2009 Ninth IEEE International Conference on Data Mining

TrBagg: A Simple Transfer Learning Method and Its Application to Personalization in Collaborative Tagging

Toshihiro Kamishima, Masahiro Hamasaki, and Shotaro Akaho
National Institute of Advanced Industrial Science and Technology (AIST)
AIST Tsukuba Central 2, Umezono 1–1–1, Tsukuba, Ibaraki, 305–8568 Japan
mail@kamishima.net ⟨http://www.kamishima.net/⟩, hamasaki@ni.aist.go.jp, s.akaho@aist.go.jp

***Abstract*—The aim of transfer learning is to improve prediction accuracy on a target task by exploiting the training examples for tasks that are related to the target one. Transfer learning has received more attention in recent years, because this technique is considered to be helpful in reducing the cost of labeling. In this paper, we propose a very simple approach to transfer learning: TrBagg, which is the extension of bagging. TrBagg is composed of two stages: Many weak classifiers are first generated as in standard bagging, and these classifiers are then filtered based on their usefulness for the target task. This simplicity makes it easy to work reasonably well without severe tuning of learning parameters. Further, our algorithm equips an algorithmic scheme to avoid negative transfer. We applied TrBagg to personalized tag prediction tasks for social bookmarks Our approach has several convenient characteristics for this task such as adaptation to multiple tasks with low computational cost.**

***Keywords*-transfer learning; bagging, ensemble learning, personalization, collaborative tagging**

I. Introduction

In this paper, we propose a new transfer learning method, *TrBagg*, which is a modification of bagging [1]. This method is applied to a task for personalization of tags in collaborative tagging.

Transfer Learning [2] has not been formally defined, but it refers broadly to "the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task[1]." Though the idea of transfer learning has been in existence since the late 1990s [3], research in this area has become active recently. This is mainly because transfer learning is considered a promising solution to the issue of labeling costs. Machine learning algorithms generally assume that training and test data are sampled from a distribution of the same domain. Therefore, observed data for new tasks must be labeled to apply machine learning algorithms. However, because data are generally labeled by hand, labeling requires a lot of time and effort. Further, the consistency of the labeling criteria and the amount of labeled data greatly affect the predictive performance of the learned rules of functions. For these reasons, the cost of labeling data is a serious obstacle to applying machine learning techniques. One way to alleviate this obstacle is to exploit existing data. Though labeled data may not exist for the exact target task that we want to solve, we are frequently able to access data for tasks related to the target one. For example, in natural language processing, a corpus of related domains is available [4], [5]. Data for past time periods can be used when analyzing the latest time period [6]. Transfer learning techniques enable us to exploit these related data for solving a target task with far lower labeling costs.

We here propose a simple algorithm, TrBagg. This algorithm is composed of two phases: learning and filtering. In the learning phase, as in standard bagging, training data sets are bootstrap-sampled from the target and source data, and weak classifiers are learned from these training sets. Some of these training sets consist of data that are largely useful for the target task, while others do not. If the weak classifiers are learned from useful data for the target task, they are selected in the filtering phase. To achieve this filtering, we assume that some such classifiers would accurately discriminate the given target data, while other weak classifiers would be abandoned if their accuracies on the target data are low. TrBagg possesses the following valuable characteristics. Due to its simplicity, TrBagg works reasonably well, if learning parameters are not severely tuned. One serious issue in transfer learning is *negative transfer* [7]. In order to alleviate this issue, our TrBagg is equipped with an algorithmic scheme that introduces a fallback classifier. When transferring source knowledge to multiple domains, the computation in the learning phase can be shared among these domains; thus, computational costs are largely saved. Another merit of TrBagg is the capability of learning weak classifiers in parallel. This is an enormously helpful property for processing very large data sets. This TrBagg is designed under the assumption that the target data are consistently labeled based on the criterion of the target task. Or, rather: some parts of the related source data are labeled based on the target criterion, but the rest of the data are not. That is to say, the source data set is a mixture of data representing target and non-target concepts.

The prediction of the tag personalization in collaborative tagging systems is one example of a task in which the above

[1]From the announcement of the "NIPS 2005 Workshop, Inductive Transfer: 10 Years Later"

1550-4786/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDM.2009.9

assumption is reasonable. Collaborative tagging services enable users to register their favorite Web pages. Users can assign tags to these registered pages, and these pages and tags can be shared among users of the service. With these shared tags, users can search pages that other users have registered, and they can find like-minded users. Social tags are labeled based on each user's personal criterion; thus, the semantics of social tags can vary greatly. Consequently, tags labeled by one user may be inappropriate for another user. When searching for documents with shared tags, users might find undesired documents or miss relevant pages. To cope with this difficulty, tags are personalized for a specific target user. By using tags labeled by the target user as training examples, classification techniques make it possible to predict tags for new Web pages that are appropriate from the user's viewpoint. However, the number of tags that are labeled by one user is generally too small for training reliable classifiers. The transfer learning approach is effective for addressing this problem, because it enables the utilization of larger numbers of tags labeled by non-target users. The target user's tags and non-target users' tags are treated as the target and source data, respectively. Presumably, some of the non-target users would label Web pages based on the same criterion as the target user's, while others would not. In short, this source data set would be a mixture of useful and useless data. Consequently, we can say that the assumption on which TrBagg is based would be reasonable for this personalization task. We will later show the experimental results of the application of our TrBagg to this task.

Our contributions can be summarized as follows: We have proposed a simple transfer learning algorithm, TrBagg, which worked reasonably well without severe parameter tuning and was less affected by the problem of negative transfer than other transfer learning algorithms. Using this algorithm, we have shown that tags can be personalized in collaborative tagging services.

In section II, we introduce our TrBagg algorithm. Section III describes a tag personalization task in collaborative filtering. Section IV shows the experimental results on real tagging data. After discussing the related work in section V, we conclude in section VI.

II. Transfer Learning and TrBagg

Here we address the transfer learning problem and introduce the details of our TrBagg, which is developed by modifying bagging so as to be applicable for transfer learning.

A. Transfer Learning

We here state the transfer learning problem in the form that is used in this paper. A more inclusive discussion of transfer learning is postponed until section V.

An object is represented by a feature vector, $\mathbf{x}$. The variable c denotes the class in which the object should be classified. The pair of a class and object, $(c_i, \mathbf{x}_i)$, is a training example. The goal of classification is to acquire classifiers that can predict an appropriate class for any input feature vector, $\mathbf{x}$, from a set of training examples. A standard classifier is learned from only one homogeneous set of training examples. These training examples are assumed to be sampled from a distribution, $\mathrm{P}[c, \mathbf{x}]$, which expresses the target concept to be learned. On the other hand, in the case of transfer learning, two types of training examples, target and source, are given. Similar to the case of standard classification, target data are assumed to be sampled from a distribution, $\mathrm{P}[c, \mathbf{x}]$, that corresponds to the target concept. This set of target data is denoted by $\mathcal{D}_T = \{(c_i, \mathbf{x}_i)\}_{i=1}^{N_T}$, where $N_T = |\mathcal{D}_T|$, and the source data set is denoted by $\mathcal{D}_S = \{(c_i, \mathbf{x}_i)\}_{i=1}^{N_S}$, where $N_S = |\mathcal{D}_S|$. Regarding these target and source data sets, we introduce three assumptions: First, the source data are sampled from distributions that express both consistent and irrelevant concepts. Second, the number of examples of the target concept is as at least a few times as large as N_T, and the learner does not know which examples express the target concept. Third, the size of the source data set is much larger than that of the target set, i.e., $N_S \gg N_T$.

These target and source data are mutually complementary. That is to say, target data are consistent with the target concept and are relatively few, while source data are less reliable and more abundant. If we can extract information about the target concept from a target data set and can exploit the information along with information from the source data, the learner can know more about the target concept. The goal of transfer learning is to acquire classifiers that can more accurately predict classes by exploiting the information in source data.

B. Bagging

To solve the above problem, we developed a *TrBagg* algorithm (**Tr**ansfer **Bagg**ing.) Before showing our TrBagg, we briefly describe the original bagging method; our algorithm is a modified version of the original method [1]. In the bagging algorithm, the following steps are iterated for $t = 1, \ldots, T$.

1) Obtain a training example set $\mathcal{D}_t$ by bootstrap sampling, that is, sampling with replacement, from a given training example set $\mathcal{D}$.
2) From this training set $\mathcal{D}_t$, learn a weak classifier $\hat{f}_t$ by using an arbitrary classification algorithm.

By this procedure, a set of T weak classifiers, $\mathcal{F} = \{\hat{f}_1, \ldots, \hat{f}_T\}$, can be acquired. To classify a new feature vector, this vector is classified by these T classifiers, and these classification results are aggregated by majority voting. Formally, the class to which the feature vector $\mathbf{x}$ should

belong is determined by

$$\hat{c} = \arg\max_{c \in \mathcal{C}} \sum_{\hat{f}_t \in \mathcal{F}} \mathrm{I}[c = \hat{f}_t(\mathbf{x})], \quad (1)$$

where $\mathrm{I}[cond]$ is an indicator function, which becomes 1 if the condition $cond$ is true and 0 otherwise, and $\mathcal{C}$ denotes the domain of classes.

The reason why this bagging improves the prediction accuracy has been explained simply based on the bias-variance theory [1], [8]. According to the bias-variance theory, a generalization error is divided into three factors: the bias, which is derived from the choice of models, the variance, which is derived from the variation in sampling of training examples, and the intrinsic error, which cannot be avoided. If a low-bias model, which can approximate various forms of functions, is used for learning, the effect of the bias factor can be lessened, while that of the variance factor has a greater effect on the generalization error. Inversely, in the case of a high-bias model, the degrees of effect caused by the bias and variance factors are exchanged. In the process of bagging, various training example sets are generated, weak classifiers are learned from each example set, and these classifiers are aggregated. Because this aggregated classifier has been trained using various training sets, the variance factor can be lessened without increasing the error caused by the bias factor. Therefore, if a low-bias model is adopted, the bagging technique contributes to reducing the generalization error. However, if a high-bias model such as Fisher's discriminant analysis is used, bagging fails to reduce the prediction error, because the ratio of the error caused by the variance factor is originally small.

C. TrBagg

We developed our TrBagg algorithm by modifying this bagging procedure. In [8, section 7], an improvement technique for bagging is reported. In standard bagging, training examples are bootstrap-sampled from a given example set, and the sampled examples are fed to a weak learner without modification. Instead, by adding noise to the feature vectors in the sampled examples, the prediction accuracy can be improved. This is because more varied weak classifiers can be learned by adding noise. An improvement in the prediction performance by perturbing weak classifiers was also reported in [9].

Our TrBagg algorithm is inspired by this technique. In the process of TrBagg, training examples are sampled from both source and target data. We expected that the variance part of the error can be more drastically reduced, because a source data set contains more diverse examples than a target set. However, there is one difficulty. A source data set contains examples of irrelevant concepts, which we want to ignore, and we cannot discriminate which examples represent the target concept. To avoid this difficulty, we exploit target data that are consistent with the target concept. Specifically, a weak classifier is learned from both source and target data, target data are classified by this weak classifier, and its empirical accuracy for the target data is computed. If the accuracy is sufficiently high, we consider that most of the examples that have been used to train the weak classifier are consistent with the target concept. If the accuracy is not sufficient, the acquired classifier is abandoned. By iterating these procedures, the learner can obtain weak classifiers that are consistent with the target concept and that additionally have wide diversity. In this way, weak classifiers are acquired, and the final class is determined by majority voting, as in standard bagging. We refer to the procedures of learning weak classifiers and the selection of useful classifiers as the learning and filtering phases, respectively. We next describe the details of each phase.

1) Learning Phase: In the learning phase, weak classifiers are learned. Formally, we generate a merged data set, $\mathcal{D} = \mathcal{D}_T \cup \mathcal{D}_S$. As in standard bagging, a training set, $\mathcal{D}_t$, is generated by bootstrap-sampling from the merged set, $\mathcal{D}$. By using an arbitrary classification algorithm, a weak classifier, $\hat{f}_t$, is learned from the training set, $\mathcal{D}_t$. By repeating these procedures for $t = 1, \ldots, T$, T weak classifiers, $\hat{f}_1, \ldots, \hat{f}_T$, are acquired, where T is the initial number of weak classifiers.

This procedure is almost the same as that of standard bagging, but the size of the training set, $|\mathcal{D}_t|$, differs. In the case of bagging, the size of the training set of a weak learner is the same as the size of the given training data set, i.e., $|\mathcal{D}_t| = |\mathcal{D}|$. In our TrBagg, the size of this set, N_t, should be smaller than $|\mathcal{D}|$. N'_T denotes the number of data that are consistent with the target concept in the source set. If N'_T is smaller than N_t, poor weak classifiers would be acquired with high probability, because training sets would contain less data that are consistent with the target concept. While N'_T is assumed to be unknown, it is also assumed to be a few times as large as N_T. Therefore, we set N_t to the size of the target set, N_T. If N_T is too small, good weak classifiers are acquired less frequently. In such cases, the larger training set is preferred, or the sampling sizes themselves can be changed randomly.

2) Filtering Phase: After the completion of the learning phase, a set of T weak classifiers, $\mathcal{F} = \{\hat{f}_1, \ldots, \hat{f}_T\}$, is available. In the filtering phase, from $\mathcal{F}$, a set of weak classifiers, $\mathcal{F}^* \subseteq \mathcal{F}$, is selected so that labels of the target concept are accurately predicted. Once a set of weak classifiers, $\mathcal{F}^*$, is given, the classification procedure of our TrBagg is the same as that of standard bagging. Majority voting in equation (1) is used to predict the classes for a new vector, except that $\mathcal{F}^*$ is adopted instead of $\mathcal{F}$.

We developed the following three filtering approaches.

Majority voting on the target set: This first approach tries to find a subset of $\mathcal{F}$ so as to minimize the empirical errors on the target set derived by majority voting of

INPUT: $\mathcal{F} = \{\hat{f}_1, \ldots, \hat{f}_T\}$ and D_T
1: learn a classifier, $\hat{f}_0$, from the target set, D_T;
$\mathcal{F} \leftarrow \mathcal{F} \cup \hat{f}_0$
2: compute empirical errors on $\mathcal{D}_T$ of each classifier in $\mathcal{F}$, and sort classifiers in ascending of these errors: $\langle f_{i_0}, f_{i_1}, \ldots, f_{i_T} \rangle$
3: $\epsilon \leftarrow$ empirical error of f_{i_0} on $\mathcal{D}_T$; $\mathcal{F}' \leftarrow \{f_{i_0}\}$; $\mathcal{F}* \leftarrow \{f_{i_0}\}$
4: for $t = 1, 2, \ldots T$ do
5: $\quad \mathcal{F}' \leftarrow \mathcal{F}' \cup \{\hat{f}_{i_t}\}$
6: $\quad \epsilon' \leftarrow$ empirical error of a majority vote in $\mathcal{F}'$ on $\mathcal{D}_T$
7: $\quad$ if $\epsilon' \leq \epsilon$ then $\mathcal{F}^* \leftarrow \mathcal{F}'$; $\epsilon \leftarrow \epsilon'$
8: output $\mathcal{F}^*$

Figure 1. Majority voting on the target set

classifiers in the subset of $\mathcal{F}$. Due to the computational cost of checking all subsets of $\mathcal{F}$, we adopted a heuristic procedure in Figure 1. In step 1, a classifier, $\hat{f}_0$, is learned from the target set, and is added to $\mathcal{F}$. We call $\hat{f}_0$ a *fallback classifier*, which is useful for avoiding negative transfer as described later. Note that this fallback classifier is trained so as not to over-fit to target training data. In step 2, weak classifiers are sorted in ascending order of their empirical errors on the target set. By this step, the more promising classifier is preferentially selected. After initialization in step 3, the searching loop from step 4 starts. In this loop, one classifier is iteratively added to the candidate set, $\mathcal{F}'$. Majority voting by classifiers in $\mathcal{F}'$ is performed for the target data, and the empirical error by this voting is calculated. By iterating this loop, this algorithm finds the subset, $\mathcal{F}^*$, with a small empirical error on $\mathcal{D}_T$. The time complexity of this algorithm is $O((T+1)N_T)$, which is the required computational cost of evaluating all target data by all weak classifiers, plus the cost of learning one weak classifier. We abbreviate this approach as MVT.

Majority voting on the validation set: Minimizing the empirical error on the target may cause over-fitting. To avoid this problem, part of the target data is used as a validation set. Empirical errors are evaluated for this validation set using the algorithm in Figure 1. This approach is effective if the target data are fully available and enough validation data can be maintained. However, because the validation data must be separated from the data for training weak classifiers in the learning stage, a small target set damages the prediction in this method more severely than in MVT. We abbreviate this approach as MVV.

Simple comparison: This approach was adopted in our preliminary studies published in workshops [10], [11]. A weak classifier, $\hat{f}_0$, is learned from the target set, $\mathcal{D}_T$. If a weak classifier in $\mathcal{F}$ has an empirical error on the $\mathcal{D}_T$ that is worse than that of $\hat{f}_0$, it is rejected. If all classifiers in $\mathcal{F}$ are rejected, the set that contains only $\hat{f}_0$ is outputted. Contrary to the above approaches, this approach ignores the potential improvement by combining weak classifiers. We abbreviate this approach as SC.

D. Useful Properties of TrBagg

TrBagg has the following desirable characteristics in practical use.

Ease in tuning and implementation: The most prominent merit of TrBagg is its simplicity: It is easy to implement, and the number of learning parameters to tune is very few. The learning parameters are those of weak learners and optionally the size of the sampled training data. Consequently, it is generally easy to tune this algorithm. And, despite this ease, this algorithm works with reasonable accuracy as shown in section IV. Though other transfer learning methods might perform better if their learning parameters are finely tuned, such tuning is can be difficult. Contrary to such methods, our TrBagg is off-the-shelf.

Addressing a negative transfer problem: In the filtering phase, we introduced a fallback classifier, $\hat{f}_0$, that is trained from the target data set. This classifier ensures output $\mathcal{F}^* = \{\hat{f}_0\}$, which contains only a fallback classifier, if no useful knowledge is found in the source data. Because the empirical error of the classifier, $\hat{f}_0$, is guaranteed, negative transfer [7] can be frequently avoided. While there is no theoretical guarantee of its effectiveness, this simple trick worked well in our experiments, shown later.

Multiple domain transfer: The learning phase is much more time-consuming than the filtering phase. We now consider the situation in which the training sets of weak learners consist of data from multiple domains, and must be transferred to each of these domains. In this situation, the generation of weak classifiers in the learning phase can be shared; thus, most of the computational cost can be saved. In the example case of tag personalization described in the introduction, given the tags labeled by all users, we wanted to personalize the results to each user. In the learning stage, training data for weak learners are sampled from tags labeled by all users. Then, these acquired weak learners are delivered for each user. Each user can perform the filtering phase by referring only to his/her own tags, and can successfully transfer other users' knowledge from received weak classifiers. We consider this characteristic to be a strong point of our TrBagg.

Parallel computation: Recently developed CPUs have many cores, and the construction cost for computer clusters is coming down. As a consequence, the adaptability of processes of parallel computation is becoming more attractive. In the time-consuming learning phase of TrBagg, weak classifiers can be learned in parallel. This adaptability to parallel computation is inherited from standard bagging

III. Personalization in Collaborative Tagging

Collaborative tagging services such as delicious.com enable users to register their favorite Web pages. For these registered pages, users can assign tags, which are words that describe the contents, characteristics, or categories of the tagged pages. These tags are useful for searching or

classifying their own registered pages. In addition, these pages and tags can be shared among users of the service. With these shared tags, users can search the pages that other users have registered and find like-minded users.

This social tagging process differs from a traditional classification scheme. In the case of a document repository or library collection, the materials are classified under well maintained and unified criteria. That is to say, the semantics of the labels are strictly defined. By contrast, social tags are based on each user's personal criterion. Users can freely choose their favorite tags; thus, the semantics of social tags can vary greatly. Golder and Huberman addressed such inconsistency among collaborative tags [12]. They discussed the causes of variations in the semantics of the tags. One variation involves the degree of specificity. For example, the official page of the programming language python can be tagged by either the specific word, "python," or the general one, "programming." Another cause is polysemous words having many related meanings. For example, one user may consider "data mining" a statistical technique for marketing research. But another user may use this term to indicate techniques for analyzing massive data sets. Note that these polysemous words are different from homonymous words, which have multiple unrelated meanings. Further, synonymous words or singular/plural forms give rise to another type of inconsistency. As a consequence, the tags of one user may be inappropriate for another user. When searching for documents with shared tags, users might find undesired documents or miss relevant pages.

If we focus on a specific target user, the choice of tags would be highly consistent, because users would presumably follow their own preference pattern, which would be highly self-consistent. We here perform a tag prediction task that is personalized to the target user as follows. First, for each candidate tag, we acquire a binary classifier to discriminate whether the target user will assign the candidate tag to Web pages. To personalize this discrimination, this classifier is trained from Web pages that are tagged by the target user, because such tags are considered to reflect the target user's concepts. Once such classifiers are learned, these can be used for judging whether each candidate tag should be assigned to a new Web page. For example, consider the case in which the target user prefers the tag "python" to "programming." The classifiers for the tag "python" and "programming" would return positive and negative outputs, respectively, and tags that are appropriate for the target user could be estimated. Of course, it is an inefficient approach to learn binary classifiers for all tags. We will tackle this problem by introducing multi-label classifiers in future research.

However, we encountered one serious problem, which is often referred to as the *cold-start problem* in a recommendation context [13]. In order that Web pages are tagged based on the concept of the target user, binary classifiers are learned from a set of Web pages that have been already tagged by the user. The number of such Web pages is generally small, because it is difficult for one user to tag thousands of pages. In this case, the learned classifiers cannot make precise predictions, due to the lack of training examples.

Transfer learning can overcome this difficulty. We can exploit the enormous number of Web pages that have been tagged by non-target users. These Web pages cannot be directly used for training classifiers, because most of them are inconsistent with the target user's concepts. We can apply our TrBagg by treating the target user's and the non-target users' tagging information as target and source data, respectively. We expect that TrBagg will enable us to learn more precise classifiers, because these source data contain some useful information for learning the target user's concept. In particular, some users would surely select the same level of specificity as that of the target user, and some users would definitely use polysemous words in the same sense as the target user. Furthermore, as described in section II-D, TrBagg can transfer knowledge to multiple domains with low computational cost. Once a set of weak classifiers is learned, the personalization to each user is fast. In addition, because only the user's own tagging data are required to perform the filtering phase, users can personalize without disclosing their own bookmarks.

IV. Experiments

We next apply our TrBagg to the task of tag personalization in collaborative tagging. After introducing experimental settings, we compare it with standard bagging and other transfer learning methods, and we show the variation of the performance in different settings.

A. Data Sets and Experimental Settings

We here describe our collaborative tagging data sets. We obtained data from two social bookmark services: *delicious* (http://delicious.com/) and *hatena* (http://b.hatena.ne.jp/). In both sites, many Web pages regarding technical topics are bookmarked. The delicious site is a pioneer of the social bookmarking service. We crawled this site in July, 2007. The number of unique URLs, tags, and users were 762454, 172817, and 6488, respectively. We found 3198185 bookmarks, or pairs of one registered URL and one tag assigned to the URL. The hatena site is a major bookmark site in Japan. We crawled this site in November, 2006. The numbers of unique URLs, tags, and users were 488978, 117264, and 15526, respectively. We found 6165052 bookmarks. Note that the sizes of data sets are generally larger than those of the delicious data, mainly because of the function that suggests tags that have been already assigned.

We counted up the number of URLs to which each tag was assigned, and selected the 20 most-assigned tags[2]. We

[2]The delicious tag "imported" was skipped, because the subsequent data processing eliminated all the positive examples in the source data

Table I
THE SIZES OF COLLABORATIVE TAGGING DATA SETS

(a) delicious data set

tag name	target	source	tag name	target	source
blog	603	24201	web2.0	917	25256
design	1405	25353	politics	5455	21857
reference	6323	19512	news	67	28385
software	3512	30264	howto	6359	23335
music	6311	22914	linux	1151	24288
programming	4498	25931	blogs	3472	18437
web	1291	31024	tutorial	3518	28593
tools	3493	23625	games	3218	22588
video	1870	30334	free	3509	23543
art	6258	16574	webdesign	1098	25427

(b) hatena data set

tag name	target	source	tag name	target	source
web	5776	70531	2ch	8691	65493
neta	8691	62146	music	5504	61914
blog	7220	72804	software	1923	46284
news	4912	44920	*to read*	1780	19201
social	3867	70390	book	2345	53362
hatena	7903	73400	design	1554	37171
neta	2420	62345	google	11132	71846
tool	4634	35245	web2.0	3289	52738
tips	4875	34566	life	5506	54355
game	2657	53533	programming	4823	28676

NOTE: The "tag name" columns show the words used as target tags. Tags written in italics were originally in Japanese. Note that *neta* means "something to talk about" and *2ch* is the title of a discussion board. The "target" and "source" columns show the sizes of the corresponding target and source data sets.

focused on each of these tags, and call it a target tag. For each target tag, we focused on the top user, who was the user who assigned the target tag to the most URLs among all users. This top user and the URLs tagged by this user were treated as the target user and as the target data, respectively. We tried a binary classification to predict whether the target tag would be assigned to a given URL or not. Among all URLs tagged by the target user, the URLs with the target tag were used as positive examples, and the rest of the URLs were used as negative ones. As the source data set, we used the URLs tagged by the second to twentieth top users of the target tag. We refer to these nineteen users as source users.

For each of the twenty target tags, we generated data sets. The sizes of target and source data are summarized in Table I. Investigating the most popular tags would certainly lead to the results that deviate from the standard tags. We therefore generated shrunken data sets in which a part of the target data was discarded. The data sets involving all the target data are denoted by "ALL." If 3/4 of target data were discarded, the data set was denoted by "1/4." For example, in the case of the *blog* tag of the delicious site, the "All" data set consisted of 603 target and 24201 source data. The "1/2" data set still had 24201 source data, but the number of target data were reduced to 302. For each target tag and site, we generated "1/2" to "1/32" data sets.

Table III
PREDICTION ACCURACIES ON THE TARGET DATA OF THE HATENA SITE

Data size	All	1/2	1/4	1/8	1/16	1/32
W/L	2/0	4/1	7/1	7/3	9/2	10/2

NOTE: The win/loss tally is shown as in Table II.

We next turned to the features to represent URLs. As features, we adopted the most popular 100 tags other than the target tag. That is to say, the i-th element of the feature was the number of users who assigned the i-th most popular tag to the URL. We then abandoned the URLs to which none of the top 100 tags was assigned. In the context of recommendation, estimation methods based on other users' tags are referred to as collaborative filtering. In this setting, methods specialized for collaborative filtering would be straightforward. However, we plan to use text features extracted from Web pages in future work. Using text features enables us to recommend tags for Web pages that are not tagged by any users, the existence of which is one of weak points of the collaborative approach. The reason why we use other users' tags as features is to avoid data cleansing. If in the present case we adopted text features that needed elaborate data cleansing, we would not be able to determine whether any potential failure of estimation was caused by the failure of transfer or the poor data cleansing.

We performed a five-fold cross-validation test. The target example set was divided into five blocks. One block was respectively picked, and the examples in these blocks were used for testing. The remaining four target blocks and all the source examples were used as the target and source training data, respectively. Note that original test data sizes were kept even for the "1/2 – 1/32" data sets. For example, if the original target size was 100, the training and test data sizes of the "All" case became 80 and 20, respectively. In the case of "1/2", the training data were shrunk to 40, but the test size was kept at 20.

Default experimental conditions were as follows: As weak learners, we adopted a naive Bayes learner with multinomial model [14], because we preferred a fast learner to deal with data sets that are much larger than popular data sets, e.g., 20Newsgroups or Reuters. The number of weak classifiers was set to 100. The size of the sampled training data for a weak learner, $\mathcal{D}_t$, was the same as that of the target data, N_T. The filtering of TrBagg was performed by the MVT method. In the statistical tests, we used the Z-test to check the difference between ratios at the significance level of 1%.

B. Comparison with a Standard Bagging

We first compared our TrBagg with a baseline method. As described in section II-D, TrBagg was consistently superior to the weak classifier learned from the target data. Therefore, as a baseline method, we chose a standard bagging method in which weak classifiers were trained using the target data.

Table II
PREDICTION ACCURACIES ON THE TARGET DATA OF THE DELICIOUS SITE

tag name	size of target data sets											
	All		1/2		1/4		1/8		1/16		1/32	
	TBt	BGt	TBt	BGt	TBt	BGt	TBt	BGt	TBt	BGt	TBt	BGt
blog	0.743	0.750	0.688	0.698	0.657	0.662	0.678	0.701	0.632	0.643	**0.663**	0.587
design	0.737	0.722	0.728	0.726	0.738	0.724	0.708	0.725	0.738	0.727	0.730	0.735
reference	0.823	0.814	**0.829**	0.806	0.828	0.814	**0.833**	0.805	**0.841**	0.810	**0.838**	0.814
software	0.768	0.766	0.774	0.769	0.781	0.773	**0.795**	0.753	**0.802**	0.774	0.806	0.787
music	0.957	0.953	0.959	0.953	**0.964**	0.955	**0.968**	0.958	**0.964**	0.953	**0.964**	0.946
programming	0.898	0.899	0.897	0.896	0.893	0.889	0.887	0.880	0.879	0.877	**0.869**	0.745
web	0.774	0.783	0.761	0.765	0.754	0.766	0.711	0.734	0.696	0.697	0.697	0.699
tools	0.767	0.745	**0.759**	0.716	**0.745**	0.697	**0.755**	0.672	**0.723**	0.668	**0.725**	0.650
video	0.882	0.889	0.896	0.891	0.891	0.894	0.891	0.899	0.906	0.899	0.879	0.871
art	**0.914**	0.899	**0.917**	0.896	**0.922**	0.900	**0.927**	0.909	**0.927**	0.911	**0.927**	0.915
web2.0	0.706	0.707	0.696	0.702	0.696	0.710	0.710	0.713	0.710	0.704	0.700	0.718
politics	0.665	0.665	0.662	0.663	0.647	0.642	0.642	0.636	0.620	0.631	0.629	0.648
news	0.896	0.806	0.866	0.731	**0.746**	0.507	0.672	0.478	0.507	0.403	0.522	0.403
howto	0.906	0.900	0.905	0.899	**0.911**	0.894	**0.918**	0.891	**0.924**	0.872	**0.917**	0.850
linux	**0.787**	0.738	0.789	0.764	0.797	0.771	0.803	0.794	0.759	0.746	0.785	0.745
blogs	0.937	0.923	**0.939**	0.919	0.939	0.930	0.938	0.930	**0.939**	0.925	**0.935**	0.919
tutorial	**0.906**	0.887	**0.911**	0.885	**0.914**	0.880	**0.919**	0.875	**0.915**	0.886	**0.916**	0.858
games	0.964	0.964	0.964	0.962	0.962	0.961	0.959	0.954	**0.965**	0.943	**0.958**	0.874
free	0.821	0.803	**0.827**	0.781	**0.841**	0.802	**0.839**	0.764	**0.839**	0.750	**0.836**	0.736
webdesign	0.732	0.734	0.721	0.727	0.701	0.729	0.704	0.727	0.722	0.705	0.708	0.692
W/L	3 / 0		6 / 0		7 / 0		8 / 0		10 / 0		11 / 0	

NOTE: The column "tag name" shows the strings of the target tags. The column pairs "All", "1/2 – 1/32", indicate the sizes of the target data sets. The left TBt and right BGt columns of each pair show the results derived by TrBagg and baseline bagging, respectively. Each row shows the accuracies for the corresponding target tag. Bold face indicates the "winner," i.e., the accuracy was statistically greater than its opponent. The last row "W/L" shows the number of target tags for which our method won/lost against baseline bagging.

The accuracies on the target data sets are shown in Tables II and III. Our TrBagg was equal to or superior to standard bagging for all experiments on the delicious data. Regarding the hatena data, though TrBagg lost against bagging in several cases, TrBagg won much more frequently. These results clearly show the effectiveness of TrBagg. Further, the effectiveness of TrBagg became more significant as the size of the target sets decreased. Specifically, the decrease of the accuracy with decreased size of target sets was less using TrBagg than that using standard bagging. This fact enhances the usefulness of our method, because transfer learning is more useful when fewer target data are available. Note that we also examined bagging in which weak classifiers were trained from both the target and source data, but the prediction accuracies on the target data were worse than in the above bagging.

In terms of this comparison, we also performed experiments on delicious data using SVMs as weak learners. Note that the hatena set was too large to apply SVM. We used the kernlab package on R, and linear kernels were adopted due to the fatal slowness of the other types of kernels. In this experiment, no statistical differences were observed between our TrBagg and standard bagging. However, even in this case, the inference by the classifiers learned by TrBagg was faster, because irrelevant weak classifiers were filtered out.

In summary, TrBagg successfully acquired more accurate classifiers than the baseline. It can be concluded that TrBagg succeeded in learning better classifiers by exploiting useful information involved in the source data.

C. Comparison with Other Transfer Learning Methods

Our TrBagg was compared with the following transfer learning methods. First, naive Bayes was chosen as a baseline. The classifier trained with the target set is denoted by NBt, and that trained with both the target and source data is denoted by NBm. The bagging classifier used in section IV-B is denoted by BGt. Further, that trained by both the target and source data is denoted by BGm. TrAdaBoost [6] is a transfer-learning version of the AdaBoost. In our implementation, naive Bayes was used as a weak classifier. If the empirical error on the target set was larger than 1/2, positive and negative classes of weak classifiers were exchanged. Further, if the empirical error was very small, it was made a small constant to avoid zero-division. The number of weak classifiers was changed to 10, 30, 50, and 100, and the best classifier was selected. The use of independent validation sets for the evaluation is denoted by TAv. As a validation set, one block in a training set was used, and the training data set was reduced to three blocks. The use of test sets for evaluation is denoted by TAt. The performance of TAt was over-estimated, because this accesses to test data. While the accuracies of TAv show the level of performance when learning parameters are selected

Table IV
COMPARISON WITH OTHER METHODS

(a) delicious

		NBt	NBm	BGt	BGm	TAv	TAt	MX	FE	TBt
All	NBt	—	5/2	0/0	4/4	8/1	7/2	7/0	8/3	0/6
	BGt	0/0	7/0	—	5/2	8/1	7/1	10/0	8/1	0/3
	TBt	6/0	9/0	3/0	7/0	9/1	8/1	12/0	10/0	—
		NBt	NBm	BGt	BGm	TAv	TAt	MX	FE	TBt
1/16	NBt	—	4/6	0/2	4/10	3/4	3/7	10/4	1/4	0/10
	BGt	2/0	6/2	—	4/10	4/2	3/6	12/2	2/3	0/10
	TBt	10/0	11/2	10/0	4/1	11/0	9/3	15/1	9/0	—

(b) hatena

		NBt	NBm	BGt	BGm	TAv	TAt	MX	FE	TBt
All	NBt	—	16/1	0/0	15/1	8/7	3/10	11/3	14/4	0/2
	BGt	0/0	16/1	—	15/1	8/7	5/9	13/3	14/3	0/2
	TBt	2/0	17/1	2/0	17/1	8/6	3/8	12/3	14/4	—
1/16	NBt	—	15/1	0/8	6/9	7/9	3/8	12/2	14/4	1/11
	BGt	8/0	17/1	—	8/8	8/7	6/6	14/1	16/3	2/9
	TBt	11/1	17/0	9/2	11/1	13/3	9/6	18/1	17/3	—

NOTE: Each entry shows the counts of target tags the row method won (left) or lost (right) against the column method. The labels "All" and "1/6" indicate the size of the data sets.

using cross-validation, those of TAt show the potential performance. We assumed that the target data were generated from a single multinomial model and the source data were generated from a mixture model of the target model and another multinomial model. MX denotes a mixture model, which implemented this assumption directly. While target data are generated purely from the target distribution, source data are generated from a mixture distribution of target and non-target distributions. This can be considered as a generative version of an approach in [5] FE denotes the *frustratingly easy* method in [15], which is as easy to implement as our TrBagg. Finally, our TrBagg is denoted by TBt. Note that for a few cases in which the target set was too small to maintain the validation set, the results are counted as a tie.

Overall, our TrBagg outperformed the other transfer learning methods as shown in Table IV. Regarding the baseline methods, i.e. single naive Bayes and bagging, TrBagg almost always won. If knowledge was well transferred, TrAdaBoost performed well, but the differences between it and TrBagg were not significant in almost all cases. Inversely, because TrAdaBoost doesn't have a method of avoiding negative transfer, the prediction accuracies were degraded if the source data contain no useful knowledge. Additionally, TrAdaBoost was performed relatively well on the hatena set, whose size is larger. TrAdaBoost determined the sampling distribution based on accuracies on the target data. To evaluate the accuracies precisely, the larger-size target data sets are preferred. Therefore, our TrBagg was advantageous for the relatively small target data set, as in this experiment. Although the mixture model directly represented

Table V
COMPARISON WITH OTHER METHODS

		NBt	BGt	TAv	TAt	TBt
rec-talk	All	0.990	0.990	0.975	0.982	0.990
	1/16	0.972	0.974	0.916	0.940	0.972
rec-sci	All	0.983	0.984	0.954	0.974	0.983
	1/16	0.933	0.931	0.901	0.927	0.933
sci-talk	All	0.977	0.975	0.954	0.953	0.977
	1/16	0.932	0.935	0.898	0.906	0.932

Table VI
CHANGING THE FILTERING APPROACHES

	delicious		hatena	
	All	1/16	All	1/16
vs MVV	1/0	3/1	2/3	7/4
vs SC	0/0	3/1	0/0	7/1

NOTE: Each cell shows the win/loss tally of the MVT approach against MVV or SC.

TrBagg's assumption, it performed poorly. We think that this is mainly because the distribution of the source data was rather complicated, and simple multinomial models failed to fit. We suppose that the reason why the frustratingly easy method performed poorly was the mismatch of the assumption. This method assumes that useful features are different for each domain, but in this personalization task, useful samples are different for each domain.

For comparison, we applied the above methods to the 20 Newsgroups corpus, which has been widely tested such as in [6]. Texts were processed as in [16], except that the most frequent 5000 words were adopted as features because of the limitation of our implementation. In addition to the original setting, we tested the case in which the target data were reduced to 1/16. According to the accuracies in Table V, no clear improvement was observed. This would be because the target data sets were larger in comparison with their corresponding source sets. Again, the superiority of TAt to TAv indicated the possibility of improvements by tuning learning parameters. On the other hand, our TrBagg worked reasonably well because of the simplicity in tuning and the functionality of avoiding negative transfer.

In summary, our TrBagg was performed well compared with other transfer learning methods especially when the target data were relatively few in number.

D. Behaviors of TrBagg

Finally, we investigate the behaviors of TrBagg.

First, we changed the filtering approaches in section II-C2. The MVT method was compared with the other two methods. Though the MVV has the disadvantage that the number of training data must be reduced to prepare independent validation data, it also has the advantage that it can reduce the expected generalization error directly. We therefore anticipated that MVV would perform well if a large target data

Table VII
CHANGING THE NUMBER OF WEAK LEARNERS

	100	300	500	1000
delicious-All	3/0	5/0	5/0	5/0
delicious-1/16	10/0	9/0	10/0	10/0
hatena-All	2/0	3/0	3/1	3/1
hatena-1/16	9/2	9/1	9/2	10/3

NOTE: The win/loss tally compared with standard bagging using the same number of weak learners.

set were given. The experimental results in Table VI show that MVV is of comparative performance if target data are abundant, but its performance is worsened by the reduction of target data. Our old SC performed poorly for the small target sets, too. This is because weak classifiers better than $\hat{f}_0$ are less frequently found, and the transfer of knowledge failed.

Second, we changed the number of weak learners used in TrBagg. We increased the size from 100 to 1000 as shown in Table VII. Slight improvements were observed between 100 and 300 for two "All" data sets, but the accuracies were almost saturated, and no remarkable improvement in performance was observed. The most important thing in this experiment was that no over-fitting was observed.

Third, we changed the sizes of the training sets for weak learners. So far, the size is fixed to the size of the target data, N_T. We experimented with sampling sizes of 5% and 10% of N_T+N_S. Overall, the larger size of sampling data seemed to be preferable if N_T was too small (data not shown).

V. RELATED WORK

The idea of using non-target data in bagging can be found in [9], in which irrelevant data are exploited to reduce the correlations among the errors of weak classifiers. Several ensemble approaches for transfer learning have been proposed. Given multiple weak classifiers that are trained in different domains, these are combined with weights based on relatedness to the target domains [17], [18]. An AdaBoost algorithm was modified so as to be better fit for transfer learning by less weighting of useless source data [6]. While boosting adjusts the weights of data or classifiers adaptively, bagging reduces variance by exploiting randomness in sampling. This is the intrinsic difference between the two approaches [8]. The above methods basically follow a boosting-like strategy; thus, they are highly effective if target data are abundant and sufficient information is available. On the other hand, because our method is a non-adaptive approach and depends on randomness, ours works when fewer target data and less information is available.

Many transfer learning methods have been proposed [2]. As described in the introduction, it is difficult to provide a formal definition of transfer learning. This is mainly due to the lack of an adequate definition of relatedness between tasks, which remains the most important open problem in the study of transfer learning [3]. Different kinds of relatedness are assumed in different techniques. For example, the sets of features that are useful for completing tasks are slightly different [5], and the data distributions of the features are different between the training and test data sets [19]. We think that it would be almost impossible to give a unified formal definition of such relatedness. Different transfer learning algorithms are designed based on their own assumptions about the relatedness. We call these algorithm-specific assumptions by *transfer assumptions*. If the involved transfer assumption is reasonable, useful knowledge can be transferred from the related source data. However, an unreasonable assumption would cause negative transfer. The most general form of transfer assumption is defined by the difference between distributions [17]. However, if more knowledge is available, a more specific assumption makes it possible to transfer more knowledge. For example, positive and negative target data are generated from respective Gaussians. The source data are generated from Gaussians of which centers are the same as those of the target distributions and of which variances are smaller than those of the target distributions. If these conditions are known, the source data can be exploited more effectively. We therefore conclude that more new transfer learning methods should be developed for specific transfer assumptions.

Given transfer assumptions must be manipulated using mathematical models, which we here call *transfer models*. Though one classification scheme of these models was proposed in [2], we offer a modified classification scheme, expressed in two axes. One axis designates whether the transferred data knowledge is processed on the source side or target side. While source data are transformed so as to fit the goal of the target domain, target models are designed so as to accept untransformed source data. The other axis represents the types of knowledge to transform. Feature-based approaches transform the feature space so as to be useful in the target domain. While feature spaces are transformed on the source side [15], target side models in [5] are designed to weigh features automatically. In the instance-based approach, each instance in data sets is weighted or selected according to its relatedness to the target domain. While the above ensemble techniques and covariate shifts [19], [20] weigh the instances on the source side, migratory-logit [21] adopts the target models to decide the importance of instances automatically.

We finally discuss tagging on Web pages or blog posts. P-TAG [22] is a system that predicts appropriate tags for given Web pages. Because this prediction is based on the user's personal repository, and other users' tags are not referred to, there is no need to consider any inconsistencies among tagging criteria. Autotag [23], TagAssist [24], and a method in [25] are designed to predict proper tags for blog articles based on similar blog posts. Though other users' tags are used, inconsistency in tags is not taken into account.

VI. Conclusion

We proposed a new transfer learning algorithm, TrBagg. This algorithm is very simple: after generating many weak classifiers from target and source data, these classifiers are filtered using the target data. This method has several advantages in practical use. TrBagg worked reasonably well in spite of its ease of tuning, and it could alleviate the influence of negative transfer. We applied our TrBagg to collaborative tagging data sets and showed its effectiveness.

In order to make our tag personalization method more useful, we would like to introduce models that can deal with multiple tags. To achieve this, we want to improve our TrBagg so as to handle multi-label classification problems more elaborately. We also need to further enhance the theoretical background of our method.

Acknowledgment

We wish to thank Dr. Yutaka Matuo and Dr. Atsushi Fujii for their valuable advices. Thanks are due to Hottolink Inc. for assistance in crawling.

References

[1] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," Dept. of Computer Science and Engineering, Hong Kong Univ. of Science and Technology, Tech. Rep. HKUST-CS08-08, 2008. [Online]. Available: http://www.cse.ust.hk/~sinnopan/publications/TLsurvey_0822.pdf

[3] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[4] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, pp. 1817–1853, 2005.

[5] H. Daumé, III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

[6] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. of The 24th Int'l Conf. on Machine Learning*, 2007, pp. 193–200.

[7] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS-2005 Workshop on Inductive Transfer: 10 Years Later*, 2005.

[8] L. Breiman, "Arcing classifiers," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[9] P. W. Munro and B. Parmanto, "Competition among networks improves committee performance," in *Advances in Neural Information Processing Systems 9*, 1997, pp. 592–598.

[10] T. Kamishima, M. Hamasaki, and S. Akaho, "Baggtaming — learning from wild and tame data," in *ECML/PKDD2008 Workshop: Wikis, Blogs, Bookmarking Tools – Mining the Web 2.0*, 2008.

[11] ——, "Personalized tag predition boosted by baggtaming — a case study of the hatena bookmark," in *The 3rd Int'l Workshop on Data-Mining and Statistical Science*, 2008.

[12] S. A. Golder and B. A. Huberman, "Usage patterns of collaborative tagging systems," *J. of Information Science*, vol. 32, no. 2, pp. 198–208, 2006.

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[14] A. McCallum and K. Nigam, "A comparison of event model for naive bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, 1998, pp. 41–48.

[15] H. Daumé, III, "Frustratingly easy domain adaptation," in *Proc. of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 256–263.

[16] G.-R. Xue, W. Dai, Q. Yang, and Y. Yu, "Topic-bridged PLSA for cross-domain text classification," in *Proc. of The 31th Annual ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2008, pp. 627–634.

[17] K. Crammer, M. Kearns, and J. Wortman, "Learning from multiple sources," *Journal of Machine Learning Research*, vol. 9, pp. 1757–1774, 2008.

[18] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," in *Proc. of The 14th Int'l Conf. on Knowledge Discovery and Data Mining*, 2008, pp. 283–291.

[19] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. of Statistical Planning and Inference*, vol. 90, pp. 227–244, 2000.

[20] M. Sugiyama and M. K. adn K. R. MÜller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, pp. 985–1005, 2007.

[21] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data streams," in *Proc. of The 22nd Int'l Conf. on Machine Learning*, 2005, pp. 505–512.

[22] P.-A. Chirita, S. Costache, S. Handschuh, and W. Nejdl, "P-TAG: Large scale automatic generation of personalized annotation TAGs for the Web," in *Proc. of The 16th Int'l Conf. on World Wide Web*, 2007, pp. 845–854.

[23] G. Mishne, "AutoTag: A collaborative approach to automated tag assignment for Weblog posts," in *Proc. of The 15th Int'l Conf. on World Wide Web*, 2006, pp. 953–954.

[24] S. C. Sood, S. H. Owsley, K. J. Hammond, and L. Birnbaum, "TagAssist: Automatic tag suggestion for blog posts," in *Int'l Conf. on Weblogs and Social Media*, 2007.

[25] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. Lee Giles, "Real-time automatic tag recommendation," in *Proc. of The 31th Annual ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2008, pp. 515–522.