Journal of Artificial Intelligence Research 26 (2006) 101-126 Submitted 8/05; published 5/06

Domain Adaptation for Statistical Classifiers

Hal Daumé III HDAUME@ISI.EDU
Daniel Marcu MARCU@ISI.EDU
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA

Abstract

The most basic assumption used in statistical learning theory is that training data and test data are drawn from the same underlying distribution. Unfortunately, in many applications, the "in-domain" test data is drawn from a distribution that is related, but not identical, to the "out-of-domain" distribution of the training data. We consider the common case in which labeled out-of-domain data is plentiful, but labeled in-domain data is scarce. We introduce a statistical formulation of this problem in terms of a simple mixture model and present an instantiation of this framework to maximum entropy classifiers and their linear chain counterparts. We present efficient inference algorithms for this special case based on the technique of conditional expectation maximization. Our experimental results show that our approach leads to improved performance on three real world tasks on four different data sets from the natural language processing domain.

1. Introduction

The generalization properties of most current statistical learning techniques are predicated on the assumption that the training data and test data come from the same underlying probability distribution. Unfortunately, in many applications, this assumption is inaccurate. It is often the case that plentiful labeled data exists in one domain (or coming from one distribution), but one desires a statistical model that performs well on another related, but not identical domain. Hand labeling data in the new domain is a costly enterprise, and one often wishes to be able to leverage the original, "out-of-domain" data when building a model for the new, "in-domain" data. We do not seek to *eliminate* the annotation of in-domain data, but instead seek to minimize the amount of new annotation effort required to achieve good performance. This problem is known both as *domain adaptation* and *transfer*.

In this paper, we present a novel framework for understanding the domain adaptation problem. The key idea in our framework is to treat the in-domain data as drawn from a mixture of two distributions: a "truly in-domain" distribution and a "general domain" distribution. Similarly, the out-of-domain data is treated as if drawn from a mixture of a "truly out-of-domain" distribution and a "general domain" distribution. We apply this framework in the context of conditional classification models and conditional linear-chain sequence labeling models, for which inference may be efficiently solved using the technique of conditional expectation maximization. We apply our model to four data sets with varying degrees of divergence between the "in-domain" and "out-of-domain" data and obtain predictive accuracies higher than any of a large number of baseline systems and a second model proposed in the literature for this problem.

The domain adaptation problem arises very frequently in the natural language processing domain, in which millions of dollars have been spent annotating text resources for morphological, syntactic and semantic information. However, most of these resources are based on text from the news domain (in most cases, the Wall Street Journal). The sort of language that appears in text from the Wall Street Journal is highly specialized and is, in most circumstances, a poor match to other domains. For instance, there has been a recent surge of interest in performing summarization (Elhadad, Kan, Klavans, & McKeown, 2005) or information extraction (Hobbs, 2002) of biomedical texts, summarization of electronic mail (Rambow, Shrestha, Chen, & Lauridsen, 2004), information extraction from transcriptions of meetings, conversations or voice-mail (Huang, Zweig, & Padmanabhan, 2001), among others. Conversely, in the machine translation domain, most of the parallel resources that machine translation system depend on for parameter estimation are drawn from transcripts of political meetings, yet the translation systems are often targeted at news data (Munteanu & Marcu, 2005).

2. Statistical Domain Adaptation

In the multiclass classification problem, one typically assumes the existence of a training set $\mathcal{D} = \{(x_n, y_n) \in \mathcal{X} \times \mathcal{Y} : 1 \leq n \leq N\}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is a finite set. It is assumed that each (x_n, y_n) is drawn from a fixed, but unknown base distribution p and that the training set is independent and identically distributed, given p. The learning problem is to find a function $f : \mathcal{X} \to \mathcal{Y}$ that obtains high predictive accuracy (this is typically done either by explicitly minimizing the regularized empirical error, or by maximizing the probabilities of the model parameters).

2.1 Domain Adaptation

In the context of domain adaptation, the situation becomes more complicated. We assume that we are given *two* sets of training data, $\mathcal{D}^{(\mathsf{o})}$ and $\mathcal{D}^{(\mathsf{i})}$, the "out-of-domain" and "in-domain" data sets, respectively. We no longer assume that there is a single fixed, but known distribution from which these are drawn, but rather assume that $\mathcal{D}^{(\mathsf{o})}$ is drawn from a distribution $p^{(\mathsf{o})}$ and $\mathcal{D}^{(\mathsf{i})}$ is drawn from a distribution $p^{(\mathsf{i})}$. The learning problem is to find a function f that obtains high predictive accuracy on data drawn from $p^{(\mathsf{i})}$. (Indeed, our model will turn out to be symmetric with respect to $\mathcal{D}^{(\mathsf{i})}$ and $\mathcal{D}^{(\mathsf{o})}$, but in the contexts we consider obtaining a good predictive model of $\mathcal{D}^{(\mathsf{i})}$ makes more intuitive sense.) We will assume that $|\mathcal{D}^{(\mathsf{o})}| = N^{(\mathsf{o})}$ and $|\mathcal{D}^{(\mathsf{i})}| = N^{(\mathsf{i})}$, where typically we have $N^{(\mathsf{i})} \ll N^{(\mathsf{o})}$. As before, we will assume that the $N^{(\mathsf{o})}$ out-of-domain data points are drawn iid from $p^{(\mathsf{o})}$ and that the $N^{(\mathsf{i})}$ in-domain data points are drawn iid from $p^{(\mathsf{i})}$.

Obtaining a good adaptation model requires the careful modeling of the relationship between $p^{(\mathsf{i})}$ and $p^{(\mathsf{o})}$. If these two distributions are independent (in the obvious intuitive sense), then the out-of-domain data $\mathcal{D}^{(\mathsf{o})}$ is useless for building a model of $p^{(\mathsf{i})}$ and we may as well ignore it. On the other hand, if $p^{(\mathsf{i})}$ and $p^{(\mathsf{o})}$ are identical, then there is no adaptation necessary and we can simply use a standard learning algorithm. In practical problems, though, $p^{(\mathsf{i})}$ and $p^{(\mathsf{o})}$ are neither identical nor independent.

2.2 Prior Work

There has been relatively little prior work on this problem, and nearly all of it has focused on specific problem domains, such as n-gram language models or generative syntactic parsing models. The standard approach used is to treat the out-of-domain data as "prior knowledge" and then to estimate maximum a posterior values for the model parameters under this prior distribution. This approach has been applied successfully to language modeling (Bacchiani & Roark, 2003) and parsing (Roark & Bacchiani, 2003). Also in the parsing domain, Hwa (1999) and Gildea (2001) have shown that simple techniques based on using carefully chosen subsets of the data and parameter pruning can improve the performance of an adapted parser. These models assume a data distribution $p(\mathcal{D} \mid \theta)$ with parameters θ and a prior distribution over these parameters $p(\theta \mid \eta)$ with hyper-parameters η. They estimate the η hyperparameters from the out-of-domain data and then find the maximum a posteriori parameters for the in-domain data, with the prior fixed.

In the context of conditional and discriminative models, the only domain adaptation work of which we are aware is the model of Chelba and Acero (2004). This model again uses the out-of-domain data to estimate a prior distribution, but does so in the context of a maximum entropy model. Specifically, a maximum entropy model is trained on the out-of-domain data, yielding optimal weights for that problem. These weights are then used as the *mean* weights for the Gaussian prior on the learned weights for the in-domain data.

Though effective experimentally, the practice of estimating a prior distribution from out-of-domain data and fixing it for the estimation of in-domain data leaves much to be desired. Theoretically, it is strange to estimate and *fix* a prior distribution from data; this is made more apparent by considering the form of these models. Denoting the in-domain data and parameters by $\mathcal{D}^{(\mathsf{i})}$ and θ, respectively, and the out-of-domain data and parameters by $\mathcal{D}^{(\mathsf{o})}$ and η, we obtain the following form for these "prior" estimation models:

$$\hat{\theta} = \arg\max_{\theta} p\left(\theta \mid \arg\max_{\eta} p(\eta)\, p\left(\mathcal{D}^{(\mathsf{o})} \mid \eta\right)\right) p\left(\mathcal{D}^{(\mathsf{i})} \mid \theta\right) \tag{1}$$

One would have a very difficult time rationalizing this optimization problem by anything other than experimental performance. Moreover, these models are unusual in that they do not treat the in-domain data and the out-of-domain data identically. Intuitively, there is no difference in the two sets of data; they simply come from different, related distributions. Yet, the prior-based models are highly asymmetric with respect to the two data sets. This also makes generalization to more than one "out of domain" data set difficult. Finally, as we will see, the model we propose in this paper, which alleviates all of these problems, outperforms them experimentally.

A second generic approach to the domain adaptation problem is to build an out of domain model and use its predictions as features for the in domain data. This has been successfully used in the context of named entity tagging (?). This approach is attractive because it makes no assumptions about the underlying classifier; in fact, multiple classifiers can be used.

2.3 Our Framework

In this paper, we propose the following relationship between the in-domain and the out-of-domain distributions. We assume that instead of two underlying distributions, there are actually *three* underlying distributions, which we will denote $q^{(\mathsf{o})}$, $q^{(\mathsf{g})}$ and $q^{(\mathsf{i})}$. We then consider $p^{(\mathsf{o})}$ to be a *mixture* of $q^{(\mathsf{o})}$ and $q^{(\mathsf{g})}$, and consider $p^{(\mathsf{i})}$ to be a mixture of $q^{(\mathsf{i})}$ and $q^{(\mathsf{g})}$. One can intuitively view the $q^{(\mathsf{o})}$ distribution as a distribution of data that is *truly out-of-domain*, $q^{(\mathsf{i})}$ as a distribution of data that is *truly in-domain* and $q^{(\mathsf{g})}$ as a distribution of data that is general to both domains. Thus, knowing $q^{(\mathsf{g})}$ and $q^{(\mathsf{i})}$ is sufficient to build a model of the in-domain data. The out-of-domain data can help us by providing more information about $q^{(\mathsf{g})}$ than is available by just considering the in-domain data.

For example, in part-of-speech tagging, the assignment of the tag "determiner" (DT) to the word "the" is likely to be a *general* decision, independent of domain. However, in the Wall Street Journal, "monitor" is almost always a verb (VB), but in technical documentation it will most likely be a noun. The $q^{(\mathsf{g})}$ distribution should account for the case of "the/DT", the $q^{(\mathsf{o})}$ should account for "monitor/VB" and $q^{(\mathsf{i})}$ should account for "monitor/NN."

3. Domain Adaptation in Maximum Entropy Models

The domain adaptation framework outlined in Section 2.3 is completely general in that it can be applied to any statistical learning model. In this section we apply it to log-linear conditional maximum entropy models and their linear chain counterparts, since these models have proved quite effective in many learning tasks. We will first review the maximum entropy framework, then will extend it to the domain adaptation problem; finally we will discuss domain adaptation in linear chain maximum entropy models.

3.1 Maximum Entropy Models

The maximum entropy framework seeks a conditional distribution $p\left(y \mid x\right)$ that is closest (in the sense of KL divergence) to the uniform distribution but also matches a set of training data $\mathcal{D}$ with respect to feature function expectations (Della Pietra, Della Pietra, & Lafferty, 1997). By introducing one Lagrange multiplier λ_i for each feature function f_i, this optimization problem results in a probability distribution of the form:

$$p\left(y \mid x \,;\, \boldsymbol{\lambda}\right) = \frac{1}{Z_{\boldsymbol{\lambda},x}} \exp\left[\boldsymbol{\lambda}^\top \boldsymbol{f}(x,y)\right] \tag{2}$$

Here, $\boldsymbol{u}^\top\boldsymbol{v}$ denotes the scalar product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, given by: $\boldsymbol{u}^\top\boldsymbol{v} = \sum_i u_i v_i$. The normalization constant in Eq (2), $Z_{\boldsymbol{\lambda},x}$, is obtained by summing the exponential over all possible classes $y' \in \mathcal{Y}$. This probability distribution is also known as an *exponential distribution* or a *Gibbs distribution*. The learning (or optimization) problem is to find the vector $\boldsymbol{\lambda}$ that maximizes the likelihood in Eq (2). In practice, to prevent over-fitting, one typically optimizes a penalized (log) likelihood, where an isotropic Gaussian prior with mean 0 and covariance matrix $\sigma^2 I$ is placed over the parameters $\boldsymbol{\lambda}$ (Chen & Rosenfeld, 1999). The graphical model for the standard maximum entropy model is depicted on the left of Figure 1. In this figure, circular nodes correspond to random variables and square nodes correspond to fixed variables. Shaded nodes are observed in the training data and empty nodes are hidden or unobserved. Arrows denote conditional dependencies.

In general, the feature functions $f(x, y)$ may be arbitrary real-valued functions; however, in this paper we will restrict our attention to binary features. In practice, this is not a harsh restriction: many problems in the natural language domain naturally employ only binary features (for real valued features, binning techniques can be applied). Additionally, for notational convenience, we will assume that the features $f_i(x, y)$ can be written in product form as $g_i(y)h_i(x)$ for arbitrary binary functions g over outputs and binary features h over inputs. The latter assumption means that we can consider x to be a binary vector where $x_i = h_i(x)$; in the following this will simplify notation significantly (the extension to the full case is straightforward, but messy, and is therefore not considered in the remainder of this paper). By considering $\boldsymbol{x}$ as a vector, we may move the class dependence to the parameters and consider $\boldsymbol{\lambda}$ to be a matrix where $\lambda_{y,i}$ is the weight for h_i for class y. We will write $\boldsymbol{\lambda}_y$ to refer to the column vector of λ corresponding to class y. As $\boldsymbol{x}$ is also considered a column vector, we write ${\boldsymbol{\lambda}_y}^\top \boldsymbol{x}$ as shorthand for the dot product between $\boldsymbol{x}$ and the weights for class y. Under this modified notation, we may rewrite Eq (2) as:

$$p\left(y \mid \boldsymbol{x} \; ; \boldsymbol{\lambda}\right) = \frac{1}{Z_{\boldsymbol{\lambda},\boldsymbol{x}}} \exp\left[{\boldsymbol{\lambda}_y}^\top \boldsymbol{x}\right] \tag{3}$$

Combining this with a Gaussian prior on the weights, we obtain the following form for the log posterior of a data set:

$$l = \log p\left(\boldsymbol{\lambda} \mid \mathcal{D}, \sigma^s\right) = -\frac{1}{2\sigma^2}\boldsymbol{\lambda}^\top\boldsymbol{\lambda} + \sum_{n=1}^{N}\left[{\boldsymbol{\lambda}_{y_n}}^\top \boldsymbol{x}_n - \log \sum_{y' \in \mathcal{Y}} \exp\left[{\boldsymbol{\lambda}_{y'}}^\top \boldsymbol{x}_n\right]\right] + \text{const} \tag{4}$$

The parameters $\boldsymbol{\lambda}$ can be estimated using any convex optimization technique; in practice, limited memory BFGS (Nash & Nocedal, 1991; Averick & Moré, 1994) seems to be a good choice (Malouf, 2002; Minka, 2003) and we will use this algorithm for the experiments described in this paper. In order to perform these calculations, one must be able to compute the gradient of Eq (4) with respect to $\boldsymbol{\lambda}$, which is available in closed form.

3.2 The Maximum Entropy Genre Adaptation Model

Extending the maximum entropy model to account for both in-domain and out-of-domain data in the framework described earlier requires the addition of several extra model parameters. In particular, for each in-domain data point $(\boldsymbol{x}_n^{(\mathsf{i})}, y_n^{(\mathsf{i})})$, we assume the existence of a binary indicator variable $z_n^{(\mathsf{i})}$. A value $z_n^{(\mathsf{i})} = 1$ indicates that $(\boldsymbol{x}_n^{(\mathsf{i})}, y_n^{(\mathsf{i})})$ is drawn from $q^{(\mathsf{i})}$ (the truly in-domain distribution), while a value $z_n^{(\mathsf{i})} = 0$ indicates that it is drawn from $q^{(\mathsf{g})}$ (the general-domain distribution). Similarly, for each out-of-domain data point $(\boldsymbol{x}_n^{(\mathsf{o})}, y_n^{(\mathsf{o})})$, we assume a binary indicator variable $z_n^{(\mathsf{o})}$, where $z_n^{(\mathsf{o})} = 1$ means this data point is drawn from $q^{(\mathsf{o})}$ (the truly out-of-domain distribution) and a value of 0 means that it is drawn from $q^{(\mathsf{g})}$ (the general-domain distribution). Of course, these indicator variables are not observed in the data, so we must infer their values automatically.

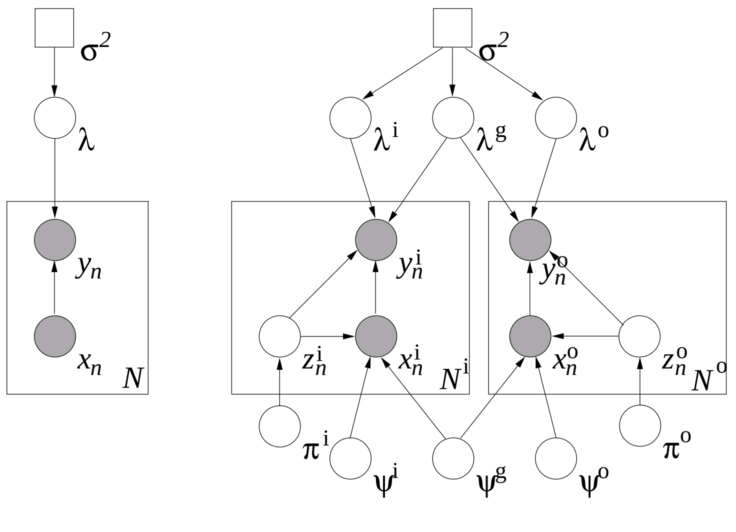

Figure 1: (Left) the standard logistic regression model; (Right) the Mega Model.

According to this model, the z_ns are binary random variables that we assume are drawn from a Bernoulli distribution with parameter $\pi^{(\mathsf{i})}$ (for in-domain) and $\pi^{(\mathsf{o})}$ (for out-of-domain). Furthermore, we assume that there are three $\boldsymbol{\lambda}$ vectors, $\boldsymbol{\lambda}^{(\mathsf{i})}$, $\boldsymbol{\lambda}^{(\mathsf{o})}$ and $\boldsymbol{\lambda}^{(\mathsf{g})}$ corresponding to $q^{(\mathsf{i})}$, $q^{(\mathsf{o})}$ and $q^{(\mathsf{g})}$, respectively. For instance, if $z_n = 1$, then we assume that $\bar{x}_n$ should be classified using $\boldsymbol{\lambda}^{(\mathsf{i})}$. Finally, we model the binary vectors $\boldsymbol{x}_n^{(\mathsf{i})}$s (respectively $\boldsymbol{x}_n^{(\mathsf{o})}$s) as being drawn independently from Bernoulli distributions parameterized by $\boldsymbol{\psi}^{(\mathsf{i})}$ and $\boldsymbol{\psi}^{(\mathsf{g})}$ (respectively, $\boldsymbol{\psi}^{(\mathsf{o})}$ and $\boldsymbol{\psi}^{(\mathsf{g})}$). Again, when $z_n = 1$, we assume that $\boldsymbol{x}_n$ is drawn according to $\boldsymbol{\psi}^{(\mathsf{i})}$. This corresponds to a naïve Bayes assumption over the generative probabilities of the $\boldsymbol{x}_n$ vectors. Finally, we place a common Beta prior over the naïve Bayes parameters, $\boldsymbol{\psi}$. Allowing ν to range over $\{\mathsf{i}, \mathsf{o}, \mathsf{g}\}$, the full hierarchical model is:

$$\begin{array}{rlrl} \boldsymbol{\psi}_f^{(\nu)} \mid a, b & \sim \mathcal{B}et(a,b) & \boldsymbol{\lambda}^{(\nu)} \mid \sigma^2 & \sim \mathcal{N}or(0, \sigma^2 I) \\ z_n^{(\mathsf{i})} \mid \pi^{(\mathsf{i})} & \sim \mathcal{B}er(\pi^{(\mathsf{i})}) & z_n^{(\mathsf{o})} \mid \pi^{(\mathsf{o})} & \sim \mathcal{B}er(\pi^{(\mathsf{o})}) \\ x_{nf}^{(\mathsf{i})} \mid z_n^{(\mathsf{i})}, \boldsymbol{\psi}_f^{(\mathsf{i})}, \boldsymbol{\psi}_f^{(\mathsf{g})} & \sim \mathcal{B}er(\boldsymbol{\psi}_f^{z_n^{(\mathsf{i})}}) & x_{nf}^{(\mathsf{o})} \mid z_n^{(\mathsf{o})}, \boldsymbol{\psi}_f^{(\mathsf{o})}, \boldsymbol{\psi}_f^{(\mathsf{g})} & \sim \mathcal{B}er(\boldsymbol{\psi}_f^{z_n^{(\mathsf{o})}}) \\ y_n^{(\mathsf{i})} \mid x_n^{(\mathsf{i})}, z_n^{(\mathsf{i})}, \boldsymbol{\lambda}^{(\mathsf{i})}, \boldsymbol{\lambda}^{(\mathsf{g})} & \sim \mathit{Gibbs}(x_n^{(\mathsf{i})}, \boldsymbol{\lambda}^{z_n^{(\mathsf{i})}}) & y_n^{(\mathsf{o})} \mid x_n^{(\mathsf{o})}, z_n^{(\mathsf{o})}, \boldsymbol{\lambda}^{(\mathsf{o})}, \boldsymbol{\lambda}^{(\mathsf{g})} & \sim \mathit{Gibbs}(x_n^{(\mathsf{o})}, \boldsymbol{\lambda}^{z_n^{(\mathsf{o})}}) \end{array} \quad (5)$$

We term this model the "Maximum Entropy Genre Adaptation Model" (the Mega Model). The corresponding graphical model is shown on the right in Figure 1. The generative story for an in-domain data point $\boldsymbol{x}^{(\mathsf{i})}$ is as follows:

1. Select whether $\boldsymbol{x}^{(\mathsf{i})}$ will be truly in-domain or general-domain and indicate this by $z^{(\mathsf{i})} \in \{i, g\}$. Choose $z^{(\mathsf{i})} = i$ with probability $\pi^{(\mathsf{i})}$ and $z^{(\mathsf{i})} = g$ with probability $1 - \pi^{(\mathsf{i})}$.

2. For each component f of $\boldsymbol{x}^{(\mathsf{i})}$, choose $x_f^{(\mathsf{i})}$ to be 1 with probability $\boldsymbol{\psi}_f^{z^{(\mathsf{i})}}$ and 0 with probability $1 - \boldsymbol{\psi}_f^{z^{(\mathsf{i})}}$.

3. Choose a class y according to Eq (3) using the parameter vector $\boldsymbol{\lambda}^{z^{(\mathsf{i})}}$.

The story for out-of-domain data points is identical, but uses the truly out-of-domain and general-domain parameters, rather than the truly in-domain parameters and general-domain parameters.

3.3 Linear Chain Models

The straightforward extension of the maximum entropy classification model to the maximum entropy Markov model (MEMM) (McCallum, Freitag, & Pereira, 2000) is obtained by assuming that the targets y_n are sequences of labels. The canonical example for this model is part of speech tagging: each word in a sequence is assigned a part of speech tag. By introducing a first order Markov assumption on the tag sequence, one obtains a linear chain model that can be viewed as the discriminative counterpart to the standard (generative) hidden Markov model. The parameters of these models can be estimated again using limited memory BFGS. The extension of the Mega Model to the linear chain framework is similarly straightforward, under the assumption that each label (part of speech tag) has its own indicator variable z (versus a global indicator variable z for the entire tag sequence).

The techniques described herein may also be applied to the conditional random field framework of Lafferty, McCallum, and Pereira (2001), which fixes a bias problem of the MEMM by performing global normalization rather than per-state normalization. There is, however, a subtle difficulty in a direct application to CRFs. Specifically, one would need to decide if a single z variable would be assigned to an entire sentence, or to each word individually. In the MEMM case, it is most natural to have one z per word. However, to do so in a CRF would be computationally more expensive. In the remainder, we continue to use the MEMM model for efficiency purposes.

4. Conditional Expectation Maximization

Inference in the Mega Model is slightly more complex than in standard maximum entropy models. However, inference can be solved efficiently using conditional expectation maximization (CEM), a variant of the standard expectation maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977), due to Jebara and Pentland (1998). At a high level, EM is useful for computing in *generative* models with hidden variables, while CEM is useful for computing in *discriminative* models with hidden variables; the Mega Model belongs to the latter family, so CEM is the appropriate choice.

The standard EM family of algorithms maximizes a *joint* likelihood over data. In particular, if $(x_n, y_n)_{n=1}^{N}$ are data and z is a (discrete) hidden variable, the M-step of EM proceeds by maximizing the bound given in Eq (6)

$$\log p\left(x, y \mid \Theta\right) = \log \sum_{z} p\left(z, x, y \mid \Theta\right) = \log \mathbb{E}_{z \sim p(\cdot \mid x; \Theta)} p\left(x, y \mid z; \Theta\right) \tag{6}$$

In Eq (6), $\mathbb{E}_z$ denotes an expectation. One may now apply Jensen's inequality to this equation, which states that $f(\mathbb{E}\{x\}) \leq \mathbb{E}\{f(x)\}$ whenever f is convex. Taking $f = \log$, we are able to decompose the log of an expectation into the expectation of a log. This typically separates terms and makes taking derivatives and solving the resolution optimization problem tractable. Unfortunately, EM cannot be directly applied to conditional models (such as the Mega Model) of the form in Eq (7) because such models result in an M-step that requires the maximization of an equation of the form given in Eq (8).

$$\log p\,(y \mid x; \Theta) = \log \sum_z p\,(z, y \mid x; \Theta) = \log \mathbb{E}_{z \sim p(\cdot \mid x, \Theta)} p\,(y \mid x, z; \Theta) \tag{7}$$

$$l = \log \sum_z p\,(z, x, y \mid \Theta) - \log \sum_z p\,(z, x \mid \Theta) \tag{8}$$

Jensen's inequality can be applied to the first term in Eq (8), which can be maximized readily as in standard EM. However, applying Jensen's inequality to the second term would lead to an *upper bound* on the likelihood, since that term appears negated.

The conditional EM solution (Jebara & Pentland, 1998) is to bound the *change* in log-likelihood between iterations, rather than the log-likelihood itself. The change in log-likelihood can be written as in Eq (9), where Θ^t denotes the parameters at iteration t.

$$\Delta l^c = \log p\left(y \mid x; \Theta^t\right) - \log p\left(y \mid x; \Theta^{t-1}\right) \tag{9}$$

By rewriting the conditional distribution $p\,(y \mid x)$ as $p\,(x, y)$ divided by $p\,(x)$, we can express Δl^c as the log of the joint distribution difference minus the log of the marginal distribution. Here, we can apply Jensen's inequality to the first term (the joint difference), but not to the second (because it appears negated). Fortunately, Jensen's is not the only bound we can employ. The standard variational upper bound of the logarithm function is: $\log x \leq x - 1$; this leads to a lower bound of the negation, which is exactly what is desired. This bound is attractive for other reasons: (1) it is tangent to the logarithm; (2) it is tight; (3) it makes contact at the current operating point (according to the maximization at the previous time step); (4) it is a simply linear function; and (5) in the terminology of the calculus of variations, it is the variational dual to the logarithm; see (Smith, 1998).

Applying Jensen's inequality to the first term in Eq (9) and the variational dual to the second term, we obtain that the change of log-likelihood in moving from model parameters Θ^{t-1} at time $t-1$ to Θ^t at time t (which we shall denote Q^t) is bounded by $\Delta l \geq Q^t$, where Q^t is defined by Eq (10), where $h = \mathbb{E}\{z \mid x; \Theta\}$ when $z = 1$ and $1 - \mathbb{E}\{z \mid x; \Theta\}$ when $z = 0$, with expectations taken with respect to the parameters from the previous iteration.

$$Q^t = \sum_{z \in \mathcal{Z}} h_z \log \frac{p\left(z, x, y \mid \Theta^t\right)}{p\left(z, x, y \mid \Theta^{t-1}\right)} - \frac{\sum_z p\left(z, x \mid \Theta^t\right)}{\sum_z p\left(z, x \mid \Theta^{t-1}\right)} + 1 \tag{10}$$

By applying the two bounds (Jensen's inequality and the variational bound), we have removed all "sums of logs," which are hard to deal with analytically. The full derivation is given in Appendix A. The remaining expression is a lower bound on the change in likelihood, and maximization of it will result in maximization of the likelihood.

As in the MAP variant of standard EM, there is no change to the E-step when priors are placed on the parameters. The assumption in standard EM is that we wish to maximize $p\,(\Theta \mid \boldsymbol{x}, \boldsymbol{y}) \propto p\,(\Theta)\, p\,(\boldsymbol{y} \mid \Theta, \boldsymbol{x})$ where the prior probability of Θ is ignored, leaving just the likelihood term of the parameters given the data. In MAP estimation, we do not make this assumption and instead use a true prior $p\,(\Theta)$. In doing so, we need only to add a factor of $\log p\,(\Theta)$ to the definition of Q^t in Eq (10).

j^{t-1}_{n,z_n}	$=$	$\log p\left(x_n, y_n, z_n \mid \Theta^{t-1}\right)$	ψ_{n,z_n}	$=$	$\prod_{f=1}^{F}\left(\psi_f^{z_n}\right)^{x_{nf}}\left(1-\psi_f^{z_n}\right)^{1-x_{nf}}$
m^{t-1}_{n}	$=$	$\left[\sum_{z_n} p\left(x_n, z_n \mid \Theta^{t-1}\right)\right]^{-1}$	$\psi_{n,z_n,-f'}$	$=$	$\prod_{f\neq f'}\left(\psi_f^{z_n}\right)^{x_{nf}}\left(1-\psi_f^{z_n}\right)^{1-x_{nf}}$

Table 1: Notation used for Mega Model equations.

It is important to note that although we do make use of a full joint distribution $p\left(x, y, z\right)$, the *objective function* of our model is *conditional.* The joint distribution is only used in the process of creating the bound: the overall optimization is to maximize the conditional likelihood of the labels *given* the input. In particular, the bound using the full joint likelihood holds for any parameters of the marginal.

5. Parameter Estimation for the Mega Model

As made explicit in Eq (10), the relevant distributions for performing CEM are the full joint distributions over the input variables x, the output variables y, and the hidden variables z. Additionally, we require the marginal distribution over the x variables and the z variables. Finally, we need to compute expectations over the z variables. We will derive the expectation step in this section and present the final solution for the maximization step for each class of variables. The derivation of the equations for the maximization is given in Appendix B.

The Q bound on complete conditional likelihood for the Mega Modelis given below:

$$
\begin{aligned}
Q^t = & \sum_{n=1}^{N^{(i)}}\left[\sum_{z_n^{(i)}} h_n^{(i)} \log \frac{p\left(z_n^{(i)}, \boldsymbol{x}_n^{(i)}, y_n^{(i)}\right)}{p'\left(z_n^{(i)}, \boldsymbol{x}_n^{(i)}, y_n^{(i)}\right)} - \frac{\sum_{z_n^{(i)}} p\left(z_n^{(i)}, \boldsymbol{x}_n^{(i)}\right)}{\sum_{z_n^{(i)}} p'\left(z_n^{(i)}, \boldsymbol{x}_n^{(i)}\right)} + 1\right] \\
& + \sum_{n=1}^{N^{(o)}}\left[\sum_{z_n^{(o)}} h_n^{(o)} \log \frac{p\left(z_n^{(o)}, \boldsymbol{x}_n^{(o)}, y_n^{(o)}\right)}{p'\left(z_n^{(o)}, \boldsymbol{x}_n^{(o)}, y_n^{(o)}\right)} - \frac{\sum_{z_n^{(o)}} p\left(z_n^{(o)}, \boldsymbol{x}_n^{(o)}\right)}{\sum_{z_n^{(o)}} p'\left(z_n^{(o)}, \boldsymbol{x}_n^{(o)}\right)} + 1\right] \qquad (11)
\end{aligned}
$$

In this equation, $p'\left(\right)$ is the probability distribution at the previous iteration. The first term in Eq (11) is the bound for the in-domain data, while the second term is the bound for the out-of-domain data. In all the optimizations described in this section, there are nearly identical terms for the in-domain parameters and the out-of-domain parameters. For brevity, we will only explicitly write the equations for the in-domain parameters; the corresponding out-of-domain equations can be easily derived from these. Moreover, to reduce notational overload, we will elide the superscripts denoting in-domain and out-of-domain when obvious from context. For notational brevity, we will use the notation depicted in Table 1.

5.1 Expectation Step

The E-step is concerned with calculating h_n given current model parameters. Since $z_n \in \{0, 1\}$, we easily find $h_n = p\left(z_n = 1 | \Theta\right)$, which can be calculated as follows:

$$
\begin{aligned}
& p\left(z_n = z \mid \boldsymbol{x}_n, y_n, \boldsymbol{\psi}, \boldsymbol{\lambda}, \pi\right) \\
& \quad = \frac{p\left(z_n = z \mid \pi\right) p\left(\boldsymbol{x}_n \mid \boldsymbol{\psi}, z_n = z\right) p\left(y_n \mid \boldsymbol{\lambda}, z_n = z\right)}{\sum_z p\left(z_n = z \mid \pi\right) p\left(\boldsymbol{x}_n \mid \boldsymbol{\psi}, z_n = z\right) p\left(y_n \mid \boldsymbol{\lambda}, z_n = z\right)} \\
& \quad \propto \pi^z (1-\pi)^{1-z} \psi_{n,z} \frac{1}{Z_{\boldsymbol{x}_n, \boldsymbol{\lambda}^z}} \exp\left[{\boldsymbol{\lambda}_{y_n}^z}^\top \boldsymbol{x}_n\right] \qquad (12)
\end{aligned}
$$

Here, Z is the partition function from before. This can be easily calculated for $z \in \{0, 1\}$ and the expectation can be found by dividing the value for $z = 1$ by the sum over both.

5.2 M-Step for π

As shown in Appendix B.1, we can directly compute the value of π by solving a simple quadratic equation. We can compute π as $-a + \sqrt{a^2 - b}$, where:

$$
\begin{aligned}
a &= \frac{1 - \sum_{n=1}^N \left(2h_n - m_n^{t-1}\left(\psi_{n,0} - \psi_{n,1}\right)\right)}{2\sum_{n=1}^N m_n^{t-1}\left(\psi_{n,0} - \psi_{n,1}\right)} \\
b &= -\frac{\sum_{n=1}^N h_n}{\sum_{n=1}^N m_n^{t-1}\left(\psi_{n,0} - \psi_{n,1}\right)}
\end{aligned}
$$

5.3 M-Step for $\boldsymbol{\lambda}$

Viewing Q^t as a function of $\boldsymbol{\lambda}$, it is easy to see that optimization for this variable is convex. An analytical solution is not available, but the gradient of Q^t with respect to $\boldsymbol{\lambda}^{(\mathsf{i})}$ can be seen to be identical to the gradient of the standard maximum entropy posterior, Eq (4), but where each data point is *weighted* according to its posterior probability, $(1 - h_n)$. We may thus use identical optimization techniques for computing optimal $\boldsymbol{\lambda}$ variables as for standard maximum entropy models; the only difference is that the data points are now weighted. A similar story holds for $\boldsymbol{\lambda}^{(\mathsf{o})}$. In the case of $\boldsymbol{\lambda}^{(\mathsf{g})}$, we obtain the standard maximum entropy gradient, computed over all $N^{(\mathsf{i})} + N^{(\mathsf{o})}$ data points, where each $x_n^{(\mathsf{i})}$ is weighted by h_n and each $x_n^{(\mathsf{o})}$ is weighted by $h_n^{(\mathsf{o})}$. This is shown in Appendix B.2.

5.4 M-Step for $\boldsymbol{\psi}$

Like the case for $\boldsymbol{\lambda}$, we cannot obtain an analytical solution for finding the $\boldsymbol{\psi}$ that maximizes Q^t. However, we can compute simple derivatives for Q^t with respect to a single component $\boldsymbol{\psi}_f$ which can be maximized analytically. As shown in Appendix B.3, we can compute $\psi_f^{(\mathsf{i})}$ as $-a + \sqrt{a^2 - b}$, where:

$$
\begin{aligned}
a &= -\frac{\sum_{n=1}^N \left(1 - h_n + j_{n,0}(1-\pi)\psi_{n,0,-f}\right)}{2\sum_{n=1}^N j_{n,0}(1-\pi)\psi_{n,0,-f}} \\
b &= \frac{1 + \sum_{n=1}^N \left(1 - h_n\right) x_{nf}}{\sum_{n=1}^N j_{n,0}(1-\pi)\psi_{n,0,-f}}
\end{aligned}
$$

```
Algorithm MegaCEM
Initialize $\psi_f^{(\nu)} = 0.5$, $\boldsymbol{\lambda}_f^{(\nu)} = 0$, $\pi^{(\nu)} = 0.5$ for all $\nu \in \{g, i, o\}$ and all $f$.
while parameters haven't converged or iterations remain do

  {- Expectation Step -}
  for $n = 1..N^{(\mathsf{i})}$ do
    Compute the in-domain marginal probabilities, $m_n^{(\mathsf{i})}$
    Compute the in-domain expectations, $h_n^{(\mathsf{i})}$, by Eq (12)
  end for
  for $n = 1..N^{(\mathsf{o})}$ do
    Compute the out-of-domain marginal probabilities, $m_n^{(\mathsf{o})}$
    Compute the out-of-domain expectations, $h_n^{(\mathsf{o})}$ by Eq (12)
  end for

  {- Maximization Step -}
  Analytically update $\pi^{(\mathsf{i})}$ and $\pi^{(\mathsf{o})}$ according to the equations shown in Section 5.2
  Optimize $\boldsymbol{\lambda}^{(\mathsf{i})}$, $\boldsymbol{\lambda}^{(\mathsf{o})}$ and $\boldsymbol{\lambda}^{(\mathsf{g})}$ using BFGS
  while Iterations remain and/or $\boldsymbol{\psi}$ haven't converged do
    Update $\boldsymbol{\psi}$s according to derivation in Section 5.4
  end while

end while
return $\boldsymbol{\lambda}, \boldsymbol{\psi}, \pi$
```

Figure 2: The full training algorithm for the Mega Model.

The case for $\boldsymbol{\psi}^{(\mathsf{o})}$ is identical. For $\boldsymbol{\psi}^{(\mathsf{g})}$, the only difference is that we must replace each sum to over the data points with two sums, one for each of the in-domain and out-of-domain points; and, as before, the $1 - h_n$s must be replaced with h_n; this is made explicit in the Appendix. Thus, to optimize the $\boldsymbol{\psi}$ variables, we simply iterate through and optimize each component analytically, as given above, until convergence.

5.5 Training Algorithm

The full training algorithm is depicted in Figure 2. Convergence properties of the CEM algorithm ensure that this will converge to a (local) maximum in the posterior space. If local optima become a problem in practice, one can alternatively use a stochastic optimization algorithm, in which a temperature is applied enabling the optimization to jump out of local optima early on. However, we do not explore this idea further in this work. In the context of our application, this extension was not required.

5.6 CEM Convergence

One immediate question about the conditional EM model we have described is how many EM iterations are required for the model to converge. In our experiments, 5 iterations of

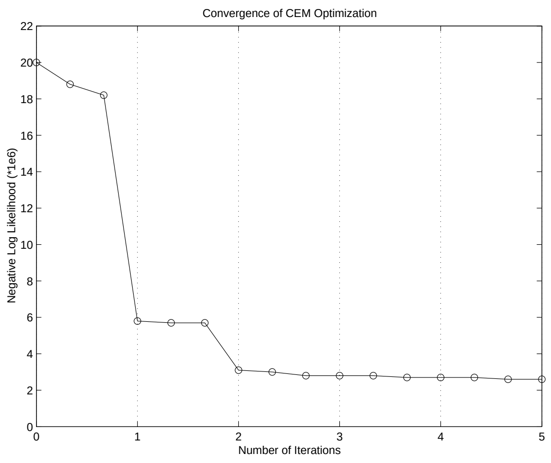

Figure 3: Convergence of training algorithm.

CEM is more than sufficient, and often only 2 or 3 are necessary. To make this more clear, in Figure 3, we have plotted the negative complete log likelihood of the model on the first data set, described below in Section 6.2. There are three separate maximizations in the full training algorithm (see Figure 2); the first involves updating the π variables, the second involves optimizing the $\boldsymbol{\lambda}$ variables and the third involves optimizing the $\boldsymbol{\psi}$ variables. We compute the likelihood after each of these steps.

Running a total 5 CEM iterations is still relatively efficient in our model. The dominating expense is in the weighted maximum entropy optimization, which, at 5 CEM iterations, must be computed 15 times (each iteration requires the optimization of each of the three sets of $\boldsymbol{\lambda}$ variables). At worst this will take 15 times the amount of time to train a model on the complete data set (the union of the in-domain and out-of-domain data), but in practice we can resume each optimization at the ending point of the previous iteration, which causes the subsequent optimizations to take much less time.

5.7 Prediction

Once training has supplied us with model parameters, the subsequent task is to apply these parameters to unseen data to obtain class predictions. We assume all this test data is "in-domain" (i.e., is drawn either from $Q^{(\mathsf{i})}$ or $Q^{(\mathsf{g})}$ in the notation of the introduction), and obtain a decision rule of the form given in Eq (13) for a new test point $\boldsymbol{x}$.

$$\begin{aligned}\hat{y} &= \arg\max_{y\in\mathcal{Y}} p\left(y \mid x; \Theta\right)\\ &= \arg\max_{y\in\mathcal{Y}} \sum_{z} p\left(z \mid x; \Theta\right) p\left(y \mid x, z; \Theta\right)\\ &= \arg\max_{y\in\mathcal{Y}} \sum_{z} p\left(z \mid \Theta\right) p\left(x \mid z; \Theta\right) p\left(y \mid x, z; \Theta\right)\end{aligned}$$

$$
\begin{aligned}
&= \arg\max_{y \in \mathcal{Y}} \ \pi \left[\prod_{f=1}^{F} \left(\psi_f^{(\mathsf{g})}\right)^{x_f} \left(1 - \psi_f^{(\mathsf{g})}\right)^{1-x_f} \right] \frac{\exp\left[\boldsymbol{\lambda}_y^{(\mathsf{g})\top} \boldsymbol{x}\right]}{Z_{\boldsymbol{x},\boldsymbol{\lambda}^{(\mathsf{g})}}} \\
&\quad + (1-\pi) \left[\prod_{f=1}^{F} \left(\psi_f^{(\mathsf{i})}\right)^{x_f} \left(1 - \psi_f^{(\mathsf{i})}\right)^{1-x_f} \right] \frac{\exp\left[\boldsymbol{\lambda}_y^{(\mathsf{i})\top} \boldsymbol{x}\right]}{Z_{\boldsymbol{x},\boldsymbol{\lambda}^{(\mathsf{i})}}} \qquad (13)
\end{aligned}
$$

Thus, the decision rule is to simply select the class which has highest probability according to the maximum entropy classifiers, weighted linearly by the marginal probabilities of the new data point being drawn from $Q^{(\mathsf{i})}$ versus $Q^{(\mathsf{g})}$. In this sense, our model can be seen as linearly interpolating an in-domain model and a general-domain model, but where the interpolation parameter is *input specific*.

6. Experimental Results

In this section, we describe the result of applying the Mega Model to several datasets with varying degrees of divergence between the in-domain and out-of-domain data. However, before describing the data and results, we will discuss the systems against which we compare.

6.1 Baseline Systems

Though there has been little literature on this problem and thus few real systems against which to compare, there are several obvious baselines, which we describe in this section.

OnlyI: This model is obtained simply by training a standard maximum entropy model on the *in-domain* data. This completely ignores the out-of-domain data and serves as a baseline case for when such data is unavailable.

OnlyO: This model is obtained by training a standard maximum entropy model on the *out-of-domain* data, completely ignoring the in-domain data. This serves as a baseline for expected performance without annotating any new data. It also gives a sense of how close the out-of-domain distribution is to the in-domain distribution.

LinI: This model is obtained by linearly interpolating the OnlyI and OnlyO systems. The interpolation parameter is estimated on held-out (development) in-domain data. This means that, in practice, extra in-domain data would need to be annotated in order to create a development set; alternatively, cross-validation could be used.

Mix: This model is obtained by training a maximum entropy model on the union of the out-of-domain and in-domain data sets.

MixW: This model is also obtained by training a maximum entropy model on the union of the out-of-domain and in-domain data sets, but where the out-of-domain data is *down-weighted* so that is effectively equinumerous with the in-domain data.

Feats: This model uses the out-of-domain data to build one classifier and then uses this classifier's predictions as features for the in-domain data, as described by ? (?).

Prior: This is the adaptation model described in Section 2.2, where the out-of-domain data is used to estimate a prior for the in-domain classifier. In the case of the maximum entropy models we consider here, the weights learned from the out-of-domain data are used as the *mean* of the Gaussian prior distribution placed over the weights in the training of the in-domain data, as is described by Chelba and Acero (2004).

In all cases, we tune model hyperparameters using performance on development data. This development data is taken to be a random 20% of the training data in all cases. Once appropriate hyperparameters are found, the 20% is folded back in to the training set.

6.2 Data Sets

We evaluate our models on three different problems. The first two problems come from the Automatic Content Extraction (ACE) data task. This data was selected because the ACE program specifically looks at data in different domains. The third problem is the same as that tackled by Chelba and Acero (2004), which required them to annotate data themselves.

6.2.1 Mention Type Classification

The first problem, **Mention Type**, is a subcomponent of the entity mention detection task (an extension of the named entity tagging task, wherein pronouns and nominals are marked, in addition to simple names). We assume that the extents of the mentions are marked and we simply need to identify their type, one of: Person, Geo-political Entity, Organization, Location, Weapon or Vehicle. As the out-of-domain data, we use the newswire and broadcast news portions of the ACE 2005 training data; as the in-domain data, we use the Fisher conversations data. An example out-of-domain sentence is:

> Once again, a prime battleground will be the constitutional allocation of power – between the federal $\underline{\text{government}}^{\text{NOM}}_{\text{GPE}}$ and the $\underline{\text{states}}^{\text{NOM}}_{\text{GPE}}$, and between $\underline{\text{Congress}}^{\text{NAM}}_{\text{ORG}}$ and federal regulatory $\underline{\text{agencies}}^{\text{BAR}}_{\text{ORG}}$.

An example in-domain sentence is:

> $\underline{\text{my}}^{\text{PRO}}_{\text{PER}}$ $\underline{\text{wife}}^{\text{NOM}}_{\text{PER}}$ if $\underline{\text{I}}^{\text{PRO}}_{\text{PER}}$ had not been transported across the $\underline{\text{continent}}^{\text{NOM}}_{\text{GPE}}$ from $\underline{\text{where}}^{\text{WHQ}}_{\text{LOC}}$ $\underline{\text{I}}^{\text{PRO}}_{\text{PER}}$ was born and and

We use $23k$ out-of-domain examples (each mention corresponds to one example), $1k$ in-domain examples and 456 test examples. Accuracy is computed as 0/1 loss. We use the standard feature functions employed in named entity models, include lexical items, stems, prefixes and suffixes, capitalization patterns, part-of-speech tags, and membership information on gazetteers of locations, businesses and people. The accuracies reported are the result of running ten fold cross-validation.

6.2.2 Mention Tagging

The second problem, **Mention Tagging** is the precursor to the **Mention Type** task, in which we attempt to tag entity mentions in raw text. We use the standard Begin/In/Out encoding and use a maximum entropy Markov model to perform the tagging (McCallum et al., 2000). As the out-of-domain data, we use again the newswire and broadcast news data; as the in-domain data, we use broadcast news data that has been transcribed by automatic speech recognition. The in-domain data lacks capitalization, punctuation, etc., and also contains transcription errors (speech recognition word error rate is approximately 15%). For the tagging task, we have $112k$ out-of-domain examples (in the context of tagging, an example is a single word), but now $5k$ in-domain examples and $11k$ test examples. Accuracy is F-measure across the segmentation. We use the same features as in the mention type identification task. The scores reported are after ten fold cross-validation.

6.2.3 Recapitalization

The final problem, **Recap**, is the task of recapitalizing text. Following Chelba and Acero (2004), we again use a maximum entropy Markov model, where the possible tags are: Lowercase, Capitalized, All Upper Case, Punctuation or Mixed case. The out-of-domain data in this task comes from the Wall Street Journal, and two separate in-domain data sets come from broadcast news text from CNN/NPR and ABC Primetime, respectively. We use $3.5m$ out-of-domain examples (one example is one word). For the CNN/NPR data, we use $146k$ in-domain training examples and $73k$ test examples; for the ABC Primetime data, we use $33k$ in-domain training examples and $8k$ test examples. We use identical features to Chelba and Acero (2004). In order to maintain comparability to the results described by Chelba and Acero (2004), we do not perform cross-validation for these experiments: we use the same train/test split as described in their paper.

6.3 Feature Selection

While the maximum entropy models used for the classification are adept at dealing with many irrelevant and/or redundant features, the naïve Bayes generative model, which we use to model the distribution of the input variables, can overfit on such features. This turned out not to be a problem for the **Mention Type** and **Mention Tagging** problems, but for the **Recap** problems, it caused some errors. To alleviate this problem, for the **Recap** problem *only*, we applied a feature selection algorithm just to the features used for the naïve Bayes model (the entire feature set was used for the maximum entropy model). Specifically, we took the $10k$ top features according to the information gain criteria to predict "in-domain" versus "out-of-domain" (as opposed to feature selection for class label); Forman (2003) provides an overview of different selection techniques.[1]

6.4 Results

Our results are shown in Table 2, where we can see that training only on in-domain data always outperforms training only on out-of-domain data. The linearly interpolated model does not improve on the base models significantly. Placing all the data in one bag helps, and there is no clear advantage to re-weighting the out domain data. The Prior model and the Feats model perform roughly comparably, with the Prior model edging out by a small margin.[2] Our model outperforms both the Prior model and the Feats model.

1. The value of $10k$ was selected arbitrarily after an initial run of the model on development data; it was not tuned to optimize either development or test performance.
2. Our numbers for the result of the Prior model on the data from Chelba and Acero (2004) differ slightly from those reported in their paper. There are two potential reasons for this. First, most of their numbers

	Mention Type	**Mention Tagging**	**Recap ABC**	**Recap CNN**	**Average**
$\lvert\mathcal{D}^{(o)}\rvert$	$23k$	$112k$	$3.5m$	$3.5m$	-
$\lvert\mathcal{D}^{(i)}\rvert$	$1k$	$5k$	$8k$	$73k$	-
Accuracy					
OnlyO	57.6	78.3	95.5	94.6	81.5
OnlyI	81.2	83.5	97.4	94.7	89.2
LinI	81.5	83.8	97.7	94.9	89.5
Mix	84.9	80.9	96.4	95.0	89.3
MixW	81.3	81.0	97.6	93.5	88.8
Feats	87.8	84.2	97.8	96.1	91.5
Prior	87.9	85.1	97.9	95.9	91.7
MegaM	92.1	88.2	98.1	96.8	93.9
% Reduction					
Mix	47.7	38.2	52.8	36.0	43.0
Prior	34.7	20.8	19.0	22.0	26.5

Table 2: Experimental results; The first set of rows show the sizes of the in-domain and out-of-domain training data sets. The second set of rows (Accuracy) show the performance of the various models on each of the four tasks. The last two rows (% Reduction) show the percentage reduction in error rate by using the Mega Model over the baseline model (**Mix**) and the best alternative method (**Prior**).

We applied McNemar's test (Gibbons & Chakraborti, 2003, section 14.5) to gage statistical significance of these results, comparing the results of the Prior model with our own Mega Model (for the mention tagging experiment, we compute McNemar's test on simple Hamming accuracy rather than F-score; this is suboptimal, but we do not know how to compute statistical significance for the F-score). For the mention type task, the difference is statistical significant at the $p \leq 0.03$ level; for the mention tagging task, $p \leq 0.001$; for the recapitalization tasks, the difference on the ABC data is significant only at the $p \leq 0.06$ level, while for the CNN/NPR data it is significant at the $p \leq 0.004$ level.

In the mention type task, we have improved a baseline model trained only on in-domain data from an accuracy of 81.2% up to 92.1%, a relative improvement of 13.4%. For mention tagging, we improve from 83.5% F-measure up to 88.2%, a relative improvement of 5.6%. In the ABC recapitalization task (for which much in-domain data is available), we increase performance from 95.5% to 98.1%, a relative improvement of 2.9%. In the CNN/NPR recapitalization task (with very little in-domain data), we increase performance from 94.6% to 96.8%, a relative improvement of 2.3%.

are reported based on using all $20m$ examples; we consider only the $3.5m$ example case. Second, there are likely subtle differences in the training algorithms used. Nevertheless, on the whole, our relative improvements agree with those in their paper.

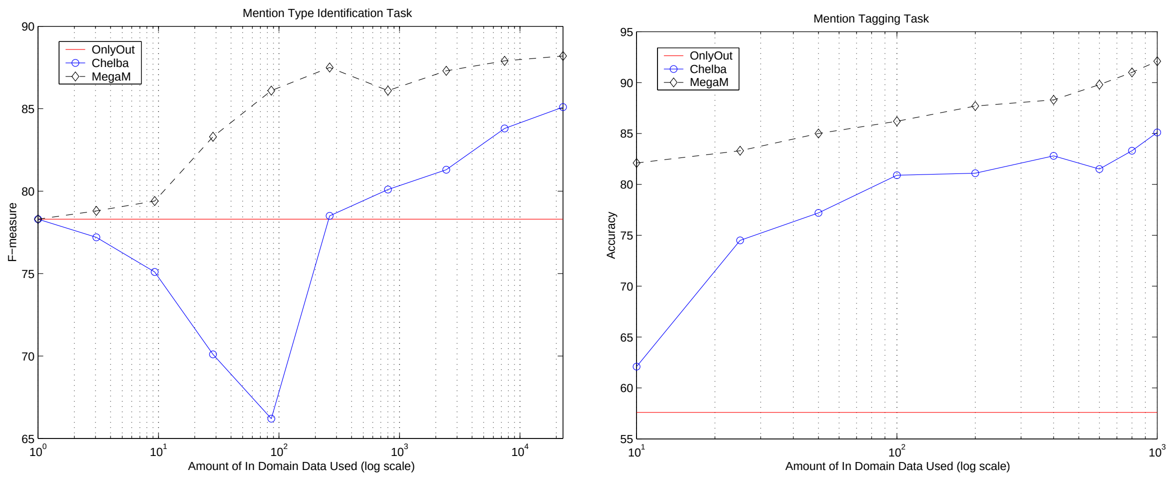

Figure 4: Learning curves for **Prior** and **MegaM** models.

6.5 Learning Curves

Of particular interest is the amount of annotated *in-domain* data needed to see a marked improvement from the **OnlyO** baseline to a well adapted system. We show in Figure 4 the learning curves on the **Mention Type** and **Mention Tagging** problems. Along the x-axis, we plot the amount of in-domain data used; along the y-axis, we plot the accuracy. We plot three lines: a flat line for the **OnlyO** model that does not use any in-domain data, and curves for the **Prior** and **MegaM** models. As we can see, our model maintains an accuracy above both the other models, while the Prior curve actually falls below the baseline in the type identification task.[3]

7. Model Introspection

We have seen in the previous sections that the Mega Model routinely outperforms competing models. Despite this clear performance improvement, a question remains open regarding the internal workings of the models. The $\pi^{(\mathrm{i})}$ variable captures the degree to which the in-domain data set is truly in-domain. The z variables in the model aim to capture, for each test data point, whether it is "general domain" or "in-domain." In this section, we discuss the particular values of the parameters the model learns for these variables.

We present two analyses. In the first (Section 7.1), we inspect the model's inner workings on the Mention Type task from Section 6.2.1. In this analysis, we look specifically at the expected values of the hidden variables found by the model. In the second analysis (Section 7.2), we look at the ability of the model to judge degree of relatedness, as defined by the π variables.

3. This is because the Fisher data is personal conversations. It hence has a much higher degree of first and second person pronouns than news. (The baseline that always guesses "person" achieves a 77.8% accuracy.) By not being able to intelligently use the out-of-domain data only when the in-domain model is unsure, performance drops, as observed in the **Prior** model.

Pre-context …	Entity	… Post-context	True	Hyp	$p(z = \mathrm{I})$
my home is in trenton …	new jersey	… and that's where	GPE	GPE	0.02
veteran's administration …	hospital	…	ORG	LOC	0.11
you know by the american …	government	… because what is	ORG	ORG	0.17
gives …	me	… chills because if	PER	PER	0.71
is he capable of getting …	anything	… over here	WEA	PER	0.92
the fisher thing calling …	me	… ha ha they screwed up	PER	PER	0.93
when i was a …	kid	… that that was a	PER	PER	0.98

Table 3: Examples from the test data for the Mention Type task. The "True" column is the correct entity type and the "Hyp" column is our model's prediction. The final column is the probability this example is truly in-domain under our model.

7.1 Model Expectations

To focus our discussion, we will consider only the Mention Type task, Section 6.2.1. In Table 3, we have shown seven test-data examples from the Mention Type task. The Pre-context is the text that appears before the entity and the post-context is the text that appears after. We report the true class and the class our model hypothesizes. Finally, we report the probability of this example being truly in-domain, according to our model.

As we can see, the three examples that the model thinks are general domain are "new jersey," "hospital" and "government." It believes that "me," "anything" and "kid" are all in-domain. In general, the probabilities tend to be skewed toward 0 and 1, which is not uncommon for naïve Bayes models. We have shown two errors in this data. In the first, our model thinks that "hospital" is a location when truly it is an organization. This is a difficult distinction to make: in the training data, hospitals were often used as locations.

The second example error is "anything" in "is he capable of getting anything over here." The long-distance context of this example is a discussion about biological warfare and Saddam Hussein, and "anything" is supposed to refer to a type of biological warhead. Our model mistakingly thinks this is a person. This error is likely due to the fact that our model identifies that the word "anything" is likely to be truly in-domain (the word is not so common in newswire). It has also learned that most truly in-domain entities are people. Thus, lacking evidence otherwise, the model incorrectly guesses that "anything" is a person.

It is interesting to observe that the model believes that the entity "me" in "gives me chills" is closer to general domain than the "me" in "the fisher thing calling me ha ha they screwed up." This likely occurs because the context "ha ha" has not occurred anywhere in the out-of-domain training data, and twice in the in-domain training data. It is unlikely this example would have been misclassified otherwise ("me" is fairly clearly a person), but this example shows that our model is able to take context into account in deciding the domain.

All of the decisions made by the model, shown in Table 3 seem qualitatively reasonable. The numbers are perhaps excessively skewed, but the ranking is believable. The in-domain data is primarily from conversations about random (not necessarily news worthy) topics, and is hence highly colloquial. Contrastively, the out-of-domain data is from formal news. The model is able to learn that entities like "new jersey" and "government" have more to do with news that words like "me" and "kid."

	Mention Type	**Mention Tagging**	**Recap CNN**	**Recap ABC**
$\pi^{(\mathsf{i})}$	0.14	0.41	0.36	0.51
$\pi^{(\mathsf{o})}$	0.11	0.45	0.40	0.69

Table 4: Values for the π variables discovered by the Mega Model algorithm.

7.2 Degree of Relatedness

In this section, we analyze the values of π found by the model. Low values of $\pi^{(\mathsf{i})}$ and $\pi^{(\mathsf{o})}$ mean that the in-domain data was significantly different than the out-of-domain data; high values mean that they were similar. This is because a high value for π means that the general domain model will be used in most cases. For all tasks but Mention Type, the values of π were middling around 0.4. For Mention Type, $\pi^{(\mathsf{i})}$ was 0.14 and $\pi^{(\mathsf{o})}$ was 0.11, indicating that there was a significant difference between the in-domain and out-of-domain data. The exact values for all tasks are shown in Table 4.

These values for π make intuitive sense. The distinction between conversation data and news data (for the Mention Type task) is significantly stronger than the difference between manually and automatically transcribed newswire (for the Mention Tagging task). The values for π reflect this qualitative distinction. The rather strong difference between the π values for the recapitalization tasks was not expected a priori. However, a post hoc analysis shows this result is reasonable. We compute the KL divergence between a unigram language model for the out-of-domain data set and each of the in-domain data sets. The KL divergence for the CNN data was 0.07, while the divergence for the ABC data 0.11. This confirms that the ABC data is perhaps more different from the baseline out-of-domain than the CNN data, as reflected by the π values.

We are also interested in cases where there is little difference between in-domain and out-of-domain data. To simulate this case, we have performed the following experiment. We consider again the Mention Type task, but use *only* the training portion of the out-of-domain data. We randomly split the data in half, assigning each half to "in-domain" and "out-of-domain." In theory, the model should learn that it may rely only on the general domain model. We performed this experiment under ten fold cross-validation and found that the average value of π selected by the model was 0.94. While this is strictly less than one, it does show that the model is able to identify that these are very similar domains.

8. Conclusion and Discussion

In this paper, we have presented the Mega Model for domain adaptation in the discriminative (conditional) learning framework. We have described efficient optimization algorithms based on the conditional EM technique. We have experimentally shown, in four data sets, that our model outperforms a large number of baseline systems, including the current state of the art model, and does so requiring significantly less in-domain data.

Although we focused specifically on discriminative modeling in a maximum entropy framework, we believe the novel, basic idea on which this work is founded—to break the in-domain distribution $p^{(\mathsf{i})}$ and out-of-domain distribution $p^{(\mathsf{o})}$ into three distributions, $q^{(\mathsf{i})}$,

$q^{(o)}$ and $q^{(g)}$—is general. In particular, one could perform a similar analysis in the case of generative models and obtain similar algorithms (though in the case of a generative model, standard EM could be used). Such a model could be applied to domain adaptation in language modeling or machine translation.

With the exception of the work described in Section 2.2, previous work in-domain adaptation is quite rare, especially in the discriminative learning framework. There is a substantial literature in the language modeling/speech community, but most of the adaptation with which they are concerned is based on adapting to new speakers (Iyer, Ostendorf, & Gish, 1997; Kalai, Chen, Blum, & Rosenfeld, 1999). From a learning perspective, the Mega Model is most similar to a mixture of experts model. Our model can be seen as a *constrained* experts model, with three experts, where the constraints specify that in-domain data can only come from one of two experts, and out-of-domain data can only come from one of two experts (with a single expert overlapping between the two). Most attempts to build discriminative mixture of experts models make heuristic approximations in order to perform the necessary optimization (Jordan & Jacobs, 1994), rather than apply conditional EM, which gives us strict guarantees that we monotonically increase the data (incomplete) log likelihood of each iteration in training.

The domain adaptation problem is also closely related to multitask learning (also known as learning to learn and inductive transfer). In multitask learning, one attempts to learn a function that solves many machine learning problems simultaneously. This related problem is discussed by Thrun (1996), Caruana (1997) and Baxter (2000), among others. The similarity between multitask learning and domain adaptation is that they both deal with data drawn from related, but distinct distributions. The primary difference is that domain adaptation cares only about predicting one label type, while multitask learning cares about predicting many.

As the various sub-communities of the natural language processing family begin and continue to branch out into domains other than newswire, the importance of developing models for new domains without annotating much new data will become more and more important. The Mega Model is a first step toward being able to migrate simple classification-style models (classifiers and maximum entropy Markov models) across domains. Continued research in the area of adaptation is likely to benefit from other work done in active learning and in learning with large amounts unannotated data.

Acknowledgments

We thank Ciprian Chelba and Alex Acero for making their data available. We thank Ryan McDonald for pointing out the Feats baseline, which we had not previously considered. We also thank Kevin Knight and Dragos Munteanu for discussions related to this project. This paper was greatly improved by suggestions from reviewers, including reviewers of a previous, shorter version. This work was partially supported by DARPA-ITO grant N66001-00-1-9814, NSF grant IIS-0097846, NSF grant IIS-0326276, and a USC Dean Fellowship to Hal Daumé III.

Appendix A. Conditional Expectation Maximization

In this appendix, we derive Eq (10) from Eq (7) by making use of Jensen's inequality and the variational bound. The interested reader is referred to the work of Jebara and Pentland (1998) for further details. Our discussion will consider a bound in the *change* of the log likelihood between iteration $t-1$ and iteration t, Δl^c, as given in Eq (14):

$$\begin{aligned} \Delta l^c &= \log \frac{p\left(y \mid x; \Theta^t\right)}{p\left(y \mid x; \Theta^{t-1}\right)} = \log \frac{p\left(x, y \mid \Theta^t\right)/p\left(y \mid \Theta^t\right)}{p\left(x, y \mid \Theta^{t-1}\right)/p\left(y \mid \Theta^{t-1}\right)} && (14) \\ &= \log \frac{p\left(x, y; \Theta^t\right)}{p\left(x, y; \Theta^{t-1}\right)} - \log \frac{p\left(x; \Theta^t\right)}{p\left(x; \Theta^{t-1}\right)} && (15) \end{aligned}$$

Here, we have effectively rewritten the log-change in the ratio of the conditionals as the difference between the log-change in the ratio of the joints and the log-change in the ratio of the marginals. We may rewrite Eq (15) by introducing the hidden variables z as:

$$\Delta l^c = \log \frac{\sum_z p\left(x, y, z; \Theta^t\right)}{\sum_z p\left(x, y, z; \Theta^{t-1}\right)} - \log \frac{\sum_z p\left(x, z; \Theta^t\right)}{\sum_z p\left(x, z; \Theta^{t-1}\right)} \tag{16}$$

We can now apply Jensen's inequality to the first term in Eq (16) to obtain:

$$\Delta l^c \geq \sum_z \underbrace{\left[\frac{p\left(x, y, z \mid \Theta^{t-1}\right)}{\sum_{z'} p\left(x, y, z \mid \Theta^{t-1}\right)}\right]}_{h_{x,y,z,\Theta^{t-1}}} \log \frac{p\left(x, y, z; \Theta^t\right)}{p\left(x, y, z; \Theta^{t-1}\right)} - \log \frac{\sum_z p\left(x, z; \Theta^t\right)}{\sum_z p\left(x, z; \Theta^{t-1}\right)} \tag{17}$$

In Eq (17), the expression denoted $h_{x,y,z,\Theta^{t-1}}$ is the joint expectation of z under the previous iteration's parameter settings. Unfortunately, we cannot also apply Jensen's inequality to the remaining term in Eq (17) because it appears negated. By applying the variational dual ($\log x \leq x - 1$) to this term, we obtain the following, final bound:

$$\Delta l^c \geq Q^t = \sum_z h_{x,y,z,\Theta^{t-1}} \log \frac{p\left(x, y, z; \Theta^t\right)}{p\left(x, y, z; \Theta^{t-1}\right)} - \frac{\sum_z p\left(x, z; \Theta^t\right)}{\sum_z p\left(x, z; \Theta^{t-1}\right)} + 1 \tag{18}$$

Applying the bound from Eq (18) to the distributions chosen in our model yields Eq (10).

Appendix B. Derivation of Estimation Equations

Given the model structure and parameterization of the Mega Modelgiven in Section 3.2, Eq (5), we obtain the following expression for the joint probability of the data:

$$\begin{aligned}
p & \left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{\psi}^{(\nu)}, \boldsymbol{\lambda}^{(\nu)}, \pi\right) \\
&= \left\{ \prod_{n=1}^{N} \mathcal{B}er(z_n \mid \pi) \prod_{f=1}^{F} \mathcal{B}er(x_{nf} \mid \boldsymbol{\psi}_f^{z_n}) \, Gibbs(y_n \mid \boldsymbol{x}_n, \boldsymbol{\lambda}^{z_n}) \right\} \\
&= \left\{ \prod_{n=1}^{N} \pi^{z_n} (1-\pi)^{1-z_n} \left[\prod_{f=1}^{F} \left(\boldsymbol{\psi}_f^{z_n}\right)^{x_{nf}} \left(1 - \boldsymbol{\psi}_f^{z_n}\right)^{1-x_{nf}} \right] \right. \\
& \qquad \left. \exp\left[\boldsymbol{\lambda}_{y_n}^{z_n \top} \boldsymbol{x}_n \right] \left(\sum_c \exp\left[\boldsymbol{\lambda}_c^{z_n \top} \boldsymbol{x}_n \right] \right)^{-1} \right\} \qquad (19)
\end{aligned}$$

The marginal distribution is obtained by removing the last two terms (the exp and the sum of exps) from the final equation. Plugging Eq (19) into Eq (10) and using the notation from Eq (12), we obtain the following expression for Q^t:

$$\begin{aligned}
Q^t &= \sum_{\nu} \left[\log \mathcal{N}or(\boldsymbol{\lambda}^{(\nu)}; 0, \sigma^2 I) + \sum_{f=1}^{F} \log \mathcal{B}et(\boldsymbol{\psi}_f^{(\nu)}; a, b) \right] \\
&+ \sum_{n=1}^{N} \left[\sum_{z_n} h_n \left\{ z_n \log \pi + (1-z_n) \log(1-\pi) + \log \psi_{n,z_n} \right. \right. \\
& \qquad \left. + \sum_{f=1}^{F} x_{nf} \log \boldsymbol{\lambda}_{y_n}^{z_n} - \log \sum_c \exp\left[\boldsymbol{\lambda}_c^{z_n \top} x_n \right] - j_{n,z_n}^t \right\} \\
& \qquad \left. - m_n^{t-1} \pi^{z_n} (1-\pi)^{1-z_n} \psi_{n,z_n} + 1 \right] \qquad (20)
\end{aligned}$$

as well as an analogous term for the out-of-domain data. j and m are defined in Table 1.

B.1 M-Step for π

For computing π, we simply differentiate Q^t (see Eq (20)) with respect to π, obtaining:

$$\frac{\partial Q^t}{\partial \pi} = \sum_{n=1}^{N} \frac{h_n}{\pi} + \frac{1-h_n}{1-\pi} + m_n^{t-1} \left(\psi_{n,0} - \psi_{n,1}\right) \qquad (21)$$

solving this for 0 leads directly to a quadratic expression of the form:

$$\begin{aligned}
0 &= \pi^2 \left[\sum_{n=1}^{N} m_n^{t-1} \left(\psi_{n,0} - \psi_{n,1}\right) \right] \\
&+ \pi^1 \left[-1 + \sum_{n=1}^{N} \left(2h_n - m_n^{t-1} \left(\psi_{n,0} - \psi_{n,1}\right) \right) \right]
\end{aligned}$$

$$+ \quad \pi^0 \left[-\sum_{n=1}^{N} h_n \right] \tag{22}$$

Solving this directly for π gives the desired update equation.

B.2 M-Step for $\boldsymbol{\lambda}$

For optimizing $\boldsymbol{\lambda}^{(\mathsf{i})}$, we rewrite Q^t, Eq (20), neglecting all irrelevant terms, as:

$$Q^t[\boldsymbol{\lambda}] = \sum_{n=1}^{N} (1 - h_n) \left\{ \sum_{f=1}^{F} x_{nf} \boldsymbol{\lambda}_{y_n,f} - \log \sum_{c} \exp \left[\boldsymbol{\lambda}_c{}^{\top} x_n \right] \right\} + \log \mathcal{N}or(\boldsymbol{\lambda}; 0, \sigma^2 I) \tag{23}$$

In Eq (23), the bracketed expression is exactly the log-likelihood term obtained for standard logistic regression models. Thus, the optimization of Q with respect to $\boldsymbol{\lambda}^{(\mathsf{i})}$ and $\boldsymbol{\lambda}^{(\mathsf{o})}$ can be performed using a weighted version of standard logistic regression optimization, with weights defined by $(1-h_n)$. In the case of $\boldsymbol{\lambda}^{(\mathsf{g})}$, we obtain a weighted logistic regression model, but over all $N^{(\mathsf{i})} + N^{(\mathsf{o})}$ data points, and with weights defined by h_n.

B.3 M-Step for $\boldsymbol{\psi}$

In the case of $\boldsymbol{\psi}^{(\mathsf{i})}$ and $\boldsymbol{\psi}^{(\mathsf{o})}$, we rewrite Eq (20) and remove all irrelevant terms, as:

$$Q^t[\boldsymbol{\psi}^{(\mathsf{i})}] = \sum_{f=1}^{F} \log \mathcal{B}et(\psi_f; a, b) + \sum_{n=1}^{N} \left[(1 - h_n) \log \psi_{n,0} - m_n^{t-1} (1 - \pi) \psi_{n,0} \right] \tag{24}$$

Due to the presence of the product term in $\boldsymbol{\psi}$, we cannot compute an analytical solution to this maximization problem. However, we can take derivatives component-wise (in F) and obtain analytical solutions (when combined with the prior). This admits an iterative solution for maximizing $Q^t_{\boldsymbol{\psi}}$ by maximizing each component separately until convergence. Computing derivatives of Q^t with respect to ψ_f requires differentiating $\psi_{n,0}$ with respect to ψ_f; this has a convenient form (recalling the notation from Table 1:

$$\frac{\partial}{\partial \psi_f} \psi_{n,0} = [\psi_{n,0,-f}] \frac{\partial}{\partial \psi_f} \left\{ x_{nf} \psi_f + (1 - x_{nf})(1 - \psi_f) \right\} = \psi_{n,0,-f} \tag{25}$$

Using this result, we can maximize Q^t with respect to ψ_f by solving:

$$\begin{aligned} \frac{\partial}{\partial \psi_f} \left[Q^t_{\psi_f} \right] &= \sum_{n=1}^{N} \left[(1 - h_n) \frac{x_{nf}(1 - \psi_f) - (1 - x_{nf}) \psi_f}{\psi_f (1 - \psi_f)} \right. \\ &\qquad \left. - j_{n,0} (1 - \pi) \psi_{n,0,-f} \right] + \frac{1}{\psi_f (1 - \psi_f)} \\ &= \frac{1}{\psi_f (1 - \psi_f)} \left[1 + \sum_{n=1}^{N} (1 - h_n)(x_{nf} - \psi_f) \right] - \sum_{n=1}^{N} j_{n,0} (1 - \pi) \psi_{n,0,-f} \end{aligned} \tag{26}$$

Equating this to zero yields a quadratic expression of the form:

$$\begin{aligned} 0 &= (\psi_f)^2 \left[\sum_{n=1}^{N} j_{n,0}(1-\pi)\psi_{n,0,-f} \right] \\ &+ (\psi_f)^1 \left[-\sum_{n=1}^{N} \left(1 - h_n + j_{n,0}(1-\pi)\psi_{n,0,-f} \right) \right] \\ &+ (\psi_f)^0 \left[1 + \sum_{n=1}^{N} (1-h_n)\, x_{nf} \right] \end{aligned} \tag{27}$$

This final equation can be solved analytically. A similar expression arises for $\psi_f^{(\mathsf{o})}$. In the case of $\psi_f^{(\mathsf{g})}$, we obtain a quadratic form with sums over the entire data set and with h_n replacing the occurrences of $(1-h_n)$:

$$\begin{aligned} 0 &= \left(\psi_f^{(\mathsf{g})}\right)^2 \left[\sum_{n=1}^{N^{(\mathsf{i})}} j_{n,1}^{(\mathsf{i})} \pi^{(\mathsf{i})} \psi_{n,1,-f}^{(\mathsf{i})} + \sum_{n=1}^{N^{(\mathsf{o})}} j_{n,1}^{(\mathsf{o})} \pi^{(\mathsf{o})} \psi_{n,1,-f}^{(\mathsf{o})} \right] \\ &+ \left(\psi_f^{(\mathsf{g})}\right)^1 \left[-\sum_{n=1}^{N^{(\mathsf{i})}} \left(h_n^{(\mathsf{i})} + j_{n,1}^{(\mathsf{i})} \pi^{(\mathsf{i})} \psi_{n,1,-f}^{(\mathsf{i})} \right) - \sum_{n=1}^{N^{(\mathsf{o})}} \left(h_n^{(\mathsf{o})} + j_{n,1}^{(\mathsf{o})} \pi^{(\mathsf{o})} \psi_{n,1,-f}^{(\mathsf{o})} \right) \right] \\ &+ \left(\psi_f^{(\mathsf{g})}\right)^0 \left[1 + \sum_{n=1}^{N^{(\mathsf{i})}} h_n^{(\mathsf{i})} x_{nf}^{(\mathsf{i})} + \sum_{n=1}^{N^{(\mathsf{o})}} h_n^{(\mathsf{o})} x_{nf}^{(\mathsf{o})} \right] \end{aligned} \tag{28}$$

Again, this can be solved analytically. The values j, m, $\psi_{\cdot,\cdot}$ and $\psi_{\cdot,\cdot,-\cdot}$ are defined in Table 1.

References

Averick, B. M., & Moré, J. J. (1994). Evaluation of large-scale optimization problems on vector and parallel architectures. *SIAM Journal of Optimization*, *4*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Bacchiani, M., & Roark, B. (2003). Unsupervised langauge model adaptation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Caruana, R. (1997). Multitask learning: A knowledge-based source of inductive bias. *Machine Learning*, *28*, 41–75.

Chelba, C., & Acero, A. (2004). Adaptation of maximum entropy classifier: Little data can help a lot. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Chen, S., & Rosenfeld, R. (1999). A Gaussian prior for smoothing maximum entropy models. Tech. rep. CMUCS 99-108, Carnegie Mellon University, Computer Science Department.

Della Pietra, S., Della Pietra, V. J., & Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B39*.

Elhadad, N., Kan, M.-Y., Klavans, J., & McKeown, K. (2005). Customization in a unified framework for summarizing medical literature. *Journal of Artificial Intelligence in Medicine*, *33*(2), 179–198.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Gibbons, J. D., & Chakraborti, S. (2003). *Nonparametric Statistical Inference*. Marcel Dekker, Inc.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hobbs, J. R. (2002). Information extraction from biomedical text. *Journal of Biomedical Informatics*, *35*(4), 260–264.

Huang, J., Zweig, G., & Padmanabhan, M. (2001). Information extraction from voicemail. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Hwa, R. (1999). Supervised grammar induction using training data with limited constituent information. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pp. 73–79.

Iyer, R., Ostendorf, M., & Gish, H. (1997). Using out-of-domain data to improve in-domain language models. *IEEE Signal Processing*, *4*(8).

Jebara, T., & Pentland, A. (1998). Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Advances in Neural Information Processing Systems (NIPS)*.

Jordan, M., & Jacobs, R. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*, 181–214.

Kalai, A., Chen, S., Blum, A., & Rosenfeld, R. (1999). On-line algorithms for combining language models. In *ICASSP*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL*.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Minka, T. P. (2003). A comparison of numerical optimizers for logistic regression. `http://www.stat.cmu.edu/~minka/papers/logreg/`.

Munteanu, D., & Marcu, D. (2005). Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, *To appear*.

Nash, S., & Nocedal, J. (1991). A numerical study of the limited memory BFGS method and the truncated Newton method for large scale optimization. *SIAM Journal of Optimization*, *1*, 358–372.

Rambow, O., Shrestha, L., Chen, J., & Lauridsen, C. (2004). Summarizing email threads. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) Short Paper Section*.

Roark, B., & Bacchiani, M. (2003). Supervised and unsupervised PCFG adaptation to novel domains. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT)*.

Smith, D. R. (1998). *Variational Methods in Optimization*. Dover Publications, Inc., Mineola, New York.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first. In *Advances in Neural Information Processing Systems (NIPS)*.