平成 25 年度茨城大学工学部情報工学科卒業研究論文

外れ値検出手法を利用した Misleading データの検出

平成 26 年 2 月 7 日

情報工学科
吉田拓夢 (10T4067L)

指導教員　　新納浩幸 准教授

平成 25 年度茨城大学工学部情報工学科卒業研究論文

外れ値検出手法を利用した Misleading データの検出

著者 : 吉田拓夢 (10T4067L)
指導教員 : 新納浩幸 准教授

論文要旨

本論文では語義曖昧性解消 (Word Sense Disambiguation，WSD) の領域適応の問題に対して，識別精度を低下させる Misleading データを検出するために，外れ値検出手法を利用する．

自然言語処理のタスクにおいて帰納学習手法を用いる際，訓練データとテストデータは同じ領域のコーパスから得ていることが通常である．ただし実際には異なる領域である場合も存在する．そこである領域（ソース領域）の訓練データから学習された分類器を，別の領域（ターゲット領域）のテストデータに合うようにチューニングすることを領域適応という．領域適応の問題の一つは負の転移である．これはソース領域のデータを使いすぎるとターゲット領域での識別精度が下がる現象である．ここでは負の転移の原因を Misleading データの存在だと考えた．Misleading データとは分類器の学習に悪影響を与えるデータであり，Misleading データを検出，削除しておくことは分類器の精度向上に寄与する．

本論文では Misleading データはターゲット領域に対して外れ値になっていると予想し，この予想を確認する．まず先に予め Misleading データを検出しておき，それを正解データとすることで提案手法の評価を行うこととする．正解とする Misleading データは，次のような手順をもって検出する．始めに，訓練データ D から分類器を作成し，テストデータでその正解率 p_0 を測る．次に D の中の各データ x に対して，$D - x$ から分類器を作成し，テストデータでその正解率 p_1 を測る．$p_1 > p_0$ のとき x を Misleading データとみなす．このようにして検出した Misleading データを正解データとして，外れ値検出手法でどの程度それらのデータを検出できるか調べる．利用する外れ値検出手法としては (1) 最近傍法，(2) LOF および (3) 確率密度比の 3 手法を設定した．

実験では現代日本語書き言葉均衡コーパス (BCCWJ コーパス) における 3 つの領域 OC (Yahoo! 知恵袋)，PB (書籍) 及び PN (新聞) を利用する．SemEval-2 の日本語 WSD タスクではこれらのコーパスの一部に語義タグを付けたデータを公開しており，そのデータを利用する．すべての領域である程度の頻度が存在する多義語 16 単語を対象にして，WSD の領域適応の実験を行う．領域適応としては OC → PB，PB → PN，PN → OC，OC → PN，PN → PB，PB → OC の計 6 通りが存在する．結果 $16 \times 6 = 96$ 通りの WSD の領域適応の問題に対して実験を行った．

実験の結果，Misleading データの存在自体は確認できたが，外れ値検出手法による Misleading データの検出精度は低かった．外れ値検出手法では本論文で設定したような Misleading データの検出は困難であるが，負の転移の有無を判定することは，ある程度可能であることが判明した．外れ値ではない Misleading データがどのような特徴を持っていたかを調べることが今後の課題である．

目次

表目次

第 1 章

序論

1.1 概要

本論文では語義曖昧性解消 (Word Sense Disambiguation，WSD) の領域適応の問題に対して，識別精度を低下させる Misleading データを検出するために，外れ値検出手法を利用する．

自然言語処理のタスクにおいて帰納学習手法を用いる際，訓練データとテストデータは同じ領域のコーパスから得ていることが通常である．ただし実際には異なる領域である場合も存在する．そこである領域（ソース領域）の訓練データから学習された分類器を，別の領域（ターゲット領域）のテストデータに合うようにチューニングすることを領域適応という (Sogaard, Anders (2013)) [*1]．領域適応の問題の一つは負の転移である (Rosenstein, Michael T and Marx, Zvika and Kaelbling, Leslie Pack and Dietterich, Thomas G (2005))．これはソース領域のデータを使いすぎるとターゲット領域での識別精度が下がる現象である．我々は負の転移の原因を Misleading データの存在だと考えている．Misleading データとは分類器の学習に悪影響を与えるデータであり，Misleading データを検出，削除しておくことは分類器の精度向上に寄与する (Jing Jiang and Chengxiang Zhai (2007))．

本論文では Misleading データはターゲット領域に対して外れ値になっていると予想し，この予想を確認する．まず先に予め Misleading データを検出しておき，それを正解データとすることで提案手法の評価を行うこととする．正解とする Misleading データは，次のような手順をもって検出する．始めに，訓練データ D から分類器を作成し，テストデータでその正解率 p_0 を測る．次に D の中の各データ x に対して，$D - x$ から分類器を作成し，テストデータでその正解率 p_1 を測る．$p_1 > p_0$ のとき x を Misleading データとみなす．このようにして検出した Misleading データを正解データとして，外れ値検出手法でどの程度それらのデータを検出できるか調べる．利用する外れ値検出手法としては (1) 最近傍法，(2) LOF および (3) 確率密度比の 3 手法を設定した (吉田拓夢・新納浩幸 (2013))(吉田拓夢・新納浩幸 (2014))．

実験では現代日本語書き言葉均衡コーパス (BCCWJ コーパス (Maekawa (2007))) における 3 つの領域 OC (Yahoo! 知恵袋)，PB (書籍) 及び PN (新聞) を利用する．SemEval-2 の日本語 WSD タスク (Okumura et al. (2010)) ではこれらのコーパスの一部に語義タグを付けたデータを公開しており，そのデータを利用する．すべての領域である程度の頻度が存在する多義語 16 単語を対象にして，WSD の領域適応の実験を行う．領域適応としては OC → PB，PB → PN，PN → OC，OC → PN，PN → PB，PB → OC の計 6 通りが存在する．結果 $16 \times 6 = 96$ 通りの WSD の領域適応の問題に対して実験を行った．

実験の結果，Misleading データの存在自体は確認できたが，外れ値検出手法による Misleading データの検出精度は低かった．外れ値検出手法では本論文で設定したような Misleading データの検出は困難であるが，負の転移の有無を判定することは，ある程度可能であることが判明した．外れ値ではない Misleading データがどのような特徴を持っていたかを調べることが今後の課題である．

[*1] 領域適応は機械学習の分野では転移学習 (神嶌敏弘 (2010)) の一種と見なされている．

1.2 構成

本論文では，WSD の領域適応において Misleading データの存在を示し，外れ値検出による手法の有用性を調べる．次ぐ 2 章では WSD と領域適応について説明をする．3 章では，WSD でよく使用される SVM(サポートベクトルマシン) の原理について述べる．4 章では，Misleading データの存在とその除去の有用性を確認する．提案手法である外れ値検出の利用については 5 章に記す．6 章で実験の設定とその結果を提示し，7 章で考察を行う．8 章の結論をもって本研究のまとめを示す．

第 2 章

WSD の領域適応

2.1 語義曖昧性解消

語義曖昧性解消 (WSD:Word-sense disambiguation) は自然言語処理におけるタスクの 1 つである．複数の語義を持つある単語について，その単語がとある文中に存在した場合，その単語の語義を識別する処理である．自然言語処理には形態素解析や機械翻訳等の技術が存在するが，それらの様々な処理において語義曖昧性の問題が発生する．例えば，動詞「やる」という単語には以下のような異なる語義がある．

- 彼はその仕事をやった．(ある動作をする)
- その日はジャズをやった．(演奏/上映する)
- プレゼントとして時計をやった．(譲渡する)
- 机の上の本を向こうへやった．(どかす)
- 心配なので人をやった．(遣いを出す)
- 目を向こうへやった．(視線を投げる)

日本語の「やる」という動詞は上記の様に複数の語義があるが，他の言語においてはそれぞれ別の単語が対応するような場合は多々ある．英語では，「play」，「give」，「remove」などがそれに当たる．このように，機械翻訳などに発生する語義曖昧性を解消することは非常に重要な問題である．

2.1.1 機械学習

語義曖昧性の解消にあたっては一般的に，コーパスとテストコレクションから機械学習を行う，といったアプローチが試みられている．機械学習の手法としては，教師付き学習，半教師付き学習，教師なし学習等が利用される．教師なし学習は最も簡単な手法で，Source 領域のラベル付きデータのみを学習に用いる手法である．対して教師付き学習は，Source 領域の十分な量のラベル付きデータに加え，Target 領域の少量のラベル付きデータを利用する．また，半教師付き学習は Source 領域の十分な量のラベル付きデータと，Target 領域の十分な量のラベル無しデータを用いる．自然言語処理においては自然言語の文書を大量に集めた，コーパスを学習データとして用いるが，特に語義曖昧性解消のタスクでは，品詞や統語構造が予めメタデータとして付与されているタグ付きコーパスが好ましい．これは，語義曖昧性解消では，確率的な言語モデルに基づいて，直前や直後の単語の品詞や単語同士の共起の関係などの特徴を利用するからである．例えば 1 つの学習データは，以下の様な素性ベクトル $\mathbf{x}$ で表現される．

$$\mathbf{x} = \{g \;\; e0 \;\; e1 \;\; e2 \;\; e3 \;\; e4 \;\; e5\}$$

g :コーパス (領域)

$e0$:単語 w

$e1$:単語 w の品詞

$e2$:単語 w_{-1}

$e3$:単語 w_{-1} の品詞

$e4$:単語 w_1

$e5$:単語 w_1 の品詞

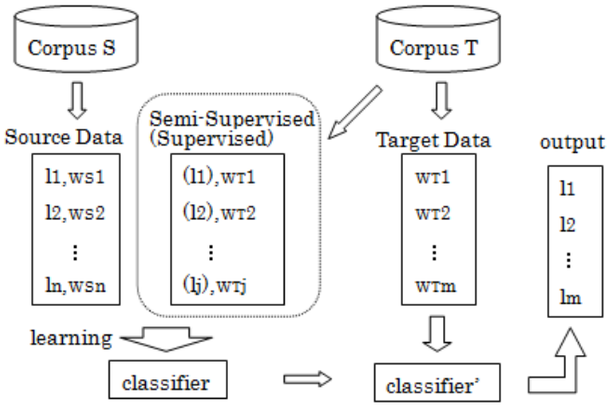

図 2.1　WSD の領域適応

2.2　領域適応

領域適応 (Domain Adaptation) は，転移学習 (transfer learning) ，帰納転移 (inductive transfer) 等多くの呼称がある．特に転移学習という語は，広く機械学習の枠組みに対して使われており，そのため，統一的な形式的定義は定めづらいとされている．しかしながら，形式的でこそないが以下の定義が広く受け入れられているという．

the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task

新規タスクの効果的な仮説を効率的に見つけ出すために，一つ以上の別のタスクで学習された知識を得て，それを適用する問題

これは転移学習のワークショップの論文募集 (2005) で示された．つまり，転移学習は，ある問題を効果的かつ，効率的に解くために，別の関連した問題のデータや学習結果を再利用することであると言える．

前述にもあるが，自然言語処理におけるタスクを機械学習で解決する際には，学習データにコーパスを用いる．そして，種々のタスク，例えば文書分類，固有表現抽出，形態素解析などの良い処理のために，コーパスには様々なメタデータが必要になる (固有表現のタグ，品詞のタグなど)．しかしながら，そのようなメタデータを付与したタグ付きコーパスを作成することには非常に多大なコストを必要とする．タグをつけることはプログラムによる自動化は難しく，人手による作業に頼らざるを得ない．そのような背景から，少ないコーパス資源を最大限利用すべく，転移学習の手法は自然言語処理で手広く受け入れられている．例えば，あるコーパス A から学習して分類器を作成したいが，コーパス A のデータはテストデータで利用するため学習データには回せないので，別のコーパス B のデータで学習して分類器を作成する，というようなケースが挙げられる．

語義曖昧性解消の領域適応で，例えば以下のようなケースを考える．日本語のカタカナ表記で「マウス」という単語の語義は，「ネズミ」，「コンピュータのポインティングデバイス」，「口」の 3 種がある．これを，コーパス A はコンピュータ雑誌のコーパス，コーパス B を医療雑誌のコーパスとして語義の曖昧性を解消しようとした場合，良い識別結果が得られないという予測は明白である．「コンピュータのポインティングデバイス」としての語義はコーパス A では頻出するであろうが，コーパス B ではそれほど多くは出てこないであろう．逆に，「ネズミ」という語義は動物実験としてコーパス B では頻出するであろうことも容易に想像できる．このように，コーパス間で語義の分布が異なることは分類器の精度の妨げとなる．

この様な問題は，以下のようにして解決できる．

WSD の対象単語 w の語義の集合を $C = \{c_1, c_2, \cdots, c_k\}$，$w$ を含む文（入力データ）を x とする．WSD の問題は事後確率最大化に基づければ以下で表せる．

$$\arg\max_{c \in C} P(c)P(x|c)$$

つまり訓練データを利用して語義の分布 $P(c)$ と各語義上での入力データの分布 $P(x|c)$ を推定することで WSD の問題は解決できる．

第 3 章

SVM

本論文のタスクでは，機械学習に SVM(Support vector machine) を用いる．SVM は，線形しきい素子を利用して 2 クラスのパターン識別器を構成する手法である．線形しきい素子はニューロンのモデルとして最も単純である単純パーセプトロンである．入力特徴ベクトル $\mathbf{x} = x_1, x_2, ..., x_N$ に対して，以下の線形識別関数により 2 値の出力を得る．

$$y = \mathrm{sign}(g(\mathbf{x}))$$

$$g(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} - h$$

ここで，$\mathbf{w}$ は重みベクトルであり，h はバイアス項である．関数 sign(a) は，以下に定める符号関数である．

$$\mathrm{sign}(a) = \begin{cases} 1 & (a > 0) \\ -1 & (a \leq 0) \end{cases}$$

このモデルは，入力 $\mathbf{x}$ と重み $\mathbf{w}$ の内積が閾値 h を越えた場合に 1 を，下回った場合に -1 を出力する．これは，識別平面 $g(x) = 0$ によって入力特徴空間を 2 分割することを意味する．各クラスラベルを $1, -1$ の 2 値とし，N 個の入力ベクトル $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ とその正答クラスラベルを $t_1, t_2, ..., t_N$ とする．また，線形分離可能であるとする．識別平面が存在するとき，$t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} - h) \geq 1$ が成立する．

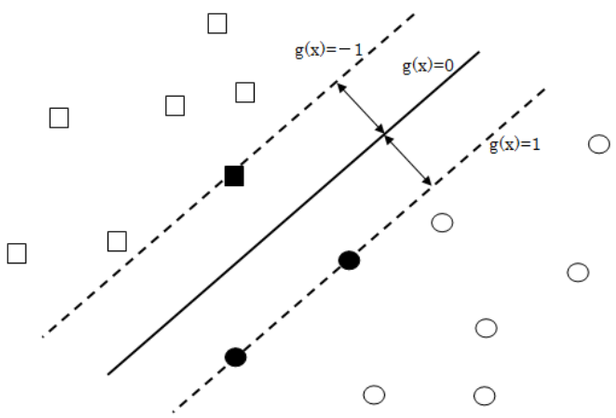

図 3.1　識別平面とマージン

また，余裕のある識別平面を考えると，それはそれぞれのクラスを通る 2 つのギリギリの識別平面の中間だと言える．そのときの 2 つの識別平面は，それぞれ $g(\mathbf{x}_{+1}) = 1, g(\mathbf{x}_{-1}) = -1$ である．$g(\mathbf{x}_{+1}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_{+1} - h = 1$, $g(\mathbf{x}_{-1}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_{-1} - h = -1$ であるから, 平面同士の距離は, $\mathbf{w}^{\mathrm{T}}(\mathbf{x}_{+1} - \mathbf{x}_{-1}) = 2$ より $(\mathbf{x}_{+1} - \mathbf{x}_{-1}) = \frac{2}{||\mathbf{w}||}$ と表せられる．求めるべき識別平面は，その 2 つの識別平面から $\frac{1}{||\mathbf{w}||}$ の距離となる．この量は特にマージンと呼ばれている．このマージンが最大となるような識別平面が，求めるべき最適な識別平面である．したがって，マージンが最大となる $\mathbf{w}$ と h を求める問題は，制約条件

$$t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} - h) \geq 1, (i = 1, 2, ..., N)$$

の下で，目的関数

$$L(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$$

を最小とするパラメータを求める問題となる．この問題は一例として，双対問題へ帰着する手法がある．ラグランジュ乗数 $\alpha_i(\geq 0), i = 1, ..., N$ を用いて，目的関数を次式とする．

$$L(\mathbf{w}, h, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i \{t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} - h) - 1\}$$

パラメータ $\mathbf{w}$ と h に関して偏微分より，

$$\bigtriangledown_{\mathbf{w}} L(\mathbf{w}, h, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i, \frac{\partial L(\mathbf{w}, h, \boldsymbol{\alpha})}{\partial h} = \sum_{i=1}^{N} \alpha_i t_i$$

これらを 0 として，以下を得る．

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i$$

$$0 = \sum_{i=1}^{N} \alpha_i t_i$$

さらに目的関数の式へ代入して，制約条件

$$\sum_{i=1}^{N} \alpha_i t_i = 0$$

$$0 \leq \alpha_i, i = 1, ..., N$$

の下で，目的関数

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}^{\mathrm{T}}{}_i \mathbf{x}_j$$

を最大とする双対問題を得る．この双対問題はラグランジュ乗数 $\alpha_i(\geq 0), i = 1, ..., N$ の最適化問題となっている．最適化問題の解決手法の 1 つに，最急降下法がある．最急降下法は，関数の傾きのみから関数の最大値 (あるいは最小値) を探索する勾配法のアルゴリズムの 1 つである．この場合は，次式によって最適解を求める．

$$\begin{aligned} \boldsymbol{\alpha}^{(k+1)} &= \boldsymbol{\alpha}^{(k)} + \eta \quad \mathrm{grad} \quad L_D(\boldsymbol{\alpha}^{(k)}) \\ &= \boldsymbol{\alpha}^{(k)} - \eta \begin{bmatrix} \frac{\partial L_D(\boldsymbol{\alpha}^{(k)})}{\partial \alpha^{(k)}{}_1} \\ \frac{\partial L_D(\boldsymbol{\alpha}^{(k)})}{\partial \alpha^{(k)}{}_2} \\ \vdots \\ \frac{\partial L_D(\boldsymbol{\alpha}^{(k)})}{\partial \alpha^{(k)}{}_N} \end{bmatrix} \end{aligned}$$

勾配法では反復法で解に近付けてゆく．k 回目の反復で解が $\alpha^{(k)}$ の位置のとき，上記のように更新を行う．ここで η は 1 回の更新の変化量を定めるパラメータであり，通常は小さな正の定数が設定される．これによって最適解 ${\alpha_i}^*(\geq 0)$ が得られたとき，最適な重みベクトル $\mathbf{w}^*$ は，

$$\mathbf{w}^* = \sum_{i \in S} {\alpha_i}^* t_i \mathbf{x_i}$$

で与えられる．ここで S はサポートベクターに対応する添え字の集合である．サポートベクターとは，${\alpha_i}^* > 0$ となる入力ベクトル x_i のことで，識別平面 $g(\mathbf{x}) = 1$，もしくは $g(\mathbf{x}) = -1$ 上にある．また，最適なバイアス項 h^* は，任意のサポートベクターを利用して求められる．任意のサポートベクターを $\mathbf{x}_s, s \in S$ として，

$$h^* = {\mathbf{w}^*}^{\mathrm{T}}\mathbf{x}_s - t_s$$

となる．以上をもって，最適な識別関数は次式で表される．

$$\begin{aligned} y &= \mathrm{sign}({\mathbf{w}^*}^{\mathrm{T}}\mathbf{x} - h^*) \\ &= \mathrm{sign}\left(\sum_{i \in S} {\alpha^*}_i t_i {\mathbf{x}^{\mathrm{T}}}_i \mathbf{x} - h^*\right) \end{aligned}$$

第 4 章

Misleading データ

4.1 misleading データの正答集合の作成

後述の手法 (6 章 1 節 2 項) により Misleading データとみなせるデータを調べたところ，語義曖昧性解消の領域適応において Misleading データは存在した．

検出したデータ数を表 4.1 に示す．表 4.1 は各単語について OC → PB，PB → PN，PN → OC，OC → PN，PN → PB，PB → OC の計 6 通りの領域適応を行ったときに Misleading データとみなすことの出来たデータ数である (下段 (%))．それらを本論文において検出すべき Misleading データの正答集合とし，本論文の提案手法によってそれらの検出を試みる．

4.2 Misleading データの除去

Misleading データは分類器の精度を悪化させるデータであり，Source 領域のデータに含まれている．前節では Misleading データの存在を確認したことについて触れ，その検出数を示した．それらを本論文では Misleading データの正答集合としたが，本節でそれら正答集合の Misleading データを除去した際の分類器の精度を確認する．

正答集合の Misleading データを除去することで，実際に領域適応で分類器の精度が向上することを示す．表 4.2 は，Misleading データを除いた際の正答率と，Misleading データを除かずにそのまま領域適応を行った場合の正答率を，領域ごとの平均で示したものである．表中の「Mislead」が正答集合の Misleading データを除いた場合の領域適応であり，「NORMAL」が Misleading データを除かずにそのまま領域適応を行った場合である．

表 4.2 にあるとおり，Misleading データを除いた際の正答率の平均は Misleading データを除かずにそのまま領域適応を行った場合の正答率の平均よりも良い精度が得られている．このように，ある一定のデータを除くことにより分類器の精度は向上する．逆に，ある一定のデータが含まれることで分類器の精度が低下しているとも言い換えられる．このようなデータが Misleading データである．

以上より，正答集合とした Misleading データが実際に分類器の精度を悪化させていること，また，それらを除くことで分類器の精度が向上することを示した．

表 4.1　検出した Misleading データ

単語	OC		PB		PN	
	PB	PN	OC	PN	OC	PB
言う	159/666	158/666	127/1114	75/1114	82/363	35/363
	23.87	23.72	11.40	6.730	22.59	9.640
入れる	6/73	28/73	19/56	15/56	3/32	1/32
	8.220	38.36	33.93	26.79	9.380	3.130
書く	21/99	39/99	0/62	2/62	12/27	15/27
	21.21	39.40	-	3.230	44.44	55.56
聞く	26/124	21/124	26/123	0/123	4/52	27/52
	20.97	16.94	21.14	-	7.700	51.92
子供	5/77	0/77	12/93	1/93	12/29	13/29
	6.490	-	12.90	1.080	41.38	44.83
時間	1/53	8/53	0/74	0/74	0/59	5/59
	1.890	15.09	-	-	-	8.470
自分	13/128	25/128	0/308	0/308	0/71	1/71
	10.16	19.53	-	-	-	1.410
出る	14/131	10/131	39/152	32/152	22/89	10/89
	10.69	7.630	25.66	21.05	24.72	11.24
取る	6/61	5/61	10/81	18/81	12/43	22/43
	9.840	8.200	12.35	22.22	27.91	51.16
場合	0/126	0/126	7/137	13/137	14/73	9/73
	-	-	5.110	9.490	19.18	12.33
入る	36/68	11/68	38/118	27/118	27/65	42/65
	52.94	16.18	32.20	22.88	41.54	64.62
前	8/105	5/105	10/160	1/160	15/106	2/106
	7.620	4.760	6.250	0.625	14.15	1.890
見る	10/262	3/262	3/273	12/273	8/87	28/87
	38.18	1.150	1.100	4.400	9.200	32.18
持つ	8/62	0/62	2/153	11/153	1/59	1/59
	12.90	-	1.310	7.190	1.690	1.690
やる	0/117	0/117	0/156	0/156	0/27	0/27
	-	-	-	-	-	-
ゆく	17/219	0/219	15/133	1/133	3/27	3/27
	7.760	-	11.29	0.752	11.11	11.11

表 4.2　正答集合の Misleading データを除いた際の平均正答率 (%)

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Mislead	74.53	71.91	72.86	79.22	74.61	78.67	75.30
NORMAL	70.77	66.96	70.29	75.56	68.49	73.26	70.89

第 5 章

外れ値検出

語義曖昧性解消の領域適応のタスクを行う際，Source 領域のデータ $\mathbf{x}^{(\mathrm{S})} = \{x^{(\mathrm{S})}{}_1, x^{(\mathrm{S})}{}_2, ..., x^{(\mathrm{S})}{}_n\}$ と Target 領域のデータ $\mathbf{x}^{(\mathrm{T})} = \{x^{(\mathrm{T})}{}_1, x^{(\mathrm{T})}{}_2, ..., x^{(\mathrm{T})}{}_m\}$ が用いられる．本論文では，Misleading データとなる様な Source 領域のデータ点 $x^{(\mathrm{S})}{}_i$ は Target 領域のデータ $\mathbf{x}^{(\mathrm{T})}$ に対して外れ値になると予想し，Misleading データの検出に外れ値検出の手法を用いた．Target 領域のデータ $\mathbf{x}^{(\mathrm{T})}$ に対して，ある Source 領域のデータ点 $\mathbf{x}^{(\mathrm{S})}{}_i$ がどれだけ外れているかを，以下の外れ値検出の 3 手法をもって測定する．

5.1 最近傍法 (Erk の手法)

Erk による外れ値検出の手法 (Katrin Erk (2006)) を示す．

外れ値の度合いを測るデータ点を点 x とする．この点 x に対して，対象データの中で最近傍となる点 t_n と，その点 t に対する最近傍点 t_n’ を定める．これらの 3 つの点について，以下の距離を求める．

点 x と点 t_n の距離 dxt
点 t_n と点 $t_n{}'$ の距離 dtt’

この 2 つの距離を用いて，以下のように外れ値 $p_{NN}(x)$ を定める．

$$p_{NN}(x) = \frac{d_{xt}}{d_{tt'}}$$

5.2 LOF

LOF(local outlier factor) は密度をベースとした外れ値検出手法である．ある点のまわりの密度が他の点と比べて小さいほど，LOF の値は大きくなる．LOF の値を測る点を x としたとき，x の k 距離近傍集合 $N_k(x)$ を以下の様に定める．

$$N_k(x) = \{y \in D \setminus \{x\} | d(x, y) \leq kdist(x)\}$$

ここで，$kdist(x)$ は以下の条件を満たす $d(x, o)$ である．

1. 少なくとも k 個のデータ $o' \in D \setminus \{x\}$ に対して $d(x, o') \leq d(x, o)$ が成立する
2. 高々 $k-1$ 個のデータ $o' \in D \setminus \{x\}$ に対してのみ $d(x, o) < d(x, o)$ が成立する

すなわち, 簡単には k 距離近傍集合 $N_k(x)$ は点 x から k 番目に近い点 o_k までの距離 $kdist(x) = dist(x, o_k)$ の範囲内にある点の集合である．LOF の算出に先立ち，まずは以下を求める．

$$lrd_k(x) = \frac{|N_k(x)|}{\sum_{y \in N_k(x)} rd_k(x,y)}$$

これは局所到達可能密度 (local reachability density，lrd) と呼ばれる値で，x の k 近傍内にあるデータの到達可能距離 (reachability distance，rd) の平均の逆数となっている．到達可能距離 rd は以下で定める値である．

$$rd_k(x,y) = \max\{d(x,y), kdist(y)\}$$

つまり，点 x と y の距離が y の k 距離よりも近い場合には y の k 距離に置き換えて到達可能距離 rd としている．以上をもって，LOF は次式により定められる．

$$LOF(x) = \frac{1}{|N_k(x)|} \sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)}$$

上式に示されるとおり，LOF は点 x の局所到達可能密度と点 x の k 近傍点の局所到達可能密度との平均を取っている．

5.3 密度比

対象単語 w の用例 $\boldsymbol{x}$ の素性リストを $\{f_1, f_2, \cdots, f_n\}$ とする．求めるのは領域 $R \in \{S, T\}$ 上の $\boldsymbol{x}$ の分布 $P_R(\boldsymbol{x})$ である．ここで Naive Bayes で使われるモデルを用いる．Naive Bayes のモデルでは以下を仮定する．

$$P_R(\boldsymbol{x}) = \prod_{i=1}^{n} P_R(f_i)$$

領域 R のコーパス内の w の全ての用例について素性リストを作成しておく．ここで用例の数を $N(R)$ とおく．また $N(R)$ 個の用例の中で，素性 f が現れた用例数を $n(R,f)$ とおく．MAP 推定でスムージングを行い，$P_R(f)$ を以下で定義する高村大也 (2010).

$$P_R(f) = \frac{n(R,f)+1}{N(R)+2}$$

以上より，ソース領域 S の用例 $\boldsymbol{x}$ に対して，確率密度比 $w(\boldsymbol{x}) = P_T(\boldsymbol{x})/P_S(\boldsymbol{x})$ が計算できる．

第 6 章

実験

6.1 実験設定

6.1.1 基本設定

BCCWJ コーパスの PB(書籍)，OC(Yahoo! 知恵袋) 及び PN (新聞) を異なった領域として実験を行う．SemEval-2 の日本語 WSD タスク Okumura et al. (2010) ではこれら領域のコーパスの一部に語義タグを付けたデータを公開しており，そのデータを利用する．この 3 つの領域からある程度頻度のある多義語 16 単語を WSD の対象単語とする．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義数を表 6.1 に示す．[*1] 領域適応の方向としては OC → PB，PB → PN，PN → OC，OC → PN，PN → PB，PB → OC の計 6 通りの方向が存在する．

6.1.2 Misleading データの存在の調査

語義曖昧性解消の領域適応において Misleading データの存在を確かめるため，以下のような実験を行った．

実験の基本設定において 6 つの領域適応が行われるが，それぞれの領域適応のための機械学習で用いる Source データを $\mathbf{x}^{(\mathrm{S})} = \{x_1, x_2, ..., x_n\}$ とする．この Source データに対して，任意の i 番目のデータ x_i1 つを取り除いた新たな Source データを $\mathbf{x}^{(\mathrm{S})}{}_i$ とする．ここで新たな Source データ $\mathbf{x}^{(\mathrm{S})}{}_i$ で学習を行った場合に，元の Source データ $\mathbf{x}^{(\mathrm{S})}$ で学習を行った場合よりも分類器の精度が向上したならば，$\mathbf{x}^{(\mathrm{S})}$ に含まれるデータ x_i は学習の精度を下げる Misleading データだったと考えられる．Source データ $\mathbf{x}^{(\mathrm{S})}$ に対して $\mathbf{x}^{(\mathrm{S})}{}_1$ から $\mathbf{x}^{(\mathrm{S})}{}_n$ までの新しい n 個の Source データを作成，学習し，データ $\mathbf{x}^{(\mathrm{S})}$ に含まれる n 個全てのデータ点 x_i が Misleading データであるかどうかをそれぞれ 1 つずつ判別する．このようにして判別された Misleading データの集合を，その領域適応における検出すべき Misleading データの正解集合とした．

6.1.3 外れ値検出

実験の基本設定による 6 つの領域適応で用いられる Source データ $\mathbf{x}^{(\mathrm{S})}$ と Target データ $\mathbf{x}^{(\mathrm{T})}$ について，$\mathbf{x}^{(\mathrm{T})}$ に対する $\mathbf{x}^{(\mathrm{S})}$ の n 個全ての各データ点 $\mathbf{x}^{(\mathrm{S})}{}_1, \mathbf{x}^{(\mathrm{S})}{}_2, ..., \mathbf{x}^{(\mathrm{S})}{}_n$ の外れ値を計算する．なお，距離の計算は，提案手法のいずれもユークリッド距離をもって算出している．

6.2 実験結果

16 単語の 6 つの領域適応において，提案手法により 3 手法の外れ値の計算を行った．LOF では算出した値を正規化し，閾値 $\theta = 1.96$ より大きな値を Misleading データとみなした．Erk, 密度比の手法はそれぞれ Misleading データの検出正答率の平均が大きくなるような閾値を探し，結果，Erk の手法では閾値 $\theta = 1.9$ より大きな値を，密度比の手法では閾値 $\theta = 0.005$ より小さな値を Misleading データとした．

[*1] 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，OC や PB では 4 つの語義がある．これは SemEval-2 の日本語 WSD タスクでは新語義のタグも許しているからである．

これらの提案手法による Misleading データの検出正答率を，6 領域適応ごとに 16 単語の平均を取った．これを表 6.2 に示す．検出正答率は検出した Misleading データの数でそのうちの Misleading データの正答集合に含まれる数を割ったものである．いずれの手法，領域適応においても検出正答率は著しく低い．

また，提案手法による misleading データを除いた場合の領域適応の正答率を表 6.3 に示す．LOF の手法のみが僅かに通常の領域適応の正答率を上回った．

表 6.1　対象単語

単語	辞書上の語義数	OC での頻度	OC での語義数	PB での頻度	PB での語義数	PN での頻度	PN での語義数
言う	3	666	2	1114	2	363	2
入れる	3	73	2	56	3	32	2
書く	2	99	2	62	2	27	2
聞く	3	124	2	123	2	52	2
子供	2	77	2	93	2	29	2
時間	4	53	2	74	2	59	2
自分	2	128	2	308	2	71	2
出る	3	131	3	152	3	89	3
取る	8	61	7	81	7	43	7
場合	2	126	2	137	2	73	2
入る	3	68	4	118	4	65	3
前	3	105	3	160	2	106	4
見る	6	262	5	273	6	87	3
持つ	4	62	4	153	3	59	3
やる	5	117	3	156	4	27	2
ゆく	2	219	2	133	2	27	2
平均	3.44	148.19	2.94	199.56	3.00	75.56	2.69

表 6.2　提案手法による検出正答率 (%)

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Erk	3.590	9.920	14.36	19.07	10.64	29.32	14.48
LOF	3.67	10.19	6.990	5.260	18.48	19.62	10.70
密度比	10.49	10.03	8.880	11.93	17.02	22.10	13.41

表 6.3　提案手法で misleading を除いた場合の正答率 (%)

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Erk	70.91	68.27	69.45	75.38	69.45	73.33	70.10
LOF	70.71	67.11	70.09	75.57	68.70	73.44	70.94
密度比	69.63	67.34	68.38	76.37	59.72	66.42	67.98
NORMAL	70.77	66.96	70.29	75.56	68.49	73.26	70.89

第 7 章

考察

7.1　最近傍距離の相関

外れ値検出の手法では Misleading データの検出能力が極めて低いと言える．外れ値検出はいずれの手法もデータ点の距離の差異を利用するものであるが，Misleading データと非 Misleading データの間にはその差異が認められなかったと考えられる．そこで，Misleading データと非 Misleading データについて，以下を調べる．

- Target データに対する最近傍点への距離についての差異の有無

実験で使用した 16 単語 6 種計 96 ケースの領域適応のデータについて，外れ値検出同様にそれぞれの Source データ点の Target データへの最近傍距離を算出した．これを Misleading データの正答集合により，Misleading データと非 Misleading データに分けて平均を取り，それらの相関を t 検定により判定した．なお，有意水準は 5% とした．

その結果，有意差が認められたケースは 96 ケース中 12 ケースであった．また，その有意差の有無と外れ値の 3 手法による Misleading データの検出結果を照らし合わせたところ，有意差が認められた 12 ケースの領域適応において Misleading データの検出率が特段高い訳でもなかった．Misleading データは Target データへの最近傍距離において非 Misleading データとの差異があるとは言えず，そのため，距離を利用する外れ値検出の手法をもって Misleading データを判別することは難しい．

7.2　負の転移現象

外れ値検出手法を用いても，ここで設定したような Misleading データを検出することは困難であった．本節では Misleading データと関連の深い負の転移の有無を外れ値検出手法で判定できるかを調べてみる．

まず論文 (新納浩幸・佐々木稔 (2014)) では本論文と同じデータを利用して，負の転移が生じなかった対象単語を選出している．その手法を次節で示す．

7.2.1　負の転移現象の調査

負の転移が生じているかどうかを調べるために以下の実験を行った．

単語 w_i についてソース領域 S からターゲット領域 T への領域適応の実験を行う．まずターゲット領域 T のラベル付きデータをランダムに 15 個取り出し，残りを評価データとする．つまり利用できる訓練データはソース領域 S のラベル付きデータとターゲット領域 T からランダムに取り出した 15 個のラベル付きデータとなる．この訓練データを用いて手法 A により分類器を作成し，先の評価データの語義識別の正解率 $P_{i,k}$ を測る．この実験を 5 回行い $P_{i,1}, P_{i,2}, \cdots, P_{i,5}$ を得る．それらの平均 P_i を「単語 w_i の S から T への領域適応における手法 A の平均正解率」とする．上記の単語 w_i を 16 種類の各対象単語 $w_1, w_2, \cdots, w_{16}$ に変えることで，16 個の平均正解率 $P_1, P_2, \cdots, P_{16}$ が得られる．それらの平均 P を「S から T への領域適応における手法 A の平均正解率」とする．

上記の手法 A としては，以下の 3 種類を試す．(1) ソース領域のラベル付きデータのみを用いる手法

(ターゲット領域の 15 個のラベル付きデータの重みを 0 とする手法) (S-Only), (2) ターゲット領域からランダムに取り出した 15 個のラベル付きデータのみを用いる手法 (ソース領域のラベル付きデータの重みを 0 とする手法) (T-Only), (3) ソース領域のラベル付きデータとターゲット領域の 15 個のラベル付きデータを用いる手法 (S+T).

S から T への領域適応における各手法の平均正解率を表 7.1 に示す.

負の転移が生じているかどうかの判定には，上記の実験でより得られた `T-Only`，`S-Only` 及び `S+T` の正解率を利用する．もしも正解率で以下の関係が成立しているなら，負の転移が生じていないと考えられる.

`T-Only, S-Only  <  S+T`

結果を表 7.2 に示す．チェックがつけられた箇所が負の転移が生じていない領域適応の問題である．96 種類の領域適応の問題の中で 44 種類において負の転移が生じていない.

7.2.2 負の転移の予想

次に本論文で行った外れ値検出手法で検出された Misleading データの割合が，全体のデータの 1 割以下である場合に，負の転移が生じないという判定を行う．これによって外れ値検出手法を利用して，負の転移が生じない対象単語の検出評価を行うことができる．検出の正解率，再現率，F 値をそれぞれ表 7.3, 表 7.4，表 7.5 に示す．Mislead は本実験で用いた Misleading の正解データを利用した検出を示す.

表 7.5 を見ると，Misleading の正解データを用いても負の転移のない単語を検出する能力は高くない．それに比較すれば外れ値検出手法を利用した場合の検出する能力は高い．外れ値検出手法を利用して負の転移が生じない単語を判定できる可能性もあり，この点で精度改善が可能であると考える.

表 7.1　各手法の平均正解率

領域適応	S-Only	T-only	S+T
OC → PB	0.7137	0.7559	0.7511
PB → PN	0.7678	0.7206	0.7801
PN → OC	0.6926	0.7716	0.7630
OC → PN	0.6829	0.7300	0.7324
PN → PB	0.7543	0.7561	0.7863
PB → OC	0.6988	0.7766	0.7533
平均	0.7184	0.7518	0.7611

表 7.2　負の転移が生じていない領域適応

単語	OC → PB	PB → PN	PN → OC	OC → PN	PN → PB	PB → OC
言う		✓	✓	✓	✓	
入れる		✓	✓	✓	✓	✓
書く	✓			✓	✓	
聞く	✓					
子供			✓		✓	
時間	✓		✓		✓	
自分	✓	✓				
出る				✓		✓
取る			✓		✓	✓
場合		✓	✓		✓	✓
入る		✓	✓		✓	✓
前		✓				
見る	✓					
持つ	✓	✓				✓
やる		✓		✓		✓
ゆく		✓		✓	✓	

表 7.3　外れ値検出手法を利用した負の転移が生じない単語の検出・正解率

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Erk	0.286	0.333	0.500	0.533	0.250	0.615	0.420
LOF	0.375	0.375	0.438	0.563	0.438	0.563	0.458
密度比	0.000	1.000	0.000	0.500	0.000	0.000	0.250
Mislead	0.125	0.333	0.375	0.583	0.286	0.429	0.355

表 7.4　外れ値検出手法を利用した負の転移が生じない単語の検出・再現率

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Erk	0.667	0.833	1.000	0.889	0.429	0.889	0.784
LOF	1.000	1.000	1.000	1.000	1.000	1.000	1.000
密度比	0.000	0.333	0.000	0.556	0.000	0.000	0.148
Mislead	0.167	0.500	0.429	0.778	0.286	0.333	0.415

表 7.5　外れ値検出手法を利用した負の転移が生じない単語の検出・F 値

	OC		PB		PN		avr
	PB	PN	OC	PN	OC	PB	
Erk	0.400	0.476	0.667	0.667	0.316	0.727	0.542
LOF	0.545	0.545	0.609	0.720	0.609	0.720	0.625
密度比	-	0.500	-	0.526	-	-	0.513
Mislead	0.143	0.400	0.400	0.667	0.286	0.375	0.378

第 8 章

結論

本論文では語義曖昧性解消の領域適応のタスクにおいて，分類器の精度を下げる Misleading データが存在することをまず確認した．続いて外れ値検出による提案手法で Misleading データの検出を試みたが，これは良い結果が得られなかった．

本論文では WSD の領域適応における Misleading データの検出に外れ値検出手法を利用することを試みた．総当たり的に各データが Misleading データと見なせるかどうかを調べることで，Misleading データの存在を確認できた．また，それらを正解集合として外れ値検出を用いた検出能力も調べた．結論的には外れ値検出手法を利用しても，本論文で設定したような Misleading データの検出は困難であることがわかった．ただし Misleading データと関連の深い負の転移現象の有無を判定することには利用可能だと考えている．今後は外れ値ではない Misleading データの特徴を調査することで，新たな Misleading データの検出法を考えたい．

謝辞

本研究を進めるにあたって，指導教員である新納浩幸准教授には多大なご指導を頂きました．また，新納研究室の皆様からは日々の議論を通じて多くの知識や示唆を頂きました．この場を借りて，私を支えて下さった皆様方にお礼申し上げます．

参考文献

[1] Jing Jiang and Chengxiang Zhai (2007) “Instance weighting for domain adaptation in NLP,” in *Proc. of ACL-2007*, pp. 264–271.

[2] Katrin Erk (2006) “Unknown Word Sense Detection As Outlier Detection,” in *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pp. 128–135.

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

[4] Manabu Okumura, Kiyoaki Shirai, Kanako Komiya, and Hikaru Yokono (2010) “SemEval-2010 Task: Japanese WSD,” in *Proc. of the 5th International Workshop on Semantic Evaluation*, pp. 69–74.

[5] Rosenstein, Michael T and Marx, Zvika and Kaelbling, Leslie Pack and Dietterich, Thomas G (2005) “To transfer or not to transfer,” in *Proc. of the NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*.

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

[7] 吉田拓夢、新納浩幸 (2013) 「語義曖昧性解消の領域適応における Misleading データの存在と検出」，第 4 回日本語ワークショップ，pp.317–324.

[8] 吉田拓夢、新納浩幸 (2014) 「外れ値検出を利用した Misleading データの検出」，第 5 回日本語ワークショップ.

[9] 高村大也 (2010) 言語処理のための機械学習入門，コロナ社.

[10] 新納浩幸、佐々木稔 (2014) 「共変量シフトの問題としての語義曖昧性解消の領域適応」，自然言語処理，第 21 巻，第 1 号.

[11] 神嶌敏弘 (2010) 「転移学習」，人工知能学会誌，第 25 巻，第 4 号，pp.572–580.

付録 ソースリスト

リスト 1　svm 入力形式ファイル作成

```
#!/usr/bin/perl

use strict;
use warnings;
use utf8;
use Encode qw/decode encode/;
binmode STDOUT, ":utf8";
my @xx=(1707,2998,7479,10487,17877,20676,22038,35478,
37713,40333,40699,48488,50038,51332,52310,52744);
#各ディレクトリ下のファイル label を読込して
#label_s(ource) と rabel_t(arget) の svm形式を作る
#WSDの各組合せごとで処理をする
#OC-PB,PB-OC
    my $s='OC';
    my $t='PB';
    make_label($s,$t);
    make_label($t,$s);
#OC-PN,PN-OC
    $t='PN';
    make_label($s,$t);
    make_label($t,$s);
#PB-PN,PN-PB
    $s='PB';
    make_label($s,$t);
    make_label($t,$s);

#同じ処理なのでサブルーチンで記述
sub make_label{
    my($source,$target)=@_;
#16単語分の WSDのタスクに使う svmfileを作成
for my $num (@xx){
#読込ファイルを開く
#sourceのファイルを開く
    my $file_s ="$source/$num/label";
    open my $fh_s, '<', $file_s
   or die qq/Can't open file "$file_s" : $!/;
#targetのファイルを開く
    my $file_t ="$target/$num/label";
    open my $fh_t, '<', $file_t
   or die qq/Can't open file "$file_t" : $!/;
#sourceとtarget両領域の出現単語の indexファイルを開く
    my $wsd="$source"."-"."$target";
    my $wordlist='wordlist_'."$source"."$target"."_"."$num";
    my $file_wl ="task_wsd/$wsd/$num/normal/$wordlist";
    open my $fh_wl, '<', $file_wl
   or die qq/Can't open file "$file_wl" : $!/;
#sourceとtarget両領域のラベルのindexファイルを開く
    my $label='svm_label_'."$source"."$target";
    my $file_label ="task_wsd/$wsd/$num/normal/$label";
    open my $fh_label, '<', $file_label
```

```
51      or die qq/Can't open file "$file_label" : $!/;
52  #INDEX->hash_index
53  #sourceとtarget両領域の出現単語とそのindexを対応
54       my %hash_index=();
55       while(my $line=<$fh_wl>){
56      $line=decode('UTF-8',$line);
57      $line=~m/(\d+)\s(.+?)\n/;
58      $hash_index{$2}=$1;
59       }
60
61  #label->hash_label
62  #sourceとtarget両領域のラベルとそのindexを対応
63       my %hash_label=();
64    while(my $line=<$fh_label>){
65      $line=decode('UTF-8',$line);
66      $line=~m/(\d+)\s(.+?)\n/;
67      $hash_label{$2}=$1;
68       }
69  #書込みファイルを開く
70  #sourceのsvm形式ファイル
71       my $name_s='label_'."$source".'_s';
72       my $wfile_s ="task_wsd/$wsd/$num/normal/$name_s";
73       open my $wfh_s, '>', $wfile_s
74      or die qq/Can't open file "$wfile_s" : $!/;
75  #targetのsvm形式ファイル
76       my $name_t='label_'."$target".'_t';
77       my $wfile_t ="task_wsd/$wsd/$num/normal/$name_t";
78       open my $wfh_t, '>', $wfile_t
79      or die qq/Can't open file "$wfile_t" : $!/;
80
81  #sourceとtargetそれぞれ1行ごとに読み込み、indexを参照して書き込む
82  #source側の処理
83  #「label」->1行ずつ処理
84       while(my $line=<$fh_s>){
85      $line=decode('UTF-8',$line);
86  #単語を抜出
87      my @tmp=$line=~m/(?:e\d=(.+?)\s)/g;
88  #labelを抜出
89      my @temp=$line=~m/([^ ]+)\n/;
90      my $svmlabel=$hash_label{$temp[0]};
91  #単語->hash{index}=頻度
92      my %hash=();
93      foreach my $word (@tmp){
94           $hash{$hash_index{$word}}++;
95          # print "word->$word index->$hash_index{$word}";
96      }
97      #print "$#tmp label->$label\n";
98  #書込み
99      print $wfh_s encode('UTF-8',$svmlabel);
100     foreach my $num (sort { $a <=> $b } keys %hash){
101          print $wfh_s " ".encode('UTF-8',$num).":$hash{$num}";
102     }
103     print $wfh_s "\n";
104      }
105      close $wfh_s or die qw/Can't close file "$wfile_s": $!/;
106
107 #target側の処理
108 #「label」->1行ずつ処理
109      while(my $line=<$fh_t>){
110     $line=decode('UTF-8',$line);
111 #単語を抜出
112     my @tmp=$line=~m/(?:e\d=(.+?)\s)/g;
113 #labelを抜出
```

```
114     my @temp=$line=~m/([^ ]+)\n/;
115     my $svmlabel=$hash_label{$temp[0]};
116 #単語->hash{index}=頻度
117     my %hash=();
118     foreach my $word (@tmp){
119         $hash{$hash_index{$word}}++;
120        # print "word->$word index->$hash_index{$word}";
121     }
122     #print "$#tmp label->$label\n";
123 #書込み
124     print $wfh_t encode('UTF-8',$svmlabel);
125     foreach my $num (sort { $a <=> $b } keys %hash){
126         print $wfh_t " ".encode('UTF-8',$num).":$hash{$num}";
127     }
128     print $wfh_t "\n";
129      }
130      close $wfh_t or die qw/Can't close file "$wfile_t": $!/;
131 #sourceとtarget書込み終了
132 #読込ファイルを閉じる
133      close $fh_s;
134      close $fh_t;
135      close $fh_wl;
136      close $fh_label;
137
138 }#close:for(@xx)
139 }
```

リスト 2　svm 入力形式ファイル作成 (Misleading データ調査用)

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use utf8;
6  use Encode qw/decode encode/;
7  binmode STDOUT, ":utf8";
8  my @xx=(1707,2998,7479,10487,17877,20676,22038,35478,
9  37713,40333,40699,48488,50038,51332,52310,52744);
10 #各ディレクトリ下のファイルlabel_XX_sを読込して
11 #label_search_misleard下にlabel_XX_Mを1～Nまで作る
12 #WSDの各組合せごとで処理をする
13 #OC-PB,PB-OC
14      my $s='OC';
15      my $t='PB';
16      make_label_lose($s,$t);
17      make_label_lose($t,$s);
18 #OC-PN,PN-OC
19      $t='PN';
20      make_label_lose($s,$t);
21      make_label_lose($t,$s);
22 #PB-PN,PN-PB
23      $s='PB';
24      make_label_lose($s,$t);
25      make_label_lose($t,$s);
26 ##########################################################
27 #同じ処理なのでサブルーチンで記述
28 sub make_label_lose{
29    my($source,$target)=@_;
30 #16単語それぞれごとに処理する
31     foreach my $index (@xx){
32 #読込ファイルを開く
33        my $dir ="$source"."-"."$target";
34        my $name ='label_'."$source".'_s';
```

```
35        my $file ="task_wsd/$dir/$index/normal/$name";
36        open my $fh, '<', $file
37          or die qq/Can't open file "$file" : $!/;
38 #もとのsvm形式ファイルを1行ごとに配列へ
39        my @oc=();
40        while(my $line=<$fh>){
41            push @oc, $line;
42        }
43 #行の数だけ繰り返し新たなファイルに書き込む
44        for(my $x=1;$x < $#oc + 2;$x++){
45 #書込みファイルを開く
46            my $num='label_'."$source".'_'."$x";
47            my $wfile ="task_wsd/$dir/$index/label_search_mislead/$num";
48            open my $wfh, '>', $wfile
49              or die qq/Can't open file "$wfile" : $!/;
50 #該当する1行以外を全て書き込む
51            for(my $y=0;$y < $#oc + 1;$y++){
52                if($y != $x -1){
53                    print $wfh $oc[$y];
54                }
55            }
56            close $wfh or die qw/Can't close file "$wfile": $!/;
57        }
58        close $fh;
59        print "$index".':finish'."\n";
60    }
61    print "$source".'-'."$target".':finish'."\n";
62 }
```

リスト 3　外れ値検出 (最近傍 (Erk))

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use utf8;
6  use Encode qw/decode encode/;
7  binmode STDOUT, ":utf8";
8  my @xx=(1707,2998,7479,10487,17877,20676,22038,35478,
9  37713,40333,40699,48488,50038,51332,52310,52744);
10 #WSDの各組合せごとで処理をする
11 #OC-PB,PB-OC
12      my $s='OC';
13      my $t='PB';
14      make_label_lose($s,$t);
15      make_label_lose($t,$s);
16 #OC-PN,PN-OC
17      $t='PN';
18      make_label_lose($s,$t);
19      make_label_lose($t,$s);
20 #PB-PN,PN-PB
21      $s='PB';
22      make_label_lose($s,$t);
23      make_label_lose($t,$s);
24 ##########################################################
25 #同じ処理なのでサブルーチンで記述
26 sub make_label_lose{
27    my($source,$target)=@_;
28    #16単語それぞれごとに処理する
29    foreach my $index (@xx){
30        #読込ファイルを開く
31        my $dir ="$source".'-'."$target";
32        my $name ='label_'."$source".'_s';
```

```
33          my $file ="task_wsd/$dir/$index/normal/$name";
34          open my $fh, '<', $file
35            or die qq/Can't open file "$file" : $!/;
36
37          my $name_t ='label_'."$target".'_t';
38          my $file_t ="task_wsd/$dir/$index/normal/$name_t";
39          open my $fh_t, '<', $file_t
40            or die qq/Can't open file "$file_t" : $!/;
41
42          my $name_i ='wordlist_'."$source"."$target";
43          my $file_i ="task_wsd/$dir/$index/normal/$name_i";
44          open my $fh_i, '<', $file_i
45            or die qq/Can't open file "$file_i" : $!/;
46          #source領域のデータを1行ごとに配列へ
47          my @arry_s=();
48          while(my $line=<$fh>){
49              push @arry_s,$line;
50          }
51          #target領域のデータを1行ごとに配列へ
52          my @arry_t=();
53          while(my $line=<$fh_t>){
54              push @arry_t,$line;
55          }
56          #wordlistのデータを1行ごとに配列へ
57          my @arry_i=();
58          while(my $line=<$fh_i>){
59              push @arry_i,$line;
60          }
61          #書込みファイルを開く
62          my $num='outlier_NN_'."$source".'_'."$index";
63          my $wfile ="task_wsd/$dir/$index/$num";
64          open my $wfh, '>', $wfile
65            or die qq/Can't open file "$wfile" : $!/;
66          #行の数だけ繰り返し外れ値の計算をする
67          for(my $x=0;$x < $#arry_s + 1;$x++){
68              my $d_xt_min=99999;my $d_tt_min=99999;my $pnn=0;
69              my $num_xt_min=0;
70              #source:xのベクトルxの要素を抽出する
71              my %hash_s=();
72              my @tmp = $arry_s[$x] =~m/\s(\d+:\d+)/g;
73              foreach my $str (@tmp){
74                  $str =~m/(\d+):(\d+)/;
75                  $hash_s{$1}=$2;
76              }
77              #source:xに近いtarget:tのデータ点を探す
78              for(my $y=0;$y < $#arry_t + 1;$y++){
79                  #target:tのベクトルtの要素を抽出する
80                  my %hash_t=();
81                  my @tmp2 = $arry_t[$y] =~m/\s(\d+:\d+)/g;
82                  foreach my $str (@tmp2){
83                      $str =~m/(\d+):(\d+)/;
84                      $hash_t{$1}=$2;
85                  }
86                  #xとtの距離を計算する([wordlist]次元のベクトル)
87                  my $calc=0;
88                  my %hash_count=();
89                  foreach my $key (keys %hash_s){
90                      $hash_count{$key}++;
91                  }
92                  foreach my $key (keys %hash_t){
93                      $hash_count{$key}++;
94                  }
95                  foreach my $key (keys %hash_count){
```

```
96                  my $a=0;
97                  my $b=0;
98                  if(exists $hash_s{$key}){
99                      $a =$hash_s{$key};
100                 }
101                 if(exists $hash_t{$key}){
102                     $b =$hash_t{$key};
103                 }
104                 $calc+=($a - $b)*($a - $b);
105             }
106             if($d_tt_min > $calc){
107                 $d_xt_min=$calc;
108                 $num_xt_min=$y
109             }
110
111         }
112         #target:t_minのベクトルt_minの要素を抽出する
113         my %hash_t_min=();
114         my @tmp3 = $arry_t[$num_xt_min] =~m/\s(\d+:\d+)/g;
115         foreach my $str (@tmp3){
116             $str =~m/(\d+):(\d+)/;
117             $hash_t_min{$1}=$2;
118         }
119
120         #target:tに近いtarget:t'のデータ点を探す
121         for(my $y=0;$y < $#arry_t + 1;$y++){
122             #target:tのベクトルt'の要素を抽出する
123             my %hash_t2=();
124             my @tmp4 = $arry_t[$y] =~m/\s(\d+:\d+)/g;
125             foreach my $str (@tmp4){
126                 $str =~m/(\d+):(\d+)/;
127                 $hash_t2{$1}=$2;
128             }
129             #xとtの距離を計算する([wordlist]次元のベクトル)
130             my $calc=0;
131             my %hash_count=();
132             foreach my $key (keys %hash_t_min){
133                 $hash_count{$key}++;
134             }
135             foreach my $key (keys %hash_t2){
136                 $hash_count{$key}++;
137             }
138             foreach my $key (keys %hash_count){
139                 my $a=0;
140                 my $b=0;
141                 if(exists $hash_t_min{$key}){
142                     $a =$hash_t_min{$key};
143                 }
144                 if(exists $hash_t2{$key}){
145                     $b =$hash_t2{$key};
146                 }
147                 $calc+=($a - $b)*($a - $b);
148             }
149             if($d_tt_min > $calc && $calc > 0){
150                 $d_tt_min=$calc;
151                 #$num_xt_min=$y
152             }
153         }
154     #ファイルに外れ値の度合いp_NN(x)を書き込む
155     $pnn=sqrt($d_xt_min / $d_tt_min);
156     print $wfh $pnn;
157     print $wfh "\n";
158     }
```

```
159         close $wfh or die qw/Can't close file "$wfile": $!/;
160         close $fh;
161         close $fh_t;
162         close $fh_i;
163         print "$index.':finish'."\n";
164     }
165     print "$source.'-'."$target.':finish'."\n";
166 }
```

リスト 4　外れ値検出 (LOF)

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use utf8;
6  use Encode qw/decode encode/;
7  binmode STDOUT, ":utf8";
8  my @xx=(1707,2998,7479,10487,17877,20676,22038,35478,
9  37713,40333,40699,48488,50038,51332,52310,52744);
10 #WSDの各組合せごとで処理をする
11 #forkで6つの領域適応を半分ずつ
12 my $pid = fork;
13 die "Cannot fork: $!" unless defined $pid;
14
15 if ($pid) {
16   # 子プロセスの終了を待機する。
17     print "start:lof_OCPB\n";
18     lof('OC','PB');
19     print "end:lof_OCPB\n";
20     print "start:lof_OCPN\n";
21     lof('OC','PN');
22     print "end:lof_OCPN\n";
23     print "start:lof_PBPN\n";
24     lof('PB','PN');
25     print "end:lof_PBPN\n";
26
27
28   my $id=wait;
29   print "id=$id\n";
30 }
31 else {
32   # 子プロセスでLOF
33     print "start:lof_PBOC\n";
34     lof('PB','OC');
35     print "end:lof_PBOC\n";
36     print "start:lof_PNOC\n";
37     lof('PN','OC');
38     print "end:lof_PNOC\n";
39     print "start:lof_PNPB\n";
40     lof('PN','PB');
41     print "end:lof_PNPB\n";
42
43   print "pid:finish\n";
44 }
45 ##########################################################
46 #同じ処理なのでサブルーチンで記述
47 sub lof{
48    my($source,$target)=@_;
49    #16単語それぞれごとに処理する
50    foreach my $index (@xx){
51       #読込ファイルを開く
52       my $dir ="$source".'-'."$target";
```

```
53          my $name ='label_'."$source".'_s';
54          my $file ="task_wsd/$dir/$index/normal/$name";
55          open my $fh, '<', $file
56            or die qq/Can't open file "$file" : $!/;
57
58          my $name_t ='label_'."$target".'_t';
59          my $file_t ="task_wsd/$dir/$index/normal/$name_t";
60          open my $fh_t, '<', $file_t
61            or die qq/Can't open file "$file_t" : $!/;
62
63          #source領域のデータを1行ごとに配列へ
64          my @arry_s=();
65          while(my $line=<$fh>){
66              push @arry_s,$line;
67          }
68          #target領域のデータを1行ごとに配列へ
69          my @arry_t=();
70          while(my $line=<$fh_t>){
71              push @arry_t,$line;
72          }
73          #書込みファイルを開く
74          my $num='outlier_lof_k10_'."$source".'_'."$index";
75          my $wfile ="task_wsd/$dir/$index/$num";
76          open my $wfh, '>', $wfile
77            or die qq/Can't open file "$wfile" : $!/;
78          #行の数だけ繰り返し外れ値の計算をする
79          for(my $x=0;$x < $#arry_s + 1;$x++){
80  ##########################################################################
81  #LOFの処理
82  my %n_k=();#$n_k{m}=rd_k(x,m) |m\in Nk(x)
83  my $lrdk_x=0;
84  #lrdkの計算
85  {
86  #a)点xと各ターゲットデータtとのそれぞれの距離を測る
87      my %dist=();
88      for(my $i=0;$i < $#arry_t +1;$i++){
89          $dist{$i +1}=&calc_dist($arry_s[$x],$arry_t[$i]);
90      }
91
92  #b)測った距離からk距離近傍のデータを探す
93      my $count=1;
94      my $k=10;
95      my $prev=0;
96      foreach my $key (sort { $dist{$a} <=> $dist{$b} } keys %dist){
97          if($count <= $k){
98              $n_k{$key}=$dist{$key};
99              if($count == $k){
100                 $prev=$dist{$key};
101             }
102             $count++;
103         }elsif($dist{$key} == $prev){
104             $n_k{$key}=$dist{$key};
105         }else {last;}
106     }
107
108 #c)k距離近傍のそれぞれのデータと点xの到達可能距離を測る
109     foreach my $key (keys %n_k){
110     #a)b)の処理
111         my %dist=();
112         my %n_k2=();
113         for(my $i=0;$i < $#arry_t +1;$i++){
114             if($key != $i +1){
115                 $dist{$i +1}=&calc_dist($arry_t[$key -1],$arry_t[$i]);
```

```
116             }
117         }
118         my @aaaa=(sort { $dist{$a} <=> $dist{$b} }keys %dist);
119         my $max=$dist{$aaaa[4]};
120         #到達可能距離rd_k=max{d(x,y),kdist(y)}
121         if($n_k{$key} < $max){
122             $n_k{$key}=$max;
123         }
124     }
125
126 #d)局所到達可能密度lrdk(x)の計算
127     $count=0;
128     my $sum_rdk=0;
129     foreach my $key (keys %n_k){
130         $sum_rdk+=$n_k{$key};
131         $count++;
132     }
133     $lrdk_x=$count/$sum_rdk;
134 }
135
136 #xのk近傍のlrdkの和を求める
137 my $sum_lrdk=0;
138 foreach my $y_nk (keys %n_k){
139 #printf "n_k{%d}=%f\n",$y_nk,$n_k{$y_nk};#デバッグ用
140 #lrdkの計算
141 {
142     my %n_k3=();
143 #a)点yと各ターゲットデータtとのそれぞれの距離を測る
144     my %dist=();
145     for(my $i=0;$i < $#arry_t +1;$i++){
146         if($y_nk != $i +1){
147             $dist{$i +1}=&calc_dist($arry_t[$y_nk -1],$arry_t[$i]);
148         }
149     }
150 #b)測った距離からk距離近傍のデータを探す
151     my $count=1;
152     my $k=10;
153     my $prev=0;
154     foreach my $key (sort { $dist{$a} <=> $dist{$b} }keys %dist){
155         if($count <= $k){
156             $n_k3{$key}=$dist{$key};
157             if($count == $k){
158                 $prev=$dist{$key};
159             }
160             $count++;
161         }elsif($dist{$key} == $prev){
162             $n_k3{$key}=$dist{$key};
163         }else {last};
164     }
165 #c)k距離近傍のそれぞれのデータと点yの到達可能距離を測る
166     foreach my $key (keys %n_k3){
167     #a)b)の処理
168         my %dist=();
169         my %n_k2=();
170         for(my $i=0;$i < $#arry_t +1;$i++){
171             if($key != $i +1){
172                 $dist{$i +1}=&calc_dist($arry_t[$key -1],$arry_t[$i]);
173             }
174         }
175         my @bbbb= (sort { $dist{$a} <=> $dist{$b} }keys %dist);
176         my $max=$dist{$bbbb[4]};
177         #到達可能距離rd_k=max{d(x,y),kdist(y)}
178         if($n_k3{$key} < $max){
```

```
179             $n_k3{$key}=$max;
180         }
181     }
182 #d)局所到達可能密度lrdk(x)の計算
183     $count=0;
184     my $sum_rdk=0;
185     foreach my $key (keys %n_k3){
186         $sum_rdk+=$n_k3{$key};
187         $count++;
188 #printf "n_k3{%f}=%f\n",$key,$n_k3{$key};#デバッグ用
189     }
190     $sum_lrdk+=$count/$sum_rdk;
191 }
192 }
193 #LOFの計算
194 my $count_nk=keys %n_k;
195 my $lof=($sum_lrdk/$lrdk_x)/$count_nk;
196
197 #LOFの処理ここまで
198 ######################################################################
199         #ファイルに外れ値の度合いLOF(x)を書き込む
200         print $wfh $lof;
201         print $wfh "\n";
202         printf "wrote.%s,%s,%s\n",($x +1).'/'.($#arry_s +1),"$index","$dir";
203         }
204         close $wfh or die qw/Can't close file "$wfile": $!/;
205         close $fh;
206         close $fh_t;
207         print "$index".':finish'."\n";
208     }
209     print "$source".'-'."$target".':finish'."\n";
210 }
211
212 sub calc_dist{
213     my($str1,$str2)=@_;
214     #str1のベクトルの要素を抽出する
215     my %hash_s=();
216     my @tmp = $str1 =~m/\s(\d+:\d+)/g;
217     foreach my $str (@tmp){
218         $str =~m/(\d+):(\d+)/;
219         $hash_s{$1}=$2;
220     }
221     #str2のベクトルの要素を抽出する
222     my %hash_t=();
223     my @tmp2 = $str2 =~m/\s(\d+:\d+)/g;
224     foreach my $str (@tmp2){
225         $str =~m/(\d+):(\d+)/;
226         $hash_t{$1}=$2;
227     }
228     #xとtの距離を計算する([wordlist]次元のベクトル)
229     my $calc=0;
230     my %hash_count=();
231     foreach my $key (keys %hash_s){
232         $hash_count{$key}++;
233     }
234     foreach my $key (keys %hash_t){
235         $hash_count{$key}++;
236     }
237     foreach my $key (keys %hash_count){
238         my $a=0;
239         my $b=0;
240         if(exists $hash_s{$key}){
241             $a =$hash_s{$key};
```

```
242         }
243         if(exists $hash_t{$key}){
244            $b =$hash_t{$key};
245         }
246         $calc+=($a - $b)*($a - $b);
247      }
248  return sqrt $calc;
249  }
```

リスト 5　最近傍距離

```
1   #!/usr/bin/perl
2
3   use strict;
4   use warnings;
5   use utf8;
6   use Encode qw/decode encode/;
7   binmode STDOUT, ":utf8";
8   my @xx=(1707,2998,7479,10487,17877,20676,22038,35478,
9   37713,40333,40699,48488,50038,51332,52310,52744);
10  #WSDの各組合せごとで処理をする
11  #OC-PB,PB-OC
12  my $pid = fork;
13  die "Cannot fork: $!" unless defined $pid;
14
15  if ($pid) {
16    # 子プロセスの終了を待機する。
17       print "start:lof_OCPB\n";
18       lof('OC','PB');
19       print "end:lof_OCPB\n";
20       print "start:lof_OCPN\n";
21       lof('OC','PN');
22       print "end:lof_OCPN\n";
23       print "start:lof_PBPN\n";
24       lof('PB','PN');
25       print "end:lof_PBPN\n";
26
27
28    my $id=wait;
29    print "id=$id\n";
30  }
31  else {
32    # 子プロセスでLOF
33       print "start:lof_PBOC\n";
34       lof('PB','OC');
35       print "end:lof_PBOC\n";
36       print "start:lof_PNOC\n";
37       lof('PN','OC');
38       print "end:lof_PNOC\n";
39       print "start:lof_PNPB\n";
40       lof('PN','PB');
41       print "end:lof_PNPB\n";
42
43    print "pid:finish\n";
44  }
45  ##########################################################
46  #同じ処理なのでサブルーチンで記述
47  sub lof{
48     my($source,$target)=@_;
49     #16単語それぞれごとに処理する
50     foreach my $index (@xx){
51        #読込ファイルを開く
52        my $dir ="$source"."-"."$target";
```

```
53        my $name ='label_'."$source".'_s';
54        my $file ="task_wsd/$dir/$index/normal/$name";
55        open my $fh, '<', $file
56          or die qq/Can't open file "$file" : $!/;
57
58        my $name_t ='label_'."$target".'_t';
59        my $file_t ="task_wsd/$dir/$index/normal/$name_t";
60        open my $fh_t, '<', $file_t
61          or die qq/Can't open file "$file_t" : $!/;
62
63        #source領域のデータを1行ごとに配列へ
64        my @arry_s=();
65        while(my $line=<$fh>){
66            push @arry_s,$line;
67        }
68        #target領域のデータを1行ごとに配列へ
69        my @arry_t=();
70        while(my $line=<$fh_t>){
71            push @arry_t,$line;
72        }
73        #書込みファイルを開く
74        my $num='dist_nn_'."$source"."$target".'_'."$index".'_ver2';
75        my $wfile ="task_wsd/$dir/$index/$num";
76        open my $wfh, '>', $wfile
77          or die qq/Can't open file "$wfile" : $!/;
78        #行の数だけ繰り返し最近傍点との距離を計算をする
79        for(my $x=0;$x < $#arry_s + 1;$x++){
80 ##########################################################################
81 #点の距離を測る
82 #a)点xと各ターゲットデータtとのそれぞれの距離を測る
83     my %dist=();
84     for(my $i=0;$i < $#arry_t +1;$i++){
85         $dist{$i +1}=&calc_dist($arry_s[$x],$arry_t[$i]);
86     }
87
88 #b)測った距離から最近傍の距離を取り出す
89 #   my $dist_nn=0;
90 #   my $count=1;
91 #   foreach my $key (sort { $dist{$a} <=> $dist{$b} } keys %dist){
92 #       if($count == 1){
93 #           $dist_nn=$dist{$key};
94 #           $count++;
95 #       }else {last;}
96 #   }
97 my @sort_arry=sort { $dist{$a} <=> $dist{$b} } keys %dist;
98 my $dist_nn=$dist{$sort_arry[0]};
99 ##########################################################################
100       #ファイルに最近傍距離を書き込む
101       print $wfh $dist_nn;
102       print $wfh "\n";
103       }
104       close $wfh or die qw/Can't close file "$wfile": $!/;
105       close $fh;
106       close $fh_t;
107       print "$index".':finish'."\n";
108   }
109   print "$source".'-'."$target".':finish'."\n";
110 }
111
112 sub calc_dist{
113     my($str1,$str2)=@_;
114     #str1のベクトルの要素を抽出する
115     my %hash_s=();
```

```
116     my @tmp = $str1 =~m/\s(\d+:\d+)/g;
117     foreach my $str (@tmp){
118         $str =~m/(\d+):(\d+)/;
119         $hash_s{$1}=$2;
120     }
121     #str2のベクトルの要素を抽出する
122     my %hash_t=();
123     my @tmp2 = $str2 =~m/\s(\d+:\d+)/g;
124     foreach my $str (@tmp2){
125         $str =~m/(\d+):(\d+)/;
126         $hash_t{$1}=$2;
127     }
128     #xとtの距離を計算する([wordlist]次元のベクトル)
129     my $calc=0;
130     my %hash_count=();
131     foreach my $key (keys %hash_s){
132         $hash_count{$key}++;
133     }
134     foreach my $key (keys %hash_t){
135         $hash_count{$key}++;
136     }
137     foreach my $key (keys %hash_count){
138        my $a=0;
139        my $b=0;
140        if(exists $hash_s{$key}){
141           $a =$hash_s{$key};
142        }
143        if(exists $hash_t{$key}){
144           $b =$hash_t{$key};
145        }
146        $calc+=($a - $b)*($a - $b);
147     }
148 return sqrt $calc;
149 }
```