# Learning under Covariate Shift for Domain Adaptation for Word Sense Disambiguation

**Hiroyuki Shinnou, Minoru Sasaki, Kanako Komiya**
Ibaraki University, Department of Computer and Information Sciences
4-12-1 Nakanarusawa, Hitachi, Ibaraki JAPAN 316-8511
hiroyuki.shinnou.0828@vc.ibaraki.ac.jp,
{minoru.sasaki.01, kanako.komiya.nlp}@vc.ibaraki.ac.jp

## Abstract

We show that domain adaptation for word sense disambiguation (WSD) satisfies the assumption of covariate shift, and then solve it by learning under covariate shift. Learning under covariate shift has two key points: (1) calculation of the weight of an instance and (2) weighted learning. For the first point, we employ unconstrained least squares importance fitting (uLSIF), which models the probability density ratio of the source domain against a target domain directly. Additionally, we propose weight only to the particular instance and using a linear kernel rather than a Gaussian kernel in uLSIF. For the second point, we employ a support vector machine (SVM) rather than the maximum entropy method (ME) that is commonly employed in weighted learning. Three corpora in the Balanced Corpus of Contemporary Written Japanese (BCCWJ) and 16 target words were used in our experiment. The experimental results show that the proposed method demonstrates the highest average precision.

## 1 Introduction

Supervised learning methods have been used in many natural language processing tasks. In supervised learning, we create training data for the target task from corpus A and learn a classifier from the training data. This classifier performs well for test data in corpus A; however, it does not perform well for test data in corpus B, which is different from corpus A. This is the problem of domain adaptation[1]. In this paper, we deal with domain adaptation for word sense disambiguation (WSD).

WSD identifies the sense $c \in C$ of an ambiguous word $w$ in a sentence $\boldsymbol{x}$. This problem can be solved by the following equation:

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

The above equation can be solved using supervised learning. However, the domain adaptation problem occurs in a real task. In domain adaptation, $P_s(c|\boldsymbol{x})$ can be derived from source domain $S$; therefore, we must estimate $P_t(c|\boldsymbol{x})$ in the target domain $T$ using $P_s(c|\boldsymbol{x})$ and other data. Note that the sense $c$ of the word $w$ in sentence $\boldsymbol{x}$ is not changed if sentence $\boldsymbol{x}$ appears in any domain corpus, i.e., $P(c|\boldsymbol{x})$ does not depend on a domain. As a result, $P_s(c|\boldsymbol{x}) = P_t(c|\boldsymbol{x})$. Therefore, it seems that we do not need to estimate $P_t(c|\boldsymbol{x})$ because we have $P_s(c|\boldsymbol{x})$. However, this is wrong because $P_s(\boldsymbol{x}) \neq P_t(\boldsymbol{x})$. The following assumption is referred to as the covariate shift:

$$P_s(\boldsymbol{x}) \neq P_t(\boldsymbol{x}), \;\; P_s(c|\boldsymbol{x}) = P_t(c|\boldsymbol{x}).$$

In other words, the domain adaptation for WSD satisfies the assumption of the covariate shift. In this paper, we solve domain adaptation for WSD by learning under covariate shift.

Briefly, learning under covariate shift is a learning method through weighted training data. Thus, it has

[1] Domain adaptation is considered as a type of transfer learning (Kamishima, 2010) in part of machine learning.



*29th Pacific Asia Conference on Language, Information and Computation: Posters, pages 215 - 223*
*Shanghai, China, October 30 - November 1, 2015*

two key points: (1) calculation of the weight of an instance and (2) weighted learning.

For the first point, the probability density ratio $w(\boldsymbol{x}) = P_t(\boldsymbol{x})/P_s(\boldsymbol{x})$ is used theoretically as the weight of the instance $\boldsymbol{x}$. There are two techniques for calculating the probability density ratio. The first is modeling $P_S(\boldsymbol{x})$ and $P_T(\boldsymbol{x})$ and then taking the ratio between them. The second is modeling $w(\boldsymbol{x})$ directly. Several studies have examined the former method (Jiang and Zhai, 2007)(Saiki et al., 2008). However, to the best of our knowledge, the latter approach has not been attempted in NLP research. In this paper, we adopt unconstrained least squares importance fitting (uLSIF) as the second calculation (Kanamori et al., 2009). Actually, there are many methods to calculate probability density ratio (Sugiyama and Kawanabe, 2011). In this paper, we use uLSIF because it shows good performance and quick calculation time. uLSIF models $w(\boldsymbol{x})$ with the sum of $N_t$ pieces of basis functions $\psi_l(\boldsymbol{x})$, where $N_t$ is the number of target data.

$$w(\boldsymbol{x}) = \sum_{l=1}^{N_t} \alpha_l \psi_l(\boldsymbol{x}).$$

Generally, a Gaussian kernel is used as the basis function. However, in this case, the width $\sigma$ of the Gaussian kernel becomes an additional parameter. Therefore, we suggest using a linear kernel to drop this parameter $\sigma$.

For the second point, the maximum entropy method (ME) is commonly employed in weighted learning. However, in domain adaptation for WSD, the number of instances is generally small. For this reason, we do not use a weighted ME but a weighted support vector machine (SVM).

Furthermore, three rough heaviness values are applied to the weighted SVM for comparison, i.e., a small weight 0.1, a normal weight 1.1, and a large weight 2.1, rather than a detailed weight for each case.

In the experiment, we use three domains, i.e., OC (Yahoo! Answer), PB (books) and PN (newspaper) in the Balanced Corpus of Contemporary Written Japanese (BCCWJ (Maekawa, 2007)) and 16 target words that appear frequently in these three domains. There are six types of domain adaptation - OC $\rightarrow$ PB, PB $\rightarrow$ PN, PN $\rightarrow$ OC, OC $\rightarrow$ PN, PN $\rightarrow$ PB, and PB $\rightarrow$ OC giving a total of 96 ($= 16 \times 6$) domain adaptation tasks. Consequently, the effects of the proposed method are confirmed.

## 2 Related Work

Generally, methods for domain adaptation can be divided into instances-based method and features-based method (Pan and Yang, 2010). The instances-based method is a learning method that gives weight to an instance of training data. Learning under covariate shift is typical method for this type. The features-based method is a method that maps the source and target features spaces to a common features space to maintain the important characteristics in each domain by reducing the difference between domains. The paper (Blitzer et al., 2006) proposed the dimension reduction method called structural correspondence learning (SCL). The paper (Pan et al., 2008) evaluated the distance between the spaces mapped in the source domain and the spaces mapped in the target domain by maximum mean discrepancy (MMD). They proposed a conversion method to minimize the distance called MMD embedding (MMDE). Moreover, the paper (Pan et al., 2011) improved MMDE and proposed a novel method called transfer component analysis (TCA). Adding weight to features is a features-based method. The paper (Daumé III, Hal, 2007) offered a weighting system for features. In this study, vector $\boldsymbol{x}_s$ of the training data in the source domain is mapped to an augmented input space $(\boldsymbol{x}_s, \boldsymbol{x}_s, \boldsymbol{0})$, and $\boldsymbol{x}_t$ is mapped to an augmented input space $(\boldsymbol{0}, \boldsymbol{x}_t, \boldsymbol{x}_t)$. The classifier that learned from the augmented vectors solves the classification problem by the usual method. Daumé's method assumes that an effect can be determined by overlapping the characteristics that are common to the source and target domains.

The domain adaptation problem is considered a data-sparse problem. Self-training and semi-supervised learning (Chapelle et al., 2006) and active learning (Settles, 2010) (Rai et al., 2010) are useful for domain adaptation.

At last, we introduce researches on domain adaptation for WSD. We assume that $P_t(c|\boldsymbol{x}) = P_s(c|\boldsymbol{x})$, but the assumption $P_t(\boldsymbol{x}|c) = P_s(\boldsymbol{x}|c)$ is also possible. Under this assumption, we can solve domain adaptation for WSD by estimating $P_t(c)$. Ac-

tually, the papers (Chan and Ng, 2006) and (Chan and Ng, 2005) estimated $P_t(c)$ by using EM algorithm to do it. The papers (Komiya and Okumura, 2012a), (Komiya and Okumura, 2011) and (Komiya and Okumura, 2012b) changed the learning method by the combination of source domain, target domain and target word. These studies are a kind of ensemble learning. In those learning methods, only the weight that is applied to data in source and target domain is different.

## 3 Domain Adaptation under Covariate Shift

In this section, we show that weighted learning can solve a domain adaptation problem under assumption of covariate shift.

We define the loss function as $l(\boldsymbol{x}, c, d)$ where $\boldsymbol{x}$, $c$ and $d$ denote an instance, the class of $\boldsymbol{x}$ and a classifier respectively. Thus, expected loss function $L_0$ in our task is expressed as the following:

$$L_0 = \sum_{\boldsymbol{x},c} l(\boldsymbol{x}, c, d) P_T(\boldsymbol{x}, c).$$

Through the assumption of covariate shift, we obtain the following:

$$\frac{P_T(\boldsymbol{x}, c)}{P_S(\boldsymbol{x}, c)} = \frac{P_T(\boldsymbol{x}) P_T(c|\boldsymbol{x})}{P_S(\boldsymbol{x}) P_S(c|\boldsymbol{x})} = \frac{P_T(\boldsymbol{x})}{P_S(\boldsymbol{x})}.$$

Now, $w(\boldsymbol{x}) = P_T(\boldsymbol{x})/P_S(\boldsymbol{x})$. It establishes as follows:

$$L_0 = \sum_{\boldsymbol{x},c} w(\boldsymbol{x}) l(\boldsymbol{x}, c, d) P_S(\boldsymbol{x}, c).$$

$D = \{(\boldsymbol{x}_i, c_i)\}_{i=1}^N$ denotes the training data. Using empirical distribution as a substitute for $P_S(\boldsymbol{x}, c)$, the following holds

$$L_0 \approx \frac{1}{N} \sum_{i=1}^{N} w(\boldsymbol{x}_i) l(\boldsymbol{x}_i, c_i, d).$$

In terms of expected loss minimization, find $d$ minimizing the following equation $L_1$ to solve the problem of covariate shift.

$$L_1 = \sum_{i=1}^{N} w(\boldsymbol{x}_i) l(\boldsymbol{x}_i, c_i, d). \tag{1}$$

Consider a classification based on a posterior probability maximizing estimation.

$$d(\boldsymbol{x}) = \arg\max_c P_T(c|\boldsymbol{x}).$$

Additionally, adapt a logarithmic loss as a loss function. Eq. (1) turn into the following:

$$L_1 = -\sum_{i=1}^{N} w(\boldsymbol{x}_i) \log P_T(c_i|\boldsymbol{x}_i).$$

In case of adopting an approach using a model of $P_T(c|\boldsymbol{x}, \boldsymbol{\lambda})$ in order to solve the classification problem, we find the parameter $\lambda$ maximizing the weighted log-likelihood $L(\boldsymbol{\lambda})$ of the following weighted by the probability density ratio in covariate shift.

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{N} w(\boldsymbol{x}_i) \log P_T(c_i|\boldsymbol{x}_i, \boldsymbol{\lambda}). \tag{2}$$

For above problem, Maximum Entropy Method (ME) is commonly used as a model.

$$P_T(c|\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\lambda})} \exp\left(\sum_{j=1}^{M} \lambda_j f_j(\boldsymbol{x}, c)\right), \tag{3}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_M)$. The function $f_j(\boldsymbol{x}, c)$ is a feature function. It returns $x_j$ when the true class is defined $c$, and it returns 0 in other cases. $Z(\boldsymbol{x}, \boldsymbol{\lambda})$ is a normalization term. Hence, we obtain

$$Z(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{c \in C} \exp\left(\sum_{j=1}^{M} \lambda_j f_j(\boldsymbol{x}, c)\right), \tag{4}$$

where $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_M)$ is a vector of weight parameters for features.

## 4 Weight through Probability Density Ratio

There are two kinds of approaches estimating the probability density ratio $w(\boldsymbol{x}) = P_T(\boldsymbol{x})/P_S(\boldsymbol{x})$. The first approach is estimating each $P_S(\boldsymbol{x})$ and $P_T(\boldsymbol{x})$, and take their ratio. The second approach is modeling $w(\boldsymbol{x})$, directly.

In this paper, we use unconstrained least-squares importance fitting (uLSIF) proposed in (Kanamori et al., 2009) as the second approach.

### 4.1 uLSIF

$\{\boldsymbol{x}_i^s\}_{i=1}^{N_s}$ and $\{\boldsymbol{x}_i^t\}_{i=1}^{N_t}$ denote a source data and a target data, respectively. In uLSIF, the probability density ratio is modeled as the following:

$$w(\boldsymbol{x}) = \sum_{l=1}^{N_t} \alpha_l \psi_l(\boldsymbol{x}) = \boldsymbol{\alpha} \cdot \boldsymbol{\psi}(\boldsymbol{x}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{N_t})$, $\boldsymbol{\psi}(\boldsymbol{x}) = (\psi_1(\boldsymbol{x}), \psi_2(\boldsymbol{x}), \cdots, \psi_b(\boldsymbol{x}))$. and $\alpha_l > 0$. Here, $\psi_l(\boldsymbol{x})$ is a basis function which is mapping from the source data to the positive real number.

In uLSIF, actually, the parameter $\boldsymbol{\alpha}$ is estimated after building the basis function $\boldsymbol{\psi}(\boldsymbol{x})$. However, for the convenience of description, we firstly explain the estimation of $\boldsymbol{\alpha}$. $\hat{w}(\boldsymbol{x})$ denotes a model of $w(\boldsymbol{x})$. In order to estimate parameter $\alpha_l$, we find $\hat{\alpha}$ minimizing a mean square error $J_0(\boldsymbol{\alpha})$ between $w(\boldsymbol{x})$ and $\hat{w}(\boldsymbol{x})$. By taking account of $w(\boldsymbol{x}) = P_T(\boldsymbol{x})/P_S(\boldsymbol{x})$, $J_0(\boldsymbol{\alpha})$ can be transformed as follows:

$$\begin{aligned}
J_0(\boldsymbol{\alpha}) &= \frac{1}{2}\int (\hat{w}(\boldsymbol{x}) - w(\boldsymbol{x}))^2 P_S(\boldsymbol{x}) d\boldsymbol{x} \\
&= \frac{1}{2}\int \hat{w}(\boldsymbol{x})^2 P_S(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad - \int \hat{w}(\boldsymbol{x}) w(\boldsymbol{x}) P_S(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad + \frac{1}{2}\int w(\boldsymbol{x})^2 P_S(\boldsymbol{x}) d\boldsymbol{x} \\
&= \frac{1}{2}\int \hat{w}(\boldsymbol{x})^2 P_S(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad - \int \hat{w}(\boldsymbol{x}) P_T(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad + \frac{1}{2}\int w(\boldsymbol{x})^2 P_S(\boldsymbol{x}) d\boldsymbol{x}.
\end{aligned}$$

Since the third term is constant, so it is independent on minimizing $J_0(\boldsymbol{\alpha})$. Therefore, minimizing $J_0(\boldsymbol{\alpha})$ means minimizing the following $J(\boldsymbol{\alpha})$.

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\int \hat{w}(\boldsymbol{x})^2 P_S(\boldsymbol{x}) d\boldsymbol{x} - \int \hat{w}(\boldsymbol{x}) P_T(\boldsymbol{x}) d\boldsymbol{x}.$$

By approximating $P_S(\boldsymbol{x})$ and $P_T(\boldsymbol{x})$ by empirical distributions, $J(\boldsymbol{\alpha})$ is transformed as the following $\widehat{J}(\boldsymbol{\alpha})$:

$$\begin{aligned}
\widehat{J}(\boldsymbol{\alpha}) &= \frac{1}{2N_s}\sum_{i=1}^{N_s} \widehat{w}(\boldsymbol{x}_i^s)^2 - \frac{1}{N_t}\sum_{j=1}^{N_t} \widehat{w}(\boldsymbol{x}_j^t) \\
&= \frac{1}{2}\sum_{l,l'=1}^{N_t} \alpha_l \alpha_{l'} \left( \frac{1}{N_s}\sum_{i=1}^{N_s} \psi_l(\boldsymbol{x}_i^s)\psi_{l'}(\boldsymbol{x}_i^s) \right) \\
&\quad - \sum_{l=1}^{N_t} \alpha_l \left( \frac{1}{N_t}\sum_{j=1}^{N_t} \psi_l(\boldsymbol{x}_j^t) \right) \\
&= \frac{1}{2}\boldsymbol{\alpha}^T \widehat{H} \boldsymbol{\alpha} - \widehat{h}^T \boldsymbol{\alpha}, \qquad (5)
\end{aligned}$$

where $\widehat{H}$ denotes $N_t \times N_t$ matrix, and $\widehat{H}_{l,l'}$ (the $(l, l')$ element of $\hat{H}$) is defined as follows:

$$\widehat{H}_{l,l'} = \frac{1}{N_s}\sum_{i=1}^{N_s} \psi_l(\boldsymbol{x}_i^s)\psi_{l'}(\boldsymbol{x}_i^s)$$

Furthermore, $\widehat{h}$ denotes $N_t$-dimensional vector, and the element of the $l$-th dimension $\widehat{h}_l$ is defined as follows:

$$\widehat{h}_l = \frac{1}{N_t}\sum_{j=1}^{N_t} \psi_l(\boldsymbol{x}_j^t).$$

As the result, we can obtain the $\hat{\alpha}$ minimizing $\widehat{J}(\boldsymbol{\alpha})$ by solving the following problem:

$$\min_{\boldsymbol{\alpha}} \left[ \frac{1}{2}\boldsymbol{\alpha}^T \widehat{H} \boldsymbol{\alpha} - \widehat{h}^T \boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^T \boldsymbol{\alpha} \right].$$

Here, we must note that the parameter $\lambda$ is added. The above minimization problem is unconstrained convex quadratic programming problem without a constrained condition, so that we obtain a global solution:

$$\tilde{\boldsymbol{\alpha}} = (\widehat{H} + \lambda I_{N_t})^{-1} \widehat{h}^T. \qquad (6)$$

Lastly, conduct the following adjustment to satisfy the condition $\boldsymbol{\alpha} > 0$:

$$\begin{aligned}
\widehat{\boldsymbol{\alpha}} &= ((\max(0, \tilde{\alpha_1}), \max(0, \tilde{\alpha_2}), \cdots, \max(0, \tilde{\alpha_{N_t}})) \\
&= \max(0_{N_t}, \tilde{\boldsymbol{\alpha}}). \qquad (7)
\end{aligned}$$

In genaral, a Gaussian kernel is used as the basis function.

$$\psi_l(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_l^t) = \exp\left( -\frac{||\boldsymbol{x} - \boldsymbol{x}_l^t||^2}{\sigma^2} \right).$$

Under this situation, remaining parameters to be determined are the regularization term $\lambda$ and a width of the Gaussian kernel $\sigma$ to obtain the probability

density ratio. These parameters are found by a cross-validation of a grid search. First, split each source and target data into $R$ pieces of subset with no intersection. Secondly, exclude the $r$-th subset from these subsets, and bind the rest. These data are regarded as a new source and target domain. Now, set certain values to $\lambda$ and $\sigma$, and obtain $\boldsymbol{\alpha}$ with Eq. (6) and Eq. (7), and find $\widehat{J}(\boldsymbol{\alpha})^{(r)}$ with Eq. (5). Calculate $R$ pieces of values of $\widehat{J}(\boldsymbol{\alpha})^{(r)}$ by varying the value $r$ from 1 to $R$, and regard the average value of them as $\widehat{J}(\boldsymbol{\alpha})$ for $\lambda$ and $\sigma$. Next, estimate $\lambda$ and $\sigma$ minimizing $\widehat{J}(\boldsymbol{\alpha})$ obtained with the above procedure by changing the values of $\lambda$ and $\sigma$. These values are denoted by $\hat{\lambda}$ and $\hat{\sigma}$, respectively.

### 4.2 Use of Linear Kernel instead of Gaussian Kernel

In this paper, we use linear kernel as the basis function in uLSIF instead of the Gaussian kernel. By this use, we can drop the parameter $\sigma$.

Generally, a kernel function is to map to non-linear high-dimensional space. However, in our tasks, the number of features is larger than the number of instances, so that there is no need to map to the high-dimensional space. In this case, calculation to adjust the parameter is easier than using Gaussian kernel.

$$\psi_l(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_l^t) = \boldsymbol{x} \cdot \boldsymbol{x}_l^t.$$

### 4.3 Weight of Particular Instances

In our task, that is domain adaptations of WSD, we must construct the model of the probability density for each target word. Additionally, the number of instances of the target word is too small compared to the number of the feature dimension in both source and target domain. Therefore, an estimated probability density ratio tends to be smaller than the true value, so that some approaches to close the estimated probability density ratio to 1 have been proposed. Sugiyama translated to the weight $w$ to the weight $w^p$ $(0 < p < 1)$ (Sugiyama, 2006), and Yamada proposed the relative probability density ratio (Yamada et al., 2011):

$$\frac{P_T(\boldsymbol{x})}{\alpha P_T(\boldsymbol{x}) + (1-\alpha) P_S(\boldsymbol{x})}.$$

These methods have an effect to close the original probability density ratio to 1.

Here, we assign the rough weight to the instance according to the estimated probability density ratio. The rough weight has 3 kinds of value: 0.1, 1.1 and 2.1.

The set of the estimated probability density ratio is expressed as $W = \{w_i\}_{i=1}^N$. The $w_i$ is normalized by the following:

$$w_i' = \frac{w_i - \mu}{\sigma},$$

where $\mu$ and $\sigma^2$ denote the mean and the variance of $W$, respectively.

Assuming that $W$ follows a normal distribution, $w_i$ is defined as 2.1 when $w_i$ is greater than 0.84, $w_i$ is defined as 0.1 when $w_i$ is smaller than -0.84, and in the other cases, $w_i$ is defined as 1.1.

The points, 0.84 and -0.84, are 20% top and lower quantile points of normal distribution, respectively.

## 5 SVM for weighted learning

Learning under covariate shift means the weighted learning using the probability density ratio. After assigning weight to each instance, we apply the weighted learning method. Conventionally we use ME and logistic regression as the the weighted learning method. However, a method based on a loss function is also available, as can be seen from the Eq. (1). In this paper, we use the method of SVM for imbalanced data (Tang et al., 2009). Through the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ $(\boldsymbol{x}_i \in R^d, y_i \in \{1, -1\})$, SVM is constructed by estimating parameters $\boldsymbol{w}$, $b$, and $\boldsymbol{\zeta}$ in the following:

$$\min_{\boldsymbol{w},b,\boldsymbol{\zeta}} \left\{ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^N \zeta_i \right\}. \tag{8}$$

Now,

$$y_i(\boldsymbol{w}^T\phi(x_i) + b) \geq 1 - \zeta_i\,, \quad \zeta_i \geq 0.$$

In the above formula, we can use the weighted SVM by using $w(\boldsymbol{x}_i)C$ instead of C (Cortes and Vapnik, 1995).

## 6 Experiment

In this paper, we chose three domains, OC (Yahoo! Answer), PB (books), and PN (newspaper) in the BCCWJ, and 16 target words with enough frequency

Table 1: Target words

| word | dictionary # of senses | OC freq. of word | OC # of senses | PB freq. of word | PB # of senses | PN freq. of word | PN # of senses |
|---|---|---|---|---|---|---|---|
| iu(言う) | 3 | 666 | 2 | 1114 | 2 | 363 | 2 |
| ireru(入れる) | 3 | 73 | 2 | 56 | 3 | 32 | 2 |
| kaku(書く) | 2 | 99 | 2 | 62 | 2 | 27 | 2 |
| kiku(聞く) | 3 | 124 | 2 | 123 | 2 | 52 | 2 |
| kodomo(子供) | 2 | 77 | 2 | 93 | 2 | 29 | 2 |
| jikan(時間) | 4 | 53 | 2 | 74 | 2 | 59 | 2 |
| jibun(自分) | 2 | 128 | 2 | 308 | 2 | 71 | 2 |
| deru(出る) | 3 | 131 | 3 | 152 | 3 | 89 | 3 |
| toru(取る) | 8 | 61 | 7 | 81 | 7 | 43 | 7 |
| baai(場合) | 2 | 126 | 2 | 137 | 2 | 73 | 2 |
| hairu(入る) | 3 | 68 | 4 | 118 | 4 | 65 | 3 |
| mae(前) | 3 | 105 | 3 | 160 | 2 | 106 | 4 |
| miru(見る) | 6 | 262 | 5 | 273 | 6 | 87 | 3 |
| motsu(持つ) | 4 | 62 | 4 | 153 | 3 | 59 | 3 |
| yaru(やる) | 5 | 117 | 3 | 156 | 4 | 27 | 2 |
| yuku(ゆく) | 2 | 219 | 2 | 133 | 2 | 27 | 2 |
| average | 3.44 | 148.19 | 2.94 | 199.56 | 3.00 | 75.56 | 2.69 |

in all three domains. Now, we have six types of domain adaptation: OC → PB, PB → PN, PN → OC, OC → PN, PN → PB, and PB → OC. Therefore, totally 96 ($= 6 \times 16$) kinds of domain adaptation tasks is set.

Table 1 indicates information of the target word, the number of senses registered in the dictionary, and the number of senses and the frequency in each corpus [2].

Here, we explain how to evaluate a domain adaptation method for our tasks. First, by using a domain adaptation method (named as `Mtd-A`), a classifier for a target word $w_i$ in a domain adaptain $S \rightarrow T$ is obtained. We can get the precision $p_{w_i}^{(ST)}$ of this classifier under this setting. Thus, given domain adaptain $S \rightarrow T$, we can get the average precision $p^{(ST)}$ for the 16 target words ($w_1, w_2, \cdots, w_{16}$). By this $p^{(ST)}$, we evalutate the method `Mtd-A` for the domain adaptain $S \rightarrow T$. We have 6 types of $S \rightarrow T$, so 6 $p^{(ST)}$ are obtained. We evaluate the method `Mtd-A` by taking average of 6 $p^{(ST)}$.

The method `Mtd-A` is composed of two approaches, a method of calculating the probability density ratio and type of the weighted learning. In this paper, 10 kinds of method are examined. Base-M and Base-S, are approaches without uLSIF. The characters, "M" and "S", mean ME and SVM, respectively. The other techniques are with uLSIF. The letters, "G" and "L" signify the Gaussian kernel and the linear kernel used in uLSIF, respectively. In addition, in Ours-*-* , convert weight into three types of weight (0.1, 1.1, and 2.1) depending on the probability density ratio calculated with uLSIF. Our proposed method is expressed as "Ours-L-S".

Results of the experiments are shown in Table. 2. As the result, relationships, `Base-M < Base-S`, `Mtd-G-M < Mtd-G-S`, `Mtd-L-M < Mtd-L-S`, `Ours-G-M < Ours-G-S`, and `Ours-L-M < Ours-L-S` are satisfied. It is found that SVM is more effective than ME. Additionally, relations, `Mtd-G-M < Mtd-L-M`, `Mtd-G-S = Mtd-L-S`, ans `Ours-G-S < Mtd-L-S` are established. The results of `Ours-G-M` and `Ours-L-M` are almost the same. Therefore, the linear kernel has better effectiveness than the Gaussian kernel. The proposed method `Ours-L-S` in this paper has the highest average accuracy rate. In each domain adaptation, it shows the highest accuracy rate, excluding `PN` → `PB`.

Here, we must note that the difference between our proposed method (`Ours-L-S`) and the baseline (`Base-S`) is slight, and we could not get statistical sig-

[2] The word "入る (hairu)" has three senses in the dictionary, but there are four senses in OC and PB. This is because our used sense tagged corpus accepts new senses.

Table 2: Experimental results (average precisions)

| | OC → PB | PB → PN | PN → OC | OC → PN | PN → PB | PB → OC | Average |
|---|---|---|---|---|---|---|---|
| Base-M | 0.7163 | 0.7700 | 0.6920 | 0.6778 | **0.7474** | 0.6991 | 0.7171 |
| Base-S | 0.7141 | 0.7676 | 0.6907 | 0.6880 | 0.7452 | 0.7011 | 0.7178 |
| Mtd-G-M | 0.7008 | 0.7289 | 0.6854 | 0.6840 | 0.7110 | 0.6760 | 0.6977 |
| Mtd-G-S | 0.7143 | 0.7692 | 0.6903 | 0.6900 | 0.7455 | 0.7034 | 0.7189 |
| Mtd-L-M | 0.7145 | 0.7339 | 0.6907 | 0.6887 | 0.7144 | 0.7008 | 0.7055 |
| Mtd-L-S | 0.7134 | 0.7699 | 0.6905 | 0.6898 | 0.7450 | 0.7045 | 0.7189 |
| Ours-G-M | 0.7145 | 0.7670 | 0.6907 | 0.6787 | 0.7446 | 0.7008 | 0.7160 |
| Ours-G-S | 0.7129 | 0.7707 | 0.6911 | 0.6884 | 0.7451 | 0.7021 | 0.7184 |
| Ours-L-M | 0.7145 | 0.7665 | 0.6907 | 0.6787 | 0.7445 | 0.7008 | 0.7159 |
| Ours-L-S (Proposed Method) | **0.7197** | **0.7723** | **0.6971** | **0.6936** | 0.7416 | **0.7062** | **0.7218** |

nificance. However, without taking account on the PN domain, our proposed method is statistical significant for the baseline. Further the use of weighted SVM is also significant for the use of weighted ME. From these points, our proposed method has its value.

## 7 Discussion

### 7.1 Effectiveness of Small and Large Weights

Our proposed method converts the weight estimated by uLSIF to 0.1, 1.1 or 2.1 according to the volume of the weight. In this section, we investigate which is effective small weight 0.1 or large weight 2.1. To do it, we modify our proposed method by following two cases: (case.1) In our method, we change the weight 2.1 to the normal weight 1.1, and other weights are not changed. (case 2) In our method, we change the weight 0.1 to the normal weight 1.1, and other weights are not changed.

We conducted experiments of the above two modification. The result is shown in Table 3. "Ours-L-S-small" and "Ours-L-S-large" in Table 3 denote (case 1) and (case 2), respectively.

This result shows that small weight 0.1 is more effective than large weight 2.1 in our proposed method, because the (case 2) is worse than baseline but the (case 1) is better than baseline. However, our proposed method is better than the (case 1). That is, the use of both weights is more effective than only small weight or only large weight.

Table 3: Importance of instances

| | Average |
|---|---|
| Base-S | 0.7178 |
| Ours-L-S | 0.7218 |
| Ours-L-S-small (only small weight, case 1) | 0.7183 |
| Ours-L-S-large (only large weight, case 2) | 0.7176 |

### 7.2 Deletion of Misleading Data

In the previous section, we mentioned that small weight is effective, that is, it is effective to decrease the weight of unimportant training data in our task. The reason comes from that there are misleading data in the training data. Misleading data is a problem of domain adaptation. In domain adaptation, some data in training data decrease the precision of the classifier. These data is called misleading data (Jiang and Zhai, 2007).

In this section, we discuss the relation of our proposed method and misleading data. First, we confirm the presence of misleading data in training data. To do it, Yoshida (Yoshida and Shinnou, 2014) checked each training data is misleading data or not one by one.

Here, we introduce the above Yoshida's method. In the domain adaptation from the source domain $S$ to the target domain $T$, labeled data $D$ in $S$ of the target word $w$ exits. First, we measure the precision $p_0$ for $T$ by the classifier learned through $D$. Second, we remove a data $x$ from $D$ and measure the precision $p_1$ for $T$ by the classifier learned through $D - x$. In the case of $p_1 > p_0$, the data $x$ is regarded as the misleading data. We apply this procedure for all data in $D$ to find the misleading data of the target word $w$ in the domain adaptation from $S$ to $T$. The result of the number of misleading data is shown in Table 4. The number in parentheses is the total number of the training data. Note that this method uses lables in $T$, so it cannot detect misleading data. This

Table 4: Number of the misleading data

| word | OC → PB | PB → PN | PN → OC | OC → PN | PN → PB | PB → OC |
|---|---|---|---|---|---|---|
| iu(言う) | 159 (666) | 75 (1114) | 82 (363) | 158 (666) | 35 (363) | 127 (1114) |
| ireru(入れる) | 6 (73) | 15 (56) | 3 (32) | 28 (73) | 1 (32) | 19 (56) |
| kaku(書く) | 21 (99) | 2 (62) | 12 (27) | 39 (99) | 15 (27) | 0 (62) |
| kiku(聞く) | 26 (124) | 0 (123) | 4 (52) | 21 (124) | 27 (52) | 26 (123) |
| kodomo(子供) | 5 (77) | 1 (93) | 12 (29) | 0 (77) | 13 (29) | 12 (93) |
| jikan(時間) | 1 (53) | 0 (74) | 0 (59) | 8 (53) | 5 (59) | 0 (74) |
| jibun(自分) | 13 (128) | 0 (308) | 0 (71) | 25 (128) | 1 (71) | 0 (308) |
| deru(出る) | 14 (131) | 32 (152) | 22 (89) | 10 (131) | 10 (89) | 39 (152) |
| toru(取る) | 6 (61) | 18 (81) | 12 (43) | 5 (61) | 22 (43) | 10 (81) |
| baai(場合) | 0 (126) | 13 (137) | 14 (73) | 0 (126) | 9 (73) | 7 (137) |
| hairu(入る) | 36 (68) | 27 (118) | 27 (65) | 11 (68) | 42 (65) | 38 (118) |
| mae(前) | 8 (105) | 1 (160) | 15 (106) | 5 (105) | 2 (106) | 10 (160) |
| miru(見る) | 10 (262) | 12 (273) | 8 (87) | 3 (262) | 28 (87) | 3 (273) |
| motsu(持つ) | 8 (62) | 11 (153) | 1 (59) | 0 (62) | 1 (59) | 2 (153) |
| yaru(やる) | 0 (117) | 0 (156) | 0 (27) | 0 (117) | 0 (27) | 0 (156) |
| yuku(ゆく) | 17 (219) | 1 (133) | 3 (27) | 0 (219) | 3 (27) | 15 (133) |

Table 5: Deletion of the misleading data

| | OC → PB | PB → PN | PN → OC | OC → PN | PN → PB | PB → OC | Average |
|---|---|---|---|---|---|---|---|
| Base-S | 0.7141 | 0.7676 | 0.6907 | 0.6880 | 0.7452 | 0.7011 | 0.7178 |
| Ours-L-S | 0.7197 | 0.7723 | 0.6971 | 0.6936 | 0.7416 | 0.7062 | 0.7218 |
| Mislead | 0.7459 | 0.7927 | 0.7450 | 0.7213 | 0.7869 | 0.7334 | 0.7542 |
| Mislead2 | 0.7117 | 0.7627 | 0.6833 | 0.6920 | 0.7399 | 0.6984 | 0.7146 |

method is just to confirm the presence of misleading data.

The result of SVM using the training data without misleading data is shown in Table 5. “Mislead” in Table 5 denotes the average accuracy rate. This result is highest in our experiments. To remove of the misleading data is to assign the weight of the data to 0. Therefore, it is possible to improve the precision by just adjusting the weight.

Now, we conduct the experiment that the data with quite small probability density ratio is regarded as the misleading data whose weight is 0. The “Mislead2” in Table 5 shows the result. However, this approach is not effective. Probably, we cannot detect the misleading data using only the probability density ratio. The method to detect misleading data is our future work.

## 8 Conclusion

We have solved domain adaptation for WSD by learning under covariate shift. This learning has two key points: (1) calculation of the weight of an instance and (2) weighted learning. For the first point, we used uLSIF and improved it by weighting only the particular instances and by using a linear rather than a Gaussian kernel in uLSIF. For the second point, we used a weighted SVM rather than the commonly used weighted ME.

Three corpora in BCCWJ and 16 target words (96 domain adaptation tasks) were used in our experiment. This experimental results show that the proposed method demonstrates the highest average precision. The proposed method is statistically significant for the baseline without considering the PN domain. In addition, the use of the weighted SVM is significant for the weighted ME.

In future, we will investigate why weighted learning does not work well for the PN → PB domain adaptation.

## References


John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP-2006*, pages 120–128.

Yee Seng Chan and Hwee Tou Ng. 2005. Word sense disambiguation with distribution estimation. *IJCAI-05*.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *COLING-ACL-2006*, pages 89–96.

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Daumé III, Hal. 2007. Frustratingly Easy Domain Adaptation. In *ACL-2007*, pages 256–263.

Jing Jiang and Chengxiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL-2007*, pages 264–271.

Toshihiro Kamishima. 2010. Transfer learning (in japanese). *The Japanese Society for Artificial Intelligence*, 25(4):572–580.

Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. 2009. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445.

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

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

Kanako Komiya and Manabu Okumura. 2012b. Automatic selection of domain adaptation method for wsd using decision tree learning (in japanese). *Journal of NLP*, 19(3):143–166.

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

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Sinno Jialin Pan, James T Kwok, and Qiang Yang. 2008. Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210.

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

Yosuke Saiki, Hiroya Takamura, and Manabu Okumura. 2008. Domain adaptation in sentiment classification by instance weighting (in japanese). *IPSJ SIG Technical Report. SIG-NL Report*, 2008(33):61–67.

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

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

Masashi Sugiyama. 2006. Supervised learning under covairant shift (in japanese). *Japanese Neural Network Society*, 13(3):111–118.

Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. 2009. SVMs modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288.

Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. 2011. Relative density-ratio estimation for robust distribution comparison. *Neural Computation*, 25(5):1370–1370.

Hiromu Yoshida and Hiroyuki Shinnou. 2014. Detection of misleading data by outlier detection methods (in japanese). In *The 5th Japanese Corpus Linguistics Workshop*, pages 49–56.