Exploiting domain and task regularities for robust named entity recognition

Ph.D. Thesis Proposal

Andrew O. Arnold

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

October 22, 2008

Thesis Committee:

William W. Cohen (CMU), *Chair*
Tom M. Mitchell (CMU)
Noah A. Smith (CMU)
ChengXiang Zhai (UIUC)

Contents

1 Thesis

Relaxing assumptions & finding regularities: It is often convenient to make certain assumptions during the learning process. Unfortunately, algorithms built on these assumptions can often break down if the assumptions are not *stable* between train and test data. We define a property of the data to be *stable* if said property remains *relatively unchanged* across *variations* in other aspects of the data, where such properties can be attributes of the data instances themselves or relationships among different parts of the data; and the 'variations' allowed among the data and the degree to which the stable property must remain 'unchanged' is defined with respect to the degree of robustness desired. For instance, in traditional learning, given $(x, y)_{train}$ drawn from some training distribution D_{train}, and $(x, y)_{test}$ drawn from some test distribution D_{test}, we assume that $p_{train}(y \mid x) = p_{test}(y \mid x)$. If we allow $p(y \mid x)$ to vary across training and testing data (that is, if we allow $D_{train} \neq D_{test}$, as in the domain adaptation setting), a standard machine learning technique like naive Bayes may fail. In the language of this thesis, this learning technique is not *robust* to this change in the data. *Our thesis is that we can make learned classifiers and extractors more robust by using data (both labeled and unlabeled) from related domains and tasks, and by exploiting stable regularities and complex relationships between different aspects of that data.*

Exploiting rich relationships: Relatedly, we can do better at various tasks (like information extraction) by exploiting the richer relationships found in real-world complex systems. When we start working with such a system, we usually find it convenient to first abstract away to a relatively simply stated learning problem, such as: *Given an example x, predict its label y.* This type of simplifying reduction is often necessary (at the expense of richer representations incorporating more domain knowledge and auxiliary sources of information) in order to frame the learning problem in a way that is consistent with the often harsh assumptions underlying many favored learning techniques. While these assumptions may be useful in providing structure in relatively simple learning problems, when faced with complex, real-world systems, they can often prove burdensome, or fail all together, and may actually be better replaced with problem-specific structure such as regularities among features or external sources of data.

Transfer learning: By exploiting these kinds of non-conventional regularities we can more easily address problems previously unapproachable, like transfer learning. In the transfer learning setting, the distribution of data is allowed to vary between the training and test domains, that is, the i.i.d. assumption linking train and test examples is severed. Without this link between the train and test data, traditional learning is difficult. Take, for example, the problem of training an extractor to identify the sender and recipient of a letter. For our training data we are given formal business letters with their senders and recipients labeled. For testing, however, we are required to identify the sender and recipient not in business letters but in student e-mails. Whereas in the non-transfer, business to business, learning case we could exploit regularities in the tokens themselves, for instance, looking for capitalized words that do not begin a sentence, in the transfer setting, this capitalization property may no longer hold between the train and test domains, that is, it is not stable. In

light of this, we need a new relationship linking the domains together, an information path linking the training data to the test data. One possibility in this example would be to exploit the common structure of the letters themselves. Specifically, the property of recipient names being located at the start of a letter, and sender names being located at the end. This tends to be true both in formal business letters and informal e-mails and thus provides a stable regularity by which our classifier can *learn*, that is, generalize from the training data to the test data. In this way we can make use of one type of regularity (document structure) when another (the conditional distribution of capitalized names) ceases to hold.

Thus, in this thesis we try to find learning techniques that can still succeed even in situations where i.i.d. and other common assumptions are allowed to fail. Specifically, we seek out and exploit regularities in the problems we encounter and document which specific assumptions we can drop and under what circumstances and still be able to complete our learning task. We further investigate different methods for dropping, or relaxing, some of these restrictive assumptions so that we may bring more resources (from auxiliary data, to known dependencies and other regularities) to bear on the problem, thus producing both better answers to existing problems, and even being able to begin addressing problems previously unanswerable, such as those in the transfer learning setting.

2 Current State of the Art

2.1 Introduction

The desire to exploit information attained from previous effort, and not to start each new endeavor *de novo* is perhaps part of human nature, and certainly a maxim of the scientific method. Nevertheless, due to the difficulty of integrating knowledge from distinct, but related, experimental *domains* (the distribution from which the data is drawn) and *tasks* (the type of prediction desired from the learner), it is common practice in most machine learning studies to focus on training and tuning a model to a single, particular domain and task pair, or *setting*, at the expense of all others. Often, once work has completed on one setting, the researcher begins afresh on the next, carrying over only the techniques and experience learned, but often not the data itself.

Consider the task of *named entity recognition* (NER). Specifically, you are given a corpus of encyclopedia articles in which all the personal name mentions have been labeled. The standard supervised machine learning problem is to learn a classifier over this training data that will successfully label unseen test data drawn from the same distribution as the training data, where "same distribution" could mean anything from having the train and test articles written by the same author to having them written in the same language. Having successfully trained a named entity classifier on this encyclopedia data, now consider the problem of learning to classify tokens as names in instant messenger data. Clearly the problems of identifying names in encyclopedia articles and instant messages are closely related, and learning to do well on one should help your performance on the other. At the same time, however, there are serious differences between the two problems that need to be addressed. For instance, capitalization, which will certainly be a useful feature in the encyclopedia problem, may prove less informative in the instant messenger data since the rules of capitalization are followed less strictly in that domain. Thus there seems to be some need for altering the classifier learned on the first problem (called the *source domain*) to fit the specifics of the second problem (called the *target domain*). This is the problem of *domain adaptation* and is considered a type of *transfer learning*.

The intuitive solution seems to be to simply train on the target domain data. Since this training data would be drawn from the same distribution as the data you will ultimately test over, this approach avoids the transfer issue entirely. The problem with this idea is that often large amounts of labeled data are not available in the target domain. While it has been shown that even small amounts of labeled target data can greatly improve transfer results [15, 22], there has been relatively little work, however, on the case when there is no labeled target data available, that is, totally unsupervised domain adaptation. In this scenario, one way to adapt a model trained on the source domain is to make the unlabeled *target test data* available to the model during training time. Leveraging unlabeled test data during training time is called *transductive learning* and is a well studied problem in the scenario when the training data and test data come from the same domain. However, transduction is not well-studied

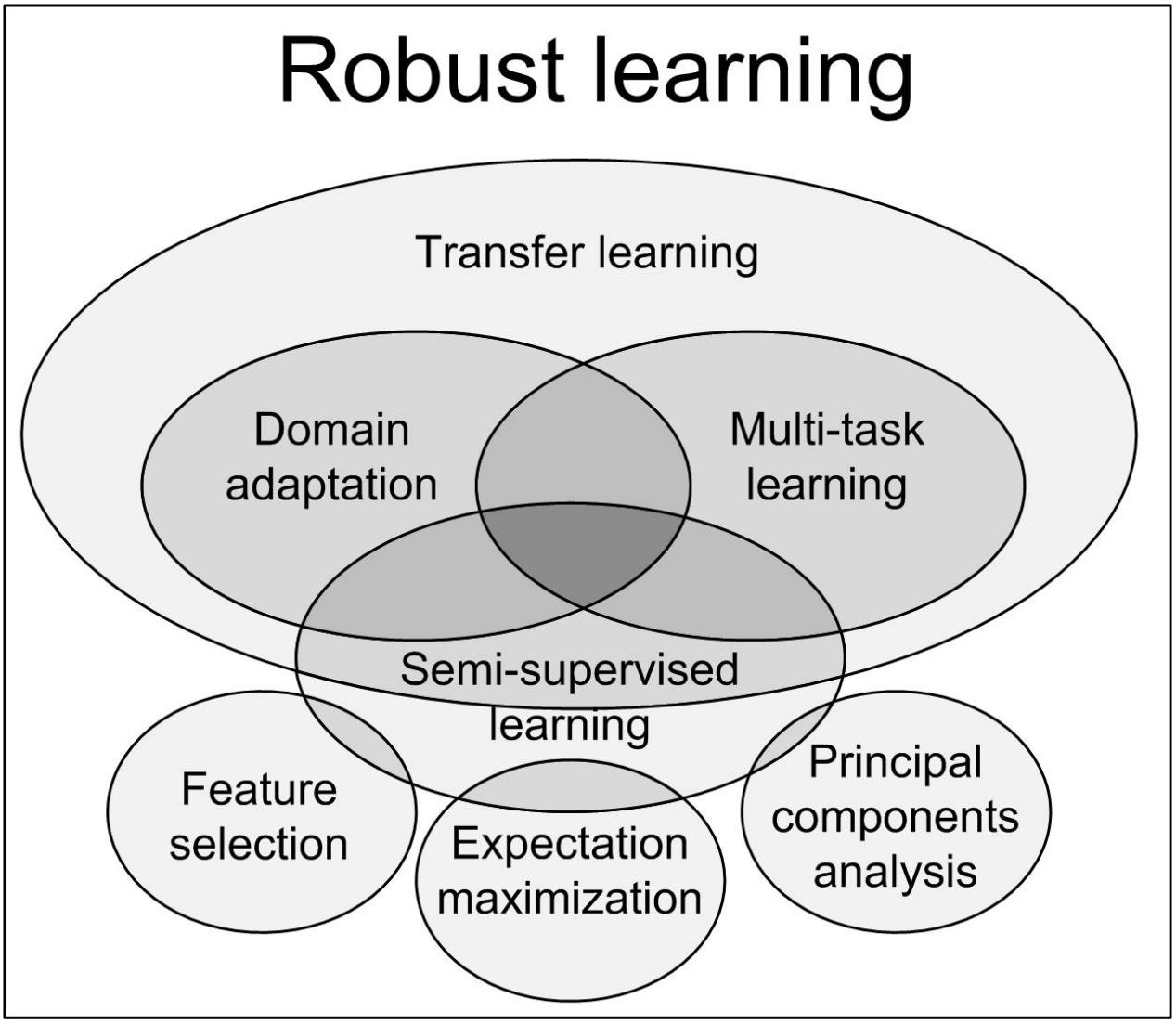

Figure 1: Venn diagram representation of the subspace of robust learning settings. Domain adaptation and multi-task learning are represented as subsets of transfer learning, which is itself a subset of all robust learning techniques. These techniques can also intersect with semi-supervised methods. A sampling of non-transfer robust learning techniques (such as sparse feature selection, expectation maximization and principal components analysis) are also included for completeness. Compare with Table 1, which structures the transfer learning sub-region into greater detail.

in a transfer setting, where the training and test data come from different domains.

In this section we explore this issue and examine existing methods to facilitate the adaptation of data from one domain (called the **source)** to problems defined on another related domain (called the **target)**. This type of problem is generally referred to as **domain adaptation** [23] in the literature and constitutes a subproblem in the broader field of **transfer learning,** which has been studied as such for at least the past ten years [6, 57]. We also introduce other forms of robust learning, both within the framework of transfer learning and without, and show how these transfer learning techniques can be combined with semi-supervised methods. Figure 1 gives a schematic overview of the way we see these techniques intersecting and overlapping with one another, while Table 1 provides a detailed breakdown of various transfer learning settings.

2.2 Problem

For the rest of this section, and for most of the proposal, we will focus on the specific problem of learning to extract protein names from articles published in biological journals. In the *named entity resolution* (NER) formalism, a document is segmented into a sequence of tokens, with each of these tokens then being classified as belonging to one of a set of possible label classes – in our case, the binary set {PROTEIN, NON_PROTEIN}. A standard technique for this kind of problem is to gather a corpus of documents drawn from the domain on which you will eventually be evaluated. These documents then need to be painstakingly hand-labeled by a domain expert in order to identify which tokens in the document represent proteins, and which do not. The 'expertise' of this domain specialist should not be underestimated, since such biological distinctions are subtle and often elude all but the most experienced annotators. The work is therefore slow, and the resulting annotated datasets are often relatively small and expensive.

We have access to such a corpus of protein-labeled abstracts from biological articles. Several techniques have been proposed for building protein-name extractors over these abstracts and their performances have been evaluated with respect to extracting new proteins from other, previously unseen abstracts drawn from a similar distribution of articles [27]. In our work, however, we are interested in identifying proteins, not in abstracts, but in the captions of papers (we use this information to create a structured search engine of images and captions from biological articles [45]). To this end we have downloaded tens of thousands of open-access, full text articles from the Internet. Unfortunately, all of these documents are wholly unlabeled and we do not have the resources to label them ourselves. Thus, our problem is: given labeled abstracts (source training domain) and unlabeled captions and full text (source auxiliary training data), how can we train a model that will extract proteins well from unseen captions (target test domain). This is at once a semi-supervised learning problem (due to the unlabeled auxiliary training data) [66], and a domain adaptation problem (due to the difference in domains from which the source and target data are drawn).

The rest of this section goes into more detail concerning the various techniques that currently

exist for robust learning, as summarized in Table 1 and Figure 1. Later, Section 3 introduces our preliminary investigations into these areas. Specifically, Section 3.4.2 introduces a key insight into the structure of documents that allows us to link the source, target, and auxiliary domains. Given this perspective, our problem, stated generally as domain adaption from the abstract (source domain) to the captions (target domain) of a paper, can be viewed more specifically as learning to transfer information from one part of a structured document to another, allowing us to overcome the 'domain-brittleness' of the commonly used lexical features, described in Section 3.4.1.

Sections 3.4.3 and 3.5 introduce three new techniques that leverage this structure to produce models that are able to exploit the unlabeled auxiliary data while at the same time being robust to shifts between train and test domains. Section 3.3 explains the particulars of the data and experiments we used to validate these new techniques, while Section 3.6.8 offers a summary of these preliminary works, anticipating our proposed works in Section 4.

2.3 Transfer learning

The phrase transfer learning covers several different subproblems. When only the type of data being examined is allowed to vary (from news articles to e-mails, for example), the transfer problem is called *domain adaptation* [23]. When the task being learned varies (say, from identifying person names to identifying protein names), the transfer problem is called *multi-task learning* [14]. Both of these are considered specific types of the over-arching *transfer learning* problem, and both seem to require a way of altering the classifier learned on the first problem (called the *source domain*, or *source task*) to fit the specifics of the second problem (called the *target domain*, or *target task*).

More formally, given an example x and a class label y, the standard statistical classification task is to assign a probability, $p(y|x)$, to x of belonging to class y. In the binary classification case the labels are $Y \in \{0, 1\}$. In the case we examine, each example x_i is represented as a vector of binary features $(f_1(x_i), \cdots, f_F(x_i))$ where F is the number of features. The data consists of two disjoint subsets: the training set $(X_{train}, Y_{train}) = \{(x_1, y_1) \cdots, (x_N, y_N)\}$, available to the model for its training and the test set $X_{test} = (x_1, \cdots, x_M)$, upon which we want to use our trained classifier to make predictions.

In the paradigm of *inductive learning*, (X_{train}, Y_{train}) are known, while both X_{test} and Y_{test} are completely hidden during training time. In this cases X_{test} and X_{train} are both assumed to have been drawn from the same distribution, $\mathcal{D}$. In the setting of *transfer learning*, however, we would like to apply our trained classifier to examples drawn from a distribution different from the one upon which it was trained. We therefore assume there are two different distributions, $\mathcal{D}^{source}$ and $\mathcal{D}^{target}$, from which data may be drawn. Given this notation we can then precisely state the transfer learning problem as trying to assign labels Y_{test}^{target} to test data X_{test}^{target} drawn from $\mathcal{D}^{target}$, given training data $(X_{train}^{source}, Y_{train}^{source})$ drawn from $\mathcal{D}^{source}$.

In this thesis we focus on two subproblems of transfer learning:

- *domain adaptation*, where we assume Y (the set of possible labels) is the same for both D^{source} and D^{target}, while D^{source} and D^{target} themselves are allowed to vary between domains.
- *multi-task learning* [2, 14, 54, 65] in which the task (and label set) is allowed to vary from source to target.

Domain adaptation can be further distinguished by the degree of relatedness between the source and target domains. For example, in this work we group data collected in the same medium (e.g., all annotated e-mails or all annotated news articles) as belonging to the same *genre*. Although the specific boundary between domain and genre for a particular set of data is often subjective, it is nevertheless a useful distinction to draw.

One common way of addressing the transfer learning problem is to use a *prior* which, in conjunction with a probabilistic model, allows one to specify *a priori* beliefs about a distribution, thus biasing the results a learning algorithm would have produced had it only been allowed to see the training data [50]. In the example from §2.1, our belief that capitalization is less strict in instant messages than in encyclopedia articles could be encoded in a prior that biased the importance of the `capitalization` feature to be lower for instant messages than encyclopedia articles. In Section 3.2 we address the problem of how to come up with a suitable prior for transfer learning across named entity recognition problems.

2.4 Domain adaptation

Domain adaptation is distinct from other forms of transfer learning (such as multitask learning [2,14,54,65]) because we are assuming that the set of possible labels, Y, remains constant across the various domains, while allowing the distribution of X and, most importantly, $Y|X$ to change. In our setting, the labels, Y, are members of the binary set {PROTEIN, NON_PROTEIN}, while the instances, X, are the tokens of the documents themselves. Another important example of domain adaptation is *concept drift*, in which the source and target data's distributions start out identical, but drifts further and further apart from each other over time [61].

In prior work, different researchers have made different assumptions about the relationship between the source and target domain, a defining characteristic of domain adaptation. In the supervised setting, one can directly compare both the marginal and conditional distributions of the data in both domains, looking for patterns of generalizability across domains [22,23,35], as well as examining the common structure of related problems [5, 7, 9, 51]. There is likewise work that tries to quantify these inter-domain relationships in the unsupervised [4], semi-supervised [10, 32], and transductive learning settings [56]. Similarly, in the biological domain, there has been work on using semi-supervised machine learning techniques to extract protein names by combining dictionaries with large, full-text corpora [52], but without the explicit modeling of differences between data domains that we attempt in this thesis. In our work, we take advantage of the fact that the source and target domains are different sections of the same structured document and use this fact to develop features that are robust across

those different domains.

2.5 Multi-task learning

Whereas in domain adaptation the set of possible labels for our learning task, Y, is held constant between source and target data, in the *multi-task* setting this label set, or *task*, is allowed to vary between the source task and target task [2, 14, 30, 54, 65]. Expanding on the example from Section 2.1, this would be like using encyclopedia articles labeled with personal names in order to train an extractor to find place names in those same types of articles. Again, there is an obvious overlap between these two learning problems and the goal of multi-task learning is to investigate how best to characterize and exploit this similarity. More nefariously, not only are the labels themselves allowed to change, but also the intended semantics of those labels. For example, the two semantically distinct problems of labeling tokens as people or places can both be represented by the same binary labeling scheme.

Although there seems to be a clear formal distinction between domain adaptation and multi-task learning, in this work we tend to consider them in much the same way. Our thesis's goal is to find robust ways of learning using as many different sources of data as we have available. Just as the data we use can come from many related domains, so too our labels (where they are available) are allowed to refer to a number of distinct, though inter-related tasks. Thus, for much of this thesis we will use the term 'task' (or alternately, *setting*) to refer both to the distribution from which our training and test data are drawn and the set of labels which our learning is trying to predict.

2.6 Semi-supervised learning

Analogously to multi-task learning, where we try to make use of data with labels related to our source task, in the semi-supervised setting we try to make use of data with no labels at all [1, 19, 63]. Indeed, in the multi-task framework, any data for which *all* labels for *all* tasks are not available can be considered, in some sense, semi-supervised. In this way, as presented in Figure 1, we consider semi-supervised learning an extra dimension of the robust learning framework that one can combine with an existing technique by making use of what unlabeled data is available. In the supervised setting, the data is usually segmented into two disjoint subsets: the training set $(X_{train}, Y_{train}) = \{(x_1, y_1) \cdots, (x_N, y_N)\}$, which can be used for training, and the test set $X_{test} = (x_1, \cdots, x_M)$, for which labels are not available at training time. In the semi-supervised setting [66], the training data is supplemented with a set of **auxiliary** data, $X_{aux} = (x_1, \cdots, x_P)$, for which no corresponding labels are provided. When using semi-supervised techniques for transfer learning, the distribution from which this unlabeled auxiliary data is drawn is allowed to vary.

Table 1: Learning settings are summarized by the type of auxiliary and test data used. For all settings we assume $(X_{train}^{source}, Y_{train}^{source})$ is available at training time, while Y_{test} is unknown. Settings for which we have run experiments are marked in bold (c.f. Table 4).

Natural name for learning setting	Auxiliary data		Test data	
	Domain	Labels	Domain	X_{test}
Inductive learning	-	-	$\mathcal{D}^{source}$	unseen
Semi-supervised inductive learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	unseen
Transductive learning	-	-	$\mathcal{D}^{source}$	seen
Transfer learning	-	-	$\mathcal{D}^{target}$	unseen
Inductive transfer learning	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	unseen
Semi-supervised inductive transfer learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	unseen
Transductive transfer learning	-	-	$\mathcal{D}^{target}$	seen
Supervised Transductive transfer learning	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	seen
Relaxed Transductive transfer learning[1]	-	-	$\mathcal{D}^{target}$	seen
Semi-supervised transductive transfer learning	$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	seen

[1] A relaxation of transductive transfer learning in which proportions of labels in the target data is known at training time.

2.7 Non-transfer robustness

Despite recent interest in and research into the problems of transfer learning as such, the idea of robust learning itself is not a new one. Feature selection has proved a very effective means of generating robust learners, especially when regularized for sparsity, as in the case of lasso and least angles regression [25,58]; or when the features are designed to succinctly summarize the relevant information contained in a dataset, as in principal components analysis [39] and mutual information techniques [64]; or when they are engineered to be resilient to deletion [31]. Researchers have also tried engineering and selecting features themselves that they believe will be robust to noise and shifts in the data [33]. Relatedly, a whole range of expectation maximization (EM) techniques have been developed for learning in situations where not all relevant information is available [24, 29]. In this thesis we build on many these techniques, combining and extending them where necessary.

2.8 Examples of transfer learning settings & techniques

In the first two sections below (§2.8.1-2.8.2), we introduce and discuss several examples of learning across the spectrum of transfer problems. These problems vary with respect to what labels and data are available from the source and target domains at train time. They are also summarized in Table 1 for the reader's convenience. Later, we survey some popular approaches to these types of problems (§2.8.3-2.8.5), and then present some comparative results to make the algorithms' relative strengths and weaknesses more concrete (§2.8.6,

Table 4). Please see our ICDM 2007 paper for a more thorough treatment of these algorithms and their performances [4].

2.8.1 Inductive learning

In the paradigm of *inductive learning*, (X_{train}, Y_{train}) are known, while both X_{test} and Y_{test} are completely hidden during training time. In the case of *semi-supervised* inductive learning [32, 53,66], the learner is also provided with auxiliary unlabeled data $X_{auxiliary}$, that is not part of the test set. It has been noted that such auxiliary data typically helps boost the performance of the classifier significantly.

2.8.2 Transductive learning

Another setting that is closely related to semi-supervised learning is *transductive learning* [36, 38, 59], in which X_{test} (but, importantly, not Y_{test}), is known at training time. That is, the learning algorithm knows exactly which examples it will be evaluated on after training. This can be a great asset to the algorithm, allowing it to shape its decision function to match and exploit the properties seen in X_{test}. One can think of transductive learning as a special case of semi-supervised learning in which $X_{auxiliary} = X_{test}$.

In the three cases discussed above, X_{test} and X_{train} are both assumed to have been drawn from the same distribution, $\mathcal{D}$. As mentioned previously, however, we are more interested in the case where these distributions are allowed to differ, that is, the transfer learning setting. One of the first formulations of the transfer learning problem was presented over 10 years ago by Thrun [57]. More recently there has been a focus on using source data to learn various types of priors for the target data [50]. Other techniques have tried to quantify the generalizability of certain features across domains [23, 35], or tried to exploit the common structure of related problems [7, 10].

Although the case of transfer learning without access to any data drawn from $\mathcal{D}^{target}$ is not completely hopeless [35], in this thesis we choose to focus on extensions to the transfer learning setting that allow us to capture some information about D^{target}. One obvious such setting is *inductive transfer learning* where we also provide a few auxiliary labeled data $(X^{target}_{auxiliary}, Y^{target}_{auxiliary})$ from the target domain in addition to the labeled data from the source domain. Due to the presence of labeled target data, this method could also be called *supervised transfer* learning and is the most common setting used by researchers in transfer learning today.

There has also been work on *transductive transfer learning*, where there is no auxiliary labeled data in the target domain available for training, but where the unlabeled test set on the target domain X^{target}_{test} can be seen during training. Again, due to the lack of labeled target data, this setting could be considered *unsupervised transfer* learning. It is important to point out that *transductive learning* is orthogonal to *transfer learning*. That is, one can have a transductive algorithm that does or does not make the transfer learning assumption,

and vice versa. Much of the work in this thesis is inspired by the belief that, although distinct, these problems are nevertheless intimately related. More specifically, when trying to solve a transfer problem between two domains or tasks, it seems intuitive that looking at the possibly *unlabeled* data of the target domain, or another related task, during training will improve performance over ignoring this source of information.

We note that the setting of *inductive transfer learning*, in which labeled data from both source and target domains are available for training, serves as an upper-bound to the performance of a learner based on *transductive transfer learning*, in which no labeled target data is available. For similar reasons, we considered an additional artificial setting, which we call *relaxed transductive transfer learning*, in our experiments. This setting is almost equivalent to the transductive transfer setting, but the model is allowed to know the proportion of positive examples in the target domain. Although this technically violates the terms of unsupervision in transductive transfer learning, in practice estimating this single parameter over the target domain does not require nearly as much labeled target data as learning all the parameters of a fully supervised transfer model, and thus serves as a nice compromise between the two extremes of transduction and supervision. Practically, this proportion is useful to know for determining thresholds [62] and guaranteeing certain semi-supervised performance results [11].

These and a few other interesting settings are summarized in Table 1. Note that we only displayed a small subset of the many possible learning settings.

2.8.3 Naive Bayes classifier

Inductive learning: maximum likelihood estimation

Naive Bayes [43] is one of the most popular and effective generative classifiers for many text-classification tasks. Like any generative model, its decision rule is given by the posterior probability of the class y given the example x, given by $P(y|x)$, which is computed using Bayes' rule as follows:

$$P(y|x) = \frac{P(x|\theta(y))\pi(y)}{\sum_{y'} P(x|\theta(y'))\pi(y')} \tag{1}$$

where $\theta(y)$ are the class-conditional parameters and $\pi(y)$ are the prior probabilities. The naive Bayes model makes the somewhat unrealistic yet practical assumption of conditional-independence between the features of each example, given its class. That is:

$$P(x|\theta(y)) = \prod_{j=1}^{F} P(f_j(x)|\theta_j(y)) \tag{2}$$

In our case, since the features are all binary, we use the Bernoulli distribution to model each feature as follows:

$$P(x|\theta(y)) = \prod_{j=1}^{F} (\theta_j(y))^{f_j(x)} (1 - \theta_j(y))^{1-f_j(x)} \tag{3}$$

where $\theta_j(y)$ can be interpreted as the probability that the feature f_j assumes a value 1 given the class y. The Bernoulli parameters $\theta_j(y)$ and $\pi(y)$ are estimated using Maximum Likelihood training with the labeled training data $(X_{train}, Y_{train}) = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ as below:

$$\begin{aligned} \theta_j(y) &= \frac{\sum_{i=1}^{N} f_j(x_i)\delta_y(y_i) + \lambda}{\sum_{i=1}^{N} \delta_y(y_i) + 2\lambda} \\ \pi(y) &= \frac{\sum_{i=1}^{N} \delta_y(y_i)}{N} \end{aligned} \tag{4}$$

where $\delta_y(y_i) = 1$ if $y = y_i$ and 0 otherwise; and λ is the Laplace smoothing parameter, which we set to 0.05 in our experiments.

Inductive transfer learning: maximum likelihood estimation with concatenated data

In the *inductive transfer* case we concatenate the entire labeled data $(X_{train}^{source}, Y_{train}^{source})$ and $(X_{train}^{target}, Y_{train}^{target})$ to generate a single training set. Then, we learn the parameters $\theta_j(y)$ and $\pi(y)$ using the maximum likelihood estimators shown in the classic supervised case (see eqn. 4). Although more sophisticated approaches are possible, we tried this algorithm as a simple baseline.

Transductive transfer learning: source-initialized EM

In the transductive transfer case, $(X_{train}^{target}, Y_{train}^{target})$ are not available for training, but X_{test}^{target} is available at training time. Learning from unlabeled examples in the generative framework is done typically using the standard Expectation Maximization algorithm [48]. The algorithm is iterative, and consists of two steps: in the E-step corresponding to the t^{th} iteration, we compute the posterior probability of each label for all the unlabeled examples w.r.t. the old parameter values $\theta_j^{(t)}(y), \pi^{(t)}(y)$ as follows:

$$\forall_y P(y|x, \theta^{(t)}, \pi^{(t)}) = \frac{P(x|\theta^{(t)}(y))\pi^{(t)}(y)}{\sum_{y'} P(x|\theta^{(t)}(y'))\pi^{(t)}(y')} \tag{5}$$

In the M-step, we estimate the new parameters $\theta_j^{(t+1)}(y), \pi^{(t+1)}(y)$ using the posterior probabilities as follows.

$$\theta_j^{(t+1)}(y) = \frac{\sum_{i=1}^{N} f_j(x_i) P(y|x_i, \theta_j^{(t)}(y))}{\sum_{i=1}^{N} P(y|x_i, \theta_j^{(t)}(y))} \tag{6}$$

$$\pi^{(t+1)}(y) = \frac{\sum_{i=1}^{N} P(y|x_i, \theta_j^{(t)}(y))}{N} \tag{7}$$

where N is the number of unlabeled examples available during training. In our case, this is the size of the set X_{test}^{target}. The iterations are continued until the likelihood of the unlabeled data converges to a maximum value. In the completely unsupervised case of the EM algorithm, the model parameters are initialized to random values before starting the iterations.

In our case, since we have $(X_{train}^{source}, Y_{train}^{source})$ at our disposal, we first do a classic supervised training of our model using the labeled source data, and initialize the parameters to the ones learned from the source data, before we start the EM iterations. This encodes the information available from the source data into the model, while allowing the EM algorithm to discover its optimal parameters on the target domain.

Relaxed transductive transfer learning: redefining the prior

In the case when the values of the prior probability of each class in the target data is available, we simply fix $\pi(y)$ to these values and only estimate $\theta(y)$ using eqn. 6 in the M-step of the EM algorithm.

2.8.4 Maximum entropy

Entropy maximization (MaxEnt) [8, 47] is a way of modeling the conditional distribution of labels given examples. Given a set of training examples $X_{train} \equiv \{x_1, \ldots, x_N\}$, their labels $Y_{train} \equiv \{y_1, \ldots, y_N\}$, and the set of features $\mathcal{F} \equiv \{f_1, \ldots, f_F\}$, MaxEnt learns a model consisting of a set of weights corresponding to each class $\Lambda = \{\lambda_{1,y} \ldots \lambda_{F,y}\}_{y \in \{0,1\}}$ over the features so as to maximize the conditional likelihood of the training data, $\mathrm{p}(Y_{train}|X_{train})$, given the model p_Λ. In exponential parametric form, this conditional likelihood can be expressed as:

$$p_\Lambda(y_i = y|x_i) = \frac{1}{Z(x_i)} \exp(\sum_{j=1}^{F} f_j(x_i)\lambda_{j,y}) \tag{8}$$

where Z is the normalization term.

In order to avoid overfitting the training data, these λ's are often further constrained by the use of a Gaussian prior [16] with diagonal covariance, $\mathcal{N}(\mu, \sigma^2)$, which tries to maximize:

$$\sum_{j,y} log \frac{1}{\sqrt{2\pi\sigma_{j,y}^2}} exp(-\frac{(\lambda_{j,y} - \mu_{j,y})^2}{2\sigma_{j,y}^2}) \tag{9}$$

Thus the entire expression being optimized is:

$$\underset{\Lambda}{\operatorname{argmax}} \sum_{i=1}^{N} \left(\log\ p_\Lambda(y_i|x_i) - \beta \sum_{j}^{F} \frac{(\lambda_{j,i} - \mu_{j,i})^2}{2\sigma_{j,i}^2} \right) \tag{10}$$

where $\beta > 0$ is a parameter controlling the amount of regularization. Maximizing this likelihood is equivalent to constraining the joint expectations of each feature and label in the learned model, $E_\Lambda[f_j, y]$, to match the Gaussian-smoothed empirical expectations $E_{train}[f_j, y]$

as shown below:

$$E_{train}\left[f_j, y\right] = \frac{1}{N}\sum_{i}^{N}\left(f_j(x_i)\delta_y(y_i) - \frac{\lambda_{j,i}-\mu_{j,i}}{\sigma_{j,i}^2}\right) \quad (11)$$

$$E_{\Lambda}\left[f_j, y\right] = \frac{1}{N}\sum_{i}^{N} f_j(x_i)P_{\Lambda}(y|x_i) \quad (12)$$

where $\delta_y(y_i) = 1$ if $y = y_i$ and 0 otherwise. In the next few subsections, we will describe how we adapt the model to various scenarios of transfer learning.

Inductive transfer: Source trained prior models

One recently proposed method [15] for transfer learning in MaxEnt models, which we call the *Chelba* model. involves modifying Λ's regularization term. First a model of the source domain, Λ^{source}, is learned by training on $\{X_{train}^{source}, Y_{train}^{source}\}$. Then a model of the target domain is trained over a limited set of labeled target data $\left\{X_{train}^{target}, Y_{train}^{target}\right\}$, but instead of regularizing this Λ^{target} to be near zero by minimizing $\|\Lambda^{target}\|_2^2$, Λ^{target} is instead regularized towards the previously learned source values Λ^{source} by minimizing $\|\Lambda^{target} - \Lambda^{source}\|_2^2$. Thus the modified optimization problem is:

$$\underset{\Lambda^{target}}{\text{argmax}} \sum_{i=1}^{N_{train}^{target}} \log\ p_{\Lambda^{target}}(y_i|x_i) - \beta\|\Lambda^{target} - \Lambda^{source}\|_2^2 \quad (13)$$

where N_{train}^{target} is the number of labeled training examples in the target domain. It should be noted that this model requires Y_{train}^{target} in order to learn Λ^{target} and is therefore a supervised form of *inductive transfer*.

Feature space expansion

Another approach to the problem of inductive transfer learning is explored by Daumé [22, 23]. Here the idea is that there are certain features that are common between different domains, and others that are particular to one or the other. More specifically, we can redefine our feature set $\mathcal{F}$ as being composed of two distinct subsets $\mathcal{F}^{specific} \bigcup \mathcal{F}^{general}$, where the conditional distribution of the features in $\mathcal{F}^{specific}$ differ between X^{source} and X^{target}, while the features in $\mathcal{F}^{general}$ are identically distributed in the source and target. Given this assumption, there is an EM-like algorithm [23] for estimating the parameters of these distributions. There is also a simpler approach [22] of just making a duplicate copy of each feature in X^{source} and X^{target}, so whereas before you had $x_i = \langle f_1(x_i)...f_F(x_i)\rangle$, you now have

$$x_i = \langle\ f_1(x_i)^{specific}, f_1(x_i)^{general} \ldots f_F(x_i)^{specific}, f_F(x_i)^{general}\ \rangle \quad (14)$$

where $specific$ is *source* or *target* respectively, and $f_j(x_i)^{specific}$ is just a copy of $f_j(x_i)^{general}$. The idea is that by expanding the feature space in this way MaxEnt will be able to assign different weights to different versions of the same feature. If a feature is common in both domains its *general* copy will get most of the weight, while its specific copies (f^{source} and f^{target}) will get less weight, and vice versa.

Conditional random fields (instance structure)

When it comes to actually training a model, we need a learning algorithm that can integrate and balance the variety of features and disparate sources of information we are trying to exploit. We used **conditional random fields** (CRF's) [41], a generalization of the common maximum entropy model from the i.i.d. case (where each token is classified in isolation), to the sequential case (where each token's classification influences the classification of its neighbors). This attribute is especially useful in a setting such as domain adaptation, where we would like to spread high-confidence predictions made on examples resembling the source domain to lower-confidence predictions of less familiar target domain instances. Similarly, like maximum entropy models, CRF's allow great flexibility with respect to the definition of the model's features, freeing us from worrying about the relative independence of specific features.

2.8.5 Support vector machines (SVM)

Support vector machines (SVM's) [37] take a different approach to the binary classification problem. Instead of explicitly modeling the conditional distribution of the data and using these estimates to predict labels, SVMs try to model the data geometrically. Each example is represented as an F-dimensional real-valued vector of features and is then projected as a point in F-dimensional space.

The *inductive SVM* exploits the label information of the training data and fits a discriminative hyperplane between the positively and negatively labeled training examples in this space, so as to best separate the two classes. This separation is called the margin, and thus SVMs belong to the margin based approach to classification. This formulation has proven very successful as inductive SVMs currently have some of the best general performance of any popular machine learning algorithm.

Inductive SVM

Recall that in the supervised inductive transfer case, we are given the training sets $(X_{train}^{source}, Y_{train}^{source})$ and $(X_{train}^{target}, Y_{train}^{target})$. Since the SVM does not explicitly model the data distribution, we simply concatenate the source and target labeled data together and provide the entire data for training. The hope is that it will improve on an SVM trained purely on labeled source data, by re-adjusting its hyperplane based on the labeled target data. It is possible to do better than such a naive approach [1], but we used this as a reasonable baseline.

[1]For example, one could impose a higher penalty for classification errors on the target data than on the source data.

Transductive SVM

Transduction with SVMs, in contrast to probabilistic models, is quite intuitive. Whereas, in the supervised case, we tried to fit a hyperplane to best separate the labeled training data, in the transductive case, we add in unlabeled testing data which we must also separate. Since we do not know the labels of the testing data, however, we cannot perform a straight forward margin maximization, as in the supervised case. Instead, one can use an iterative algorithm [36]. Specifically, a hyperplane is trained on the labeled source data and then used to classify the unlabeled testing data. One can adjust how confident the hyperplane must be in its prediction in order to use a pseudo-label during the next phase of training (since there are no probabilities, large margin values are used as a measure of confidence). The pseudo-labeled testing data is then, in turn, incorporated in the next round of training. The idea is to iteratively adjust the hyperplane (by switching presumed pseudo-labels) until it is very confident on most of the testing points, while still performing well on the labeled training points.

Transductive SVMs were originally designed for the case where the training and test sets were drawn from the same domain. Again, since SVMs do not model the data distribution, it is not immediately obvious how one would model different distributions in the SVM algorithm. Hence in this work, we directly test the applicability of transductive SVMs to the transductive transfer setting.

2.8.6 Comparison of existing techniques

Domain

We now turn to *protein name extraction*, an interesting problem domain [34, 52, 60] in which to compare these methods within various learning settings. In this problem you are given text related to biological research (usually abstracts, captions, and full body text from biological journal articles) which is known to contain mentions of protein names. The goal is to identify which words are part of a protein name mention, and which are not. One major difficulty is that there is a large variance in how these proteins are mentioned and annotated between different authors, journals, and sub-disciplines of biology. Because of this variance it is often difficult to collect a large corpus of truly identically distributed training examples. Instead, researchers are often faced with heterogeneous sources of data, both for training and testing, thus violating one of the key assumptions of most standard machine learning algorithms. Hence the setting of transfer learning is very relevant and appropriate to this problem.

Data and evaluation

Our corpora are abstracts from biological journals coming from two sources: University of Texas, Austin (UT) [13] and Yapex [27]. Each abstract was tokenized and each token was hand-labeled as either being part of a protein name or not. We used a standard natural language toolkit [17] to compute tens of thousands of binary features on each of these tokens, encoding such information as capitalization patterns and contextual information of

Table 2: Summary of data used in experiments

Corpus name (Abbr.)	Abstracts	Tokens	% Positive
UTexas (UT)	748	216,795	6.6%
Yapex (Y)	200	60,530	15.0%
Yapex-train (YTR)	160	48,417	15.1%
Yapex-test (YTT)	40	12,113	14.5%

Table 3: Training and testing data used in the settings of Inductive learning (I), Inductive Transfer (IT), Transductive Transfer (TT) and Relaxed Transductive Transfer (RTT). Abbreviations of data sets are described in Table 2.

Setting	Source-train	Target-train	Target-test
I	-	YTR	YTT
IT	UT	YTR	YTT
TT	UT	-	Y
RTT	UT	-	Y

surrounding words.

Some summary statistics for these data are shown in Table 2. We purposely chose corpora that differed in two important dimensions: the total amount of data collected and the relative proportion of positively labeled examples in each dataset. Specifically, UT has over three times as many tokens as Yapex but has only half the proportion of positively labeled protein names. This disparity is not uncommon in the domain and could be attributed to differing ways the data sources were collected and annotated. Specifically, if the protein mention annotations in Yapex tend to be longer (that is, extend for more tokens) then the proportion of positively labeled tokens will be higher in Yapex. For all our experiments, we used the larger UT dataset as our source domain and the smaller Yapex dataset as our target. We also split the Yapex data into two parts: *Yapex-train* (YTR) consisting of 80% of the data, and *Yapex-test* (YTT), consisting of the remaining 20%.

In Table 3, we display the subsets of data used for various learning settings in our experiments. Note that the transductive methods use different testing data from the inductive methods. This choice is made deliberately to provide a chance for the classifiers in each setting to achieve their peak performance, i.e., transductive algorithms work best when there is abundance of unlabeled test data and inductive algorithms work best when there is plenty of labeled data. However, since the data is slightly different between inductive and transductive settings, one must use caution in comparing the transductive results to the inductive ones.

Because of the relatively small proportion of positive examples in both the UT and Yapex datasets, we are more interested in achieving both high precision and recall of protein name mentions instead of simply maximizing classification accuracy. Since we were dealing with binary, and not sequential classification, the definition of these measures is straightforward

Table 4: Summary of % precision (**Prec**), recall (**Rec**), and F1 for regular maximum entropy (**Basic**), prior-based regularized MaxEnt (**Regularize**), and feature expansion MaxEnt (**Expand**), inductive SVM (**ISVM**), transductive SVM (**TSVM**), Maximum Likelihood Naive Bayes (**NB-ML**), and EM based Naive Bayes (**NB-EM**) models under the conditions of classic inductive learning, (**Induction**), unsupervised transductive transfer learning, (**TransductTransfer**), relaxed transductive transfer, (**RelaxTransductTransfer**), and supervised inductive transfer (**InductTransfer**), as introduced in the previous sections and summarized in Table 1.

Method	Induction			TransductTransfer			RelaxTransductTransfer			InductTransfer		
	Prec	Rec	**F1**	Prec	Rec	**F1**	Prec	Rec	**F1**	Prec	Rec	**F1**
MAXIMUM ENTROPY												
Basic	85	78	**82**	75	42	**54**	65	68	**67**	81	54	**65**
Regularize	-	-	-	-	-	-	-	-	-	87	84	**85**
Expand	-	-	-	-	-	-	-	-	-	84	62	**72**
SUPPORT VECTOR MACHINES												
ISVM	78	58	**67**	86	40	**54**	86	40	**55**	86	52	**65**
TSVM	68	79	**73**	86	46	**60**	72	75	**73**	86	58	**70**
NAIVE BAYES												
NB-ML	80	93	**86**	50	81	**62**	48	85	**61**	55	84	**67**
NB-EM	-	-	-	40	84	**54**	41	82	**55**	-	-	-

as summarized below:

$$\begin{aligned} \text{accuracy} &= \frac{\text{\# of tokens labeled correctly by the model}}{\text{total \# of tokens}} \\ \text{precision} &= \frac{\text{\# of POS-tokens labeled POS by the model}}{\text{\# of tokens labeled POS by the model}} \\ \text{recall} &= \frac{\text{\# of POS-tokens labeled POS by the model}}{\text{\# of POS-tokens}} \\ \text{F1} &= \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \end{aligned} \tag{15}$$

We use the $F1$ measure, which combines precision and recall into one metric, as our main evaluation measure.

Experiments and results

We assessed the relative performance of these methods on the four different learning settings described in previous sections. In addition to running the corresponding adaptations of each model for each of the settings, we did a few additional runs across the settings for purposes of illustration. For example, we ran the transductive SVM not only on the transductive settings, but also on the two inductive settings. Note that TSVM, when run on the inductive case corresponds to transductive learning (see Table 1) and when run on the inductive transfer

case, corresponds to the supervised transductive transfer learning in Table 1. There are other extra runs we did for the purposes of comparison, which will become apparent from the following discussion.

Table 4 summarizes the results under all four settings. The inductive experiment is dominated by Naive Bayes, achieving an F1 of 86% compared to MaxEnt's 82% and TSVM's 73%. This should not be surprising since generative models are known to be robust when large amount of labeled training data is available.

Moving to the transductive transfer setting causes all three methods' performances to fall, but MaxEnt falls most sharply, causing it to lose its entire lead over TSVM. Note that in this setting, basic MaxEnt and ISVM have equivalent performance of about 54% F1. The inductive Naive Bayes (using maximum likelihood estimator) proves to the top performer in this setting. TSVM, on the other-hand, is able to adjust its hyperplane in light of the transfer test data and stabilize its performance at 60%, even though it is unlabeled, because it knows where these points lie relative to the labeled training points in feature space. The transductive version of the naive Bayes (using EM), however, fares worse than its inductive counterpart. Since EM's optimization function is the marginal log-likelihood of the test data, it is not guaranteed to improve the classification performance in some cases.

In the relaxed transductive transfer setting, finally, where the target dataset is still unlabeled but all algorithms are told the expected proportion of positive examples, TSVM excels. Again, while MaxEnt is able to make significant use of this information (note the jump to 67% from 54%), it seems TSVM does a better job leveraging the prior knowledge into better performance. Maximum Likelihood based Naive Bayes, on the other hand loses out. It seems that the class conditional probability is more critical in naive Bayes than the prior, so tuning the latter's value does not have any positive impact on its performance. Also, notice that the EM based naive Bayes is even worse, repeating the pattern in the transductive transfer case.

Finally, the last column of Table 4 compares the performance of the three methods for inductive transfer learning: the prior-based regularized maximum entropy method (*Regularize*, described in section 2.8.4), and the feature expanding version (*Expand*, described in section 2.8.4). We can see that both methods handily outperform the transductive transfer methods described in the second column of Table 4, and for the most part outperform even the relaxed transductive transfer versions in column three. This should not be surprising given the fact that the inductive transfer methods can actually see some labeled examples from the target domain and thus, in the case of MaxEnt, better estimate the conditional expectation of the features in the target data. Likewise, since they have access to labeled target data, they can also assess the proportion of positive examples and adjust their decision functions accordingly. What is more surprising, however, is the fact that these methods do not significantly outperform the inductive learning methods described in the first column of Table 4. This suggests that these inductive transfer methods are relying almost entirely on their labeled target data in order to train their classifiers, and are not making full use of the large amount of labeled source data. One might assume that having access to almost four times as much

related data, in the form of the labeled source data, would significantly boost their ability to classify the target data (this is, after all, one of the stated goals of transfer learning). Dishearteningly, in this instance, this seems not to be the case. The regularized maximum entropy model does outperform[2] the basic MaxEnt in the inductive setting, but not by as much as might have been hoped for.

In order to measure how much these inductive transfer methods' explicit modeling of the transfer problem was responsible for their performance, we compared them to the baselines of ISVM, TSVM, MaxEnt and Naive Bayes trained on a simple concatenation of the labeled source and target training data. These transfer-agnostic methods clearly benefited from the addition of labeled target data (as compared to column *TransductiveTransfer*), yet still yielded consistently lower F1 than the transfer-aware *Regularize* and *Expand* methods, suggesting that the mere presence of labeled sets of both types (source and target) of data is not enough to account for the transfer methods' superior results. Instead, it seems it is the modeling of the different domains in the transfer problem, even in simple ways, that provides the extra boost to performance.

Conclusions

These experiments and analysis have shed light on a number of important issues and considerations related to the problems of transduction and transfer learning.

We have seen that in the case of discriminative models, even a small amount of prior knowledge about the target domain can greatly improve performance in a transductive transfer problem. Generative model is not able to exploit this information. For all these models, we notice that even large amounts of source data cannot overcome the advantage of having access to labeled data drawn from the target distribution.

We have also seen the degree to which pseudo-labeling based schemes can improve performance by incorporating the unlabeled structure of the target domain. However, this improvement is not seen in the generative Naive Bayes model. We believe this is because discriminative models directly optimize classification accuracy, while the EM based Naive Bayes model optimizes an unrelated function, namely, the marginal log-likelihood.

Finally, we have seen that the generative Naive Bayes model is robust in the inductive setting with large amount of labeled data, while the discriminative models are at least as good or better in the transductive setting. Of the two discriminative models considered, the margin based SVM seems to adapt better to the unlabeled data.

[2] *Regularize* has F1 of 85 vs. *MaxEnt*'s 82. Significance was determined by comparing the 99% binomial confidence intervals for each method's recall and precision.

3 Overall Objective & Preliminary Results

3.1 Overall objective

Our thesis attempts to explore the ways we can relax assumptions and exploit regularities in order to better solve real-world learning problems. This section introduces three examples of problems involving violated assumptions, and the solutions we can up with for overcoming this broken assumptions. Figure 2 shows one way of visualizing the various types of structure and regularity that can be tapped in solving various learning problems. In this model, instances x, their labels y, and constituent features F, can be joined in various relationships. For instance, the standard assumption joining instances is that they are all drawn independently from an identical distribution (i.i.d.). In the problem we face, however, this assumption is violated as instances (words) are drawn from different sections of a document (abstract, caption, etc.) and therefore have different distributions within those sections. Therefore, in this setting the i.i.d. assumption linking the instances to each other (most importantly, linking the training instances to the test instances) is severed, resulting in training and testing sets of seemingly unrelated instances among which it appears impossible to generalize. If we exploit a different regularity, however, re-linking the instances to each other in some way and taking the place of the invalidated i.i.d. assumption (see the top-left cloud in Figure 2), we are again able to learn and generalize across samples of training and test data. In this example (further explained in §3.4), the new regularity is the structure of the document itself and the distribution of instances across that structure (what we call *structural frequency features*) is what ties the examples together. Similarly, when the assumption that instances share the same set of features fails to hold, we develop a new method for relating these distinct, though related, features to one another via a natural linguistically-inspired hierarchy (the bottom cloud in Figure 2). These are the *feature hierarchies* explained in §3.2. Finally, the *snippets* of §3.5 are represented by the upper-right cloud in the diagram, linking the data not by the distribution of the instances or the features, but rather by their labels. Thus data that have very different attributes, but similar labels, will be joined together, while instances that appear to have similar features, but distinct labels, are segregated to allow for variation between domains.

In the following sections we further introduce and explore these three examples (visually summarized in Figure 2) and show how they contribute to this thesis' goal of robust learning in real-world systems. More detailed treatments can also be found in our ACL 2008 and CIKM 2008 publications [3, 5].

3.2 Feature hierarchy

By exploiting the hierarchical relationship present in many different natural language feature spaces, we are able to transfer knowledge across domains, both relating similar features to one another, while allowing distinct ones to vary across domains, genres and tasks [5].

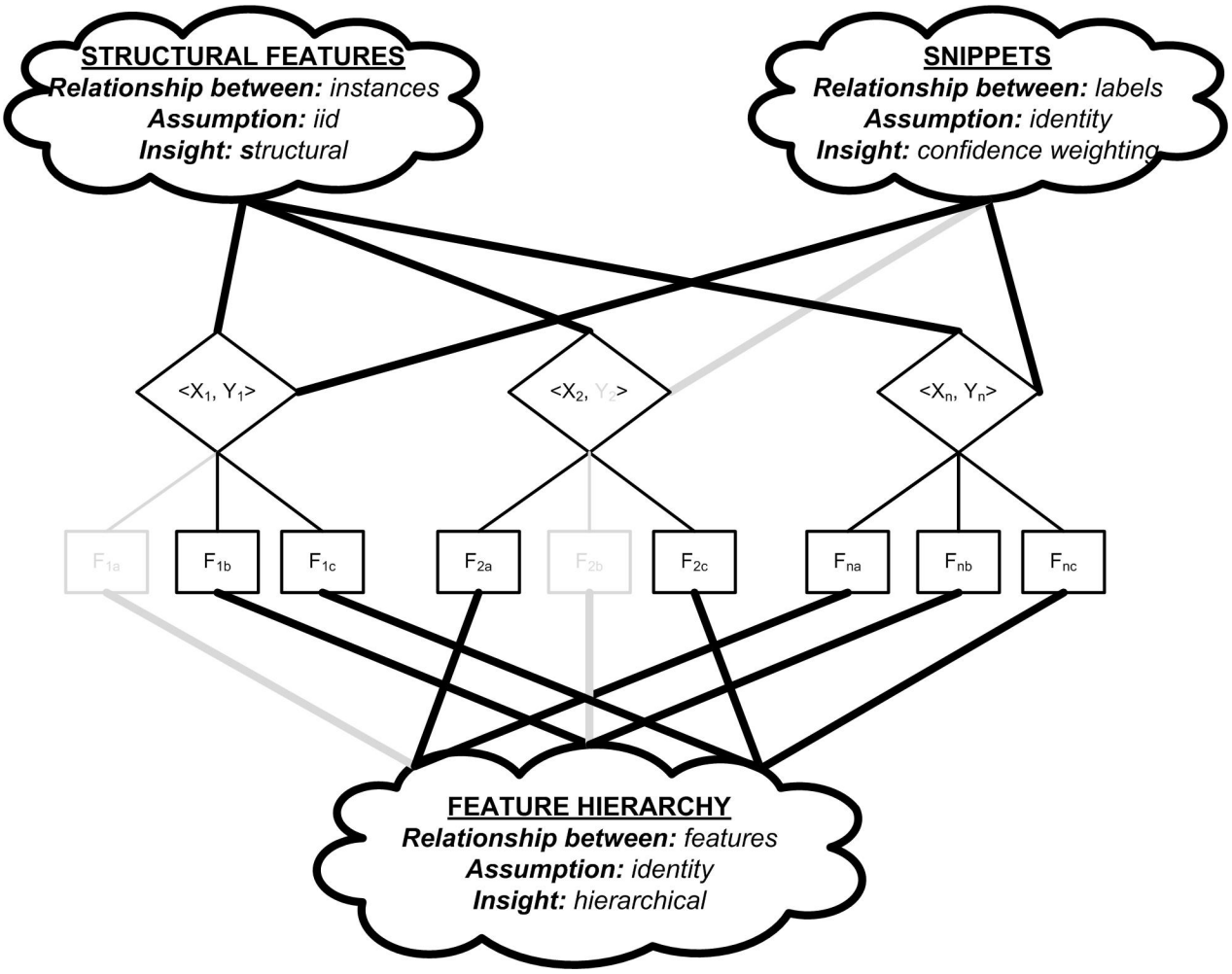

Figure 2: Visualization of the various types of structure used for transfer learning. Dark lines denote observed variables and relationships, while light lines symbolize unobserved data. Paths between and among instances, features and labels are conducted via *clouds* representing common relationships between these attributes. These paths allow information to flow from one type of observation in a certain domain to other related, though possibly distinct types of observations in related domains. For example, knowledge about instance-label tuple $\langle x_1, y_1 \rangle$ can directly inform an observer about another, unseen label, y_2, due to the i.i.d. relationship between x_1 and x_2 and the stability of $p(y|x)$. Similarly, knowledge of x_1's value for feature b (F_{1b}) can help you estimate the value for the unobserved F_{1a} if there is some relationship (as in our hierarchical lexical features example) linking the features to each other. In much of the work of this thesis these relationships are manifested as *external facts* and assumptions, for example, external linguistic knowledge about the hierarchy relating lexical features to one another, or external biological knowledge constraining which proteins can occur in which regions of a cell. These external data sources can often provide the information paths necessary to link various aspects of the data together, allowing us to learn in complex settings where common assumptions, like i.i.d., may not hold.

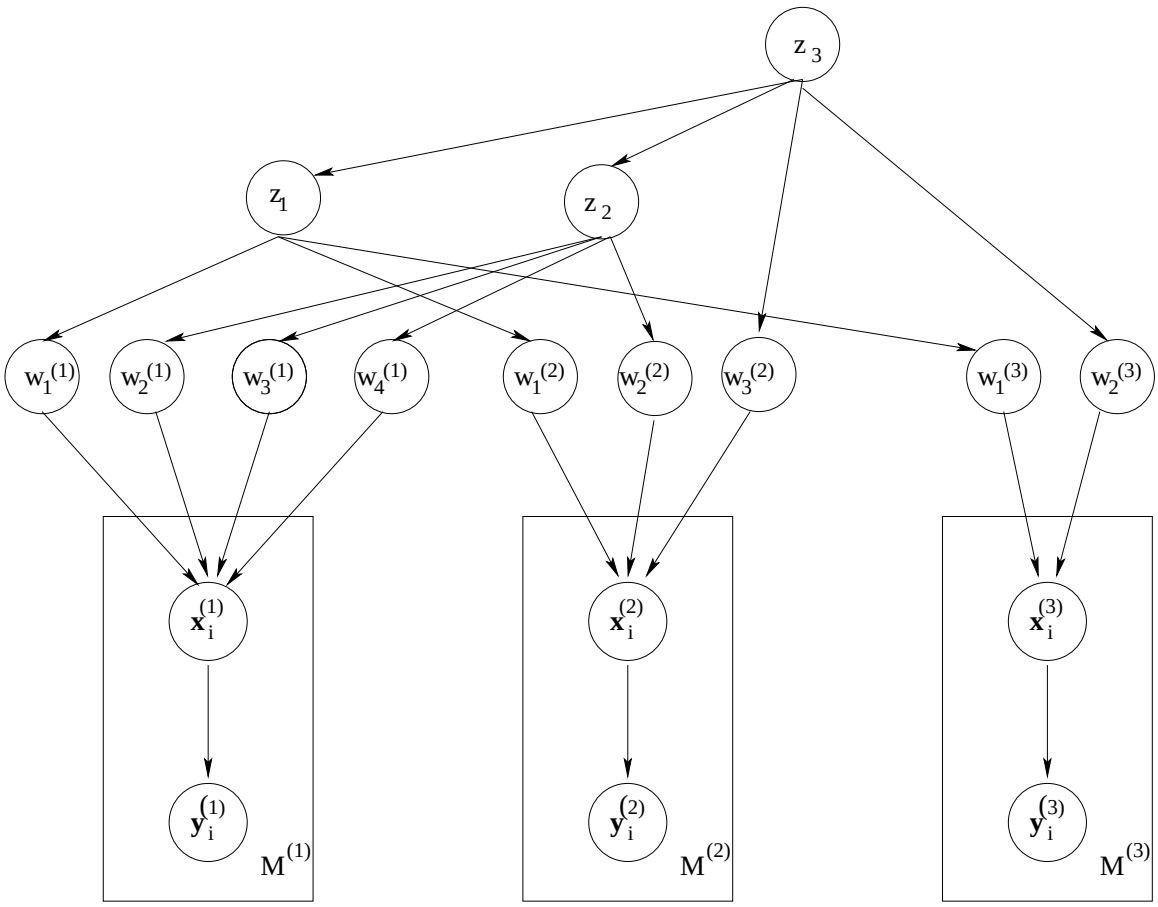

Figure 3: Graphical representation of the hierarchical transfer model.

3.2.1 Hierarchical feature trees

In many NER problems, features are often constructed as a series of transformations of the input training data, performed in sequence. Thus, if our task is to identify tokens as either being *(O)utside* or *(I)nside* person names, and we are given the labeled sample training sentence:

$$\begin{array}{cccccccc} O & O & O & O & & O & & I \\ \texttt{Give} & \texttt{the} & \texttt{book} & \texttt{to} & & \texttt{Professor} & & \texttt{Caldwell} \end{array} \tag{16}$$

one such useful feature might be: *Is the token one slot to the left of the current token* ***`Professor`****?* We can represent this symbolically as *L.1.Professor* where we describe the whole space of useful features of this form as: {***`direction`*** *= (L)eft, (C)urrent, (R)ight*}.{***`distance`*** *= 1, 2, 3, ...*}.{***`value`*** *= Professor, book, ...*}. We can conceptualize this structure as a tree, where each slot in the symbolic name of a feature is a branch and each period between slots represents another level, going from root to leaf as read left to right. Thus a subsection of the entire feature tree for the token `Caldwell` could be drawn as in Figure 4 (zoomed in on

LeftToken.*
LeftToken.IsWord.*
LeftToken.IsWord.IsTitle.*
LeftToken.IsWord.IsTitle.equals.*
LeftToken.IsWord.IsTitle.equals.mr

Table 5: A few examples of the feature hierarchy

the section of the tree where the *L.1.Professor* feature resides).

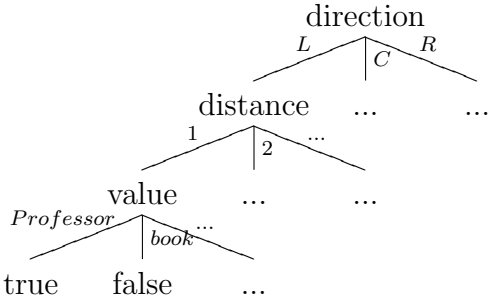

Figure 4: Graphical representation of a hierarchical feature tree for token `Caldwell` in example Sentence 16.

Representing feature spaces with this kind of tree, besides often coinciding with the explicit language used by common natural language toolkits [17], has the added benefit of allowing a model to easily back-off, or smooth, to decreasing levels of specificity. For example, the leaf level of the feature tree for our sample Sentence 16 tells us that the word `Professor` is important, with respect to labeling person names, when located one slot to the left of the current word being classified. This may be useful in the context of an academic corpus, but might be less useful in a medical domain where the word `Professor` occurs less often. Instead, we might want to learn the related feature *L.1.Dr*. In fact, it might be useful to generalize across multiple domains the fact that the word immediately preceding the current word is often important with respect to the named entity status of the current word. This is easily accomplished by backing up one level from a leaf in the tree structure to its parent, to represent a class of features such as *L.1.**. It has been shown empirically that, while the significance of *particular* features might vary between domains and tasks, certain generalized *classes* of features retain their importance across domains [44]. By backing-off in this way, we can use the feature hierarchy as a prior for transferring beliefs about the significance of entire *classes* of features across domains and tasks. Some examples illustrating this idea are shown in Table 5.

3.2.2 New model: hierarchical prior model

In this section, we will present a new model that learns simultaneously from multiple domains, by taking advantage of a feature hierarchy.

We will assume that there are D domains on which we are learning simultaneously. Let there be M_d training data in each domain d. For our experiments with non-identically distributed, independent data, we use conditional random fields (cf. §2.8.4). However, this model can be extended to any discriminative probabilistic model such as the MaxEnt model. Let $\Lambda^{(d)} = (\lambda_1^{(d)}, \cdots, \lambda_{F_d}^{(d)})$ be the parameters of the discriminative model in the domain d where F_d represents the number of features in the domain d.

Further, we will also assume that the features of different domains share a common hierarchy represented by a tree $\mathcal{T}$, whose leaf nodes are the features themselves (cf. Figure 4). The model parameters $\Lambda^{(d)}$, then, form the parameters of the leaves of this hierarchy. Each non-leaf node $n \in$ non-leaf($\mathcal{T}$) of the tree is also associated with a hyper-parameter z_n. Note that since the hierarchy is a tree, each node n has only one parent, represented by pa(n). Similarly, we represent the set of children nodes of a node n as ch(n).

The entire graphical model for an example consisting of three domains is shown in Figure 3. The conditional likelihood of the entire training data $(\mathbf{y}, \mathbf{x}) = \{(\mathbf{y}_1^{(d)}, \mathbf{x}_1^{(d)}), \cdots, (\mathbf{y}_{M_d}^{(d)}, \mathbf{x}_{M_d}^{(d)})\}_{d=1}^{D}$ is given by:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{z}) &= \left\{ \prod_{d=1}^{D} \prod_{k=1}^{M_d} P(\mathbf{y}_k^{(d)}|\mathbf{x}_k^{(d)}, \Lambda^{(d)}) \right\} \\ &\times \left\{ \prod_{d=1}^{D} \prod_{f=1}^{F_d} \mathcal{N}(\lambda_f^{(d)}|z_{\text{pa}(f^{(d)})}, 1) \right\} \\ &\times \left\{ \prod_{n \in \mathcal{T}_{nonleaf}} \mathcal{N}(z_n|z_{\text{pa}(n)}, 1) \right\} \end{aligned} \tag{17}$$

where the terms in the first line of eq. (17) represent the likelihood of data in each domain given their corresponding model parameters, the second line represents the likelihood of each model parameter in each domain given the hyper-parameter of its parent in the tree hierarchy of features and the last term goes over the entire tree $\mathcal{T}$ except the leaf nodes. Note that in the last term, the hyper-parameters are shared across the domains, so there is no product over d.

We perform a MAP estimation for each model parameter as well as the hyper-parameters.

Accordingly, the estimates are given as follows:

$$\begin{aligned} \lambda_f^{(d)} &= \sum_{i=1}^{M_d} \frac{\partial}{\partial \lambda_f^{(d)}} \left(\log P(\mathbf{y}_i^d | \mathbf{x}_i^{(d)}, \Lambda^{(d)}) \right) \\ &+ z_{\text{pa}(f^{(d)})} \\ z_n &= \frac{z_{\text{pa}(n)} + \sum_{i \in \text{ch}(n)} (\lambda|z)_i}{1 + |\text{ch}(n)|} \end{aligned} \tag{18}$$

where we used the notation $(\lambda|z)_i$ because node i, the child node of n, could be a parameter node or a hyper-parameter node depending on the position of the node n in the hierarchy. Essentially, in this model, the weights of the leaf nodes (model parameters) depend on the log-likelihood as well as the prior weight of its parent. Additionally, the weight of each hyper-parameter node in the tree is computed as the average of all its children nodes and its parent, resulting in a *smoothing* effect, both up and down the tree.

3.2.3 An approximate hierarchical prior model

The Hierarchical prior model is a theoretically well founded model for transfer learning through feature hierarchy. However, our preliminary experiments indicated that its performance on real-life data sets is not as good as expected. We conjecture that the main reason for this phenomenon is over-smoothing. In other words, by letting the information propagate from the leaf nodes in the hierarchy all the way to the root node, the model loses its ability to discriminate between its features.

As a solution to this problem, we propose an approximate version of this model that weds ideas from the exact hierarchical prior model and the Chelba model.

As with the Chelba prior method in §2.8.4, this approximate hierarchical method also requires two distinct data sets, one for training the prior and another for tuning the final weights. Unlike Chelba, we smooth the weights of the priors using the feature-tree hierarchy presented in §3.2, like the hierarchical prior model.

For smoothing of each feature weight, we chose to back-off in the tree as little as possible until we had a large enough sample of prior data (measured as M, the number of subtrees below the current node) on which to form a reliable estimate of the mean and variance of each feature or class of features. For example, if the tuning data set is as in Sentence 16, but the prior contains no instances of the word `Professor`, then we would back-off and compute the prior mean and variance on the next higher level in the tree. Thus the prior for *L.1.Professor* would be $\mathcal{N}$(mean(L.1.*), variance(L.1.*)), where mean() and variance() of *L.1.** are the sample mean and variance of all the features in the prior dataset that match the pattern *L.1.** – or, put another way, all the siblings of *L.1.Professor* in the feature tree. If fewer than M such siblings exist, we continue backing-off, up the tree, until an ancestor with sufficient descendants is found. A detailed description of the approximate hierarchical algorithm is shown in Table 6.

Input: $\mathcal{D}^{source} = (X^{source}_{train}, Y^{source}_{train})$
$\mathcal{D}^{target} = (X^{target}_{train}, Y^{target}_{train})$;
Feature sets $\mathcal{F}^{source}$, $\mathcal{F}^{target}$;
Feature Hierarchies $\mathcal{H}^{source}$, $\mathcal{H}^{target}$
Minimum membership size M

Train CRF using $\mathcal{D}^{source}$ to obtain
feature weights Λ^{source}
For each feature $f \in \mathcal{F}^{target}$
    Initialize: node $n = f$
    While ($n \notin \mathcal{H}^{source}$
    or $|\text{Leaves}(\mathcal{H}^{source}(n))| \leq M$)
    and $n \neq root(\mathcal{H}^{target})$
        $n \leftarrow \text{Pa}(\mathcal{H}^{target}(n))$
    Compute μ_f and σ_f using the sample
    $\{\lambda^{source}_i \mid i \in \text{Leaves}(\mathcal{H}^{source}(n))\}$
Train Gaussian prior CRF using $\mathcal{D}^{target}$ as data
and $\{\mu_f\}$ and $\{\sigma_f\}$ as Gaussian prior parameters.
Output:Parameters of the new CRF Λ^{target}.

Table 6: Algorithm for approximate hierarchical prior: $\text{Pa}(\mathcal{H}^{source}(n))$ is the parent of node n in feature hierarchy $\mathcal{H}^{source}$; $|\text{Leaves}(\mathcal{H}^{source}(n))|$ indicates the number of leaf nodes (basic features) under a node n in the hierarchy $\mathcal{H}^{source}$.

It is important to note that this smoothed tree is an approximation of the exact model presented in §3.2.2 and thus an important parameter of this method in practice is the degree to which one chooses to smooth up or down the tree. One of the benefits of this model is that the semantics of the hierarchy (how to define a feature, a parent, how and when to back-off and up the tree, etc.) can be specified by the user, in reference to the specific datasets and tasks under consideration. For our experiments, the semantics of the tree are as presented in §3.2.1.

The Chelba method can be thought of as a hierarchical prior in which no smoothing is performed on the tree at all. Only the leaf nodes of the prior's feature tree are considered, and, if no match can be found between the tuning and prior's training datasets' features, a $\mathcal{N}(0, 1)$ prior is used instead. However, in the new approximate hierarchical model, even if a certain feature in the tuning dataset does not have an analog in the training dataset, we can always back-off until an appropriate match is found, even to the level of the root.

Henceforth, we will use only the approximate hierarchical model in our experiments and discussion.

Table 7: Summary of data used in experiments

Corpus	Genre	Task
UTexas	Bio	Protein
Yapex	Bio	Protein
MUC6	News	Person
MUC7	News	Person
CSPACE	E-mail	Person

3.3 Investigation of hierarchical feature models

3.3.1 Data, domains and tasks

For our investigations into hierarchical feature models, we chose five different corpora (summarized in Table 7). Although each corpus can be considered its own *domain* (due to variations in annotation standards, specific task, date of collection, etc), they can also be roughly grouped into three different *genres*. These are: *abstracts from biological journals* [UT [13], Yapex [27]]; *news articles* [MUC6 [26], MUC7 [12]]; and *personal e-mails* [CSPACE [40]]. Each corpus, depending on its *genre*, is labeled with one of two name-finding *tasks*:

- protein names in biological abstracts
- person names in news articles and e-mails

We chose this array of corpora so that we could evaluate our hierarchical prior's ability to generalize across and incorporate information from a variety of domains, genres and tasks.

In each case, each item (abstract, article or e-mail) was tokenized and each token was hand-labeled as either being part of a name (protein or person) or not, respectively. We used a standard natural language toolkit [17] to compute tens of thousands of binary features on each of these tokens, encoding such information as capitalization patterns and contextual information from surrounding words. This toolkit produces features of the type described in §3.2.1 and thus was amenable to our hierarchical prior model. In particular, we chose to use the simplest default out-of-the-box feature generator and purposefully did not use specifically engineered features, dictionaries, or other techniques commonly employed to boost performance on such tasks. The goal of our experiments was to see to what degree named entity recognition problems naturally conformed to hierarchical methods, and not just to achieve the highest performance possible.

3.3.2 Experiments & results

We evaluated the performance of various transfer learning methods on the data and tasks described in §3.3.1. Specifically, we compared our approximate hierarchical prior model (HIER), implemented as a CRF, against three baselines:

- GAUSS: CRF model tuned on a single domain's data, using a standard $\mathcal{N}(0,1)$ prior
- CAT: CRF model tuned on a concatenation of multiple domains' data, using a $\mathcal{N}(0,1)$

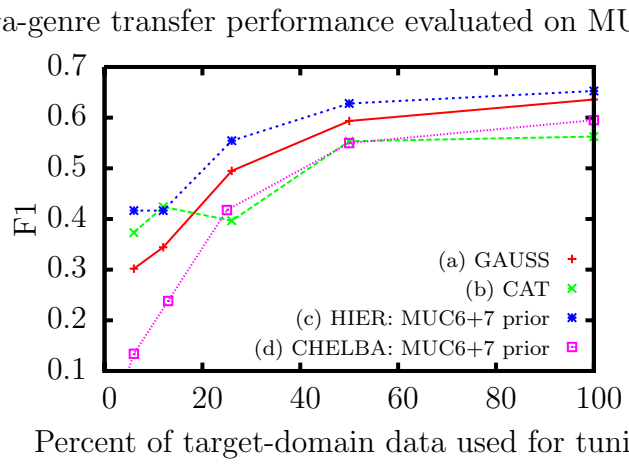

Figure 5: Adding a relevant HIER prior helps compared to the GAUSS baseline (**(c)** > **(a)**), while simply CAT'ing or using CHELBA can hurt (**(d)** ≈ **(b)** < **(a)**, except with very little data), and never beats HIER (**(c)** > **(b)** ≈ **(d)**). All models were tuned on MUC6 except CAT (**b**), tuned on MUC6+7.

prior

- CHELBA: CRF model tuned on one domain's data, using a prior trained on a different, related domain's data (cf. §2.8.4)

We use token-level $F1$ as our main evaluation measure, combining precision and recall into one metric.

3.3.3 Intra-genre, same-task transfer learning

Figure 5 shows the results of an experiment in learning to recognize person names in MUC6 news articles. In this experiment we examined the effect of adding extra data from a different, but related domain from the same genre, namely, MUC7. Line a shows the F1 performance of a CRF model tuned only on the target MUC6 domain (GAUSS) across a range of tuning data sizes. Line b shows the same experiment, but this time the CRF model has been tuned on a dataset comprised of a simple concatenation of the training MUC6 data from (a), along with a different training set from MUC7 (CAT). We can see that adding extra data in this way, though the data is closely related both in domain and task, has actually hurt the performance of our recognizer for training sizes of moderate to large size. This is most likely because, although the MUC6 and MUC7 datasets are closely related, they are still drawn from different distributions and thus cannot be intermingled indiscriminately. Line c shows the same combination of MUC6 and MUC7, only this time the datasets have been combined using the HIER prior. In this case, the performance actually does improve, both with respect to the single-dataset trained baseline (a) and the naively trained double-dataset (b). Finally, line d shows the results of the CHELBA prior. Curiously, though the domains are closely related, it does more poorly than even the non-transfer GAUSS. One possible explanation

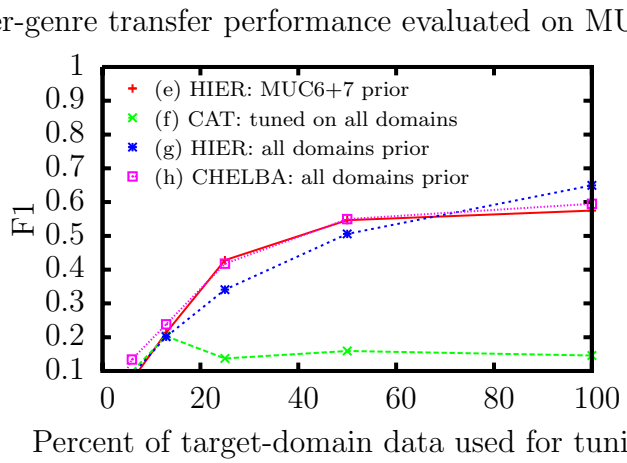

Figure 6: Transfer aware priors CHELBA and HIER effectively filter irrelevant data. Adding more irrelevant data to the priors doesn't hurt ((**e**) $\approx$ (**g**) $\approx$ (**h**)), while simply CAT'ing it, in this case, is disastrous ((**f**) << (**e**). All models were tuned on MUC6 except CAT (**f**), tuned on all domains.

is that, although much of the vocabulary is shared across domains, the interpretation of the features of these words may differ. Since CHELBA doesn't model the hierarchy among features like HIER, it is unable to smooth away these discrepancies. In contrast, we see that our HIER prior is able to successfully combine the relevant parts of data across domains while filtering the irrelevant, and possibly detrimental, ones. This experiment was repeated for other sets of intra-genre tasks, and the results are summarized in §3.3.5.

3.3.4 Inter-genre, multi-task transfer learning

In Figure 6 we see that the properties of the hierarchical prior hold even when transferring across tasks. Here again we are trying to learn to recognize person names in MUC6 e-mails, but this time, instead of adding only other datasets similarly labeled with person names, we are additionally adding biological corpora (UT & YAPEX), labeled not with person names but with protein names instead, along with the CSPACE e-mail and MUC7 news article corpora. The robustness of our prior prevents a model trained on all five domains (g) from degrading away from the intra-genre, same-task baseline (e), unlike the model trained on concatenated data (f). CHELBA (h) performs similarly well in this case, perhaps because the domains are so different that almost none of the features match between prior and tuning data, and thus CHELBA backs-off to a standard $\mathcal{N}(0, 1)$ prior.

This robustness in the face of less similarly related data is very important since these types of transfer methods are most useful when one possesses only very little target domain data. In this situation, it is often difficult to accurately estimate performance and so one would like assurance than any transfer method being applied will not have negative effects.

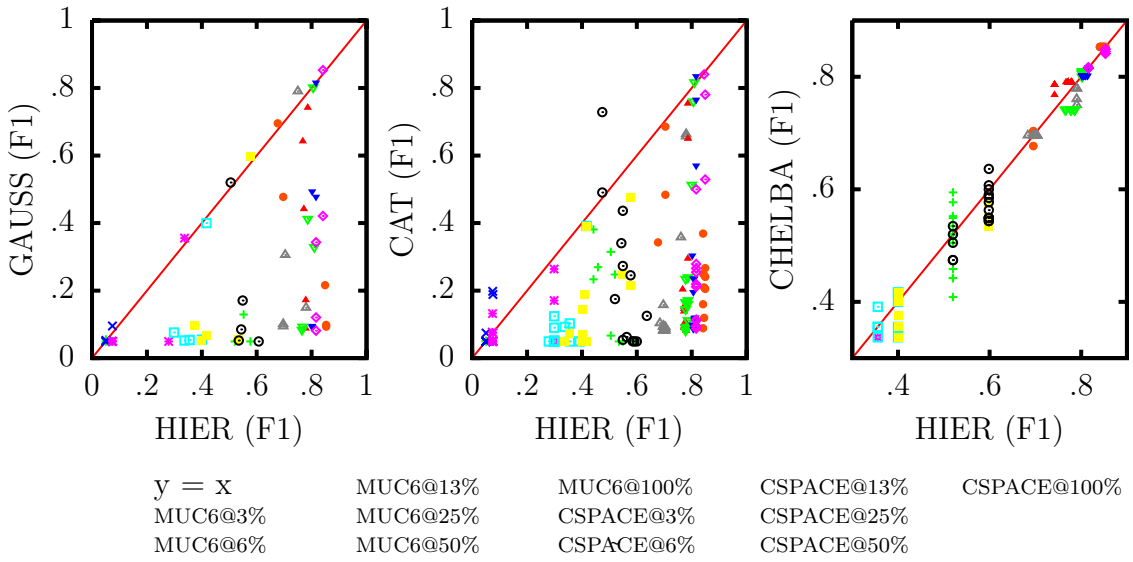

Figure 7: Comparative performance of baseline methods (GAUSS, CAT, CHELBA) vs. HIER prior, as trained on nine prior datasets (both pure and concatenated) of various sample sizes, evaluated on MUC6 and CSPACE datasets. Points below the y = x line indicate HIER outperforming baselines.

3.3.5 Comparison of HIER prior to baselines

Each scatter plot in Figure 7 shows the relative performance of a baseline method against HIER. Each point represents the results of two experiments: the y-coordinate is the F1 score of the baseline method (shown on the y-axis), while the x-coordinate represents the score of the HIER method in the same experiment. Thus, points lying below the y = x line represent experiments for which HIER received a higher F1 value than did the baseline. While all three plots show HIER outperforming each of the three baselines, not surprisingly, the non-transfer GAUSS method suffers the worst, followed by the naive concatenation (CAT) baseline. Both methods fail to make any explicit distinction between the source and target domains and thus suffer when the domains differ even slightly from each other. Although the differences are more subtle, the right-most plot of Figure 7 suggests HIER is likewise able to outperform the non-hierarchical CHELBA prior in certain transfer scenarios. CHELBA is able to avoid suffering as much as the other baselines when faced with large difference between domains, but is still unable to capture as many dependencies between domains as HIER.

3.3.6 Conclusions: hierarchical feature models

In this work we have introduced hierarchical feature tree priors for use in transfer learning on named entity extraction tasks. We have provided evidence that motivates these models on intuitive, theoretical and empirical grounds, and have gone on to demonstrate their effectiveness in relation to other, competitive transfer methods. Specifically, we have shown

that hierarchical priors allow the user enough flexibility to customize their semantics to a specific problem, while providing enough structure to resist unintended negative effects when used inappropriately. Thus hierarchical priors seem a natural, effective and robust choice for transferring learning across NER datasets and tasks.

Other techniques have tried to quantify the generalizability of certain features across domains [23, 35], or tried to exploit the common structure of related problems [7, 51]. Most of this prior work deals with supervised transfer learning, and thus requires labeled source domain data, though there are examples of unsupervised [4], semi-supervised [10, 32], and transductive approaches [56].

Recent work using so-called meta-level priors to transfer information across tasks [42], while related, does not take into explicit account the hierarchical structure of these meta-level features often found in NLP tasks. Daumé allows an extra degree of freedom among the features of his domains, implicitly creating a two-level feature hierarchy with one branch for *general* features, and another for *domain specific* ones, but does not extend his hierarchy further [22]). Similarly, work on hierarchical penalization [55] in two-level trees (concurrent with our ACL paper [5]) tries to produce models that are parsimonious with respect to a relatively small number of *groups* of variables as structured by the tree, as opposed to transferring knowledge between and among the branches of the tree themselves, as in our transfer setting.

3.4 Structural frequency features

By modeling the distribution of instances across various related domains in a single unified feature space, structural frequency features are able to combine these disparate source of information in order to create a stronger learner [3].

3.4.1 Lexical features

Most modern information extraction systems rely on some kind of representation, usually a set of **features**, that distills the document into a form the algorithm can interpret and manipulate. The exact form of these features is a vital component of the overall system, balancing the complexity of a rich representation with the parsimony of an insightful view of the domain and problem being solved. For named entity recognition, **lexical features**, which try to capture patterns of words within the text of a document, are one of the most common, and intuitive, types of these representations. Generally, a lexical feature is a function of a word and its context. The specific definition of this function may vary widely across domains and implementations. In our setting, each lexical feature is a boolean function over a token in a document representing the value and morphology of that token and its neighbors. For example, given the sentence fragment from a caption of a biological paper: 'Figure 4: Tyrosine phosphorylation...', some lexical features for the token 'Tyrosine' would look like:

CurrentToken.isWord.Tyrosine
CurrentToken.charPattern.Xx
CurrentToken.endsWith.ine
Right1Token.endsWith.ation
Left1Token.isWord.:
Left3Token.isWord.Figure

Table 8: Lexical features for token 'Tyrosine' in sample caption: 'Figure 4: *Tyrosine* phosphorylation...'.

Notice that, although these features are defined with respect to a certain current token, 'Tyrosine', they also take into account the context of that word in the document. In this example, if we knew that this occurrence of 'Tyrosine' was labeled as a protein, the fact that the token immediately to the left of the current token was a semi-colon (*Left1Token.isWord.:*) might be useful in predicting whether other, heretofore unseen tokens besides 'Tyrosine', that also happen to be preceded by a semi-colon, might also be proteins.

Since each word in one's vocabulary may constitute a feature (e.g., CurrentToken.isWord.A, CurrentToken.isWord.B, ...), it is not uncommon to have tens or even hundreds of thousands of such binary lexical features defined in one's feature space. The benefit of this is that such a large feature space can richly represent most any training set. The examples in Table 8 also include domain-specific features such as 'CurrentToken.endsWith.ine' (a common suffix for amino-acids). These custom features allow the researcher to bias his feature space towards specific features that he feels might be more informative with respect to his particular problem domain. While this specificity may be advantageous for an expert dealing with a limited domain, it can become a liability when that domain is uncertain, or even variable, as is the case in our transfer learning setting.

For instance, while the occurrence of a semi-colon or the word 'Figure' may be very informative in terms of identifying words as proteins in the captions of papers, if our extractor is trained only on abstracts it may never see those types of features. Indeed, since lexical features are merely functions of the specific sections of text seen during training, they are unable to capture information residing in other sections of the document which may prove useful. Even in the semi-supervised case where the learning algorithm has access to unlabeled target domain data, lexical features are unable to take advantage of this information since there is no way to relate the unlabeled tokens to the labeled ones.

Lexical features thus provide a valuable, but brittle, representation of the training data. Our work augments these rich, though domain-specific, lexical features with other non-lexical features based on the internal structure of a document, contributing another view of the data that is more robust to changes in the domain. We show that combining these types of domain-specific and domain-robust features produces a classifier that performs well across domains.

3.4.2 Document structure

We begin by highlighting the common observation that most documents are written with some kind of internal structure. For instance, the biological papers we studied in this experiment (like most academic papers) can be divided into three sections:

- **Abstract**: summarizing, at a high level, the main points of the paper such as the problem, contribution, and results.
- **Caption**: summarizing the figure it is attached to. These are especially important in biological papers where most important results are represented graphically. Unlike computer science papers, which usually have brief captions, in our corpus the average caption was over 125 words long, thus supporting our belief that they might contain useful information for our NER task.
- **Full text**: the main text of a paper, that is, everything else besides the abstract and captions.

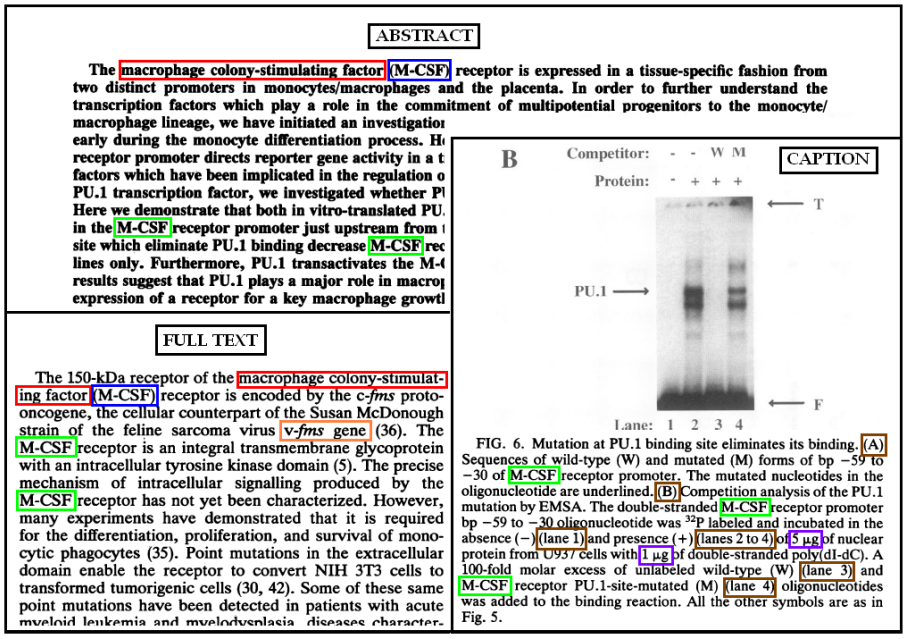

Figure 8: Sample biology paper. Each large black box represents a different subsection of the document's structure: abstract, caption and full text. Each small highlighted color box represents a different type of information: full protein name (red), abbreviated protein name (green), parenthetical abbreviated protein name (blue), non-protein parentheticals (brown), genes (orange), and measurement units (purple).

An example of such a structured document is provided in Figure 8. In this figure we see the various ways a protein can be referred to throughout the sections of a document. Notice how

the distribution of these types of occurrences varies across the structure of the document. For instance, full name references (red) do not appear in the caption, while non-protein parentheticals (brown) do not appear in the abstract. This is similar to the complex way the instances in Figure 2 are related to each other: not through a common distribution (as in the i.i.d. case), but rather through another mediating relationship (in this case, the structural features relating the occurrence of tokens across the **common** structure of a document). Here we see the importance of explicitly modeling the difference between the source and target domains: if one were to naïvely train a purely lexical feature based extractor on the abstracts and try to apply it to the captions, the extractor might be confused by the non-protein parentheticals, having never seen them in its training data. Likewise, it might waste significant probability mass on features representing the unabbreviated form of protein names which it might never see in its caption test data. It is important to note that in order to support this interpretation of the data we have to make the so-called *one-sense-per-discourse* assumption [28], namely, that tokens in one section of a document have the same meaning as identical tokens in other sections of the same document. This can be visualized as another layer of edges in Figure 2, lining occurrences of words across sections of a document, and ultimately, bridging the gap between the source and target domains.

Since we have no labeled target domain data, however, it is not obvious how we might amend or supplement our source domain training data so as to avoid these problems. The key insight is the fact that these domains, while distinct, are nevertheless related by the overarching structure of the documents in which they reside. For instance, while unabbreviated protein names never appear in the caption, and non-protein parentheticals never appear in the abstract, both of these occur in the full text of the paper. Thus, our goal is to find some class of features that can relate these different types of occurrences together across the differing subsections of a document's structure. We will achieve this by leveraging the one-sense-per-discourse assumption and our knowledge about our documents' structure to create two new types of features:

- **Structural frequency features**: Informative with respect to protein extraction, but make repeated occurrences of the same token in different sections look similar.
- **Snippets**: Pseudo-examples that push a learned classifier towards being consistent with the one-sense-per-discourse assumption.

3.4.3 Structural frequency features

Structural frequency features, like lexical features, are simply functions of tokens in context. Unlike purely lexical features, however, structural frequency features *are* able to leverage the occurrence of tokens across all sections of a document, including the unlabeled captions and full text. The idea is to leverage the fact that different types of tokens (e.g., unabbreviated protein names, non-protein parentheticals, etc.) occur with different frequencies in different sections of a document. In the example from Figure 8 in §3.4.2, we noticed that non-protein

parentheticals occurred quite often in the caption, but not at all in the abstract. While this seems informative, in our setting, unfortunately, we do not have labels for the caption data. We are therefore unable to make a distinction between *protein* and *non-protein* parentheticals in the caption section of the document. We can, however, make such a distinction in the *abstract* section of the same document, for which we do have labels. Thus, if we see a parenthesized token in a caption, and see the same token parenthesized in the abstract, we might be able to transfer that abstract token's label to the unlabeled caption occurrence. In this respect, these structural frequency features provide the links necessary to perform a kind of label propagation across the subsections of a document [67].

Given our previously stated one-sense-per-discourse assumption, we now have a means of transferring our labels across the different unlabeled sections of a document and may have a useful, non-transfer, semi-supervised learning model. Our ultimate goal, however, is semi-supervised domain adaptation, and these structural features, as described thus far, still lack a way of ensuring they will be robust across shifts in domain. The key to addressing that issue is to consider the occurrence of tokens not in isolation within each subsection of a document, but rather jointly across sections. For instance, in Figure 8 we see the token '(lane *)' occurs quite often in the caption, but never in the full text. In fact, there are many such non-proteins that only ever appear in the caption section of the document. In contrast, the token 'M-CSF' occurs with high frequency across all three sections of the document. Indeed, there are relatively few proteins that do *not* occur in the abstract of a paper. It seems we can use the relative distribution of tokens across the different sections of a document, in and of itself and without any lexical information, as a signal of that token's likelihood of being a protein. This makes sense, since authors are conveying different kinds of information, in different ways, across the various sections of a document and so are not equally likely to mention a protein, in the same particular way, across the entire document.

	Times in:			Log prob. in:			Log cond. prob. in:	
Word	A	C	F	A	C	F	P(C|A)	P(F|A)
'M-CSF'	3	3	4	-1.84	-1.61	-3.10	-1.20	-1.12
'macrophage'	2	0	1	-2.01	-Inf	-3.70	-Inf	-1.72
'(M-CSF)'	1	0	1	-2.30	-Inf	-3.70	-Inf	-1.72
'PU.1'	5	2	0	-1.61	-1.78	-Inf	-1.37	-Inf
'kDa'	0	0	1	-Inf	-Inf	-3.70	Never	Never

Table 9: Sample structural frequency features for *specific* tokens in example paper from Figure 8, as distributed across the (A)bstract, (C)aptions and (F)ull text. Log probabilities are computed assuming the following number of *total* tokens are found in each section of the paper: $A = 206$, $C = 121$, $F = 4,971$, $C|A = 47$, $F|A = 53$.

Specifically, for each unique word-type in a document, we counted the number of times it appeared in each of the different sections of that document (for example, the word-type 'M-

CSF' occurs three times in the abstract, four times in the full text, and three times in the caption of the example in Figure 8). We then normalized these counts by the total number of tokens in a given section to come up with an empirical probability of a word-type occurring in a particular section. We also computed the conditional forms of these features, that is, we counted the number of times a token appeared in section x, given that it also appeared in section y, again normalizing to form an empirical probability distribution. Continuing our example, the token 'macrophage' never occurs in the caption and thus, although the token does occur in the abstract, p(word occurring in caption|word occurs in abstract) is still zero (see Table 9 for more examples). These conditional structural frequency features allow us to characterize the particular distribution patterns that different types of words have across the sections of a document. In particular, we might be interested in modeling things like p(word is a protein|word appears in caption but not in abstract). Figures 9 and 10 show the distribution of two such features across our training data.

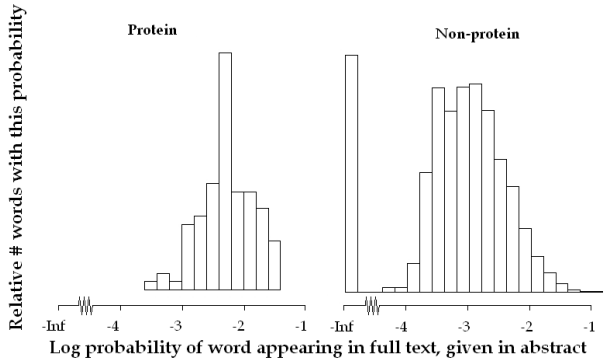

Figure 9: Histogram of the number of occurrences of protein (left) and non-protein (right) words with the given log normalized probability of appearing in *full text*, given that they also appear in an article's *abstract*.

Figure 9 shows a histogram of the number of times words labeled in the *abstract* as proteins (left) and non-proteins (right) occurred with a given log normalized probability in the document's full text, given that it also appeared (at least once) in the same document's abstract section. Since these probabilities are plotted on the log scale, any zero values (i.e., words that appear in abstracts but never in the full text), will be assigned to the bin at negative infinity. The lack of instances at negative infinity in the left plot is evidence that, if a protein is in an abstract, it is also always in the full text at least once. But this is not so for non-proteins – the large spike on the left side of the right plot shows a large number of non-proteins that appear in abstracts but never in the full text. Also notice the general right-shift of the entire distribution in the left plot, indicating an overall higher proportion of proteins occurring in full-text, given that they appear in an abstract, as compared to non-proteins.

Figure 10 shows a similar distribution, only this time the conditional structural frequency feature is measuring the likelihood of a word occurring in the *captions* of a paper, given that it appeared in the abstract. Notice, again, the left spike in the non-protein histogram on

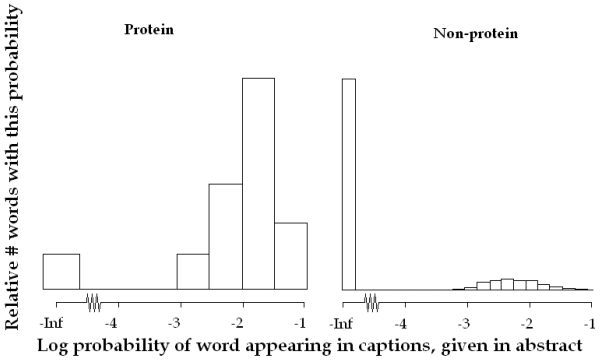

Figure 10: Histogram of the number of occurrences of protein (left) and non-protein (right) words with the given log normalized probability of appearing in *captions*, given that they also appear in an article's *abstract*.

the right, indicating that a large number of non-proteins never appear in article's captions, despite appearing in its abstract. In contrast, the higher peaks to the right of the protein plot on the left show a much higher proportion of proteins appearing in captions, given they also appear in the abstract.

These plots clearly demonstrate a significant difference in the distribution of protein and non-protein tokens across the various subsections (abstract, captions, and full text) of a document's structure and suggest these structural frequency features may be informative with respect to identifying and extracting proteins. Thus, at training time, we compute these structural frequency features for each token in our labeled training abstracts. Since counting token occurrences across document sections, however, does not require labels itself, we can freely use all the unlabeled text from the papers we have to calculate the features. Likewise, by leveraging the one-sense-per-discourse assumption, we can attach the word-type's label (found in the abstract) to each of these features defined across the various sections of the document. In the end, we are left with a semi-supervised intra-document representation of the labeled abstract data that is, due to its cross structural nature robust to shifts across the various document section domains.

3.5 Snippets

Although structural frequency features provide domain-robust signals to our extractor, they do not directly ameliorate the domain-brittleness of the lexical features discussed in §3.4.1. To address this issue, we introduce a kind of pseudo-data we call **snippets**. Snippets are tokens or short phrases taken from one of the unlabeled sections of a document and added to the training data, having been automatically positively or negatively labeled by some high confidence method [3]. Together, they help make the target distribution 'look' more like the source distribution with respect to the characteristics they share, while reshaping the target

distribution away from the source distribution in regards to the ways in which they differ.

3.5.1 Positive snippets

Positive snippets (i.e., snippets automatically labeled as positive examples) are an attempt to leverage the overlap between and across domains, by taking high confidence examples from one domain and transferring them to the other. In this sense, it is related to co-training [11]. Specifically, positive snippets leverage the one-meaning-per-discourse assumption (which we again rely upon due to our lack of labeled target data). The unlabeled target sections of a document are searched for tokens that match positively labeled tokens from the labeled source sections. Any matching instances are copied, along with a bit of neighboring context, into the training data, with the matching tokens labeled positive, and their context (where it does not match a protein name observed in the abstract) labeled negative, the idea being that this surrounding context will help inform the extractor of the differences in the distribution of lexical features in the target domain. Since our goal is to train an extractor that will be robust to shifts from source to target domain, we would like to introduce some examples of the target domain into the source domain training data to make it look more like the target domain. Since we don't have labels for the target domain, however, we have to rely on this high-confidence token matching heuristic.

3.5.2 Negative snippets

Similarly, **negative snippets** (i.e., snippets automatically labeled as negative examples) provide examples of tokens which may appear to be proteins when viewed with respect to the source domain, but are in fact not proteins in the target domain. These must rely on some form of prior knowledge about the target domain for their high-confidence automatic labeling, perhaps some kind of extractor previously trained for the target domain. For example, a researcher may have previously trained an extractor to identify tokens in captions that refer to specific panel locations in the accompanying image (e.g., the token '(B)' in Figure 8's caption). We call these types of references *image pointers* [18]. Although this kind of token pattern may look like a parenthetical protein mention if seen in an abstract, since we have an existing extractor able to identify it as an image pointer in captions (and thus, by assumed mutual exclusion, not a protein), we are able to add all occurrences in a paper's captions of similarly identified image pointers (labeled as negative) to that paper's labeled training data. A similar process can be followed for all kinds of high-confidence negative labels, such as measurement units, bibliographic citations, and various stoplists.

In this way, snippets allow us to use our unlabeled target data not just to add new inter-domain information (as with structural frequency features), but also, perhaps as importantly, to adjust and augment the distribution of existing source domain derived lexical features to make them more in accord with the target domain, ultimately producing extractors that are more robust to changes between training and test domains.

3.6 Investigation of structural frequency and snippet models

3.6.1 Data

Our training data for these experiments was drawn from two sources:

- GENIA: a corpus of Medline abstracts with each token annotated as to whether it is a protein names or not [49]
- PubMed Central (PMC): a free, on-line archive of biological publications [46]

Since our methods rely on having access to a document's labeled abstract along with the unlabeled captions and full text, and GENIA only provided labeled abstracts, we had to search PMC for the corresponding full text, where available. Of GENIA's 1,999 labeled abstracts, we were able to find the corresponding full article text (in PDF format) for 303 of them on PMC. These PDF's were (noisily) converted to text[3] and segmented into abstract, captions, and full text using automated tools. Figure 8 shows an example of one such segmented PDF.

Of these 303 papers, consisting of abstracts labeled with protein names along with corresponding unlabeled captions and full text, 218 (consisting of over 1.5 million tokens) were used for training, and 85 (almost 640,000 tokens) were used for testing.

3.6.2 Experiment

Experimentally, we used ablation studies to assess the amount of information our novel features:

- Structural frequency features (FREQ)
- Positive snippets (POS)
- Negative snippets (NEG)

each contribute to the task of protein name extraction, both in the non-transfer (abstract to abstract) and domain adaptation (abstract to caption) setting. In each case, we trained an extractor on a version of the training data constructed with the appropriate set of features. In all experiments we used the Minorthird toolkit to construct the lexical features and perform the CRF training [17].

[3]e-PDF PDF to Text Converter v2.1: http://www.e-pdfconverter.com

3.6.3 Results

3.6.4 Structural frequency features

Figure 11 compares the performance on held-out abstracts (in terms of precision and recall) of extractors training only on lexical features (**LEX** of §3.4.1), only on structural frequency features (**FREQ** of §3.4.3), and on a combination of both types of features (**LEX+FREQ**).

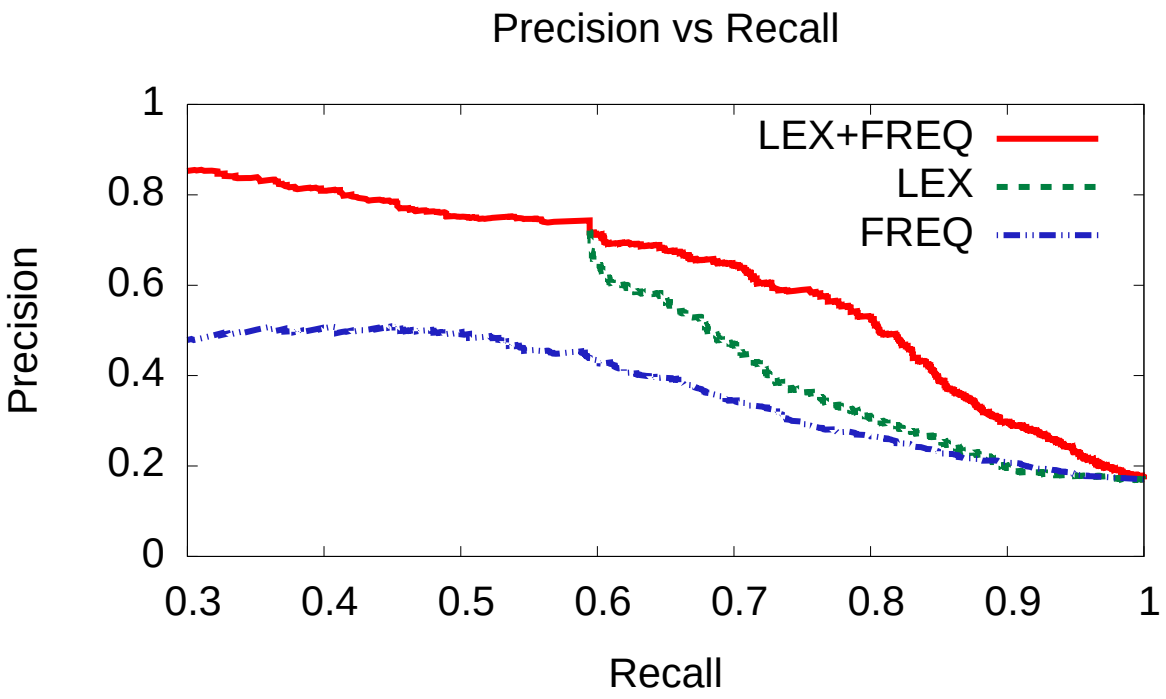

Figure 11: Precision versus recall of extractors trained on only lexical features (LEX), only structural frequency features (FREQ), and both sets of features (LEX+FREQ).

We can observe that, while the lexically trained model always outperforms the strictly structural frequency informed model (LEX dominates FREQ), the FREQ model nevertheless produces a competitive precision-recall curve despite having no access to any lexical information. This supports the intuition developed from observing the difference between protein and non-protein distributions in Figures 9 and 10.

Similarly, the fact that the combined model LEX+FREQ dominates each constituent model (LEX and FREQ individually) demonstrates that each type of feature (lexical and structural) is contributing a share of unique information, not represented by the other. This supports the connection with co-training, proposed in §3.5, by indicating that the feature sets are somewhat independent with respect to identifying protein names. The fact that their effect in the combined model is not completely additive suggests they are not wholly independent

either.

3.6.5 Non-transfer: abstract to abstract

Table 10 shows the performance of seven different extractors (sorted by F1), each trained on a unique combination of our proposed features: positive snippets (POS), negative snippets (NEG), and structural frequency features (FREQ), all along with the standard lexical features (LEX). A check mark in a feature's column means that row's extractor was provided with that column's features at train-time. In this non-transfer experiment, our model labeled tokens of held-out *abstracts* as protein or not, and these predictions were automatically evaluated with respect to token-level precision, recall and F1 measure using the held-out GENIA labels for those abstracts.

Model name	POS	NEG	FREQ	Prec	Rec	F1
FULL	✓	✓	✓	.738	.673	.704
FREQ			✓	.744	.640	.688
POS_FREQ	✓		✓	.727	.637	.679
POS	✓			.760	.555	.641
POS_NEG	✓	✓		.760	.547	.636
BASE				.753	.550	.636
NEG_FREQ		✓	✓	.751	.535	.625

Table 10: Summary of ablation study results for extractors trained on full papers and evaluated on *abstracts*.

From this table we can notice a number of trends. With respect to the baseline model (BASE) trained only on lexical features, adding positive snippets (POS) doesn't seem to help precision or recall much, while adding structural frequency features (FREQ) improves recall (and thus F1) dramatically. This makes sense, since positive snippets were proposed as a method of increasing domain-robustness, and these results are for the non-transfer setting. On the other hand, structural frequency features were proposed as a general purpose method of using an article's internal structure to help extract useful information from the unsupervised sections of the document. In this respect, FREQ features might be expected to aid in even the non-transfer setting, as they do here. Interestingly, although in isolation, and even in combination, POS and NEG snippets themselves don't seem to improve on the baseline model in the non-transfer setting, when combined with FREQ features (FULL) they do seem to provide another boost to recall. This may be due to the fact the inter-domain information implicitly incorporated by the structural frequency features allows the model to better make use of the cross-domain snippets.

We should note that, although this non-transfer, abstract to abstract setting is convenient (since we can get precise evaluation numbers) and the results encouraging, it is unclear what they might indicate about performance in the transfer setting.

3.6.6 Transfer: abstract to caption, full vs. baseline

Finally, we present the results of a user study in the domain adaptation setting. We trained extractors on various combinations of features computed on the training data, and compared them to the full model trained on lexical, structural, positive and negative snippets, evaluating each with respect to the proteins they predicted in held-out *captions*. Unlike the non-transfer setting, however, since we had no labels for any captions, we could not perform automatic evaluation. Instead, we employed human experts to manually compare the predictions made by variously constructed extractors and evaluate which they preferred. Using this method we found that our proposed model (FULL, the joint combination of all three new feature types: POS, NEG and FREQ) was preferred by users significantly more often ($p < .01$, see Table 11) than the baseline model trained only on lexical features.

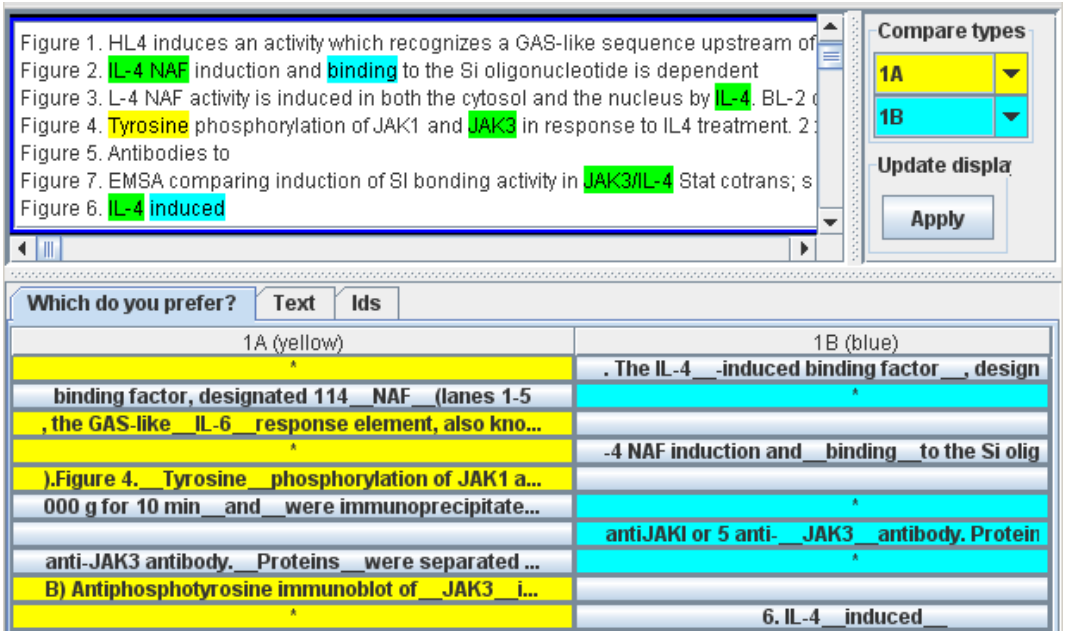

Figure 12: Screenshot of application used to compare various protein extractors' performance on captions in the face of no labeled data.

Figure 12 shows a screenshot of the tool we used to perform these evaluations. In the top-right, two extractors are being compared: 1A in yellow and 1B in blue (their names have been blinded from the evaluator). The top-left panel shows the captions of a particular test article with each extractor's positive (protein) predictions highlighted in its color, with green highlights representing tokens on which both extractors predict positive. The bottom

panel shows two columns of buttons: 1A's predictions are on the left, and 1B's on the right. Since we are evaluating user preference, only the predictions where the extractors disagree are shown. For each row (corresponding to a disagreement between extractors) the human expert clicks the cell of the prediction he prefers: clicking an empty cell in one column means the user believes the other column's extractor made a type I (false positive) error, while clicking a non-empty cell implies the other column's extractor made a type II (false negative) error. Each of these judgments can be viewed as the outcome of a paired trial, and by using a paired t-test, we can assess how the extractors differ along with which the user prefers, but can't exactly quantify by how much one has improved with respect to the other.

Evaluation is an important consideration in semi-supervised domain adaptation, since, by definition, no labeled test (target domain) data is available. The type of comparative evaluation we performed could be instrumented into various end-user applications (for example, click-through logs from protein name search engines such as SLIF[4]) to automatically extract the necessary user-preference information, thus obviating the need of a special evaluator.

3.6.7 Transfer: abstract to caption, full vs. ablated

Having established that a model based on a combination of our new features (incorporated in the FULL model) improved user preference over the baseline, purely lexical model, we then performed an ablation study to ascertain which of these new features (structural frequency (FREQ), positive snippets (POS), or negative snippets (NEG)) were responsible for the improvements observed. Table 11 summarizes these results for each ablation considered. In each such study comparing the full model to a degraded model, the full model was preferred significantly more often than the ablated model (one-sided paired t-test, $p<.01$), indicating that our proposed features are, in fact, useful for unsupervised domain adaptation.

Model	Compared to	p-value	# user labels
FULL	BASE	3.6 E-4	182
FULL	NEG_FREQ	9.9 E-9	78
FULL	POS_NEG	1.8 E-4	120
FULL	POS_FREQ	1.1 E-4	46

Table 11: Summary of transfer results for extractors trained on full papers and evaluated on *captions*. The preferred model is in bold.

From these results we can further observe that adding POS snippets seems to have a noticeable effect on user preference. This is a nice complement to the result from §3.6.5 which indicated that POS snippets are not as useful in the non-transfer setting. Indeed, it is the

[4]http://slif.cbi.cmu.edu/

ability of POS snippets to shape the labeled training source data to look more like the target data that allows the extractors so trained to be robust across shifts in domains. Similar user preference is seen for the contribution of NEG snippets and FREQ features, indicating that they too aid in domain-adaptation, both by leveraging unlabeled training data and by helping to inform the training data with some target domain attributes.

3.6.8 Conclusions: structural frequency features and snippets

In this section we have shown how exploiting structure, in the form of frequency features and positive and negative snippets, can help in the problem of semi-supervised domain adaptation. We have defined a new set of features based on structural frequency statistics and demonstrated their utility in representing inter-domain information drawn from both supervised and unsupervised sources, in a manner somewhat orthogonal to the traditional lexically based feature sets. Similarly, we have defined a technique for introducing high-confidence positively and negatively labeled pseudo examples (snippets) from the target domain into the source domain, and shown that these too provide a convenient, and effective, method for producing an extractor that is robust to domain shifts between training and testing data sets. Finally, through a comparative analysis of each new feature's contribution to same-domain and inter-domain information extraction performance, we have discovered an intriguing relationship between a feature's utility in the non-transfer and transfer settings. Along the way, in order to assess our transfer techniques' performance in the face of a lack of labeled test data, we have also developed a novel framework for human evaluation that facilitates statistically interpretable paired testing.

4 Proposed Work & Schedule

Given our results thus far (both the assumptions we have been able to drop and the regularities we have been able to exploit) and in light of the ultimate goal of this thesis, making learned classifiers and extractors more robust by using data (both labeled and unlabeled) from related domains and tasks and by exploiting stable regularities and complex relationships between different aspects of that data, we are encouraged to explore the following research tasks:

(I) Since our preliminary work has demonstrated that robustness can be achieved when we have stable relationships among the various components of a learning task, §4.1 proposes investigations into what *other*, further stable relationships and regularities there are to discover and exploit between the many tasks, features, labels and data available in the biological named entity recognition learning problem (cf. Figure 2). Similarly, these assumptions we use to tie together different aspects of the data may come with varying degrees of certainty. For example, how sure are we that an image pointer can *never* be protein name (i.e., that they are 100% mutually exclusive)? What about a measurement unit and a protein name? If these assumptions have different degrees of certainty, how does that affect the stability of the implied relationships and, ultimately, a learner's robustness? Is there a safe way to back off from these assumptions?

(II) Relatedly, §4.2 seeks to explore how to best make use of the many sources of *external* knowledge available related to the tasks of named entity recognition in general and in the biological domain in particular. Specifically, we plan to focus on integrating these various external sources of information into the learning process as surrogates for violated assumptions by leveraging their relationship to the knowledge already being *derived* from the data itself. At the same time, we need to ensure that these relationships between the derived and external data are, in fact, stable across domains and tasks. For instance, an ontology that knows about yeast proteins should be stable across different papers that are all mutually concerned with yeast, yet might not be equally stable across certain subsections of those papers. In contrast, a database that holds facts regarding entire individual papers should be stable across subsections of those papers (e.g., abstracts and captions), but not necessarily from one paper to another.

(III) Finally, §4.3 proposes to combine these techniques and to verify both the existence of the proposed relationships in a well-constrained domain and these relationships' ability to contribute to robust learning.

4.1 Task I. Cross-task & cross-domain learning

This section is generally concerned with assessing the relationship between the number and variety of domains and tasks a classifier has been trained on and the classifier's ultimate robustness.

4.1.1 Domain adaptation

Given the success of using structural frequency features and snippets to perform domain adaptation for protein extraction, we would like to see if these techniques generalize to the extraction of cell-line names under the same inter-domain settings, namely across both document structure ($cell_{abstract} \Rightarrow cell_{caption}$) and corpora boundaries ($cell_{Yapex} \Rightarrow cell_{UT}$). If we have time, we would also like to examine a non-biological setting, perhaps transferring from news article sources to the blogs that write about them, or from blogs to their comments.

4.1.2 Multi-task learning

Same domain multi-task transfer

The goal of this work is to investigate how much mutual simultaneous learning on different tasks can aid in developing robust extractors. We propose using abstract data labeled with *proteins* to try to predict *cell name* occurrences in abstracts. As in the protein case, we have access to abstracts labeled with cell line names and can attempt $protein_{abstract} \Rightarrow cell_{abstract}$ task transfer on the abstract domain. It is important to note that we are focusing on the goal of using multiple tasks in order to develop stable relationships for mutual robust learning, and not for pure target-task learning itself. That is, we want to use the relationships between tasks to improve learning for *all* tasks, as opposed to simply transferring from one 'solved' source task to a separate, as yet unlearned, target task.

In addition to this supervised multi-task transfer we can also try *semi-unsupervised* multi-task transfer from labeled proteins to unlabelled cells. Although this seems difficult given we have no labeled cell training data at all, nevertheless, given the fact we were successful in a similar unsupervised transfer case from abstracts to captions in the protein domain adaptation setting, with no labeled captions at all, perhaps we can find a similar result in the multi-task case by utilizing a regularity relating common structures across different tasks, analogous to the one-sense-per-discourse assumption that allowed us to tie together information across different sections of a document. For instance, if, via our source training data, we can identify proteins, and in addition we have external biological data relating certain proteins to certain cells, we may be able to use the protein information itself to help locate cells in the captions.

Domain adaptive multi-task transfer

Similarly, we can examine the seemingly more complex case of multi-task transfer across

domains. Specifically, trying to predict cell occurrence in captions given protein occurrence in abstracts: $protein_{abstract} \Rightarrow cell_{caption}$ transfer. Although this may seem ambitious, I believe it is worth some investigation.

4.1.3 Parallel labels: image pointers

Parallel labels are labels from other tasks related, but not identical to, the problem domain. For instance, in the captions of fluorescence microscopy images within biological publications, parenthetical references to image locations can often appear morphologically similar to parenthetical protein name mentions (cf. Figure 8). This can result in false positives. Thus for example, as we saw with the negative snippets from §3.5.2, if we had a high-confidence image location extractor (which we call *image pointers*), we could censor those examples from our result list, increasing our protein name precision. Other ideas include using image pointers to determine which section of an image a corresponding caption segment is aligned with, and using the stable properties of that image to transfer information across captions. This is related to work previously done for information extraction on the Web [21] that used the relationships between and amongst different classes of instances to influence their classification.

4.1.4 Parallel labels: image & experiment type

The images in biological publications can often be categorized by the type of experiment they depict. For instance, a time-series experiment in which a process or treatment is watched over time might commonly be depicted as a series of nearly identical images arranged in a sequence. Likewise, the comparison of a gene's expression across cell types might be represented as a bar chart. These experiments could be labeled by hand, or inferred from graphical or textual information [45]. The idea for this work is to classify these images by their experiment type and use these classifications as parallel labels to aid in NER on their associated captions. For instance, it may be easier to perform *abstract* $\Rightarrow$ *caption* domain adaptation only for captions relating to certain types of experiments. Similarly, it may be easier to transfer among captions that share an experiment type than across those whose experiment types differ. The general lesson is that the more information a learner has available (whether in the form of related data, labels, or even predictions) the more opportunities there are to discover exploitable regularities.

4.2 Task II. Relating external and derived knowledge

4.2.1 External data sources

In addition to the features, labels and structure available directly in a data set, in complex real-world applications it is often possible to find external sources of data related (if only

tangentially) to the learning problem at hand. For instance, in the protein name extraction problem there is an ontology of words describing gene and gene product attributes and how they relate to each other [20]. One question is: if two proteins co-occur in a relationship in this ontology (or some other *external* source) are they more likely to also co-occur in the text (or some other *internally derived* representation), or vice-versa? Another challenge is to determine when external data sources represent a stable regularity, and then they do not. This is related to the question of assumption confidence raised in Task I.

4.2.2 Hard & soft labels

A related problem in this area is the different ways hard and soft labels can be used. **Hard labels** offer high confidence, high precision predictions (both positive and negative) for examples but at the cost of being low recall, expensive to build and difficult to maintain. Examples include specialized dictionaries, gazetteers and stoplists. In contrast, **soft labels** are external data sources that are noisy, ambiguous or only partially labeled (perhaps by a curator, weak learner, or some other mostly automated method). Although this information seems much weaker than the hard labels above, it is usually much more abundant and easily acquired. Thus, we are interested to see if we can we still relate this external information to the internal regularities of the data, and its derived products, in order to allow us to relax some of our assumptions and improve the robustness of our learning.

4.3 Task III. Combining & verifying techniques

4.3.1 Combining techniques

We can benefit not only from introducing various new sources of information to our problem, but also by processing and combing these sources in new ways. For example, we may have a noisy external source of potential cell names which we want to use to help reduce the false-positives of cells identified as protein names. Combining techniques can quickly increase the amount of information available for a problem, but care must be taken to avoid creating inscrutable complexity.

4.3.2 Verifying hypotheses on limited domains

Although we have some intuitions (borne out empirically, albeit anecdotal) regarding the types of regularities that allow for robust learning between entities, we would nevertheless like to confirm and further explore these ideas. To do so, we will focus on a more limited domain where the task of named entity recognition is relatively easy, namely, in the yeast organism. *Yeast* genes and gene products follow a very regular naming convention, making them very easy to recognize and allowing us to automatically produce very accurate labels. We can then use these labels as a gold standard against which to compare the effectiveness

of exploiting various regularities in and outside the data, from co-occurrence in the text to relational properties in a database, determining how often they occur and how useful they are when they do. For instance, if we know from an external data source that a certain gene is expressed in a certain cell type, does that imply that those genes and cells will co-occur often in a paper's text as well? Likewise, if two entities co-occur in an abstract, are they more likely to also co-occur in a figure's caption? Or even in the figure itself (we have some computer vision tools to aid in the investigation of this question [45])?

4.4 Proposed Schedule

To accomplish these proposed works I will follow this schedule:

Approximate Period	Task	Possible publication venues
November 2008 - January 2009	Task I	ICML/SDM: 1/26/09, KDD: 2/6/09
January - March 2009	Task II	ACL: 2/22/09
March - June 2009	Task III	CIKM, EMNLP ~ June/July/09
June - August 2009	Write thesis	
August 2009	Defend thesis	

Table 12: Timeline for proposed work.

5 References

[1] S. Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall, 2007.

[2] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. In *JMLR 6*, pages 1817 – 1853, 2005.

[3] A. Arnold and W. W. Cohen. Intra-document structural frequency features for semi-supervised domain adaptation. In *CIKM '08*, 2008.

[4] A. Arnold, R. Nallapati, and W. W. Cohen. A comparative study of methods for transductive transfer learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM) 2007 Workshop on Mining and Management of Biological Data*, 2007.

[5] A. Arnold, R. Nallapati, and W. W. Cohen. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL:HLT '08*, 2008.

[6] J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[7] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *NIPS 20*, Cambridge, MA, 2007. MIT Press.

[8] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[9] D. M. Blei, J. A. Bagnell, and A. McCallum. Learning with scope, with application to information extraction and classification. In *UAI*, pages 53–60, 2002.

[10] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, Sydney, Australia, 2006.

[11] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 92–100, 1998.

[12] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. NYU: Description of the MENE named entity system as used in MUC-7, 1998.

[13] R. Bunescu, R. Ge, R. Kate, E. Marcotte, R. Mooney, A. Ramani, and Y. Wong. Comparative experiments on learning information extractors for proteins and their interactions. In *Journal of AI in Medicine*, 2004. Data from *ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/proteins.tar.gz*.

[14] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[15] C. Chelba and A. Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. In D. Lin and D. Wu, editors, *EMNLP 2004*, pages 285–292. ACL, 2004.

[16] S. Chen and R. Rosenfeld. A gaussian prior for smoothing maximum entropy models, 1999.

[17] W. W. Cohen. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net, 2004.

[18] W. W. Cohen, R. Wang, and R. Murphy. Understanding captions in biomedical publications. In *KDD*, pages 499–504, 2003.

[19] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[20] T. G. O. Consortium. Gene ontology: tool for the unification of biology. In *Nature Genet*, volume 25, pages 25–29, 2000.

[21] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *National Conference on Artificial Intelligence*, 1998.

[22] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[23] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research 26*, pages 101–126, 2006.

[24] A. Dempster, N. Laird, , and D. Rubin. Likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society, Series B*, 1977.

[25] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. In *Annals of Statistics*, 2004.

[26] D. Fisher, S. Soderland, J. McCarthy, F. Feng, and W. Lehnert. Description of the UMass system as used for MUC-6, 1995.

[27] K. Franzén, G. Eriksson, F. Olsson, L. Asker, P. Lidn, and J. Cöster. Protein names and how to find them. In *International Journal of Medical Informatics*, 2002.

[28] W. A. Gale, K. W. Church, and D. Yarowsky. One sense per discourse. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 233–237, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[29] Z. Ghahramami and M. Jordan. Learning from incomplete data. In *Technical Report, AI Lab No. 1509*, 1994.

[30] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *CIKM*, 2005.

[31] A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, 2006.

[32] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *CAP*, Nice, France, 2005.

[33] M. Janche and S. P. Abney. Information extraction from voicemail transcripts. In *EMNLP*, 2002.

[34] K. Ji, M. Ohta, and Y. Tsujii. Tuning support vector machines for biomedical named entity recognition. In *ACL Workshop on Natural Language Processing in the Biomedical Domain.*, 2002.

[35] J. Jiang and C. Zhai. Exploiting domain structure for named entity recognition. In *Human Language Technology Conference*, pages 74 – 81, 2006.

[36] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML 16*, 1999.

[37] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer, 2002.

[38] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

[39] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[40] R. Kraut, S. Fussell, F. Lerch, and J. Espinosa. Coordination in teams: evidence from a simulated management game, 2004.

[41] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[42] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2007.

[43] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *In AAAI Workshop on Learning for Text Categorization*, 1998.

[44] E. Minkov, R. C. Wang, and W. W. Cohen. Extracting personal names from email: Applying named entity recognition to informal text. In *HLT/EMNLP*, 2005.

[45] R. F. Murphy, Z. Kou, J. Hua, M. Joffe, and W. W. Cohen. Extracting and structuring subcellular location information from on-line journal articles: The subcellular location image finder. In *KSCE*, 2004.

[46] National Institues of Health. http://www.pubmedcentral.nih.gov/.

[47] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.

[48] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[49] T. Ohta, Y. Tateisi, H. Mima, and J. Tsujii. Genia corpus: an annotated research abstract corpus in molecular biology domain. In *HLT: Human Language Technology Conference*, pages 92–100, 2002.

[50] R. Raina, A. Y. Ng, and D. Koller. Transfer learning by constructing informative priors. In *ICML 22*, 2006.

[51] B. Schölkopf, F. Steinke, and V. Blanz. Object correspondence as a machine learning problem. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 776–783, New York, NY, USA, 2005. ACM.

[52] L. Shi and F. Campagne. Building a protein name dictionary from full text: a machine learning term extraction approach. *BMC Bioinformatics*, 6(88), 2005.

[53] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, pages 824–831. ACM, 2005.

[54] C. Sutton and A. McCallum. Composition of conditional random fields for transfer learning. In *HLT/EMLNLP*, 2005.

[55] M. Szafranski, Y. Grandvalet, and P. Morizet-Mahoudeaux. Hierarchical penalization. In *Advances in Neural Information Processing Systems 20*. MIT press, 2007.

[56] B. Taskar, M.-F. Wong, and D. Koller. Learning on the test data: Leveraging 'unseen' features. In *Proc. Twentieth International Conference on Machine Learning (ICML)*, 2003.

[57] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, volume 8, pages 640–646. MIT, 1996.

[58] R. Tibshirani. Regression shrinkage and selection via the lasso. In *J. Royal. Statist. Soc B*, 1996.

[59] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[60] R. C. Wang, A. Tomasic, R. E. Frederking, and W. W. Cohen. Learning to extract gene-protein names from weakly-labeled text in preparation. In *preparation*, 2006.

[61] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. In *Machine Learning*, 1996.

[62] Y. Yang. A study of thresholding strategies for text categorization. In *SIGIR*, 2001.

[63] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, 1995.

[64] M. Zaffalon and M. Hutter. Robust feature selection by mutual information distributions. In *18th Conference on Uncertainty in Artificial Intelligence*, 2002.

[65] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis, 2005.

[66] X. Zhu. Semi-supervised learning literature survey. In *Technical Report 1530*. University of Wisconsin, 2005.

[67] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.