Multi-Domain Learning: When Do Domains Matter?

Mahesh Joshi
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
maheshj@cs.cmu.edu

Mark Dredze
Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, Maryland 21211
mdredze@cs.jhu.edu

William W. Cohen
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
wcohen@cs.cmu.edu

Carolyn P. Rosé
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
cprose@cs.cmu.edu

Abstract

We present a systematic analysis of existing multi-domain learning approaches with respect to two questions. First, many multi-domain learning algorithms resemble ensemble learning algorithms. (1) Are multi-domain learning improvements the result of ensemble learning effects? Second, these algorithms are traditionally evaluated in a balanced class label setting, although in practice many multi-domain settings have domain-specific class label biases. When multi-domain learning is applied to these settings, (2) are multi-domain methods improving because they capture domain-specific class biases? An understanding of these two issues presents a clearer idea about where the field has had success in multi-domain learning, and it suggests some important open questions for improving beyond the current state of the art.

1 Introduction

Research efforts in recent years have demonstrated the importance of domains in statistical natural language processing. A mismatch between training and test domains can negatively impact system accuracy as it violates a core assumption in many machine learning algorithms: that data points are independent and identically distributed (*i.i.d.*). As a result, numerous domain adaptation methods (Chelba and Acero, 2004; Daumé III and Marcu, 2006; Blitzer et al., 2007) target settings with a training set from one domain and a test set from another.

Often times the training set itself violates the *i.i.d.* assumption and contains multiple domains. In this case, training a single model obscures domain distinctions, and separating the dataset by domains reduces training data. Instead, multi-domain learning (MDL) can take advantage of these domain labels to improve learning (Daumé III, 2007; Dredze and Crammer, 2008; Arnold et al., 2008; Finkel and Manning, 2009; Zhang and Yeung, 2010; Saha et al., 2011). One such example is sentiment classification of product reviews. Training data is available from many product categories and while all data should be used to learn a model, there are important differences between the categories (Blitzer et al., 2007)[1].

While much prior research has shown improvements using MDL, this paper explores what properties of an MDL setting matter. Are previous improvements from MDL algorithms discovering important distinctions between features in different domains, as we would hope, or are other factors contributing to learning success? The key question of this paper is: when do domains matter?

Towards this goal we explore two issues. First, we explore the question of whether domain distinctions are used by existing MDL algorithms in meaningful ways. While differences in feature behaviors between domains will hurt performance (Blitzer et al., 2008; Ben-David et al., 2009), it is not clear if the improvements in MDL algorithms can be attributed to correcting these errors, or whether they are benefiting from something else. In particular, there are many similarities between MDL and ensemble methods, with connections to instance bag-

[1] Blitzer et al. (2007) do not consider the MDL setup, they consider a single source domain, and a single target domain, with little or no labeled data available for the target domain.

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1302–1312, Jeju Island, Korea, 12–14 July 2012.

ging, feature bagging and classifier combination. It may be that gains in MDL are the usual ensemble learning improvements.

Second, one simple way in which domains can change is the distribution of the prior over the labels. For example, reviews of some products may be more positive on average than reviews of other product types. Simply capturing this bias may account for significant gains in accuracy, even though nothing is learned about the behavior of domain-specific features. Most prior work considers datasets with balanced labels. However, in real world applications, where labels may be biased toward some values, gains from MDL could be attributed to simply modeling domain-specific bias. A practical advantage of such a result is ease of implementation and the ability to scale to many domains.

Overall, irrespective of the answers to these questions, a better understanding of the performance of existing MDL algorithms in different settings will provide intuitions for improving the state of the art.

2 Multi-Domain Learning

In the multi-domain learning (MDL) setting, examples are accompanied by both a class label and a domain indicator. Examples are of the form $(\boldsymbol{x}_i, y_i, d_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^N$, d_i is a domain indicator, $\boldsymbol{x}_i$ is drawn according to a fixed domain-specific distribution D_{d_i}, and y_i is the label (e.g. $y_i \in \{-1, +1\}$ for binary labels). Standard learning ignores d_i, but MDL uses these to improve learning accuracy.

Why should we care about the domain label? Domain differences can introduce errors in a number of ways (Ben-David et al., 2007; Ben-David et al., 2009). First, the domain-specific distributions D_{d_i} can differ such that they favor different features, i.e. $p(\boldsymbol{x})$ changes between domains. As a result, some features may only appear in one domain. This aspect of domain difference is typically the focus of unsupervised domain adaptation (Blitzer et al., 2006; Blitzer et al., 2007). Second, the features may behave differently with respect to the label in each domain, i.e. $p(y|\boldsymbol{x})$ changes between domains. As a result, a learning algorithm cannot generalize the behavior of features from one domain to another. The key idea behind many MDL algorithms is to target one or both of these properties of domain difference to improve performance.

Prior approaches to MDL can be broadly categorized into two classes. The first set of approaches (Daumé III, 2007; Dredze et al., 2008) introduce parameters to capture domain-specific behaviors while preserving features that learn domain-general behaviors. A key of these methods is that they do not explicitly model any relationship between the domains. Daumé III (2007) proposes a very simple "easy adapt" approach, which was originally proposed in the context of adapting to a specific target domain, but easily generalizes to MDL. Dredze et al. (2008) consider the problem of learning how to combine different domain-specific classifiers such that behaviors common to several domains can be captured by a shared classifier, while domain-specific behavior is still captured by the individual classifiers. We describe both of these approaches in § 3.2.

The second set of approaches to MDL introduce an explicit notion of relationship between domains. For example, Cavallanti et al. (2008) assume a fixed task relationship matrix in the context of online multi-task learning. The key assumption is that instances from two different domains are half as much related to each other as two instances from the same domain. Saha et al. (2011) improve upon the idea of simply using a fixed task relationship matrix by instead *learning* it adaptively. They derive an online algorithm for updating the task interaction matrix.

Zhang and Yeung (2010) derive a convex formulation for adaptively learning domain relationships. We describe their approach in § 3.2. Finally, Daumé III (2009) proposes a joint task clustering and multi-task/multi-domain learning setup, where instead of just learning pairwise domain relationships, a hierarchical structure among them is inferred. Hierarchical clustering of tasks is performed in a Bayesian framework, by imposing a hierarchical prior on the structure of the task relationships.

In all of these settings, the key idea is to learn both domain-specific behaviors and behaviors that generalize between (possibly related) domains.

3 Data

To support our analysis we develop several empirical experiments. We first summarize the datasets and methods that we use in our experiments, then

proceed to our exploration of MDL.

3.1 Datasets

A variety of multi-domain datasets have been used for demonstrating MDL improvements. In this paper, we focus on two datasets representative of many of the properties of MDL.

Amazon (AMAZON) Our first dataset is the Multi-Domain Amazon data (version 2.0), first introduced by Blitzer et al. (2007). The task is binary sentiment classification, in which Amazon product reviews are labeled as positive or negative. Domains are defined by product categories. We select the four domains used in most studies: `books`, `dvd`, `electronics` and `kitchen appliances`.

The original dataset contained 2,000 reviews for each of the four domains, with 1,000 positive and 1,000 negative reviews per domain. Feature extraction follows Blitzer et al. (2007): we use case insensitive unigrams and bigrams, although we remove rare features (those that appear less than five times in the training set). The reduced feature set was selected given the sensitivity to feature size of some of the MDL methods.

ConVote (CONVOTE) Our second dataset is taken from segments of speech from United States Congress floor debates, first introduced by Thomas et al. (2006). The binary classification task on this dataset is that of predicting whether a given speech segment supports or opposes a bill under discussion in the floor debate. We select this dataset because, unlike the AMAZON data, CONVOTE can be divided into domains in several ways based on different metadata attributes available with the dataset. We consider two types of domain divisions: the bill identifier and the political party of the speaker. Division based on the bill creates domain differences in that each bill has its own topic. Division based on political party implies preference for different issues and concerns, which manifest as different language. We refer to these datasets as BILL and PARTY.

We use Version 1.1 of the CONVOTE dataset, available at `http://www.cs.cornell.edu/home/llee/data/convote.html`. More specifically, we combine the training, development and test folds from the `data_stage_three/` version, and sub-sample to generate different versions of the dataset required for our experiments. For BILL we randomly sample speech segments from three different bills. The three bills and the number of instances for each were chosen such that we have sufficient data in each fold for every experiment. For PARTY we randomly sample speech segments from the two major political parties (Democrats and Republicans). Feature processing was identical to AMAZON, except that the threshold for feature removal was two.

3.2 Learning Methods and Features

We consider three MDL algorithms, two are representative of the first approach and one of the second approach (learning domain similarities) (§2). We favored algorithms with available code or that were straightforward to implement, so as to ensure reproducibility of our results.

FEDA Frustratingly easy domain adaptation (`FEDA`) (Daumé III, 2007; Daumé III et al., 2010b; Daumé III et al., 2010a) is an example of a classifier combination approach to MDL. The feature space is a cross-product of the domain and input features, augmented with the original input features (shared features). Prediction is effectively a linear combination of a set of domain-specific weights and shared weights. We combine `FEDA` with both the SVM and logistic regression algorithms described below to obtain `FEDA-SVM` and `FEDA-LR`.

MDR Multi-domain regularization (`MDR`) (Dredze and Crammer, 2008; Dredze et al., 2009) extends the idea behind classifier combination by explicitly formulating a classifier combination scheme based on Confidence-Weighted learning (Dredze et al., 2008). Additionally, classifier updates (which happen in an online framework) contain an explicit constraint that the combined classifier should perform well on the example. Dredze et al. (2009) consider several variants of `MDR`. We select the two best performing methods: `MDR-L2`, which uses the underlying algorithm of Crammer et al. (2008), and `MDR-KL`, which uses the underlying algorithm of Dredze et al. (2008). We follow their approach to classifier training and parameter optimization.

MTRL The multi-task relationship learning (`MTRL`) approach proposed by Zhang and Yeung

(2010) achieves states of the art performance on many MDL tasks. This method is representative of methods that learn similarities between domains and in turn regularize domain-specific parameters accordingly. The key idea in their work is the use of a matrix-normal distribution $p(\boldsymbol{X}|\boldsymbol{M}, \boldsymbol{\Omega}, \boldsymbol{\Sigma})$ as a prior on the matrix $\boldsymbol{W}$ created by column-wise stacking of the domain-specific classifier weight vectors. $\boldsymbol{\Omega}$ represents the covariance matrix for the variables along the columns of $\boldsymbol{X}$. When used as a prior over $\boldsymbol{W}$ it models the covariance between the domain-specific classifiers (and therefore the tasks). $\boldsymbol{\Omega}$ is learned jointly with the domain-specific classifiers. This method has similar benefits to `FEDA` in terms of classifier combination, but also attempts to model domain relationships. We use the implementation of `MTRL` made available by the authors[2]. For parameter tuning, we perform a grid search over the parameters λ_1 and λ_2, using the following values for each (a total of 36 combinations): $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$.

In addition to these multi-task learning methods, we consider a common baseline: ignoring the domain distinctions and learning a single classifier over all the data. This reflects single-domain learning, in which no domain knowledge is used and will indicate baseline performance for all experiments. While some earlier research has included a separate *one classifier per domain* baseline, it almost always performs worse, since splitting the domains provides much less data to each classifier (Dredze et al., 2009). So we omit this baseline for simplicity.

To obtain a single classifier we use two classification algorithms: SVMs and logistic regression.

Support Vector Machines A single SVM run over all the training data, ignoring domain labels. We use the SVM implementation available in the LIBLINEAR package (Fan et al., 2008). In particular, we use the L_2-regularized L_2-loss SVM (option `-s 1` in version 1.8 of LIBLINEAR, and also option `-B 1` for including a standard bias feature). We tune the SVM using five-fold stratified cross-validation on the training set, using the following values for the trade-off parameter C: $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 1\}$.

[2] `http://www.cse.ust.hk/~zhangyu/codes/MTRL.zip`

Logistic Regression (LR) A single logistic regression model run over all the training data, ignoring domain labels. Again, we use the L_2-regularized LR implementation available in the LIBLINEAR package (option `-s 0`, and also option `-B 1`). We tune the LR model using the same strategy as the one used for SVM above, including the values of the trade-off parameter C.

For all experiments, we measure average accuracy over K-fold cross-validation, using 10 folds for AMAZON, and 5 folds for both BILL and PARTY.

4 When Do Domains Matter?

We now empirically explore two questions regarding the behavior of MDL.

4.1 Ensemble Learning

Question: *Are MDL improvements the result of ensemble learning effects?*

Many of the MDL approaches bear a striking resemblance to ensemble learning. Traditionally, ensemble learning combines the output from several different classifiers to obtain a single improved model (Maclin and Opitz, 1999). It is well established that ensemble learning, applied on top of a diverse array of quality classifiers, can improve results for a variety of tasks. The key idea behind ensemble learning, that of combining a diverse array of models, has been applied to settings in which data preprocessing is used to create many different classifiers. Examples include instance bagging and feature bagging (Dietterich, 2000).

The core idea of using diverse inputs in making classification decisions is common in the MDL literature. In fact, the top performing and only successful entry to the 2007 CoNLL shared task on domain adaptation for dependency parsing was a straightforward implementation of ensemble learning by creating variants of parsers (Sagae and Tsujii, 2007). Many MDL algorithms, among them Dredze and Crammer (2008), Daumé III (2009), Zhang and Yeung (2010) and Saha et al. (2011), all include some notion of learning domain-specific classifiers on the training data, and combining them in the best way possible. To be clear, we *do not* claim that these approaches can be reduced to an existing ensemble learning algorithm. There are crucial elements

in each of these algorithms that separate them from existing ensemble learning algorithms. One example of such a distinction is the learning of domain relationships by both Zhang and Yeung (2010) and Saha et al. (2011). However, we argue that their core approach, that of combining parameters that are trained on variants of the data (all data or individual domains), is an ensemble learning idea.

Consider instance bagging, in which multiple classifiers are each trained on random subsets of the data. The resulting classifiers are then combined to form a final model. In MDL, we can consider each domain a subset of the data, albeit non-random and non-overlapping. The final model combines the domain-specific parameters and parameters trained on other instances, which in the case of FEDA are the shared parameters. In this light, these methods are a complex form of instance bagging, and their development could be justified from this perspective.

However, given this justification, are improvements from MDL simply the result of standard ensemble learning effects, or are these methods really learning something about domain behavior? If knowledge of domain was withheld from the algorithm, could we expect similar improvements? As we will do in each empirical experiment, we propose a contrarian hypothesis:

Hypothesis: *Knowledge of domains is irrelevant for MDL.*

Empirical Evaluation We evaluate this hypothesis as follows. We begin by constructing a true MDL setting, in which we attempt to improve accuracy through knowledge of the domains. We will apply three MDL algorithms (FEDA, MDR, and MTRL) to our three multi-domain datasets (AMAZON, BILL, and PARTY) and compare them against a single classifier baseline. We will then withhold knowledge of the true domains from these algorithms and instead provide them with random "pseudo-domains," and then evaluate the change in their behavior. The question is whether we can obtain similar benefits by ignoring domain labels and relying strictly on an ensemble learning motivation (instance bagging).

For the "True Domain" setting, we apply the MDL algorithms as normal. For the "Random Domain" setting, we randomly shuffle the domain labels within a given class label within each fold, thus maintaining the same number of examples for each domain label, and also retaining the same class distribution within each randomized domain. The resulting "pseudo-domains" are then similar to random subsets of the data used in ensemble learning.

Following the standard practice in previous work, for this experiment we use a balanced number of examples from each domain and a balanced number of positive and negative labels (no class bias). For AMAZON (4 domains), we have 10 folds of 400 examples per fold, for BILL (3 domains) 5 folds of 60 examples per fold, and for PARTY (2 domains) 5 folds of 80 examples per fold. In the "Random Domain" setting, since we are randomizing the domain labels, we increase the number of trials. We repeat each cross-validation experiment 5 times with different randomization of the domain labels each time.

Results Results are shown in Table 1. The first row shows absolute (average) accuracy for a single classifier trained on all data, ignoring domain distinctions. The remaining cells indicate absolute improvements against the baseline.

First, we note for the well-studied AMAZON dataset that our results with true domains are consistent with the previous literature. FEDA is known to not improve upon a single classifier baseline for that dataset (Dredze et al., 2009). Both MDR−L2 and MDR−KL improve upon the single classifier baseline, again as per Dredze et al. (2009). And finally, MTRL also improves upon the single classifier baseline. Although the MTRL improvement is not as dramatic as in the original paper[3], the average accuracy that we achieve for MTRL (84.2%) is better than the best average accuracy in the original paper (83.65%).

The main comparison to make in Table 1 is between having knowledge of true domains or not. "Random Domain" in the table is the case where domain identifiers are randomly shuffled within a given fold. Ignoring the significance test results for now, overall the results indicate that knowing the true domains is useful for MDL algorithms. Randomizing the domains does not work better than knowing true domains in any case. However, in all except one case, the improvements of MDL algorithms are

[3]This might be due to a different version of the dataset being used in a cross-validation setup, rather than their train/test setup, and also because of differences in baseline approaches.

	AMAZON		BILL		PARTY	
	SVM	**LR**	**SVM**	**LR**	**SVM**	**LR**
	Single Classifier					
	83.93%	83.78%	66.67%	68.00%	62.75%	64.00%
	FEDA					
True Domain	-0.35	-0.10	+2.33	+ 1.00	+4.25 ▲	+1.25
Random Domain	-1.30 ▼	-1.02 ▼	-1.20	-2.07	-2.05	-2.10
	MDR−L2					
True Domain	+1.87 ▲	+2.02 ▲	+0.00	-1.33	+2.25	+1.00
Random Domain	+0.91 ▲	+1.07 ▲	-2.67	-4.00	-2.80	-4.05
	MDR−KL					
True Domain	+1.85 ▲	+2.00 ▲	+1.00	-0.33	+3.00	+1.75
Random Domain	+1.36 ▲	+1.51 ▲	+0.60	-0.73	-1.30	-2.55 ▼
	MTRL					
True Domain	+0.27	+0.42	+0.67	-0.67	+1.50	+0.25
Random Domain	-0.37	-0.21	-1.47	-2.80	-3.55	-4.80

Table 1: A comparison between MDL methods with access to the "True Domain" labels and methods that use "Random Domain" information, essentially ensemble learning. The first row has raw accuracy numbers, whereas the remaining entries are absolute improvements over the baseline. ▲: Significantly better than the corresponding SVM or LR baseline, with $p < 0.05$, using a paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a paired t-test.

significantly better only for the AMAZON dataset[4]. And interestingly, exactly in the same case, randomly shuffling the domains also gives significant improvements compared to the baseline, showing that there is an ensemble learning effect in operation for MDR−L2 and MDR−KL on the AMAZON dataset. For FEDA, randomizing the domains significantly hurts its performance on the AMAZON data, as is the case for MDR−KL on the PARTY data. Therefore, while our contrarian hypothesis about irrelevance of domains is not completely true, it is indeed the case that some MDL methods benefit from the ensemble learning effect.

A second observation to be made from these results is that, while all of empirical research on MDL assumes the definition of domains as a given, the question of how to split a dataset into domains given various metadata attributes is still open. For example, in our experiments, in general, using the political party as a domain distinction gives us more improvements over the corresponding baseline approach[5].

We provide a detailed comparison of using true vs. randomized domains in Table 6, after presenting the second set of experimental results.

4.2 Domain-specific Class Bias

Question: *Are MDL methods improving because they capture domain-specific class biases?*

In previous work, and the above section, experiments have assumed a balanced dataset in terms of class labels. It has been in these settings that MDL methods improve. However, this is an unrealistic assumption. Even in our datasets, the original versions demonstrated class bias: Amazon product reviews are generally positive, votes on bills are rarely tied, and political parties vote in blocs. While it is common to evaluate learning methods on balanced data, and then adjust for imbalanced real world datasets, it is unclear what effect *domain-specific* class bias will have on MDL methods. Domains can differ in their proportion of examples of different classes. For example, it is quite likely that less controversial bills in the United States Congress will have more yes votes than controversial bills. Similarly, if instead of the category of a product, its brand is considered as a domain, it is likely that some brands receive a higher proportion of positive reviews than others.

Improvements from MDL in such settings may simply be capturing domain-specific class biases.

[4]Some numbers in Table 1 might appear to be significant, but are not. That is because of high variance in the performance of the methods across the different folds.

[5]The BILL and the PARTY datasets are not directly comparable to each other, although the prediction task is the same.

domain	class	cb1	cb2	cb3	cb4
AMAZON					
b	−	20	80	60	40
	+	80	20	40	60
d	−	40	20	80	60
	+	60	80	20	40
e	−	60	40	20	80
	+	40	60	80	20
k	−	80	60	40	20
	+	20	40	60	80
BILL					
031	N	16	4	8	12
	Y	4	16	12	8
088	N	12	16	4	8
	Y	8	4	16	12
132	N	8	12	16	4
	Y	12	8	4	16
PARTY					
D	N	10	30	15	25
	Y	30	10	25	15
R	N	30	10	25	15
	Y	10	30	15	25

Table 2: The table shows the distribution of instances across domains and class labels *within one fold* of each of the datasets, for four different class bias trials. These datasets with varying class bias across domains were used for the experiments described in §4.2

Consider two domains, where each domain is biased towards the opposite label. In this case, domain-specific parameters may simply be capturing the bias towards the class label, increasing the weight uniformly of features predictive of the dominant class. Similarly, methods that learn domain similarity may be learning class bias similarity.

Why does the effectiveness of these domain-specific bias parameters matter? First, if capturing domain-specific class bias is the source of improvement, there are much simpler methods for learning that can be just as effective. This would be especially important in settings where we have many domains, and learning domain-specific parameters for each feature becomes infeasible. Second, if class bias accounted for most of the improvement in learning, it suggests that such settings could be amenable to unsupervised adaptation of the bias parameters.

Hypothesis: *MDL largely capitalizes on domain-specific class bias.*

Empirical Evaluation To evaluate our hypothesis, for each of our three datasets we create 4 random versions, each with some domain-specific class-bias. A summary of the dataset partitions is shown in Table 2. For example, for the AMAZON dataset, we create 4 versions (cb1 ... cb4), where each domain has 100 examples *per fold* and each domain has a different balance between positive and negative classes. For each of these settings, we conduct a 10-fold cross validation experiment, then average the CV results for each of the 4 settings. The resulting accuracy numbers therefore reflect an average across many types of bias, each evaluated many times. We do a similar experiment for the BILL and PARTY datasets, except we use 5-fold CV.

In addition to the multi-domain and baseline methods, we add a new baseline: DOM−ID. In this setting, we augment the baseline classifier (which ignores domain labels) with a new feature that indicates the domain label. While we already include a general bias feature, as is common in classification tasks, these new features will capture domain-specific bias. This is the only change to the baseline classifier, so improvements over the baseline are indicative of the change in domain-bias that can be captured using these simple features.

Results Results are shown in Table 3. The table follows the same structure as Table 1, with the addition of the results for the DOM−ID approach. We first examine the efficacy of MDL in this setting. An observation that is hard to miss is that MDL results in these experiments show significant improvements in almost all cases, as compared to only a few cases in Table 1, despite the fact that even the baseline approaches have a higher accuracy. This shows that MDL results can be highly influenced by systematic differences in class bias across domains. Note that there is also a significant negative influence of class bias on MTRL for the AMAZON data.

A comparison of the MDL results on true domains to the DOM−ID baseline gives us an idea of how much MDL benefits purely from class bias differences across domains. We see that in most cases, about half of the improvement seen in MDL is accounted for by a simple baseline of using the domain identifier as a feature, and all but one of the improvements from DOM−ID are significant. This

	AMAZON		BILL		PARTY	
	SVM	**LR**	**SVM**	**LR**	**SVM**	**LR**
	Single Classifier					
	85.52%	85.46%	70.50%	70.67%	65.44%	65.81%
	FEDA					
True Domain	+0.11	+0.31	+4.25 ▲	+4.00 ▲	+4.81 ▲	+4.69 ▲
Random Domain	+0.94 ▲	+1.03 ▲	+3.68 ▲	+4.03 ▲	+4.24	+3.73
	MDR-L2					
True Domain	+0.92 ▲	+0.98 ▲	+4.42 ▲	+4.25 ▲	+1.31	+0.94
Random Domain	+1.86 ▲	+1.92 ▲	+3.93 ▲	+3.77 ▲	+0.65	+0.28
	MDR-KL					
True Domain	+1.54 ▲	+1.59 ▲	+5.17 ▲	+5.00 ▲	+4.25 ▲	+3.88 ▲
Random Domain	+2.84 ▲	+2.90 ▲	+4.13 ▲	+3.97 ▲	+3.81 ▲	+3.44
	MTRL					
True Domain	-1.22 ▼	-1.17 ▼	+4.50 ▲	+4.33 ▲	+6.44 ▲	+6.06 ▲
Random Domain	-0.69 ▼	-0.63 ▼	+3.53 ▲	+3.37 ▲	+4.87 ▲	+4.50 ▲
	DOM-ID					
True Domain	+0.36	+0.38 ▲	+2.83 ▲	+2.75 ▲	+3.75 ▲	+4.00 ▲
Random Domain	+1.73 ▲	+1.76 ▲	+4.50 ▲	+4.98 ▲	+5.24 ▲	+5.31 ▲

Table 3: A comparison between MDL methods with class biased data. Similar to the setup where we evaluate the ensemble learning effect, we have a setting of using randomized domains. ▲: Significantly better than the corresponding SVM or LR baseline, with $p < 0.05$, using a paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a paired t-test.

suggests that in a real-world scenario where difference in class bias across domains is quite likely, it is useful to consider DOM-ID as a simple baseline that gives good empirical performance. To our knowledge, using this approach as a baseline is not standard practice in MDL literature.

Finally, we also include the "Random Domain" evaluation in the our class biased version of experiments. Each "Random Domain" result in Table 3 is an average over 20 cross-validation runs (5 randomized trials for each of the four class biased trials cb1 ... cb4). This setup combines the effects of ensemble learning and bias difference across domains. As seen in the table, for MDL algorithms the results are consistently better as compared to knowing the true domains for the AMAZON dataset. For the other datasets, the performance after randomizing the domains is still significantly better than the baseline. This evaluation on randomized domains further strengthens the conclusion that differences in bias across domains play an important role, even in the case of noisy domains. Looking at the performance of DOM-ID with randomized domains, we see that in all cases the DOM-ID baseline performs *better* with randomized domains. While the difference is significant mostly only on the AMAZON dataset (details in Table 6, columns under "**Varying Class Bias**,") this trend is still counter-intuitive. We suspect this might be because randomization creates a noisy version of the domain labels, which helps learners to avoid over-fitting that single feature.

domain	**class**	**cb5**	**cb6**	**cb7**	**cb8**
	AMAZON				
b	−	20	40	60	80
	+	80	60	40	20
d	−	20	40	60	80
	+	80	60	40	20
e	−	20	40	60	80
	+	80	60	40	20
k	−	20	40	60	80
	+	20	40	60	80

Table 4: The table shows the distribution of instances across domains and class labels *within one fold* of the AMAZON dataset, for four different class bias trials. For the BILL and PARTY datasets, similar folds with consistent bias were created (number of examples used was different). These datasets with *consistent class bias* across domains were used for the experiments described in §4.2.1

4.2.1 Consistent Class Bias

We also performed a set of experiments that apply MDL algorithms to a setting where the datasets have different class biases (unlike the experiments reported in Table 1, where the classes are balanced), but, unlike the experiments reported in Table 3, the class bias is the *same* within each of the domains. We refer to this as the case of *consistent class bias* across domains. The distribution of classes within each domain within each fold is shown in Table 4. The results for this set of experiments are reported in Table 5. The structure of Table 5 is identical to that of Table 1. Comparing these results to those in Table 1, we can see that in most cases the improvements seen using MDL algorithms are lower than those seen in Table 1. This is likely due to the higher baseline performance in the *consistent class bias* case. A notable difference is in the performance of MTRL — it is significantly worse for the AMAZON dataset, and significantly better for the PARTY dataset. For the AMAZON dataset, we believe that the domain distinctions are less meaningful, and hence forcing MTRL to learn the relationships results in lower performance. For the PARTY dataset, in the case of a class-biased setup, knowing the party is highly predictive of the vote (in the original CONVOTE dataset, Democrats mostly vote "no" and Republicans mostly vote "yes"), and this is rightly exploited by MTRL.

4.2.2 True vs. Randomized Domains

In Table 6 we analyze the difference in performance of MDL methods when using true vs. randomized domain information. For the three sets of results reported earlier, we evaluated whether using true domains as compared to randomized domains gives significantly **better**, significantly **worse** or **equal** performance. Significance testing was done using a paired t-test with $\alpha = 0.05$ as before. As the table shows, for the first set of results where the class labels were balanced (overall, as well as within each domain), using true domains was significantly better mostly only for the AMAZON dataset. FEDA−SVM was the only approach that was consistently better with true domains across all datasets. Note, however, that it was significantly better than the baseline approach only for PARTY.

For the second set of results (Table 3) where the class bias varied across the different domains, using true domains was either no different from using randomized domains, or it was significantly worse. In particular, it was consistently significantly worse to use true domains on the AMAZON dataset. This questions the utility of domains on the AMAZON dataset in the context of MDL in a domain-specific class bias scenario. Since randomizing the domains works better for all of the MDL methods on AMAZON, it suggests that an ensemble learning effect is primarily responsible for the significant improvements seen on the AMAZON data, when evaluated in a domain-specific class bias setting.

Finally, for the case of consistent class bias across domains, the trend is similar to the case of no class bias — using true domains is useful. This table further supports the conclusion that domain-specific class bias highly influences multi-domain learning.

5 Discussion and Open Questions

Our analysis of MDL algorithms revealed new trends that suggest further avenues of exploration. We suggest three open questions in response.

Question: *When are MDL methods most effective?*

Our empirical results suggest that MDL can be more effective in settings with domain-specific class biases. However, we also saw differences in improvements for each method, and for different domains. Differences emerge between the AMAZON and CONVOTE datasets in terms of the ensemble learning hypothesis. While there has been some theoretical analyses on the topic of MDL (Ben-David et al., 2007; Ben-David et al., 2009; Mansour et al., 2009; Daumé III et al., 2010a), our results suggest performing new analyses that relate ensemble learning results with the MDL setting. These analyses could provide insights into new algorithms that can take advantage of the specific properties of each multi-domain setting.

Question: *What makes a good domain for MDL?*

To the best of our knowledge, previous work has assumed that domain identities are provided to the learning algorithm. However, in reality, there may be many ways to split a dataset into domains. For example, consider the CONVOTE dataset, which we split both by BILL and PARTY. The choice of splits

	AMAZON		BILL		PARTY	
	SVM	**LR**	**SVM**	**LR**	**SVM**	**LR**
	Single Classifier					
	86.06%	86.22%	76.42%	75.58%	69.31%	68.38%
	FEDA					
True Domain	-0.25	-0.33	-0.83	+0.25	+0.88	+1.25
Random Domain	-1.17 ▼	-1.26 ▼	-1.33	-0.82	-0.55	-0.04
	MDR-L2					
True Domain	+0.39 ▲	+0.23	-0.42	+0.42	-2.12	-1.19
Random Domain	-0.38	-0.53 ▼	-3.57	-2.73	-4.30 ▼	-3.36 ▼
	MDR-KL					
True Domain	+0.81 ▲	+0.65 ▲	-0.83	+0.00	+1.31	+2.25 ▲
Random Domain	+0.22	+0.06	-1.90	-1.07	-0.60	+0.34
	MTRL					
True Domain	-1.52 ▼	-1.68 ▼	-1.92	-1.08	+3.12 ▲	+4.06 ▲
Random Domain	-2.12 ▼	-2.28 ▼	-0.95	-0.12	+0.19	+1.12 ▲

Table 5: A comparison between MDL methods with data that have a *consistent class bias* across domains. Similar to the setup where we evaluate the ensemble learning effect, we have a setting of using randomized domains. ▲: Significantly better than the corresponding SVM or LR baseline, with $p < 0.05$, using a paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a paired t-test.

MDL Method	**No Class Bias (Tab. 1)**			**Varying Class Bias (Tab. 3)**			**Consistent Class Bias (Tab. 5)**		
	better	**worse**	**equal**	**better**	**worse**	**equal**	**better**	**worse**	**equal**
FEDA-SVM	AM, BI, PA				AM	BI, PA	AM, PA		BI
FEDA-LR	AM		BI, PA		AM	BI, PA	AM, BI		PA
MDR-L2	AM		BI, PA		AM	BI, PA	AM, BI	PA	
MDR-KL	PA		AM, BI		AM	BI, PA	AM, PA		BI
MTRL	AM		BI, PA		AM	BI, PA	AM, PA	BI	
DOM-ID-SVM	–	–	–		AM	BI, PA	–	–	–
DOM-ID-LR	–	–	–		AM, BI	PA	–	–	–

Table 6: The table shows the datasets (AM:AMAZON, BI:BILL, PA:PARTY) for which a given MDL method using true domain information was significantly **better**, significantly **worse**, or not significantly different (**equal**) as compared to using randomized domain information with the same MDL method.

impacted MDL. This poses new questions: what makes a good domain? How should we choose to divide data along possible metadata properties? If we can gain improvements simply by randomly creating new domains ("Random Domain" setting in our experiments) then there may be better ways to take advantage of the provided metadata for MDL.

Question: *Can we learn class-bias for unsupervised domain adaptation?*

Experiments with domain-specific class biases revealed that a significant part of the improvements could be achieved by adding domain-specific bias features. Limiting the multi-domain improvements to a small set of parameters raises an interesting question: can these parameters be adapted to a new domain without labeled data? Traditionally, domain adaptation without target domain labeled data has focused on learning the behavior of new features; beliefs about existing feature behaviors could not be corrected without new training data. However, by collapsing the adaptation into a single bias parameter, we may be able to learn how to adjust this parameter in a fully unsupervised way. This would open the door to improvements in this challenging setting for real world problems where class bias was a significant factor.

Acknowledgments

Research presented here is supported by the Office of Naval Research grant number N000141110221.

References

Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2008. Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition. In *Proceedings of ACL-08: HLT*, pages 245–253.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Proceedings of NIPS 2006*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2009. A theory of learning from different domains. *Machine Learning*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. 2008. Learning Bounds for Domain Adaptation. In *Advances in Neural Information Processing Systems (NIPS 2007)*.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. 2008. Linear Algorithms for Online Multitask Classification. In *Proceedings of COLT*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of Maximum Entropy Capitalizer: Little Data Can Help a Lot. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 285–292.

Koby Crammer, Mark Dredze, and Fernando Pereira. 2008. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010a. A Co-regularization Based Semi-supervised Domain Adaptation. In *Neural Information Processing Systems*.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010b. Frustratingly Easy Semi-Supervised Domain Adaptation. In *Proceedings of the ACL 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Hal Daumé III. 2009. Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. *Proceedings of the 25th international conference on Machine learning - ICML '08*.

Mark Dredze, Alex Kulesza, and Koby Crammer. 2009. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2).

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR : A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874.

Jenny R. Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian Domain Adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610.

Richard Maclin and David Opitz. 1999. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain Adaptation with Multiple Sources. In *Proceedings of NIPS 2008*, pages 1041–1048.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Conference on Natural Language Learning (Shared Task)*.

Avishek Saha, Piyush Rai, Hal Daumé III, and Suresh Venkatasubramanian. 2011. Online learning of multiple tasks and their relationships. In *Proceedings of AISTATS 2011*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335.

Yu Zhang and Dit-Yan Yeung. 2010. A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In *Proceedings of the Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*.