Online Domain Adaptation by Exploiting Labeled Features and Pro-active Learning

*

Raghu Krishnapuram
Ramaiah Institute of Technology
Bangalore, INDIA
raghukrishnapuram@gmail.com

Arun Rajkumar
Conduent Labs
Bangalore, INDIA
Arun.rajkumar@conduent.com

Adithya Acharya
Ramaiah Institute of Technology
Bangalore, INDIA
adithyamundkoor@gmail.com

Nikhil Dhara
Ramaiah Institute of Technology
Bangalore, INDIA
nikhildhara21@gmail.com

Manjunath Goudar
Ramaiah Institute of Technology
Bangalore, INDIA
manju.mg15@gmail.com

Akshay P. Sarashetti
Ramaiah Institute of Technology
Bangalore, INDIA
aps1310@gmail.com

ABSTRACT

Domain adaptation algorithms have gained considerable interest in recent years due to their empirical success. However, most of the existing literature assumes the availability of a pool of unlabeled target domain data which is not the case in several practical scenarios. We consider the problem of online domain adaptation where the target (test) data arrives one at a time, and there is a fixed budget available for obtaining their labels. We propose a novel online domain adaptation algorithm which uses the budget judiciously by balancing the cost and reliability of the oracles which provide the labels. The proposed algorithm uses a probabilistic model which allows the seamless integration of previously unseen features (i.e., features that may appear during test time) into the model. Experiments on several benchmark real world datasets empirically establish the efficacy of the proposed algorithm.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning** ; • **Information systems** → *Decision support systems* ;

KEYWORDS

domain adaptation, online learning, proactive learning

ACM Reference format:
Raghu Krishnapuram, Arun Rajkumar, Adithya Acharya, Nikhil Dhara, Manjunath Goudar, and Akshay P. Sarashetti. 2018. Online Domain Adaptation by Exploiting Labeled Features and Pro-active Learning. In *Proceedings of The ACM India Joint International Conference on Data Science & Management of Data, Goa, India, January 11–13, 2018 (CoDS-COMAD '18),* **??** pages.
https://doi.org/10.1145/3152494.3152507

*The full version of the author's guide is available as `acmart.pdf` document

CoDS-COMAD '18, January 11–13, 2018, Goa, India

ACM ISBN 978-1-4503-6341-9/18/01…$15.00
https://doi.org/10.1145/3152494.3152507

1 INTRODUCTION

One of the fundamental assumptions under which machine learning methods for supervised learning perform well is that the source (training) and the target (test) data follow the same distribution. This is quite a strong assumption, and does not hold in many practical scenarios. Domain adaptation methods overcome this problem by *transferring* the knowledge (in the form of representations or density estimates) obtained in the source domain to the target domain when the source and the target distributions are related but not the same. While these methods have been shown to work well empirically, a common assumption they make is that a pool of labeled examples from the source domain and a pool of unlabeled examples from the target domain are available upfront.

We consider a more realistic and practically-observed case of *online domain adaptation* where the unlabeled target domain data arrives sequentially. For instance, in a sentiment classification problem of a newly launched product, the user reviews become available only one at a time. In this case, most of the existing algorithms, which require a pool of unlabeled target data, are no longer applicable. As the data arrives online, new (previously unseen) features may appear (new keywords in the case of reviews) and the algorithm needs to be capable of handling this situation. Moreover, in practice, multiple source domains are usually available, and hence we also need to decide which domain to adapt from. Optionally, we can often obtain labels for the test data (by using a human expert), but this comes at a high cost.

In this work, we propose a novel algorithm for online pro-active domain adaptation when only a fixed budget is available for labeling. Our algorithm works by observing one target data point at a time. It seamlessly integrates new (previously unseen) features into the classifier as and when they are encountered. A fixed budget for labeling is also an important aspect of the proposed approach. We assume the presence of one *infallible (reliable) but costly* oracle (that has full knowledge of the target domain), and several *fallible but cheap* oracles (which derive their knowledge from existing source domains) for providing labels for target data when requested. The queries to these oracles come at different costs. The algorithm balances cost and reliability and makes a decision to choose the right oracle to query for the label of an target instance whenever needed.

We demonstrate the efficacy of our proposed algorithm by measuring its performance on standard benchmark review data sets as well as image repositories, and by comparing it against standard iterative adaptation algorithms and active learning based approaches.

2 RELATED WORK

The proposed algorithm draws its motivation from different machine learning paradigms such as domain adaptation [**? ? ? ?**], active learning [**? ? ? ? ?**] and pro-active learning [**? ?**]. However, this paper proposes a novel solution to the problem of online domain adaptation by using pro-active learning that has not been explored in the literature.

Domain adaptation methods, e.g., [**?**], attempt to leverage previous knowledge learned in resource-rich source domain(s) for adapting models to a resource-poor target domain. They typically try to minimize the need for labeled data from the target domain and the need for re-training new models from scratch. Approaches to domain adaptation generally assume the availability of target domain data upfront. This data is used for qualitative and quantitative analysis such as estimating the target domain distribution, or computing the similarity with the source domain data, or identifying similarity features [**? ? ?**] . Online domain adaptation settings are more challenging, because the target data is not available upfront but appears sequentially one instance at a time. Hence, traditional domain adaptation approaches are not applicable in these settings. Online transfer learning approaches such as [**?**] address sequential learning, but make the assumption that the labels for the target instances are also available. Other online learning approaches [**? ?**], designed for cases where incremental data can be seen only once, also work under this assumption. Hence, they are not pertinent to the online pro-active domain adaptation scenario.

Active learning [**? ? ? ? ?**] based techniques have come a long way in reducing the need for labeled data requirements. They select a fixed number of informative samples from the pool of unlabeled instances to be annotated by an expert so as to maximize learning. However, active learning relies on strong assumptions which do not hold in many practical settings. These assumptions include availability of unlabeled data upfront and an omniscient oracle that is always correct and charges a uniform cost. Due to these assumptions, active learning based techniques are also not appropriate in the more general online adaptation settings.

Pro-active learning [**? ?**] is a new paradigm that relaxes the harsh assumptions of active learning while providing a significant reduction in the labeling efforts for training the algorithms. Unlike active learning, pro-active learning fixes a maximum budget envelop to select instances and uses oracles with variable costs and reliability. However, pro-active learning for an online domain adaptation scenario, where the unlabeled test data is available in a sequential manner, has not been explored in the literature. Unlike traditional scenarios, the setting we consider, namely, online learning, does not have the flexibility to select informative instances from a pool of unlabeled instances. Therefore, given an overall fixed budget, for each target domain instance, we need to optimize the selection of oracle(s) depending upon the cost and usefulness of the instance.

Table 1: Notations used in this research.

Symbol	Description
$D^{S_k}; D^T$	k-th source domain and target domain
$N_k^S; N^D$	# instances in k-th source domain/target domain
$\mathbf{x}_i^{S_k} \in \mathbb{R}^d;$	Instances from source domain D^{S_k}
$y_i^{S_k} \in \{y_1, \ldots, y_{N_c}\}$	Class labels for N_c classes
$\{\mathbf{x}_i^{S^k}, y_i^{S_k}\}_{i=1:N^{S_k}}$	Labelled instances from D^{S_k}
$\{\mathbf{x}_i^T\}_{i=1:N^T}$	Incremental unlabeled target domain instances
$\mathbf{x}_i^{TL}; \mathbf{x}_i^{TN}$	labeled and new feature set segregation from $\mathbf{x}_i^T$
$O^T; c_T$	the target oracle and its cost
$O^{S_k}; c_k^S; R_i^k$	k-th fallible oracle, its cost and reliability measure
θ_1	Confidence threshold for feature based classifier C_f
θ_2	Confidence threshold for fallible oracle O^{S_k}
$featStat$	Statistics of labeled features

3 ONLINE DOMAIN ADAPTATION

Table **??** lists the notation used in this paper. Our algorithm assumes the availability of one or more source domain(s) D^{S_i} and a target domain D^T. Each source domain provides a sufficient number of labeled instances N^{S_i} to train a model for itself. The target domain data is not available in advance and the unlabeled instances arrive sequentially, i.e., one at a time. Since different domains vary in data distributions, a model trained on an individual (or all) source domains will not generally perform well in classifying the sequential target domain data.

As shown in Figure **??**, the proposed algorithm starts by processing target domain instances one at a time (i.e., as and when they are available). The algorithm first exploits knowledge about features previously found in the source domains to predict the label for the target domain instance and obtains implicit feedback from a feature-based classifier, C_f, in the form of a confidence measure α_1. If α_1 is greater than a predefined threshold, θ_1, the algorithm assigns the predicted label to the target domain instance. Otherwise, it uses pro-active learning to exploit external knowledge sources for explicit feedback. This is done by querying one of the source domain oracles O^{S_i} for the label of the given target domain instance. The source domain oracle O^{S_i} is in general fallible because it is not very familiar with the target domain D^T. If the confidence level α_2 of O^{S_i} is greater than a pre-defined threshold θ_2, the label given by it is accepted. Otherwise, the algorithm will approach the target domain (reliable) oracle O^T, and seek a lable for the target domain instance. The cost of querying the target domain (reliable) oracle O^T is assumed to be much higher than that of querying any of the source domain (fallible) oracles O^{S_i}. In this manner, the proposed algorithm leverages previous knowledge about the features or uses external sources to obtain labels for the target domain instances. The labels may not be correct unless they came from the reliable oracle. The labels are used by an online supervised learning algorithm which is used to classify subsequent target domain instances. The following sub-sections elaborate different stages of the proposed algorithm.

3.1 Exploiting Labeled Features for Classification

Generally, in a standard classification scenario, features are selected based on the training set and the feature space is assumed to be static. In an incremental setting where data from the target domain

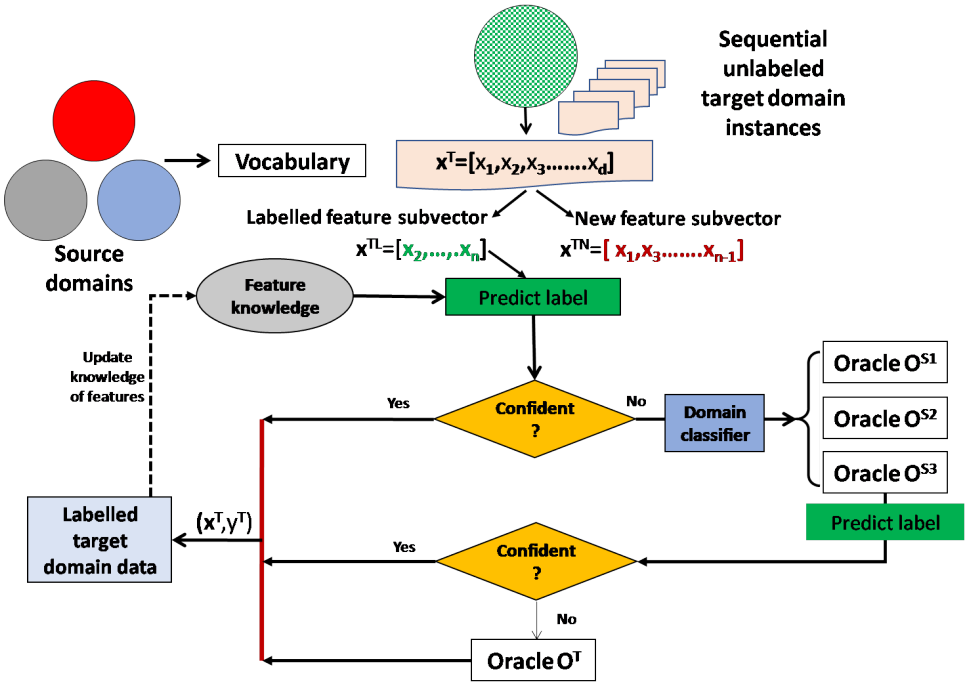

Figure 1: Flow chart of the proposed approach for online domain adaptation.

arrives one instance at a time, we encounter features that may not have been observed earlier. This leads to a scenario where the feature space of the target domain is discovered in a dynamic fashion. We refer to features that have been observed previously as "labeled features", meaning that they have been used for labelling in the past by source domain classifiers. The incremental target domain instances comprise labeled as well as new features. While some prior knowledge can be associated with the labeled features, there is no prior knowledge about the new features that are observed in the target instances which arrive sequentially. To include new features in the classification process, the model needs to be re-trained by adding the new instance to the pool of historical instances. However, frequently re-training the model as well as storing all historical instances is expensive. To overcome this problem, we use a technique that can work with partial feature vectors and account separately for the new features found during the dynamic discovery of the target domain feature space.

The proposed technique segregates an instance $\mathbf{x}_i^T$ from the target domain into *labeled feature* subvector, $\mathbf{x}_i^{TL}$ and *new feature* subvector, $\mathbf{x}_i^{TN}$ such that $\mathbf{x}_i^T = [x_{i1}^T, \ldots, x_{id}^T] = [\mathbf{x}_i^{TL} \,||\, \mathbf{x}_i^{TN}] = [[x_{i1}^{TL}, \ldots, x_{id_L}^{TL}] \,||\, [x_{i1}^{TN}, \ldots, x_{id_N}^{TN}]]$, where d is the dimensionality of $\mathbf{x}_i^T$ (i.e., the size of the vocabulary *Vocab*), d_L is the dimensionality of $\mathbf{x}_i^{TL}$ and d_N is the dimensionality of $\mathbf{x}_i^{TN}$. The algorithm maintains the occurrence statistics for labeled features in a table, referred to as $featStat$, learned from the source domain. For example in a two-way classification scenario, suppose a feature x_i occurred P times with the (+)ve class and N times with the (-)ve class in a source domain, then $featStat$ will store P in the row corresponding to feature x_i and the column corresponding to the (+)ve class. Similarly, it will store N in the row corresponding to feature x_i and the column corresponding to the (-)ve class.These statistics are used in a Naïve Bayes classifier to predicting the label y for a new target domain instance, as shown in (??)

$$\hat{y} = \arg\max_{y_k \in Y} \frac{P(y = y_k) \prod_{j=1,\ldots,d_L} P(x_{ij}^{TL} | y = y_k)}{\sum_{l=1}^{N_c} P(y = y_l) \prod_{j=1,\ldots,d_L} P(x_{ij}^{TL} | y = y_l)} \quad (1)$$

where $x_i j^{TL}$ is the j^{th} feature in the labeled feature set, $P(y = y_k)$ is the prior of the class label y_k, $\hat{y}$ is the predicted label, and Y is the set of class labels.

The Naïve Bayes classifier has been popular as a baseline algorithm in bag-of-words models primarity because of its simplicity (due to the assumption that features are independent). The model is separable in terms of the paramenters corresponding to diffrent features, and allows us to work with partial feature vectors easily. As seen in (??), only labeled features from the target domain instance participate in the classification, since no statistics are available for the new features yet. A feature cover coefficient, θ_i^{cover}, is computed for every new target domain instance $\mathbf{x}_i^T$ as the ratio of the number of labeled features to the number of total features in the instance, as shown in (??).

$$\theta_i^{cover} = \frac{|\mathbf{x}_i^{TL}|}{|\mathbf{x}_i^T|} \quad (2)$$

The posterior class probability computed for the new target domain instance and the cover coefficient are used to associate a confidence measure, α_1, with the predicted label. If α_1 is greater than a pre-defined threshold (θ_1), the predicted label is assigned to the new target domain instance. The predicted label is further used to update the statistics in $featStat$ for the labeled features as well as to initialize the statistics for any new features observed in this

Algorithm 1 Leveraging Labeled Features for Classification

Input: Let $\mathbf{x}_i^{S_k}$ be the i^{th} labeled instance from the k^{th} source domain D^{S_k} with label $y_i^{S_k} \in Y$, let $x_{ij}^{S_k}$ be the value of x_j, the j^{th} feature of $\mathbf{x}_i^{S_k}$. Note that all features that occur in the source domains $D^{S_k}, k = 1, \ldots, K$, are considered to be labeled features. $featStat$ captures the occurrence statistics of all labeled features. Let $Vocab$ be the vocabulary comprising the d_L unique features x_j in all K source domains. Note that $featStat$ is a matrix indexed by the features x_j and class labels y_i.
for $k = 1$ to K & $j = 1$ to d_L **do**
  $featStat[x_j][y_i^{S_k}] += x_{ij}^{S_k}$
end for
Process: $\mathbf{x}_i^T \rightarrow$ Target instance
Segregate $\mathbf{x}_i^T = [\mathbf{x}_i^{TL} \,||\, \mathbf{x}_i^{TN}]$
Predict:
$\hat{y} = \arg\max_{y_k \in Y} P(y = y_k) \prod_{i=1,\ldots,d_L} P(x_{ij}^{TL} | y = y_k)$
& compute confidence α_1 for $\mathbf{x}_i^T$
If $\alpha_1 > \theta_1$
  Update:
  Add features in $\mathbf{x}_i^{TN}$ to $Vocab$ and extend $featStat$
  for $j = 1$ to d_N **do**
    $featStat[x_i][y_i^T] \; += \; x_{ij}^T$
  end for
end If.

target domain instance. The vocabulary $Vocab$, a set consisting of all known features, is expanded by adding new features that are in $\mathbf{x}_i^{TN}$. Once the statistics for the new features are initialized, they are also considered as labeled features from then on. If α_1 is less than a pre-defined threshold, we seek external knowledge using online pro-active learning, as described in the next section. Algorithm **??** summarizes the proposed approach for classifying a new target domain instance using labeled features.

3.2 Online Pro-active Learning

Algorithm **??** using labeled features may not be able to predict the label for the target domain instance with high confidence, and we may want to exploit external knowledge sources (oracles) to seek explicit feedback in the form of a label for the target domain instance. We assume the availability of a target oracle O^T that has complete knowledge about the target domain[1] and therefore, this will be expensive. In addition to O^T, given K source domains, we model K fallible oracles, O^{S_k} $(k = 1, ..., K)$, where each oracle is a classifier trained on the labeled data from the k^{th} source domain D^{S_k}. The fallible oracles are cheaper than the reliable oracle O^T, and their answers are generally correct if the target domain instance is similar to the data they have been trained on. These fallible oracles have different types of expertise, and hence have different *reliabilities* and *costs*. We model these characteristics as follows:

Reliability: The target oracle O^T has the maximum reliability score of 1. The reliability of each of fallible oracles is measured in terms of the confidence of the prediction for a target instance. Since each fallible oracle O^{S_k} is modeled as an SVM classifier trained on the labeled data from the its source domain, the reliability of O^{S_k} in predicting the label of a target domain instance $\mathbf{x}_i^T$ is measured as the distance from decision boundary [**?**] which is computed as shown in (**??**).

$$R_i^k = \frac{d_i^{S_k}}{|\mathbf{w}^{S_k}|} \tag{3}$$

where $d_i^{S_k}$ is the un-normalized output from the SVM of O^{S_k} for $\mathbf{x}_i^T$, and $\mathbf{w}^{S_k}$ is the weight vector associated with the SVM of O^{S_k}.

Cost: We assume that the cost c_k^S associated with each of the fallible oracles O^{S_k} is the same, and the cost c_T associated with the target oracle is such that $c_T >> c_k^S \forall\, k$.

Fallible oracle selection: Assuming that there are K source domains $D^{S_k}, k = 1, \ldots, K$, we train a classifier, H_D, with instances from the source domains. This K-way domain classifier (H_D) is then applied to a new target domain instance to choose the best-matching source domain it can be assigned to. The best-matching source domain is selected to predict the label for the target domain instance at a cost of c_k.

If all oracles had a uniform cost, then selecting the target oracle O^T every time would have been the optimal strategy. However, due to varied costs, an optimal strategy should be to leverage different types of oracles to obtain reasonably reliable labels for as many target domain instances as possible, within a fixed budget.

3.3 Algorithm

Inputs to our algorithm are K source domains D^{S_k} with labeled instances $\{\mathbf{x}_i^{S_k}, y_i^{S_k}\}_{i=1:N^{S_k}}$, a pool of labeled features along with occurrence statistics, $featStats$, a domain classifier H_D, a target oracle O^T with cost c_T per query, K fallible oracles O^{S_k} $(k = 1, ..., K)$ with cost c_k^S per query, and a budget B. We denote by c_i the cost incurred till the i^{th} target domain instance is processed. The proposed algorithm processes an unlabeled target domain instance $\mathbf{x}_i^T$ as described below:

(1) Initialize: $i = 0$ and $c_i = 0$.
(2) While ($c_i \leq B$ OR no more training instances are left):
  - Pick next target domain instance, $\mathbf{x}_i^T$, as it arrives, and segergate it into labeled subvector ($\mathbf{x}_i^{TL}$) and new subvector ($\mathbf{x}_i^{TN}$).
  - Classification is performed using the sub-vector of labeled features $\mathbf{x}_i^{TL}$ as shown in (**??**) (using the current $featStat$), and a confidence score (α_1) is computed for the prediction.
    (a) If $\alpha_1 \geq \theta_1$, the predicted label is assigned as the label for the target domain instance $\mathbf{x}_i^T$.
    (b) Otherwise, the algorithm leverages online pro-active learning.
  - The algorithm first queries one of the the fallible oracles (O^{S_k}) $(k = 1, .., K)$ to seek the label for the instance $\mathbf{x}_i^T$. The domain classifier (H_D) is used to select one of the fallible oracles O^{S_k}.
  - The selected fallible oracle O_{S_k} is queried for the label of the target domain instance at cost c_k^S which is added to the overall cost incurred so far,i.e., $c_i = c_{i-1} + c_k^S$.

[1] Assuming that all of the labeled target domain data is available to this oracle

(a) If it predicts the label with a confidence, $\alpha_2 \geq \theta_2$, then the predicted label is assigned as the label for the target domain instance $\mathbf{x}_i^T$.
(b) Otherwise, the target oracle O^T is queried for the label of the target domain instance at cost c_T which is added to the overall cost incurred, $c_i = c_{i-1} + c_T$.

- Once the label is obtained, the algorithm updates the statistics of labeled features (i.e., $featStat$) and extends it to include the new features. The new features from $\mathbf{x}_i^{TN}$ now move to the pool of labeled features (i.e., the vocabulary $Vocab$).
- i=i+1

(3) If a predetermined number of target domain instances have not been labeled, continue to assign labels to new arrivals using (**??**).
(4) Train/update a classifier to perform classification on the remaining target domain instances as and when they arrive.

As can be seen in the algorithm, the process of adaptation is repeated for every new instance that arrives from the target domain until the budget B is exhausted. If instances are left over after the budget is exhausted, we use the Naïve Bayes classifier as in (**??**), to assign labels to them, assuming that there is no cost associated with this classifier. At the end of this process, we train a classifier on the labeled target domain instances to perform the classification task on the remaining target domain instances as and when they arrive.[2]

4 EXPERIMENTAL EVALUATION

In order to verify whether the proposed approach is general and can work with different types of data, we selected two popular fields where bag-of-words type of representations are used, namely, text data and image data. We conducted similar experiments on both types of data sets. The experimental details are described in the remainder of this section.

4.1 Text Datasets

In the case of text, we evaluated the efficacy of the proposed algorithm for sentiment analysis of user generated data. The huge amount of data available on the web in the form of reviews and short text offers a lot of potential for businesses to analyze opinions of people on a large scale about their products and services. It is essential for businesses to quickly capture the user sentiment and transform this feedback into actionable business insights. User generated content on the web is generally triggered by an event and is short-lived; hence, it is required to analyze it on-the-fly as and when the data is available. These factors motivated us to evaluate the proposed online domain adaptation technique on the online cross-domain sentiment categorization problem.

The dataset that we used in this case is the Amazon review dataset [**?**] which has four different domains, namely, Books, DVDs, Kitchen Appliances and Electronics. Each domain comprises 2, 000 reviews, of which 1, 000 reviews are positive and 1, 000 reviews are negative. All 2000 labeled reviews from the k^{th} source domain are used for training the corresponding fallible oracle O^{S_k}. The dataset was pre-processed by converting it to lowercase followed by stemming and TF-IDF was used for feature weighing. Feature selection based on document frequency ($DF = 5$) was used, which reduces the number of features as well as speeds up the classification task. We evaluated the performance of the proposed algorithm for cross-domain text classification, where labeled instances are available from one or more source domains and the unlabeled target domain data is available one-at-a-time. The performance on a two-class classification task is reported here in terms of classification accuracy.

[2]We can also train a classifier in online manner as and when the labels for the target domain instances are available.However, this is expensive and not useful.

4.2 Experimental Protocol for Text Data

We evaluated the proposed algorithm under two scenarios:

(1) **Scenario 1:** One target oracle and one fallible oracle. In this scenario, the classification pipeline starts with the labeled features classifier followed by the only fallible oracle and then the target oracle.
(2) **Scenario 2:** Multiple fallible oracles and one target oracle. In this scenario, the algorithm needs to select which fallible oracle to query using the domain classifier, H_D.

We compared the efficacy of the proposed approach with (1) traditional active learning [**?**] which assumes the availability of unlabeled data to select most informative instances to query to the target oracle, (2) an iterative similarity-based domain adaptation algorithm [**?**], and (3) the in-domain performance where a classifier is trained and tested on the same target domain data. It is to be noted that all of these algorithms assume the availability of target domain data upfront, and hence the performance is not directly comparable with the performance of the proposed algorithm which operates in an online mode. However, the comparison suggests how well the proposed approach can perform in a more challenging and practical setting relative to its counterparts that work under strict assumptions.

- In active learning, a budget B was allocated to select the most informative samples from a pool of unlabeled instances in the target domain, and to query the oracle for their labels. It assumes the availability of the target oracle O^T at a uniform cost c_T per query. In our comparison, the budget allocated to query the oracle in the active learning algorithm and the proposed algorithm were the same. We used a density weighted uncertainty score [**?**] approach for active learning.
- The iterative adaptation algorithm [**?**] leverages the generalizable knowledge learned from one or more source domains to learn domain-specific features in an iterative fashion from the unlabeled target domain data.
- The in-domain classifier assumes availability of labeled training data from the target domain to learn a classification model. This classifier will give an idea of the best performance that can be achieved via a supervised learning algorithm.

In all of the above mentioned experiments, 1, 600 instances were used for training (or as the unlabeled pool for the active learning and iterative adaptation algorithms) and the performance is reported on the remaining non-overlapping 400 instances. The classification with labeled features is considered confident if the labeled feature

cover coefficient was ≥ 0.7 and the posterior probability was ≥ 0.8[3]. The fallible oracles were modeled as SVM classifiers (with a radial basis function (RBF) kernel) trained on the source domain data. The domain classifier H_D, used to select the fallible oracles, was again an SVM classifier with an RBF kernel.

4.3 Text Data Results and Analysis

Tables **??** and **??** summarize the performance of the proposed approach for adapting the classifier for the target domain. The key observations and analysis are listed below:

- Results suggest that the proposed algorithm enables learning an accurate model in the target domain by transforming the unlabeled sequential data into labeled instances in the most efficient way. This is done by leveraging previous knowledge in terms of labeled features as well as external sources such as oracles.
- Tables **??** and **??** summarize the performance of the algorithm in Scenario 1 and Scenario 2 respectively. Scenario 2, which has multiple source domains, has access to richer information in terms of labeled features as well as to more diverse expertise in the form of fallible oracles, and hence leads to higher accuracy than that of Scenario 1, which is based on a single source domain.
- Results in Tables **??** and **??** show that the proposed algorithm outperforms the traditional active learning approach by 9.8% and 18.4% (on average) in Scenraios 1 and 2 respectively. However, active learning is not applicable as such in the online setting, since it requires that the unlabeled data is available upfront in order to select the most informative samples. This comparison is only meant to show that the proposed approach performs better in a more challenging scenario than what active learning does in a more conservative scenario. The main reason for this improvement is the ability to select the most appropriate oracle for each incrementally available target domain instance. It allows us to obtain feedback for more target domain instances than traditional active learning within the same budget.
- Results in Tables **??** and **??** illustrate that the performance of the proposed algorithm is comparable to that of the iterative similarity-based adaptation algorithm, which is a state-of-the-art static domain adaptation algorithm for cross-domain classification. This is a significant accomplishment, as the proposed algorithm does not assume the availability of test data upfront (that can be used to do qualitative and quantitative analysis), and transforms the target domain instances into labeled instances on-the-fly (i.e. as and when the unlabeled target domain data is encountered).

4.4 Image Dataset

We chose the Amazon, Webcam, DSLR and Caltech-256 data sets for our experiments. The total number of images in this collection is 2533. These data sets have been used by several authors in previous papers [**?** **?** **?** **?**]. The Amazon data is a downloaded collection from the web, whereas the Webcam and DSLR images correspond to low-resolution and high resolution images, respectively. The fourth data set, namely, Caltech-256, was added by the authors of [**?**], to create an additional domain. There are differences in both quality and resolution of images in these data sets. All of the data sets have 10 classes: BACKPACK, TOURING-BIKE, CALCULATOR, HEAD PHONES, COMPUTER-KEYBOARD, LAPTOP-101, COMPUTER-MONITOR, COMPUTER-MOUSE, COFFEEMUG, AND VIDEO-PROJECTOR. The number of samples in each class (in each domain) varies from 8 to 151.

Each image in these data sets is represented (encoded) by a histogram of SURF features [**?**]. There are a total of 800 SURF features. The codebook is trained by a subset of the Amazon images. The frequency of occurrence of each SURF feature is recorded for each image, and then each image is represented by an 800-dimensional histogram. The frequency counts in these histograms range from 0 to 115. The histogram-based feature vectors are similar to the vector space model used for the text data in the previous section. This allows us to use a very similar approach in the realm of images. For example, we could use a Naïve Bayes approach to classify this data, even though it may not give the very best results.

4.5 Experimental Protocol for the Image Dataset

In the case of classification of images represented by SURF features based on labelled features, the probabilities $P(\mathbf{x}_i^{TL}|y_k)$ in (**??**) were computed based on featStat, where y_k is one of the 10 classes. We kept track of only whether the feature occurred in a particular image or not, and built $featStat$ accordingly. The frequency with which the feature occurred in an image is was ignored. More specifically, $featStat$ was a matrix with 800 rows and 10 columns, where the (i, j)-th element of the matrix stored the number of images of class j in which feature i occurred. This representation corresponds to a multinomial approach to classification, where the frequency is set to 1 for all features. (Note: We also tried the multinomial approach that keeps track of the frequency. However, it gave us inferior results.)

There is a numerical issue in computing the confidence score α_1. Since the number of features is very large (800 in the case of images), the probabilities $P(\mathbf{x}_i^{TL}|y_k)$ are very small. Therefore, the product in the numerator of (**??**) will become extremely small. A similar problem occurs in the denominator. While the numerator of (**??**) can be converted to a simple summation by taking the log on both sides of (**??**), the denominator will still remain an issue. To overcome this problem, we have used a different confidence score for the case of images, given by the ratio of the maximum of $P(\mathbf{x}_i^{TL}|y_k)$ over all k to the second highest value. This score can be computed by storing the log values of the probabilities, without performing multiplication or division.

The in-domain performance is the best performance we can hope to reach, assuming we have reliably labelled data for the target domain. We computed this for all target domains by using a 5-fold jackknifing process, i.e., we used 80% of the data for training and 20% of the data for testing, using reliable labels for all data. We repeated this 5 times to compute the average in-domain performance. The value of "C" used in the SVM algorithm (with a linear kernel) was determined by grid search for each domain [**?**]. The value was 0.001

[3]These thresholds are set empirically on a held-out validation set.

Table 2: Results comparing the performance of the proposed algorithm with the static iterative adaptation algorithm, active learning, and the in-domain performance for Scenario 1 comprising one target domain and one source domain. In this table, B refers to the books domain, K to the kitchen domain, E to the electronics domain, and D to the DVDs domain in the Amazon dataset.

	Target	Source	Proposed	Iterative adaptation	Active learning	In-domain
Scenario 1	B	E	77.8	78.9	66.5	80.4
		K	73.2	74.8		
		D	79.3	80.0		
	E	B	78.4	80.3	68.4	84.4
		D	74.6	76.4		
		K	70.1	83.7		
	K	B	78.6	80.1	70.3	87.7
		D	81.3	82.0		
		E	86.0	87.9		
	D	B	79.9	81.9	67.2	82.4
		E	78.4	80.1		
		K	77.8	78.8		

Table 3: Results comparing the performance of the proposed algorithm with the static iterative adaptation algorithm, active learning, and in-domain performance for Scenario 2 comprising one target domain and multiple source domains.

	Target	Source	Proposed	Iterative adaptation	Active learning
Scenario 2	B	E, K, D	83.4	85.6	66.5
	E	B, K, D	87.8	89.6	68.4
	K	B, E, D	89.7	91.7	70.3
	D	B, E, K	85.2	86.4	67.2

Table 4: Results comparing the performance of the proposed algorithm with those of an in-domain classifier. The budgets leading to a prediction accuracy within 10%-20% after adaptation are shown, along with the fraction of instances handled by the Naïve Bayes classifier, the fallible oracles, and the target domain oracle.

Target	Source	Budget	NB frac	FO frac	RO frac	In-dom SVM	In-dom NB	Randm Smpl	Test acc	% shortfall
Amazon	Caltech	2266	0.22	0.03	0.29	71.1	68.7	48.3	57.8	18.7
	DSLR	6766	0.24	0.20	0.56				76.5	19.9
	Web	5766	0.44	0.04	0.53				56.9	20.1
Caltech	Amazon	4898	0.36	0.00	0.39	53.8	51.7	48.1	43.7	18.7
	DSLR	4898	0.16	0.03	0.52				43.0	20.0
	Web	4898	0.19	0.02	0.52				43.5	19.2
DSLR	Amazon	325	0.17	0.81	0.02	79.4	75.0	31.9	64.4	18.9
	Caltech	125	0.28	0.70	0.02				71.2	10.2
	Web	125	1.00	0.00	0.00				71.2	10.2
Web	Amazon	1456	0.18	0.00	0.52	88.5	83.7	31.2	72.5	18.0
	Caltech	556	0.22	0.56	0.18				73.9	16.5
	DSLR	256	0.79	0.20	0.01				71.9	18.8
	Average	2695	0.35	0.22	0.30	73.19	69.77	37.07	62.20	17.43

for the Amazon and Caltech-256 datasets, 0.005 for DSLR and 0.1 for Web.

While there have been several attempts to generalize various active learning approaches to multiple classes, we are not aware of a well-accepted method that could be used for comparison purposes here. Hence, we have included the results of random sampling, which is based on choosing a number of samples from the training data in such a way that the cost of obtaining the labels for them from the reliable oracle would be the same as the cost incurred by the proposed approach from a domain that sends the lowest number of queries to the target oracle O^T. The random sampling was repeated using a 5-fold jackknifing process, and the results were averaged.

For the fallible oracle selection, intead of building the classifier H_D as described in the algorithm in Section **??**, for every target domain instance $\mathbf{x}_i^T$ that is not processed by the Naïve Bayes algorithm, we query all fallible oracles. We compute the reliability score for each oracle using (**??**), and choose the label given by the oracle that has the best reliability score. While this approach incurs a slightly higher cost (3 units, instead of 1 unit), we found it to perform marginally better. The value of "C" used in the SVM algorithm was the same as the optimal value used in computing the in-domain performance.

The online domain adaptation results reported here are again based on a 5-fold jackknifing approach. We used 80% of the data from the target domain for training, and 20% of the data for testing. (We used "stratified shuffle split" in in the "sklearn" python libary.) The training data was presented to the algorithm one instance at a time, until the budget was exhausted. The cost of querying the Naïve Bayes classifier was assumed to be zero. The cost for the fallible oracles, O^{S_k}, $k = 1, \ldots, 4$, was 1 and the cost for the target oracle, O^T, was 10. We continued the online adaptation process until either the budget was exhausted or all the training instances were exhausted. If any training instances are still remaining at the end of the process (because the budget was exhausted), we generated labels for these instances using the zero-cost Naïve Bayes classifier (ignoring the confidence score). At the end of the adaptation, we

trained an SVM (with a linear kernel) with the set of reliable and unreliable labels that were available, to build a classifier. We tested the accuracy of the classifier on the remaining 20% of the target domain data. This process was repeated 5 times (5-fold jackknifing), and the accuracy numbers were averaged.

In the online domain adaptation algorithm, we need to choose the confidence threshold θ_1 for the Naïve Bayes classifier. There is also another confidence threshold θ_2 associated with the fallible oracles. In a practical setting, they need to be chosen based on a validation process. For example, we can approach the target oracle O^T to label a small number of instances, and then choose the combination of θ_1 and θ_2 that will produce the closest set of labels. However, in our experiments, we chose these values by trial and error. It was difficult to pick a single value for θ_1 or θ_2 that works for all four target domains. However, we noticed that it is possible to choose θ_1 and θ_2 values that worked well for a given target domain, and the results did not vary significantly around the chosen values. The value of θ_1 was typically between 10^4 and 10^8 for the four domains, and the value of θ_2 was between 0.5 and 1.9.

4.6 Results and Analysis

We ran the proposed online pro-active learning algorithm for domain adaptation with each of four domains, namely Amazon, Caltech, DSLR, and Web as the target domain. For a particular target domain, the adaptation can start from any of the remaining three source doamins. In other words, $featStat$ can be initialized based on any of the remaining three domains. This gives rise to a total of $3 \times 4 = 12$ domain adaptation cases. In each case, there are three fallible oracles, which are trained as described in the previous subsection. Each of the 12 adaptation cases was run for a variety of budgets. Given that the cost c_k^S was 1 for all k, and the cost c_T was 10, to allow for the case when we end up querying the reliable (target domain) oracle for all training data (which was 80% of the total instances N^D in the target domain), the budget values ranged from 0 to $0.8 \times N^D \times 10$, with suitable increments.

Table **??** summarizes the results for the values of the budget that gave rise to classification accuracies that were within 10%-20% of the ideal accuracy achieved by an in-domain classifier. The table shows the budget value, the fraction of instances handled by the Naïve Bayes classifier (NB frac), the fraction of instances handled by the set of fallible oracles (FO frac), the fraction of instances handled by the reliable (target domain) oracle (RO frac), the in-domain performance using an SVM (In-dom SVM), the in-domain performance using Naïve Bayes (In-dom NB), the testing accuracy after the adaptation process is terminated (Test acc), and the shortfall in the testing accuracy with respect to the in-domain accuracy (Shortfall). The last row of the table shows the average values under each heading. It can be seen that the Naïve Bayes classifier handles anywhere between 16% and 100% of the instances, and the fallible oracles handle anywhere between 0% and 81%. Only for between 0% and 56% of the instances the algorithm resorts to seeking the opinion of the reliable orcle, O^T. On average, only about 30% of the instances are handled by O^T, to achieve an accuracy within 17.43% of the ideal. It can also be seen that the performance of the in-domain Naïve Bayes classifier is quite comparable to that of SVM, which again justifies its use as the first-level implicit feedback mechanism. The results of the random sampling method is also shown as a benchmark. It can be seen that the accuracy level of the proposed algorithm is generally much higher than that of random sampling.

While we have only presented typical results in Table **??**, Our experiments also showed that the average fraction of instances handled by O^T for a shortfall of about 8% is 0.52, and the average fraction of instances handled by O^T for a shortfall of about 26% is 0.15. These results suggest that the unreliable oracles do a good job of making up for lack of labels.

5 CONCLUSION

In this paper, we present a novel approach to online domain adaptation where unlabeled data is assumed to arrive in the target domain in a sequential fashion, i.e., one at a time. Our approach transforms unlabeled instances into labeled instances in an efficient manner by using pro-active learning, i.e., it leverages knowledge from previous source domain(s) or an external source, in a cost-effective manner. The approach exploits the fact that implicit feedback can be obtained via labeled features and the use of a classifier whose model parameters are separable with respect to the individual features. In our case, we have chosen the Naïve Bayes algorithm for its simplicity and known effectiveness. This allows us to add new (hitherto unseen) features that my occur in the target domain in a natural way. In addition, the proposed algorithm leverages explicit feedback from fallible oracles which are constructed based on knowledge available in other source domains. The target domain oracle is approached only when the implicit feedback and the explicit feedback fail. This results in an online pro-active strategy for adaptation. To our knowledge, such an approach for online domain adaptation has not been explored earlier, and hence it is the major contribution of this paper. The proposed approach is particularly suitble for domains that have text or image data. This is because in these domains, bag-of-(visual)-words models have been shown to work well. Our experimental results on text and image data clearly demonstrate that the proposed approach can be used for online domain adaptation in an efficient and effective manner. More specifically, the experimental results in the text domain show that the proposed algorithm outperforms traditional active learning techniques and performs comparably with the static domain adaptation techniques, both of which assume upfront availability of target domain data. In the case of image data, our results show that we can achieve a high degree of adapatation even with a small fraction of instances being labeled by a target domain (reliable) oracle.

ACKNOWLEDGMENTS

The authors are deeply indebted to Himanshu Sharad Bhatt, a former colleague of the first two authors (who is currently at American Express Big Data Labs, India), for his contributions to many of the ideas in this paper and for providing us help with the experiments on the text datasets.

REFERENCES

[] Vamshi Ambati. 2012. *Active Learning and Crowdsourcing for Machine Translation in Low Resource Scenarios.* Ph.D. Dissertation. Pittsburgh, PA, USA. Advisor(s) Carbonell, Jaime. AAI3528171.

[] H. Bay and L. Tuytelaars, T.and Van Gool. 2013. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision*. 404–417.

[] H. S. Bhatt, D. Semwal, and S. Roy. 2015. An Iterative Similarity based Adaptation Technique for Cross-domain Text Classification. In *Proceedings of CoNLL*.

[] J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of Association for Computational Linguistics*. 187–205.

[] J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*. 120–128.

[] Avrim Blum. 1996. On-Line Algorithms in Machine Learning. In *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl*. 306–325.

[] Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* (2009).

[] Pinar Donmez and JaimeG. Carbonell. 2010. From Active to Proactive Learning Methods. In *Advances in Machine Learning I*. Studies in Computational Intelligence, Vol. 262. 97–120.

[] Pinar Donmez and Jaime G. Carbonell. 2008. Paired-Sampling in Density-Sensitive Active Learning. In *International Symposium on Artificial Intelligence and Mathematics*.

[] Pinar Donmez and Jaime G. Carbonell. 2008. Proactive Learning: Cost-sensitive Active Learning with Multiple Imperfect Oracles. In *Proceedings of Conference on Information and Knowledge Management*. 619–628.

[] B. Gong, Y. Shi, F. Sha, and K Grauman. 2012. Geodesic Flow Kernel for Unsupervised Domain Adaptation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

[] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko. 2013. Efficient Learning of Domain-invariant Image Representations. In *Proceedings of the International Conference on Learning Representations*.

[] C. W. Hsu, C. C. Chang, and C. J. Lin. 2003. *A practical guide to support vector classification*. Technical Report. Department of Computer Science, National Taiwan University.

[] Abhishek Kumar, Piyush Rai, and Hal Daumé III. 2011. Co-regularized Multi-view Spectral Clustering. In *Proceedings of International Conference on Neural Information Processing Systems*.

[] D. Lewis and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR*. 3–12.

[] Seungwhan Moon, Calvin McCarter, and Yu-Hsin Kuo. 2014. Active Learning with Partially Featured Data. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. ACM, New York, NY, USA, 1143–1148. https://doi.org/10.1145/2567948.2580062

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

[] N. Roy and A. McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of International Conference on Machine Learning*. 441–448.

[] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. 2010. Adapting Visual Category Models to New Domains. In *Proceedings of European Conference on Computer Vision*. 213–226.

[] Shai Shalev-Shwartz. 2012. Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning* 4, 2 (2012), 107–194.

[] B. Sun, J. Feng, and K. Saenko. 2016. Return of frustratingly easy domain adaptation. In *Proceedings of AAAI*.

[] Shiliang Sun, Honglei Shi, and Yuanbin Wu. 2015. A survey of multi-source domain adaptation. *Information Fusion* 24 (2015), 84 – 92. https://doi.org/10.1016/j.inffus.2014.12.003

[] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari. 2011. A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification. *, IEEE Journal of Selected Topics in Signal Processing* 5, 3 (2011), 606–617.

[] J. Yu and J. Jiang. 2015. A Hassle-Free Unsupervised Domain Adaptation Method Using Instance Similarity Features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 168–173.

[] P. Zhao and S. Hoi. 2010. OTL: A Framework of Online Transfer Learning. In *Proceedings of International Conference on Machine Learning*. 1231–1238.