Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition

Andrew Arnold, Ramesh Nallapati and William W. Cohen
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA
{aarnold, nmramesh, wcohen}@cs.cmu.edu

Abstract

We present a novel hierarchical prior structure for supervised transfer learning in named entity recognition, motivated by the common structure of feature spaces for this task across natural language data sets. The problem of transfer learning, where information gained in one learning task is used to improve performance in another related task, is an important new area of research. In the subproblem of domain adaptation, a model trained over a source domain is generalized to perform well on a related target domain, where the two domains' data are distributed similarly, but not identically. We introduce the concept of groups of closely-related domains, called *genres*, and show how inter-genre adaptation is related to domain adaptation. We also examine multitask learning, where two domains may be related, but where the concept to be learned in each case is distinct. We show that our prior conveys useful information across domains, genres and tasks, while remaining robust to spurious signals not related to the target domain and concept. We further show that our model generalizes a class of similar hierarchical priors, smoothed to varying degrees, and lay the groundwork for future exploration in this area.

1 Introduction

1.1 Problem definition

Consider the task of *named entity recognition* (NER). Specifically, you are given a corpus of news articles in which all tokens have been labeled as either belonging to personal name mentions or not. 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 news data, now consider the problem of learning to classify tokens as names in e-mail data. An intuitive solution might be to simply retrain the classifier, *de novo*, on the e-mail data. Practically, however, large, labeled datasets are often expensive to build and this solution would not scale across a large number of different datasets.

Clearly the problems of identifying names in news articles and e-mails 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 news problem, may prove less informative in the e-mail data since the rules of capitalization are followed less strictly in that domain.

These are the problems we address in this paper. In particular, we develop a novel prior for named entity recognition that exploits the hierarchical feature space often found in natural language domains (§1.2) and allows for the transfer of information from labeled datasets in other domains (§1.3). §2 introduces the *maximum entropy* (maxent) and *conditional random field* (CRF) learning techniques employed, along with specifications for the design and training of our hierarchical prior. Finally, in §3 we present an empirical investigation of our prior's performance against a number of baselines, demonstrating both its effectiveness and robustness.

1.2 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:

```
   O   O   O   O        O          I
Give the book to Professor Caldwell
```
(1)

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 1 (zoomed in on the section of the tree where the *L.1.Professor* feature resides).

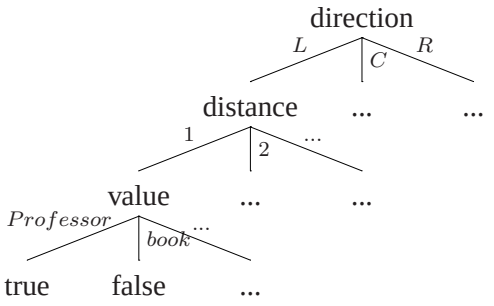

Figure 1: Graphical representation of a hierarchical feature tree for token `Caldwell` in example Sentence 1.

Representing feature spaces with this kind of tree, besides often coinciding with the explicit language used by common natural language toolkits (Cohen, 2004), 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 1 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 (Minkov et al., 2005). 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 1.

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

Table 1: A few examples of the feature hierarchy

1.3 Transfer learning

When only the type of data being examined is allowed to vary (from news articles to e-mails, for example), the problem is called *domain adaptation* (Daumé III and Marcu, 2006). When the task being learned varies (say, from identifying person names to identifying protein names), the problem is called *multi-task learning* (Caruana, 1997). Both of these are considered specific types of the overarching *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 paper 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* (Ando and Zhang, 2005; Caruana, 1997; Sutton and McCallum, 2005; Zhang et al., 2005) 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 (Raina et al., 2006). In the example from §1.1, our belief that capitalization is less strict in e-mails than in news articles could be encoded in a prior that biased the importance of the `capitalization` feature to be lower for e-mails than news articles. In the next section we address the problem of how to come up with a suitable prior for transfer learning across named entity recognition problems.

2 Models considered

2.1 Basic Conditional Random Fields

In this work, we will base our work on Conditional Random Fields (CRF's) (Lafferty et al., 2001), which are now one of the most preferred sequential models for many natural language processing tasks.

The parametric form of the CRF for a sentence of length n is given as follows:

$$p_\Lambda(\mathbf{Y} = \mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(\sum_{i=1}^{n} \sum_{j=1}^{F} f_j(\mathbf{x}, y_i)\lambda_j) \tag{2}$$

where $Z(\mathbf{x})$ is the normalization term. CRF learns a model consisting of a set of weights $\Lambda = \{\lambda_1 ... \lambda_F\}$ over the features so as to maximize the conditional likelihood of the training data, p($Y_{train}|X_{train}$), given the model p_Λ.

2.2 CRF with Gaussian priors

To avoid overfitting the training data, these λ's are often further constrained by the use of a Gaussian prior (Chen and Rosenfeld, 1999) with diagonal covariance, $\mathcal{N}(\mu, \sigma^2)$, which tries to maximize:

$$\underset{\Lambda}{\operatorname{argmax}} \sum_{k=1}^{N} \left(\log\ p_\Lambda(\mathbf{y}_k|\mathbf{x}_k) \right) - \beta \sum_{j}^{F} \frac{(\lambda_j - \mu_j)^2}{2\sigma_j^2}$$

where $\beta > 0$ is a parameter controlling the amount of regularization, and N is the number of sentences in the training set.

2.3 Source trained priors

One recently proposed method (Chelba and Acero, 2004) for transfer learning in Maximum Entropy models [1] involves modifying the μ's of this Gaussian prior. 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 (i.e. setting $\mu = 0$), Λ^{target} is instead regularized towards the previously learned source values Λ^{source} (by setting $\mu = \Lambda^{source}$, while σ^2 remains 1) and thus minimizing $(\Lambda^{target} - \Lambda^{source})^2$.

[1] Maximum Entropy models are special cases of CRFs that use the I.I.D. assumption. The method under discussion can also be extended to CRF directly.

Note that, since this model requires Y_{train}^{target} in order to learn Λ^{target}, it, in effect, requires two distinct labeled training datasets: one on which to *train* the prior, and another on which to learn the model's final weights (which we call *tuning*), using the previously trained prior for regularization. If we are unable to find a match between features in the training and tuning datasets (for instance, if a word appears in the tuning corpus but not the training), we back-off to a standard $\mathcal{N}(0,1)$ prior for that feature.

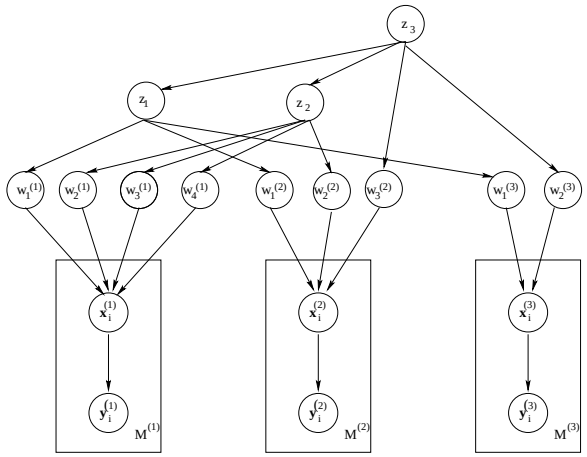

Figure 2: Graphical representation of the hierarchical transfer model.

2.4 New model: Hierarchical prior model

In this section, we will present a new model that learns simultaneously from multiple domains, by taking advantage of our 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. §2.1). 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 1). 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 2. 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:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x},\mathbf{w},\mathbf{z}) &= \left\{ \prod_{d=1}^{D} \prod_{k=1}^{M_d} P(\mathbf{y}_k^{(d)}|\mathbf{x}_k^{(d)}, \Lambda^{(d)}) \right\} \\ &\times \left\{ \prod_{d=1}^{D} \prod_{f=1}^{F_d} \mathcal{N}(\lambda_f^{(d)}|z_{\text{pa}(f^{(d)})}, 1) \right\} \\ &\times \left\{ \prod_{n \in \mathcal{T}_{nonleaf}} \mathcal{N}(z_n|z_{\text{pa}(n)}, 1) \right\} \end{aligned} \tag{3}$$

where the terms in the first line of eq. (3) 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:

$$\begin{aligned} \lambda_f^{(d)} &= \sum_{i=1}^{M_d} \frac{\partial}{\partial \lambda_f^{(d)}} \left(\log P(\mathbf{y}_i^d|\mathbf{x}_i^{(d)}, \Lambda^{(d)}) \right) \\ &+ z_{\text{pa}(f^{(d)})} \\ z_n &= \frac{z_{\text{pa}(n)} + \sum_{i \in \text{ch}(n)} (\lambda|z)_i}{1 + |\text{ch}(n)|} \end{aligned} \tag{4}$$

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.

2.5 An approximate Hierarchical prior model

The Hierarchical prior model is a theoretically well founded model for transfer learning through feature heirarchy. However, our preliminary experiments indicated that its performance on real-life data sets is not as good as expected. Although a more thorough investigation needs to be carried out, our analysis indicates 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 heirarchical prior model and the Chelba model.

As with the Chelba prior method in §2.3, 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 §1.1, 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 1, 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 2.

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 |Leaves($\mathcal{H}^{source}(n)$)| $\leq M$)
  and $n \neq root(\mathcal{H}^{target})$
    $n \leftarrow$ Pa($\mathcal{H}^{target}(n)$)
  Compute μ_f and σ_f using the sample
  $\{\lambda^{source}_i \mid i \in$ 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 2: Algorithm for approximate hierarchical prior: Pa($\mathcal{H}^{source}(n)$) is the parent of node n in feature hierarchy $\mathcal{H}^{source}$; |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 §2.4 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 §1.1.

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 3: 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 Investigation

3.1 Data, domains and tasks

For our experiments, we have chosen five different corpora (summarized in Table 3). 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 (Bunescu et al., 2004), Yapex (Franzén et al., 2002)]; *news articles* [MUC6 (Fisher et al., 1995), MUC7 (Borthwick et al., 1998)]; and *personal e-mails* [CSPACE (Kraut et al., 2004)]. 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 (Cohen, 2004) 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 §1.2 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.

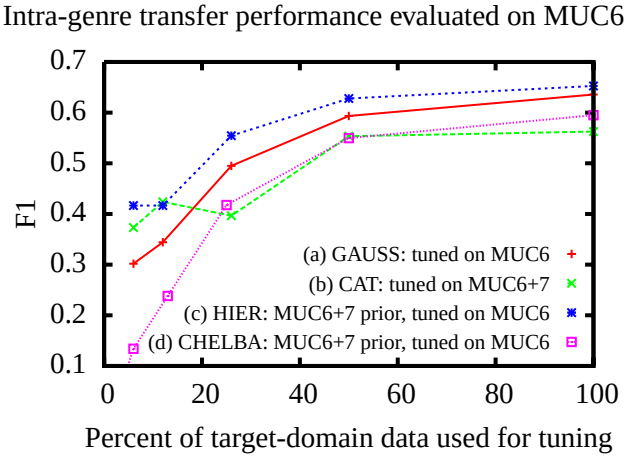

Figure 3: Adding a relevant HIER prior helps compared to the GAUSS baseline (**(c)** $>$ **(a)**), while simply CAT'ing or using CHELBA can hurt (**(d)** $\approx$ **(b)** $<$ **(a)**, except with very little data), and never beats HIER (**(c)** $>$ **(b)** $\approx$ **(d)**).

3.2 Experiments & results

We evaluated the performance of various transfer learning methods on the data and tasks described in §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 $\mathcal{N}(0,1)$ prior
- CHELBA: CRF model tuned on one domain's data, using a prior trained on a different, related domain's data (cf. §2.3)

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

3.2.1 Intra-genre, same-task transfer learning

Figure 3 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

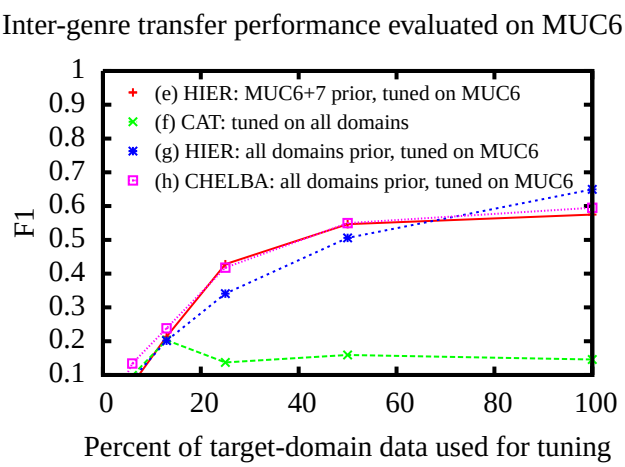

Figure 4: Transfer aware priors CHELBA and HIER effectively filter irrelevant data. Adding more irrelevant data to the priors doesn't hurt (**(e)** $\approx$ **(g)** $\approx$ **(h)**), while simply CAT'ing it, in this case, is disastrous (**(f)** $<<$ **(e)**.

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 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 §3.2.3.

3.2.2 Inter-genre, multi-task transfer learning

In Figure 4 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.

3.2.3 Comparison of HIER prior to baselines

Each scatter plot in Figure 5 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 5 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

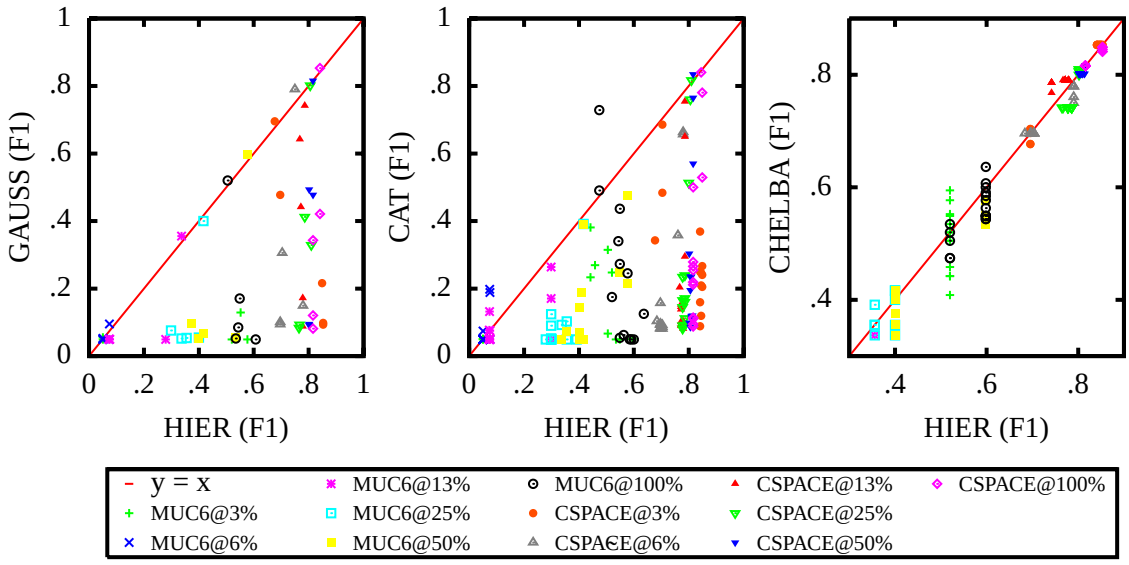

Figure 5: Comparative performance of baseline methods (GAUSS, CAT, CHELBA) vs. HIER prior, as trained on nine prior datasets (both pure and concatenated) of various sample sizes, evaluated on MUC6 and CSPACE datasets. Points below the y = x line indicate HIER outperforming baselines.

as many dependencies between domains as HIER.

4 Conclusions, related & future work

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.

Some of the first formulations of the transfer learning problem were presented over 10 years ago (Thrun, 1996; Baxter, 1997). Other techniques have tried to quantify the generalizability of certain features across domains (Daumé III and Marcu, 2006; Jiang and Zhai, 2006), or tried to exploit the common structure of related problems (Ben-David et al., 2007; Schölkopf et al., 2005). Most of this prior work deals with supervised transfer learning, and thus requires labeled source domain data, though there are examples of unsupervised (Arnold et al., 2007), semi-supervised (Grandvalet and Bengio, 2005; Blitzer et al., 2006), and transductive approaches (Taskar et al., 2003).

Recent work using so-called meta-level priors to transfer information across tasks (Lee et al., 2007), 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 (Daumé III, 2007)). Similarly, work on hierarchical penalization (Szafranski et al., 2007) in two-level trees tries to produce models that rely only on a relatively small number of groups of variable, as structured by the tree, as opposed to transferring knowledge between branches themselves.

Our future work is focused on designing an algorithm to optimally choose a smoothing regime for the learned feature trees so as to better exploit the similarities between domains while neutralizing their differences. Along these lines, we are working on methods to reduce the amount of labeled target domain data needed to tune the prior-based models, looking forward to semi-supervised and unsupervised transfer methods.

References

Rie K. Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. In *JMLR 6*, pages 1817 – 1853.

Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2007. 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*.

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

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *NIPS 20*, Cambridge, MA. MIT Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, Sydney, Australia.

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

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

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In Dekang Lin and Dekai Wu, editors, *EMNLP 2004*, pages 285–292. ACL.

S. Chen and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models.

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

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

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass system as used for MUC-6.

Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidn, and Joakim Cöster. 2002. Protein names and how to find them. In *International Journal of Medical Informatics*.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *CAP*, Nice, France.

Jing Jiang and ChengXiang Zhai. 2006. Exploiting domain structure for named entity recognition. In *Human Language Technology Conference*, pages 74 – 81.

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. 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.

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

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *HLT/EMNLP*.

Rajat Raina, Andrew Y. Ng, and Daphne Koller. 2006. Transfer learning by constructing informative priors. In *ICML 22*.

Bernhard Schölkopf, Florian Steinke, and Volker Blanz. 2005. 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. ACM.

Charles Sutton and Andrew McCallum. 2005. Composition of conditional random fields for transfer learning. In *HLT/EMLNLP*.

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

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

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

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