BaggTaming — Learning from Wild and Tame Data

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. We address a new machine learning problem, *taming*, that involves two types of training sets: wild data and tame data. These two types of data sets are mutually complementary. A wild data set is less consistent, but is much larger in size than a tame set. Conversely, a tame set has consistency but not as many data. The goal of our taming task is to learn more accurate classifiers by exploiting the strong points of each training set. For this task, we develop a new algorithm, BaggTaming, which is a modified version of bagging. Such an algorithm would be useful for predicting tags in collaborative tagging services, such as del.icio.us. With such services, users can register Web documents, assign tags to them, and share these tags. Such tags are similar to the notion of class labels that have been adopted in supervised learning. The tags are poorer in consistency than well-managed labels, but more tags are available. If we consider tags as wild data, and labels as tame ones, Web pages could be more accurately classified with our taming technique.

1 Introduction

In this paper, we address a new learning problem, which we call *taming*, and develop a method for solving this problem. The learner for this taming requests two types of training data sets, *tame* and *wild*. The label information of tame data is carefully maintained based on a consistent target concept, which we actually want to learn. In contrast, wild data are not so well maintained; thus, some labels are consistent with the target concept, while some others are not. Additionally, we assume that wild data are much more abundant than tame data. This assumption is reasonable, because it is generally difficult to provide a large tame data set due to its high maintenance cost. The goal of the taming is to acquire more accurate classifiers by exploiting wild data than those learned only from tame data.

One example of such wild data is found in collaborative tagging systems. Collaborative tagging services such as del.icio.us[1] enable users to register their favorite Web pages. To 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 registered pages and assigned tags can be shared among users of the service. With these shared

[1] http://del.icio.us/

tags, users can search the pages that other users have registered, and they can 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 labeling scheme is highly restricted, and the semantics of the labels is strictly defined. In contrast, social tags are labeled 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 have pointed out such inconsistency among collaborative tags [1]. They discussed the causes of the variation 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 and phrases, which have many related senses. For example, one may consider the term "data mining" as 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 and phrases are different from homonymous words that have multiple unrelated meanings. As a consequence, the tags labeled by one user may be inappropriate to another user. When searching for documents with shared tags, users might find undesired documents or miss relevant pages.

Our taming technique can be applied to cope with this difficulty. We prepare a set of Web pages that are consistently labeled based on the target concept in which we are interested. These labeled pages are treated as tame data. Because making tame data is labor intensive, the size of the tame data set tends to be small. For the task whose labeled training samples are fully available, statistical techniques often fail to find the regularity in the tame data. In contrast, it is relatively easy to collect many tagged pages by using collaborative tagging services, but these tags may not consistent with the target concept as described above; thus, we treat these pages as wild data. These tame and wild data are mutually complementary; the tame data are well maintained, while the size of the wild data set is large. Therefore, it would be fruitful to use both data sets together. Our taming technique lets the user know whether any collaborative tags are consistent with his/her own concept. Further, it is also useful for finding more appropriate tags for new Web pages. In this paper, we will show a method for this taming, which we call BaggTaming. This is a modified version of bagging [2] so as to be applicable to taming. We employ BaggTaming in a tag-prediction task, and show the effectiveness of our method. We expect that this taming technique would be generally helpful for any situation where many unreliable observations are available together with a small amount of highly reliable information.

In section 2, we state a taming learning problem and describe our BaggTaming method. Section 3 show the experimental results for synthetic and real tagging data, respectively. Finally, after discussing the related work in section 4, we conclude in section 5.

2 Taming Task and Its Solution

In this section, we address the learning problem of taming and describe our BaggTaming algorithm, which is developed by modifying bagging so as to be applicable for taming.

2.1 What's Taming

We first address the learning problem of taming. Here, we are focused on classification, though taming techniques can be applied to other types of supervised learning tasks, such as regression. An object is represented by a feature vector, $\mathbf{x}$. The variable c denotes the class to 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 a classifier 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 independently sampled from an identical distribution, $\mathrm{P}[c, \mathbf{x}]$, which expresses the target concept to be learned. On the other hand, in our taming case, two types of training examples, tame and wild, are adopted. Similar to a standard classification case, tame data are assumed to be independently sampled from an identical distribution, $\mathrm{P}[c, \mathbf{x}]$, that corresponds to the target concept. This set of tame data is denoted by $\mathcal{D}_T = \{(c_i, \mathbf{x}_i)\}_{i=1}^{N_T}$, where $N_T = |\mathcal{D}_T|$. A wild data set might include examples that are sampled from distributions that express irrelevant concepts, together with examples that are consistent with the target concept. We further assume that, in the wild data, the number of examples of the target concept is at least a few times N_T and that the learner does not know which examples are generated from the target distribution. This wild data set is denoted by $\mathcal{D}_W = \{(c_i, \mathbf{x}_i)\}_{i=1}^{N_W}$, where $N_W = |\mathcal{D}_W|$. Finally, we assume the size of the wild data set is much larger than that of the tame set, i.e., $N_W \gg N_T$. The goal of the learning problem of taming is to acquire a classifier that can more accurately predict classes by exploiting the information in wild data.

2.2 BaggTaming

To solve the above problem of taming, we developed the algorithm BaggTaming (Bootstrap AGGregated TAMING). Before showing our BaggTaming, we briefly describe the original bagging method, because this algorithm is a modified version of this bagging [2]. 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 $\hat{f}_t(\mathbf{x})$ by using a classification algorithm, such as naive Bayes.

By this procedure, T weak classifiers, $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_T(\mathbf{x})$, can be acquired. These weak classifiers are then aggregated, and any input feature vectors are finally classified. In the case of classification, this aggregation is done by majority voting. Formally, the class to which the feature vector $\mathbf{x}$ should belong is determined by

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

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

The reason why this bagging improves the prediction accuracy is explained simply based on the bias-variance theory [2, 3]. 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 more affects 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 by 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.

We developed our BaggTaming algorithm by modifying this bagging. In [3, section 7], an improvement technique for bagging is reported. In the case of 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 a 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 this modification. Our BaggTaming algorithm is inspired by this improvement technique. In the process of BaggTaming, training examples are sampled from wild data instead from tame data. We expected that the variance part of the error can be more drastically reduced, because a wild data set contains more diverse examples than a tame set. Note that the idea of reducing variance by using the relevant data sets was also proposed in [4]. However, there is one difficulty. A wild data set contains examples of irrelevant concepts, which we want to ignore, and there is no information whether each example is of the target concept or not. To avoid this difficulty, we exploit a tame data, which is consistent with the target concept. Specifically, learning of our BaggTaming iterates the following two steps:

- Training examples are randomly sampled from the wild data set, and a weak classifier is learned from these training examples.
- The empirical accuracy of the weak classifier on the tame data is computed. If the accuracy is sufficiently high, the classifier is accepted; otherwise, it is rejected, and a new classifier is repeatedly learned.

Note that we assume that if most of examples that have been used for training the weak classifier are consistent with the target concept, the empirical accuracy of the classifier would be high. By iterating these procedures, the learner can obtain weak classifiers that are consistent with the target concept and that have wide diversity. In this way, T weak classifiers are acquired, and the final class is determined by majority voting.

```
1:  $t = 1$
2:  while $t \leq T$ do
3:      $s = 1$
4:      repeat
5:          generate a training set $\mathcal{D}_t$ by sampling with replacement from $\mathcal{D}_W$
6:          learn a weak classifier $\hat{f}_t(\mathbf{x})$ from training set $\mathcal{D}_t$
7:          calculate the accuracy $p_t$ of the classifier $\hat{f}_t(\mathbf{x})$ on the tame set n$\mathcal{D}_T$
8:          if $p \geq \mathrm{AcceptLimit}$ then $t = t + 1$, goto step 2
9:          if $s \geq \mathrm{FailureLimit}$ then
                let the most accurate classifier in this inner loop be $\hat{f}_t(\mathbf{x})$
                $t = t + 1$, goto step 2
10:         $s = s + 1$, goto step 4
11: output the weak classifiers $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_T(\mathbf{x})$,
    together with their corresponding accuracies $p_1, \ldots, p_T$.
```

Fig. 1. BaggTaming algorithm

Our BaggTaming algorithm is shown in Figure 1. Two types of training sets, wild and tame data, and a weak learner are given as inputs for this algorithm. T weak classifiers are acquired in the outer loop beginning from step 2. The counter s for the rejection of weak classifiers is initialized at step 3, and the inner loop starts at step 4. In steps 4 to 6, training examples are sampled from wild data, $\mathcal{D}_W$, a weak classifier, $\hat{f}_t(\mathbf{x})$, is acquired, and its accuracy, p_t, on the tame set $\mathcal{D}_T$ is calculated. At step 7, if the accuracy is greater than or equal to the threshold, AcceptLimit, the weak classifier is accepted, and then the algorithm turns to the next iteration of the outer loop. In the next step, if the number of rejections exceeds the threshold, FailureLimit, the most accurate weak classifier is set to $\hat{f}_t(\mathbf{x})$ as a backup, and then the next iteration of the outer loop begins. At step 9, after incrementing the rejection counter s, the acquisition of a weak classifier is re-tried. Finally, the algorithm outputs the weak classifiers $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_T(\mathbf{x})$, together with their corresponding accuracies, $p_1, \ldots, p_T$.

We describe an option for our BaggTaming algorithm. As the threshold, AcceptLimit, we adopted three options: three classifiers are leaned from all wild data, all tame data, and merged data, respectively. The accuracies of these classifiers on the tame set are calculated, and these accuracies are used as thresholds. Generally, the frequency of accepted classifiers rises in the order of tame, merged, and wild options.

We describe a few comments and options for our BaggTaming algorithm. If the ratio of examples consistent with the target concept to the total number of wild data is low, weak classifiers are too frequently rejected, and the algorithm slows. In such a condition, a weak classifier is learned from a union set of $\mathcal{D}_t$ and $\mathcal{D}_T$ at step 6. By adopting this option, weak classifiers are accepted more frequently, but the effect of reducing the variance in the prediction error is lessened.

The classification procedure of our BaggTaming is similar to that of standard bagging. Majority voting is used to predict the classes for a new vector, but each vote is weighted. As described before, the accuracy, p_t, would be high if the $\mathcal{D}_t$ contains many examples that are consistent with the target concept. Therefore, we weigh each weak classifier by p_t. Specifically, the class to which the feature vector $\mathbf{x}$ should belong is determined by

$$\hat{c} = \arg\max_{c \in \mathcal{C}} \sum_{t=1}^{T} p_t \mathrm{I}[c = \hat{f}_t(\mathbf{x})]. \tag{2}$$

2.3 Application of our BaggTaming Technique to a Tag Prediction Task

We then apply the above BaggTaming technique to the task to predict personalized tags in a social bookmark system. As described in the introduction, different users may assign different tags to the same Web page. For example, one may choose “python”, but another may do “programming”, according to his/her preference in the specificity level. Further, the synonymous words or the singular/plural forms give rise to another type of inconsistency. However, if we give attention to the specific target user, the choice of tags would be highly consistent, because any users would behave according to 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 be personalized this discrimination, this classifier is trained from the Web pages that are tagged by the target user, because such tags are considered to be reflecting the target user’s concept. 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 that the target user prefers the tag “python” to “programming” in terms of a given Web page. The classifiers for the tag “python” and “programming” would return positive and negative outputs, respectively, and tags that are appropriate for the target user can be selected.

However, we encounter a serious problem, which is often referred as a *cold-start problem* in a recommendation context [5]. In order that Web pages are tagged based on the concept of the target user, the binary classifiers are learned from the set of Web pages that have been tagged by the user before. The number of such Web pages are generally small, because it is difficult for one user to tag thousands of pages. In this case, the learned classifiers cannot make precise prediction, due to the insufficiency of training examples.

To cope with this difficulty, we adopt our BaggTaming technique. The Web pages that tagged by the target user are highly consistent with the user’s concept, while the number of pages are generally small. Additionally, we can exploit the enormous Web pages that have been tagged by the non-target users. These Web pages cannot be directly used for training classifiers, because the most of these are inconsistent with the target user’s concept. However, We can apply our BaggTaming by treating the target user’s and the non-target users' tagging information as tame and wild data, respectively. We expect that our BaggTaming enables to learn the more precise classifier, because these wild data partially contain 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 synonymous words in the same sense as the target user. In the next experimental section, we will show the more precise personalized classifiers can be learned by employing our BaggTaming in conjunction with both the target user's and non-target users' tagging data.

3 Experiments for Collaborative Tagging Data

In this section, we applied our BaggTaming to collaborative tagging data.

3.1 Data Sets and Experimental Settings

We first describe our collaborative tagging (CT) data set. We crawled the collaborative tagging site, del.icio.us, in July, 2007. The number of unique URLs, tags, and users were $762,454$, $172,817$, and $6,488$, respectively. We found $3,198,185$ bookmarks, which were the pairs of one registered URL and one tag assigned to the URL.

We counted the number of URLs to which each tag was assigned, and selected the 20 most-assigned tags. We focused on one of these tags, and call it a target tag. In this experiment, we dealt with a binary classification to predict whether the target tag would be assigned to a given URL or not, as described in section 2.3. 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. We treated the URLs tagged by the top user as the tame data. We hereafter refer to this top user as a tame user. Among all URLs tagged by the tame user, the URLs to which the target tag was assigned were used as positive examples, and the rest of the URLs were used as negative ones. As the wild 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 wild users. The most of tags assigned by these wild users would be inconsistent, but a part of them might share the tame user's concept. Thus, the URLs tagged by these wild users were treated as wild data.

We next turned to the features that describe the URLs. As features, we adopted the most popular 100 tags except for 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. Note that the Web pages' texts is frequently used as features, but we didn't employ them to avoid steps for cleaning irrelevant information, such as advertise messages. For each of the twenty target tags, we generated data sets. The sizes of tame and wild data are summarized in Table 1.

3.2 Results for Collaborative Tagging Data

We first compared our BaggTaming with a baseline method, which is standard bagging whose weak classifiers were trained by using the tame data. For both methods, we

Table 1. The sizes of collaborative tagging data sets

target tag	tame	wild	target tag	tame	wild
blog	2908	28214	web2.0	1784	13829
design	2511	26791	politics	1234	13709
reference	2355	22847	news	2473	13429
software	2658	22529	howto	1685	13407
music	2898	19725	imported	405	12862
programming	1697	18668	linux	1535	12231
web	2296	18503	blogs	1465	12217
tools	2365	18488	tutorial	1883	12001
video	2538	16734	games	2097	11291
art	2054	16521	free	1960	11258

NOTE: The "target tag" columns show the words used as target tags. The "tame" and "wild" columns show the sizes of the corresponding tame and wild data sets.

adopted again a naive Bayes learner with multinominal model [6] as weak classifiers, because this learner is computationally fast. The number of weak classifiers, T, was 30. The size of the sampled data was one-half of the tame data, i.e., $|\mathcal{D}_t| = N_T/2$. The FailureLimit threshold was set to 100, and the AcceptLimit threshold was the accuracy of the classifier learned from the merged data. We performed a 5-fold cross-validation test. Tame 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 tame blocks and all the wild examples were used as the tame and wild training data, respectively.

The accuracies on the tame data set are shown in Table 2. We changed the tame data set without changing the wild data. For all ALL – 1/16 cases, our BaggTaming was clearly superior to bagging for the tame data. This result leads to the conclusion that our BaggTaming method can select the target information in the wild data and can use the information for learning classifiers. Further, the effectiveness of BaggTaming became more significant as the size of the tame sets decreased. This fact enhances the usefulness of BaggTaming, because a taming technique is more useful under a cold-start condition where less tame data are available. We also tested bagging classifiers whose weak classifiers were trained by using the wild set or merged data sets. We observed that almost all of these classifiers were inferior than bagging with weak classifiers trained from the tame set. Especially, for the tags, "web" or "web2.0", the classifiers learned from the wild data are fairly worse. This fact indicates that these wild data sets don't contain useful information for solving the target classification problem; thus, for these tags, we considered that our BaggTaming is inferior to the baseline as observed in Table 2.

We finally examined the effects of the parameters and options of BaggTaming. We applied BaggTaming under different settings to the tagging data with the ALL type tame set, and three statistics were calculated for each setting as shown in Table 3. As described in section 2.2, we prepared three options for the AcceptLimit, which is the threshold used for checking whether a candidate weak classifier is fully accurate or not. The columns "tame", "merged", and "wild" show results derived by BaggTaming with the tame, merged, and wild options. Adopting the tame option produces a set of highly

Table 2. Prediction Accuracies on The Tame Data Set

tag name	size of tame data sets									
	ALL		1/2		1/4		1/8		1/16	
	BT	bagg	BT	bagg	BT	bagg	BT	bagg	BT	bagg
blog	0.700	0.755	0.697	0.701	0.665	0.658	0.663	0.697	**0.713**	0.637
design	0.744	0.723	0.738	0.725	0.737	0.728	0.738	0.718	0.746	0.729
reference	0.825	0.813	**0.833**	0.808	**0.836**	0.812	**0.835**	0.816	**0.849**	0.815
software	0.752	0.771	0.764	0.767	0.784	0.775	**0.800**	0.753	**0.801**	0.775
music	0.956	0.953	**0.964**	0.954	**0.966**	0.956	**0.967**	0.958	**0.966**	0.954
programming	0.896	0.900	0.895	0.895	0.893	0.889	0.888	0.878	0.881	0.878
web	0.661	**0.775**	0.657	**0.763**	0.660	**0.760**	0.651	**0.731**	0.645	**0.702**
tools	0.772	0.749	**0.769**	0.728	**0.768**	0.700	**0.760**	0.667	**0.741**	0.681
video	0.884	0.887	0.892	0.891	0.891	0.897	0.896	0.901	0.883	0.901
art	**0.920**	0.899	**0.924**	0.895	**0.930**	0.898	**0.932**	0.908	**0.928**	0.915
web2.0	0.614	**0.708**	0.614	**0.696**	0.630	**0.706**	0.627	**0.716**	0.709	0.706
politics	0.649	0.666	0.633	**0.658**	0.625	0.645	0.628	0.638	0.622	0.631
news	**0.940**	0.806	**0.925**	0.687	**0.791**	0.522	0.657	0.478	**0.910**	0.403
howto	0.901	0.902	0.908	0.901	**0.917**	0.893	**0.920**	0.893	**0.926**	0.877
imported	1.000	0.971	1.000	0.971	0.994	0.965	**1.000**	0.907	0.936	0.930
linux	**0.783**	0.740	0.794	0.769	0.793	0.775	0.798	0.796	0.778	0.749
blogs	**0.937**	0.923	**0.941**	0.923	0.940	0.928	0.938	0.931	0.938	0.925
tutorial	**0.908**	0.889	**0.914**	0.884	**0.918**	0.880	**0.922**	0.877	**0.920**	0.889
games	0.961	0.963	0.964	0.962	0.964	0.960	**0.965**	0.953	**0.963**	0.945
free	0.810	0.802	**0.830**	0.783	**0.835**	0.801	**0.840**	0.753	**0.852**	0.756
win / lose	5 / 2		8 / 3		8 / 2		10 / 2		11 / 1	

NOTE: The column "tag name" shows the strings of the target tags. The column pair "ALL" shows the results when entire tame data were used, and the column pairs labeled "1/2" – "1/16" show the results when the tame data were reduced to the 1/2 – 1/16 of the ALL case, respectively. The left "BT" and right "bagg" columns of each pair show the results derived by our BaggTaming and baseline bagging, respectively. Each row shows the accuracies for the corresponding target tag. Bold face indicates that the accuracy was larger than that derived by the other method, and the difference was statistically significant at the significance level of 1% using Z-test statistic. The last row "win/lose" shows the number of target tags for which our method won/lost baseline bagging.

Table 3. Statistics when AcceptLimit and the number of samples were changed

# samples	tame	merged	wild
100%	[0.477 52.55 3/3]	[0.060 15.33 3/3]	[0.004 4.82 4/4]
50%	[0.437 49.68 5/4]	[0.076 17.12 5/2]	[0.009 6.89 5/4]
20%	[0.404 46.40 8/1]	[0.124 24.00 8/2]	[0.032 13.32 7/3]

NOTE: The columns "tame", "merged", and "wild" show results when the tame, merged, and wild options for the AcceptLimit threshold (Section 2.2) were adopted. The "100%" – "20%"rows show the results when the training set size, $|\mathcal{D}_t|$, was set to the 100%, 50%, and 20% of N_T, respectively. Each entry contains three statistics. The left one is the failure ratio, which is the ratio of the failed weak learning in step 9 of BaggTaming to the total number of trials. The middle one is the mean number of trials until the resultant weak classifier is accepted. The right one is the number of wins and loses as shown in Table 2.

qualified weak classifiers, but makes the algorithm slow due to frequent rejections. On the other side, using the wild option accelerates BaggTaming, but weak classifiers can be poorer. The "100%," "50%," and "20%" rows show the results when the training set size, $|\mathcal{D}_t|$, was set to the 100%, 50%, and 20% of the size of the tame set, N_T, respectively. Each entry contains three statistics. The left one is the *failure ratio*, which is the ratio of the failed weak learning in step 9 of the BaggTaming algorithm to the total number of trials. The middle one is *mean trial*, which is the mean number of trials until the resultant weak classifier is accepted. That is to say, the mean number of iterations of the BaggTaming inner loop. The right one is the *win/lose*, which is number of wins and loses as shown in Table 2.

We first discuss the AcceptLimit option. In the case of the tame option, the failure ratio and mean trial are so large that the algorithm get slow. Inversely, inaccurate classifiers are more frequently selected under the wild option; thus, we employed the merged option. We next discuss the sample size, $|\mathcal{D}_t|$. If the number of training examples, $|\mathcal{D}_t|$, grows, the weak learner gets the more chance to acquire better classifiers. However, at the same time, this increases the probability that the non-target wild data are involved in the training set. This size should be determined according as how many samples in the wild data are tagged based on the target concept, but this number is unknown. By considering the balance of the efficiency and effectiveness, we set $|\mathcal{D}_t|$ to the 50% of N_T. We empirically confirmed that BaggTaming worked well for the wide ranges of data sets under this setting. Finally, we also changed the number of weak classifiers, T, from 10 to 100. We observed that the accuracies rises until $T = 30$, but further improvement was not observed. We expected that the reason for this is as follows. As described in section 2.2, bagging cannot reduce the bias factor of the total generalization error. We used naive Bayes, which is rather high-bias, as a weak learner. Therefore, we expected that the error derived by the variance factor would be almost diminished at $T = 30$. In future, we want to test lower-bias weak learners.

In summary, our BaggTaming successfully acquired the more accurate classifier than the classifier that were learned solely from the tame data. It can be concluded that BaggTaming succeed to learn a better classifier by exploiting useful information involved in the wild data.

4 Related Work

There are several machine learning schemes that adopt two types of training data sets. Semi-supervised learning [7] employs both labeled and unlabeled data sets. Because both tame and wild data are labeled, our taming approach clearly differs from semi-supervised learning.

Our taming problem can be considered as a sort of inductive transfers [8–15, 4]. Inductive transfer refers 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[2]. There are many variant tasks in this inductive transfer, such as multitask learning [8] or domain adaptation [11]. However, we may say that there is no strict consensus about the relations or definitions of these tasks in this research community. We consider that these problems are different in their motivation. Multitask learning [8] aims to simultaneously solve multiple tasks. For this aim, the learner have to find the common knowledge shared by these tasks in a form, such as hyper prior or similarity measure. In the case of domain adaptation [11], the tasks other than the target task don't need to be solved. Thus, it is important to find out useful information for solving the target task from the data of the non-target domains. According to this interpretation, we can say that our taming is more relevant to a domain adaptation task. However, constituents of training data sets for non-target tasks are different between taming and domain adaptation. In a case of domain adaptation, a non-target training data is labeled based on some single concept that is related to the target concept. Therefore, it is possible to fit a non-target model to such training data, as in [11]. On the other hand, in a case of taming, because the wild training set includes the examples of many kinds of tasks, it is difficult to fit a simple single model to them due to their diversity. We therefore designed our BaggTaming so as to ignore non-target data in the wild set. Indeed, we tried an approach fitting mixture models for solving the tag prediction task, but no improvement was observed.

Dai et al. recently proposed the problem more related to taming [9]. This problem is treated as inductive transfer, but is more close to domain adaptation according to the above interpretation. They prepared two types of data sets. One is highly similar to the target concept, while the other is less similar. Again, what is the similarity is not so formally defined as above problems, the relation to taming is not yet clear. We plan to investigate the relation between their method and ours.

In the covariate shift [16] approach, the data distributions of test and training data sets are assumed to differ, while the criteria of labeling are the same for both data sets. This problem differs from our taming because in the case of covariate shift one homogeneous training set is assumed.

We finally discuss tagging on Web pages or blog pages. The P-TAG [17] is the system that predicts appropriate tags for given Web pages. Because this prediction is based on the users' personal repository, and other users' tags are not referred to, there is no need for considering inconsistency among tagging criteria. The Autotag [18] is designed to predict proper tags for blog articles. Though other users' tags are used, inconsistency in tags is not taken into account.

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

5 Conclusion

We proposed a new machine problem, which we call *taming*. This exploits two types of data sets, tame and wild. The tame data are labeled based on a consistent concept, while the size of the data set is relatively small. The labels of the wild data can be inconsistent, but are more easily collected. We proposed a method, BaggTaming, for solving this problem. We applied BaggTaming to synthetic and real collaborative tagging data sets, and showed that labels are more accurately predicted by our method.

We plan to enhance the theoretical background, to improve the efficiency by using adaptive sampling, and to acquire more accurate classifier by using other probabilistic models.

Acknowledgments: 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. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. J. of Information Science **32**(2) (2006) 198–208
2. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
3. Breiman, L.: Arcing classifiers. The Annals of Statistics **26**(3) (1998) 801–849
4. Wu, P., Dietterich, T.G.: Improving SVM accuracy by training on auxiliary data sources. In: Proc. of The 21st Int'l Conf. on Machine Learning. (2004) 871–878
5. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. on Information Systems **22**(1) (2004) 5–53
6. McCallum, A., Nigam, K.: A comparison of event model for naive bayes text classification. In: AAAI-98 Workshop on Learning for Text Categorization. (1998) 41–48
7. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-supervised Learning. MIT Press (2006)
8. Caruana, R.: Multitask learning. Machine Learning **28** (1997) 41–75
9. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proc. of The 24th Int'l Conf. on Machine Learning. (2007) 193–200
10. Daumé III, H.: Frustratingly easy domain adaptation. In: Proc. of the 45th Annual Meeting of the Association of Computational Linguistics. (2007) 256–263
11. Daumé III, H., Marcu, D.: Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research **26** (2006) 101–126
12. Do, C.B., Ng, A.Y.: Transfer learning for text classification. In: Advances in Neural Information Processing Systems 18. (2006) 299–306
13. Liao, X., Xue, Y., Carin, L.: Logistic regression with an auxiliary data streams. In: Proc. of The 22nd Int'l Conf. on Machine Learning. (2005) 505–512
14. Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. In: Proc. of The 23rd Int'l Conf. on Machine Learning. (2006) 713–720
15. Thrun, S.: Is learning the n-th thing any easier than learning the first? In: Advances in Neural Information Processing Systems 8. (1996) 640–646
16. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. J. of Statistical Planning and Inference **90** (2000) 227–244
17. Chirita, P.A., Costache, S., Handschuh, S., Nejdl, W.: 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) 845–854
18. Mishne, G.: AutoTag: A collaborative approach to automated tag assignment for Weblog posts. In: Proc. of The 15th Int'l Conf. on World Wide Web. (2006) 953–954