun2train_plsi.pl」

```
#! perl -w

use encoding "shiftjis";
use open ":encoding(cp932)";
binmode STDERR, ":encoding(shiftjis)";

use warnings;
use File::Basename;
use Tie::IxHash;
use Fcntl;

if(@ARGV == 3) {

  #入力／出力ディレクトリ名の設定
  my $data_dir = $ARGV[0];
  my $noun_dir = $ARGV[1];
  my $lex_dir = $ARGV[2];

  # ディレクトリオープン
  opendir(LEX_DIR, $lex_dir);

  # lexファイル名を取得
  @result = grep !/^\.\.*$/, readdir(LEX_DIR);

  my $i = 0;

  # IDより単語ディレクトリを読み込み
  foreach(@result){
    # IDを取得
    ($id, $ext) = &split_path($result[$i]);
    print "ID -> $id\n";

    # 単語をハッシュに登録
    open(LEX_FILE, "< $lex_dir/$result[$i]") or die("Error:$!");

    tie %noun, 'Tie::IxHash';

    my $num = 0;

    while(my $line = <LEX_FILE>){
      $num++;
```

```
42        chomp($line);
43        $noun{$line} = 0;
44      }
45
46      close(LEX_FILE);
47
48      my $flag = 0;
49      $num = 0;
50
51      my $train_file = "train_plsi/$id";
52
53      # 書き込みファイルのオープン
54      if( -f $train_file ) {
55        print "Error: \"$train_file\" is exist...";
56      } else {
57        open(OUTFILE, ">>:encoding(cp932)", $train_file) or die("Error: $_");
58
59        # filenameのチェック
60        opendir(ID_DIR, "$data_dir/$id");
61
62        foreach(readdir(ID_DIR)){
63          # filenameの存在判定
64          if($_ eq "filename"){
65            # ファイルオープン
66            open(LIST_FILE,"< $data_dir/$id/filename");
67
68            # ファイル読み込み
69            while ($line = <LIST_FILE>){
70
71              # ファイル名取得
72              @foo = split (/\s+/, $line);
73              ($base, $ext) = &split_path($foo[0]);
74
75              # nounファイルを開く
76              open(NOUN_FILE, "< $noun_dir/$base.noun") or die("Error:$!");
77
78              # ファイルから１行入力
79              while(my $line = <NOUN_FILE>){
80                chomp($line);
81
82                # 単語の存在チェック
83                if ( exists($noun{$line}) ) {
84                  # 単語のカウント
85                  $noun{$line}++;
86                }
87
88              }
89              close(NOUN_FILE);
90
91              my $flag = 0;
92              $num = 0;
93              # カウントした単語を書き込む
94              foreach my $key ( keys %noun ){
95                if($noun{$key} != 0) {
96                  if($flag) {
97                    print OUTFILE " ";
98                  }
99                  print OUTFILE "$num".":"."$noun{$key}";
100                 $noun{$key} = 0;
101                 $flag = 1;
102               }
103               $num++;
104             }
105             print OUTFILE "\n";
106           }
107
108           #ファイルクローズ
109           close(LIST_FILE);
110
111           # 出力ファイルのクローズ
112           close(OUTFILE);
113
```

```
            # ハッシュの削除
            foreach $key ( keys( %noun ) ) {
              delete $noun{$key};
            }

            print "$id is success.\n";
          }
        }
      }
      $i++;
    }

    # ディレクトリクローズ
    closedir(LEX_DIR);
} else {
    print "引数を3つ指定して下さい\n";
}

sub split_path {
    local($path) = @_;
    local($dir, $file, $base, $ext);

    # ディレクトリ名の取得
    if ($path =~ /(.*)\/(.*)/) {
        $dir  = $1.'/';
        $file = $2;
    } else {
        $dir  = './';
        $file = $path;
    }

    # 拡張子の取得
    if ($file =~ /(.*)\.(.*)/) {
        $base = $1;
        $ext  = $2;
    } else { # 拡張子なし
        $base = $file;
        $ext  = "";
    }

    return($base, $ext);
}
```

// 素性ベクトルを記したファイルにトピックベクトルを加えるプログラム「insert-pzd.pl」

```
#! perl -w

use encoding "shiftjis";
use open ":encoding(cp932)";
binmode STDERR, ":encoding(shiftjis)";

use warnings;
use File::Basename;
use Fcntl;

if(@ARGV == 3 or @ARGV == 4) {

  #入力／出力ディレクトリ名の設定
  my $pzd_dir = $ARGV[0];
  my $pzd_w = $ARGV[1];
  my $out_dir = $ARGV[2];
  my $ftr_dir = $ARGV[3];

  my @in_file;
  my @out_file;

  # ディレクトリオープン
  opendir(PZD_DIR, $pzd_dir);
```

```
24
25    # pdzファイル名を取得
26    @result = grep !/^\.\.*$/, readdir(PZD_DIR);
27
28    # ディレクトリクローズ
29    closedir(PZD_DIR);
30
31    my $num = 0;
32
33    # IDよりPDZディレクトリを読み込み
34    foreach(@result){
35      # IDを取得
36      ($id, $ext) = &split_path($result[$num]);
37
38      # PZDファイル判定
39      if($ext eq "pzd") {
40        print "ID -> $id\n";
41
42        # pzdファイルのオープン
43        open(PZD_FILE, "< $pzd_dir/$result[$num]") or die("Error:$!");
44
45        # 出力ディレクトリの作成
46        print "OUTPUT Directory ... ";
47        if (!-d "./$out_dir/$id"){
48          mkdir "./$out_dir/$id";
49          print"OK\n";
50        } else{
51          print "NG (exist)\n";
52        }
53
54        # 追加先ファイルのパス
55        if (@ARGV == 4) {
56          $in_file[0] = "$ftr_dir/$id/test.dat";
57          $in_file[1] = "$ftr_dir/$id/train.dat";
58        }
59
60        # 出力ファイルのパス
61        $out_file[0] = "$out_dir/$id/test_new.dat";
62        $out_file[1] = "$out_dir/$id/train_new.dat";
63
64
65        my $k = 0;
66
67        while($k < 2){
68          #--- P(Z|D)を特徴ベクトルデータに追加 ---#
69          if( -f $out_file[$k] ) {
70            print "Error: \"$out_file[$k]\" is exist...\n";
71          } else {
72
73            my @no;
74            my @count;
75
76            # 出力ファイルのオープン
77            open(OUTFILE, ">>:encoding(cp932)", $out_file[$k]) or die("Error: $_");
78
79            # 特徴ベクトルファイルのオープン
80            if($in_file[$k]) {
81              open(FT_FILE, "< $in_file[$k]") or die("Error:$!");
82
83              # 特徴ベクトルファイルより1行読み込み
84              while(my $f_line = <FT_FILE>){
85                chomp($f_line);
86                my @f_vec = split(/\s+/, $f_line);
87                my $f_len = @f_vec;
88
89                # PZDファイルより1行読み込み
90                my $pzd_line = <PZD_FILE>;
91                chomp($pzd_line);
92                my @pzd_vec = split(/\s+/, $pzd_line);
93                my $pzd_len = @pzd_vec;
94
95                $i = 1;
```

```
96
97                  # 特徴ベクトルのペアデータを分割
98                  while($i < $f_len){
99                    ($no[$i-1], $count[$i-1]) = split(/:/, $f_vec[$i]);
100                   $i++;
101                 }
102
103                 $i = 0;
104
105                 # ベクトル列を再構成して書き込み
106                 while($i < $f_len + $pzd_len) {
107                   if($i == 0) {
108                     print OUTFILE $f_vec[0];
109                   } elsif (0 < $i and $i <= $pzd_len) {
110                     print OUTFILE " ";
111                     print OUTFILE $i-1 . ":";
112                     print OUTFILE sprintf("%12.10f", $pzd_vec[$i-1] * $pzd_w);
113                   } else {
114                     print OUTFILE " ";
115                     print OUTFILE $no[$i-$pzd_len-1] + $pzd_len . ":" . $count[$i-$pzd_len
                          -1];
116                   }
117                   $i++;
118                 }
119                 print OUTFILE "\n";
120               }
121
122             close(FT_FILE);
123           } else {
124             if($k == 0){
125               my $j = 0;
126
127               while($j < 50) {
128                 my $pzd_line = <PZD_FILE>;
129                 chomp($pzd_line);
130                 my @pzd_vec = split(/\s+/, $pzd_line);
131                 my $pzd_len = @pzd_vec;
132
133                 $i = 0;
134
135                 # ダミークラスを書きこみ
136                 print OUTFILE "0";
137
138                 # ベクトル列を書き込み
139                 while($i < $pzd_len) {
140                   print OUTFILE " ";
141                   print OUTFILE $i . ":";
142                   print OUTFILE sprintf("%12.10f", $pzd_vec[$i] * $pzd_w);
143
144                   $i++;
145                 }
146
147                 print OUTFILE "\n";
148
149                 $j++;
150               }
151             } else {
152               open(TRAIN_FILE, "< feature/$id/train.dat") or die("Error:$!");
153
154               while(my $pzd_line = <PZD_FILE>) {
155                 chomp($pzd_line);
156                 my @pzd_vec = split(/\s+/, $pzd_line);
157                 my $pzd_len = @pzd_vec;
158
159                 $i = 0;
160
161                 # クラスを書きこみ
162                 my $train_line = <TRAIN_FILE>;
163                 chomp($train_line);
164                 my @train_vec = split(/\s+/, $train_line);
165
166                 print OUTFILE "$train_vec[0]";
```

```
167
168                 # ベクトル列を書き込み
169                 while($i < $pzd_len) {
170                   print OUTFILE " ";
171                   print OUTFILE $i . ":";
172                   print OUTFILE sprintf("%12.10f", $pzd_vec[$i] * $pzd_w);
173
174                   $i++;
175                 }
176
177                 print OUTFILE "\n";
178               }
179
180               close(TRAIN_FILE);
181             }
182           }
183           close(OUTFILE);
184         }
185
186         $k++;
187       }
188
189       # PZDファイルのクローズ
190       close(PZD_FILE);
191
192       print"finished!\n\n";
193     }
194
195     $num++;
196   }
197
198
199 } else {
200   print "引数を指定して下さい\n";
201 }
202
203 sub split_path {
204     local($path) = @_;
205     local($dir, $file, $base, $ext);
206
207     # ディレクトリ名の取得
208     if ($path =~ /(.*)\/(.*)/) {
209         $dir  = $1.'/';
210         $file = $2;
211     } else {
212         $dir  = './';
213         $file = $path;
214     }
215
216     # 拡張子の取得
217     if ($file =~ /(.*)\.(.*)/) {
218         $base = $1;
219         $ext  = $2;
220     } else { # 拡張子なし
221         $base = $file;
222         $ext  = "";
223     }
224
225     return($base, $ext);
226 }
```