Multi–Domain Learning: Analysis, and Methods for Multi–Attribute Domains

Mahesh Joshi

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Carolyn Penstein Rosé, Co-Chair
William W. Cohen, Co-Chair
Noah A. Smith
Mark Dredze, Johns Hopkins University

An offering to dearest Mother Sai

Abstract

A common assumption in many machine learning techniques is that the data points are independent and identically distributed (*i.i.d.*). However, often data can be divided into subgroups of data points that are related in some way, and this violates the assumption that they are identically distributed. Such subgroups are commonly referred to as domains or subpopulations. Domain information can be used to learn better machine learning models, and multi–domain learning techniques provide one way of using domain information in data. An important question when using multi–domain learning techniques is that of defining how a given dataset is divided into domains. Often some metadata attribute associated with the instances is used for defining domains. In this thesis, we consider the impact of the definition of domains on multi–domain learning, and propose approaches that can handle the case where domains can be defined for a given dataset in more than one way.

We first present an empirical analysis of existing multi–domain learning methods, with the aim of understanding how the definition and properties of domains influence their performance. We show that the performance of multi–domain learning techniques can be affected by two factors: (i) an ensemble learning effect due to classifier combination; and (ii) the distribution of class labels across the different domains.

We then show that it is possible to design a problem–driven approach to multi–domain learning. We propose a feature representation that is motivated by knowledge about the domains available in the data. Our feature representation explicitly accounts for the structural similarity among syntactic features across multiple domains, even when the domains can be defined in more than one way.

Finally, we present learning methods that go beyond the current multi–domain learning paradigm, which assumes a single way of dividing the data into domains. For many text classification tasks, multiple metadata attributes associated with the text can influence the behavior of textual features as well as the performance on the task. The different metadata attributes can have varying utility for the purpose of defining domains for multi–domain learning. Choosing a single metadata attribute to define domains in such cases may not be optimal. We propose methods that allow the use of multiple metadata attributes for defining domains, leading to better models that are still efficient.

Acknowledgments

The pursuit of a Ph.D. is a challenging[1] journey. However, it is made enjoyable by the many souls that touch upon the life of an aspiring Ph.D. in numerous ways. This is my heartfelt attempt to express gratitude to all those wonderful folks who have guided and supported me in my journey.

I am very grateful to my thesis advisors: Carolyn and William.

Carolyn has been my advisor since my Master's days at Carnegie Mellon, and has patiently guided me throughout the years, despite my at–times–wild research ideas. She gave me the freedom to pursue my research interests, while also bringing me back on track when the focus of my work seemed diluted. She also brought to my work the much–needed linguistic perspective that is valuable in thinking about any problem in natural language processing. She has taught me to think deeply about research problems, and to always meaningfully question and challenge one's own work.

William has been the wise sage on my committee — with his calm demeanor, and very thoughtful advise on all matters, technical or non–technical. I am very thankful to him that he agreed to be my co–advisor starting Fall of 2010, when I needed a core machine learning perspective for my work going forward. He has taught me that seemingly small things can matter in research, and therefore no thing is too small to pay attention to when doing research.

My thesis committee members, Noah A. Smith and, in particular, Mark Dredze, have played a very significant role in shaping my dissertation work.

Noah has always amazed me with his quickness in grasping the core of whatever I described to him (and my descriptions got pretty verbose at times), despite long periods of time between our meetings. His energy is contagious, and I have always come away from his office feeling enthusiastic about my work. I have learned a lot from my interactions with him, including my collaboration with him on work in text–driven forecasting.

Mark is without a doubt *the* best "external" thesis committee member that one could hope for, and a lot more than that. I doubt if anyone else in that role would even imagine being on a phone call at 1 a.m. before a paper deadline, discussing paper edits, and the best way to present results in order to make a point. He has been almost like a third advisor to me starting from the time I did my thesis proposal. He has always asked me the most penetrating questions, and guided me patiently in finding out the answers.

Prior to coming to Carnegie Mellon, my thesis advisors at the University of Minnesota Duluth, Richard Maclin and Ted Pedersen were instrumental in igniting my interest in machine learning and natural language processing. I am fortunate that I got to work with them at the time, and it paved my way to Carnegie Mellon.

Many amazing teachers have influenced my academic pursuit starting from my school years, and I am thankful to all of them. I would like to particularly mention Mrs. Mitali Chaudhury, Mrs. Mokashi, Mrs. Nazare, Mr. Rajput, and Ms. Rose from my school years; Mr. Gadgil, and Mr. Jadhav from my high school years; and Mr. Kajave, and Mr.

[1] Some have said it is harder than having a baby. However, since I cannot ever experience (thankfully!!) the physical ordeal involved in having a baby, I will skip that comparison.

Sawant from my college years. Each one of them has left an impression on me for the better. At the University of Minnesota Duluth, probability and statistics wouldn't have been as much fun without Prof. Barry James and Prof. Ronald Regal. At Carnegie Mellon, Prof. Matthew Harrison (now at Brown University) and Prof. Tom Mitchell made the Intermediate Statistics and Machine Learning classes always something to look forward to, despite whatever else I had on my mind.

My thanks go to the entire current staff and ex–staff at the Language Technologies Institute (LTI) at Carnegie Mellon for helping me navigate all the administrative requirements. Special thanks to Linda Hager, Dana Houston, Radha Rao, and most of all Stacey Young! Thanks also to all the folks at SCS Help Desk, including Kirk Berthold who I have most interacted with for disk backups — you all keep the computing experience at SCS enviable!

At LTI, and outside, I have been extremely lucky to be working with very smart colleagues, collaborating with them, and learning from them in several ways. Collaborators include Shilpa Arora, Sourish Chaudhury, Dipanjan Das, Kevin Gimpel, Rohit Kumar, and Yi-Chia Wang at LTI. I have also benefited from discussion about my work with Sivaraman Balakrishnan and Kriti Puniyani. My internship experience at the Mayo Clinic Bioinformatics department, supervised by Serguei Pakhomov, was very valuable in shaping and contributing toward my Master's thesis at University of Minnesota Duluth. The internship at Google Inc., supervised by Gaurav Garg, was a great experience for me in terms of taking research ideas "to the field." Thanks also go to Venki Ganti, who was closely involved in my internship work at Google. All the members from Carolyn's as well as William's research group are to be especially thanked for enduring the various research discussions I had with them, and presentations on which they gave feedback — ranging from discussing half–baked ideas that I had, to practice talks for conferences and such. Over the years the groups have included Rohit, Sourish, Yi-Chia, Jenny, Gahgene, Mahaveer, Manaj, Philip, Nitin, Dong, Naman, Moonyoung, Gregory, Hua, Ranjitha, Guang, Iris, Elijah, Miaomiao, Ryan, Hye-Ju, David, Phani, Zeyu, Abhimanyu, Mario, Vitor, Richard, Ni, Frank, Tae, Bhavana, and Ramnath. Working from office was way more exciting due to discussions about all sorts of random stuff, thanks to my excellent office–mates, present and past: Rohit, Yi-Chia and Alok.

All of my friends, at some point or the other, have helped me walk on this path to Ph.D., making it more fun when it seemed fun, and more importantly, making it surmountable when it seemed not so. These include, Namrata and Satanjeev, Shilpa and Gautam, Barkha and Abhay, Sudipti and Abhimanyu, Abhay B.R., Abhaya, Suresh, Sachin, Sameer, Dipanjan, Usha, and Gaurav. Special thanks go to Shilpa for coordinating all the birthday (and other) gatherings for our group; to Abhay B.R. for awesome music practices, and chat discussions on every topic conceivable; to Barkha and Abhay for sharing parenting joys (and, at times, woes); and to Namrata and Satanjeev for long late–night discussions about technology, philosophy, and everything in between!

The Pittsburgh Sai family has been a home away from home. All the members, over the many years, are too numerous to name here. Each one of them has helped me and my

family in one way or the other, welcoming us into their homes as their own family. Their love and compassion make them ideal role models. Know that you all will be missed the most!

My entire family has always given me unconditional love and support, irrespective of the geographical distance that separated us. Dad, aajoba, and aaji, wherever you are, I know that your love and blessings have always been with me, and that this occasion makes you immensely happy. Mom: I cannot ever imagine the sacrifices you have made to help us reach where we are, your patience is only matched by your unselfish love. Shrikant, and now Priya too: young as you are, you have always been there for our family, especially when I could not be there due to the long distance. Kaka and Kaku, your guidance and advice from early age has helped me be a better person. Apurva, your wiser-than-your-years thinking will always inspire me. Anagha's Mom and Dad: besides your love and support, you have shown me by example how to overcome challenges and keep going, an invaluable lesson for a graduate student! To all my family once again: your love, blessings, and confidence in me has kept me motivated.

My son Sohum is only two years old, and yet somehow seems to magically know the secret to soothing all worries by a mere look into my eyes, and the brightest and most innocent of all smiles! Dear Sohum: you are the best gift we have ever received; we hope one day you'll feel that all these hours we have had to spend away from you were somehow worthwhile in making us better parents.

Most of all, my love and gratitude is expressed toward Anagha. She has been there with me in my journey all along, simultaneously walking the path toward her own Ph.D., and meanwhile also being a fantastic Mom to Sohum. No words will ever express my thankfulness toward her, and for the fact that I have her in my life. Being who she is, she does not expect the words either, she just knows. Anagha: you have an equal (if not more) share in every joy and every accomplishment that we have experienced together. This would have been neither possible, nor worthwhile without you.

Contents

List of Tables

1 | Introduction

Statistical machine learning has become an integral part of the techniques used for natural language processing (NLP). Machine learning methods represent the state of the art in classification, regression, as well as structured prediction problems. One of the fundamental assumptions in many machine learning approaches is that the training as well as test data instances are drawn independently from the same underlying distribution. This is commonly known as the *i.i.d.* (*independent and identically distributed*) assumption. However, the *i.i.d.* assumption is often violated — that is, the data points are either *not independent*, or *not drawn from the exact same underlying distribution*, or both. This thesis addresses the second case: the data points are *not drawn from the same underlying distribution*, and therefore are not "identically distributed."

Several machine learning problems exist where data points are not identically distributed. For example, for the task consumer advertising, one cannot assume that the spending behavior is identically distributed across all consumers — it is correlated to factors such as their income level, gender, and age. For a task such as segmentation of an image into background and foreground, one can expect different underlying distributions for indoor versus outdoor images, or images of people versus images of objects. For weather forecasting, while it might be useful to consider weather patterns in nearby geographical areas, one can clearly not assume that they are identically distributed as the local weather. In each of the above scenarios, ignoring the fact that the data points are not identically distributed (and in fact, can be meaningfully grouped into subsets that emerge naturally based on some metadata attribute) can adversely affect the models learned by machine learning methods. This can lead to inaccurate predictions such as advertising of irrelevant products, incorrect image segmentation, and inaccurate weather forecasts.

Tasks in natural language processing are no exception when it comes to violating the "identically distributed" assumption. For example, consider the task of learning a classifier to predict if a consumer review for some product on the Amazon.com website has a

positive or negative polarity.[1] We will call this the polarity prediction task. On the face of it, it might seem reasonable to model a dataset of Amazon.com reviews assuming that the reviews are identically distributed, that is, the features appearing in all reviews share a common background distribution over some fixed vocabulary. However, a little more thinking reveals several aspects that can affect the feature distribution in a review. The category of a product (whether it is a book, or a movie, or a kitchen appliance) will naturally influence the vocabulary used in the reviews of the corresponding products (Blitzer et al., 2007). Even within a category such as "movies," the genre of the movie might affect the language in the reviews. The price range of a product might influence customer expectations and therefore their language in the reviews. Thus, assuming that reviews are identically distributed is an oversimplification when designing a machine learning approach to the problem of polarity prediction. A model that ignores this aspect is in effect ignoring structure in data that can be used to make better predictions. For example, the word "engaging" is much more likely to be an indicator of positive polarity in book reviews or movie reviews, than in kitchen appliance reviews, where it may not be indicative of any sentiment. This should help a model make more accurate predictions when classifying reviews. It is only possible if the learning method, or the feature representation that is used, accounts for such structure in data in some way.

Recognizing this fact, research in recent years has focused attention on going beyond the assumption of identically distributed data points. While a detailed description of related threads of research is in Chapter 2, we will now briefly introduce the multi–domain learning framework, which is one way to exploit structure in data in the form of data points that are not identically distributed, but instead belong to different subgroups.

1.1 Multi–Domain Learning

We will formally set up the multi–domain learning problem in Chapter 3. At this time, we only provide a high–level overview of the problem to motivate the goals and contributions of this thesis.

Multi–domain learning (Dredze and Crammer, 2008, Dredze et al., 2009) aims to exploit group structure in data. While there are a few variations in the different problem scenarios

[1] This is a practically useful task for multiple reasons. There are often plain–text opinions expressed on the internet about different products. Polarity classification models can be useful for such unstructured data which is not associated with a star–rating. Even for Amazon.com, assigning polarity to video–based reviews that are not tagged with a star–rating is an important task.

considered originally by Dredze and Crammer (2008), the key factor in all of them is that the training data can be divided into subsets or groups of data points, which are variously referred in the literature as *domains*, *subpopulations*, or *tasks*.[2] We will refer to them as *domains*. Domains are usually defined based on some metadata attribute associated with each of the instances. For example, on a widely used dataset of Amazon.com product reviews (Blitzer et al., 2007), the *product category* (such as `books`, `dvds`, `electronics`, or `kitchen appliances`) is used to define domains. The key intuition typically used in defining domains for a dataset is that the data points within a domain should be similar to each other in some way, and also different to some extent from the data points in other domains. There is also the assumption that there are some similarities *across domains* that can be exploited toward modeling some task of interest (such as polarity prediction on Amazon.com reviews) in a better way. Given such a dataset that is divided into multiple domains, the goal in multi–domain learning is to learn shared parameters across domains, as well as domain–specific parameters for each of the domains in the training data.[3]

1.2 Defining Domains

As mentioned before, the definition of domains on a given dataset is mostly driven by some intuition about the problem, and how the proposed domain structure might influence the modeling of the task that one is interested in solving for the given dataset. To re–consider the polarity prediction on task for the Amazon.com dataset, the *product category* seems to suggest a reasonable definition of domains for the following reasons. First, when classifying reviews as positive or negative, one can generally expect that across different product types, there are certain words that will consistently indicate one or the other polarity. For example, the word "good" will in most cases suggest a positive polarity, and the word "bad" will in most cases suggest a negative polarity.[4] Second, one expects that the language used in the reviews about products from the different product categories will differ from category to category. For example, when reviewing `books`, the users will

[2]Calling them *tasks* can be potentially confusing, since the original definition of *tasks* in the seminal work on multitask learning by Caruana (1993) was a different one. There he considered *different, but related* tasks on the *same* full set of input instances, without considering subsets of those instances.

[3]Large datasets can potentially contain millions of domains. Therefore one variation of the online multi–domain learning approach presented by Dredze and Crammer (2008) learns cluster–specific parameters, instead of domain–specific ones, by clustering together similar domains in the data.

[4]For the purpose of this discussion we are not considering some of the more nuanced linguistic phenomena such as the use or sarcasm, or irony. Also, while negations too are not mentioned here, they will affect the connection of these words to polarity in ways that will not much differ much across product categories.

tend to talk more about storyline, characters and so on, as opposed to characteristics such as the sturdiness and durability, which are more likely to be discussed in the context of products from the `electronics` or `kitchen appliances` category. Turney (2002) has also provided the more extreme example of the *same* word behaving differently in different domains — "unpredictable" can indicate a positive polarity when used in a movie review (as in, *an unpredictable plot*), but is more likely to be a negative word in the domain of automobile reviews (as in, *unpredictable steering behavior*). And third, the previous point also implies that one expects some similarity in the language used in the reviews for products from a given category such as `books`, or `movies`.

This approach of defining domains using researcher intuition about the problem is a great way of injecting human knowledge into the learning process. However, there are several limitations of this approach as well. First, there can be situations where it is not easy to arrive at intuitions about whether the available domain structure in a dataset matters for the task at hand or not. Related to this issue, one cannot rule out for certain the possibility of the researcher intuition being only partially correct, or just completely wrong. Second, and most importantly, restricting the multi–domain learning setting to allow only a *single way* of defining domains limits its use in scenarios where in fact *several* metadata attributes can be hypothesized to influence the prediction task of interest. We will now motivate the work in this thesis by considering each one of these limitations in detail.

1.3 Motivation for this Thesis

The question of defining domains for the multi–domain learning problem is central to this thesis. We examine several aspects related to the definition of domains for multi–domain learning for text classification tasks.

The previous section talked about the limitations of the approach of defining domains for a dataset based on researcher intuition. We would first like to emphasize that defining domains based on intuition is a perfectly reasonable option in some cases, and more importantly, is always a great way to think about a problem at a deeper level. That having said, the limitations mentioned earlier can pose a problem in many situations.

The first limitation is related to whether one truly understands the influence that a given domain structure has on the prediction task at hand. There might, for example, be cases where the domains can only be defined for a given dataset based on a single

metadata attribute that is available for all instances. Or it might be the case that among a very small number of metadata attributes, it is seemingly clear based on the task what single metadata attribute will have the most dominant influence on the task. In such supposedly "clear–cut" cases, one can always use multi–domain learning techniques with the available definition of domains. However in such cases, the success or failure of the multi–domain learning methods can be hard to explain. In particular, if the multi–domain learning methods succeed, we show in this thesis that there are still confounding factors that can incorrectly lead us to believe that multi–domain learning is working for the right reasons. To end–users of machine learning, this may not seem very important — after all, there was an improvement, and it may rarely matter if it came about as a result of the assumptions that we modeled. However, as researchers, it is crucial to understand if our methods are working for the right reason. More crucially however, if multi–domain learning fails to improve performance in such cases, then whether as an end–user, or as a researcher, one would want to understand if the failure was due to improper definition of domains, or due to the limitations of the multi–domain learning method used for modeling the problem. The evaluation methodology that we propose in this thesis is useful in isolating such potential reasons for success or failure of multi–domain learning methods.

The second limitation is related to a fundamental assumption that the current multi–domain learning paradigm makes — domain structure for a dataset is defined based on a single way of partitioning the data, generally using a single metadata attribute. As mentioned in the previous paragraph, this might be feasible when there is only one, or just a few metadata attributes in a dataset, and the choice seems relatively straightforward. However, in many datasets, more than a single metadata attribute is available for each data point. To continue our example of Amazon.com reviews, it is not just the *product category* that can define domains. One can easily imagine any of the following metadata attributes defining domains that are reasonable as per all the criteria presented in the first paragraph in Section 1.2: *product identifier* (this is more specific than the *product category*), *price range*, and *manufacturer brand*. Although it is possible to formulate intuitions about why each of these can be reasonable for defining domains, it is difficult to weight their relative influence on the task of polarity prediction. Hence, in such cases, choosing a single metadata attribute to define the domains might end up being an arbitrary choice, or perhaps based on convenience of what metadata is easily available. More crucially though, it does not have to be the case that only a single metadata attribute influences the language in reviews. All of the above metadata attributes might influence the reviews in different

ways, and it can be useful to model those similarities and differences *simultaneously*.

This thesis makes significant advances in addressing both of the above limitations. We have conducted a thorough empirical analysis of existing multi–domain learning algorithms in order to understand the impact of domain definitions on their performance. We also consider the problem of employing multi–domain learning systems "in the wild," in particular, the issue of how class imbalance across domains can influence the performance of multi–domain learning approaches. We then proceed to propose a problem–driven approach of designing a feature representation for multi–domain learning, and *ensuring* that the representation indeed capitalizes on the properties of the multi–domain learning scenario. And finally, we propose a new paradigm of multi–domain learning in the presence of multiple metadata attributes that can define domains for a given dataset. We extend existing multi–domain learning techniques to handle this new paradigm, and show results on several datasets where this new paradigm is useful in capturing multiple domain definitions.

We now proceed to present our thesis statement, and thesis contributions.

1.4 Thesis Statement

> *Multiple interacting metadata attributes can define the domains that influence prediction tasks. Multi–domain learning methods should therefore exploit the domain structure induced by multiple attributes simultaneously.*

1.5 Thesis Contributions

We group the contributions that this thesis makes into the following areas:

I. *empirical analysis*

- **a.** A critical empirical evaluation of existing multi–domain learning techniques on multiple datasets

II. *evaluation methodology*

- **a.** A methodology to identify confounding success factors in multi–domain learning
- **b.** A methodology for evaluating whether the benefit from novel feature representations for multi–domain learning is indeed capturing evidence across

multiple domains

III. *new methods*

a. A problem–driven feature representation based approach that capitalizes on the multi–domain learning setting

b. A new "multi–attribute multi–domain learning" paradigm that allows multiple metadata attributes to influence the definition of domains for a dataset

c. Extensions of multi–domain learning methods that can operate in the new multi–attribute multi–domain learning paradigm

1.6 Layout of the Thesis

The remainder of this thesis is laid out as follows. In Chapter 2, we will provide a thorough review of several related areas of research that focus on using domain structure in data. This will also help us in fixing some terminology for the rest of the thesis.

That will be followed by a systematic empirical evaluation of a representative set of multi–domain learning approaches in Chapter 3. We will explore two confounding factors in the evaluation of multi–domain learning — the ensemble learning effect, and the effect of domain–specific class bias. This chapter will thus describe the contributions listed under **I.a.** and **II.a.** in the previous section.

Subsequently, in Chapter 4, we will present a problem–centric feature representation oriented approach to multi–domain learning, which can potentially complement the algorithmic solutions to the multi–domain learning problem. We will also present a deeper analysis of our feature representation showing that it indeed capitalizes on the presence of multiple domains. This chapter will thus describe the contributions listed under **III.a.** and **II.b.** in the previous section.

Chapter 5 will then introduce the multi–attribute multi–domain learning paradigm, and present our extensions to existing multi–domain learning techniques to enable them to operate in this new paradigm. This chapter will thus describe the contributions listed under **III.b.** and **III.c.** in the previous section.

Finally, Chapter 6 will summarize the work presented in the thesis, discuss the limitations of the current work, and in that light, outline several possible directions for interesting future work that can address those limitations.

2 | Related Work

In the previous chapter we provided an introduction to the idea of exploiting *non-i.i.d.* structure in data. In particular, for the purposes of this thesis, we are interested in exploiting *domain structure*. We saw that domain structure is typically induced in the data as a set of partitions or subgroups that one can create based on the value of some metadata attribute for each data point. For example, we saw that a dataset consisting of consumer reviews from the website of Amazon.com, can be partitioned based on the *product category* of the product for which a review has been written. These partitions are referred to as domains throughout this thesis. Finally, we also described at an abstract level, the framework of multi–domain learning, which primarily assumes the presence of *multiple domains* in the training data, and leverages domain structure for learning better models.

In this chapter, we take a broader look at related research that covers modeling of domain structure in several different scenarios. The related literature includes the areas of *domain adaptation* (Chelba and Acero, 2004, Daumé III and Marcu, 2006, Blitzer et al., 2006), *multitask learning* (Caruana, 1997), and different scenarios in *multi-domain learning* (Dredze and Crammer, 2008), all of which can be broadly grouped under the category of *transfer learning*, where the goal is to be able to transfer knowledge between two or more learning tasks. The exact nature of the difference in the learning tasks (whether it is the difference between the input feature distributions, or the output labels for the tasks, or both) determines which specific area a technique belongs to. A complementary line of related work, specially for the case of domain adaptation, considers the task of finding an appropriate feature representation for domain adaptation. We review work from each of these areas, and describe the similarities and differences from the work that we present in this thesis.

2.1 Probabilistic View

In order to gain an understanding of the various approaches within transfer learning, let us consider a general probabilistic model for any supervised classification problem. Given labeled data $\langle \mathbf{X}, \mathbf{y} \rangle$, we are typically interested in modeling either the joint probability $P(\mathbf{X}, \mathbf{y})$ (in generative models), or the conditional probability $P(\mathbf{y}|\mathbf{X})$ (in discriminative models). The joint probability can be decomposed as $P(\mathbf{X}, \mathbf{y}) = P(\mathbf{y})P(\mathbf{X}|\mathbf{y})$. Based on this decomposition, assuming that the data points are non-*i.i.d.*, and in particular, assuming they have some form of subgroup or domain structure, two kinds of differences can be present between such groups. To be concrete, let us consider a scenario where we have just two[1] subgroups in our data $\langle \mathbf{X}, \mathbf{y} \rangle$: $\mathcal{A}$ and $\mathcal{B}$. Then the following two differences are important from a learning perspective:

DIFF I. The distribution of the input features for each of the groups might be different, that is, $P(\mathbf{X}_{\mathcal{A}}) \neq P(\mathbf{X}_{\mathcal{B}})$. This is also known as *covariate shift* (Shimodaira, 2000). This difference naturally affects generative models because of its direct influence on the $P(\mathbf{X}|\mathbf{y})$ term. However, Shimodaira (2000) showed that this difference also affects discriminative models, when the model is misspecified. A model is misspecified if the optimal model learned from the available input data does not match the true relationship between the input features $\mathbf{X}$ and the predictions $\mathbf{y}$ for all possible inputs $\mathbf{X}$ and predictions $\mathbf{y}$. Intuitively, for discriminative models, even though we only model $P(\mathbf{y}|\mathbf{X})$, since in practice it is learned from a finite sample of $\mathbf{X}$, the best model chosen over that finite set may not be optimal for a dataset with a different distribution over the features.

DIFF II. The set of possible classification labels for each of the groups might be different, that is $\mathcal{Y}_{\mathcal{A}}$ may not be the same as $\mathcal{Y}_{\mathcal{B}}$. This clearly leads to completely different learning tasks for either generative or discriminative models, since the output labels to be predicted are changing. However, assuming the input features are the same, there might still be some transfer of knowledge that can happen between the two tasks, depending on the semantic relationship between them.

We will now explain the different research areas within transfer learning by connecting them to one or both of the above differences.

[1] In general, there can be several subgroups in the data.

2.2 Domain Adaptation

The simplest setting for domain adaptation is where the training set forms a subgroup, and the test set forms another subgroup. The training set is typically called the *source* domain, and the test set, the *target* domain. Let us denote them by S and T respectively. In domain adaptation, **DIFF I** (Section §2.1: the difference in the distribution of features for S and T) is the focus of research. In particular, given a source domain S and a target domain T, $p(\mathbf{X}_S)$ and $p(\mathbf{X}_T)$ are assumed to be different. However, $p(\mathbf{y}|\mathbf{X}_S)$ and $p(\mathbf{y}|\mathbf{X}_T)$ are expected to be the same. Implicitly, the set of possible values for y are assumed to be the same over the source and the target domain.

As an example of a domain adaptation problem, consider the following scenario. We might want to apply a polarity prediction model[2] learned on book reviews (the *source* domain) to a set of reviews for kitchen appliances (the *target* domain). Thus the set of output class values {`positive`, `negative`} is identical across the source and target domains, but the distribution of the input data (features derived from book reviews and kitchen appliance reviews) will differ. Note in this case that the assumption that the conditional distributions $p(\mathbf{y}|\mathbf{X}_{\texttt{books}})$ and $p(\mathbf{y}|\mathbf{X}_{\texttt{kitchen\_appliances}})$ are the same is not strictly true. For instance, reusing the example from Turney (2002) in the current context, the word "unpredictable" can refer to a positive sentiment for some book's plot, but a negative sentiment for the operational behavior of a kitchen appliance. However, in general, it is still reasonable to assume that for polarity prediction, most words will have the same orientation for both of the domains. What differs critically in the feature distribution then is *new words* appearing in the target domain, which were not seen in the source domain at training time.

In prior work related to domain adaptation, two distinct task formulations have been used. The first formulation is where the training set is divided into multiple domains. The assumption is that one has access to labeled data from several different domains (usually related in some way), and the goal is to improve the performance for *some specific target domain* of interest, using the labeled instances from that target domain, as well as from the other source domains. This is usually referred to as *supervised domain adaptation*, because at least some amount of labeled data from the target domain is assumed to be available. Examples of work using this task formulation include Chelba and Acero (2004), Daumé III and Marcu (2006), Daumé III (2007), Arnold et al. (2008), Dai et al. (2009), and Finkel and

[2]Recall from Chapter 1 that this is a model which predicts whether a product review is positive or negative.

Manning (2009). We note that in most of the papers cited in the previous sentence, the amount of labeled target data assumed to be available is fairly small compared to the entire set of source data from one or more source domains. We also note that this setting (especially the training step) is very similar to one of the multi–domain learning formulations proposed by Dredze and Crammer (2008). The key difference is that in supervised domain adaptation, a *single* target domain of interest is assumed or considered in the test data. In multi–domain learning, the test data too can contain several domains.

In the second formulation of domain adaptation, the assumption is that one has access to a relatively large number of labeled data instances from some source domain that is related to the target domain of interest. However, there are no labeled data instances available from the target domain. Instead, a relatively large set of *unlabeled* data from the target domain is available. This is usually referred to as *semi–supervised domain adaptation*, since labeled data is available only from the source domain, but unlabeled data is available from the target domain. Examples of work using this task formulation include Blitzer et al. (2006, 2007), Dai et al. (2007), Jiang and Zhai (2007), and Ling et al. (2008). We note here, however, that some of the work on semi–supervised domain adaptation (such as Blitzer et al. (2007)) shows the benefit of extending their approach using a very small amount of labeled data from the target domain.

In this review we will focus on providing details of the supervised domain adaptation approaches, as they are more closely related to our work. However, some of the semi–supervised domain adaptation techniques that make use of unlabeled data (such as the one by Blitzer et al. (2007)) are complementary to our approach, and therefore can be effectively utilized in combination with our proposed techniques.

Chelba and Acero (2004) focus on transferring knowledge from the source domain to the target domain in the form of the weights $\mathbf{w}^S$ learned for the features on the source domain. They focus on the text capitalization task in a maximum entropy framework, where regularization for feature weights is usually achieved by putting a zero-mean uniform-variance Gaussian prior $\mathcal{N}(\mathbf{0}, diag(\boldsymbol{\sigma}^2))$ on the feature weights (Chen and Rosenfeld, 1999). Instead, Chelba and Acero (2004) propose using $\mathcal{N}(\mathbf{w}^S, diag(\boldsymbol{\sigma}^2))$ as the regularization prior, thus using the weights of features learned on the source domain as the means for the prior used while learning the model on the target domain labeled data. The intuition is to bias the weights for the target domain towards prior knowledge about those features based on the labeled data from the source domain. This way, if there is not enough evidence to update weights of some features based on the target domain data alone, then the

knowledge about those features from the source domain can be useful, with the assumption that the source and target domains share some common features that are interpreted similarly in both domains. This approach of regularizing the model learned on target domain training data using weights of features from the source domain data cannot be straightforwardly extended to our problem setting of multi–domain learning where we have training data from several "source domains." Also, our test data does not comprise of a single target domain of interest, but consists of instances from all of the "source domains."

Daumé III and Marcu (2006) focus on the selection of the right instances for transferring knowledge across domains. The basic premise is that any data instance belongs to either a *truly in-domain* distribution, a *truly out-of-domain* distribution or a *general* distribution. The distribution that an instance comes from is unknown and is modeled as a hidden variable. The source domain data is considered a mixture of the truly out-of-domain and the general distributions, and the target domain data is similarly considered a mixture of the truly in-domain and the general distributions. In the generative story of the proposed model, each feature of any data instance is generated independently of the other features. However, the distribution from which the features of an instance are generated is based on which distribution the data instance comes from (truly in-domain, or truly out-of-domain, or general). Given this model, the idea is to run a conditional expectation-maximization (CEM) procedure to iteratively estimate the parameters of the model. One limitation of this approach is that it is about 10 to 15 times slower (Finkel and Manning, 2009) than standard supervised learning approaches, although there are significant gains in performance. Another limitation is that an instance can only belong to one of the distributions, and so do all of its features. So it is not possible to have an instance generated as a set of features where some features come from a domain–specific distribution and others from a domain–general distribution. Finally, while it is possible to extend this approach to the case of multiple domains in training data, the associated computational cost will increase linearly in the number of domains, and can thus be prohibitively high for a large number of domains.

Daumé III (2007) looks at the domain adaptation problem from the perspective of finding the right set of shared features that can help in transferring knowledge across the domains. The approach is designed to overcome a drawback of the previous approach by Daumé III and Marcu (2006) – the fact that an instance cannot be a mixture of domain-specific and domain-general features. The idea proposed is to transform the feature rep-

resentation of each (sparse) instance such that a copy of its original features is created in a domain–specific feature subspace corresponding to the domain that the instance belongs to. The original features represent the *domain–agnostic* part of the feature space, or the *general* domain. This approach can be essentially implemented as a very simple preprocessing step on the data, and shows significant improvements across several domain adaptation tasks. It is also applicable in a setting where there are multiple "source" domains, and in fact, we use it as a baseline approach wherever relevant in the experiments presented in this thesis.

Arnold et al. (2008) also focus on shared features as the means to transfer knowledge across the different domains. The key idea is to utilize the hierarchical nature of features in certain natural language processing tasks such as named entity recognition. They construct a hierarchical prior on the feature weights in a discriminative learning framework. A common feature hierarchy across the different domains determines the influence that more general features have on the more specific features for each domain. The feature hierarchy is based on the way features are generated and the semantics associated with the feature generation process. For example, if the most specific version of a feature for the task of named entity recognition is f_j = "the first word to the left of the token to be classified is *Professor*" then a more general class of such features, which will become the parent of f_j (and all other such specific instantiations) in the shared feature hierarchy would be "the first word to the left of the token to be classified." The idea is that while the word *Professor* might be an important clue for identifying a named entity in one domain, some other word might be a better clue for other domains. While the authors did perform experiments using this exact version of the hierarchical priors that they propose, they found that instead of the exact version, an approximation to the hierarchical prior worked better empirically. The approximation involves learning the feature weights for each domain individually, based on the labeled data from each domain, and then using the shared feature hierarchy to influence the priors on the target domain. This is similar to Chelba and Acero (2004), but it involves using the parameters of the more general feature classes when a more specific feature in the target domain is not found in the source domain. We comment later on the applicability of this and the next method by Finkel and Manning (2009) (which is similar to the exact version of Arnold et al. (2008)) to our problem.

Finkel and Manning (2009) put a hierarchical Bayesian prior on the weights to be learned for features in a discriminative learning setup. Normally, the weights of features

in a discriminative learning setup (such as logistic regression or conditional random field) are regularized by putting a zero-mean, uniform variance Gaussian prior on them. The authors instead propose a hierarchical structure for the prior — each domain has its own Gaussian prior (potentially with a different non-zero mean and uniform variance from the priors of other domains), and on top of each of these domain-specific parameter priors, there is a zero-mean uniform variance Gaussian hyper-prior. The idea is to allow the model to learn different weights for the same feature across different domains, and also if a particular domain does not have enough evidence to assign a meaningful weight to some features, then allowing it to "borrow" from the weights learned for the other domains, through the root-level hyper-prior. The authors show that this model is equivalent to that of Daumé III (2007), albeit with some parameters made explicit — and that tuning these explicitly makes a difference in performance.

The approaches by Arnold et al. (2008) and Finkel and Manning (2009) are similar to each other, and also both applicable in the multi–domain learning scenario that we consider in this thesis. However, the approach of Arnold et al. (2008) is driven by a hierarchical structure of features, which is not the focus of our work; and since the approach by Finkel and Manning (2009) is essentially equivalent to that of Daumé III (2007), we only consider the Daumé III (2007) baseline in our work.

While our work on multi–domain learning is closely related to the supervised domain adaptation approaches presented in this section, a key difference is that multi–domain learning is not limited to the case where the test data points belong to a single target domain of interest. Having said that, we also realize that most supervised domain adaptation approaches can in fact be also evaluated in a setup where the test set contains all of the domains seen at training time. Another difference from our work is that research on supervised domain adaptation so far has been restricted to only a few domains. In the setting that we consider, the number of distinct domains can be much larger. Finally, prior approaches to domain adaptation have assumed that any given data point belongs to just one of the few domains being considered. Contrary to that, in our case, we will consider a scenario where a data point can belong to more than one domain *simultaneously*. For example, a product review can simultaneously belong to the domain formed by "all the reviews for product P" and also the domain formed by "all the reviews for product category C"; the latter being a "broader" definition of domains than the former, since every product belongs to some product category.

2.3 Multitask Learning

Multitask learning (Caruana, 1993, 1997) involves learning multiple related prediction tasks "in parallel." In the original definition of multitask learning, **DIFF II** (Section §2.1: the difference in the possible classification labels) is the focus of research. In general, this translates to having multiple related labels or output variables for each training instance. Thus, the group of instances is the *same*, leading to identical $P(\mathbf{X})$. However, the set of associated labels is different for each learning task, thus leading to different sets of $\mathcal{Y}$ values. For example, as discussed by Caruana (1997), when the principal task is learning to identify the vertical position of a door knob in any given image of a single or double door, one can have other auxiliary tasks that are related to the principal task. Examples of such auxiliary tasks are predicting whether the image contains a single door or double door, predicting the horizontal position of the door knob, predicting the position of the left and right edges of the door and so on. Thus, the feature distribution $P(\mathbf{X})$ across the tasks is the same (for example, the pixel information from the same image). However, the y values being predicted are different from each other, but related (since they are all present together, relative to each other in the same image). Thus in this case the regression function $y = f(\mathbf{x}_i)$ to be learned is different for each task because the output variables have different semantics.

We also note here that the structural correspondence learning (SCL) approach proposed by Blitzer et al. (2006) builds upon a multi-task learning framework to create an augmented feature space that encodes a correspondence or mapping across the source and target domains.[3] The auxiliary predictors that SCL learns for predicting *pivots* (features that are shared across the source and the target domain) represent multiple related tasks. The auxiliary predictors utilize almost the same set of features for predicting the presence or absence of different pivots. Thus, $P(\mathbf{X})$ is almost the same, but the classification task differs, since each auxiliary predictor learns to predict the presence or absence of a different pivot.

Later formulations of multitask learning, however, have generalized the setting introduced by Caruana (1993) beyond the setting of using multiple tasks on the same set of input instances. In particular, the work by Baxter (2000) on *bias learning* formulates a multitask learning setup that can in fact accommodate the multitask learning setup as formulated by Caruana (1993), *supervised domain adaptation*, as well as the multi–domain

[3] The work by Blitzer et al. (2006) is also related to developing useful feature representations for domain adaptation. We elaborate upon this in § 2.5 below.

learning scenario that is central to this thesis. When using the term "multitask learning," a significant number of prior papers refer to this broader formulation of multitask learning. However, their evaluation scenario is typically focused on the multi–domain learning setting where there are multiple *tasks* or *domains* at training as well as test time. We will review some of those papers in the next section on multi–domain learning.

2.4 Multi–domain Learning

The multi–domain learning setting in its most general form was first introduced by Dredze and Crammer (2008). The key aspect to all the scenarios that they consider is that there are data points from multiple domains available at training time. Also, similar to domain adaptation, the problem or the prediction task for each of the domains is identical. What can differ, however, is the distribution of the input features across the different domains, and the (classification) function to be learned between the inputs $\mathbf{X}$ and outputs $\mathbf{y}$, but not the set of possible y values. Thus, in multi–domain learning, again **DIFF I** (Section § 2.1) is the focus of research. However, the fact that the classification function relating $\mathbf{X}$ and $\mathbf{y}$ can be different for different domains gives it a flavor of multitask learning. The three different multi–domain learning scenarios that Dredze and Crammer (2008) consider are as follows.

The first scenario is where data points from several source domains are available at training time, but at test time, data points from *novel domains* that were not seen at training time are to be classified. This is a typical scenario in applications such as classification of email into "spam" or "not–spam" — new email users are created all the time, and training data does not contain any emails for those users. Yet, the spam classification system should make a prediction for these new users. For this scenario, they propose a classifier combination approach that combines the domain–specific classifiers that are built on each of the source domains (in this case the different users whose labeled emails are available at training time).

The base classifiers that Dredze and Crammer (2008) use throughout their experiments are confidence–weighted (CW) linear classifiers. The key idea in confidence–weighted linear classification is to learn, in addition to a set of feature weights, a set of confidence parameters, one for each feature weight. The confidence parameter for a feature represents the learner's confidence in the current weight of the feature in an online learning setup. CW learning builds upon the passive–aggressive online learning framework intro-

duced by Crammer et al. (2006). For passive–aggressive online algorithms, the derivation of the online update involves solving a minimization problem at every step (every input example), to set the learning rate (or step size) to be used for the current input example. In addition to the learning rate, CW learning introduces another multiplier in the update equation – the inverse of the confidence parameter associated with a given feature. Therefore, the updates are higher for features where the confidence of the CW learner is lower, and vice–versa. Dredze et al. (2008) model the CW parameters as a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\boldsymbol{\mu}$ represents the weight vector of the CW classifier, and $\boldsymbol{\sigma}$ is a diagonal[4] covariance matrix representing the (inverse of the) confidence of the classifier in its weights. The variance $\boldsymbol{\sigma}_i$ for the i-th feature is the inverse of the confidence that the learner has in the feature. In an online learning setup, if at time step t (when given the t-th input example $\langle \mathbf{x}_t, y_t \rangle$), the computed learning rate is τ_t, then a regular passive–aggressive update to compute the new weight vector $\boldsymbol{\mu}_{t+1}$ would be: $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \tau_t y_t \mathbf{x}_t$. However, in the CW setup, the update rule becomes: $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \tau_t \boldsymbol{\sigma}_t y_t \mathbf{x}_t$, thus forcing larger updates for features that have a higher variance (lower confidence). A key benefit of this approach is faster convergence, as demonstrated by Dredze et al. (2008). The variance parameters in CW learning are updated such that they are non–increasing — that is, the more the number of times a feature is seen, the higher is the confidence of the model in its weight. The idea is to allow rapid variations in the feature weight initially, but only allow smaller updates to weights in which the model has gained higher confidence over time.

A Note On Combining Confidence–Weighted Classifiers: Since the classifier combination approach used by Dredze and Crammer (2008) is one of the key aspects in their work, and also important for the work that we present in this thesis, we elaborate upon that here. Given multiple CW classifiers learned individually on each of the source domains in the input training data, the classifier combination proposed by Dredze and Crammer (2008) involves combining them by taking into consideration the confidence estimates of each CW classifier. Since each of the domain-specific CW classifiers is essentially a multivariate Gaussian distribution, the goal of finding the combination is formulated as finding an optimal multivariate Gaussian distribution for which the sum of distances from each of the individual domain-specific multivariate Gaussians is the smallest. Dredze and Crammer (2008) consider two distance measures for solving this optimization problem — the Euclidean distance between the parameters of the combined Gaussian and each of the individual Gaussians, and also the Kullback-Leibler divergence between them, and

[4]Thus the assumption is that the features are not correlated.

solve the minimization problem to derive the combined classifier in each case. A combined classifier formed in such a way incorporates domain-specific aspects from each of the individual classifiers, and neutralizes any aspects that are conflicting across domains.

The second scenario that Dredze and Crammer (2008) consider is also the one that this thesis primarily addresses. The setting is such that data from multiple source domains is available at training time, and at test time, new data from the *same set of source domains* needs to be classified. This is a common situation in many tasks – the number of domains is fixed, and known in advance, and test data points are only likely to belong to one of the known domains. Examples include modeling genders as domains for various tasks on data from social media; modeling individual people as domains for tasks in computer–supported collaborative learning (Stahl et al., 2006); modeling a fixed set of income ranges as domains for advertising; modeling geographic areas within a country as domains for capturing language variation on social media, and so on. For this scenario, Dredze and Crammer (2008) propose an approach to enable learning across multiple domains. In an online learning setting, they learn a shared CW classifier that is updated based on every input instance irrespective of its domain. Additionally, they also learn domain–specific CW classifiers for each of the known domains, which are updated based on input instances from the corresponding domain. An instance is classified by creating a classifier combination of the shared CW classifier with the appropriate domain–specific classifier for the given example. The key problem they address in this setting is the derivation of online updates for the shared and the domain–specific classifiers — this needs to take into account the fact that predictions are always made using the combined classifier, and not using the shared or the domain–specific classifier.

The third scenario is an extension of the previous scenarios to large–scale datasets. In such datasets, the number of domains can be very large. For example, for an email system such as Gmail,[5] there are millions of users. Modeling each one of those users as a domain will lead to computational challenges of storing millions of domain–specific classifiers having a very large feature vocabulary. Instead, in such a situation, it makes sense to cluster users into a more manageable number of groups that represent the domains. Dredze and Crammer (2008) implement such a clustering scheme in their multi–domain learning framework. The key idea is simple — instead of maintaining a single shared classifier, they maintain k shared classifiers, and learn to map domains to one of the shared classifiers. A domain is mapped to the shared classifier that has made the least number of

[5] http://mail.google.com/

mistakes on previous examples from that domain.

As mentioned earlier, the second scenario considered by Dredze and Crammer (2008) is the focus of this thesis. However, in addition to their setting, we will consider the case where instances can belong to *multiple domains simultaneously*. For this new setting, the approach proposed by Dredze and Crammer (2008) needs to be adapted appropriately, because the assumption in their approach is that the training data is divided into domains based on a *single* metadata attribute (such as the user identifier in the spam classification task).

We will now briefly summarize another set of approaches to the problem of multi–domain learning. First we note that several of the following papers describe their approach using the "multitask learning" terminology. However, as mentioned before, they are evaluated in a multi–domain learning scenario, in particular, the second scenario where training as well as test data contain instances from the same set of multiple domains. The common aspect to all of the approaches we describe next is the fact that they explicitly aim to model *relationships* between the domains (or *tasks* in their terminology). While we will follow their terminology in our descriptions below, for our purposes, tasks and domains are interchangeable.

Cavallanti et al. (2008) assume a fixed task relationship matrix in the context of online multitask learning. For K tasks, the task relationship matrix is assumed to be $\frac{1}{K+1}\left(I_{K\times K} + 1_{K\times K}\right)$, where $I_{K\times K}$ is the identity matrix of size $K\times K$ and $1_{K\times K}$ is a matrix of all ones, with the same size. This effectively assumes that every pair of tasks or domains are equally related, and performs "half-updates" (as described by Saha et al. (2011)) to a given task's weight vector upon receiving an example from some other task.

Saha et al. (2011) improve upon the perceptron-based framework by Cavallanti et al. (2008), by learning the task relationship matrix adaptively, instead of assuming it to be fixed. They derive an algorithm for updating the task interaction matrix online.

In a more traditional batch learning setup, Daumé III (2009) proposes a joint task clustering and multitask learning setup. The task clustering is performed using hierarchical clustering in a Bayesian framework, using a hierarchical prior on the structure of the task relationships.

Also in a batch learning setup, Zhang and Yeung (2010) propose a convex formulation to learning task relationships. The key idea in their work is the use of a matrix-normal prior on the matrix composed of the task-specific weight vectors along the columns. The column covariance matrix parameter of the matrix-normal prior, which represents the

covariance between the task-specific weight vectors is the task interaction matrix that is learned adaptively.

In this thesis, we do not focus on this line of work on multitask learning that aims to learn relationships between the domains. The primary reason we do not focus on those approaches is that we are interested in a scenario where a given data instance can belong to several domains simultaneously, and in such a scenario, the task of learning relationships between domains can become computationally prohibitive. However, for our empirical evaluation of multi–domain learning approaches, we do consider the work by Zhang and Yeung (2010) as a representative approach in this category of approaches.

2.5 Feature Representations for Domain Adaptation

A complementary line of work to the approaches to domain adaptation that we have discussed earlier is that of designing an appropriate feature representation that facilitates learning of models that generalize across domains. One application area where this is particularly interesting is that of natural language processing — and all of the tasks that we consider belong to this category. We now review some of the existing approaches that consider representational issues for domain adaptation.

Ben-David et al. (2007) provide a formal analysis for the error bound on the performance of an algorithm on the target domain. The bound that they derive includes two terms that depend on the feature representation — the error on the source training domain, and the *domain distance* (termed as the $\mathcal{A}$-distance) between the source and the target domains. The lower the $\mathcal{A}$-distance between the source and the target domains, the better the error bound. Thus, they show that it is important to design a feature representation that minimizes the $\mathcal{A}$-distance between the source and the target domains. In other words, a feature representation that helps the learner to see similarities across domains is better. In Chapter 4, we will demonstrate one such feature representation that generalizes across domains for the task of detecting sentences in product reviews that contain an opinion (either positive or negative).

Blitzer et al. (2006) and Blitzer et al. (2007) propose using the structural learning framework by Ando and Zhang (2005) in order to learn correspondences across the source and the target domain using *pivot features* (henceforth *pivots*) that are common to both the domains. The idea is to learn which other features correlate well with the presence/absence of these pivots. These "correlations" are captured in terms of the classifier weights learned

for the other features for the task of predicting the presence of pivots in an instance. Any new instance can then be projected on to a low-dimensional approximation of this learned "correlation matrix," resulting in the creation of new features for the instance. These new features form a shared representation across the source and the target domains. Ben-David et al. (2007) have validated this structural correspondence learning approach in their formal framework, and shown that it does indeed lead to a reduction in the $\mathcal{A}$-distance between the source and the target domains, leading to a better empirical performance.

For many natural language processing tasks, a commonly employed feature representation is the so-called bag-of-words or bag-of-features model, which uses lexical *n*-grams in a text as features. While such a representation is an extremely simple and yet very competitive baseline, it suffers from lack of generalization across different domains. For example, as discussed earlier, the word "unpredictable" can refer to different sentiments in different domains. Another problem is that different words can be relevant for different domains, and thus a model that is learned on one domain may not generalize to other domains due its lack of knowledge about the target-specific *n*-grams. In order to address this issue, several "template–based" feature representations have been explored in prior work. We present one such feature representation in Chapter 4. It is based on syntactic dependency features, and aims to exploit the structural similarity in syntactic dependency features across multiple domains (Joshi and Rosé, 2009).

Among other work on template–based features, McDonald et al. (2007) have used "back–off *n*-grams" by replacing all possible combinations of the words in an *n*-gram with their corresponding part–of–speech tags. Following our work in Joshi and Rosé (2009), Arora et al. (2010) proposed a more general framework of using *subgraph features*, which are based on different feature templates extracted from an *annotation graph* for a given text. Annotation graphs can contain several different types of annotations on text, including part-of-speech labels and dependency parse information. Arora et al. (2010) use subgraph–mining techniques (Yan and Han, 2002) to extract frequent subgraphs of such an annotation graph to identify the subgraph features. Finally, (Gianfortoni et al., 2011) have shown that generalizing across domains (formed by the occupation of bloggers) using "stretchy patterns" is effective for the tasks of gender and age prediction.

2.6 Summary

In this chapter we have reviewed related work on domain adaptation, multitask learning and multi–domain learning. We have highlighted the similarities and differences with respect to the problem that we consider in this thesis — a specific scenario in multi–domain learning where training as well as test data belong to multiple domains, and its extension to the case where instances can belong to multiple domains simultaneously. The primary difference with respect to prior domain adaptation and transfer learning work is that in our case we consider a setting where the goal is not to build optimal models for some specific target task, but to build better models for any of the source domains. We perform a thorough empirical analysis of techniques for this setting in Chapter 3, and present a problem–driven feature representation for this setting in Chapter 4. The primary difference with respect to the prior work in multi–domain learning is that we propose to consider a setting where multiple domains can be simultaneously assigned to any instance (Chapter 5).

With respect to the complementary line of work on feature representation for domain adaptation, we propose a feature representation based on generalizing dependency features for the task of identifying if a sentence from a product review contains an opinion. This work is described in Chapter 4.

3 | Empirical Analysis of Multi–Domain Learning

How domains are defined for multi–domain learning is of central interest in this thesis. While there has been much prior research into developing new methods for the problem of multi–domain learning, there has not been any work that deals with the question of defining the appropriate set of domains for a given task or a given dataset. In other words, there is no well–defined methodology for splitting up a dataset into domains that will be most beneficial for the prediction task of interest. A consequence of this is that in the empirical evaluations of multi–domain learning approaches, domains have been defined primarily based on researcher intuition. The definition of domains is usually based on some metadata associated with the main data of interest. For example, in the case of product reviews on Amazon.com, several metadata attributes can be associated with each of the reviews, including the product identifier, the product category assigned by Amazon.com, the identity of the reviewer, the brand name of the manufacturer, and so on. In prior work that uses the Amazon.com dataset (Blitzer et al., 2007), the product category attribute has been used for defining domains. The intuition is that reviews within a given product category (such as `books`) will have some similarity in their language, and reviews across different product categories will have differences in their language that are important to model. Such problem–driven intuition is very useful, and no doubt valuable in defining domains. However, as we will demonstrate in this chapter, it is difficult to correlate such intuition to improvements seen due to multi–domain learning techniques. This is because, first, there does not exist a well–defined methodology for choosing the "right" pairing between a given definition of domains and any of the multi–domain learning techniques. Second, even if intuition might seem to justify such a choice, it is not always reliable. Thus, even positive experimental results might not be attributable to the claimed reasons for their success*There can be other confounding factors that are respon-*

sible for the success, especially when no established methodology is being used for defining domains..

Toward analyzing this question of the meaningfulness of domain definitions for multi–domain learning, this chapter will present an empirical analysis of three existing multi–domain learning approaches that are representative of the different multi–domain learning methods proposed in prior work. The goal of this analysis is to evaluate the sensitivity of existing multi-domain learning algorithms to the definition of domains used. In particular, we will explore two questions related to multi–domain learning:

Q I. Many multi–domain learning techniques use a combination of classifiers learned over domain–specific subsets of the data. This is an idea central to ensemble learning, which in some cases uses randomized subsets of data to learn base classifiers and combines them to achieve better performance. Therefore a valid question to ask is: to what extent can the improvements coming from multi–domain learning techniques be attributed to an ensemble learning effect?

Q II. One simple way in which domains can differ for classification tasks is by having different class distributions on their output labels. This is often the case in real–world datasets where class distributions tend to be non–uniform, or unbalanced. We will evaluate multi–domain learning algorithms in this settings to evaluate their effectiveness under such conditions, where much simpler approaches than multi–domain learning can give robust improvements. The question we consider here is: to what extent do multi–domain learning techniques capitalize on domain–specific class bias that is common in real–world datasets?

Irrespective of the specific answers to these questions, a better understanding of the performance of existing multi–domain learning algorithms in different settings will provide intuitions for improving the state of the art. Thus, a systematic empirical analysis of existing multi–domain learning techniques is also an important goal of this chapter.

We will begin by first formally defining the multi–domain learning problem. Subsequently we will describe in detail the three multi–domain learning methods that we use in this chapter. And then we will present a critical empirical analysis of these methods, in the context of their sensitivity to the definition of domains and their properties. A significant portion of the work presented in this chapter appears in Joshi et al. (2012).

3.1 Multi–Domain Learning

We now formally define the multi–domain learning scenario that we consider in this thesis. In the multi-domain learning setting, examples are accompanied by both a class label and a domain indicator. Examples (training as well as test) are of the form $\langle \mathbf{x}_i, y, d_i \rangle$, where $\mathbf{x}_i \in \mathbb{R}^N$, d_i is a domain indicator, $\mathbf{x}_i$ 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 multi–domain learning uses these to improve learning accuracy. Additionally, in the setting that we consider, our test data points are also drawn from one of the domain–specific distributions for $d_i \in \mathcal{D}$, where $\mathcal{D}$ is a finite set of domains that are *known* in advance.

As much of the prior work on domain adaptation shows, domain differences can introduce errors in a number of ways (Ben-David et al., 2007, 2009). First, the domain-specific distributions D_{d_i} can differ such that they favor different features, i.e. $p(\mathbf{x})$ 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, 2007). Second, the features may behave differently with respect to the label in each domain, i.e. $p(y|\mathbf{x})$ 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 multi–domain learning algorithms is to target one or both of these properties of domain difference to improve performance.

An example where multi–domain learning is appropriate is that of polarity prediction for 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] that must be modeled as well.

We now describe the three multi–domain learning approaches that we evaluate in this chapter.

3.1.1 FEDA

Daumé III (2007) introduced the so–called "Frustratingly Easy Domain Adaptation" (FEDA) method for domain adaptation. We gave an overview of his approach in Chapter 2. We now describe it in more detail. While the technique was proposed for a supervised do-

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

main adaptation setting, it is directly applicable to a multi–domain learning setting as well. FEDA is essentially a transformation of the feature representation for a problem, based on the knowledge of domains for the instances. Therefore it can be applied as a preprocessing step for any dataset before using any machine learning algorithm. Let us assume that the original feature space for our problem is N-dimensional, that is, all instances $\mathbf{x}_i \in \mathbb{R}^N$. Let us further assume that there are K different domains in our data. FEDA maps an instance from $\mathbb{R}^N$ to $\mathbb{R}^{(K+1)N}$. The augmented feature space (in $\mathbb{R}^{(K+1)N}$) consists of the original feature space, along with an augmented set of domain–specific feature subspaces, one for each of the K domains. The transformation is very simple: for an instance from domain k, its transformed representation consists of the original set of features, along with the same set of features *copied into the k-th domain–specific feature subspace*. The features in the other $k-1$ subspaces for that instance have a value of zero (since that instance does not belong to any of those domains). Formally, if there are 3 domains in our data, the transformation Φ for an instance $\mathbf{x}_i \in \mathbb{R}^N$ having domain $d = 2$ works as follows ($\mathbf{0}$ is a zero-vector in $\mathbb{R}^N$):

$$\Phi(\langle \mathbf{x}_i, d_i = 2 \rangle) = \langle \mathbf{x}_i, \mathbf{0}, \mathbf{x}_i, \mathbf{0} \rangle \tag{3.1}$$

After the data instances are transformed in this way, any standard machine learning technique can be applied to this new feature representation. The intention is to allow the learner to discover what features matter across domains (based on their presence across domains in the general subspace), and also enable learning domain–specific feature behavior (based on the presence of features within the domain–specific subspaces). The simplicity of FEDA, and its good empirical performance makes it an attractive starting point for exploring the use of multi–domain learning for a problem.

3.1.2 MTRL

The **M**ulti–**T**ask **R**elationship **L**earning (MTRL) approach proposed by Zhang and Yeung (2010), aims to simultaneously learn domain–specific classifiers and the similarities between the domains, which are used to regularize the domain–specific classifiers. If there are K domains in the data, the relationships between them are learned as a $K \times K$ covariance matrix. Zhang and Yeung (2010) have derived a convex formulation for learning the covariance matrix and the domain–specific classifiers simultaneously. The domain–

specific classifiers $\mathbf{w}_{d_1} \ldots \mathbf{w}_{d_K}$ are stacked together column-wise to create a matrix $\mathbf{W}$. A matrix normal prior is then imposed on $\mathbf{W}$. The matrix normal distribution $\mathcal{MN}_{N\times K}(\mathbf{M}_{N\times K}, \mathbf{\Sigma}_{N\times N}, \mathbf{\Omega}_{K\times K})$ models a $N \times K$ matrix with mean $\mathbf{M}$, where the covariance between the rows is $\mathbf{\Sigma}$, and the covariance between the columns is $\mathbf{\Omega}$. Thus, when applied as a prior on $\mathbf{W}$, $\mathbf{\Sigma}$ models the covariance between the features, and $\mathbf{\Omega}$ models the covariance between the domains (or tasks). Thus, in their learning step, $\mathbf{W}$ and $\mathbf{\Omega}$ have to be learned simultaneously. They assume the mean $\mathbf{M}$ of the matrix normal prior to be zero, and the row covariance matrix $\mathbf{\Sigma}$ to be an identity matrix (that is, the features are assumed to be uncorrelated). As mentioned in Chapter 2, multi–domain learning approaches that model domain relationships are not a focus of this thesis. However, we considered MTRL in our empirical analysis for including a representative approach from the category of multi–domain learning techniques that model domain relationships.

3.1.3 MDR

Multi–**D**omain **R**egularization (MDR) is the approach proposed by Dredze and Crammer (2008), based on a classifier combination of shared and domain–specific classifiers, where both the shared and the domain–specific classifiers are confidence–weighted (CW) linear classifiers (Dredze et al., 2008). Based on CW learning, Dredze and Crammer (2008) propose an online learning framework for multi–domain learning, and learn a shared CW classifier $\mathbf{w}_g$, as well as, domain–specific CW classifiers $\mathbf{w}_1 \ldots \mathbf{w}_K$ for each of the K domains. Prediction for an example from domain d_i is based on a *combined classifier* $\mathbf{w}_{(g,d_i)}$, which is a combination of the shared CW classifier $\mathbf{w}_g$ and the domain–specific CW classifier $\mathbf{w}_{d_i}$. While learning the classifiers, they explicitly enforce the constraint that the combined classifiers at every round of learning perform well on the input example. This requires the derivation of online updates for the shared classifier and the domain–specific classifiers, such that either (i) the updates computed for the individual classifiers should result in a combined classifier that correctly classifies the input example, or (ii) the updates computed for the combined classifier should be distributed to the underlying shared and domain–specific classifiers such that the combination of these *updated* underlying classifiers will yield the *updated* combined classifier.

Since this thesis builds upon the MDR framework in Chapter 5, we now present some more details of the MDR approach here.

Combination of confidence–weighted (CW) classifiers is one crucial component of MDR. We gave an overview of the CW classifier combination in Section §2.4. The goal of

computing a combination is to identify a classifier that incorporates the characteristics of the both the shared classifier $\mathbf{w}_g$, as well as the domain–specific classifier $\mathbf{w}_{d_i}$ (henceforth referred together as the *underlying classifiers*), weighing them appropriately based on their confidence in the different features. For example, if a feature f_j has a low weight with low confidence in the domain–specific classifier, but a high weight with higher confidence in the shared classifier, then the combined classifier should be such that it trusts the shared classifier more than the domain–specific classifier for that feature. In general, this can be achieved by finding a combined classifier $\mathbf{w}_{(g,d_i)}$ such that the sum of its distance from the underlying classifiers is minimized. Distance between two CW classifier can be computed in several ways. Dredze and Crammer (2008) considered the squared Euclidean (l_2) distance and the Kullback–Leibler divergence (KL-divergence) as distance measures, and derived the combined classifiers in both cases. The `MDR` variations that use these two distance measures will be referred to as `MDR-L2` and `MDR-KL` respectively, throughout this thesis. We now show the equations to compute the combined classifier parameters in each case. For a derivation of the equations, the reader is referred to Dredze et al. (2009, pp. 128–129).[2]

Let us denote a confidence–weighted classifier by the N-dimensional multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\boldsymbol{\mu}$ represents the weight vector of the linear classifier, and $\boldsymbol{\sigma}$ is a diagonal covariance matrix representing the confidence of the classifier in its weights. The equations are derived for a combination of M CW classifiers. Let $\mathcal{N}(\boldsymbol{\mu}^C, \boldsymbol{\sigma}^C)$ denote the combined CW classifier, and $\mathcal{N}(\boldsymbol{\mu}^m, \boldsymbol{\sigma}^m)$ denote each of the CW classifiers to be combined. Also, let $\boldsymbol{\mu}_j$ and $\boldsymbol{\sigma}_j$ denote the weight and the variance for the j-th feature ($j = 1 \ldots N$).

Then, for `MDR-L2`, the equation for computing the mean of the combined classifier (the feature weights) is:

$$\boldsymbol{\mu}_j^C = \sum_{m=1}^{M} \mathbf{c}_j^m \boldsymbol{\mu}_j^m \tag{3.2}$$

where

[2]We use a slightly different notation than that of Dredze et al. (2009), but the mapping should be clear based on the equations.

$$\mathbf{c}_j^m = \frac{\mathbf{b}_j^m}{\sum_{m'=1}^{M} \mathbf{b}_j^{m'}} \tag{3.3}$$

The term $\mathbf{b}_j^m$ represents a weight that one can associate with (the features of) the CW classifiers being combined. This is useful if there is some external knowledge that needs to be injected into the combination. It is possible to emphasize or de-emphasize different features to a different extent for each CW classifier. Dredze et al. (2009) refer to this process of setting the $\mathbf{b}$ weight vector as $\mathbf{b}$-*weighting*.

The equation for computing the variance (feature confidence) of the combined classifier using `MDR-L2` is:

$$\boldsymbol{\sigma}_j^C = \sum_{m=1}^{M} \mathbf{c}_j^m \boldsymbol{\sigma}_j^m \tag{3.4}$$

Thus, for the `MDR-L2` technique, the combined classifiers are simply a weighted combination of the underlying classifiers.

For the `MDR-KL` approach, the equation for computing the mean of the combined classifier is:

$$\boldsymbol{\mu}_j^C = \frac{\sum_{m=1}^{M} \frac{\mathbf{c}_j^m}{\boldsymbol{\sigma}_j^m} \boldsymbol{\mu}_j^m}{\sum_{m=1}^{M} \frac{\mathbf{c}_j^m}{\boldsymbol{\sigma}_j^m}} \tag{3.5}$$

And the variance of the combined classifier for `MDR-KL` is computed as:

$$\boldsymbol{\sigma}_j^C = \frac{1}{\sum_{m=1}^{M} \frac{\mathbf{c}_j^m}{\boldsymbol{\sigma}_j^m}} \tag{3.6}$$

For `MDR-KL`, again the combined classifier is a weighted combination of the underlying classifiers. Note however that the weighting scheme for computing the combined classifier mean now incorporates the variance (feature confidence) of the underlying classifiers — the higher the variance of an underlying classifier's mean, the lower is the weight given to it in computing the combined mean. The combined variance in this case is a weighted

harmonic mean of the variance of the underlying classifiers.

Based on the observation that the combination for MDR-KL takes into account the confidence of the underlying classifiers when computing the combined *mean*, Dredze and Crammer (2008) proposed a heuristic to encourage that behavior explicitly. For this purpose, they make use of the fact that setting $\mathbf{b}_j^m$ will influence the weighting scheme as discussed earlier. Thus, to take feature confidence into account, they use **b**-weighting as follows:

$$\mathbf{b}_j^m = a - \boldsymbol{\sigma}_j^m \tag{3.7}$$

The term a in equation 3.7 is the value to which $\boldsymbol{\sigma}_j^m$ is initialized. Note that since $\boldsymbol{\sigma}_j^m$ in CW classifiers is non–increasing, $\mathbf{b}_j^m \geq 0$ always. The **b**-weighting used in equation 3.7 is termed as *variance* **b**-weighting by Dredze and Crammer (2008). This is in contrast to the other setting they use, where all $\mathbf{b}_j^m$ terms are set to 1, which they call *uniform* **b**-weighting.

Despite the fact that the combination equations for MDR-KL already take feature confidence into account, Dredze and Crammer (2008) apply the variance **b**-weighting to MDR-KL as well, and obtain improved results in some cases. However, overall, the uniform b-weighting scheme was the best for both MDR-L2 as well as MDR-KL approaches in their experiments, and we use the same for all experiments with MDR in this chapter.

We note here that while the MDR-L2 and the MDR-KL techniques only require combining *two* underlying classifiers (a shared CW classifier and a domain–specific CW classifier), the above equations for deriving the combined classifier hold for a combination of more than two CW classifiers. This is in fact also used by Dredze and Crammer (2008) for the evaluation scenario where they consider novel domains at test time and combine multiple CW classifiers, each trained on a separate source domain in the training data. We will use a combination of more than two underlying classifiers in our extension of the MDR framework to handling multiple domains per instance (Chapter 5).

The second crucial aspect of MDR is that during learning, an explicit constraint enforces that the combined classifier should perform well on the input example. As mentioned before, this translates to one of the following two update scenarios:

UPD I. If updates are computed for the underlying classifiers, then the *updated* underlying classifiers $\mathcal{N}(\boldsymbol{\mu}^m, \boldsymbol{\sigma}^m)$ should result in a combined classifier $\mathcal{N}(\boldsymbol{\mu}^C, \boldsymbol{\sigma}^C)$ that correctly classifies the input example $\langle \mathbf{x}_i, d_i \rangle$.

UPD II. If the updates are computed for the combined classifier, then they should be distributed to the underlying shared and domain–specific classifiers such that the combination of the *updated* underlying classifiers (computed using the distributed updates) will yield the *updated* combined classifier.

Both of these update scenarios require solving a minimization problem that satisfies the required constraints. Dredze et al. (2009) show the derivation of the update rules in each of the above update scenarios, for the case when the l_2 distance measure is used for creating the combined classifier. A similar update that is suitable for online learning is unfortunately not possible for the case where KL-divergence is used as the distance measure. Even in the case of the l_2 distance, the derived updates are not in a closed form, and require a line search for some parameters, thus leading to an approximation of the correct solution to the desired objective.

Instead, what Dredze et al. (2009) found to be more useful was an *averaged update* scheme. The averaged update only considers the update scenario **UPD II** above. It first computes the updates for the combined classifier, as one would do for a standard CW classifier. Those computed updates are then directly distributed to the underlying classifiers (the shared classifier, and the relevant domain–specific classifier), *without explicitly ensuring whether the combination of these updated underlying classifiers matches the updated combined classifier.* The advantage of this averaged update procedure is that it has a closed–form update (the same as a CW update), and it also performs well empirically. Also, most importantly, the averaged update scheme can be also used in the case where KL-divergence is used as a distance measure for classifier combination. In our use of the `MDR` approach and its variants in this thesis, we consistently use the averaged update scheme. When doing the averaged update scheme, there can be several possible strategies used for dividing the updates between the underlying classifiers. Dredze et al. (2009) explore two options:

AVG I. Divide the updates *in equal proportion* to the underlying classifiers, that is, in their case split the updates equally between the shared CW classifier and the domain–specific classifier

AVG II. Divide the updates such that they are *inversely proportional to the feature variance, or directly proportional to the feature confidence*, using a **b**-weighting procedure identical to the one in equation 3.7. Thus, the underlying classifier with higher variance will get a *lower* fraction of the update, and vice versa.

Notice the similarity of both these update distribution heuristics to the **b**-weighting

procedure used in classifier combination (Section §3.1.3). Both of the heuristics can be implemented using a **b**-weighting procedure (**AVG I** using $\mathbf{b}_j^m = 1$ for all features of all underlying classifiers, and **AVG II** using equation 3.7), similar to *uniform* **b**-weighting and *variance* **b**-weighting. Dredze et al. (2009) always use **AVG I** updates with the *uniform* **b**-weighting scheme for classifier combination, and **AVG II** updates with the *variance* **b**-weighting scheme. However, it should be noted that it does not necessarily have to be so.

3.2 Data and Methods

To support our empirical analysis we develop several experiments. In this section, we describe the datasets and the different methods that we use in our experiments, then proceed to our exploration of multi–domain learning.

3.2.1 Datasets

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

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 polarity prediction, also known as 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 for the Amazon dataset 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 set size of some of the multi–domain learning methods.

ConVote (Convote): Our second dataset is taken from transcribed segments of speech from the 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 multiple ways, based on the different metadata attributes available with the dataset. We consider two types of domain divisions: one based on the bill identifier, and the second based on 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. Based on the attribute that defines the domains, we create two versions of the CONVOTE dataset. We refer to these two 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.

3.2.2 Learning Methods and Features

Among multi–domain learning algorithms, we consider the three approaches described earlier in this chapter in Section §3.1 — `FEDA`, `MTRL`, and the two variations of `MDR` (`MDR-L2` and `MDR-KL`). For `FEDA`, which is a technique that can be used with any machine learning algorithm, we use two learning algorithms — support vector machines (SVMs), and logistic regression (LR), which are described below for the "single classifier" baseline. The `FEDA`–enhanced SVM and LR models are called `FEDA-SVM` and `FEDA-LR` in this chapter, respectively.

In addition to these multi–domain 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). We omit this baseline for simplicity.

To obtain a single domain–agnostic classifier we use two classification algorithms:

support vector machines (SVMs) and logistic regression.

Support Vector Machine (SVM): This represents 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: $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 1\}$.

Logistic Regression (LR): This represents 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.

Now we will consider each one of the two empirical questions we proposed at the beginning of this chapter. For each question, we will motivate our empirical analysis, propose a contrarian hypothesis, present our experimental methodology for evaluating the hypothesis, and present our experimental results with a discussion of their implications.

3.3 Classifier Ensembles

Here we explore the question of whether domain definitions are used by existing multi–domain learning algorithms in meaningful ways. While it is well–understood that 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 multi–domain learning algorithms can be attributed to correcting these errors, or whether they are benefiting from something else. In particular, there are many similarities between multi–domain learning and ensemble methods, with connections to instance bagging, feature bagging and classifier combination. It may be that gains in multi–domain learning are the usual ensemble learning improvements. Thus, we consider the following question for our analysis:

> **Q I**: *Are multi–domain learning improvements the result of ensemble learning effects?*

Many of the multi–domain learning 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 multi–domain learning 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 multi–domain learning 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 multi–domain learning, 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, for example, 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 multi–domain learning 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 multi–domain learning.*

Empirical Evaluation: We evaluate this hypothesis as follows. We begin by constructing a true multi–domain learning setting, in which we attempt to improve accuracy through knowledge of the domains. We will apply three multi–domain learning 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 multi–domain learning 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. We call this the "no class bias" condition. 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 a different randomization of the domain labels each time.

Results: Results are shown in Table 3.1. 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 Zhang and Yeung (2010),[3] the average accuracy that we achieve for MTRL

[3]This might be due to a different version of the dataset being used in a cross-validation setup, rather

	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 3.1: **No Class Bias**: A comparison between multi–domain learning 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 two–tailed paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a two–tailed paired t-test.

(84.2%) is better than the best average accuracy in the original paper (83.65%).

The main comparison to make in Table 3.1 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 multi–domain learning algorithms. Randomizing the domains does not work better than knowing true domains in any case. However, in all except one case, the improvements of multi–domain learning algorithms are significantly better only for the Amazon dataset.[4] And interestingly, for the Amazon dataset, 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

than their train/test setup, and also because of differences in baseline approaches.

[4]Some numbers in Table 3.1 might appear to be significant, but are not. That is because of high variance in the performance of the methods across the different folds.

is not completely true, it is indeed the case that some multi–domain learning methods benefit from the ensemble learning effect.

A second observation to be made from these results is that, while all of empirical research on multi–domain learning 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 relative improvements over the corresponding baseline approach than using bills as domains.[5]

Table 3.6 shows the details about which of the differences between true and randomized domains are significant. For the results presented in Table 3.1, the columns of Table 3.6 under the heading "**No Class Bias**" should be referred. We see that in eight out of 15 cases (5 multi–domain learning algorithms × 3 datasets), that is effectively in more than half of the cases, there is *no significant difference between using true domains and using randomized domains*.

Thus, we draw the following conclusion from our analysis of the connection between multi–domain learning and ensemble learning:

> **Conclusion**: *While the knowledge of domains is not irrelevant for multi–domain learning, a significant part of the gains in multi–domain learning can be attributed to ensemble learning effects.*

The main implication of this conclusion is for the evaluation methodology used for multi–domain learning techniques. As mentioned at the beginning of the chapter, empirical evaluations of multi–domain learning are based on domain definitions that are created based on researcher intuition. However, as we saw in the case of the AMAZON dataset, which is one of the canonical datasets used in multi–domain learning, there can be other factors that lead to improvements when using multi–domain learning techniques. In particular, improvements due to ensemble learning can be misinterpreted as apparent improvements due to multi–domain learning techniques. Also, in more than half of the cases, the performance using true domains was no different from the performance using randomized domains. Thus, future research in multi–domain learning should consider using the methodology of evaluating multi–domain learning techniques on randomized domains as a standard practice. It provides a good way for judging the utility of the domain definitions that are used.

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

3.4 Domain–Specific Class Bias

One simple way in which domains can be different is with respect to the marginal distribution of the 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 multi–domain learning 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. Therefore, the question we consider here is:

> **Question**: *Are multi–domain learning 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 multi–domain learning 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 multi–domain learning 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 multi–domain learning in such settings may simply be capturing domain-specific class biases. 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

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 3.2: **Varying Class Bias**: 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 §3.4

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**: *Multi–domain learning largely capitalizes on domain-specific class bias.*

Empirical Evaluation: To evaluate our hypothesis, for each of our three datasets we create 4 different versions, each with a different setting of the domain-specific class-bias. We call this the "varying class bias" condition. A summary of the dataset partitions is shown in Table 3.2. 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

	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 3.3: **Varying Class Bias**: A comparison between multi–domain learning 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 two–tailed paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a two–tailed paired t-test.

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.

DOM-ID: In addition to the multi-domain learning approaches, and the SVM and LR 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. This is done both for the SVM and LR methods. While we already include a general bias feature in our baseline, 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 3.3. The table follows the same structure as Table 3.1, with the addition of the results for the `DOM-ID` approach. We first examine the efficacy of multi–domain learning in this setting. An observation that is hard to miss is that multi–domain learning results in these experiments show significant improvements in almost all cases, as compared to only a few cases in Table 3.1, despite the fact that even the baseline approaches have a higher accuracy. This shows that multi–domain learning 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 multi–domain learning results on true domains to the `DOM-ID` baseline gives us an idea of the extent to which much multi–domain learning benefits purely from class bias differences across domains. We see that in most cases, about half of the improvement seen in multi–domain learning 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 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 multi–domain learning literature.

Finally, we also include the "Random Domain" evaluation in the our class biased version of experiments. Each "Random Domain" result in Table 3.3 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 the AMAZON dataset, the multi–domain learning results using randomized domains are consistently better as compared to knowing the true domains. 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 class bias across domains play an important role, even in the case of randomized (and therefore 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 3.6, 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
BILL					
031	N	16	12	8	4
	Y	4	8	12	16
088	N	16	12	8	4
	Y	4	8	12	16
132	N	16	12	8	4
	Y	4	8	12	16
PARTY					
D	N	10	30	15	25
	Y	30	10	25	15
R	N	10	30	15	25
	Y	30	10	25	15

Table 3.4: **Consistent Class Bias**: 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 §3.4

Consistent Class Bias

We also ensure that the results we saw in the previous section on *varying class bias* are in fact due to the domain–specific class biases, and not just due to having a class bias in general. For that purpose, we performed a set of experiments that apply multi–domain learning algorithms to a setting where the datasets have different class biases (unlike the experiments reported in Table 3.1, where the classes are balanced). However, unlike the experiments reported in Table 3.3, this time 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 3.4. The results for this set of experiments are reported in Table 3.5. The structure of Table 3.5 is iden-

	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
	DOM-ID					
True Domain	-0.06	-0.11	-0.25	+0.83	-0.06	+0.94
Random Domain	-0.08	-0.04	-0.72	+0.07	-0.31	▲ +0.59

Table 3.5: **Consistent Class Bias**: A comparison between multi–domain learning 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 two–tailed paired t-test. ▼: Significantly worse than corresponding baseline, with $p < 0.05$, using a two–tailed paired t-test.

Method	No Class Bias (Tab. 3.1)			Varying Class Bias (Tab. 3.3)			Consistent Class Bias (Tab. 3.5)		
	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 3.6: The table shows the datasets (AM:AMAZON, BI:BILL, PA:PARTY) for which a given multi–domain learning 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 multi–domain learning method.

tical to that of Table 3.3. Comparing these results to those in Table 3.1, we can see that in most cases the improvements seen using multi–domain learning algorithms are lower than those seen in Table 3.1. 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. Finally, we note that although we included the DOM-ID baseline in our experiments on datasets with *consistent class bias*, we *did not* expect it to perform well since the class bias is identical across all the domains. The results in Table 3.5 confirm this intuition.

True vs. Randomized Domains

In Table 3.6 we analyze the difference in performance of multi–domain learning 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 two–tailed 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 3.3) 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 multi–domain learning in a domain-specific class bias scenario. Since randomizing the domains works better for all of the multi–domain learning methods on AMAZON, it suggests that a combination of ensemble learning and domain–specific class bias is mostly 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.

Thus, we draw the following conclusion from our analysis of multi–domain learning in the presence of domain–specific class biases:

> **Conclusion**: *Multi–domain learning capitalizes significantly on domain–specific class bias. Additionally, multi–domain learning applied to randomized domains in combination with domain–specific class biases leads to significant improvements in most cases. This strengthens the connection between ensemble learning and multi–domain learning methods.*

This conclusion too, has implications for the evaluation of multi–domain learning methods. First, multi–domain learning should be evaluated in a realistic setting where domain–specific biases might be common. Second, a simple approach to capture domain–specific class biases (`DOM-ID`) should be considered as a standard baseline in such scenarios. And finally, in an unbalanced class setting, the ensemble learning effect seems to be even more prominent, and therefore should be accounted for by evaluating the performance of multi–domain learning techniques on randomized domains.

3.5 Summary

This chapter presented a systematic empirical analysis of three existing multi–domain learning algorithms — `FEDA`, `MTRL`, and `MDR`. We explored their empirical performance on three different datasets, using a different definition of domains in each case, and explored

two questions.

The first question analyzed the connection between multi–domain learning approaches and ensemble learning. This is an intuitive connection to make since many multi–domain learning approaches involve a classifier combination of domain–specific classifiers. We showed that there can be a significant ensemble learning effect in the results of multi–domain learning techniques, and it is linked to the definition of domains used by the multi–domain learning techniques. As a consequence, a concrete suggestion we have for future evaluation methods used for multi–domain learning is as follows: the meaningfulness of the domain definitions for multi–domain learning should be evaluated by testing the multi–domain learning technique after randomly shuffling the domain identifiers among the data points (ideally several times for statistical validity), effectively creating an ensemble learning method. The performance of the multi–domain learning technique using randomized domains will help in assessing to what extent the domain definitions are truly helping beyond providing gains due to an ensemble learning effect.

The second question analyzed the performance of multi–domain learning techniques in a real–world data scenario, where class biases might differ across the different domains. We showed that multi–domain learning significantly capitalizes on domain–specific class bias, and also that the ensemble learning effect is more prominent in the case where data has domain–specific class biases. We also introduced a simple baseline `DOM-ID` that captures domain–specific class bias and shows significant improvements over a domain–agnostic baseline. A concrete implication of this finding is that multi–domain learning should be evaluated in realistic data situations, along with a simpler baseline `DOM-ID` that can capture domain–specific class biases.

4 | Generalizing Features Across Domains

As with most machine learning problems, there are two conceptual approaches to tackle the problem of multi–domain learning — an algorithmic approach that is aimed at designing new machine learning algorithms to solve the problem, and a feature engineering approach that is often driven by knowledge about the specific prediction task of interest. We note that these are not necessarily mutually exclusive, and in fact can even complement each other, depending on the nature of data. In the previous chapter we presented an empirical analysis of three algorithmic approaches to multi–domain learning, and concluded with suggestions to modify the evaluation methodology of multi–domain learning techniques. In this chapter, we explore the second conceptual approach of designing a problem–driven feature representation for multi–domain learning.

With the goal of taking the properties of a given definition of domains into account, this chapter will present one way of designing a feature representation that is capable of generalizing across domains. Experiments will be presented that demonstrate the effectiveness of the new feature representation for the task of opinion mining on sentences from product reviews. We will consider the task of opinion mining in this chapter, and show that when classifying sentences from product reviews as "opinion–bearing" or "non–opinion–bearing," syntactic dependency features are more useful when they are engineered to adapt across multiple domains in the training data. Similar ideas of engineering features that adapt between a source and a target domain, or generalize across multiple domains (Arnold et al., 2008), and generalize across different users in an email dataset (Dredze, 2009), have been proposed in prior work. Also, following the work presented in this chapter (part of which also appears in Joshi and Rosé (2009)), generalizing across domains using the so–called "stretchy patterns" has been shown to be effective on the tasks of gender and age prediction (Gianfortoni et al., 2011). This chapter presents

another type of features that can be engineered to perform better across domains.

Generalization strategies for features can, and often do work even in the absence of any domains in data. A key question, therefore, for our purposes is whether our proposed approach really capitalizes on the presence of multiple domains. To answer this question, we will present an experimental analysis which demonstrates that the designed features do indeed capitalize upon the presence of multiple domains in our training data.

Part of the work presented in this chapter appears in Joshi and Rosé (2009).

4.1 Beyond Lexicalized Features

The approach we present in this chapter is based on the observation that for many NLP tasks, machine learning models might overfit the training data due to the highly lexicalized features that are often used for representing the data points. These include typical features such as lexical *n*-grams and lexicalized dependency relation features. This limitation has long been recognized in previous work on opinion mining (Wilson et al., 2004), and sentiment analysis in customer feedback data (Gamon, 2004). Even in much earlier work, Hearst (1992) used *lexico-syntactic* patterns to automatically extract hyponyms from text, although her work was not motivated from the perspective of avoiding overfitting. In our work, we establish the utility of abstracting away from lexicalized features especially in the context of multi–domain learning.

To see the problem that lexicalized features can cause in a multi–domain learning setting, let us consider the task of opinion mining in product reviews. In particular, we consider identifying whether a sentence in some product review contains an opinion about one of the aspects[1] of the product (Hu and Liu, 2004). Since different products will have different aspects that the users talk about in reviews, lexical *n*-grams or lexicalized dependency relation features for each product can contain product–specific aspect words. Thus, here we consider the product identifier as the source of domain information. In such a scenario, a machine learning algorithm may not be able to generalize patterns across the different domains represented by the different products. This holds especially for features with some internal structure, such as *n*-grams, or syntactic dependency relations. For example, the two bigrams "great camera" and "great software" clearly share the structure that both of them have the word "great" in them, and are therefore both very

[1]We use the term *aspect* to refer to any feature of a given product, such as the *lens quality* of a camera. However to avoid confusion with the machine learning usage of the word "feature," we refer to it as an aspect.

good indicators of an opinion–bearing sentence. However, the fact that the second word in each of those bigrams is different prevents a machine learning model from seeing the structural similarity between them, and thus their similar relationship to the *opinion* class label. Abstracting away from the lexicalized representation can help a machine learning algorithm to notice this similarity. One approach to achieve this has been explored by McDonald et al. (2007), where they replace every possible combination of words in an *n*-gram by the corresponding part–of–speech tags. For example, the bigram "great camera" generates the following four features: "great camera", "`JJ` camera", "great `NN`," and "`JJ` `NN`," where `JJ` and `NN` denote the part–of–speech tags (representing adjective and noun) for "great" and "camera" respectively. Motivated by this idea of abstracting away from lexicalized features and the prior work on lexico–syntactic patterns (Hearst, 1992), as well as work by (Gamon, 2004), and (Wilson et al., 2004), we explore the utility of abstracting dependency features for multi–domain learning. In particular, we found that abstracting dependency features *only in part*, creating what we term as "composite backoff" features, is especially useful for the task of opinion mining.

In the remainder of the chapter, we first present a summary of the prior work towards using syntactic dependency features for opinion mining. We then motivate our approach and describe the simple abstraction process for dependency features, including generation of the composite backoff features. This is followed by experimental results and discussion. Finally, we will present experiments that demonstrate the utility of our features especially for the problem of multi–domain learning, as opposed to just being better features in general.

4.2 Syntactic Features for Opinion Mining

The use of syntactic or deep linguistic features for opinion mining has yielded mixed results in the literature so far. On the positive side, Gamon (2004) found that the use of deep linguistic features extracted from phrase structure trees yield significant improvements on the task of predicting satisfaction ratings in customer feedback data. His features include phrase structure patterns for each constituent, syntactic dependency relationships where words are replaced by their corresponding part–of–speech (POS) tags, POS trigrams and information such as transitivity of the predicates, and tense. Matsumoto et al. (2005) show that the use of frequently occurring sub–trees of dependency parse trees as features shows significant improvement in performance on the task of classifying movie

reviews as having positive or negative polarity. Finally, Wilson et al. (2004) use several different features extracted from dependency parse trees to improve performance on the task of predicting the *strength* of opinion phrases. Both Gamon (2004) and Wilson et al. (2004) employ one feature set that is identical to one of our dependency backoff variations; we remind the reader of this connection in § 4.4.

On the flip side, Dave et al. (2003) found that for the task of polarity prediction, adding *adjective-noun* dependency relationships as features does not provide any benefit over a simple bag–of–words feature space. Ng et al. (2006) proposed that rather than focusing on just *adjective-noun* relationships, the *subject-verb* and *verb-object* relationships should also be considered for polarity classification. However, they observed that the addition of these dependency relationships does not improve performance over a feature space that includes unigrams, bigrams and trigrams.

One difference that seems to separate the successes from the failures is that of using the entire set of dependency relations obtained from a dependency parser and allowing the learning algorithm to generalize, rather than picking a small subset of dependency relations based on intuition. However, in such a situation, one critical issue is the sparseness of the very specialized linguistic features, which may cause the classifier learned from such features to not generalize, similar to the case of *n*-grams we discussed in Section §4.1. Fortunately, features that have some internal structure, such as those based on dependency relations, provide a nice way to enable generalization to the desired extent. In the next section, we motivate this idea in the context of our task, from a linguistic as well as machine learning perspective.

4.2.1 Generalizing Dependency Features

We believe that with respect to the use of syntactic features (syntactic dependency relations in particular), there is a need to alter their representation so that they generalize well. Toward validating this hypothesis, we show that by altering syntactic dependency relation triples in a particular way (namely, "backing off" only the head word in a dependency relation to its POS tag), they generalize better across domains and yield a significant improvement on the task of identifying opinions from product reviews.

Before motivating the use of dependency relations as features for our task, a brief overview about dependency relations follows.

Dependency Relations	
nsubj(camera-5, it-1)	cop(camera-5, is-2)
det(camera-5, a-3)	amod(camera-5, fantastic-4)
advmod(worth-8, well-7)	det(price-10, the-9)
amod(price-10, worth-8)	conj_and(camera-5, price-10)

Table 4.1: The table shows dependency relations generated by the Stanford parser for the sentence "*it is a fantastic camera and well worth the price*".

Dependency Relations

The dependency parse for a given sentence is essentially a set of triples, each of which is composed of a grammatical relation and the pair of words from the sentence among which the grammatical relation holds. For example, using the notation of the Stanford parser,[2] in the dependency triple $rel_i(w_j,w_k)$, rel_i is the dependency relation among the words w_j and w_k. The set of dependency relations is specific to a given parser. We use the Stanford parser for computing dependency relations. More formally, for an n-word long sentence $\{w_1, ..., w_n\}$, with m possible grammatical relations $\{rel_1, ..., rel_m\}$, any dependency triple is of the form $rel_i(w_j,w_k)$ where $1 \leq i \leq m$ and $1 \leq j,k \leq n$. The word w_j is usually referred to as the *head word* in the dependency triple, and the word w_k is usually referred to as the *modifier word*. Given the set of dependency triples for a sentence, they can be organized as a directed graph, with the words forming the nodes of the graph and the relations forming the directed edges. This representation is called the dependency parse tree.

Table 4.1 provides an example of the set of dependency relations for one of the review sentences. Two of the grammatical relations shown in Table 4.1 are as follows: **(i)** *amod*: This is the adjectival modifier relation between the adjectival modifier "fantastic" and the noun "camera." **(ii)** *advmod*: This is the adverbial modifier relation between the adverbial modifier "well" and the modified adjective (the head word in the dependency triple) "worth." For a complete list of the dependency relations that can be generated by the Stanford parser, the reader is referred to de Marneffe et al. (2006) and the Stanford Dependencies manual online.[3]

One straightforward way to use dependency relations as features for machine learning is to generate features of the form `RELATION_HEAD_MODIFIER` and use them in a standard bag-of-words type binary or frequency-based representation. The indices of the head

[2] http://nlp.stanford.edu/software/lex-parser.shtml
[3] http://nlp.stanford.edu/software/dependencies_manual.pdf

and modifier words are dropped for the obvious reason that one does not expect them to generalize across sentences. We refer to such features as *lexicalized dependency relation features*, labeled as `LEXDEP` in our results. Next we motivate our backoff approach for dependency features.

4.3 Motivation for our Approach

Consider the following examples from our experimental data:

I. *The iPod's sound quality is pretty good.*

II. *Why is this phone so great?*

Both of these sentences contain a copular verb — "is." In the Stanford parser, this yields the following two dependency relations (among others) for the two sentences: `cop_good_is` for the first sentence, and `cop_great_is` for the second sentence.

Notice, for both sentences, the already known advantage of dependency relations being able to capture associations between non-contiguous words.

Although both of the above dependency features are good indicators of opinion sentences and are closely related, any machine learning algorithm that treats these features independently will not be able to generalize their relationship to the opinion class. Also, any new test sentence that contains the copular verb "is" in relationship with an adjective different from either "good" or "great", will not receive any importance in favor of the opinion class if it was not observed in the training data.

Now consider the case where we backoff the head word in each of the above features to its part-of-speech tag (`JJ`, since both "good" and "great" are adjectives). This leads to a single feature: `cop_JJ_is`. This has two advantages: first, the learning algorithm can now learn a weight for a more general feature that has stronger evidence of association with the opinion class, and second, any new test sentence that contains an unseen adjective in relationship with the copular verb "is" will receive some weight in favor of the opinion class.

A similar scenario plays out for the following two sentences:

III. *overall this is the best phone i have ever owned.*

IV. *But, for the money it is a good machine.*

They contain the following dependency relations (among others): `cop_phone_is` in sentence **III**, and `cop_machine_is` in sentence **IV**. The difference here is that their com-

mon backoff version is now `cop_NN_is`, which contains a noun tag (`NN`) instead of the adjective tag (`JJ`) for the previous backoff version.

Thus, assuming that for review sentences for unseen products, adjectives and nouns are still tagged correctly by the part-of-speech tagger used, the backoff versions of the original lexicalized dependency features will still fire although the corresponding lexicalized dependency features may not have been observed at training time. The backoff operation is therefore a generalization (or abstraction) of the lexicalized dependency relations.

4.4 Dependency Backoff Features

We now describe all the generalizations of dependency features that we experimented with.

Composite Backoff Features: The idea behind our composite backoff features is to create more generalizable, but not overly general backoff features by backing off to the part-of-speech (POS) tag of either the head word or the modifier word (but not both at once, as in Gamon (2004) and Wilson et al. (2004)) – hence the description "composite," as there is a lexical part to the feature, coming from one word, and a POS tag coming from the other word, along with the dependency relation itself. The most specific version of a dependency relation feature that one can create using POS tags is roughly in the form of a *five-tuple* – $rel_i(w_j/\texttt{POS}_j, w_k/\texttt{POS}_k)$, where $\texttt{POS}_j$ and $\texttt{POS}_k$ are the POS tags for the words w_j and w_k respectively. Note again that although we always keep the word and POS tag indices in our notation, they are not part of the actual features provided to the machine learning algorithm. In our initial experiments we experimented with using this most specific five-tuple version and compared it to the lexicalized dependency triples (the ones without any POS tags) and found the five-tuple version to be slightly worse (though not statistically significant). Therefore we use only the lexicalized dependency triples as our baseline.

The two types of composite backoff features that we create from lexicalized dependency triples are as follows:

I. **`HEAD-BO`**: Here we use features of the form $rel_i(\texttt{POS}_j, w_k)$ where the head word is replaced by its POS tag, but the modifier word is retained.

II. **`MOD-BO`**: Here we use features of the form $rel_i(w_j, \texttt{POS}_k)$, where the modifier word is replaced by its POS tag, but the head word is retained.

Our hypothesis is that the HEAD-BO features will perform better than purely lexicalized dependency relations (LEXDEP) for reasons mentioned in § 4.3. Although MOD-BO features also generalize the lexicalized dependency features, in a relation such as an adjectival modifier (discussed in § 4.3), the head noun is a better candidate to backoff for enabling generalization across different products (the domains), rather than backing off the modifier adjective. For this reason, we do not expect their performance to be comparable to HEAD-BO features.

4.5 Baseline Features

We compare our composite backoff features with other similar ways of generalizing dependency relations and lexical ngrams that have been tried in previous work. We describe these below.

FULL-BO (Full Backoff Features): Both Gamon (2004) and Wilson et al. (2004) utilize features based on the following version of dependency relationships: $rel_i(\text{POS}_j,\text{POS}_k)$, where they backoff the head word as well as the modifier word to their respective POS tags (POS_j and POS_k).

Backoff *n*-grams : Similar to McDonald et al. (2007), we utilize backoff versions of lexical bigrams and trigrams (BI-BO and TRI-BO respectively). In backoff *n*-grams, all possible combinations[4] of the words in a given *n*-gram are replaced by their POS tags. For example, a bigram $w_j\_w_k$ generates the backoff bigrams $w_j\_\text{POS}_k$, $\text{POS}_j\_w_k$, $\text{POS}_j\_\text{POS}_k$. Similarly, each trigram generates seven different backoff trigrams.

In addition to these baseline backoff approaches, we also use other standard set of baseline features, including regular lexical bigrams (BI), lexical trigrams (TRI), POS bigrams (POS-BI), POS trigrams (POS-TRI) and lexicalized dependency relations (LEXDEP). We note here that POS-BI and POS-TRI are actually special cases (and subsets) of the backoff *n*-gram feature sets BI-BO and TRI-BO respectively. Our primary baseline is the set of unigram features (UNI), and for testing our remaining feature sets, we evaluate each of them individually by adding them one at a time to the set of unigram features (UNI).

[4]This *does not* include the case of "no backoff" where no words are replaced with their POS tags to retain original *n*-grams.

4.6 Experiments and Results

Details of our experiments and results using the generalized dependency features follow.

4.6.1 Review Sentences Dataset

The product reviews dataset released by Hu and Liu (2004) and Ding et al. (2008) consists of Amazon.com and CNet.com reviews for 14 different products (digital cameras, MP3 players and so on). Manually annotated class labels are available on this dataset for the task is identifying whether a sentence contains an opinion or not. The definition of an opinion–bearing sentence as proposed by Hu and Liu (2004) is as follows: "*If a sentence contains one or more product features and one or more opinion words, then the sentence is called an opinion sentence.*" Examples of opinion–bearing sentences from the data include: "*for the price it is a well spent investment!*", and "*it is very light weight and has a good signal strength.*" Any other sentence in a review that does not fit the above definition of an opinion–bearing sentence is considered as a non–opinion–bearing sentence. In general, these can be expected to be verifiable statements or facts such as product specifications and so on. Examples of non-opinion sentences from the data include: "*if this doesnt bring back the picture, try pressing this button without playing a dvd.*", and "*after i took their picture with their camera, they offered to take a picture of us.*"

The domains in this dataset are defined based on the product for which a review is written. Each sentence in a review is assigned the domain of the product that the review belongs to.

For our experiments we use use a randomly chosen subset consisting of 2,200 review sentences — 200 sentences each for 11 different products (three cameras, three MP3 players, two network routers, and two cellphones). Our exact subset has been made available for future research.[5] Three products were discarded since they did not have at least 200 review sentences each. The only task that we consider in this work is that of identifying opinionated sentences. The class distribution among the chosen 2,200 sentences is as follows: 1,053 (47.86%) opinion sentences and 1,147 (52.14%) non-opinion sentences.

4.6.2 Machine Learning Parameters

We used the Support Vector Machine (SVM) learner (Chang and Lin, 2011) from the MinorThird Toolkit (Cohen, 2004), along with the χ-squared feature selection procedure,

[5] See http://www.cs.cmu.edu/~maheshj/datasets/acl09short.html

where we reject features if their χ-squared score is not significant at the 0.05 level. For SVM, we use the default linear kernel with all other parameters also set to defaults.

We consider two evaluation scenarios, which differ in the definition of domains that we used for deciding our evaluation folds.

First, we perform an 11–fold cross–validation experiment, where each test fold contains all the sentences for one of the 11 products, and the sentences for the remaining ten products are in the corresponding training fold. We note here that this multi–domain learning setting is slightly different from the one that we considered in the previous chapter. Here, every time we have a *novel* domain in our test set. This is identical to the first multi–domain learning scenario considered by Dredze et al. (2009) (described in this thesis in Chapter 2, Section §2.4). This is in some sense the most difficult multi–domain learning scenario since it tests the ability of our features to generalize to domains that were not seen previously. Later in this chapter when we perform a deeper analysis of our back-off dependency features, we also consider the multi–domain learning scenario where the domains seen at test time are the same as those seen at training time.

Second, we perform four–fold cross–validation experiment, where each test fold contains all the sentences for a given *category of products* — cameras, MP3 players, routers, and cellphones. The folds are not identical in size in this second evaluation since the number of products in the different categories is not the same.

Our results are reported in terms of average accuracy values across the 11 folds.

Results

Table 4.2 shows the full set of results from our experiments. Our results are comparable to those reported by Hu and Liu (2004) on the same task, as well as those by Arora et al. (2009) on a similar task of identifying qualified vs. bald claims in product reviews. For our 11–fold cross–validation, the composite features with the head word backed-off (`HEAD-BO`) are the only ones that achieve a statistically significant improvement over the `UNI` baseline. None of the other backoff strategies achieve a statistically significant improvement over `UNI`. The lexicalized dependency relation features (`LEXDEP`) perform worse than unigrams alone. These results thus demonstrate that composite backoff features based on dependency relations, where only the head word is backed off to its POS tag, present a useful alternative to encoding dependency relations as features for opinion mining. For our 4–fold cross–validation experiment, the trend is almost identical, except that the most general backoff version of dependency features (`FULL-BO`) also achieves

	Features	11–fold	4–fold
	UNI	65.23 (±1.45)	63.14 (±2.32)
no backoff	**UNI+BI**	65.73 (±1.99)	63.73 (±1.85)
	UNI+TRI	65.55 (±1.88)	63.89 (±1.92)
	UNI+LEXDEP	64.18 (±1.62)	61.53 (±2.46)
backoff	**UNI+BI-BO**	65.05 (±1.70)	64.73 (±1.87)
	UNI+TRI-BO	64.68 (±1.55)	63.99 (±1.07)
	UNI+POS-BI	67.59 (±1.71)	66.44 (±0.61)
	UNI+POS-TRI	66.14 (±1.50)	65.33 (±1.32)
	UNI+HEAD-BO	▲ 68.36 (±1.77)	▲ 66.80 (±1.76)
	UNI+MOD-BO	65.59 (±1.63)	63.12 (±2.82)
	UNI+FULL-BO	66.73 (±1.51)	▲ 67.16 (±2.05)

Table 4.2: The table shows the average accuracy (±standard error) values for our 11–fold and 4–fold cross-validation experiments for the different feature configurations. ▲: Significantly better than the UNI baseline, with $p < 0.05$, using a two–tailed paired t-test.

a statistically significant improvement over UNI. FULL-BO is not significantly different from HEAD-BO.

Thus, we show that for the task of opinion mining, a backoff strategy for transforming syntactic dependency features is useful for generalizing a model across the different domains. In our case, we experimented with domains defined by the 11 different products, as well as by the four different product categories. In particular, the composite backoff features were found to be significantly better in both evaluation settings.

4.7 Domains and Backoff Features

The results presented so far in the chapter show that our proposed dependency backoff features improve performance on the task of opinion mining. They were evaluated in a scenario where multiple domains are present in our training data, as is the case for multi–domain learning. However, the evaluation does not include a comparable setting where the data *does not* contain any domain information. Thus, from the earlier results, it is not possible to conclude that the dependency backoff features are *especially useful* when the training data contains multiple domains. For example, it might be the case that the proposed backoff features are just better features in general for the task of opinion mining, *irrespective of the presence or absence of multiple domains in the training data.* In

this section we will present experiments designed to test the following hypothesis:

> **Hypothesis**: *Dependency backoff features are especially useful when the training set contains multiple domains.*

In order to test this hypothesis, we consider three different evaluation scenarios. In the first scenario (`IN-DOM`), we evaluate the dependency backoff features in the presence of a *single domain*. Thus, our training and test sets contain sentences that belong to the *same* product. This is a scenario that evaluates the backoff features when the training set does not contain multiple domains.[6]

In the second evaluation scenario (`MULTI-DOM`), we consider training and test sets that are randomly created from the full dataset, and therefore contain a mix of sentences from *all of the domains (all products)* in our dataset.

The third evaluation scenario (`CROSS-DOM`) is identical to the evaluation presented earlier in the chapter — the test set always contains a *novel domain*, never seen at training time.

4.7.1 Dataset

For this evaluation, we used a larger subset of the data released by Hu and Liu (2004). The primary reason for this is that we have to control for the size of datasets across the three evaluation scenarios above, and the smaller dataset that we used earlier would result in very small training set sizes per domain, especially for the `MULTI-DOM` and `CROSS-DOM` settings where we need to have multiple domains in the training set.

For the `IN-DOM` setting, we randomly picked 500 sentences each for eight of the 14 products in the dataset released by Hu and Liu (2004). Note that since we increased the minimum number of sentences required for each product, we only had eight products instead of the 11 products in the earlier experiment. An eight–fold cross–validation is performed *within each product*.

For the `MULTI-DOM` setting, we created eight different subsets, each of them containing 500 sentences, which consisted of a mixture of sentences from the same eight products, as in the `IN-DOM` setting. For each of these `MULTI-DOM` subsets, we then performed randomized eight–fold cross–validation, where each training and test fold contained a mix of sentences from all of the eight products.

[6] This is based on the assumption that the product identifier, the product type, and the brand name are the factors that matter for defining domains. They are thus are held constant in this setting for a given training and test set.

For the CROSS-DOM setting, we started off with the same eight subsets that were created in the MULTI-DOM setting. However, on each of them we then performed leave–one–product–out evaluation, that is, an eight–fold cross–validation where seven of the products appear in the training set, and the eighth one appears in the test set.

Given the three settings described above, if our proposed hypothesis is true, we should expect higher relative improvements over the LEXDEP baseline in the MULTI-DOM and the CROSS-DOM case, as compared to the improvements that we see in the IN-DOM case.

4.7.2 Features

For this experiment, in order to isolate the effect of just the dependency backoff features, we used only the lexicalized dependency features (LEXDEP) as our baseline feature set. We then compared the performance of the three different backoff versions (HEAD-BO, MOD-BO, FULL-BO) to this baseline.

In order to control for as much variability as possible across the three different evaluation scenarios, we used the full set of extracted features, without any feature selection.

4.7.3 Machine Learning Parameters

We used two different learning algorithms to evaluate the performance of the different versions of the dependency features. First, as before, we used a support vector machine with a linear kernel. Second, we used l_2-regularized logistic regression. In both cases, the regularization penalty was tuned over a validation set, using the following values for the trade off parameter C: $0.001, 0.01, 0.1, 1, 10, 100$.

4.7.4 Results

The results using the SVM learner are shown in Table 4.3, and those using l_2-regularized logistic regression are shown in Table 4.4. We report the results in terms of classification accuracy. Every accuracy value in both the tables is averaged over eight different runs of eight–fold cross validation (thus, an average of 64 different values).

Several trends are clear from both the result tables. First, the performance of the baseline feature set (LEXDEP) decreases from IN-DOM to MULTI-DOM to CROSS-DOM. This is in line with what one would expect. The IN-DOM setting contains homogeneous training and test sets (no domains), a case where it is relatively easier to generalize. The MULTI-DOM setting contains a mixture of domains in both the training and test sets, which is

	LEXDEP	HEAD-BO	MOD-BO	FULL-BO
IN-DOM				
accuracy	63.86 (±0.68)	▲ 66.15 (±0.62)	64.54 (±0.70)	▲ 65.50 (±0.62)
rel. impr.	–	+3.59%	+1.07%	+2.57%
MULTI-DOM				
accuracy	59.51 (±0.53)	▲ 63.22 (±0.73)	▲ 60.94 (±0.56)	▲ 62.66 (±0.73)
rel. impr.	–	+6.24%	+2.41%	+5.31%
CROSS-DOM				
accuracy	56.90 (±1.37)	▲ 59.42 (±1.25)	57.79 (±1.28)	▲ 59.65 (±0.93)
rel. impr.	–	+4.42%	+1.56%	+4.84%

Table 4.3: The table shows the results for an SVM with a linear kernel, for each of the different settings that we evaluated to test the hypothesis that dependency backoff features are *especially useful* in the presence of multiple domains in training data. For each setting, the table shows the absolute accuracy value for each feature set, and the relative improvement of the three different backoff feature sets over the LEXDEP baseline feature set. ▲: Significantly better than the LEXDEP baseline, with $p < 0.05$, using a two–tailed paired t-test.

	LEXDEP	HEAD-BO	MOD-BO	FULL-BO
IN-DOM				
accuracy	64.39 (±0.70)	▲ 67.08 (±0.60)	64.51 (±0.65)	65.95 (±0.61)
rel. impr.	–	+4.18%	+0.19%	+2.43%
MULTI-DOM				
accuracy	58.81 (±0.48)	▲ 64.03 (±0.69)	▲ 60.96 (±0.61)	▲ 63.17 (±0.70)
rel. impr.	–	+8.87%	+3.66%	+7.42%
CROSS-DOM				
accuracy	58.09 (±1.27)	▲ 61.45 (±1.09)	▲ 59.74 (±1.16)	▲ 61.94 (±1.06)
rel. impr.	–	+5.80%	+2.85%	+6.63%

Table 4.4: The table shows the results for l_2-regularized logistic regression, for each of the different settings that we evaluated to test the hypothesis that dependency backoff features are *especially useful* in the presence of multiple domains in training data. For each setting, the table shows the absolute accuracy value for each feature set, and the relative improvement of the three different backoff feature sets over the LEXDEP baseline feature set. ▲: Significantly better than the LEXDEP baseline, with $p < 0.05$, using a two–tailed paired t-test.

harder than the IN-DOM setting because we have a smaller number of data points for each of the domains (products). And finally, the CROSS-DOM setting contains a *novel* domain at test time, a case that is hardest among the three settings from a generalization perspective.

We compute the relative improvements of the three backoff feature sets over the corresponding baseline feature set (LEXDEP) in each of the evaluation settings. Consistently, the relative improvements in the MULTI-DOM and the CROSS-DOM setting are higher than the relative improvements in the IN-DOM setting. This is true for all the three backoff feature sets. Thus, these results clearly show that the backoff dependency features are especially useful in the case of MULTI-DOM and CROSS-DOM settings, where we have multiple domains in the training set.

The relative improvements due to backoff dependency features in the case of CROSS-DOM setting are not as good as the ones in the MULTI-DOM setting. Error analysis of the misclassified examples in the CROSS-DOM setting shows that irrelevant (to the task of opinion mining) backoff features are the main cause of this trend. In the CROSS-DOM case, since the test set contains a novel product, the test instances often contain new features that were not seen at training time. This results in the classification of those instances based on a small set of features. If these are irrelevant backoff features with incorrect weights, then it is easy to make an incorrect prediction.

For example, the backoff feature det_NN_the, which fires for any of the phrases "the camera", "the player", "the ipod" and so on, got a high weight for the "opinion" class in both the CROSS-DOM and the MULTI-DOM case. However, the weight of this backoff feature is more influential[7] in the CROSS-DOM case than in the MULTI-DOM case. This, combined with the fact that sometimes it is one of the few usable features from a test instance, can easily lead to misclassification errors. The case of the backoff feature det_NN_a is similar. In general, it appears that the weights of some irrelevant backoff features are driven to the extreme when we leave out an entire product from the training set.

Based on these experimental results that compare improvements due to generalized dependency features across different domain settings (IN-DOM, MULTI-DOM, and CROSS-DOM), we conclude the following:

> **Conclusion**: *Since the relative improvements in MULTI-DOM and CROSS-DOM settings are higher as compared to relative improvements in the IN-DOM setting, the generalized dependency features are indeed especially useful when there are multiple domains in the training set.*

[7] The influence is considered relative to the weight of the class intercept or the weight of the bias feature.

4.8 Summary

In this chapter we explored a feature representation driven approach to the problem of multi–domain learning, using the task of opinion mining from product reviews as an example. We have shown that for this task, a feature representation based on a simple transformation ("backing off" the head word in a dependency relation to its part–of–speech tag) of syntactic dependency relations captures more generalizable and useful patterns in data than purely lexicalized dependency relations, yielding a statistically significant improvement. Further, we empirically evaluated the backoff dependency features in different settings to establish that they are especially useful in the case where the training data contains multiple domains.

5 | Multi–Domain Learning for Multi–Attribute Data

The empirical analysis that we presented in Chapter 3 showed that depending on the definition of domains used for multi–domain learning algorithms, their performance and utility can vary. In particular, we saw that for the CONVOTE dataset involving prediction of votes, using the political party affiliation was a better choice for defining domains as opposed to using the speaker identifiers as domains. Such variation in performance of multi–domain learning techniques, caused by the particular definition of domains being used, is not desirable. However, it cannot be avoided as long as we are forced to choose a single attribute that defines domains. Thus, to circumvent this limitation, we now present methods that do not restrict the definition of domains to a single metadata attribute in data. Instead, they can use many metadata attributes simultaneously to define domains for a given dataset.

Datasets in text classification often contain multiple metadata attributes associated with the text, many of which are reasonable candidates for defining domains. Consider the case of restaurant reviews, which can be categorized into domains corresponding to the cuisine, location, price range, or a host of other factors. For multi–domain learning, we should use the metadata attribute most likely to characterize a domain: a change in vocabulary (i.e. features) that most impacts the classification decision (Ben-David et al., 2009). However, this choice is not easy. First, we may not know which metadata attribute is most likely to fit this role. Perhaps the location most impacts the review language, but it could easily be the price of the meal. Second, multiple metadata attributes could impact the classification decision, and picking a single one might reduce classification accuracy. Therefore, we seek multi–domain learning algorithms which can simultaneously learn from many types of domains (metadata attributes).

We introduce the *multi–attribute multi–domain* (MAMD) learning problem, in which

each learning instance is associated with multiple metadata attributes, each of which may impact feature behavior. We present extensions to two popular multi–domain learning algorithms, FEDA (Daumé III, 2007) and MDR (Dredze et al., 2009). Rather than selecting a single domain division, our algorithms consider all attributes as possible distinctions and discover changes in features across attributes. We evaluate our algorithms using three different datasets, each containing a varying number of metadata attributes that are potentially relevant for the classification task. We demonstrate that multi–attribute algorithms improve over their multi–domain counterparts, which can learn distinctions from only a single attribute.

In the absence of our proposed methods, a single "best" attribute can be chosen among all the available metadata attributes using empirical tuning on some validation set. However, this method can be brittle if the definition of the "best" attribute changes over time. Also, this method does not benefit from the knowledge of multiple metadata attributes, each of which might influence the text classification task to a varying degree. In such cases, we will demonstrate the benefit of our methods over choosing a single "best" domain definition empirically.

A potential disadvantage in using multiple attributes to define domains could be that irrelevant metadata attributes might hurt the performance. We demonstrate experimentally that on datasets where there is indeed a "single best" metadata attribute[1], our approach is not adversely affected by the presence of irrelevant (for the purpose of defining domains) metadata attributes.

Finally, we also present empirical evaluation that shows that using multiple metadata attributes is at least as computationally efficient as empirically evaluating multiple attributes to find the "single" best one, and in most cases is much faster.

Part of the work presented in this chapter is to appear in Joshi et al. (2013).

5.1 Multi–Attribute Multi–Domain Learning

In multi–domain learning, each instance $\mathbf{x}$ is drawn from a domain d with distribution $\mathbf{x} \sim \mathcal{D}_d$ over a vector space $\mathbb{R}^D$ and labeled with a domain specific function f_d with label $y \in \{-1, +1\}$ (for binary classification). In multi–attribute multi–domain learning, we have M metadata attributes in a data set, where the m-th metadata attribute has K_m possible unique values. Each instance $\mathbf{x}_i$ is drawn from a distribution $\mathbf{x}_i \sim \mathcal{D}_{\mathcal{A}_i}$ specific to a

[1]The "single best" metadata attribute is decided empirically, after testing each of the available metadata attributes one-by-one.

set of attribute values $\mathcal{A}_i$ associated with each instance. Additionally, each unique set of attributes indexes a function $f_{\mathcal{A}}$.[2] $\mathcal{A}_i$ could contain a value for each attribute, or no values for any attribute. In the absence of any metadata attributes, a "background" distribution and the corresponding labeling function can be indexed. Such a background distribution represents the general distribution over the input space $\mathbb{R}^D$, and the corresponding labeling function is domain–agnostic. In this new setup, each metadata attribute can change a feature's probability and behavior, similar to a domain's effect on feature behavior in multi–domain learning.

Examples of data for multi–attribute multi–domain learning abound. The commonly used Amazon product reviews data set (Blitzer et al., 2007) only includes product types, but the original reviews can be attributed with the reviewer, product price, and so on. In prior work, the domains on that dataset have been defined based on the Amazon.com product categorization. While that is a reasonable choice, some of the other metadata attributes can also be potentially used as domains. The price of a product, for example, can affect the expectations about it, and therefore influence the language of the reviewer. A similar effect is possible for the product manufacturer attribute. Even within a given product category such as `books`, the genre of the book might affect the language of the reviews. Additional examples of multi–attribute datasets include restaurant reviews (cuisine, location, price, and other such attributes defining domains) (Chahuneau et al., 2012), congressional floor debate records (political party, speaker, bill defining domains) (Joshi et al., 2012), and online chat room discussions on cancer support websites (speaker, facilitator, room, time defining domains) (Mayfield et al., 2012). We present experiments on three such datasets, each of which has multiple metadata attributes that can potentially influence the classification task of interest.

It is difficult to apply multi–domain learning algorithms when it is unclear which metadata attribute to choose for defining domains. It may be the case that there is a "best" attribute to use for learning, one that when used in single–attribute multi–domain learning will yield the best classifier. To find this attribute, one must rely on one's intuition about the problem,[3] or perform an exhaustive empirical search over all attributes using some validation set. Both these strategies can be brittle, because as the nature of data changes over time so may the "best" domain distinction. Additionally, single–attribute

[2] Distributions and functions that share attributes could share parameters.

[3] Intuition is often critical for learning and in some cases can help, such as in the Amazon product reviews data set, where product type clearly corresponds to domain. However, for other data sets the choice is far less clear.

multi–domain learning cannot benefit from multiple helpful attributes that may be complementary or compensating toward each other with respect to the differences in feature behavior corresponding to their different values (**DIFF I** in § 2.1). For example, when considering restaurant reviews, while it may be the case that the cuisine of the restaurant affects the customer expectations (and therefore the language in the reviews) in one way, its location might further reinforce those expectations, or compensate for them. Concretely, the expectations from a seafood restaurant in the state of Kansas might be very different compared to the expectations from a seafood restaurant in any of the states nearer to the coast (such as California or Florida). The latter will likely be judged much more stringently than the former.

We note here that Eisenstein et al. (2011) worked with a "multifaceted topic model" using their framework of sparse additive generative (SAGE) models. Also, Wang et al. (2012) proposed a special–case extension of SAGE to model a collection of historical legal documents to consider the influence of geographical regions and time, in addition to the influence of latent topics. Both models capture interactions between topics and multiple aspects, and can be adapted to the case of multi–attribute multi–domain learning. While our problem formulation has significant conceptual overlap with the SAGE–like multi-faceted topic models framework, our proposed methods are motivated from a fast online learning perspective.

A naive approach for multi–attribute multi–domain learning would be to treat every unique set of attributes (including unique subsets of attributes to account for the case of missing attributes in some instances) as a domain.[4] However, that introduces an exponential number of domains and requires a similar increase in training data, clearly an infeasible requirement. Instead, we develop multi–attribute extensions for two multi–domain learning algorithms, such that the increase in parameters is linear in the number of metadata attributes, and no special handling is required for the case where some metadata attributes might be missing from an instance.

5.2 Multi–Attribute FEDA (MFEDA)

As discussed in Chapter 2, the key idea behind FEDA (Daumé III, 2007) is to encode each domain using its own parameters, one per feature. FEDA maps a feature vector $\mathbf{x}$ in $\mathbb{R}^D$ to $\mathbb{R}^{D(K+1)}$, where K is the number of domains created by a single metadata attribute.

[4]Note that while we used a similar setup for formulating our problem, we did not rule out the potential for factoring the distributions.

This provides a separate parameter sub-space for every domain $k \in 1 \ldots K$, and also maintains a domain-agnostic shared sub-space. Essentially, each feature is duplicated for every instance in the appropriate sub-space of $\mathbb{R}^{D(K+1)}$ that corresponds to the instance's domain as defined by the value of the metadata attribute used for creating domains.

We extend this idea to multi–attribute multi–domain learning by using one parameter sub-space *per attribute value*. The original instance $\mathbf{x} \in \mathbb{R}^D$ is now mapped into $\mathbb{R}^{D(1+\sum_m K_m)}$, where K_m represents the distinct set of possible values for the m-th metadata attribute. This leads to a separate parameter sub-space for each attribute value, in addition to a shared set of parameters as before. In effect, for every metadata attribute $a \in \mathcal{A}_i$, the original features for an instance are copied into the appropriate sub-space. This grows linearly with the number of metadata attribute values, as opposed to exponentially in our naive solution. While this is still substantial growth, each instance retains the same feature sparsity as in the original input space. Furthermore, memory efficiencies can be attained through feature hashing (Ganchev and Dredze, 2008, Weinberger et al., 2009) if required. In this new setup, MFEDA allows an instance to contribute towards learning the shared parameters, and the attribute–specific parameters for all the attributes present on an instance. Just like multi–domain FEDA, any supervised learning algorithm can be applied to the transformed representation created by MFEDA.

5.3 Multi–Attribute MDR (MMDR)

We make a similar change to MDR (Dredze et al., 2009) to extend it for the multi–attribute setting. In the original formulation, Dredze et al. used confidence-weighted (CW) learning (Dredze et al., 2008) for learning shared and domain–specific classifiers, which are combined based on the confidence scores associated with the feature weights.

In our multi–attribute setup confidence-weighted (CW) classifiers (overview in § 2.4) are learned for each of the $\sum_m K_m$ attribute values in addition to a shared CW classifier. At classification time, a combined classifier is computed for every instance. However, instead of combining the shared classifier and a *single* domain–specific classifier, we combine the shared CW classifier and $|\mathcal{A}_i|$ different domain–specific CW classifiers associated with $\mathbf{x}_i$.

When learning the shared and domain-specific classifiers, we follow the best result in Dredze et al. (2009) and use the "averaged update" strategy (§7.3 in Dredze et al. (2009); also, §3.1.3), where updates are computed for the combined classifier, and are then distributed to the underlying shared and domain–specific classifiers.

5.3.1 Updating Underlying Classifiers

In §3.1.3 (Chapter 3), we described in detail the different update strategies used by Dredze et al. (2009). In particular, we discussed the "averaged update" strategy for distributing the updates for the combined classifier to the underlying shared and domain–specific classifiers, and its two variants: **AVG I**, and **AVG II**. Recall that **AVG I** distributes the combined classifier updates *equally* to the underlying classifiers, and **AVG II** distributes them *inversely proportional to the feature variance*, using the **b**-weighting scheme in equation 3.7.[5] The idea behind **AVG II** updates is to give a lower portion of the update to the underlying classifier that has higher variance (or less confidence), since its contribution to the combined classifier is lower.

However, the **AVG II** updates conflict with the original intuition behind confidence–weighted classifiers — that features with higher variance (lower confidence) should receive higher updates; they are more in need of change. Therefore, we implemented a modified update scheme **AVG III**, where the updates are distributed to the underlying classifiers such that *higher variance features receive larger updates*. We refer to this as `MDR-KL-NV` for the single–attribute case, and `MMDR-KL-NV` for the multi–attribute case. The equation for computing the proportion of the combined updates given to each of the underlying classifiers in this case is as follows:

$$\mathbf{b}_j^m = \boldsymbol{\sigma}_j^m \tag{5.1}$$

Thus, in **AVG III**, each of the underlying classifiers receives updates directly proportional to the feature variance, similar to the original confidence–weighted classifier.

We compared **AVG III** to the **AVG II** update strategy (called `MDR-KL-V` and `MMDR-KL-V` in our results), and found significant improvements using our modified scheme. We also compared it with the **AVG I** update scheme (uniform, or equal updates, denoted `MDR-KL-U` and `MMDR-KL-U` in our results), and found **AVG III** to be better overall.

[5] Note that the **b**-weighting scheme was originally introduced for classifier combination, not for updates. However, as mentioned before, it is applicable for distributing updates in the averaged update setting.

5.4 Datasets

To evaluate our multi–attribute multi–domain learning algorithms we consider three different datasets. First, we use the restaurant reviews dataset introduced by Chahuneau et al. (2012). Second, we use the full version of the United States Congressional Floor Debates dataset (CONVOTE) introduced in Chapter 3 (Thomas et al., 2006). Third, we use an online chat room discussion dataset from a cancer support forum (Mayfield et al., 2012). We now describe each of them in detail.

5.4.1 Restaurant Reviews Dataset (WORDSALAD)

We use two subsets of the restaurant reviews dataset introduced by Chahuneau et al. (2012). The full dataset consists of 1,180,308 reviews. The classification task is that of labeling the reviews as either positive or negative. Following the approach of Blitzer et al. (2007), scores above and below 3-stars indicated positive and negative reviews, while 3-star reviews were discarded.

The first subset of the restaurant reviews dataset that we use (`50K-RND`) consists of a set of randomly chosen 50,000 reviews while the second subset (`50K-BAL`) is a class–balanced sample, consisting of 25K positive and 25K negative reviews.

Each restaurant in our dataset can have many metadata attributes, including a unique identifier, name (which may not be unique), address (we extract the zip code), type (Italian, Chinese, etc.) among others. We select the 20 most common metadata attributes for our experiments (excluding latitude, longitude, and the average rating).[6] Metadata attributes such as the identifier and the name of the restaurant can influence the language of the reviews due to the reputation that restaurants build over time. Geographical location (zip code, neighborhood, city) might influence the language in the reviews because of variation in language that is commonly observed across different geographical regions (Chambers and Trudgill, 1998), and even in informal language in social media such as Twitter[7] (Eisenstein et al., 2010). Other metadata attributes such as the price range, the alcohol policy, kid–friendliness, type or category of the restaurant and so on can influence customer expectations, which in turn can be reflected in the language used in the reviews.

[6] Our method requires categorical metadata attributes, although real-valued attributes can be discretized.
[7] http://www.twitter.com/

5.4.2 Congressional Votes Dataset (CONVOTE)

We have described this dataset in detail in Chapter 3 (§3.2). We provide a quick summary here. This dataset consists of transcribed speech segments from the United States Congressional floor debates. The classification task is that of predicting whether a speech segment expresses support or opposition for the corresponding bill. The metadata attributes that we use on this dataset are the political party affiliation of the speaker (democrat, independent, or republican) and the speaker identifier itself. The ideology of the political party to which a speaker belongs can influence the language used in floor debates over bills. A similar influence of the speaker's personal ideology is also possible. Hence both of these attributes are reasonable candidates for defining domains.

In contrast to our experiments in Chapter 3, we use the full CONVOTE dataset for our experiments in the current chapter.

5.4.3 Cancer Chat Dataset (STRESS)

This dataset consists of multi–party online chat discussions among cancer patients. The discussions are facilitated by a professional therapist. Our dataset consists of a total of 1,459 chat sessions or discussions. The complete set of utterances from a given user in a given session are concatenated together to create a single input instance. Every user joining the chat session is expected to log an *entry stress level* on a scale of 0 to 10. In general, not all users enter this data, but our dataset only consists of those users who provided this input. A useful task to consider on this dataset is that of modeling the stress levels of users over time, with the expectation that the trajectory of stress levels should decrease over time for participants who find the online support group to be useful. However, as mentioned before, not all users provide the entry and exit stress levels when they participate in the sessions. Therefore, the classification task that we consider on this data is that of predicting whether the entry stress level of the user for a given chat session is *above the mean stress level* for that user in our dataset. This can be considered as a first step toward the goal of eventually modeling the stress trajectory per user. In order to have a robust estimate of the mean stress per user, we only consider users that are a part of at least 10 sessions in our data. The metadata attributes that we consider for this dataset include the user identifier (ID), the facilitator identifier, the chat room identifier, and the month of the year. The user ID is naturally a big factor in identifying the user's stress level since it provides strong prior information about the user, based on available

training data. Also, speaking styles vary across different people, and therefore having a personal identifier as a domain–defining metadata attribute is useful. The facilitator ID can similarly influence the language in the chat room discussions, and in turn, also the utterances from other users in response to the facilitator. The room ID determines the overall language of the particular support group of which the user is a part. And finally, the month of the year could influence the external environment of the users, and therefore their language. For example, it may be the case that the time period around holidays is more or less stressful for people, depending on their personality.

5.5 Methods

In addition to our multi–attribute algorithms, we evaluate several baselines. All methods use confidence-weighted (CW) learning (Crammer et al., 2012).

BASE A single classifier trained on all the data, and which ignores metadata attributes and uses unigram features. For CW, we use the best-performing setting from Dredze et al. (2008) — the "variance" algorithm, which computes approximate but closed–form updates, which also lead to faster learning. Parameters are tuned over a validation set within each training fold.

META Identical to BASE with a unique bias feature added for each attribute value (Joshi et al., 2012).

1-META This is a special case of META where a unique bias feature is added *only for a single attribute* (which is also used for defining domains for single–attribute multi–domain learning).

To use existing single–attribute multi–domain learning approaches directly, we could select a single attribute as the domain. We consider several strategies for picking this attribute and evaluate both FEDA and MDR in this setting.

1-MEAN Choose an attribute randomly. This will give us the expected error over all attributes if an attribute was chosen at random to define the domains. This can be seen as an empirical lower bound on the performance of our multi–attribute multi–domain learning methods.

1-TUNE Select the best performing attribute on a validation set. This is the approach that would be used in practice if the definition of domains is restricted to using a single metadata attribute.

1-ORCL Select the best performing attribute on the *test* set. While obviously impossible in practice, this gives the oracle upper bound on single–attribute multi–domain learning.

All our experiments use ten-fold cross-validation. We report the mean accuracy, along with standard error.

5.6 Results

We report the results for each of the three datasets in two parts. For each dataset, first we show the full set of results for the case where only a single metadata attribute is used to define domains. These are compared to the BASE, META, and 1-META baseline approaches. Second, we show the results of our multi–attribute multi–domain learning techniques, and compare them to the BASE, and META approaches, and to each of the three strategies for choosing a single "best" metadata attribute (1-MEAN, 1-TUNE, and 1-ORCL).

We restrict the main results to only using the KL–divergence approach for combining underlying classifiers in MDR and MMDR. In almost all cases this was a better approach than using an L_2 combination. For reference, we have included the full set of results using the L_2 combination in Appendix A (TODO).

5.6.1 WordSalad

Tables 5.1 and 5.2 show the performance of various methods that use only a single metadata attribute (except META) for the WordSalad dataset. In particular, it includes the following single–attribute multi–domain learning algorithms: FEDA, and three variants of MDR. The key comparison in these result tables is that of the multi–domain learning approaches with the two baseline approaches: BASE (which does not use any metadata attributes) and META (which uses all metadata attributes by including them as regular features). The 1-META approach is included to demonstrate that no single metadata attribute (used as a regular feature) outperforms the META baseline.

From both set of results, it is clear that several of the attributes can provide a useful definition of domains for multi–domain learning. Several attributes including ID, NAME, CATEGORY, ZIPCODE, and NEIGHBORHOOD show a significant improvement over the META baseline, which is the simplest way to include all possible metadata information in a model. This suggests that multi–domain learning methods that can use multiple attributes at once for defining domains might be desirable.

While many attributes provide significant improvements over BASE, only a few of

metadata	1-META	FEDA	MDR-KL-U	MDR-KL-V	MDR-KL-NV
NONE (BASE)	92.29 (±0.14)				
ALL (META)	† 92.69 (±0.10)				
ALCOHOL	† 92.46 (±0.09)	† **92.79 (±0.13)**	† 92.76 (±0.11)	90.98 (±0.17)	† 92.64 (±0.16)
ACCEPTSCARDS	† **92.42 (±0.12)**	92.42 (±0.11)	92.13 (±0.09)	92.02 (±0.15)	91.99 (±0.12)
PARKING	92.36 (±0.13)	92.50 (±0.09)	† **92.78 (±0.15)**	92.25 (±0.15)	92.49 (±0.14)
CATEGORY	† 92.48 (±0.11)	92.47 (±0.10)	†‡ 92.99 (±0.12)	91.16 (±0.16)	†‡ **93.24 (±0.13)**
CITY	† 92.45 (±0.11)	92.53 (±0.09)	† **92.76 (±0.13)**	91.06 (±0.13)	† 92.70 (±0.14)
GOODFORKIDS	† 92.41 (±0.11)	**92.49 (±0.13)**	92.45 (±0.07)	91.14 (±0.17)	92.33 (±0.07)
GOODFORMEAL	92.40 (±0.12)	† 92.59 (±0.11)	†‡ 92.99 (±0.13)	91.39 (±0.15)	†‡ **93.15 (±0.10)**
ID	† 92.52 (±0.12)	†‡ 92.87 (±0.12)	†‡ 93.05 (±0.13)	91.01 (±0.18)	†‡ **93.12 (±0.13)**
NAME	† 92.55 (±0.10)	†‡ 92.91 (±0.13)	†‡ 93.04 (±0.12)	91.10 (±0.17)	†‡ **93.19 (±0.12)**
NEIGHBORHOOD	92.42 (±0.11)	† 92.65 (±0.13)	†‡ 93.02 (±0.13)	91.17 (±0.21)	†‡ **93.21 (±0.12)**
OUTDOOR	92.41 (±0.11)	† **92.55 (±0.14)**	92.43 (±0.13)	91.40 (±0.16)	92.22 (±0.16)
ATTIRE	92.38 (±0.13)	**92.48 (±0.15)**	92.21 (±0.14)	91.92 (±0.09)	92.27 (±0.14)
DELIVERY	† 92.50 (±0.08)	† **92.63 (±0.10)**	92.31 (±0.10)	91.46 (±0.18)	92.28 (±0.11)
GOODFORGROUPS	92.40 (±0.10)	† **92.53 (±0.15)**	92.25 (±0.14)	91.61 (±0.12)	92.20 (±0.16)
PRICERANGE	† 92.43 (±0.10)	† **92.62 (±0.11)**	92.56 (±0.16)	91.11 (±0.15)	† 92.56 (±0.12)
RESERVATIONS	92.36 (±0.09)	† **92.61 (±0.12)**	† 92.57 (±0.14)	91.08 (±0.19)	92.36 (±0.17)
TABLESERVICE	92.32 (±0.11)	† **92.69 (±0.08)**	92.36 (±0.12)	91.73 (±0.12)	92.33 (±0.07)
TAKEOUT	92.38 (±0.12)	† **92.58 (±0.12)**	92.19 (±0.14)	91.60 (±0.09)	92.10 (±0.12)
ACCESSIBLE	92.42 (±0.10)	**92.44 (±0.12)**	92.06 (±0.16)	91.83 (±0.12)	92.00 (±0.13)
ZIPCODE	92.40 (±0.09)	† 92.73 (±0.09)	†‡ 92.99 (±0.12)	91.19 (±0.20)	†‡ **93.22 (±0.11)**

Table 5.1: 50K-RND: Average accuracy (± standard error) when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

them are informative enough to provide an improvement over the META baseline. Most of the significant improvements over META are observed for the MDR variants. In particular, our modification to MDR (MDR-KL-NV) works the best in every single case where a significant improvement over the META baseline is observed. Comparing across the two tables (5.1 and 5.2), the improvements due to our new variation of MDR (MDR-KL-NV) are more consistent than the previous variance–driven version (MDR-KL-V). In particular, the older variant MDR-KL-V does not significantly outperform META on the 50K-BAL dataset when using ID, NAME, CATEGORY, NEIGHBORHOOD, and ZIPCODE attributes for defining domains.

We next demonstrate multi–attribute improvements over the multi–domain baselines (Tables 5.3 and 5.4). The tables compare multi–attribute FEDA and multi–attribute MDR to three possible strategies of choosing a single–best metadata attribute to use with existing multi–domain learning techniques. Recall that 1-ORCL is an "oracle" baseline that assumes we know the best single metadata attribute for each test fold in our ten-fold cross-validation setup. 1-TUNE picks the best metadata attribute to use for every fold based on a validation set taken from each of the training folds. Finally, 1-MEAN is the mean expected performance over several experimental runs if we randomly pick one of the 20 metadata attributes for each test fold. We compute this by simply taking the mean

metadata	1-META	FEDA	MDR-KL-U	MDR-KL-V	MDR-KL-NV
NONE (BASE)	89.95 (±0.10)				
ALL (META)	† 90.39 (±0.09)				
ALCOHOL	89.85 (±0.12)	† **90.57 (±0.13)**	† 90.19 (±0.08)	87.44 (±0.23)	† 90.21 (±0.14)
ACCEPTSCARDS	89.99 (±0.11)	**90.14 (±0.07)**	89.12 (±0.16)	88.87 (±0.16)	88.95 (±0.13)
PARKING	89.99 (±0.10)	**90.16 (±0.08)**	89.92 (±0.12)	89.18 (±0.16)	89.85 (±0.16)
CATEGORY	90.09 (±0.11)	† 90.50 (±0.11)	† 90.60 (±0.11)	87.89 (±0.13)	†‡ **91.33 (±0.08)**
CITY	89.91 (±0.15)	† 90.28 (±0.08)	† 90.22 (±0.09)	87.78 (±0.13)	† **90.28 (±0.11)**
GOODFORKIDS	89.91 (±0.12)	† **90.26 (±0.06)**	89.74 (±0.09)	88.34 (±0.13)	89.67 (±0.10)
GOODFORMEAL	89.97 (±0.14)	† 90.43 (±0.10)	†‡ 90.78 (±0.12)	88.24 (±0.13)	†‡ **90.84 (±0.10)**
ID	† 90.42 (±0.11)	†‡ 90.64 (±0.11)	† 90.50 (±0.11)	87.78 (±0.25)	†‡ **91.27 (±0.09)**
NAME	† 90.28 (±0.13)	†‡ 90.63 (±0.08)	† 90.58 (±0.10)	87.88 (±0.24)	†‡ **91.25 (±0.10)**
NEIGHBORHOOD	90.13 (±0.12)	† 90.59 (±0.13)	† 90.52 (±0.09)	87.84 (±0.13)	†‡ **91.26 (±0.08)**
OUTDOOR	89.99 (±0.12)	**90.13 (±0.09)**	89.52 (±0.08)	88.16 (±0.11)	89.40 (±0.12)
ATTIRE	89.94 (±0.11)	**90.14 (±0.08)**	89.20 (±0.11)	88.78 (±0.15)	89.10 (±0.14)
DELIVERY	89.96 (±0.13)	† **90.21 (±0.08)**	89.64 (±0.09)	88.39 (±0.10)	89.71 (±0.12)
GOODFORGROUPS	90.01 (±0.09)	**90.11 (±0.06)**	89.36 (±0.09)	88.53 (±0.16)	89.18 (±0.10)
PRICERANGE	89.93 (±0.11)	† **90.42 (±0.05)**	90.13 (±0.13)	87.92 (±0.10)	90.04 (±0.17)
RESERVATIONS	89.90 (±0.13)	† **90.27 (±0.14)**	89.90 (±0.07)	88.06 (±0.12)	89.91 (±0.10)
TABLESERVICE	89.97 (±0.10)	**90.06 (±0.09)**	89.47 (±0.11)	88.72 (±0.10)	89.42 (±0.12)
TAKEOUT	89.98 (±0.12)	**90.06 (±0.09)**	89.28 (±0.13)	88.73 (±0.11)	89.14 (±0.10)
ACCESSIBLE	**89.96 (±0.11)**	**89.96 (±0.06)**	89.19 (±0.10)	88.71 (±0.16)	89.05 (±0.13)
ZIPCODE	89.97 (±0.12)	† 90.42 (±0.13)	† 90.56 (±0.09)	87.78 (±0.16)	†‡ **91.30 (±0.10)**

Table 5.2: 50K-BAL: Average accuracy (± standard error) when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

of the results in Tables 5.1 and 5.2, for reporting the 1-MEAN numbers in Tables 5.3 and 5.4 respectively.

Our extensions that can use all metadata attributes simultaneously are consistently better than both the 1-MEAN and the 1-TUNE strategies, except for the MMDR-KL-V variant. The MMDR-KL-V suffers a degradation in performance when using multiple metadata attributes simultaneously. Our new variant MMDR-KL-NV does not degrade, and in fact, is the best–performing method for 50K-RND. The best–performing method is MFEDA for 50K-BAL, and MMDR-KL-NV is not significantly different from it.

For the skewed subset 50K-RND, MFEDA is significantly better than the corresponding 1-TUNE version; MMDR-KL-U is significantly better than the corresponding 1-TUNE version; and MMDR-KL-NV is not significantly different from the corresponding 1-TUNE version. For the balanced subset 50K-BAL, a similar pattern holds, except that MMDR-KL-NV *is* significantly better than the corresponding 1-TUNEversion. Thus, our multi–attribute algorithms provide a benefit over existing approaches in most cases. Also, often their performance is better than or equal to the corresponding 1-ORCL strategy, which (impractically) assumes that the "best" single attribute for any test set is known beforehand.

Although MMDR-KL-NV is not significantly better than the corresponding 1-TUNE

#attributes	MFEDA	MMDR-KL-U	MMDR-KL-V	MMDR-KL-NV
NONE (BASE)	92.29 (±0.14)			
ALL (META)	† 92.69 (±0.10)			
ALL	†‡ 93.07 (±0.19)	†‡ 93.12 (±0.11)	87.08 (±1.72)	†‡ **93.19 (±0.12)**
1-ORCL	†‡ 93.06 (±0.11)	†‡ 93.17 (±0.11)	92.37 (±0.11)	†‡ **93.39 (±0.12)**
1-TUNE	† 92.64 (±0.12)	† 92.81 (±0.16)	92.15 (±0.17)	†‡ **93.07 (±0.14)**
1-MEAN	† **92.61 (±0.09)**	† 92.59 (±0.10)	91.41 (±0.12)	† 92.58 (±0.10)

Table 5.3: 50K-RND: Average accuracy (± standard error) using 10-fold cross-validation for methods that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table 5.1.

#attributes	MFEDA	MMDR-KL-U	MMDR-KL-V	MMDR-KL-NV
NONE (BASE)	89.95 (±0.10)			
ALL (META)	† 90.39 (±0.09)			
ALL	†‡ **91.42 (±0.09)**	†‡ 91.06 (±0.04)	81.43 (±2.79)	†‡ 91.40 (±0.08)
1-ORCL	†‡ 90.89 (±0.10)	†‡ 90.87 (±0.11)	89.33 (±0.13)	†‡ **91.45 (±0.07)**
1-TUNE	† 90.33 (±0.10)	†‡ 90.70 (±0.14)	89.13 (±0.16)	†‡ **91.26 (±0.08)**
1-MEAN	† **90.30 (±0.06)**	89.92 (±0.07)	88.25 (±0.07)	90.06 (±0.08)

Table 5.4: 50K-BAL: Average accuracy (± standard error) using 10-fold cross-validation for methods that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table 5.1.

strategy on the 50K-RND set, we found that for every single fold in our ten–fold cross–validation experiments, the "best" single metadata attribute decided using a validation set did not match the best–performing single metadata attribute on the corresponding test set. This shows the potential instability of choosing a single best attribute, even using a validation set. Note also that MMDR-KL-NV is a variant that we have proposed in the current work, and in fact for the earlier variant of MDR (MMDR-KL-U, which uniformly distributes updates to underlying classifiers), as well as for MFEDA, we do see significant improvements when using all metadata attributes.

5.6.2 Convote

Table 5.5 shows the results of single–attribute multi–domain learning methods for the Convote dataset. The first observation to be made on this dataset is that neither the Party, nor the Speaker attribute, when used by themselves as domains for multi–domain learning, achieve significant improvement over the META baseline, which uses both these attributes as features. This is in contrast with the results on the WordSalad dataset,

metadata	1-META	FEDA	MDR-KL-U	MDR-KL-V	MDR-KL-NV
NONE (BASE)	67.08 (±1.74)				
ALL (META)	† 82.60 (±1.95)				
PARTY	† 78.81 (±1.47)	† **84.19 (±2.44)**	† 83.23 (±2.48)	† 81.38 (±2.22)	† 83.92 (±2.31)
SPEAKER	† 77.49 (±1.75)	† **82.88 (±2.43)**	† 78.32 (±1.91)	62.43 (±2.20)	† 72.26 (±1.37)

Table 5.5: CONVOTE: Average accuracy (± standard error) when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

#attributes	MFEDA	MMDR-KL-U	MMDR-KL-V	MMDR-KL-NV
NONE (BASE)	67.08 (±1.74)			
ALL (META)	† 82.60 (±1.95)			
ALL	†‡ 85.71 (±2.74)	† 84.12 (±2.56)	50.44 (±1.78)	†‡ **86.19 (±2.49)**
1-ORCL	† **84.77 (±2.47)**	† 83.88 (±2.27)	† 81.38 (±2.22)	† 83.92 (±2.31)
1-TUNE	† **84.19 (±2.44)**	† 83.23 (±2.48)	† 81.38 (±2.22)	† 83.92 (±2.31)
1-MEAN	† **83.53 (±2.40)**	† 80.77 (±1.92)	† 71.91 (±1.82)	† 78.09 (±1.69)

Table 5.6: CONVOTE: Average accuracy (± standard error) using 10-fold cross-validation for methods that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table 5.5.

where some attributes by themselves showed an improvement over the META baseline. Thus, this dataset represents a more challenging setup for our multi–attribute multi–domain learning methods — they need to exploit the two relatively weaker attributes simultaneously.

Confirming the trend reported in Joshi et al. (2012), the PARTY attribute is more beneficial for defining domains in a single–attribute multi–domain learning setup. The best results are observed using the FEDA approach for both attributes. MDR-KL-NV is not significantly different from FEDA for the PARTY attribute, but significantly worse for the SPEAKER attribute.

Similar to the trend seen on the WORDSALAD dataset, MDR-KL-NV is better than its older variant MDR-KL-V. It is however not better than MDR-KL-U for the SPEAKER attribute. In general, the MDR variants do not perform very well with the SPEAKER attribute. One hypothesis to explain this trend could be that the number of distinct domain values for the SPEAKER attribute is fairly large (378), and therefore noisy as compared to the number of distinct domain values for the PARTY attribute (3). However, that hypothesis breaks down on attributes such as ID and NAME on the WORDSALAD dataset, where the MDR variants perform well.

Table 5.6 shows the results of our multi–attribute multi–domain learning methods,

comparing them to the three baseline strategies for choosing a single "best" attribute for defining domains. As mentioned earlier, this dataset is a challenging setting for our methods due to the fact that no single attribute is strong enough to yield improvements over the `META` baseline. In this setting, both `MFEDA` and `MMDR-KL-NV` achieve a significant improvement over the `META` baseline, with `MMDR-KL-NV` being the best (though not significantly better than `MFEDA`). Additionally, both of them are significantly better than their corresponding `1-TUNE` strategies. This result further supports our claim that using multiple attributes in combination for defining domains (even when any single one of them is not particularly beneficial for multi–domain learning) is important.

5.6.3 Stress

Table 5.7 shows the results of the single–attribute multi–domain learning methods on the Stress dataset. This dataset has a different characteristic from the other two — a single attribute (UserId) is in fact the best way for defining domains, as none of the other attributes (FacId, RoomId, or Month) provide any value as domains — they do not improve performance even over the `BASE` approach which ignores domains. Thus, this is a challenging setting for multi–attribute multi–domain learning in a different way — there are many irrelevant attributes which will be used for defining domains. This tests the noise tolerance of our methods.

Among the single–attribute multi–domain learning methods that use the UserId attribute to define domains, `FEDA` gives the best result. Two variants of `MDR` (`MDR-KL-U` and `MDR-KL-NV`) also perform significantly better than both `BASE` and `META`. However, they are significantly worse than `FEDA`.

Table 5.8 shows the results of our proposed multi–attribute multi–domain learning methods, comparing them to the three strategies of picking a single "best" attribute for defining domains. Again, both `MFEDA` and `MMDR-KL-NV` significantly improve performance over `BASE`, and `META` approaches, `MFEDA` being numerically better. `MFEDA` is not significantly better than its corresponding `1-TUNE` strategy. `MMDR-KL-NV` is marginally significantly better ($p < 0.10$) as compared to its corresponding `1-TUNE` strategy. However, the `1-TUNE` accuracy using `MDR-KL-NV` is lower than the `1-TUNE` accuracy using `FEDA` to begin with. The key positive result on this dataset is that despite having several irrelevant attributes that are used to define domains, and only a single good attribute (UserId), the multi–attribute multi–domain learning methods still improve performance, though not significantly over the `1-TUNE` strategy. This demonstrates the noise tolerance

metadata	1-META	FEDA	MDR-KL-U	MDR-KL-V	MDR-KL-NV
NONE (**BASE**)	63.44 (±0.58)				
ALL (**META**)	63.61 (±0.62)				
UserId	63.14 (±0.53)	†‡ **68.39 (±0.59)**	†‡ 65.56 (±0.60)	61.57 (±0.43)	†‡ 66.75 (±0.40)
FacId	63.02 (±0.57)	**63.78 (±0.50)**	62.97 (±0.68)	59.31 (±1.86)	63.37 (±0.76)
RoomId	63.06 (±0.48)	64.39 (±0.43)	63.41 (±0.77)	61.29 (±0.66)	**64.53 (±0.56)**
Month	**63.81 (±0.58)**	62.13 (±0.69)	63.33 (±0.43)	60.62 (±1.43)	63.30 (±0.68)

Table 5.7: Stress: Average accuracy (± standard error) when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

#attributes	MFEDA	MMDR-KL-U	MMDR-KL-V	MMDR-KL-NV
NONE (**BASE**)	63.44 (±0.58)			
ALL (**META**)	63.61 (±0.62)			
ALL	†‡ **68.17 (±0.68)**	65.26 (±0.74)	54.82 (±0.92)	†‡ 67.36 (±0.70)
1-ORCL	†‡ **68.47 (±0.58)**	†‡ 65.60 (±0.60)	62.77 (±0.50)	†‡ 66.75 (±0.40)
1-TUNE	†‡ **67.28 (±1.06)**	† 65.01 (±0.43)	60.03 (±1.39)	†‡ 66.40 (±0.43)
1-MEAN	**64.67 (±0.39)**	63.81 (±0.56)	60.69 (±0.91)	64.49 (±0.52)

Table 5.8: Stress: Average accuracy (± standard error) using 10-fold cross-validation for methods that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table 5.7.

of our proposed multi–attribute extensions.

5.6.4 Computation Time

Table 5.9 shows the total training and model–tuning time (in minutes) for the key methods used in our experiments. These experiments were run on a cluster consisting of 8–core processors, each processor having a 32GB RAM. Larger memory is required for the FEDA and MDR variants, since they work with a much larger parameter space than the BASE or META approaches.

As seen from the table, in almost all cases the multi–attribute multi–domain learning techniques take *less* time than the corresponding 1-TUNE strategy, which has to evaluate every metadata attribute on some validation set. The times for the 1-TUNE versions in the table are actually optimistic estimates since they do not include the time to re–train a model using the selected single "best" attribute. We do realize that the 1-TUNE strategy is "embarrassingly parallel" in the number of metadata attributes.[8] We are however reporting the combined computation time in Table 5.9, and not the experimental turnaround

[8] It is also possible to parallelize the MMDR variants in similar ways, although some parts of the algorithm cannot be parallelized.

Method	WordSalad		Convote	Stress
	50K-RND	50K-BAL		
BASE	1.67	1.61	0.25	1.08
META	1.98	1.93	0.25	1.11
1-TUNE-FEDA	70.98	65.00	0.92	10.13
MFEDA	87.26	81.22	0.76	9.62
1-TUNE-MDR-KL-NV	1315.02	1234.78	13.70	124.82
MMDR-KL-NV	673.98	591.85	10.03	65.10

Table 5.9: Total training and model–tuning time (in minutes) for different multi–domain learning methods, on each of the three datasets (WordSalad, Convote, and Stress).

time in a parallelized setting.

The exception to the pattern of multi–attribute methods being faster are the timing numbers on the WordSalad dataset for the MFEDA approach. The large number of metadata attributes seems to be the key factor in reducing MFEDA's efficiency in that case. However, the time difference is not very large, and certainly worth the effort since MFEDA is always better than 1-TUNE-FEDA.

It should be noted, however, that the timing for FEDA *does not* include the time required to replicate the features for a given instance in the appropriate feature sub–space(s). The feature replication was done as a preprocessing step. However, in practice, to avoid this preprocessing step, the feature replication could be done "on the fly" in relatively negligible time.

Thus, in general, the multi–attribute multi–domain learning variants that we propose are not only better in terms of accuracy, but also provide a computational benefit as compared to the 1-TUNE strategy.

5.7 Summary

This chapter introduced the problem of multi–attribute multi–domain learning. The problem arises when datasets contain multiple metadata attributes, each of which is a reasonable candidate for defining domains for multi–domain learning. Using existing multi–domain learning techniques with such datasets requires that a single "best" attribute be chosen for defining domains. This may not be ideal, since the classification task of interest might in fact be meaningfully influenced by multiple attributes. We therefore proposed extensions to two existing multi–domain learning techniques to handle this scenario where an instance belongs to multiple domains simultaneously. Using our proposed methods,

the definition of domains does not have to be restricted to a single metadata attribute.

We compared our methods to traditional multi–domain learning methods that are tuned to use a single "best" attribute based on a validation set (the `1-TUNE` strategy). Our methods achieve better performance on three different datasets (WordSalad, Convote, and Stress) that contain multiple domain–defining attributes. Additionally, we demonstrated that our methods are noise–tolerant, and therefore not influenced by the presence of irrelevant domain attributes. Finally, they are also computationally faster in terms of the total training and model–tuning time required as compared to the `1-TUNE` strategy.

6 | Conclusions and Future Work

A core assumption of many machine learning approaches — that data points are independent and identically distributed — has been challenged in recent years. One of the ways this assumption is violated in practice is that training data points are often *not* identically distributed. This thesis makes contributions toward addressing such scenarios, where data points in the training data can be grouped together to form subgroups, which are called *domains*.

Several advances have been made in recent years to address this problem, variously called multi–domain learning, multi–task learning, or domain adaptation, depending on the specific scenario being addressed. In this thesis we focus on the multi–domain learning scenario where the training data points belong to several domains, and the test data points are also drawn from the same set of domains which are known at training time. An example of a task where this scenario manifests is classification of user reviews as `positive` or `negative`, where the reviews belong to a fixed set of product categories which serve as domains (Dredze and Crammer, 2008).

We consider the critical question of how domains are defined for a given dataset, and the influence that the defined domains and their properties have on the performance of multi–domain learning methods.

We will first summarize the main contributions of this thesis, and then suggest some open research questions that can further advance the work presented in this thesis.

6.1 Contributions

This section summarizes the main ideas and results presented in this thesis, and ties them to the key contributions of this thesis.

6.1.1 Empirical Analysis

While much prior research has been done on new methods for multi–domain learning, the question of what defines "domains" for a dataset (especially in empirical experimentation that evaluates the proposed multi–domain learning methods) has not been systematically addressed.

Toward establishing the importance of defining the *right* domains, this thesis first presented a systematic empirical analysis (Chapter 3, and also Joshi et al. (2012)) of some of the representative multi–domain learning approaches from prior work. An important goal of the empirical analysis was to understand if the multi–domain learning methods do indeed capitalize on the domain information in meaningful ways. Two main questions were explored.

First, a number of existing multi–domain learning methods involve a classifier combination, an idea that is at the core of many ensemble learning methods. Thus, connecting this aspect to the question of domain definitions, we explored the effect of randomly defining domains for multi–domain learning methods. With randomized domains, multi–domain learning methods are turned into variants of ensemble learning. In such situations, we found that it is still possible to get an improvement from multi–domain learning methods, even though the domain information is largely erroneous. This showed that part of the improvements due to multi–domain learning can be attributed to an ensemble learning effect.

Second, we evaluated multi–domain learning on class–imbalanced datasets — a typical scenario in real–world use of machine learning. We found that if there are strong domain–specific class biases, that is, if data points from a given domain exhibit a different class distribution as compared to other domains, then multi–domain learning largely capitalized on that aspect. Moreover, a simple method to capture such domain–specific class biases often gives significant improvements. Furthermore, the ensemble learning effect becomes more prominent in the presence of domain–specific class biases.

Both these findings have important methodological consequences for the evaluation of multi–domain learning approaches. The suggested evaluation methodology is one of the key contributions of this thesis. The ensemble learning result provides a method to evaluate whether the effect of the defined domains is significant beyond a mere combination of different classifiers trained on subsets of the training data. The experiments related to domain–specific class bias provide a simple baseline to be used when deploying multi–domain learning systems in real–world situations where class imbalance across domains

is common.

6.1.2 Feature Representation for Multi–Domain Learning

While there has been some prior work showing the importance of appropriate feature representation for domain adaptation (Ben-David et al., 2007), in this thesis we show for the first time that it is possible to design a feature representation that generalizes across several domains, even if the domains can be defined in more than one way.

In Chapter 4 (also Joshi and Rosé (2009)), we proposed a novel feature representation in the form of a generalization of syntactic dependency features. By backing off the head word in a syntactic dependency to its part–of–speech tag, we showed that one can obtain "composite" features. The composite features retain some lexical specificity, but also allow for generalization across products and product types on the task of opinion mining (identifying opinionated sentences) on product reviews.

Furthermore, we also performed a systematic analysis to show that the generalization ability of our proposed features indeed capitalizes on the presence of multiple domains in training data — the key element in multi–domain learning.

This finding is another key contribution of this thesis, as it suggests a potentially complementary approach to multi–domain learning, which can be combined with some of the algorithmic approaches proposed in prior and current work.

6.1.3 Multi–Attribute Multi–Domain Learning

All of the multi–domain learning research in the past relies on a definition of domains that partitions the training set in a unique way. In practice, this often means using a single metadata attribute to define domains. However, there are many cases where multiple metadata attributes are reasonable candidates for defining domains. In such cases, it is not clear which single attribute to use for deciding the domains. Also, more importantly, it is possible that many metadata attributes *simultaneously* influence the problem of interest. In such cases, choosing only one of them to define domains might lead to sub–optimal performance.

With this scenario in mind, we proposed the paradigm of multi–attribute multi–domain learning (Chapter 5, and also Joshi et al. (2013)). An instance in multi–domain learning belongs to only one domain. In our new paradigm, an instance can belong to *multiple domains simultaneously*, corresponding to all the domain–defining metadata attributes associated with the instance. A naive way of translating this scenario into the regular

multi–domain learning setup is to consider every possible combination of the domain–defining metadata attributes as a domain by itself. However, this leads to an exponential number of domains, and sparsity issues for learning domain–specific parameters.

Instead, we proposed extensions of two well–known multi–domain learning techniques (Daumé III, 2007, Dredze and Crammer, 2008) such that the increase in the number of parameters is only linear in the number of domain–defining metadata attributes. We showed that our extensions can handle several kinds of multi–attribute domains in different datasets. They perform well in cases where there are multiple strong domain–defining attributes (WordSalad dataset), multiple weak domain–defining attributes (Convote dataset), and one strong and multiple irrelevant domain–defining attributes (Stress dataset). We also showed that our methods are more accurate, as well as computationally more efficient than a strategy that forces the choice of a single "best" definition of domains via tuning on some validation set.

The paradigm of multi–attribute multi–domain learning is another significant contribution of this thesis. The extensions of two existing multi–domain learning techniques in this paradigm provide the current state–of–the–art results in this setting.

6.2 Future Work

While this thesis has made significant contributions toward understanding the various facets of domain definitions for multi–domain learning, there are many other open challenging questions that still remain. We now suggest possible future directions for pursuing those open questions.

6.2.1 Connections to Ensemble Learning

We showed empirically that certain multi–domain learning methods have a flavor of ensemble learning, by virtue of the classifier combination techniques that they use. While there has been some theoretical analysis of learning from multiple domains (Ben-David et al., 2009, Mansour et al., 2009), our results suggest that a theoretical analysis of the connections between multi–domain learning and ensemble learning can be useful for further progress in terms of understanding the definition of domains.

6.2.2 Unsupervised Learning of Bias

Another result from this thesis is that domain–specific class biases significantly influence multi–domain learning. An interesting question therefore is whether such biases can be learned for new domains based on only unlabeled data? This is related to the problem of detecting a shift in class bias for test data (Smith and Elkan, 2007). However, the difference is that the training data points themselves are divided into domains that have a varying class bias.

6.2.3 Feature Representation

The problem–driven feature representation approach presented in this thesis was evaluated in isolation from any of the multi–domain learning algorithms. Thus, a natural extension is to verify if the feature representation and algorithmic techniques are complementary to each another and have a greater combined benefit. A more interesting question, however, is whether it is possible to automatically *discover* the appropriate feature representation for a problem, given the full set of raw features and domain definitions. This is especially important for large–scale machine learning where the raw features might be several million or higher in number, and therefore, exploring the more promising subset of combinations of feature–domain pairs efficiently is critical.

6.2.4 Discovering Domains

In the presence of multiple metadata attributes that can serve the purpose of defining domains, one option is to learn a mapping from the metadata attributes to a unique set of domains, which can be used with traditional multi–domain learning methods. This can either be done as a separate preprocessing stage, or it could be combined as a joint task while simultaneously learning domain–specific behaviors. This idea is appealing for large–scale datasets, since the number of unique values for the full set of metadata attributes can become pretty large. Similar to the technique used by Dredze et al. (2009, pp. 141–144) for clustering a large number of domains, a set of domains can be discovered based on *multiple metadata attributes*, instead of clustering based on a single metadata attribute as in their case. Another potential approach to consider for this problem is that of logistic model trees (Landwehr et al., 2005), where the idea would be to learn a decision tree over the metadata attributes such that the leaves represent the learned domains, each of which has a domain–specific classifier (which uses features other than the metadata

features) that is learned simultaneously while learning the tree.

In the absence of any metadata attributes that can serve as domains, it is still possible to consider the problem of discovering domains. Although not presented in the context of multi–domain learning or domain adaptation, the work on latent class analysis for classification (Vermunt and Magidson, 2003) is very relevant for this scenario. However, a key question to address when discovering domains using latent class analysis is again that of their definition and desired properties. For example, Hoffman et al. (2012) use a constrained clustering approach to discover latent domains in image data. Using an iterative algorithm, they first discover local domain clusters within each class (semantic category of the image), and then merge the local clusters to create global domain clusters such that only one local domain cluster from each class is allowed to be a part of a given global domain cluster. This strategy forces the discovered domain clusters to be distinct from the classes. However, depending on the amount of data available, it may not be desirable to perform a local clustering step within each class. Thus, alternative ways of enforcing constraints on the discovered latent domains are worth exploring.

Appendices

A | Additional Multi–Attribute Multi–Domain Learning Results

For completeness, we have included in this chapter the results of the different MDR-L2 and MMDR-L2 variants on the three datasets (WordSalad, Convote, and Stress). In almost all cases, the improvements are not significant over the BASE and META approaches. Details for each dataset are included in the three sections below.

A.1 WordSalad

Tables A.1 and A.2 show the results of using the MDR-L2 and MMDR-L2 variants respectively for the 50K-RND dataset. Almost none of the results here are significantly better than the BASE approach. None are significantly better than the META baseline. A comparison to the corresponding main result tables 5.1 and 5.3 shows consistently lower performance by MDR-L2 and MMDR-L2 variants as compared to the MDR-KL and MMDR-KL variants.

Tables A.3 and A.4 show the results of using the MDR-L2 and MMDR-L2 variants respectively for the 50K-BAL dataset. The trends are similar to the ones seen on the 50K-RND version of the WordSalad dataset.

A.2 Convote

Tables A.5 and A.6 show the results of the MDR-L2 and MMDR-L2 variants respectively on the Convote dataset. While a number of settings achieve improvement over the BASE approach, none of them improves over the META baseline. All the MDR-L2 and MMDR-L2 variants, except MDR-L2-NV in Table A.5 for the Speaker attribute, are worse than the corresponding MDR-KL and MMDR-KL variants.

metadata	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (BASE)	92.29 (±0.14)		
ALL (META)	† 92.69 (±0.10)		
ALCOHOL	**91.23 (±0.19)**	73.25 (±3.24)	90.99 (±0.24)
ACCEPTSCARDS	90.69 (±0.15)	90.07 (±0.34)	**90.85 (±0.13)**
PARKING	**90.85 (±0.17)**	88.22 (±0.63)	90.79 (±0.11)
CATEGORY	91.48 (±0.15)	62.76 (±2.88)	**92.37 (±0.09)**
CITY	**91.32 (±0.16)**	66.46 (±0.67)	91.27 (±0.14)
GOODFORKIDS	90.88 (±0.16)	83.55 (±2.51)	**91.00 (±0.14)**
GOODFORMEAL	91.20 (±0.13)	66.74 (±0.58)	**91.26 (±0.16)**
ID	**92.01 (±0.15)**	66.65 (±0.80)	90.56 (±0.28)
NAME	**91.85 (±0.16)**	65.64 (±0.90)	90.89 (±0.29)
NEIGHBORHOOD	92.25 (±0.13)	63.78 (±2.37)	† **92.63 (±0.14)**
OUTDOOR	**91.02 (±0.13)**	85.19 (±2.30)	90.88 (±0.17)
ATTIRE	**90.76 (±0.14)**	88.99 (±0.35)	90.45 (±0.21)
DELIVERY	**90.77 (±0.22)**	88.57 (±0.47)	90.70 (±0.26)
GOODFORGROUPS	**91.05 (±0.17)**	85.85 (±2.21)	90.87 (±0.18)
PRICERANGE	91.15 (±0.16)	73.34 (±2.89)	**91.21 (±0.19)**
RESERVATIONS	90.89 (±0.14)	79.25 (±3.50)	**91.02 (±0.17)**
TABLESERVICE	**90.94 (±0.13)**	88.13 (±0.41)	90.91 (±0.16)
TAKEOUT	**90.62 (±0.19)**	88.31 (±0.39)	90.35 (±0.37)
ACCESSIBLE	**90.80 (±0.15)**	86.79 (±2.14)	90.78 (±0.16)
ZIPCODE	92.11 (±0.14)	63.43 (±2.69)	**92.56 (±0.12)**

Table A.1: 50K-RND: Average accuracy (± standard error) for MDR-L2 variants when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

A.3 STRESS

Tables A.7 and A.8 show the results of the MDR-L2 and MMDR-L2 variants respectively on the STRESS dataset. Again, almost none of the variants improve over the BASE or the META approaches. Although MMDR-L2-NV is significantly better than both BASE and META for the 1-ORCL setting, it is not better than the best numbers in the main results reported in Tables 5.7 and 5.8, and is also a practically infeasible scenario (knowing the best single metadata attribute for the *test set*).

#attributes	MMDR-L2-U	MMDR-L2-V	MMDR-L2-NV
BASE	92.29 (±0.14)		
META	† 92.69 (±0.10)		
ALL	**84.55 (±0.38)**	63.08 (±4.54)	61.59 (±0.28)
1-ORCL	92.41 (±0.11)	90.34 (±0.34)	† **92.71 (±0.12)**
1-TUNE	91.92 (±0.19)	84.52 (±2.85)	† **92.63 (±0.12)**
1-MEAN	**91.19 (±0.11)**	77.75 (±0.58)	91.12 (±0.11)

Table A.2: 50K-RND: Average accuracy (± standard error) using 10-fold cross-validation for MMDR-L2 variants that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table A.1.

metadata	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (**BASE**)	89.95 (±0.10)		
ALL (**META**)	† 90.39 (±0.09)		
alcohol	88.19 (±0.17)	56.60 (±0.43)	**88.24 (±0.18)**
AcceptsCards	**87.40 (±0.18)**	85.72 (±0.22)	87.15 (±0.27)
parking	87.21 (±0.27)	82.00 (±2.86)	**87.30 (±0.20)**
category	88.94 (±0.18)	52.21 (±0.24)	**89.95 (±0.11)**
city	**88.53 (±0.08)**	56.15 (±0.80)	88.32 (±0.14)
GoodForKids	87.77 (±0.12)	74.13 (±4.13)	**87.87 (±0.10)**
GoodForMeal	88.48 (±0.17)	54.72 (±0.90)	**88.64 (±0.09)**
id	88.92 (±0.16)	54.05 (±0.40)	† **90.28 (±0.12)**
name	88.87 (±0.25)	55.53 (±0.47)	† **90.36 (±0.12)**
neighborhood	89.18 (±0.14)	52.99 (±0.30)	† **90.27 (±0.14)**
outdoor	**87.84 (±0.14)**	80.30 (±2.70)	87.72 (±0.16)
attire	**87.69 (±0.15)**	85.09 (±0.25)	87.25 (±0.24)
delivery	**87.80 (±0.15)**	80.41 (±2.66)	87.73 (±0.08)
GoodForGroups	**87.60 (±0.23)**	83.61 (±0.45)	86.99 (±0.34)
PriceRange	88.07 (±0.14)	59.48 (±2.54)	**88.09 (±0.13)**
reservations	**87.84 (±0.16)**	75.98 (±3.69)	87.67 (±0.13)
TableService	**87.74 (±0.21)**	83.82 (±0.30)	87.57 (±0.12)
TakeOut	**87.63 (±0.19)**	84.59 (±0.29)	87.43 (±0.17)
accessible	**87.59 (±0.18)**	85.04 (±0.19)	87.37 (±0.21)
zipcode	89.23 (±0.14)	54.01 (±0.42)	† **90.41 (±0.15)**

Table A.3: 50K-BAL: Average accuracy (± standard error) for MDR-L2 variants when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

#attributes	MMDR-L2-U	MMDR-L2-V	MMDR-L2-NV
NONE (BASE)	89.95 (±0.10)		
ALL (META)	† 90.39 (±0.09)		
ALL	**81.23 (±0.66)**	50.07 (±0.08)	56.77 (±0.17)
1-ORCL	89.44 (±0.13)	86.13 (±0.15)	† **90.57 (±0.13)**
1-TUNE	89.07 (±0.10)	85.14 (±0.35)	† **90.31 (±0.13)**
1-MEAN	88.13 (±0.08)	69.82 (±0.45)	**88.33 (±0.07)**

Table A.4: 50K-BAL: Average accuracy (± standard error) using 10-fold cross-validation for MMDR-L2 variants that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table A.3.

metadata	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (BASE)	67.08 (±1.74)		
ALL (META)	† 82.60 (±1.95)		
Party	† **75.61 (±2.79)**	59.17 (±2.79)	† 75.14 (±2.10)
Speaker	† 78.30 (±1.59)	54.59 (±0.62)	† **80.39 (±2.01)**

Table A.5: Convote: Average accuracy (± standard error) for MDR-L2 variants when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

#attributes	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (BASE)	67.08 (±1.74)		
ALL (META)	† 82.60 (±1.95)		
ALL	† **78.49 (±2.33)**	49.56 (±3.56)	68.22 (±2.17)
1-ORCL	† 79.77 (±1.93)	61.31 (±1.15)	† **80.60 (±1.98)**
1-TUNE	† **77.83 (±1.73)**	58.26 (±2.87)	† 77.43 (±2.17)
1-MEAN	† 76.95 (±2.09)	56.88 (±1.26)	† **77.77 (±1.93)**

Table A.6: Convote: Average accuracy (± standard error) using 10-fold cross-validation for MMDR-L2 variants that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table A.5.

metadata	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (BASE)	63.44 (±0.58)		
ALL (META)	63.61 (±0.62)		
USERID	59.53 (±1.60)	54.84 (±0.74)	† **65.65 (±0.64)**
FACID	60.40 (±0.53)	57.72 (±1.28)	**60.79 (±1.24)**
ROOMID	60.29 (±1.13)	53.98 (±0.93)	**61.89 (±0.71)**
MONTH	59.51 (±0.93)	54.17 (±0.86)	**61.01 (±0.46)**

Table A.7: STRESS: Average accuracy (± standard error) for MDR-L2 variants when using a single attribute at a time. Results that are *numerically the best* within a row are in **bold**. Results significantly better than BASE are marked with †, and better than META are marked with ‡. Significance is measured using a two-tailed paired t-test with $\alpha = 0.05$.

#attributes	MDR-L2-U	MDR-L2-V	MDR-L2-NV
NONE (BASE)	63.44 (±0.58)		
ALL (META)	63.61 (±0.62)		
ALL	56.85 (±0.77)	52.31 (±0.75)	**58.56 (±0.51)**
1-ORCL	63.17 (±0.85)	58.56 (±1.07)	†‡ **65.77 (±0.60)**
1-TUNE	60.20 (±1.48)	54.42 (±1.28)	**65.09 (±0.74)**
1-MEAN	59.93 (±0.65)	55.18 (±0.70)	**62.33 (±0.56)**

Table A.8: STRESS: Average accuracy (± standard error) using 10-fold cross-validation for MMDR-L2 variants that use all attributes, either directly (our proposed methods) or for selecting the "best" single attribute using one of the strategies described earlier. Formatting and significance symbols are the same as in Table A.7.

Bibliography

Rie Kubota Ando and Tong Zhang. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, 6, 2005. 21

Andrew Arnold, Ramesh Nallapati, and William W Cohen. Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition. In *Proceedings of ACL-08: HLT*, pages 245–253, 2008. 11, 14, 15, 51

Shilpa Arora, Mahesh Joshi, and Carolyn Rosé. Identifying Types of Claims in Online Customer Reviews. In *Proceedings of NAACL 2009*, 2009. 60

Shilpa Arora, Elijah Mayfield, Carolyn Rosé, and Eric Nyberg. Sentiment Classification using Automatically Extracted Subgraph Features. In *Proceedings of the workshop on Emotion in Text at NAACL 2010*, 2010. 22

Jonathan Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000. 16

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Proceedings of NIPS 2006*, 2007. 21, 22, 27, 87

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 2009. 27, 36, 67, 88

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics, 2006. 9, 12, 16, 21, 27

John Blitzer, Mark Dredze, and Fernando Pereira. 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. As-

sociation for Computational Linguistics, 2007. 2, 3, 12, 21, 25, 27, 34, 69, 73

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems*, 2008. 36

Rich Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, 1993. 3, 16

Rich Caruana. Multitask Learning. *Machine Learning*, 28, 1997. 9, 16

Giovanni Cavallanti, Nicolò. Cesa-Bianchi, and Claudio Gentile. Linear Algorithms for Online Multitask Classification. In *Proceedings of COLT*, 2008. 20

Victor Chahuneau, Kevin Gimpel, Bryan R. Routledge, Lily Scherlis, and Noah A. Smith. Word Salad: Relating Food Prices and Descriptions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP 2012)*, 2012. 69, 73

Jack K. Chambers and Peter Trudgill. *Dialectology*. Cambridge University Press, 1998. 73

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 59

Ciprian Chelba and Alex Acero. 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. Association for Computational Linguistics, 2004. 9, 11, 12, 14

Stanley F. Chen and Ronald Rosenfeld. A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999. 12

William Cohen. Minorthird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data, 2004. URL http://minorthird.sourceforge.nets. 59

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7, 2006. 18

Koby Crammer, Mark Dredze, and Fernando Pereira. Confidence-Weighted Linear Classification for Text Categorization. *Journal of Machine Learning Research (JMLR)*, 2012. 75

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, pages 210–219, 2007. 12

Wenyuan Dai, Ou Jin, Gui-Rong Xue, Qiang Yang, and Yong Yu. EigenTransfer: A Unified Framework for Transfer Learning. In *International Conference on Machine Learning*, 2009. 11

Hal Daumé III. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics, 2007. 11, 13, 15, 27, 68, 70, 88

Hal Daumé III. Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009. 20, 37

Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *J. Artif. Int. Res.*, 26(1):101–126, 2006. 9, 11, 13

Kushal Dave, Steve Lawrence, and David Pennock. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *Proceedings of WWW 2003*, 2003. 54

Marie-Catherine de Marneffe, Bill Maccartney, and Christopher Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC 2006*, 2006. 55

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1007607513941. 37

Xiaowen Ding, Bing Liu, and Philip S. Yu. A holistic lexicon-based appraoch to opinion mining. In *Proceedings of First ACM International Conference on Web Search and Data Mining (WSDM-2008)*, 2008. 59

Mark Dredze. *Intelligent Email: Aiding Users with AI*. PhD thesis, University of Pennsylvania, 2009. 51

Mark Dredze and Koby Crammer. Online methods for multi-domain learning and adaptation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, 2008. 2, 3, 9, 12, 17, 18, 19, 20, 29, 30, 32, 37, 85, 88

Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-Weighted Linear Clas-

sification. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008. 18, 29, 71, 75

Mark Dredze, Alex Kulesza, and Koby Crammer. Multi-domain Learning by Confidence-Weighted Parameter Combination. *Machine Learning*, 79(1-2), 2009. 2, 30, 31, 33, 34, 35, 38, 60, 68, 71, 72, 89

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. A Latent Variable Model for Geographic Lexical Variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, 2010. 73

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. Sparse Additive Generative Models of Text. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011. 70

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-rui Wang, and Chih-Jen Lin. LIBLINEAR : A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 36

Jenny R Finkel and Christopher D Manning. 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. Association for Computational Linguistics, 2009. 11, 13, 14, 15

Michael Gamon. Sentiment Classification on Customer Feedback Data: Noisy Data, Large Feature Vectors, and the Role of Linguistic Analysis. In *Proceedings of COLING 2004*, 2004. 52, 53, 54, 57, 58

Kuzman Ganchev and Mark Dredze. Small Statistical Models by Random Feature Mixing. In *Proceedings of the ACL08 HLT Workshop on Mobile Language Processing*, pages 19–20, 2008. 71

Philip Gianfortoni, David Adamson, and Carolyn P. Rosé. Modeling of Stylistic Variation in Social Media with Stretchy Patterns. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pages 49–59, 2011. 22, 51

Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING)*, pages 539–545, 1992. 52, 53

Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering Latent Domains

for Multisource Domain Adaptation, 2012. 90

Minqing Hu and Bing Liu. Mining and Summarizing Customer Reviews. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2004. 52, 59, 60, 62

Jing Jiang and ChengXiang Zhai. Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics, 2007. 12

Mahesh Joshi and Carolyn Penstein Rosé. Generalizing Dependency Features for Opinion Mining. In *Proceedings of the joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL–IJCNLP)*, pages 313–316, 2009. 22, 51, 52, 87

Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn P. Rosé. Multi-Domain Learning: When Do Domains Matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1302–1312, 2012. 26, 69, 75, 80, 86

Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn P. Rosé. What's in a Domain? Multi-Domain Learning for Multi-Attribute Data. In *Proceedings of NAACL HLT 2013*, page (to appear), 2013. 68, 87

Niels Landwehr, Mark Hall, and Eibe Frank. Logistic Model Trees. *Machine Learning*, 59 (1-2):161–205, 2005. 89

Xiao Ling, Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Spectral domain-transfer learning. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08*, pages 488–496, 2008. 12

Richard Maclin and David Opitz. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. 37

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain Adaptation with Multiple Sources. In *Proceedings of NIPS 2008*, pages 1041–1048. MIT Press, 2009. 88

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees. In *Proceedings of the 9th PAKDD*, 2005. 53

Elijah Mayfield, David Adamson, and Carolyn P. Rosé. Hierarchical Conversation Struc-

ture Prediction in Multi-Party Chat. In *Proceedings of the SIGDIAL 2012 Conference, The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 60–69, 2012. 69, 73

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. Structured Models for Fine-to-Coarse Sentiment Analysis. In *Proceedings of ACL 2007*, 2007. 22, 53, 58

Vincent Ng, Sajib Dasgupta, and S. M. Niaz Arifin. Examining the Role of Linguistic Knowledge Sources in the Automatic Identification and Classification of Reviews. In *Proceedings of the COLING/ACL 2006*, 2006. 54

Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Conference on Natural Language Learning (Shared Task)*, 2007. 37

Avishek Saha, Piyush Rai, Hal Daumé III, and Suresh Venkatasubramanian. Online learning of multiple tasks and their relationships. In *Proceedings of AISTATS 2011*, 2011. 20, 37

Hidetoshi Shimodaira. Improving Predictive Inference Under Covariate Shift by Weighting the Log-likelihood Function. *Journal of Statistical Planning and Inference*, 90(2), 2000. 10

Andrew Smith and Charles Elkan. Making Generative Classifiers Robust to Selection Bias. In *Proceedings of KDD*, pages 657–666, 2007. 89

Gerry Stahl, Timothy Koschmann, and Dan Suthers. Computer–Supported Collaborative Learning: An Historical Perspective. In R. K. Sawyer, editor, *Cambridge Handbook of the Learning Sciences*, pages 409–426. Cambridge University Press, 2006. 19

Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335, 2006. 34, 73

Peter .D Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the Association for Computational Linguistics, 40th Anniversary Meeting*, 2002. 4, 11

Jeroen K. Vermunt and Jay Magidson. Latent class models for classification. *Computational Statistics & Data Analysis*, 41(3):531–537, 2003. 90

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. Historical

Analysis of Legal Opinions with a Sparse Mixed-Effects Latent Variable Model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, 2012. 70

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 1113–1120, 2009. 71

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. Just How Mad Are You? Finding Strong and Weak Opinion Clauses. In *Proceedings of AAAI 2004*, 2004. 52, 53, 54, 57, 58

Xifeng Yan and Jiawei Han. gSpan: Graph-based Substructure Pattern Mining. In *ICDM 2002*, 2002. 22

Yu Zhang and Dit-Yan Yeung. 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)*, 2010. 20, 21, 28, 37, 38