Semi-Supervised Sequence Classification Using Abstraction Augmented Markov Models*

Cornelia Caragea
Department of Computer Science
Iowa State University
cornelia@iastate.edu

Adrian Silvescu
Department of Computer Science
Iowa State University
silvescu@iastate.edu

Doina Caragea
Computer and Information Sciences
Kansas State University
dcaragea@ksu.edu

Vasant Honavar
Department of Computer Science
Iowa State University
honavar@iastate.edu

ABSTRACT

Supervised methods for learning sequence classifiers rely on the availability of large amounts of labeled data. However, in many applications because of the high cost and effort involved in labeling the data, the amount of labeled data is quite small compared to the amount of unlabeled data. Hence, there is a growing interest in semi-supervised methods that can exploit large amounts of unlabeled data together with small amounts of labeled data. In this paper, we introduce a novel Abstraction Augmented Markov Model (AAMM) based approach to semi-supervised learning. We investigate the effectiveness of AAMMs in exploiting *unlabeled* data. We compare semi-supervised AAMMs with: (i) the Markov models (MMs) (which do not take advantage of unlabeled data); and (ii) an expectation maximization (EM) based approach to semi-supervised training of MMs (that make use of unlabeled data). The results of our experiments on three protein subcellular localization prediction tasks show that semi-supervised AAMMs: (i) can effectively exploit unlabeled data; and (ii) are more accurate than both the MMs and the EM based semi-supervised MMs.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Semi-supervised learning; H.2.8 [**Database Applications**]: Data mining; J.3 [**Life and Medical Sciences**]: [Biology and genetics]

General Terms

Semi-supervised learning; Markov models; Expectation Maximization

1. INTRODUCTION

Many problems in computational biology, e.g., protein function prediction, subcellular localization prediction, can be formulated as sequence classification tasks [2]. Sequence data contain intrinsic dependencies between their constituent elements. Hence, effective sequence models need to incorporate these dependencies. Markov models (MMs) are examples of such sequence models [12]. They capture dependencies between neighboring elements in a sequence and, thus, provide more accurate models of the data. The accuracy of MMs obtained in a supervised setting depends on the availability of large amounts of *labeled* data. Labeling sequences often involves costly and time consuming manual curation.

Recent advances in sequencing technologies have resulted in an exponential increase in the rate at which DNA and protein sequence data are being acquired [1]. Because of the growing gap between the rate of acquisition of sequence data and the rate of manual curation, there is significant interest in semi-supervised algorithms that can exploit large amounts of unlabeled data together with limited amounts of labeled data in training sequence classifiers [32].

Formally, the semi-supervised learning problem can be defined as follows: Given training data $\mathcal{D} = (\mathcal{D}_L, \mathcal{D}_U)$ of labeled and unlabeled examples, where $\mathcal{D}_L = (\mathbf{x}_l, y_l)_{l=1,\cdots,|D_L|}$, $\mathbf{x}_l \in \mathbb{R}^d$, $y_l \in \mathcal{Y}$, and $\mathcal{D}_U = (\mathbf{x}_u)_{u=1,\cdots,|D_U|}$, $\mathbf{x}_u \in \mathbb{R}^d$, $|\mathcal{D}_L| \ll |\mathcal{D}_U|$, respectively; a hypothesis space $\mathcal{H}$; and a performance criterion P, a learning algorithm L outputs a classifier $h \in \mathcal{H}$ that optimizes P. If $|D_L| = 0$, the problem reduces to supervised learning; if $|D_U| = 0$, it reduces to unsupervised learning. The input $\mathbf{x}$ can represent sequences over a finite alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$. During classification, the task of the classifier h is to accurately assign a new example $\mathbf{x}_{test}$ to a class label $y \in \mathcal{Y}$.

We introduce a novel approach to semi-supervised learning of sequence classifiers. Specifically, we use abstraction augmented Markov models (AAMMs), which are variants of Markov models, to incorporate information available in the unlabeled data. AAMMs model the dependency of each element in a sequence on *abstractions* of k preceding elements [7]. The abstractions are organized into an abstraction hierarchy that groups together k-grams that induce similar conditional probabilities of the next letter in the sequence. An AAMM corresponds to a generative model for sequence data expressed in terms of random variables whose values correspond to abstractions over k-grams, in addition to the MMs random variables [7]. AAMMs provide a simple way to incorporate unlabeled data into the model: first, the abstraction hierarchy is constructed using the entire training set including the unlabeled data. Next, the labeled data is

*This research was funded in part by an NSF grant IIS 0711356 to Vasant Honavar and Doina Caragea.

used to estimate the parameters of a set of AAMMs (one for each class) based on the resulting abstraction hierarchy.

Thus, in effect, AAMMs: (i) exploit the relatively large amount of unlabeled data to discover abstractions that transform the sequence data $\mathbf{x}$ and, hence, effectively reduce the number of parameters used to specify the probability $p(\mathbf{x})$; and (ii) use the resulting representation to estimate the posterior probability $p(y|\mathbf{x})$. Hence, AAMMs are likely to yield more robust estimates of $p(y|\mathbf{x})$ than MMs when the amount of labeled data is much smaller compared to the amount of unlabeled data.

To test our hypothesis, we compare AAMMs that use both *labeled* and *unlabeled* data with AAMMs that use only *labeled* data, with the standard MMs, which can not make use of *unlabeled* data, and also with MMs that can incorporate *unlabeled* data through an expectation maximization approach (EM-MM). The results of our experiments show that AAMMs can make effective use of *unlabeled* data and significantly outperform EM-MMs when the amount of *labeled* data are very small, and relatively large amounts of *unlabeled* data are readily available. Here, because of the small amounts of *labeled* data available for estimating parameters, the ability of AAMMs to minimize overfitting (through parameter smoothing) turns out to be especially useful.

The rest of the paper is organized as follows: We discuss related work in Section 2 and provide some background on Markov models for sequence classification in Section 3. We describe our novel AAMM-based approach to semi-supervised learning in Section 4. We present results of experiments comparing AAMMs, MMs and EM-MMs in Section 5. We conclude with a summary and discussion in Section 6.

2. RELATED WORK

A variety of approaches to semi-supervised learning have been studied in the literature (see [9], [32] for reviews). Most of the existing semi-supervised learning algorithms including those based on co-training [5], Expectation Maximization (EM) [25], Transductive Support Vector Machines (TSVM) [19], cluster kernel [30], manifold based approaches [4, 18], essentially involve different means of transferring labels from *labeled* to *unlabeled* samples in the process of learning a classifier that can generalize well on new unseen data.

EM-based methods provide a way to estimate the parameters of a generative model from *incomplete* data [10], i.e., samples that contain missing values for some of the variables. Semi-supervised learning is a special case of such inference where it is the class labels that are missing for a subset of the data [25]. Specifically, the parameters of the model are estimated initially from the labeled fraction of the training data, $\mathcal{D}_L$, and the resulting model is used to predict $p(y|\mathbf{x})$ for each of the unlabeled samples in $\mathcal{D}_U$. The parameters are re-estimated using the entire training data $\mathcal{D}$ and this process is repeated until the estimates converge. Co-training [5] is a variant of this approach where unlabeled data are labeled with two different classifiers trained on different subsets of the features in $\mathbf{x}$.

Several semi-supervised learning algorithms based on discriminative approaches to classification have been investigated. TSVM [19] can be seen as a discriminative counterpart of EM. TSVM starts by training an SVM on the labeled data and uses the trained SVM to label the unlabeled data. The algorithm iteratively attempts to maximize the margin of separation between the sets of samples labeled by the SVM (by considering at each iteration, alternative labels for pairs of originally unlabeled samples that have been assigned different labels by the SVM). A similar outcome can be achieved by adding an additional regularization term for unlabeled data to the objective function optimized by SVM [32]. Similar approaches for exploiting unlabeled data in training discriminative classifiers include [23], [29], [16], [17].

An alternative approach to exploiting unlabeled data relies on the manifold assumption: high-dimensional data lies on a lower dimensional manifold, making it possible to propagate labels from labeled samples to unlabeled samples based on some measure of closeness of the data points on the manifold. The manifold can be approximated by a weighted graph in which the nodes correspond to data samples and the weights on the links between nodes correspond to the pairwise similarity of the corresponding data points [3]. A number of techniques for label propagation have been proposed [4], [18]. Note that graph laplacian based techniques can be interpreted as a more general type of regularization where not only the $L2$ norm of the hypothesis is penalized but also the $L2$ norm of the hypothesis gradient.

Semi-supervised learning methods have been successfully applied in many areas including text classification [25], [5], [19], natural language processing [26], [15], [27], image annotation [6], and more recently bioinformatics and computational biology, [20], [22], [21].

In contrast to the approaches reviewed above, we present a novel *abstraction-based* approach to semi-supervised learning of sequence classifiers.

3. PRELIMINARIES

Before introducing our *abstraction-based* approach to semi-supervised learning, we first provide some background on standard Markov models and their use for sequence classification.

3.1 Markov Models

Markov models (MMs) are probabilistic generative models that assume a mixture model as the underlying model that generated the sequence data. Each mixture component corresponds to a class $c_j \in \mathcal{C} = \{c_1, \cdots, c_{|\mathcal{C}|}\}$. A sequence is generated according to a set of parameters, denoted by θ, that define the model.

Let $\mathbf{x} = x_0 \cdots x_{n-1}$ be a sequence over a finite alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$, and let y denote $\mathbf{x}$'s class (note that if $\mathbf{x}$ was generated by the j^{th} mixture component, then $y = c_j$). Let X_i, for $i = 0, \cdots, n-1$, denote the random variables corresponding to the sequence elements x_i in $\mathbf{x}$. In a k^{th} order MM, the sequence elements satisfy the *Markov property*:

$$X_i \perp\!\!\!\perp \{X_0, \cdots, X_{i-k-1}\} \,|\, \{X_{i-k}, \cdots, X_{i-1}\}.$$

That is, X_i is conditionally independent of $X_0, \cdots, X_{i-k-1}$ given $X_{i-k}, \cdots, X_{i-1}$ for $i = k, \cdots, n-1$. $X_{i-k}, \cdots, X_{i-1}$

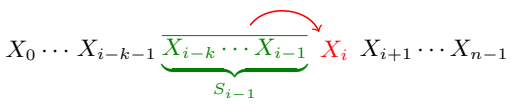

Figure 1: Dependency of X_i on $X_{i-k}, \cdots, X_{i-1}$ in a k^{th} order Markov model.

are called *parents* of X_i. Figure 1 shows the dependency of X_i on $X_{i-k}, \cdots, X_{i-1}$ in a k^{th} order MM. Hence,

$$p(x_i|x_0 \cdots x_{i-1}, c_j; \theta) = p(x_i|x_{i-k} \cdots x_{i-1}, c_j; \theta). \quad (1)$$

The probability of $\mathbf{x}$ given its class c_j, $p(\mathbf{x}|c_j;\theta)$, can be written as follows: $p(\mathbf{x}|c_j;\theta) =$

$$p(c_j|\theta)p(x_0 \cdots x_{k-1}|c_j;\theta) \prod_{i=k}^{n-1} p(x_i|x_{i-k} \cdots x_{i-1}, c_j;\theta). \quad (2)$$

Let S_{i-1} denote the *parents* $X_{i-k} \cdots X_{i-1}$ of X_i. The values of S_{i-1} represent instantiations of $X_{i-k} \cdots X_{i-1}$, which are substrings of length k (i.e., k-grams) over the alphabet $\mathcal{X}$. Let $\mathcal{S}$ denote the set of k-grams over $\mathcal{X}$, s denote a k-gram in $\mathcal{S}$, and σ a symbol in $\mathcal{X}$. The cardinality of $\mathcal{S}$ is $|\mathcal{X}|^k$ and is denoted by N.

The set of parameters θ of an MM is: $\theta = \{\theta_{\sigma|s,c_j} : \sigma \in \mathcal{X}, s \in \mathcal{S}, c_j \in \mathcal{C}; \theta_{s|c_j} : s \in \mathcal{S}, c_j \in \mathcal{C}; \theta_{c_j} : c_j \in \mathcal{C}\}$, where $\theta_{\sigma|s,c_j} = p(\sigma|s,c_j;\theta)$, $\theta_{s|c_j} = p(s|c_j;\theta)$, and $\theta_{c_j} = p(c_j|\theta)$.

3.2 Learning Markov Models

Given a *labeled* training set $\mathcal{D}_L = (\mathbf{x}_l, y_l)_{l=1,\cdots,|\mathcal{D}_L|}$, learning a Markov model reduces to estimating the set of parameters θ from $\mathcal{D}_L$, using the maximum likelihood estimation [8]. The estimate $\hat{\theta}_{\sigma|s,c_j}$ of $\theta_{\sigma|s,c_j}$ is obtained from $\mathcal{D}_L$ as follows:

$$\hat{\theta}_{\sigma|s,c_j} = \frac{1 + \sum_{l=1}^{|\mathcal{D}_L|} \#[s\sigma, \mathbf{x}_l] \cdot p(y_l = c_j|\mathbf{x}_l)}{|\mathcal{X}| + \sum_{\sigma' \in \mathcal{X}} \sum_{l=1}^{|\mathcal{D}_L|} \#[s\sigma', \mathbf{x}_l] \cdot p(y_l = c_j|\mathbf{x}_l)}, \quad (3)$$

where $\#[s\sigma, \mathbf{x}_l]$ is the number of times the symbol σ "follows" the k-gram s in the sequence $\mathbf{x}_l$, and $p(y_l = c_j|\mathbf{x}_l) \in \{0, 1\}$ is obtained based on the sequence label.

The estimate $\hat{\theta}_{s|c_j}$ of $\theta_{s|c_j}$ is obtained from $\mathcal{D}_L$ as follows:

$$\hat{\theta}_{s|c_j} = \frac{1 + \sum_{l=1}^{|\mathcal{D}_L|} \#[s, \mathbf{x}_l] \cdot p(y_l = c_j|\mathbf{x}_l)}{|\mathcal{S}| + \sum_{s' \in \mathcal{S}} \sum_{l=1}^{|\mathcal{D}_L|} \#[s', \mathbf{x}_l] \cdot p(y_l = c_j|\mathbf{x}_l)}, \quad (4)$$

where $\#[s, \mathbf{x}_l]$ is the number of times s occurs in $\mathbf{x}_l$.

The class prior probabilities $\hat{\theta}_{c_j}$ are estimated as follows:

$$\hat{\theta}_{c_j} = \frac{1 + \sum_{l=1}^{r} p(y_l = c_j|\mathbf{x}_l)}{|\mathcal{C}| + |\mathcal{D}_L|}. \quad (5)$$

We used Laplace correction to obtain smoothed estimates.

3.3 Using Markov Models for Classification

Classification of a new sequence $\mathbf{x}$ requires computation of conditional probability $p(y = c_j|\mathbf{x};\hat{\theta})$. Applying Bayes rule:

$$p(y = c_j|\mathbf{x};\hat{\theta}) \propto p(\mathbf{x}|c_j;\hat{\theta})p(c_j|\hat{\theta}). \quad (6)$$

The class with the highest posterior probability, $\arg\max_j p(y = c_j|\mathbf{x};\hat{\theta})$ is assigned to $\mathbf{x}$.

4. SEMI-SUPERVISED AAMM

We first provide the AAMM definitions and then describe how we learn AAMMs in a semi-supervised setting.

4.1 AAMMs

AAMMs effectively reduce the number of parameters of a k^{th} order MM (which is exponential in k) by learning an abstraction hierarchy (AH) over the set of k-grams $\mathcal{S}$.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ over a set of k-grams $\mathcal{S}$ is a rooted tree such that: (1) the root of $\mathcal{T}$ denotes $\mathcal{S}$; (2) the leaves of $\mathcal{T}$ correspond to singleton sets containing individual k-grams in $\mathcal{S}$; (3) the children of each internal node (say* a*) correspond to a partition of the set of k-grams denoted by* a*. Thus,* a *denotes an abstraction or grouping of "similar" k-grams.*

Note that each internal node (or *abstraction* a) contains the subset of k-grams at the leaves of the subtree rooted at a. Figure 2(a) shows an example of an AH $\mathcal{T}$ on a set $\mathcal{S} = \{s_1, \cdots, s_9\}$ of 2-grams over an alphabet of size 3.

Definition 2 (m-Cut) *An m-cut γ_m through an abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ such that for any leaf $s \in \mathcal{S}$, either $s \in \gamma_m$ or s is a descendant of some node in γ_m. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of $\mathcal{S}$.*

Specifically, an m-cut γ_m partitions the set $\mathcal{S}$ of k-grams into m ($m \leq N = |\mathcal{S}|$) non-overlapping subsets $\mathcal{A} = \{a_1 : \mathcal{S}_1, \cdots, a_m : \mathcal{S}_m\}$, where a_i denotes the i-th abstraction and $\mathcal{S}_i$ denotes the subset of k-grams that are grouped together into the i-th abstraction based on some similarity measure. Note that $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_m = \mathcal{S}$ and $\forall 1 \leq i, j \leq m$, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$. In Figure 2(a), the subset of nodes $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

AAMMs extend the graphical structure of MMs by introducing new variables A_i that represent abstractions over the values of S_{i-1}, for $i = k, \cdots, n-1$ (Figure 2(b)). Each A_i takes values in the set of abstractions $\mathcal{A} = \{a_1, \cdots, a_m\}$ corresponding to an m-cut, γ_m. We model the fact that A_i is an abstraction of S_{i-1} by defining $p(A_i = a_i|S_{i-1} = s_{i-1}) = 1$ if $s_{i-1} \in a_i$, and 0 otherwise, where $s_{i-1} \in \mathcal{S}$ and $a_i \in \mathcal{A}$ represent instantiations of variables S_{i-1} and A_i, respectively. Furthermore, in AAMMs, the node X_i directly depends on A_i instead of being directly dependent on S_{i-1}, as in the standard MMs. Hence, the probability of $\mathbf{x}$ given its class, $p(\mathbf{x}|c_j;\theta)$, can be written as follows: $p(\mathbf{x}|c_j;\theta) =$

$$p(c_j|\theta)p(s_{k-1}|c_j;\theta) \prod_{i=k}^{n-1} p(x_i|a_i, c_j;\theta)p(a_i|s_{i-1}). \quad (7)$$

The set of parameters θ of an AAMM is: $\theta = \{\theta_{\sigma|a,c_j} : \sigma \in \mathcal{X}, a \in \mathcal{A}, c_j \in \mathcal{C}; \theta_{s|c_j} : s \in \mathcal{S}, c_j \in \mathcal{C}; \theta_{c_j} : c_j \in \mathcal{C}\}$, where $\theta_{\sigma|a,c_j} = p(\sigma|a,c_j;\theta)$, $\theta_{s|c_j} = p(s|c_j;\theta)$, and $\theta_{c_j} = p(c_j|\theta)$.

4.2 Learning Semi-Supervised AAMMs

In what follows we show how to learn AAMMs from both *labeled* and *unlabeled* data. This involves: learning abstraction hierarchies from both *labeled* and *unlabeled* data; and learning model parameters from *labeled* data using the resulting abstraction hierarchy.

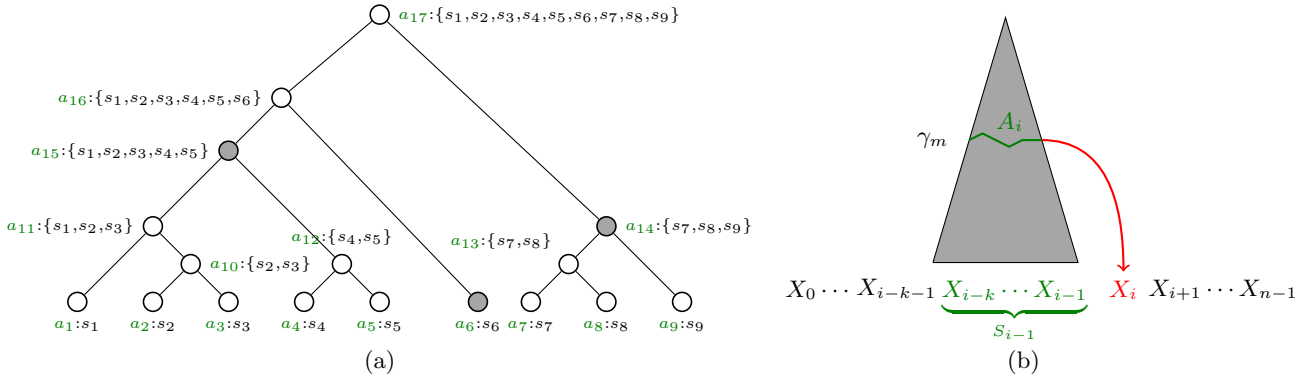

Figure 2: (a) An abstraction hierarchy $\mathcal{T}$ on a set $\mathcal{S} = \{s_1, \cdots, s_9\}$ of 2-grams over an alphabet of size 3. The abstractions a_1 to a_9 correspond to the 2-grams s_1 to s_9, respectively. The subset of nodes $\mathcal{A} = \{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$; (b) Dependency of X_i on A_i, which takes values in a set of abstractions $\mathcal{A}$ corresponding to an m-cut γ_m, in a k^{th} order AAMM.

4.2.1 Learning Abstraction Hierarchies

The algorithm for learning AHs over a set $\mathcal{S}$ of k-grams starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction $a_j \in \mathcal{A}$ corresponds to a k-gram $s_j \in \mathcal{S}$, $j = 1, \cdots, N$. The leaves of the AH $\mathcal{T}$ are initialized with elements of $\mathcal{S}$. The algorithm recursively merges pairs of abstractions that are most "similar" to each other and terminates with an abstraction hierarchy after $N-1$ steps. We store $\mathcal{T}$ in a last-in-first-out (LIFO) stack. For a given choice of the size m of an m-cut through $\mathcal{T}$, the set of abstractions that define an AAMM can be extracted by discarding $m-1$ elements from the top of the stack.

We consider two k-grams to be "similar" if *they occur within similar contexts*. In our case, we define the *context of a k-gram $s \in \mathcal{S}$* to be the conditional probability distribution of the next letter in the sequence given the k-gram, $p(X_i|s)$, independent of the class variable. Hence, this can be estimated from both *labeled* sequences $\mathcal{D}_L$ and *unlabeled* sequences $\mathcal{D}_U$ as follows: $\hat{\theta}_{\sigma|s_t} =$

$$\left[\frac{1 + \sum_{l=1}^{|\mathcal{D}_L|} \#[s\sigma, \mathbf{x}_l] + \sum_{u=1}^{|\mathcal{D}_U|} \#[s\sigma, \mathbf{x}_u]}{|\mathcal{X}| + \sum_{\sigma' \in \mathcal{X}} \left[\sum_{l=1}^{|\mathcal{D}_L|} \#[s\sigma', \mathbf{x}_l] + \sum_{u=1}^{|\mathcal{D}_U|} \#[s\sigma', \mathbf{x}_u]\right]}\right]_{\sigma \in \mathcal{X}} \quad (8)$$

where $\#[s\sigma, \mathbf{x}_l]$ and $\#[s\sigma, \mathbf{x}_u]$ represent the number of times the symbol σ "follows" the k-gram s in the sequence $\mathbf{x}_l$, and $\mathbf{x}_u$, respectively.

Since an abstraction is a set of k-grams, the *context of an abstraction* $a = \{s_1, \cdots, s_{|a|}\}$ is obtained by a weighted aggregation of the contexts of its k-grams. That is,

$$\hat{\theta}_{\sigma|a} = \sum_{t=1}^{|a|} \frac{\#s_t}{\sum_{t=1}^{|a|} \#s_t} \cdot \hat{\theta}_{\sigma|s_t}, \quad (9)$$

where $\#s_t = |\mathcal{X}| + \sum_{l=1}^{|\mathcal{D}_L|} \#[s_t, \mathbf{x}_l] + \sum_{u=1}^{|\mathcal{D}_U|} \#[s_t, \mathbf{x}_u]$.

We identify the most "similar" abstractions as those that have the smallest weighted Jensen-Shannon divergence between their contexts. JS divergence provides a natural way to compute the distance between two probability distributions that represent contexts of two abstractions [24].

4.2.2 Learning AAMM Parameters

Given a *labeled* training set $\mathcal{D}_L = (\mathbf{x}_l, y_l)_{l=1,\cdots,|\mathcal{D}_L|}$, learning an AAMM reduces to estimating the set of parameters θ from $\mathcal{D}_L$, denoted by $\hat{\theta}$. This can be done as follows: use Equation (3) to obtain the estimates $\left[\hat{\theta}_{\sigma|s,c_j}\right]_{\sigma \in \mathcal{X}}$ of $\left[\theta_{\sigma|s,c_j}\right]_{\sigma \in \mathcal{X}}$ for any k-gram $s \in \mathcal{S}$ (note that these estimates correspond to the estimates $\left[\hat{\theta}_{\sigma|a,c_j}\right]_{\sigma \in \mathcal{X}}$ when $a = \{s\}$, i.e., the leaf level in the AH $\mathcal{T}$). The estimates $\left[\hat{\theta}_{\sigma|a,c_j}\right]_{\sigma \in \mathcal{X}}$ of $\left[\theta_{\sigma|a,c_j}\right]_{\sigma \in \mathcal{X}}$, when $a = \{s_1, \cdots, s_{|a|}\}$, are a weighted aggregation of the estimates of a's constituent k-grams, i.e.,

$$\hat{\theta}_{\sigma|a,c_j} = \sum_{t=1}^{|a|} \frac{\#s_t}{\sum_{t=1}^{|a|} \#s_t} \cdot \hat{\theta}_{\sigma|s_t,c_j}, \quad (10)$$

where $\#s_t = |\mathcal{X}| + \sum_{l=1}^{|\mathcal{D}_L|} \#[s_t, \mathbf{x}_l] \cdot p(y_l = c_j|\mathbf{x}_l)]$. Use Equations (4) and (5) to obtain the estimates $\hat{\theta}_{s|c_j}$ of $\theta_{s|c_j}$ and $\hat{\theta}_{c_j}$ of θ_{c_j}, respectively.

4.2.3 Using AAMMs for Classification

Given a new sequence $\mathbf{x} = x_0, \cdots, x_{n-1}$ and an m-cut γ_m through $\mathcal{T}$, $p(\mathbf{x}|c_j; \hat{\theta})$ can be computed as follows: initialize $p(\mathbf{x}|c_j; \hat{\theta})$ by $\hat{\theta}_{x_0 \cdots x_{k-1}|c_j}$; parse the sequence from left to right. For each k-gram $x_{i-k} \cdots x_{i-1}$ find the abstraction $a_w \in \gamma_m$ it belongs to and retrieve the parameters associated with a_w. Successively multiply $\hat{\theta}_{\sigma|a_w,c_j}$ for $i = k, \cdots, n-1$ to obtain $p(\mathbf{x}|c_j; \hat{\theta})$.

As in MMs, apply Bayes rule to obtain $p(y = c_j|\mathbf{x}; \hat{\theta})$ and assign the class with the highest posterior probability to $\mathbf{x}$.

5. EXPERIMENTS AND RESULTS

5.1 Experimental Design

Our experiments are designed to explore the following questions: (i) How effective are AAMMs at exploiting unlabeled data to improve classification accuracy when the amount of labeled data is limited? Specifically, how does the performance of an AAMM trained using both *labeled* and *unlabeled* data compare to that of an AAMM trained using only *labeled*

data when both take advantage of *abstraction*? (ii) How do semi-supervised AAMMs which use both *labeled* and *unlabeled* data compare with MMs trained on only *labeled* data? (iii) How do AAMMs compare with MMs when both use *unlabeled* data? To answer the first question, we compared AAMMs trained using an abstraction hierarchy constructed from both *labeled* and *unlabeled* data with AAMMs trained using an abstraction hierarchy constructed only from *labeled* data. To answer the second and third questions, we compared AAMMs trained using an abstraction hierarchy constructed from both *labeled* and *unlabeled* data with the standard MMs, which can not make use of *unlabeled* data, and with MMs that can incorporate *unlabeled* data through an expectation maximization approach (EM) [10] .

EM applied to MMs (EM-MMs) involves an iterative process of E- and M-steps. Specifically, an initial Markov model is learned only from *labeled* sequences $\mathcal{D}_L$ using Equations (3), (4), and (5) (initialization step). The current model is used to assign *probabilistic* labels to the (originally) *unlabeled* sequences $\mathcal{D}_U$ (i.e., to calculate the probability that each class generated an unlabeled sequence, $p(c_j|\mathbf{x}_u;\hat{\theta})$, $u = 1,\cdots,|\mathcal{D}_U|$) using Equation 6 (E-step). Next, a new model is learned from originally labeled sequences $(\mathbf{x}_l, y_l)_{l=1,\cdots,|\mathcal{D}_L|}$ combined with the newly probabilistically labeled sequences $(\mathbf{x}_u, [p(c_j|\mathbf{x}_u)]_{c_j\in\mathcal{C}})_{u=1,\cdots,|\mathcal{D}_U|}$, which were originally unlabeled using Equations (3), (4), and (5) (M-step). E- and M-steps are repeated until the model does not change from one iteration to another [25].

In the first set of experiments, we fixed the number of *unlabeled* examples and varied the number of *labeled* examples. Specifically, we performed experiments with 1%, 5%, 10%, 15%, 20%, 25%, 35%, and 50% of the training data being used as labeled examples, and 50% being treated as unlabeled examples (by ignoring the class label). To obtain the subsets of labeled examples and that of unlabeled examples, we sampled using a uniform distribution, from the training set. Note that the unlabeled subset of the training data is the same across all the experiments; the labeled subset of the training data is successively augmented to increase the amount of labeled data that is provided to the learner. The class distribution in each subset is the same as that in the entire training set.

In the second set of experiments, we fixed the number of *labeled* examples and varied the number of *unlabeled* examples. Hence, we performed experiments with (i) 1% of training data being treated as labeled, while 1%, 10%, 25%, 50%, 75%, 90%, and 99% being treated as unlabeled; (ii) 10% of training data being treated as labeled, while 1%, 10%, 25%, 50%, 75%, and 90% being treated as unlabeled; (iii) 25% of training data being treated as labeled, while 1%, 10%, 25%, 50%, and 75% being treated as unlabeled. As before, the class distribution in each subset is the same as that in the entire training set.

For all of the experiments, we report the average classification accuracy obtained in a 5-fold cross-validation experiment. We define the relative reduction in classification error between two classifiers to be the difference in error divided by the larger of the two error rates. To test the statistical significance of results, we used the 5-fold cross-validated paired t test for the difference in two classification accuracies [11]. The null hypothesis that the two learning algorithms $\mathcal{M}_1$ and $\mathcal{M}_2$ have the same accuracy on the same test set can be rejected if $|t(\mathcal{M}_1,\mathcal{M}_2)| > t_{4,0.975} = 2.776$ ($p < 0.05$). We abbreviate $|t(\mathcal{M}_1,\mathcal{M}_2)|$ by $|t|$ in what follows.

5.2 Data sets

The problem of predicting subcellular protein localization is important in cell biology, because it can provide valuable information for predicting protein function and protein-protein interactions, among others.

The first data set used in our experiments, PSORTdb v.2.0[1] Gram-negative sequences, introduced in [14], contains experimentally verified localization sites. We refer to this data set as **psortNeg**. We use all proteins that belong to exactly one of the following five classes: *cytoplasm* (278), *cytoplasmic membrane* (309), *periplasm* (276), *outer membrane* (391) and *extracellular* (190). The total number of examples (proteins) in this data set is 1444.

The second and third data sets used in our experiments, **plant** and **non-plant**[2], were first introduced in [13]. The **plant** data set contains 940 examples belonging to one of the following four classes: *chloroplast* (141), *mitochondrial* (368), *secretory pathway/signal peptide* (269) and *other* (consisting of 54 examples with label nuclear and 108 examples with label cytosolic). The **non-plant** data set contains 2738 examples, each in one of the following three classes: *mitochondrial* (361), *secretory pathway/signal peptide* (715) and *other* (consisting of 1214 examples labeled nuclear and 438 examples labeled cytosolic).

5.3 Results

We trained AAMMs, MMs, and EM-MMs using 3-grams extracted from the data[3] (for **psortNeg**, **plant**, and **non-plant** data sets, the number of 3-grams is 7970, 7965, and 7999, respectively). In the case of AAMMs, we trained classifiers for $m = 1500$, where m is the cardinality of the set of abstractions $\mathcal{A}$ used as "features" in the classification model (m is set to 1500 because this partitioning of the set of k-grams produces classifiers that use substantially smaller number of "features" compared to MMs, i.e., ≈ 8000 k-grams, and at the same time, the model compression is not very stringent so as to lose important information in the data through *abstraction*). We denote by AAMM(l+u) an AAMM trained using an AH constructed from both *labeled* and *unlabeled* data, and by AAMM(l) an AAMM trained using an AH constructed only from *labeled* data, when it is necessary to distinguish between AAMMs training procedures. Note that EM-MMs are trained on the same fractions of labeled and unlabeled data as their AAMM(l+u) counterparts, and AAMM(l) and MMs are trained on the same fraction of labeled data as their AAMM(l+u) counterparts.

[1] www.psort.org/dataset/datasetv2.html

[2] www.cbs.dtu.dk/services/TargetP/datasets/datasets.php

[3] The number of all unique k-grams is exponential in k. However, for large values of k, many of the k-grams may not appear in the data (consequently, the counts for such k-grams are zero). Note that the number of unique k-grams is bounded by the cardinality of the *multiset* of k-grams extracted from $|\mathcal{D}|$.

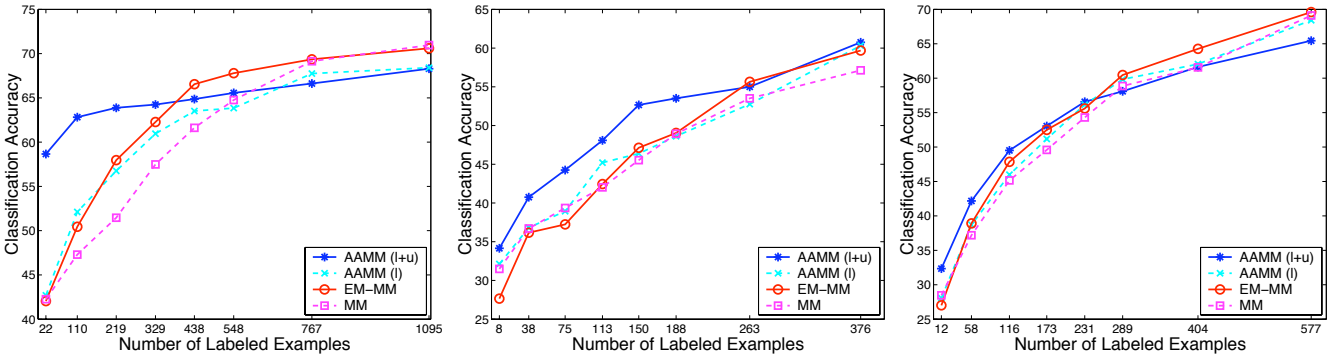

Figure 3: Comparison of AAMMs trained using an abstraction hierarchy learned from both *labeled* and *unlabeled* data, AAMM(l+u), with (i) AAMMs trained using an abstraction hierarchy learned only from *labeled* data, AAMM(l); (ii) Expectation-Maximization with Markov models, EM-MM; and (iii) Markov models, MM, on non-plant (left), plant (center), and psortNeg (right) data sets. x axis indicates the number of labeled examples in each data set corresponding to fractions of 1%, 5%, 10%, 15%, 20%, 25%, 35%, 50% of training data being treated as labeled data. The fraction of unlabeled data in each data set is fixed to 50%.

Figure 3 shows results of the first set of experiments that compare AAMM(l+u) with AAMM(l), MM, and EM-MM on **non-plant**, **plant**, and **psortNeg** data sets. The x axis indicates the number of labeled examples in each data set. The number of unlabeled examples is kept fixed and is equal to the rightmost number of labeled examples on the x axis of each plot.

Comparison of AAMM(l+u) with AAMM(l) trained on a fixed amount of unlabeled data and varying amounts of labeled data. As can be seen from Figure 3, AAMM(l+u) significantly outperforms AAMM(l) on all three data sets when small fractions of labeled data are available. For example, with 110 labeled sequences on **non-plant** (i.e., 5% of labeled data), AAMM(l+u) achieves 63% accuracy while AAMM(l) achieves 52%, which gives 23% reduction in classification error ($|t| = 7.2$). Strikingly, on the same data set, with only 22 labeled sequences (i.e., 1% of labeled data), AAMM(l+u) achieves 59% accuracy as compared to 43% obtained by AAMM(l), which gives 28% reduction in classification error ($|t| = 9.73$). Hence, AAMM(l+u) are able to incorporate information available in the unlabeled data (i.e., joint probability distributions of contiguous amino acids in a sequence) to learn more robust abstraction hierarchies than AAMM(l) when the labeled training set is limited in size (thereby, reducing the risk of *overfitting*).

Furthermore, AAMM(l+u) decreases the need for large numbers of labeled data. Specifically, on **non-plant**, AAMM(l+u) achieves 63% accuracy with 110 labeled examples, which is matched by that of AAMM(l) with 438 labeled examples (≈ 4 times more labeled data). However, when the fraction of labeled data is large, and hence, good estimates of model parameters can be obtained from such data, there is not much need for unlabeled data. For example, AAMM(l+u) becomes similar in performance with AAMM(l) on **non-plant** using 35% and 50% of labeled data (the null hypothesis is not rejected, $|t| = 1.38$ and $|t| = 0.26$, respectively).

As expected, the performance of AAMM(l+u) increases with the increase in the amount of labeled data. For example, on **psortNeg** with 12 labeled sequences (i.e., 1% of labeled data), AAMM(l+u) achieves 32% accuracy while AAMM(l+u) with 289 labeled sequences (i.e., 25% of labeled data) achieves 58% accuracy, which corresponds to 38% reduction in classification error.

Comparison of AAMM(l+u) with MM and EM-MM trained on a fixed amount of unlabeled data and varying amounts of labeled data. Figure 3 also shows the comparison of AAMM(l+u) with MM. AAMM(l+u) is superior in performance to MM, especially when small amounts of labeled data are available. For example, on **plant**, with 75 labeled sequences (i.e., 10% of labeled data), MM achieves 39% accuracy as compared to 44% obtained using AAMM(l+u) ($|t| = 3.07$). On **non-plant**, with 219 labeled sequences (i.e., 10% of labeled data), MM achieves 51% accuracy whereas AAMM(l+u) achieves 64% ($|t| = 14$). AAMM(l+u) not only incorporates information available in the unlabeled data (see previous comparison), but also performs parameter smoothing. Thus, AAMM(l+u) provides more robust estimates of model parameters than MMs, and hence, help reduce *overfitting* when the labeled training set is limited in size.

Both AAMM(l+u) and EM-MM make use of information available in the unlabeled data (i.e., both improve the performance of their counterpart classifiers trained only from labeled data) on all three data sets, although the improvement is not very large on **psortNeg** (Figure 3). However, AAMM(l+u) uses the joint distribution over amino acids (independent of the class variable) to learn a more robust abstraction hierarchy (i.e., a finer partitioning of the set of k-grams), especially when the amount of labeled data is small, so that better estimates of parameters can be obtained. On the other hand, EM-MM uses the joint distribution over amino acids after an initial classifier has made predictions on the unlabeled data. When small amounts of labeled data are available, the predictions made by the initial classifier may not be reliable. As can be seen in Figure 3, AAMM(l+u) significantly outperforms EM-MMs on **non-plant**, **plant**, and **psortNeg** data sets, when the fraction of labeled data is small. For example, with only 22 labeled sequences on

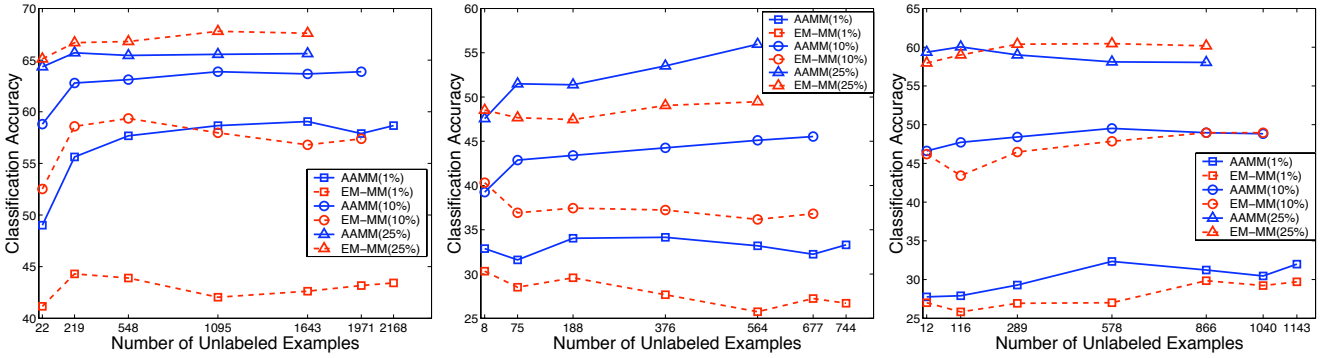

Figure 4: Comparison of AAMMs with EM-MMs for three different fractions of labeled data (i.e., 1%, 10%, and 25%) while varying the amount of unlabeled data on non-plant (left), plant (center), and psortNeg (right) data sets. x axis indicates the number of unlabeled examples in each data set corresponding to fractions of 1%, 10%, 25%, 50%, 75%, 90%, 99% of training data being treated as unlabeled data (by ignoring the class).

non-plant (i.e., 1% of labeled data), AAMM(l+u) achieves 59% accuracy while EM-MM achieves 42%, which gives 29% reduction in classification error ($|t| = 8.83$). Similarly, with only 8 labeled sequences on **plant** (i.e., 1% of labeled data), AAMM(l+u) achieves 34% accuracy as compared to 28% obtained by EM-MM, which gives 8% reduction in classification error ($|t| = 4.66$). As the amount of labeled data increases, EM-MM significantly outperforms AAMM(l+u). For example, with 767 labeled sequences on **non-plant** (i.e., 35% of labeled data), EM-MM achieves 69% accuracy while AAMM(l+u) achieves 67% ($|t| = 4.87$).

We conclude that AAMM(l+u) is able to incorporate information available in the unlabeled data, and hence, produce more robust classifiers than AAMM(l), MM, and EM-MM, when the fraction of labeled data is small. When larger amounts of labeled data become available, EM-MM makes better use of unlabeled data than AAMM(l+u).

Comparison of AAMM with EM-MM trained on a fixed amount of labeled data and varying amounts of unlabeled data. Figure 4 shows results of the second set of experiments that compare AAMMs with EM-MMs on **non-plant**, **plant**, and **psortNeg** data sets, respectively, while varying the amount of unlabeled data for three different fractions of labeled data (i.e., 1%, 10%, and 25% of labeled data) that are kept fixed. The x axis indicates the number of unlabeled examples in each data set.

As can be seen from Figure 4, the improvement in performance of AAMMs over EM-MMs is rather dramatic when the amount of labeled data is quite small. For example, when only 1% of labeled data is used regardless of the amount of unlabeled data, AAMMs consistently significantly outperform EM-MMs on **non-plant** and **plant** data sets (the largest and smallest t values on **non-plant** are 10.96 and 5.66, respectively). However, the difference in performance between AAMMs and EM-MMs diminishes as more and more labeled data become available (and eventually levels off). When the amount of labeled data is increased (e.g., 25% of labeled data), EM-MMs often significantly outperform AAMMs (Figures 4(a) and 4(c)). For example, on **non-plant** with 25% of unlabeled data, EM-MM achieves 68% accuracy, whereas AAMM achieves 66% ($|t| = 7$).

Again as can be seen from Figure 4 , the classification accuracy of AAMMs typically increases with the amount of unlabeled data (when the subset of labeled data is fixed). For example, on **non-plant**, AAMM with 22 labeled sequences (i.e., 1% of labeled data) and 219 unlabeled sequences (i.e., 10% of unlabeled data) achieves an accuracy of 56% as compared to 49% obtained by AAMM with 22 labeled sequences (i.e., 1% of labeled data) and 22 unlabeled sequences (i.e., 1% of unlabeled data), 14% reduction in classification error.

We conclude that AAMMs significantly outperforms EM-MMs when there is less labeled data. Also the more unlabeled data is available, the higher the performance of AAMMs.

6. SUMMARY AND DISCUSSION

We have introduced a novel abstraction-based approach to learning sequence classifiers in a semi-supervised setting. Our approach utilizes abstraction augmented Markov models [7], which extend higher order Markov models by adding new variables corresponding to abstractions of k-grams. The results of our experiments have shown that AAMMs can make effective use of *unlabeled* data. The results also show that AAMMs significantly outperform EM-MMs when the amount of *labeled* data is very small, and relatively large amounts of *unlabeled* data are readily available. Here, because of the small amounts of *labeled* data available, the ability of AAMMs to minimize overfitting (through parameter smoothing) turns out to be especially beneficial.

Consistent with the results reported in [25], we found that EM may decrease rather than increase the accuracy of classifiers if the generative model assumptions are not satisfied (Figure 3 on the **plant** data set). A weighted EM (i.e., weighting *unlabeled* sequences less) helped improved the performance of EM-MMs (data not shown). A similar approach could be considered in AAMMs.

The results presented in this paper demonstrate the effectiveness of an abstraction-based approach to exploiting unlabeled data in a semi-supervised setting. Such an approach can in principle be combined with existing semi-supervised

learning techniques including those that use EM, manifold assumption (propagation of labels from labeled to unlabeled samples based on some similarity measure between samples).

Our implementation of AAMM constructs an abstraction hierarchy over the values of the k predecessors of a sequence element by grouping them together if they induce similar conditional distributions over that element. It would be interesting to explore alternative approaches to building abstraction hierarchies, e.g., probabilistic suffix trees [28].

AAMMs reduce the complexity of the learned model at the risk of some information loss due to *abstraction*. It is of interest to trade off the complexity of the model against its accuracy, e.g., by designing an MDL-based scoring function to guide a top-down search for an optimal cut [31].

7. REFERENCES

[1] W. Ansorge. Next-generation DNA sequencing techniques. *New Biotechnology*, 25(4):195–203, 2009.

[2] P. Baldi and S. Brunak. *Bioinformatics: the Machine Learning Approach*. MIT Press, 2001.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, (7):2399–2434, 2006.

[4] Y. Bengio, O. Delalleau, and N. Le Roux. Label propogation and quadratic criterion. In O. Chapelle, B. Schoelkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 193–217. MIT Press, 2006.

[5] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of COLT' 98*, pages 92–100, New York, NY, USA, 1998. ACM.

[6] G. Camps-valls, S. Member, T. V. B, and D. Zhou. Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45:2044–3054, 2007.

[7] C. Caragea, A. Silvescu, D. Caragea, and V. Honavar. Abstraction Augmented Markov Models. In *NIPS Workshop on "Machine Learning in Comp. Biol."*, '09.

[8] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, 2002.

[9] O. Chapelle, B. Schöelkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[11] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[12] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press., 2004.

[13] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *J. Mol. Biol.*, 300:1005–1016, 2000.

[14] J. L. Gardy and et al. Psort-b: improving protein subcellular localization prediction for gram-negative bacteria. *NAR*, 31(13):3613–17, 2003.

[15] A. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs*, 2006.

[16] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17*, pages 529–236. MIT Press, 2005.

[17] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Neural Information Processing Systems*, volume 12, 1999.

[18] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *ICML '09: Proc. of the 26th Annual ICML*, pages 441–448. ACM, 2009.

[19] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. of the ICML'99*, pages 200–209, 1999.

[20] L. Käll, J. Canterbury, J. Weston, W. Noble, and M. MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4(11):923–925, 2007.

[21] P. Kuksa, P.-H. Huang, and V. Pavlovic. Efficient use of unlabeled data for protein sequence classification: a comparative study. *BMC Bioinformatics*, 10(Suppl 4):S2, 2009.

[22] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *The 21st ICML*, 2004.

[23] N. D. Lawrence and M. I. Jordan. Semi-supervised learning via Gaussian processes. In L. Saul, Y. Weiss, and L. Bottou, editors, *NIPS-17*, 2005.

[24] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Trans. on Inf. Thr.*, 37:145–151, 1991.

[25] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. In *Machine Learning*, pages 103–134, 1999.

[26] Z.-Y. Niu, D.-H. Ji, and C.-L. Tan. Word sense disambiguation using label propagation based semi-supervised learning. In *Proc. of the ACL*, 2005.

[27] Y. Qi, P. Kuksa, R. Collobert, K. Sadamasa, K. Kavukcuoglu, and J. Weston. Semi-supervised sequence labeling with self-learned features. In *Proc. of ICDM*, pages 428–437, Washington, DC, USA, 2009.

[28] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. In *Machine Learning*, pages 117–149, 1996.

[29] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *Advances in Neural Information processing systems 15*, 2002.

[30] J. Weston, C. S. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble. Semi-supervised protein classification using cluster kernels. In *NIPS*, 2004.

[31] J. Zhang, D.-K. Kang, A. Silvescu, and V. Honavar. Learning accurate and concise naive bayes classifiers from attribute value taxonomies and data. *Knowledge and Information Systems*, 9(2):157–179, 2006.

[32] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.