Abstraction-based probabilistic models for sequence classification

by

Cornelia Caragea

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Iowa State University

Ames, Iowa

2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my Ph.D. advisor, Dr. Vasant Honavar, for his valuable advice, guidance and continuous support throughout my Ph.D. studies. I would also like to thank all my committee members, Dr. Vasant Honavar, Dr. David Fernández-Baca, Dr. Drena Dobbs, Dr. Leslie Miller, Dr. Jin Tian, for the knowledge I have acquired in their classes: Machine Learning, Algorithms, Bioinformatics, Databases, and Graphical Models - Bayesian Networks classes, knowledge that helped advance my research.

I would like to thank Doina Caragea and Adrian Silvescu for motivating me to go to graduate school and for fruitful collaborations. I would like to thank my family and my friends for their support.

Also, I would like to thank my colleagues in the AI Group for helpful discussions throughout the years and for valuable interactions and collaborations on several projects.

The research presented in this thesis was supported in part by grants from the National Institutes of Health (GM 066387) and the National Science Foundation (IIS 0711356) to Vasant Honavar.

ABSTRACT

Many applications e.g., biomolecular sequence analysis, image classification, text classification, social network analysis, among others require classification of topologically structured data, i.e. data whose topology reflects intrinsic dependencies between the constituent elements that make up the data. Hence, there is a growing need for automated methods for building predictive models from topologically structured data. Machine learning offers a promising approach to the design of algorithms for training computer programs to *efficiently* and *accurately* classify topologically structured data.

One of the main challenges in learning from topologically structured data has to do with the representation of the data that is presented to a learner. The representation has to be rich enough to capture distinctions that are relevant from the standpoint of learning, but not so rich as to make the task of learning harder due to *overfitting*.

Against this background, we present an abstraction-based probabilistic approach to learning *accurate* and, at the same time, *compact* predictive models from topologically structured data with applications to biological sequence and text data. Specifically, we address the problem of simplifying the data representation used by a learner on sequence classification tasks in two settings: (i) the setting in which sequences are *independent and identically distributed* and (ii) the setting in which sequences are *not independent* (e.g., as in the case of protein sequences that are linked by function). Our abstraction-based approach exploits the complementary strengths of *super-structuring* (constructing complex features by combining existing features) and *abstraction* (grouping similar features to generate more abstract features). Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (hence, super-structuring increases the complexity of the learned models), whereas abstraction helps reduce the number of model parameters by simplifying the data representation.

Furthermore, we introduce and study abstraction augmented Markov models (AAMMs). AAMMs are directed acyclic graphical models that capture dependencies between variables and, at the same time, allow for compact representations of conditional probability tables. More precisely, AAMMs simplify the data representation used by the standard MMs by grouping similar entities to generate more abstract entities that are organized in an abstraction hierarchy.

Experimental results on text document classification tasks and protein subcellular localization prediction tasks demonstrate the feasibility of the proposed approach: the results show that simplifying data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use substantially smaller number of features (by one to three orders of magnitude) than those obtained using super-structuring alone (whose size grows exponentially with the length of direct dependencies). Super-structuring and abstraction-based models are competitive with and, in some cases, outperform, models that use only super-structuring.

Collectively, the techniques developed in this thesis advance the current state of the art in terms of algorithms for building accurate and concise predictive models from topologically structured data.

CHAPTER 1. INTRODUCTION

1.1 Background and Motivation

Recent technological advances have resulted in large amounts of data. Examples of such data include text, images, biological sequences, social networks, etc. One important characteristic that these types of data share is the structured topology that they present, i.e., a topology that reflects intrinsic dependencies between the constituent elements of the data. Any discrete topology can be represented as a graph, where the nodes represent random variables corresponding to the constituent elements, and the edges represent direct dependencies between neighboring elements, i.e., dependencies between adjacent elements in the data. For example, in the case of biological sequence data, the neighborhood of an element (a residue) is given by the elements to its left and to its right in the sequence (called, a “window” around the residue of interest). In the case of image data, the neighborhood is given by the eight adjacent pixels in the image.

While the task of classifying a topologically structured object, for example a text document, can be performed with little or no effort by people, this task remains difficult for computers. Hence, there is a growing need for automated methods for classifying topologically structured data. Machine learning [Bishop, 2006], [Duda *et al.*, 2001], [Mitchell, 1997] currently offers one of the most cost-effective approaches to constructing predictive models in applications where representative training data are available.

One of the main challenges in learning from topologically structured data has to do with the choice of features that are used to describe the data presented to a learner, and the level of detail at which they describe the data. The feature representation can have a major impact on the difficulty of learning, and the accuracy, complexity, and comprehensibility of the learned predictive model [Valiant, 1984]. It has to be rich enough to capture distinctions that are relevant from the standpoint of learning, but not so rich as to make the task of learning harder due to *overfitting*.

This thesis attempts to address this problem. We present an abstraction-based approach to learning classifiers from topologically structured data with applications to biological sequence and text data. Specifically, we address the problem of adapting the data representation used by a learner on **sequence classification tasks**. Our approach exploits the complementary strengths of *super-structuring* (constructing complex features by combining existing features) and *abstraction* (grouping similar features to generate more abstract features). Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (and hence, it increases the complexity of the learned models), whereas *abstraction* helps reduce the number of model parameters by simplifying the data representation.

1.2 Supervised Sequence Classification

We introduce the framework of supervised sequence classification in two settings: the setting in which sequences in a data set are *independent and identically distributed*, and the setting in which sequences in a data set are *highly inter-connected*.

1.2.1 Independent and Identically Distributed Sequence Data

The supervised learning problem [Duda *et al.*, 2001] can be formally defined as follows: Given a training set $\mathcal{D}$ of *i.i.d.* (i.e., *independent and identically distributed*) labeled instances $(\mathbf{x}_i, y_i)_{i=1,\cdots,n}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$ ($\mathcal{Y}$ is the discrete set of all possible class labels), and a hypothesis space $\mathcal{H}$ (i.e., all functions h, $h : \mathbb{R}^d \rightarrow \mathcal{Y}$), the task of a learning algorithm L reduces to searching the hypothesis space $\mathcal{H}$ for a hypothesis (or classifier) h that optimizes a performance criterion P, e.g., maximizes accuracy on the training set $\mathcal{D}$.

Supervised sequence classification is an example of supervised learning problem, where $\mathbf{x}_i$ represents an input sequence over a finite alphabet (or vocabulary) $\mathcal{X}$, $\mathbf{x}_i \in \mathcal{X}^*$. $\mathcal{X}^*$ is the space of all input sequences over $\mathcal{X}$ (e.g., protein sequences) and $\mathcal{Y}$ is the discrete space of all possible outputs for sequences in $\mathcal{X}^*$ (e.g., protein functions). A classifier $h \in \mathcal{H}$ is a function from the input space $\mathcal{X}^*$ of sequences to the output space $\mathcal{Y}$, $h : \mathcal{X}^* \rightarrow \mathcal{Y}$.

Sequence data contain intrinsic dependencies between their constituent elements. For example, in English, sequences of words form sentences under the grammar constraints rather than random sequences of words. In biology, sequences of nucleic or amino acids form DNA or protein sequences rather than random sequences of letters. Consequently, an input sequence $\mathbf{x} = (x_0, \cdots, x_{t-1}) \in \mathcal{X}^*$ has a *chain-like structure* that can be represented by a directed linear-chain graph where the nodes X_i in the graph are random variables corresponding to the sequence elements x_i, $i = 0, \cdots, t-1$, and the edges represent direct dependencies between neighboring elements.

The input sequence $\mathbf{x}$ is refered to as *structured input* or *structured object* [Taskar *et al.*, 2005]. The elements x_i in the input sequence are atomic elements, e.g., nucleic or amino acids, words.

Examples of applications that involve sequence classification include: text classification (assigning a topic to each document in a corpus) [McCallum and Nigam, 1998], [Kang *et al.*, 2005]; protein function prediction (classifying a protein in functional classes) [Andorf *et al.*, 2004], [Borgwardt *et al.*, 2005], [Noble and Ben-Hur, 2007].

The sequence classification problem is strongly related to another problem refered to as the *structured prediction problem* [Taskar, 2004]. In the structured prediction problem, a labeled instance is a pair of input/output sequences $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} = (x_0, x_1, \cdots, x_{t-1})$ and $\mathbf{y} = (y_0, y_1, \cdots, y_{t-1})$, where y_j in the output sequence is the label for x_j in the input sequence, $j = 0, \cdots, t-1$. The labels y_j in the output sequence take values in a discrete set $\mathcal{Y}$. The output sequence $\mathbf{y} \in \mathcal{Y}^*$ is refered to as *structured output* [Taskar *et al.*, 2005].

Examples of applications that involve labeling sequence data include: *part-of-speech tagging* (labeling each word in a sentence with its part-of-speech, e.g., noun, verb) [Lafferty *et al.*, 2001]; *named-entity recognition* (identifying proper names in text) [Sutton and McCallum, 2006]; *RNA-protein interface prediction* (identifying RNA-binding residues in a protein sequence) [Terribilini *et al.*, 2006], [Caragea *et al.*, 2009e]; *protein secondary structure prediction* (classifying each residue in a protein sequence in one of the three classes: helix (H), strand (E) or coil (C)) [Qian and Sejnowski, 1988]; *post-translational modification site prediction* (identifying residues in a protein sequence that undergo different post-translational modifications) [Blom *et al.*, 2004], [Caragea *et al.*, 2007b].

In many sequence classification tasks, the choice of appropriate features used to encode sequences is crucial for the performance of the learning algorithm. Hence, an important challenge in sequence classification is to capture relevant "structures" in the data (i.e., identify "features") that are discriminative with respect to the target variable.

Feature construction methods that build complex features by combining existing features [Honavar and Uhr, 1989a], [Honavar, 1992], [Uhr and Vossler, 1961], [Uhr, 1972], [Srinivasan and King, 1999], [Rendell and Seshu, 2007], [Honavar and Uhr, 1989b], [Pazzani, 1996], [Piramuthu and Sikora, 2008], [Silvescu and Honavar, 2001], [Silvescu *et al.*, 2004], [Pietra *et al.*, 1997], [Jonyer *et al.*, 2004] aim to increase the richness of the representation (see [Liu and Motoda, 1998a] for a survey).

One common approach to feature construction on sequence data is to consider contiguous subsequences of a certain length k (i.e., k-grams) and use them to encode sequences in $\mathcal{D}$ ("bag of k-grams" representation) [Andorf *et al.*, 2004], [Wang and McCallum, 2005]. We refer to this approach of constructing complex features by combining existing features as *super-structuring.* Super-structuring corresponds to considering higher order moments [Casella and Berger, 2002], [Mardia *et al.*, 1979] in the particular case of multivariate statistics.

While *super-structuring* provides a way to improve predictive accuracy, the number of parameters of the learned models increases exponentially with the length k, resulting in complex models that may not *generalize* well, that is, models that may work well on training data, but poorly on test data, due to *overfitting* (e.g., memorize training data and give random outputs on other data).

Abstraction, on the other hand, helps simplify the data representation by grouping "similar" features to generate more abstract features. Thus, abstraction reduces the model input size and

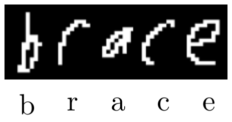


Figure 1.1: An example from handwriting recognition.

can potentially improve the statistical estimates of complex models by reducing the number of parameters that need to be estimated from data.

These two operations, *super-structuring* and *abstraction,* are detailed in what follows.

Super-structuring As already mentioned, sequence data contain intrinsic dependencies between their constituent elements. Given a sequence $\mathbf{x} = (x_0, \cdots, x_{t-1})$ over $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$, the dependencies between neighboring elements can be modeled using *super-structuring* in order to identify "relevant" structures (i.e., complex features) of $\mathbf{x}$ that are discriminative for the task at hand. Super-structuring is the operation of generating all the contiguous (potentially overlapping) sub-sequences of a certain length k from $\mathbf{x}$, $(x_{i-k}, \cdots, x_{i-1})$ for all $i = k, \cdots, t$, $k > 1$, called "super-structures", a.k.a. k-grams (see e.g., [Charniak, 1993] for more details). Super-structuring aims to increase the richness of the representation [Liu and Motoda, 1998a].

To see how super-structuring captures the dependencies between neighboring elements, consider the following example from bioinformatics: in *N-linked glycosylation*, the oligosaccharide chain (a.k.a. glycan) is attached to the amide nitrogen of asparagine (*Asp, N*), which *is part* of the characteristic sequence motifs *N-X-T* (very often), *N-X-S* (often) or *N-X-C* (very rare), where X can be any residue except proline [Gavel and von Heijne, 1990]. Super-structuring identifies these sequence motifs (i.e., super-structures) in order to determine if an *Asp* residue serves as an acceptor site for glycan attachment. Seeing the N site of the super-structure N-$\{P\}$-$[T|S|C]$ in isolation makes it difficult to determine if N is acceptor site or not ($\{P\}$ means any residue except proline, $[T|S|C]$ means either T, S, or C).

Consider now another example from handwriting recognition where the task is to identify handwritten characters (as shown in Figure 1.1 adapted from [Taskar, 2004]). Correctly predicting the fourth letter as the character "c" is very difficult if we see it in isolation from the

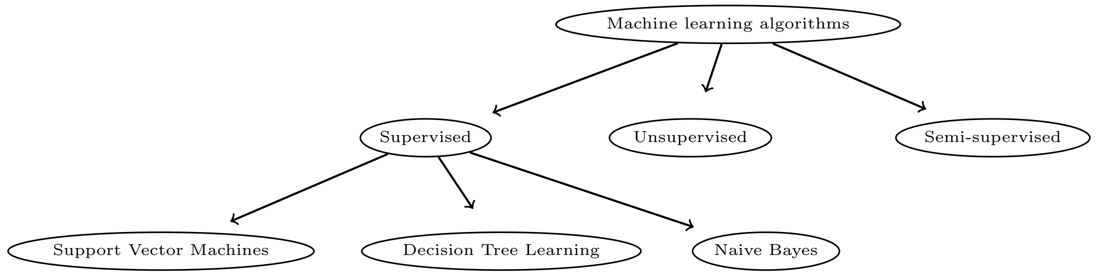


Figure 1.2: A fragment of an *ISA* hierarchy over the terms in a corpus that consists of Machine Learning research articles.

other letters (it appears to be similar to the handwritten letter "r"). However, predicting the character "c" becomes much easier if we condition on the super-structure "bra".

In text classification, if "George Bush" is encountered in a text document, then most likely that the topic of the document is politics. However, seeing "George" and "Bush" in isolation, it would be more complicated to predict the topic of the document.

Hence, super-structuring provides a way to improve predictive accuracy, but the number of parameters of the resulting models increases exponentially with the length k of super-structures.

Abstraction Many real world data, and in particular sequence data, can be organized in abstraction hierarchies that capture the content of the data and their relations. Consider, for example, the bibliographic domain that consists of Machine Learning research articles. The terms (i.e., words) in the corpus can be aranged in an *ISA* hierarchy, where the more general terms are situated at a smaller depth of the hierarchy than the more specific terms. Figure 1.2 shows a fragment of an *ISA* hierarchy over the terms in the corpus: *Machine learning algorithms* term is more general than *Supervised*, *Unsupervised*, and *Semi-supervised* terms; *Support Vector Machines*, *Decision Tree Learning*, and *Naive Bayes* are more specific than *Supervised*.

Abstraction is the operation of grouping "similar", existing entities (e.g., features) to generate more abstract entities. Given a set $\mathcal{S}$ of N features that are used to describe a set of sequences $(\mathbf{x}_i)_{i=1,\cdots,n}$ over $\mathcal{X}$, $\mathbf{x}_i \in \mathcal{X}^*$, abstraction assumes that all N features contain information that is useful for making accurate predictions. Instead of removing redundant/irrelevant features, abstraction partitions the set $\mathcal{S}$ of features into m ($m \leq N$) 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 features 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 $\forall 1 \leq i, j \leq m$, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$.

Our algorithm for constructing m abstractions (i.e., for finding an m-partition of $\mathcal{S}$) is a *greedy* agglomerative procedure. It starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction a_i corresponds to a feature in $\mathcal{S}$. Next, the algorithm recursively groups pairs of abstractions until m abstractions are obtained. More precisely, $N - m$ times the algorithm searches for the most "similar" two abstractions $a_{i_{min}}$ and $a_{j_{min}}$, adds a new abstraction a_u to the set of abstractions $\mathcal{A}$ by taking the union of their features, and removes $a_{i_{min}}$ and $a_{j_{min}}$ from $\mathcal{A}$. Finally, after $N - m$ such steps the algorithm returns m abstractions as the final result.

Note that we can construct the entire abstraction hierarchy over the set $\mathcal{S}$ of features which is returned by our algorithm after $N - 1$ steps and store the resulting abstraction hierarchy $\mathcal{T}$ in a last-in-first-out (LIFO) stack. For a given choice of the size m of an m-partition through $\mathcal{T}$, we can easily extract the set of abstractions $\mathcal{A}$, by discarding $m - 1$ elements from the top of the stack.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ associated with a set of features $\mathcal{S}$ is a rooted tree such that: (1) The root of $\mathcal{T}$ corresponds to the abstraction that consists of all features in $\mathcal{S}$; (2) The tree $\mathcal{T}$ has exactly N leaves corresponding to the N features in $\mathcal{S}$; (3) The internal nodes of $\mathcal{T}$ correspond to abstractions over features (i.e., subsets of "similar" features); (4) The edges of $\mathcal{T}$ correspond to partial order relationships $\prec$ (e.g., ISA relationships) between their corresponding nodes.*

Figure 2.1 shows an example of an abstraction hierarchy $\mathcal{T}$ produced by our algorithm 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 (or level of abstraction) through the abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ satisfying the following properties: (1) For any leaf s_i,*

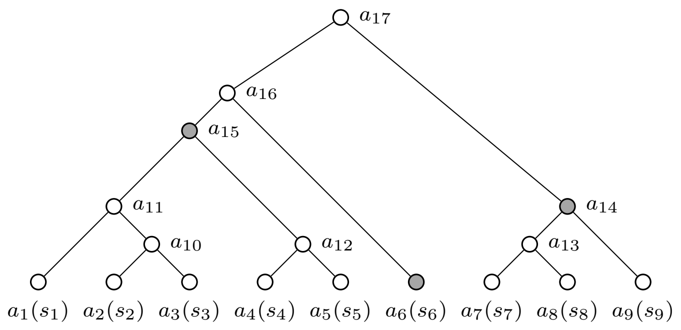

a_{17}
a_{16}
a_{15}
a_{11}
a_{10}
a_{12}
a_{14}
a_{13}
$a_1(s_1)$ $a_2(s_2)$ $a_3(s_3)$ $a_4(s_4)$ $a_5(s_5)$ $a_6(s_6)$ $a_7(s_7)$ $a_8(s_8)$ $a_9(s_9)$

Figure 1.3: 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 $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

either $s_i \in \gamma_m$ or s_i is a descendant of a node $a_j \in \gamma_m$; and (2) for any two nodes $a_j, a_l \in \gamma_m$, a_j is neither a descendant nor an ancestor of a_l. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of the set $\mathcal{S}$ of features.

In Figure 2.1, the subset of nodes $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

Our criterion for establishing similarity between features and, thus, deriving useful abstractions is based on the following observation: *similar features occur within similar contexts.* In our case, we define the *context of a feature* as the conditional probability of a target (i.e., "relevance) variable given the feature, where the target variable can be either the class variable or the element following the feature in the sequence. Since an abstraction is a set of features, the *context of an abstraction* is obtained by aggregating the contexts of its elements. We identify the most "similar" abstractions as those that have the smallest 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.

Abstraction aims to control the level of detail in the data. By simplifying the data representation, abstraction helps maintain comprehensible, and at the same time, accurate models (i.e., can potentially provide better statistical estimates of model parameters).

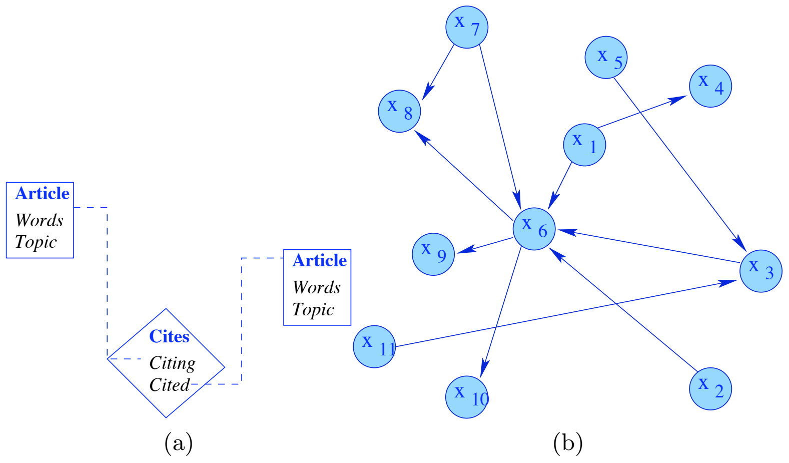


Figure 1.4: (a) A simple schema corresponding to a bibliographic domain (the figure is adapted from Getoor et al., 2002). (b) A sample instantiation graph corresponding to the schema in (a). The nodes x_i represent `Article` instances, i.e., $x_i \in$ `Article`, while the edges (x_i, x_j) represent `Cites` instances, i.e., $(x_i, x_j) \in$ `Cites`.

1.2.2 Relational Sequence Data

In a more general setting, sequence data can be highly inter-connected. For example, research articles *cite* related research articles; proteins *interact* with other proteins, DNAs or RNAs in order to properly perform their functions. We refer to such data as *relational* or *network* data.

A *schema* $\mathcal{S}$ of a relational data source describes a set of *concepts* $\mathcal{X} = \{X_1, \cdots, X_t\}$, and the *relations* between them, $\mathcal{R}(X_i, X_j)$. An instance of a concept X_i is a structured object, e.g. a string, a document, or an image. These instances can be described by *features* such as a set of attributes, a histogram of words, or features from spatially contiguous elements of an image. An attribute A of a concept X_i, denoted by $X_i.A$ takes values in a set $\mathcal{V}(X_i.A)$. A relation $R(X_i, X_j)$ corresponds to a set defined as $x_i.R = \{x_j \in X_j$ s.t. x_j is related to x_i through $R\}$, where x_i is an instance of the concept X_i. A tuple (x_i, x_j) is an instance of the relation R. A *data set* $\mathcal{D}$ that specifies a set of instances and the relations between them is an *instantiation* $\mathcal{I}(\mathcal{S})$ of a schema $\mathcal{S}$. It can be represented as an *instantiation graph* in which nodes denote instances and edges denote relations between instances [Getoor *et al.*, 2002].

Figure 1.4a shows a simple schema of a bibliographic domain which consists of the `Article` concept and the `Cites` relation. Note that this schema consists of only one concept and the relation is to itself. Figure 4.1b shows a sample instantiation graph corresponding to the relational schema in Figure 1.4a. The nodes x_i in the graph represent research articles (i.e., `Article` instances, $x_i \in$ `Article`) and the edges (x_i, x_j) represent the "citation" relation between articles (i.e., $(x_i, x_j) \in$ `Cites`). Note that each instance can have a variable number of related instances and therefore, a variable number of *features* [Neville *et al.*, 2003].

In the relational learning framework [Getoor *et al.*, 2007], it is common to encode the current object (i.e., sequences in our case) using relevant features of the related objects in the network in addition to its own features [Lu and Getoor, 2003]. Transformations of the original problem into one that can be handled by a traditional machine learning method involve operations, e.g., *aggregation*, to handle *multisets* of feature values. However, such transformations result in loss of information that can not be recovered [Neville *et al.*, 2003], [Perlich and Provost, 2003].

1.3 Semi-supervised Sequence Classification

Semi-supervised learning lies at the intersection of unsupervised and supervised learning [Chapelle *et al.*, 2006].

Unsupervised learning problem The unsupervised learning problem [Duda *et al.*, 2001] can be formally defined as follows: Given a dataset $\mathcal{D}$ of unlabeled instances $(\mathbf{x}_i)_{i=1,\cdots,m}$, $\mathbf{x}_i \in \mathbb{R}^d$, and a similarity measure (a distance function) defined on pairs of instances, the learning algorithm L partitions the data into dissimilar clusters of similar instances. The dissimilarity between different clusters and the similarity within each cluster are measured by the predefined distance function. A new instance $\mathbf{x}_{test}$ is assigned to one of the closest clusters.

Hierarchical clustering is an example of unsupervised learning problem, in which clusters contain subclusters that can further contain other subsubclusters, being most often represented as a tree-like structure, e.g. kingdom biological hierarchy. Hierarchical clustering attempts to uncover the hidden structure that exists in the unlabeled data.

Semi-supervised learning problem Unlike supervised and unsupervised learning problems, semi-supervised learning problem [Duda *et al.*, 2001] uses both labeled and unlabeled instances to construct a classifier h from the input space to the output space. As mentioned before, experimental determination of labeled instances is an expensive and laborious process, while the unlabeled data are readily available and often provide valuable information. Thus, given a relatively small amount of labeled data in many domains, and a vast amount of unlabeled data, the semi-supervised learning has recently become an attractive approach to various classification tasks. Formally, the semi-supervised learning problem can be defined as follows: Given a dataset $\mathcal{D} = (\mathcal{D}_l, \mathcal{D}_u)$ of labeled and unlabeled instances, $\mathcal{D}_l = (\mathbf{x}_i, y_i)_{i=1,\cdots,n}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$, and $\mathcal{D}_u = (\mathbf{x}_i)_{i=n+1,\cdots,n+m}$, 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 $m = 0$, the problem reduces to the supervised learning; if $n = 0$, it reduces to the unsupervised learning. The input $\mathbf{x}_i$ can represent sequences over the finite alphabet (or vocabulary) $\mathcal{X}$, $\mathbf{x}_i \in \mathcal{X}^*$.

Incorporating evolutionary information through the means of *Position Specific Scoring Matrices (PSSMs)* when building a predictive model represents an example of semi-supervised learning problem. The *PSSMs* are obtained by aligning a target labeled sequence with unlabeled sequences from large databases.

We have studied the use of abstraction hierarchies for learning semi-supervised sequence classifiers from scarcely labeled data.

Our approach to learning abstraction hierarchies over the values of the *parents* of each node in a sequence is based on a hierarchical agglomerative clustering algorithm. The values of the *parents* of a node are grouped together based on the similarity between the conditional distributions of the node given its parents, independent of the class variable. This approach is especially appealing as it allows to incorporate information available in the unlabelled sequence data in order to improve the performance of classifiers trained in a semi-supervised setting.

It is worth noting the difference between semi-supervised and transductive learning.

Transductive learning Transductive learning is a special case of semi-supervised learning. Given a set of labeled instances $\mathcal{D}_l = (\mathbf{x}_i, y_i)_{i=1,\cdots,n}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$, and a set of unlabeled instances $\mathcal{D}_u = (\mathbf{x}_i)_{i=n+1,\cdots,n+m}$, a learning algorithm L assigns a label to the unlabeled instances (rather than learning a function for classifying unseen test examples). Thus, transductive learning is considered to be simpler than semi-supervised learning.

Hence, our abstraction-based approach can be applied not only in the special case of transductive learning, but also in the more general setting of semi-supervised learning.

1.4 Related Approaches

1.4.1 Generating compact data representations

Feature selection Feature selection is a traditional approach to reducing a feature set $\mathcal{S}$ of size $N$to a desired number of features m [Liu and Motoda, 1998a], [Guyon and Elisseeff, 2003]. Feature selection selects a subset $\mathcal{F}$ of features from the entire set $\mathcal{S}$, $\mathcal{F} \subseteq \mathcal{S}$, $|\mathcal{F}| = m$ and $m \leq N$, based on some chosen criterion. The features are ranked according to a scoring function $Score$ and the top m best ranked features are selected. Mutual information [Cover and Thomas, 1991] between the probability of the class variable and the probability of a feature which measures how dependent the two variables are can be used as the scoring function $Score$.

Mutual information between random variable X and Y is given by:

$$I(X,Y) = KL(p(X,Y)||p(X)p(Y)) = \sum_{x \in X, y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Agglomerative Information Bottleneck Agglomerative Information Bottleneck (AIB) [Slonim and Tishby, 1999] is a data organization method where the goal is to find a compression of a variable X while preserving as much information as possible about a target (or relevance) variable Y. This has a variational formulation which yields exact solutions. In AIB, data is organized in an AH based on the similarity of the class conditional distributions. Specifically, AIB has been applied to cluster words in a given text document data set based on the similarity of the conditional distributions of the document topics.

The algorithm starts with one word per cluster and recursively merges pairs of sets of words, in a greedy fashion, to generate an AH. A merge that minimizes the loss in the mutual information between the intermediate clustering and the class variable is performed at each step of the algorithm. The AIB method captures the information that one variable (e.g., a word) contains about another variable (e.g., the topic of a document).

Sequential Information Bottleneck Sequential Information Bottleneck (sIB) [Slonim *et al.*, 2002] is a clustering algorithm that provides improvement over AIB both in the quality of the clusters found and in time and space complexity. However, one limitation of sIB is that the tree structure returned by AIB is lost. The number of clusters K is given as input to the sIB algorithm, and, in general, the value of K is not known in advance. The sIB is similar to the K-means algorithm [Mitchell, 1997]. The main difference between them is that in sIB the updates are performed sequentially, while in K-means, the updates are performed in parallel.

The sIB algorithm starts with a random partition of a set of objects X into K clusters, $\{c_1, \cdots, c_K\}$. At each step, an object x in X is drawn from its current cluster and assigned to a newly constructed cluster, c_{K+1}. The cluster c_{K+1} is merged with $c_j \in \{c_1, \cdots, c_K\}$ such that the cost of merging c_{K+1} and c_j is minimized. The process is repeated until there are no more updates in the data.

Dimensionality reduction Dimensionality reduction is the problem of representing high dimensional data in a lower dimensional space. It is widely used both in unsupervised learning for visualization, compression, etc., and in supervised learning to reduce the input size presented to a learner. This problem of "embedding" high dimensional objects into a lower dimensional space has received much attention in the literature [Globerson *et al.*, 2007], [Globerson and Tishby, 2003], [Tenenbaum *et al.*, 2000], [Roweis and Saul, 2000], [Hinton and Roweis, 2002].

Spectral clustering methods fall within the category of graph partitioning algorithms that partition the data into disjoint clusters by exploiting the eigenstructure of a similarity matrix. In general, to find an optimal graph partitioning is NP complete. Shi and Malik [Shi and Malik, 2000] proposed an approximate spectral clustering algorithm that optimizes the normalized cut (NCut) objective function. It is a divisive, hierarchical clustering algorithm that recursively bi-partitions the graph until some criterion is reached, producing a tree structure.

In this context, it is also worth noting that kernel methods (e.g., kernel PCA) project data into higher (possibly infinite) dimensional feature spaces and then project it back to the original space. Kernel PCA [Mika *et al.*, 1999] is a generalization of linear principal component analysis (PCA).

1.4.2 Sequence Models

Markov Models In fixed-order Markov models (MMs) [Durbin *et al.*, 2004], [Duda *et al.*, 2001], the elements of a sequence satisfy the *Markov property*: Each element in the sequence directly depends on a fixed number of previous elements, called *parents*, and is independent of the rest of the elements in the sequence.

Let $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1})$ be an input sequence over an alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$. An MM represents the joint probability distribution of $\mathbf{x}$ under the Markov assumption. The graphical representation of an MM is a directed linear-chain graph where the nodes X_i represent random variables corresponding to the sequence elements x_i, $i = 0, \cdots, n-1$, and the edges represent direct dependencies between neighboring elements. For a k^{th} order MM, the conditional independencies are:

$$X_i \perp\!\!\!\perp \{X_0, \cdots, X_{i-k-1}\} \,|\, \{X_{i-k}, \cdots, X_{i-1}\} \text{ for } i = k, \cdots, n-1$$

That is, X_i is conditionally independent of $X_0, \cdots, X_{i-k-1}$ given $X_{i-k}, \cdots, X_{i-1}$ for any $i = k, \cdots, n-1$. $X_{i-k}, \cdots, X_{i-1}$ are called the *parents* of X_i. The joint probability distribution $p(\mathbf{X})$, where $\mathbf{X}$ denotes the set of nodes, can be factorized as follows:

$$p(\mathbf{X}) = p(X_0, \cdots, X_{k-1}) \prod_{i=k}^{n-1} p(X_i | X_{i-k}, \cdots, X_{i-1})$$

The directed graph for a first-order Markov model on the nodes of the input sequence $\mathbf{x}$, $\{X_0, X_1, \cdots, X_{n-1}\}$, is shown in Figure 1.5a.

Interpolated Markov Models Interpolated MMs (IMMs) [Jelinek and Mercer, 1980] combine several fixed-order MMs that capture important sequence patterns that would otherwise be ignored by a single fixed-order MM. IMMs are potentially more powerful than MMs and can produce more accurate results [Salzberg *et al.*, 1999].

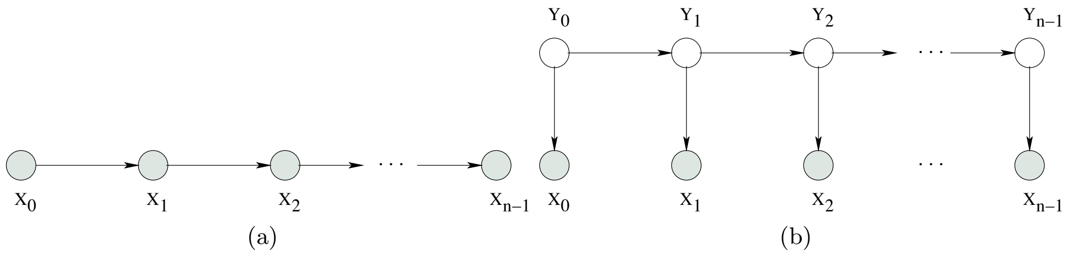


Figure 1.5: (a) First-Order Markov Model; (b) First-Order Hidden Markov Model

Hidden Markov Models Hidden Markov Models (HMMs) [Duda *et al.*, 2001], [Durbin *et al.*, 2004], [Rabiner, 1989] are a class of well-studied generative graphical models that are used for modeling sequence data. HMMs augment the graphical structure of MMs with unobservable (or hidden) variables and, thus are more expressive than MMs. The hidden variables control the dependency of the current element in an observable sequence on the previous elements.

More specifically, an HMM consists of a set of states $\mathcal{Y}$, a set of transition probabilities between states, and an alphabet $\mathcal{X}$ of emission symbols. An HMM generates a sequence $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1})$ by transitioning between states and emitting a symbol in each state. In an HMM, $\mathbf{x}$ is the observable sequence, while the sequence of states $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1})$ that have produced $\mathbf{x}$ is hidden.

The directed graph for a first-order HMM on the pair of sequences $(\mathbf{x}, \mathbf{y})$ is shown in Figure 1.5b. Its joint probability distribution $p(\mathbf{X}, \mathbf{Y})$, can be factorized as follows:

$$p(\mathbf{X}, \mathbf{Y}) = p(Y_0) \prod_{i=0}^{n-2} p(Y_{i+1}|Y_i) \prod_{i=0}^{n-1} p(X_i|Y_i)$$

Learning HMMs reduces to estimating the model parameters, i.e., transition probabilities, $p(Y_{i+1}|Y_i)$, and emission probabilities $p(X_i|Y_i)$, from the training data. This involves an EM-like algorithm, called the *Forward-Backward* algorithm.

1.5 Contributions

In this thesis, we address the problem of learning abstraction-based models in order to *accurately* classify biological sequence and text data. Abstraction helps reduce the model input size and, at the same time, can potentially improve the statistical estimates of complex models by reducing the number of parameters that need to be estimated from data (hence reducing the risk of *overfitting*). However, one limitation of our abstraction-based approach is that, while it provides simpler models, the simplicity is achieved by losing some information through abstraction. The challenge is to trade off the complexity of the model against its predictive accuracy.

In what follows, we briefly discuss the context of our work and point out the contributions of this thesis. We start with the setting in which the sequences in a data set are *independent and identically distributed.*

1.5.1 Independent and identically distributed sequence data

Silvescu [Silvescu, 2008] has introduced an approach to exploiting the complementary strengths of super-structuring and abstraction or super-structuring and feature selection to adapt the data representation used by a learner on sequence data. We refer to the approach of combining super-structuring and abstraction by SS-A. In SS-A, Naïve Bayes classifiers have been trained to predict protein subcellular localization. The constructed abstraction hierarchies group “similar” k-grams based on the class conditional distributions that the k-grams induce. This work is similar in spirit to the agglomerative information bottleneck (AIB) method of Slonim and Tishby [Slonim and Tishby, 1999]. However, AIB requires an interative process to estimate parameters for the bottleneck variables, whereas SS-A requires only a simple weighted aggregation of the existing parameters. Although AIB has been applied to cluster words in a given text data set, it can also be used to cluster k-grams.

- **We have extended SS-A to text classification (Chapter 2).** The results of our experiments show that:
  - The performance of classifiers trained using super-structuring representation was worse than that of classifiers trained using a “bag of words” feature representation which is consistent with the work of Dumais et al. [Dumais *et al.*, 1998].

- We compared Naïve Bayes trained using an abstraction-based representation with Naïve Bayes trained using a "bag of words" representation, and with Naïve Bayes trained using feature selection-based representation. Consistent with the results obtained using SS-A, the abstraction-based representations result in better performing models than those based on feature selection, and in similar performing models compared to the "bag of words" representations.
- Although feature selection and abstraction are seemingly antagonistic, we found that better value can be derived by using them sequentially whereas feature selection is used in order to remove irrelevant features followed by abstraction that can further reduce feature redundancy.

• **We have explored Support Vector Machine (SVM) classifiers in the SS-A framework on protein subcellular localization prediction tasks (Chapter 2).** Specifically, we have compared SVMs using the combination of super-structuring and abstraction as the feature representation with their counterparts trained using a unigram representation of sequences, super-structuring representation, and super-structuring followed by feature selection representation. The results show that:

  - The combination of super-structuring and abstraction results in better performing models than that of super-structuring and feature selection.
  - Unlike Naïve Bayes classifiers, SVMs trained on super-structuring followed by abstraction representation substantially outperform those trained on super-structuring representation. Hence, abstraction can be seen as a *regularizer* for SVM classifiers which are sensitive to *overfitting*.

Markov models are effective sequence models [Durbin *et al.*, 2004] that capture dependencies between neighboring elements in a sequence and thus provide more accurate models of the data. While dependencies between neighboring elements provide a way to improve the predictive accuracy, the number of model parameters increases exponentially with the range of direct dependencies, thereby increasing the risk of *overfitting* when the data set is limited in size.

Hence, learning Markov models accurately can be difficult in practice (e.g., for 2^{nd} and 3^{rd}order Markov models, there are $20^{2+1} = 8,000$ and $20^{3+1} = 160,000$ parameters, respectively, that need to be estimated from data).

- **We have introduced and studied Abstraction Augmented Markov Models (AAMMs) (Chapter 3)**, which are *coherent* directed acyclic graphical models that simplify the data representation used by standard Markov models (the directed graphs for a 2^{nd} order MM and for a 2^{nd} order AAMM are shown in Figures 1.6a and 1.6b):
  - **We have shown how to represent and learn AAMMs**. Specifically, the graphical structure of AAMMs extends that of the standard MMs by constructing new variables that represent abstractions over the values of the *parents* of each node (i.e., k-grams). Learning AAMMs from sequence data involves two steps: learning abstraction hierarchies and learning model parameters. We have learned AAMMs on protein subcellular localization prediction tasks.
  - The quality of our classifiers that use as input *abstract features* highly depends on the quality of the abstraction hierarchies, just as the quality of a classical classifier depends on the quality of the data (i.e., noisy data may result in very poor classifiers). Hence, **we have designed a clustering algorithm that produces suitable abstraction hierarchies for AAMMs**, and hence, well performing AAMMs. Our clustering algorithm is an extension of the Agglomerative Information Bottleneck (AIB). The primary difference between our AAMM clustering algorithm and AIB is in the similarity used to cluster k-grams together, i.e., in AAMM, the k-grams are clustered based on the similarity between the conditional distributions of the next letter in the sequence rather than based on the class conditional distributions induced by the k-grams as in AIB. More precisely, the AIB method captures the information that one variable (e.g., a word) contains about another variable (e.g., the topic of a document), while our method captures the information that a k-gram contains about the next element in a sequence, independent of the class.

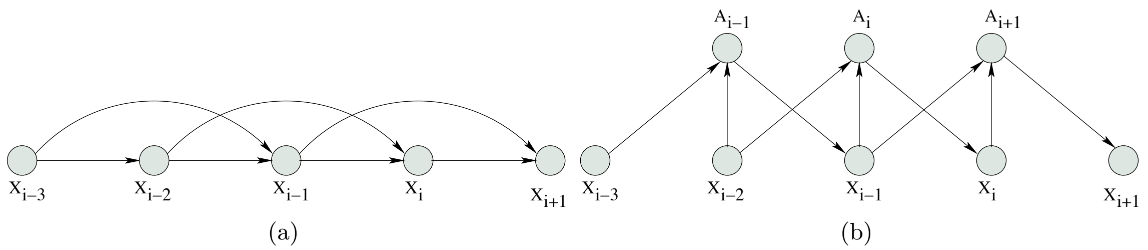


Figure 1.6: (a) 2^{nd} Order Markov Model; (b) 2^{nd} Order Abstraction Augmented Markov Model

  - With the explosion in the amount of sequence data in recent years, sequence annotations do not keep pace with the available sequence data. **We have shown how to produce more accurate models compared to the standard MMs in a semi-supervised setting** (i..e., in a setting where there are relatively small amounts of labeled data, but rather plentiful unlabeled data that would be very expensive to be labeled). Our idea was to incorporate information available in the unlabeled data into the constructed abstraction hierarchy, information that is ignored by MMs.
  - An AAMM can be simulated by an appropriately constructed Hidden Markov Model (HMM) [Durbin *et al.*, 2004] where the number of hidden states is equal to the number of abstractions (i.e., clusters) in the AAMM. However, as a state corresponds to a cluster (i.e., a set of k-grams) over the observable variables, the state is not really "hidden". Although AAMMs are less expressive models than HMMs, **we have shown that they are easier to learn models compared to HMMs**.

- The results of our experiments show that:
  - Abstraction-based Markov models substantially outperform the standard Markov models in both supervised and semi-supervised settings.
  - The AAMM clustering algorithm helps produce more suited abstraction hierarchies than agglomerative information bottleneck for abstraction-based Markov models.

1.5.2 Relational sequence data

Next, we discuss the context of our work and point out the contributions of this thesis in the setting where the data are highly inter-connected (i.e., relational or network data).

Lu and Getoor [Lu and Getoor, 2003], Neville et al. [Neville *et al.*, 2003] and Taskar et al. [Taskar *et al.*, 2002] (among others) have learned classifiers from relational data. Zhang et al. [Zhang *et al.*, 2006] (among others) have proposed an approach to learning classifiers from partially specified data over nominal attributes. McCallum et al. [McCallum *et al.*, 1998] (among others) have used shrinkage in a hierarchy of classes to improve classification.

- **We have described an Expectation-Maximization algorithm for learning link-based Naïve Bayes classifiers from relational data that are organized in abstraction hierarchies associated with both the input (i.e., sequence data) and the output (i.e., class variable) (Chapter 4).**
  - We have evaluated our classifiers on bibliographic data that is a special case of relational data, which consists of `Article` concept and `Cites` relation.
- **We have provided a solution to the problem of dealing with partially specified data (Chapter 4)** that inevitably results from choosing a level of abstraction (when a multinomial model is assumed as the underlying generative model for text data). Our solution is based on a statistical approach, called *shrinkage*.
- Results of experiments show that:
  - more abstract levels can yield better performing classifiers
  - incorporating partially specified data using shrinkage yields better estimates of model parameters

Kargupta and Chan [Kargupta and Chan, 2000] and Caragea et al. [Caragea *et al.*, 2005] (among others) have learned classifiers from semantically heterogeneous distributed data that are annotated with relevant meta data.

- **We have described a strategy for learning link-based Naïve Bayes classifiers from a collection of semantically heterogeneous, distributed, relational data sources that are annotated with relevant meta data (Chapter 4):**
  - Our strategy exploits mappings between a user ontology and data source ontologies to gather the necessary statistics needed for learning classifiers.
  - The classifiers are learned without centralized access to data and are *provably exact* with respect to their centralized counterparts, i.e. the hypothesis produced in the distributed setting is identical to that produced in the centralized setting when there is no overlap between the distributed data sources (the data sources are integrated according to a user view, via mappings).

1.6 Thesis Outline

In what follows, we summarize the rest of the chapters in the thesis. Each chapter is self-contained and corresponds to a paper. Some of the work presented in this thesis has been already published in conference proceedings, is under review, or is in preparation (see below).

Chapter 2 presents our approach to adapting the data representation used by a learner on sequence classification tasks. The approach that exploits the complementary strengths of *super-structuring* (constructing complex features by combining existing features) and *feature abstraction* (grouping of similar features to generate more abstract features), yields smaller and, at the same time, accurate models. Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (and hence, it increases the complexity of the learned models) whereas feature abstraction helps reduce the number of model parameters by simplifying the data representation.

Chapter 3 presents abstraction augmented Markov models (AAMMs), which are directed acyclic graphical models that simplify the data representation used by the standard Markov models (MMs). AAMMs group similar entities to generate more abstract entities that are organized in an abstraction hierarchy. *Abstraction* reduces the Markov model size and improves the statistical estimates of complex models by reducing the number of parameters that need to be estimated from data.

Chapter 4 addresses the problem of learning classifiers from structured relational data that are annotated with relevant meta data. Specifically, in Chapter 4, we show how to learn classifiers at different levels of abstraction in a relational setting, where the structured relational data are organized in an abstraction hierarchy that describes the semantics of the content of the data. We show how to cope with some of the challenges presented by *partial specification* in the case of structured data, that unavoidably results from choosing a particular level of abstraction. Our solution to partial specification is based on a statistical method, called *shrinkage*.

Furthermore, in this chapter, we address the problem of learning predictive models from multiple large, distributed, autonomous, and hence almost invariably semantically disparate, relational data sources from a user's point of view. We show under fairly general assumptions, how to exploit data sources annotated with relevant meta data in building predictive models from a collection of distributed relational data sources, without the need for a centralized data warehouse, while offering strong guarantees of *exactness* of the learned classifiers relative to their centralized relational learning counterparts.

Chapter 5 concludes the thesis, and provides a summary of contributions and directions for further research.

Published work

- **Chapter 2** on *combining super-structuring and abstraction on sequence classification* has been published in **ICDM 2009** conference proceedings with experiments on protein subcellular localization prediction using Naïve Bayes and SVM classifiers [Silvescu *et al.*, 2009]. A journal version of the work in this chapter is under preparation and will contain additional experiments on data set size variation, experiments using SVM on text classification, statistical significance tests, comparisons of abstraction-based methods with more sophisticated feature selection methods, etc.

- **Chapter 3** on *abstraction augmented Markov models* is currently under review at **SDM 2010** (joint work with Adrian Silvescu, Doina Caragea and Vasant Honavar) [Caragea *et al.*, 2009c]. However, preliminary results obtained using AAMMs on protein subcellular localization prediction has been accepted for presentation by *Machine Learning for Computational Biology* **NIPS Workshop 2009** [Caragea *et al.*, 2009d]. A journal version of the work in this chapter is also under preparation and will contain experiments using SVM on biological sequence classification tasks, comparisons of AAMMs with HMMs and IMMs, etc.

- **Chapter 4** on *learning link-based classifiers from ontology-extended textual data* has been published in **ICTAI 2009** conference proceedings with experiments on text classification [Caragea *et al.*, 2009a]. An extension of this problem to the distributed setting, i.e., *learning link-based Naïve Bayes classifiers from ontology-extended distributed data* has been published in **ODBASE 2009** conference proceedings with experiments on text classification [Caragea *et al.*, 2009b].

Other published work (not included in the thesis)

- **Using Global Sequence Similarity to Enhance Biological Sequence Labeling**, *In: BIBM 2008 (joint work with Jivko Sinapov, Drena Dobbs, and Vasant Honavar)* [Caragea *et al.*, 2008]. Identifying functionally important sites from biological sequences, formulated as a biological sequence labeling problem, has broad applications ranging from rational drug design to the analysis of metabolic and signal transduction networks. In this paper, we present an approach to biological sequence labeling that takes into account the global similarity between sequences. Our approach combines unsupervised and supervised learning techniques. Given a set of sequences and a similarity measure defined on pairs of sequences, we learn a mixture of experts model by using spectral clustering to learn the hierarchical structure of the model and by using bayesian approaches to combine the predictions of the experts. We evaluate our approach on two important biological sequence labeling problems: RNA-protein and DNA-protein interface prediction problems. Experimental results show that global sequence similarity can be exploited to improve performance of classifiers trained to label biological sequence data.

- **Assessing the Performance of Macromolecular Sequence Classifiers**, *In: BIBE 2007 (joint work with Jivko Sinapov, Drena Dobbs, and Vasant Honavar)* [Caragea *et al.*, 2007a]. Machine learning approaches offer some of the most cost-effective approaches to building predictive models (e.g., classifiers) in a broad range of applications in computational biology. Comparing the effectiveness of different algorithms requires reliable procedures for accurately assessing the performance (e.g., accuracy, sensitivity, and specificity) of the resulting predictive classifiers. The difficulty of this task is compounded by the use of different data selection and evaluation procedures and in some cases, even different definitions for the same performance measures. We explore the problem of assessing the performance of predictive classifiers trained on macromolecular sequence data, with an emphasis on cross-validation and data selection methods. Specifically, we compare sequence-based and window-based cross-validation procedures on three sequence-based prediction tasks: identification of glycosylation sites, RNA-Protein interface residues, and Protein-Protein interface residues from amino acid sequence. Our experiments with two representative classifiers (Naive Bayes and Support Vector Machine) show that sequence-based and windows-based cross-validation procedures and data selection methods can yield different estimates of commonly used performance measures such as accuracy, Matthews correlation coefficient and area under the Receiver Operating Characteristic curve. We argue that the performance estimates obtained using sequence-based cross-validation provide more realistic estimates of performance than those obtained using window-based cross-validation.

- **Learning from Large Autonomous Data Sources using Sufficient Statistics**, *In: WI 2008 (joint work with Neeraj Koul, Vikas Bahirwani, Doina Caragea, and Vasant Honavar)* [Koul *et al.*, 2008]. In this paper we introduce a framework to learn from autonomous data sources using sufficient statistics to focus on challenges such as massive data size as well as lack of access to underlying data due to constraints such as privacy or the cost of moving the massive data set to a centralized location. We address the important steps in machine learning such as data ltering and dealing with missing values in this setting where access to the underlying data is not possible. We introduce the concept of query complexity as the number and the number of queries required to get the sufficient statistics and analyze various algorithms in terms of their query complexities. Finally we present a system where the autonomous data source is assumed to be an RDBMS and the sufficient statistic queries can be see as SQL count queries.

- **Striking Similarities in Diverse Telomerase Proteins Revealed by Combining Structure Prediction and Machine Learning Approaches**, *In: PSB 2008 (joint work with Jae-Hyung Lee, Michael Hamilton, Colin Gleeson, Peter Zaback, Jef Sander, Xue Li, Feihong Wu, Michael Terribilini, Vasant Honavar, and Drena Dobbs)* [Lee *et al.*, 2008]. Telomerase is a ribonucleoprotein enzyme that adds telomeric DNA repeat sequences to the ends of linear chromosomes. The enzyme plays pivotal roles in cellular senescence and aging, and because it provides a telomere maintenance mechanism for $\approx 90\%$ of human cancers, it is a promising target for cancer therapy. Despite its importance, a high-resolution structure of the telomerase enzyme has been elusive, although a crystal structure of an N-terminal domain (TEN) of the telomerase reverse transcriptase subunit (TERT) from *Tetrahymena* has been reported. In this study, we used a comparative strategy, in which sequence-based machine learning approaches were integrated with computational structural modeling, to explore the potential conservation of structural and functional features of TERT in phylogenetically diverse species. We generated structural models of the N-terminal domains from human and yeast TERT using a combination of threading and homology modeling with the *Tetrahymena* TEN structure as a template. Comparative analysis of predicted and experimentally verified DNA and RNA binding residues, in the context of these structures, revealed significant similarities in nucleic acid binding surfaces of *Tetrahymena* and human TEN domains. In addition, the combined evidence from machine learning and structural modeling identified several specific amino acids that are likely to play a role in binding DNA or RNA, but for which no experimental evidence is currently available.

- **Machine Learning in Computational Biology**, *In: Encyclopedia of Database Systems (with Vasant Honavar)* [Caragea and Honavar, 2009]. Advances in high throughput sequencing and omics technologies and the resulting exponential growth in the amount of macromolecular sequence, structure, gene expression measurements, have unleashed a transformation of biology from a data-poor science into an increasingly data-rich science. Despite these advances, biology today, much like physics was before Newton and Leibnitz, has remained a largely descriptive science. Machine learning currently o?ers some of the most cost-e?ective tools for building predictive models from biological data, e.g., for annotating new genomic sequences, for predicting macromolecular function, for identifying functionally important sites in proteins, for identifying genetic markers of diseases, and for discovering the networks of genetic interactions that orchestrate important biological processes. Advances in machine learning e.g., improved methods for learning from highly unbalanced datasets, for learning complex structures of class labels (e.g., labels linked by directed acyclic graphs as opposed to one of several mutually exclusive labels) from richly structured data such as macromolecular sequences, 3-dimensional molecular structures, and reliable methods for assessing the performance of the resulting models, are critical to the transformation of biology from a descriptive science into a predictive science.

- **Mixture of experts models to exploit global sequence similarity on biomolecular sequence labeling**, *In: BMC Bioinformatics (joint work with Jivko Sinapov, Drena Dobbs, and Vasant Honavar)* [Caragea *et al.*, 2009e]. This article has been invited for publication in BMC Bioinformatics as an extension of the BIBM 2008 conference paper.

- **Struct-NB: Predicting Protein-RNA Binding Sites Using Structural Features**, *In: International Journal of Data Mining and Bioinformatics (joint work with Fadi Towfic, David Gemperline, Drena Dobbs, and Vasant Honavar)* [Towfic *et al.*, 2009]. We explore whether protein-RNA interfaces differ from non-interfaces in terms of their structural features and whether structural features vary according to the type of the bound RNA (e.g., mRNA, siRNA, etc.), using a non-redundant dataset of 147 protein chains extracted from protein-RNA complexes in the Protein Data Bank. Furthermore, we use machine learning algorithms for training classifiers to predict protein-RNA interfaces using information derived from the sequence and structural features. We develop the *Struct-NB* classifier that takes into account structural information. We compare the performance of *Naïve Bayes* and *Gaussian Naïve Bayes* with that of *Struct-NB* classifiers on the 147 protein-RNA dataset using sequence and structural features respectively as input to the classifiers. The results of our experiments show that *Struct-NB* outperforms *Naïve Bayes* and *Gaussian Naïve Bayes* on the problem of predicting the protein-RNA binding interfaces in a protein sequence in terms of a range of standard measures for comparing the performance of classifiers.

- **Glycosylation site prediction using ensembles of Support Vector Machine classifiers**, *In: BMC Bioinformatics (joint work with Jivko Sinapov, Adrian Silvescu, Drena Dobbs, and Vasant Honavar)* [Caragea *et al.*, 2007b]. Glycosylation is one of the most complex post-translational modifications (PTMs) of proteins in eukaryotic cells. Glycosylation plays an important role in biological processes ranging from protein folding and subcellular localization, to ligand recognition and cell-cell interactions. Experimental identification of glycosylation sites is expensive and laborious. Hence, there is significant interest in the development of computational methods for reliable prediction of glycosylation sites from amino acid sequences. We explore machine learning methods for training classifiers to predict the amino acid residues that are likely to be glycosylated using information derived from the target amino acid residue and its sequence neighbors. We compare the performance of Support Vector Machine classifiers and ensembles of Support Vector Machine classifiers trained on a dataset of experimentally determined N-linked, O-linked, and C-linked glycosylation sites extracted from O-GlycBase version 6.00, a database of 242 proteins from several different species. Experimental results show that the ensembles of Support Vector Machine classifiers outperform single Support Vector Machine classifiers on the problem of predicting glycosylation sites in terms of a range of standard measures for comparing the performance of classifiers. The resulting methods have been implemented in *EnsembleGly*, a web server for glycosylation site prediction.

CHAPTER 2. COMBINING SUPER-STRUCTURING AND ABSTRACTION ON SEQUENCE CLASSIFICATION

We present an approach to adapting the data representation used by a learner on sequence classification tasks. Our approach that exploits the complementary strengths of super-structuring (constructing complex features by combining existing features) and feature abstraction (grouping of similar features to generate more abstract features), yields smaller and, at the same time, accurate models. Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (and hence, it increases the complexity of the learned models) whereas feature abstraction helps reduce the number of model parameters by simplifying the data representation. The results of our experiments on several data sets drawn from macromolecular sequence and text classification applications show that adapting data representation by combining super-structuring and abstraction, makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those that are obtained using super-structuring alone, without sacrificing predictive accuracy. Our experiments also show that simplifying data representation using abstraction yields better performing models than those obtained using feature selection.

2.1 Introduction

Sequence classification arises in many real-world problems. For example, in computational biology, predicting protein function or subcellular localization can be formulated as sequence classification tasks, where the amino acid sequence of the protein is used to classify the protein in functional and localization classes [Baldi and Brunak, 2001]. In text classification, predicting the topic of a document based on its sequence of words can also be seen as a sequence classification task.

Representational commitments, i.e., the choice of features or attributes that are used to describe the sequence data presented to a learner, and the level of detail at which they describe the data, can have a major impact on the difficulty of learning, and the accuracy, complexity, and comprehensibility of the learned predictive model [Valiant, 1984]. The representation has to be rich enough to capture distinctions that are relevant from the standpoint of learning, but not so rich as to make the task of learning harder due to overfitting.

Sequence data contain intrinsic dependencies between their constituent elements. Given a sequence $\mathbf{x} = (x_0, \cdots, x_{t-1})$ over a finite alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$, the dependencies between neighboring elements can be modeled using *super-structuring*. Super-structuring involves generating all the contiguous (potentially overlapping) sub-sequences of a certain length k, $(x_{i-k}, \cdots, x_{i-1})$, $i = k, \cdots, t$, $k > 1$, called super-structures, a.k.a. k-grams (see e.g., [Charniak, 1993] for more details). Super-structuring aims to increase the richness of the representation [Liu and Motoda, 1998a]. While super-structuring provides a way to improve predictive accuracy, the number of parameters of the resulting models increases exponentially with the length k of super-structures.

Traditional feature selection methods [Liu and Motoda, 1998a], [Guyon and Elisseeff, 2003] select a subset of the available features based on some chosen criteria. On a sequence classification task, selecting a subset of super-structures by feature selection can substantially reduce the number of model parameters. However, this approach of combining super-structuring and feature selection may not capture important sequence patterns (*motifs*, in the case of biological sequences) that are removed during the selection process.

Feature abstraction methods [Zhang *et al.*, 2006], [Kang *et al.*, 2004], on the other hand, group "similar", existing features to generate more abstract features. These methods aim to control the level of detail in the data representation. We present an approach to adapting the data representation used by a learner that exploits the complementary strengths of super-structuring and feature abstraction. Specifically, we propose an algorithm to construct new features in a sequence classification scenario by combining *super-structuring* and *abstraction*. Super-structuring improves classification performance at the expense of increasing the model size, and abstraction reduces the model size and improves the statistical estimates of complex models (especially when data is sparse) by reducing the number of parameters to be estimated from data.

We evaluate our approach on two protein subcellular localization data sets and two text classification 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. Also, with the exponential increase in the amount of online text data, problems such as classifying web pages and filtering e-mail have become very important.

The results of our experiments show that adapting the data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those that are obtained using super-structuring independently without sacrificing predictive accuracy. The results also show that, for the same number of features, models obtained using abstraction substantially outperform those obtained using feature selection.

The rest of the paper is organized as follows. We begin by describing our approach of combining super-structuring and abstraction in Section 2. Next we discuss the alternative approach of combining super-structuring and feature selection in Section 3. In Section 4, we present the experimental design and provide results on four sequence classification tasks. We conclude in Section 5.

2.2 Combining Super-structuring and Abstraction

Let $\mathcal{D} = (\mathbf{x}_l, y_l)_{l=1,\cdots,N}$ be a data set of sequences $\mathbf{x}_l$ over a finite alphabet $\mathcal{X}$ along with their associated class labels y_l from a finite set $\mathcal{Y}$. Our approach to feature construction that combines super-structuring and abstraction is illustrated in Algorithm 3. The *input* of the algorithm is: the data set $\mathcal{D}$; the length k of the super-structures; and a number m of features

Algorithm 1 Feature Construction

```
Input: $\mathcal{D} = (\mathbf{x}_l, y_l)_{l=1,\cdots,N}$, $\mathbf{x}_l \in \mathcal{X}^*$, $y_l \in \mathcal{Y}$; $k$ = length of super-structures; $m$ = number of features to construct
Output: A transformed data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$
$\mathcal{S} \leftarrow$ Generate-Super-structures($\mathcal{D}$)
$\mathcal{D}_{\mathcal{S}} \leftarrow$ Encode($\mathcal{D}, \mathcal{S}$)
$\mathcal{A} \leftarrow$ Construct-Abstractions($\mathcal{D}_{\mathcal{S}}, m$)
$\mathcal{D}_{\mathcal{S}+\mathcal{A}} \leftarrow$ Encode($\mathcal{D}_{\mathcal{S}}, \mathcal{A}$)
```

to construct. The *output* of the algorithm is a transformed data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$ of instances encoded with the newly constructed features.

The algorithm starts by generating a set $\mathcal{S} = \{s_1, \cdots, s_n\}$ of (complex) unique features (i.e., super-structures or k-grams) from the sequences in $\mathcal{D}$. Specifically, the k-grams are substrings of length k over $\mathcal{X}$ that are generated by sliding a window of length k over sequences in $\mathcal{D}$. The cardinality of $\mathcal{S}$ is $|\mathcal{S}| = n \leq |\mathcal{X}|^k$ (note that if a k-gram does not appear in $\mathcal{D}$, it is not considered as a feature[1]). In the next step of the algorithm, the original data set $\mathcal{D}$ is transformed into a data set $\mathcal{D}_{\mathcal{S}}$ as follows: each instance $(\mathbf{x}_l, y_l)$ in $\mathcal{D}$ is encoded by $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ where $[s_i : \#s_i^l]$, $i = 1, \cdots, n$ represents frequency counts for the k-gram s_i in $\mathbf{x}_l$. This representation is also known as "bag of features (k-grams)" [McCallum and Nigam, 1998].

With the transformed data set $\mathcal{D}_{\mathcal{S}}$ where each instance consists of a bag of k-grams over the set $\mathcal{S}$ and its associated class label, the algorithm proceeds to reduce the feature set $\mathcal{S}$ to the desired number of features m. The reduction is accomplished by constructing new features or *abstractions* over the k-grams inside the procedure `Construct-Abstractions`$(\mathcal{D}_{\mathcal{S}}, m)$. Specifically, the set of k-grams is partitioned into m non-overlapping sets $\mathcal{A} = \{a_1 : \mathcal{S}_1, \cdots, a_m : \mathcal{S}_m\}$ where a_i denotes the i-th abstraction and $\mathcal{S}_i$ denotes the subset of k-grams that are grouped together into this i-th abstraction. Thus, an abstraction is identified with the collection of k-grams/features that are grouped together. Note that $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_m = \mathcal{S}$ and $\forall 1 \leq i, j \leq$

[1]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.

Algorithm 2 `Construct-Abstractions`

```
**Input:** $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$, $m$ = number of abstractions to construct
**Output:** A set of abstractions $\mathcal{A}$, such that $|\mathcal{A}| = m$
Initialize $\mathcal{A} = \{a_1 :\{s_1\}, \cdots, a_n :\{s_n\}\}$
**for** $u = n + 1$ **to** $2n - m$ **do**
  $(i_{min}, j_{min}) = \arg\min_{i,j} dist(a_i, a_j)$
  $a_u = a_{i_{min}} \cup a_{j_{min}}$
  $\mathcal{A} = \mathcal{A} \backslash \{a_{i_{min}}, a_{j_{min}}\} \cup \{a_u\}$
**end for**
```

m, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$.

Finally, the algorithm transforms the data set $\mathcal{D}_{\mathcal{S}}$ into another data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$ where each instance is now represented by m features (aside from the class label) as follows: given the set of m abstractions $\mathcal{A} = \{a_1 : \mathcal{S}_1, \cdots, a_m : \mathcal{S}_m\}$, where $\mathcal{S}_i \subseteq \mathcal{S}, \forall 1 \leq i \leq m$, an instance $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ from $\mathcal{D}_{\mathcal{S}}$ is transformed into an instance $([a_1 : \#a_1^l], \cdots, [a_m : \#a_m^l]\}, y_l)$ where

$$\#a_i^l = \sum_{s_q \in \mathcal{S}_i} \#s_q^l, \ \forall 1 \leq i \leq m$$

The advantage of this approach is that the resulting representation of sequences can be used with any learning algorithm. In this study we used Naive Bayes Multinomial [Mitchell, 1997] and Support Vector Machines [Burges, 1998] with linear kernel. In the next subsection we present the `Construct-Abstractions`($\mathcal{D}_{\mathcal{S}}$,$m$) procedure of Algorithm 3.

2.2.1 Constructing Abstractions

The procedure for constructing abstractions is illustrated in Algorithm 2. The *input* of the algorithm is a data set where each instance consists of a bag of k-grams and a class label, $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$, and a number $m \leq n$ that represents the reduced number of abstractions that is desired. The *output* is a set of abstractions $\mathcal{A}$ over the k-grams such that $|\mathcal{A}| = m$.

The algorithm, that is a *greedy* agglomerative procedure, starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction a_i corresponds to a k-gram s_i in $\mathcal{S}$. Next, the algorithm recursively groups pairs of abstractions until m abstractions are obtained. More precisely, $n - m$ times the algorithm searches for the most "similar" two abstractions $a_{i_{min}}$ and $a_{j_{min}}$, adds a new abstraction a_u to the set of abstractions $\mathcal{A}$ by taking the union of their k-grams, and removes $a_{i_{min}}$ and $a_{j_{min}}$ from $\mathcal{A}$. Finally, after $n - m$ such steps the algorithm returns m abstractions as the final result.

Note that we can construct the entire abstraction hierarchy over the set $\mathcal{S}$ of k-grams which is returned by Algorithm 2 after $n - 1$ steps.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ associated with a set of k-grams $\mathcal{S}$ is a rooted tree such that: (1) The root of $\mathcal{T}$ corresponds to the abstraction that consists of all k-grams in $\mathcal{S}$; (2) The tree $\mathcal{T}$ has exactly n leaves corresponding to the n k-grams in $\mathcal{S}$; (3) The internal nodes of $\mathcal{T}$ correspond to abstractions over k-grams (i.e., subsets of "similar" k-grams); (4) The edges of $\mathcal{T}$ correspond to partial order relationships $\prec$ (e.g., ISA relationships) between their corresponding nodes.*

Figure 2.1 shows an example of an abstraction hierarchy $\mathcal{T}$ 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 the abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ satisfying the following properties: (1) For any leaf s_i, either $s_i \in \gamma_m$ or s_i is a descendant of a node $a_j \in \gamma_m$; and (2) for any two nodes $a_j, a_l \in \gamma_m$, a_j is neither a descendant nor an ancestor of a_l. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of the set $\mathcal{S}$ of k-grams.*

In Figure 2.1, the subset of nodes $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

We can store the resulting abstraction hierarchy $\mathcal{T}$ in a last-in-first-out (LIFO) stack. For a given choice of the size m of an m-cut through $\mathcal{T}$, we can easily extract the set of abstractions $\mathcal{A}$, by discarding $m - 1$ elements from the top of the stack.

In order to complete the description of Algorithm 2, we need to define the similarity between two abstractions. Our general criterion for establishing similarity between items and, thus, deriving useful abstractions is based on the following observation: *similar items occur within similar contexts.* In our case, we define the context of a k-gram as probability of the class

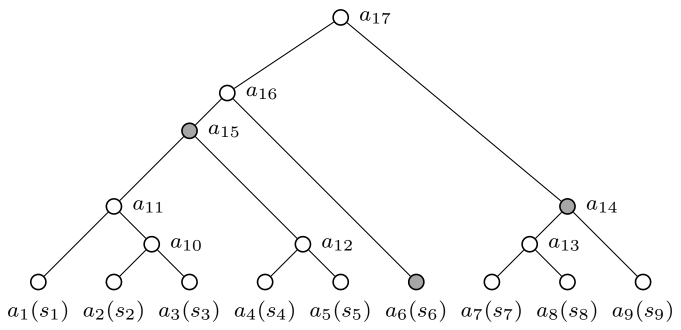

a_{17}
a_{16}
a_{15}
a_{11}
a_{14}
a_{10}
a_{12}
a_{13}
$a_1(s_1)$ $a_2(s_2)$ $a_3(s_3)$ $a_4(s_4)$ $a_5(s_5)$ $a_6(s_6)$ $a_7(s_7)$ $a_8(s_8)$ $a_9(s_9)$

Figure 2.1: 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 $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

variable given the k-gram (see next subsection). Since an abstraction is a set of k-grams, the class context of an abstraction will be obtained by aggregating the class contexts of its elements. Next, we define a distance $dist(a_i, a_j)$ between the class contexts of two abstractions a_i and a_j and identify the most "similar" abstractions as those that have the smallest distance between their class contexts.

2.2.1.1 Class Context for Abstractions

Given a transformed data set $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$ of $\mathcal{D}$, we define the *class context* of a k-gram s_i in $\mathcal{S}$ with respect to $\mathcal{D}$ as follows:

$$CContext_{\mathcal{D}}(s_i) := [p(Y|s_i), \#s_i] = \left[\left[\frac{\#[s_i, y_j]}{\sum_{y_j \in \mathcal{Y}} \#[s_i, y_j]} \right]_{y_j \in \mathcal{Y}}, \sum_{y_j \in \mathcal{Y}} \#[s_i, y_j] \right]$$

That is, the class context of a k-gram s_i is the conditional probability distribution of the class variable Y given the k-gram s_i, $p(Y|s_i)$ estimated from $\mathcal{D}$, along with the frequency counts of the k-gram s_i in $\mathcal{D}$, $\#s_i$. The variable Y takes values $y_j \in \mathcal{Y}$. The counts $\#[s_i, y_j]$ can be computed from $\mathcal{D}_{\mathcal{S}}$ by summing $\#s_i^l$ over all instances that belong to the class y_j in $\mathcal{Y}$. Intuitively, $p(Y|s_i)$ represents the context, while the frequency counts $\#s_i$ represent a weight that is used to aggregate the "contexts" of two or more k-grams appropriately.

More generally, we define the class context of an abstraction $a_i = \{s_{i_1,\ldots,}s_{i_q}\}$ as follows:

$$CContext_{\mathcal{D}}(\{s_{i_1,\cdots,}s_{i_q}\}) := \left[\sum_{r=1}^{q} \pi_r p(Y|s_{i_r}), \sum_{r=1}^{q} \#s_{i_r} \right]$$

$$\text{where } \pi_r := \frac{\#s_{i_r}}{\sum_{r=1}^{q} \#s_{i_r}}$$

If we "abstract out" the difference between all the k-grams $\{s_{i_1,\cdots,}s_{i_q}\}$ in the abstraction a_i and replace each of their occurrences in $\mathcal{D}$ by a_i, then $\#a_i = \sum_{r=1}^{q} \#s_{i_r}$ and $\#[a_i, y_j] = \sum_{r=1}^{q} \#[s_{i_r}, y_j]$. Furthermore, for any $y_j \in \mathcal{Y}$:

$$p(y_j|a_i) = \frac{\#[a_i, y_j]}{\#a_i} = \frac{\sum_{r=1}^{q} \#[s_{i_r}, y_j]}{\sum_{r=1}^{q} \#s_{i_r}} = \sum_{r=1}^{q} \frac{\#s_{i_r}}{\sum_{r=1}^{q} \#s_{i_r}} \frac{\#[s_{i_r}, y_j]}{\#s_{i_r}} = \sum_{r=1}^{q} \pi_r p(y_j|s_{i_r})$$

Hence, we have shown that:

$$CContext_{\mathcal{D}}(a_i) = [p(Y|a_i), \#a_i] = \left[\sum_{r=1}^{q} \pi_r p(Y|s_{i_r}), \sum_{r=1}^{q} \#s_{i_r}\right] = CContext_{\mathcal{D}}(\{s_{i_1,\cdots,}s_{i_q}\})$$

This is a natural definition of an abstraction based on "abstracting out" the difference between the k-grams $\{s_{i_1,\cdots,}s_{i_q}\}$ and considering them as being a single feature a_i. Note that we have also shown that $\sum_{r=1}^{q} \pi_r p(Y|s_{i_r})$ is actually a probability distribution over $\mathcal{Y}$ because it is the same as $p(Y|a_i)$. Next, we define a distance between the class contexts of two abstractions, motivated by ideas from information theory [Cover and Thomas, 1991].

2.2.1.2 Distance Between Abstractions

Our goal is to obtain compact, yet accurate models. If we denote by A a random variable that takes values in a set of abstractions $\mathcal{A} = \{a_1, \cdots, a_m\}$, our goal reduces to constructing the set $\mathcal{A}$ of abstractions such that the dependency between A and the class variable Y is preserved as much as possible. One way to measure the dependency between two variables is to use mutual information [Cover and Thomas, 1991]. Hence, we want to construct $\mathcal{A}$ such that the reduction in the mutual information between A and the class variable Y, $I(A, Y)$, is minimized at each step of Algorithm 2.

The reduction in the mutual information between A and Y due to a single merge of Algorithm 2 can be calculated as follows: Let $\mathcal{A}_m$ and $\mathcal{A}_{m-1}$ denote the sets of abstractions corresponding to two consecutive steps of Algorithm 2, or equivalently, corresponding to two cuts γ_m and γ_{m-1} through the abstraction hierarchy $\mathcal{T}$, respectively. Let $\{a_i, a_j\} \rightarrow a_u$ denote the merge that produced $\mathcal{A}_{m-1}$ from $\mathcal{A}_m$. Furthermore, let π_i and π_j denote the prior probabilities of a_i and a_j in the set a_u, i.e. $\pi_i = \frac{p(a_i)}{p(a_i)+p(a_j)}$ and $\pi_j = \frac{p(a_j)}{p(a_i)+p(a_j)}$.

Proposition 1: *The reduction in the mutual information between A and Y, due to the above merge is given by $\delta I(\{a_i, a_j\}, a_u) = (p(a_i) + p(a_j)) \cdot JS_{\pi_i,\pi_j}(p(Y|a_i), p(Y|a_j)) \geq 0$, where $JS_{\pi_i,\pi_j}(p(Y|a_i), p(Y|a_j))$ represents the weighted Jensen-Shannon divergence between two probability distributions $p(Y|a_i)$ and $p(Y|a_j)$ with weights π_i and π_j, respectively. (See Appendix A for a proof.)*

The weighted Jensen-Shannon divergence between two probability distributions p_i and p_j with weights π_i and π_j, is given by: $JS_{\pi_i,\pi_j}(p_i, p_j) = \pi_i KL(p_i||\bar{p}) + \pi_j KL(p_j||\bar{p})$, where $\bar{p} = \pi_i p_i + \pi_j p_j$, and $KL(p_i||\bar{p})$ represents the Kullback-Leibler divergence between p_i and $\bar{p}$. The weighted Jensen-Shannon divergence is a generalization of the Jensen-Shannon divergence between two probability distributions that allows for an asymmetry in the consideration of the two elements (in the case of standard Jensen-Shannon divergence $w_1 = w_2 = \frac{1}{2}$) [Lin, 1991].

Hence, we define the distance between two abstractions a_i and a_j, denoted by $dist(a_i, a_j)$, as follows:

$$dist(a_i, a_j) = \delta I(\{a_i, a_j\}, a_u) \text{ where } a_u = \{a_i \cup a_j\}$$

$$= (p(a_i) + p(a_j)) \cdot JS(CContext_{\mathcal{D}}(a_i), CContext_{\mathcal{D}}(a_j))$$

2.2.2 Algorithm Analysis

Recall that $\mathcal{S} = \{s_1, \cdots, s_n\}$ is the set of unique k-grams in $\mathcal{D}$, $n = |\mathcal{S}|$, and that $\mathcal{A} = \{a_1, \cdots, a_m\}$ is the set of constructed abstractions, $m = |\mathcal{A}|$. We start with the analysis of Algorithm 2 for constructing abstractions. The algorithm is a *greedy* procedure that searches for the most "similar" two abstractions at each step based on the "similarity" between the probability distributions of the class variable given the abstractions. The computation of $dist(a_i, a_j)$ takes $O(|\mathcal{Y}|)$ time. At each step of the algorithm, for each $\mathcal{A}_u = \{a_1 : \mathcal{S}_1, \cdots, a_u : \mathcal{S}_u\}$, $u = n, \cdots, m+1$, there are $\frac{u(u-1)}{2}$ possible pairs of abstractions, and for each pair we need to compute $dist(a_i, a_j)$. The computation time can be reduced by a factor of n by initially computing the distances $dist(a_i, a_j)$ between any two abstractions a_i and a_j in $\mathcal{A}_n$ (the trivial set of abstractions) and then, at each step, updating only the distances between pairs containing $a_{i_{min}}$ and $a_{j_{min}}$. Thus, the time complexity of Algorithm 2 is $O(n^2|\mathcal{Y}|)$.

Now we return to the analysis of Algorithm 3. After each round of adaptation of the data representation (which consists of super-structuring, performed by `Generate-Super-structures`($\mathcal{D}$), followed by feature abstraction, performed by `Construct-Abstractions`($\mathcal{D}_{\mathcal{S}}$,$m$)), the data instances are encoded in using the features resulting from this change in representation (as a result of `Encode`($\mathcal{D}, \mathcal{S}$) and `Encode`($\mathcal{D}_{\mathcal{S}}, \mathcal{A}$) steps of Algorithm 3). All the procedures of the algorithm except `Construct-Abstractions`($\mathcal{D}_{\mathcal{S}}$,$m$) maintain the "one pass through the data" property and take $O(\sum_{i=1}^{n} \#s_i)$ time ($\#s_i$ represents the frequency counts of the k-gram s_i in $\mathcal{D}$). Hence the time complexity of Algorithm 3 is given by the maximum between $O(\sum_{i=1}^{n} \#s_i)$ and $O(n^2|\mathcal{Y}|)$.

2.3 Feature Selection

In this section we discuss an alternative approach, i.e., feature selection, to reducing the feature set $\mathcal{S}$ of super-structures to a desired number of features m. Feature selection is performed by selecting a subset $\mathcal{F}$ of features (super-structures or k-grams) from the entire set $\mathcal{S}$, $\mathcal{F} \subseteq \mathcal{S}$ such that $|\mathcal{F}| = m$ and $m \leq n$. The features are ranked according to a scoring function *Score* and the top m best ranked features are selected.

We used mutual information [Cover and Thomas, 1991] between the probability of the class variable $p(Y)$ and the probability of the feature s_i, which measures how dependent the two variables are. More exactly, let $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], ..., [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$ be a transformed data set of $\mathcal{D}$, let Y be the class variable that takes values $y_j \in \mathcal{Y}$, and let $s_i \in \{s_1, \cdots, s_n\}$ be a feature. We define a scoring function *Score* of the feature s_i as follows:

$$Score_{\mathcal{D}}(s_i) = KL(p(s_i, Y)||p(s_i)p(Y))$$

where $p(s_i, Y)$ is estimated from counts gathered from $\mathcal{D}$ and $p(s_i)$ and $p(Y)$ are obtained by marginalization from $p(s_i, Y)$. As before, these counts can be computed from $\mathcal{D}_{\mathcal{S}}$. The $Score_{\mathcal{D}}(s_i)$ is also known as information gain because $KL(p(s_i, Y)||p(s_i)p(Y)) = H(p(Y)) - H(p(Y|s_i)|p(Y))$ where $H(p(Y)) = -\sum_{y_j} p(y_j) \log p(y_j)$ is the Shannon's entropy.

The data set $\mathcal{D}_{\mathcal{S}}$ is transformed into another data set $\mathcal{D}_{\mathcal{S}+\mathcal{F}}$ where each instance is represented by m features (besides the label) as follows: given the selected set of features $\mathcal{F} =$

$\{s_{i_1}, \cdots, s_{i_m}\}$, an instance $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ from $\mathcal{D}_{\mathcal{S}}$ is transformed into an instance $([s_{i_1} : \#s_{i_1}^l], ..., [s_{i_m} : \#s_{i_m}^l], y_l)$ in $\mathcal{D}_{\mathcal{S}+\mathcal{F}}$.

2.4 Experiments and Results

We proceed to describe two sets of experiments. In our first set of experiments, we compare Naïve Bayes (NB) and Support Vector Machine (SVM) classifiers trained using the combination of super-structuring and abstraction as the feature representation with their counterparts trained using a unigram representation of sequences, super-structuring representation, and super-structuring followed by feature selection representation. We show results of these comparisons on two protein subcellular localization data sets: **plant** and **non-plant**. Furthermore, in our second set of experiments, we compare NB trained using abstraction with NB trained using a bag of unigrams representation, and with NB trained using feature selection in a domain where the number of primary features is very large, i.e., text categorization, and show results on two data sets: **20 Newsgroups** and **WebKB**.

For the **plant**, **non-plant**, and **20 Newsgroups** data sets we report the average classification accuracy obtained in a 5-fold cross-validation experiment. For the **WebKB** data set we report the average classification accuracy obtained during 4 runs of a cross-validation experiment, each run using the pages from a university for testing and the pages from the remaining three universities plus the *misc* collection for training the classifier.

2.4.1 Data Sets

The **plant** and **non-plant** data sets[2], were first introduced in [Emanuelsson *et al.*, 2000]. The class associated with each protein sequence represents the subcellular compartment where the protein is located in the cell. The **plant** data set consists of 940 examples belonging to one of the following four classes: *chloroplast*, *mitochondrial*, *secretory pathway/signal peptide* and *other*. The **non-plant** data set consists of $2,738$ examples, each in one of the following three classes: *mitochondrial*, *secretory pathway/signal peptide,* and *other*. For both **plant** and

[2]Available at http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php

non-plant data sets, the size of the alphabet is 22, that is $|\mathcal{X}| = 22$ (20 amino acids plus two special characters inserted at the beginning and at the end of each sequence).

The **20 Newsgroups** data set[3] contains 20,000 UseNet articles collected by Ken Lang from 20 different discussion newsgroups [Joachims, 1997]. The task is to predict the newsgroup an article was posted to. While processing the data set, we skipped the UseNet headers, thus skipping the subject line. We then removed punctuation and words that contain numbers, and performed stemming and removal of stop words and words that occur only once in the whole collection. The vocabulary size is 46,524.

The **WebKB** data set[4] [Craven *et al.*, 1998] contains web pages collected from computer science departments of various universities. The web pages are classified into seven categories: *student*, *faculty*, *staff*, *department*, *course*, *project*, and *other*. In this study, we used the same subset of the data set as in [McCallum and Nigam, 1998] containing 4,199 web pages classified into one of the four categories: *student*, *faculty*, *course*, and *project*. We skipped the MIME headers and the *html* tags and performed stemming and removal of words that occur only once. Consistent with the results reported in [McCallum and Nigam, 1998], we found that removing stop words decreases performance of classifiers, hence we did not remove the stop words. The vocabulary size is 21,618.

2.4.2 Experimental Design and Results

The first set of experiments. The feature representations used to train the NB and SVM classifiers on **plant** and **non-plant** data sets are as follows:

- unigrams: a bag of letters representation of protein sequences, no super-structuring, abstraction or feature selection (UNIGRAM);
- super-structuring: a bag of k-grams ($k = 3$) representation of protein sequences (SS);
- super-structuring and feature selection: a bag of m k-grams ($k = 3$) chosen using feature selection from the bag of k-grams obtained by super-structuring (See Section 3 for details)

[3]Available at http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups
[4]Available at http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

(SS+FSEL);

- super-structuring and abstraction: a bag of m abstractions over k-grams ($k = 3$) obtained using the combination of super-structuring and abstraction (See Section 2 for details) (SS+ABS).

For **plant** and **non-plant** data sets the number of 3-grams is 8293 and 8404, respectively. Note that these represent the numbers of 3-grams that occur in the data. For both data sets, the number of unigrams is 22.

Figures 2.2a and 2.2b show the results of the comparison of SS+ABS with UNIGRAM, SS, and SS+FSEL on the **plant** data set using NB and SVM with linear kernel, respectively. Figures 2.2c and 2.2d show similar results on the **non-plant** data set. The average classification accuracy is shown as a function of the number of features used in the classification model, ranging from 1 to $n = |\mathcal{S}|$ (the number of unique k-grams). The x axis of Figure 2.2 shows the number of features on a logarithmic scale.

Comparison of SS+ABS with UNIGRAM. Figure 2.2 shows that, for any choice of the number of features, classifiers trained using SS+ABS outperform those trained using UNIGRAM. For example, on the **plant** data set, with 22 features, NB and SVM achieve 74.36% and 70.74% accuracy (respectively) using SS+ABS, as compared to 59.46% and 63.08% (respectively) obtained using UNIGRAM. Similarly, on the **non-plant** data set, with 22 features, NB and SVM achieve 77.24% and 77.61% accuracy (respectively) using SS+ABS, whereas NB and SVM achieve 67.42% and 73.59% (respectively) using UNIGRAM.

Comparison of SS+ABS with SS. As can be seen in Figure 2.2, SS+ABS matches the performance of SS with substantially smaller number of features. Specifically, the performance of SS-based NB and SVM classifiers trained using more than 8000 3-grams is matched by SS+ABS-based classifiers trained using only 22 and 8 features (respectively) on the **plant** data set, and only 270 and 2 features (respectively) on the **non-plant** data set. We conclude that SS+ABS can match the performance of SS using substantially (by one to three orders of magnitude) more compact classifiers.

The performance of SVM on SS is worse than that of NB on SS on both data sets, and also worse than that of SVM on UNIGRAM (Figure 2.2). This could be due to *overfitting* (see [Vapnik, 1998] for a theoretical analysis of overfitting for the SVM algorithm). It is interesting to note that, for many choices of the number of features, SS+ABS substantially outperforms SS in the case of SVM on both data sets (Figures 2.2b and 2.2d). Thus, SS+ABS can help avoid overfitting by providing more robust estimates of model parameters.

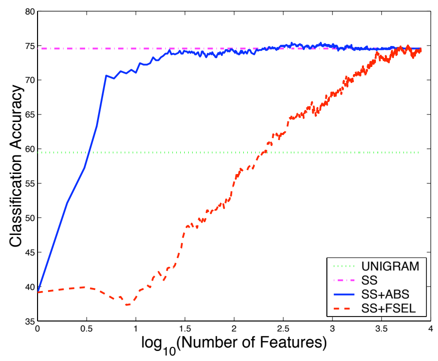
Comparison of SS+ABS with SS+FSEL. As can be seen from Figure 2.2, SS+ABS outperforms SS+FSEL over a broad range of choices for the number of features. For example, with only 10 features on the **plant** data set, NB using SS+ABS achieves an accuracy of 71.06% as compared to NB using SS+FSEL which achieves an accuracy of only 38.40%. On the same data set with only 10 features, SVM using SS+ABS achieves an accuracy of 67.87% as compared to 41.06% achieved by SVM using SS+FSEL.

The second set of experiments. The feature representations used to train the NB classifiers on **20 Newsgroups** and **WebKB** data sets are as follows:

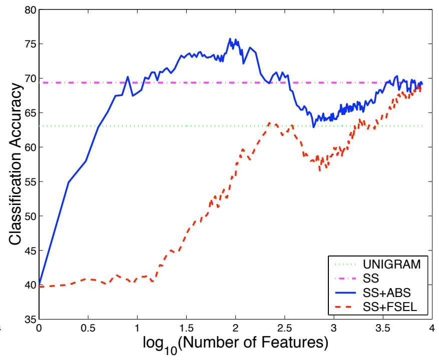
- unigrams: a bag of words representation of the text sequences, where feature selection is initially performed to reduce the original set of words to a smaller size set (UNIGRAM);
- feature selection: a bag of m words chosen using feature selection from the bag of unigrams (FSEL);
- abstraction: a bag of m abstractions over unigrams obtained by abstraction (ABS).

On text data, we found that selecting a subset of features as input for abstraction yields better performing models. To reduce the vocabulary size, we performed feature selection on each fold of cross-validation to select a set of $10,000$ features and 196 features with the highest information gain with the class variable for **20 Newsgroups** and **WebKB**, respectively.

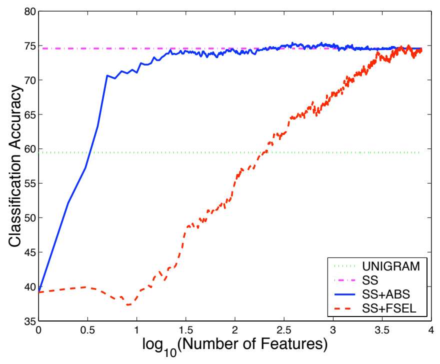
Similar to the work of Dumais et al. [Dumais *et al.*, 1998], we found that the performance of classifiers trained using SS was worse than that of classifiers trained using UNIGRAM. Hence, we compare ABS with FSEL and UNIGRAM (but no SS) on text data. Figures 2.3a and 2.3b



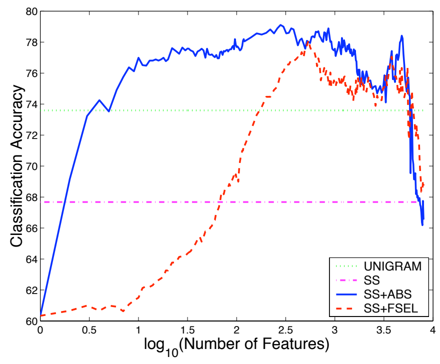
(a) **plant** (NB)



(b) **plant** (SVM)



(c) **non-plant** (NB)

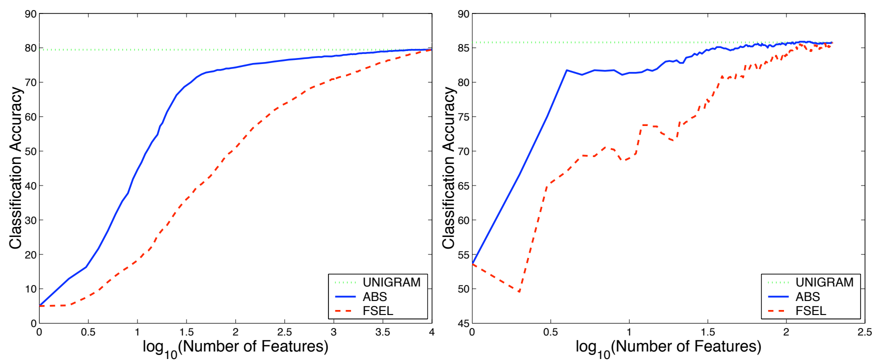


(d) **non-plant** (SVM)

Figure 2.2: Comparion of super-structuring and abstraction (SS+ABS) with super-structuring alone (SS), super-structuring and feature selection (SS+FSEL) and UNIGRAM on the **plant** and **non-plant** data sets using Naïve Bayes (NB), (a) and (c) respectively, and Support Vector Machines (SVM) with linear kernel (b) and (d), respectively. The plots show the accuracy as a function of the number of features used in the classification model, ranging from 1 to $\approx 8,000$ on both data sets. The x axis shows the number of features on a logarithmic scale.

show the results on the **20 Newsgroups** and **WebKB**, respectively, using NB [McCallum and Nigam, 1998].

Comparison of ABS with FSEL and UNIGRAM. As can be seen in Figure 2.3, for the same number of features, classifiers trained using ABS outperform those trained using FSEL on both data sets. For example, on **20 Newsgroups**, with only 100 features, NB achieves 74% accuracy using ABS, as compared to 51% obtained using FSEL. On **WebKB**, with only 20 features, NB using ABS achieves 83.18% accuracy, whereas NB using FSEL achieves 71.75%. NB using ABS approaches the performance of NB using UNIGRAM with



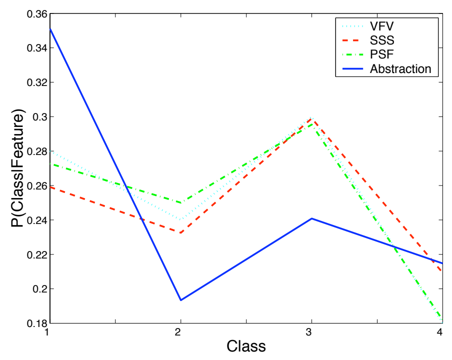
(a) **20 Newsgroups** (b) **WebKB**

Figure 2.3: Comparion of abstraction (ABS) with feature selection (FSEL) and UNIGRAM on the **20 Newsgroups** (a) and **WebKB** (b) data sets using Naïve Bayes. The plots show the accuracy as a function of the number of features used in the classification model, ranging from 1 to $10,000$ on **20 Newsgroups** and from 1 to 196 on **WebKB**. The x axis shows the number of features on a logarithmic scale.

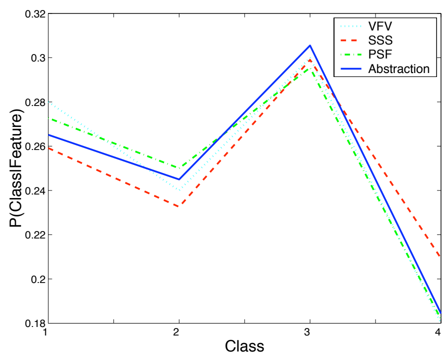
smaller numbers of features, more precisely, $3,000$ features (78.96% vs. 79.59% accuracy) on **20 Newsgroups**, and 35 features (85.11% vs. 85.78% accuracy) on **WebKB**. Hence, the performance of ABS can approach that of UNIGRAM with a reduction of model complexity by one order of magnitude.

2.4.3 Analysis of Abstractions

To gain some insight into the data representation generated by the combination of super-structuring and abstraction, we took a closer look at the k-grams that are grouped into a particular abstraction. Let us consider that abstraction is performed (after super-structuring) to reduce the number of features (or abstractions) from $8,293$ to 10 on the **plant** data set. Figure 2.4a shows the class distribution induced by one of the 10 abstractions, namely a_1, and the class distributions induced by three 3-grams sampled from the abstraction a_1, namely “VFV”, “SSS”, “PSF”. Note that the class distribution induced by the abstraction a_1 is very similar to those induced by the individual 3-grams (the average number of 3-grams per abstraction is 829). Hence, the individual 3-grams can be replaced by their abstraction a_1 without significant reduction in the accuracy of the resulting classifiers (67.87% accuracy for SS+ABS with 10



(a) 10 abstractions



(b) 100 abstractions

Figure 2.4: Class probability distributions induced by one of the m abstractions, namely a_i, and by three 3-grams, namely "VFV", "SSS", and "PSF", on the **plant** data set, where (a) $m = 10$ and $i = 1$; and (b) $m = 100$ and $i = 3$. The three 3-grams are initially sampled from a_3 (when $m = 100$). The number of classes in the data set is 4.

features vs. 69.36% for SS using SVM). In contrast, selecting a subset of 10 features by feature selection (after super-structuring) decreases classification performance (41.06% accuracy for SS+FSEL with 10 features vs. 67.87% for SS+ABS also with 10 features using SVM).

When we performed a similar analysis to reduce the number of abstractions from $8,293$ to 100 on the same data set, we found (as expected) that the class distributions induced by the same three 3-grams, "VFV", "SSS", "PSF", are more similar to the class distribution induced by their abstraction, namely, a_3 (Figure 2.4b) (the average number of 3-grams per abstraction is 83). Going from a coarser to a finer set of abstractions (e.g., 10 to 100), the class distributions induced by individual 3-grams become more and more similar to that induced by their abstraction (until convergence). We conclude that SS+ABS provides better estimates of the model parameters than SS+FSEL. For example, SS+ABS achieves 75.63% accuracy with 100 features using SVM, whereas SS achieves 69.36%. of abstractions increases, the performance of SS+ABS drops (64.14% accuracy with 1000 features using SVM).

2.5 Summary and Discussion

We have presented an approach to feature construction in a sequence classification scenario to trade off the predictive accuracy of the learned model against its complexity. Our approach combines *super-structuring* with *abstraction*. Super-structuring involves generating all the contiguous (potentially overlapping) subsequences of a certain length k, a.k.a. k-grams. Abstraction involves constructing more abstract features by grouping "similar" features.

The results of our experiments on two protein subcellular localization data sets and two text categorization data sets show that adapting data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those obtained using super-structuring alone (whose size can grow exponentially with k). Moreover, combining super-structuring and abstraction yields models whose performance is similar and, in some cases, even better compared to that obtained by using super-structuring alone. In this case, abstraction can be seen as a *regularizer*. The results also show that simplifying data representation using abstraction yields better performing models than those obtained by feature selection.

2.5.1 Related Work

Segal et al. [Segal *et al.*, 2002] and McCallum et al. [McCallum *et al.*, 1998] have used abstraction hierarchies over *classes* to improve classification accuracy. Zhang et al. have used abstraction hierarchies over nominal variables to build compact, yet accurate decision tree [Zhang and Honavar, 2003] and Naïve Bayes [Zhang *et al.*, 2006] classifiers. In contrast, we have learned abstraction hierarchies from sequence data, that is a type of topologically constrained data. Our abstraction hierarchies group "similar" k-grams based on the class conditional distributions that the k-grams induce.

Our work is similar in spirit to the *agglomerative information bottleneck* (AIB) method of Slonim and Tishby [Slonim and Tishby, 1999]. However, AIB requires an iterative process to estimate parameters for the *bottleneck variables*, whereas our method requires only a simple weighted aggregation of the existing parameters.

Feature construction methods aim to increase the richness of the data representation [Rendell and Seshu, 2007], [Piramuthu and Sikora, 2008] (see [Liu and Motoda, 1998a] for a survey).

Super-structuring corresponds to considering higher order moments [Mardia *et al.*, 1979] in the particular case of multivariate statistics. Feature selection (or extraction) methods that select a subset of the available features based on some chosen criteria [Liu *et al.*, 2008], [Yu and Liu, 2004] [Peng *et al.*, 2005], [Koller and Sahami, 1996] (see [Liu and Motoda, 1998b] for a survey) and feature abstraction methods that group *similar* features to generate more abstract features [Kang *et al.*, 2004], [Slonim and Tishby, 1999], [Baker and McCallum, 1998] aim to control the level of detail in the representation. With the exception of Jonyer et al. [Jonyer *et al.*, 2004] who have described an algorithm for Context-Free Graph Grammar Induction that combines in effect super-structuring and abstraction (although not explicitly, and not aimed at predicting a target class variable), there has been relatively limited exploration of techniques that sequentially combine super-structuring with feature abstraction or feature selection in adapting the data representation used for training predictive models to obtain accurate and, at the same time, comprehensible classifiers.

2.5.2 Discussion

Super-structuring is used to enrich the data representation, and hence increase the performance of learned models. In this study, we have focused on super-structuring in the case of sequence data. However, in general, any topology that reflects the interactions between elements of structured data can be used to guide super-structuring. For example, in the case of image data, this involves generating features from spatially contiguous elements of an image.

Feature selection or abstraction simplify the data representation, and hence help maintaining comprehensible models. Although these two approaches to reducing model complexity are seemingly antagonistic, better value can be derived by using them sequentially whereas feature selection is used in order to reduce an initial large set of features to a medium size set followed by abstraction that can further reduce the size with minor or no loss in predictive accuracy.

Possible directions for further research include extensions of the methods developed here for sequence classification to settings where the data instances have a richer structure (e.g., multi-modal data consisting of images and text, and combination of audio and visual data).

2.6 Appendix

Lemma 1: Let X and Z be two random variables such that Z can take on k possible values. Let $p(z_i)$ be the prior probability of z_i and $p(x|z_i)$ be the conditional distribution of X given z_i for $i = 1, \cdots, k$. Then:

$$JS_{p(z_1),\cdots,p(z_k)}\left(p(x|z_1),\cdots,p(x|z_k)\right) = H\left(\sum_{i=1}^{k} p(z_i)p(x|z_i)\right) - \sum_{i=1}^{k} p(z_i)H\left(p(x|z_i)\right) = I(X;Z)$$

where $H(\cdot)$ is the Shannon's entropy [Lin, 1991].

Proof of Proposition 1: Without loss of generality, let us assume that the merge is $\{a_1, a_2\} \to a$.

Let $\delta I(\{a_1, a_2\}, a) = I(A^{(\mathcal{A}_m)}, Y) - I(A^{(\mathcal{A}_{m-1})}, Y)$ denote the reduction in the mutual information $I(A;Y)$, where $A^{(\mathcal{A}_m)}$ represents the variable A that takes values in the set $\mathcal{A}_m = \{a_1, \cdots, a_m\}$. We use the above lemma. Hence,

$\delta I(\{a_1, a_2\}, a)$

$$\begin{aligned}
&= JS_{p(a_1),p(a_2),\cdots,p(a_m)}\left[p(y|a_1), p(y|a_2), \cdots, p(y|a_m)\right] - JS_{p(a),\cdots,p(a_m)}\left[p(y|a), \cdots, p(y|a_m)\right] \\
&= H\left(\sum_{i=1}^{m} p(a_i)p(y|a_i)\right) - \sum_{i=1}^{m} p(a_i)H\left(p(y|a_i)\right) - H\left(p(a)p(y|a) + \sum_{i=3}^{m} p(a_i)p(y|a_i)\right) \\
&\quad + p(a)H\left(p(y|a)\right) + \sum_{i=3}^{m} p(a_i)H\left(p(y|a_i)\right) \\
&= H\left(\sum_{i=1}^{m} p(a_i)p(y|a_i)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) - H\left(p(a)\frac{1}{p(a)}\sum_{i=1}^{2} p(y,a_i) + \sum_{i=3}^{m} p(a_i)p(y|a_i)\right) \\
&\quad + p(a)H\left(p(y|a)\right) \\
&= p(a)H\left(p(y|a)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) \\
&= p(a)H\left(\frac{1}{p(a)}\sum_{i=1}^{2} p(y,a_i)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) \\
&= p(a)\left(H\left(\sum_{i=1}^{2} \frac{p(a_i)}{p(a)}p(y|a_i)\right) - \sum_{i=1}^{2} \frac{p(a_i)}{p(a)}H\left(p(y|a_i)\right)\right) \\
&= (p(a_1) + p(a_2)) \cdot JS_{\pi_1,\pi_2}(p(Y|a_1), p(Y|a_2)).
\end{aligned}$$

CHAPTER 3. ABSTRACTION AUGMENTED MARKOV MODELS

We present abstraction augmented Markov models (AAMMs), which are directed acyclic graphical models that simplify the data representation used by the standard Markov models (MMs). AAMMs group similar entities to generate more abstract entities that are organized in an abstraction hierarchy. *Abstraction* reduces the Markov model size and improves the statistical estimates of complex models by reducing the number of parameters to be estimated from data. We evaluate the AAMMs on three protein subcellular localization prediction tasks. The results of our experiments show that: (1) AAMMs can achieve significantly lower model sizes (by 1 to 3 orders of magnitude) for a minor drop in accuracy over the standard MMs, and in some cases even higher accuracy while simultaneously lowering the model size; and (2) AAMMs substantially outperforms MMs in settings where only a small fraction of available data is labeled.

3.1 Introduction

Many real-world problems can be regarded as sequence classification tasks. For example, in computational biology predicting protein function or protein subcellular localization can be addressed as sequence classification problems, where the amino acid sequence of the protein is used to classify a protein in functional or localization classes [Baldi and Brunak, 2001]. 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 [Durbin *et al.*, 2004]. They capture dependencies between neighboring elements in a sequence and thus provide more accurate models of the data. In fixed-order MMs which are special cases of directed probabilistic graphical models, the elements of a sequence satisfy the *Markov property*: Each element in the sequence directly depends on a fixed number of previous elements, called *parents*, and is independent of the rest of the elements in the sequence. MMs have been successfully applied in many applications including natural language processing [Charniak, 1993] and molecular sequence classification [Durbin *et al.*, 2004]. Interpolated MMs [Jelinek and Mercer, 1980] combine several fixed-order MMs that capture important sequence patterns that would otherwise be ignored by a single fixed-order MM.

While dependencies between neighboring elements provide a way to improve the predictive accuracy, the number of model parameters increases exponentially with the range of direct dependencies, thereby increasing the risk of *overfitting* when the data set is limited in size. We present abstraction augmented Markov models (AAMM) aimed at addressing this difficulty by reducing the number of model parameters through *abstraction.* AAMMs construct an abstraction hierarchy over the values of the *parents* of each node using a hierarchical agglomerative clustering algorithm. The AAMM clustering algorithm groups the values of the *parents* of a node based on the similarity between the conditional distributions of the node given the *parents.* The graphical structure of AAMMs extends that of the standard MMs by constructing new variables that represent abstractions over the values of the *parents* of each node.

We evaluate AAMMs on three protein subcellular localization prediction tasks. The results of our experiments show that adapting the data representation by abstraction makes it possible to construct predictive models that use substantially smaller number of parameters (by 1 to 3 orders of magnitude) than standard MMs without sacrificing predictive accuracy. Our results also show that AAMM substantially outperforms MMs in settings where there exist only a small fraction of labeled data, but a vast amount of unlabeled data. Furthermore, the results show that organizing the sequence data in an abstraction hierarchy using the AAMM clustering algorithm produces more suitable abstraction hierarchies for AAMMs than Agglomerative Information Bottleneck [Slonim and Tishby, 1999], which is an alternative approach used to construct abstraction hierarchies.

The rest of the paper is organized as follows. In Section 2, we review the higher order

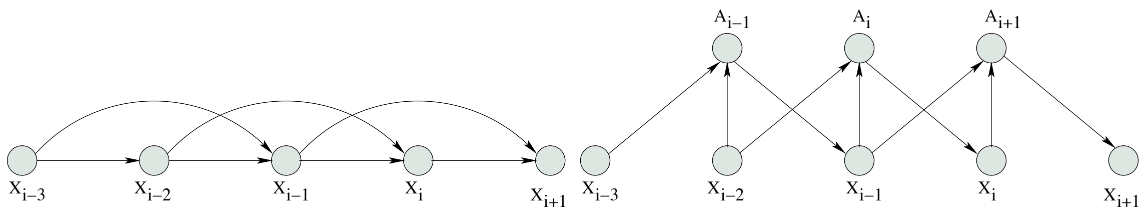


Figure 3.1: (a) 2^{nd} Order Markov Model; (b) 2^{nd} Order Abstraction Augmented Markov Model

Markov models and present our abstraction augmented Markov model. We present the experimental design and results on three sequence classification tasks in Section 3. Section 4 provides an analysis of abstractions obtained using the AAMM clustering algorithm. We conclude in Section 5 with a summary, discussion and future directions.

3.2 From Markov Models To Abstraction Augmented Markov Models

3.2.1 Markov Models

Let $\mathbf{x} = x_0, \cdots, x_{n-1}$ be an input sequence over an alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$. A Markov Model represents the joint probability distribution of $\mathbf{x}$ under the Markov assumption. The graphical representation of a Markov model is a directed linear-chain graph where the nodes X_i in the graph are random variables corresponding to the sequence elements x_i, $i = 0, \cdots, n-1$, and the edges represent direct dependencies between neighboring elements. The directed graph for a 2^{nd} order Markov model on a subset of nodes of sequence $\mathbf{x}$, $\{X_{i-3}, \cdots, X_{i+1}\}$, i.e., one that assumes 3^{rd} order dependencies, is shown in Figure 3.1a. The conditional independencies implied by the graph are:

$$X_i \perp\!\!\!\perp \{X_0, \cdots, X_{i-3}\} \,|\, \{X_{i-2}, X_{i-1}\} \text{ for } i = 2, \cdots, n-1$$

That is, X_i is conditionally independent of $X_0, \cdots, X_{i-3}$ given X_{i-2} and X_{i-1} for any $i = 2, \cdots, n-1$. More generally, for a k^{th}-order Markov model, the conditional independencies implied by the graph are:

$$X_i \perp\!\!\!\perp \{X_0, \cdots, X_{i-k-1}\} \,|\, \{X_{i-k}, \cdots, X_{i-1}\} \text{ for } i = k, \cdots, n-1$$

Thus, a node X_i in the graph depends on its k preceding nodes $X_{i-k}, \cdots, X_{i-1}$ and is independent of the other preceding nodes $X_0, \cdots, X_{i-k-1}$. The k preceding nodes are called the *parents* of X_i. For a k^{th}-order Markov model, the conditional distribution of a node X_i, $i = k, \cdots, n-1$, is:

$$p(X_i|X_0, \cdots, X_{i-1}) \quad = \quad p(X_i|X_{i-k}, \cdots, X_{i-1})$$

The joint probability distribution $p(\mathbf{X})$, where $\mathbf{X}$ denotes the set of nodes, can be factorized as:

$$p(\mathbf{X}) = p(X_0, \cdots, X_{k-1}) \prod_{i=k}^{n-1} p(X_i|X_{i-k}, \cdots, X_{i-1})$$

3.2.2 Abstraction Augmented Markov Models

Abstraction is the operation of grouping similar entities to generate more abstract entities. Let S_{i-1} denote the *parents* $X_{i-k} \cdots X_{i-1}$ of X_i in a standard k^{th} order Markov model. The values of S_{i-1} represent instantiations of $X_{i-k} \cdots X_{i-1}$ for $i = k, \cdots, n-1$. These instantiations are substrings of length k over the finite alphabet $\mathcal{X}$, a.k.a. k-grams. We denote by $\mathcal{S}$ the set of k-grams over $\mathcal{X}$. Thus, the cardinality of $\mathcal{S}$ is $N = |\mathcal{X}|^k$. To reduce the cardinality of the set $\mathcal{S}$, abstraction augmented Markov models (AAMMs) construct an abstraction hierarchy over the set $\mathcal{S}$ of k-grams.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ associated with a set of k-grams $\mathcal{S}$ is a rooted tree such that: (1) The root of $\mathcal{T}$ corresponds to the abstraction that consists of all k-grams in $\mathcal{S}$; (2) The tree $\mathcal{T}$ has exactly N leaves corresponding to the N k-grams in $\mathcal{S}$; (3) The internal nodes of $\mathcal{T}$ correspond to abstractions over k-grams (i.e., subsets of "similar" k-grams); (4) The edges of $\mathcal{T}$ correspond to partial order relationships $\prec$ (e.g., ISA relationships) between their corresponding nodes.*

Figure 3.2 shows an example of an abstraction hierarchy $\mathcal{T}$ 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 the abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ satisfying the following properties: (1) For any leaf s_i, either $s_i \in \gamma_m$ or s_i is a descendant of a node $a_j \in \gamma_m$; and (2) for any two nodes $a_j, a_l \in \gamma_m$, a_j is neither a*

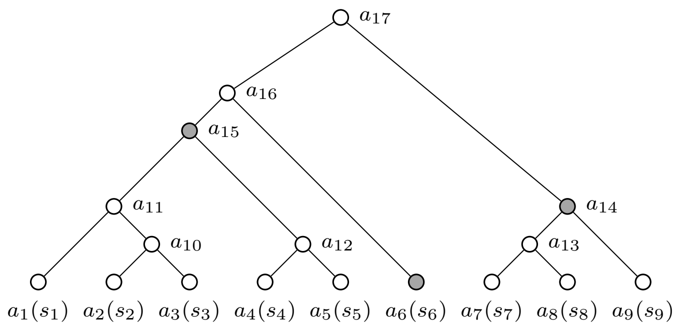

a_{17}
a_{16}
a_{15}
a_{11}
a_{10}
a_{12}
a_{14}
a_{13}
$a_1(s_1)$ $a_2(s_2)$ $a_3(s_3)$ $a_4(s_4)$ $a_5(s_5)$ $a_6(s_6)$ $a_7(s_7)$ $a_8(s_8)$ $a_9(s_9)$

Figure 3.2: 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 $\{a_{15}, a_6, a_{14}\}$ represents a 3-cut γ_3 through $\mathcal{T}$.

descendant nor an ancestor of a_l. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of the set $\mathcal{S}$ of k-grams.

In Figure 3.2, 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 the standard Markov models by constructing new variables A_i that represent abstractions over the set of values of the variables S_{i-1} for $i = k, \cdots, n-1$. More precisely, the variables A_i take values in a 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 A_i and S_{i-1}, respectively. Furthermore, in the AAMMs, the node X_i directly depends on A_i instead of being directly dependent on S_{i-1}, as in the standard Markov models. We define the conditional distribution of the node X_i induced by A_i as $p(X_i = x_i | A_i = a_i) = \theta_{xa}^{(i)}$, where $x_i \in \mathcal{X}$ and $a_i \in \mathcal{A}$ represent instantiations of variables X_i and A_i, respectively. The prior distribution of S_{k-1} is defined as $p(S_{k-1}) = \theta_s$.

The directed graph for a 2^{nd} order AAMM on a subset of variables $\mathbf{X} \cup \mathbf{A}$ is shown in Figure 3.1b. The joint probability distribution over the entire set of variables can be factorized as follows:

$$p(\mathbf{X}, \mathbf{A}) = p(S_{k-1}) \cdot \prod_{i=k}^{n-1} p(X_i | A_i) \cdot p(A_i | S_{i-1})$$

In what follows we show how to learn AAMMs from data which involves two steps: learning abstraction hierarchies (AHs) and learning model parameters.

3.2.2.1 Learning Abstraction Hierarchies

The procedure for constructing AHs over the set $\mathcal{S}$ of k-grams (values that any S_{i-1} can take) is shown in Algorithm 3. The *input* of the algorithm consists of the set $\mathcal{S}$ and a set $\mathcal{D}$ of sequences over the finite alphabet $\mathcal{X}$, $\mathcal{D} = \{x_l\}$, $\mathbf{x}_l \in \mathcal{X}^\star$. The *output* of the algorithm is a binary tree $\mathcal{T}$, specifically an AH over the set $\mathcal{S}$.

The algorithm starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction a_j corresponds to a k-gram s_j in $\mathcal{S}$, $j = 1, \cdots, N$. For each k-gram s_j, or equivalently for each abstraction a_j, the algorithm creates a node in the tree $\mathcal{T}$. Pairs of abstractions are then recursively merged until one abstraction is obtained. Specifically, $N-1$ times, the algorithm searches for the most "similar" two abstractions $a_{u_{min}}$ and $a_{v_{min}}$, adds a new abstraction a_w to $\mathcal{A}$ by taking the union of their k-grams, and removes $a_{u_{min}}$ and $a_{v_{min}}$ from $\mathcal{A}$. Simultaneously, the nodes in the tree corresponding to $a_{u_{min}}$ and $a_{v_{min}}$ are merged into a newly created node which is their parent. After $N-1$ steps, the algorithm returns the AH $\mathcal{T}$ over the set $\mathcal{S}$ of k-grams. We store the resulting AH $\mathcal{T}$ in a last-in-first-out (LIFO) stack. For a given choice of the size m of an m-cut through $\mathcal{T}$, we can easily extract the set of abstractions that define an AAMM, by discarding $m-1$ elements from the top of the stack.

To complete the description of our algorithm we need to define the similarity between abstractions. We define a distance $d_{\mathcal{D}}(a_u, a_v)$ between the *contexts* of two abstractions a_u and a_v and identify the most "similar" abstractions as those that have the smallest distance between their contexts. The *context* of an abstraction is defined as the conditional probability of a node X_i (that takes values in $\mathcal{X}$) given the abstraction (see below). Since an abstraction is a set of k-grams, we define the context of such a set by aggregating the contexts of its constituent elements.

Algorithm 3 Abstraction Construction

```
Input: A set of $k$-grams $\mathcal{S} = \{s_1, \cdots, s_N\}$; a set of sequences $\mathcal{D} = \{\mathbf{x}_l\}$, $\mathbf{x}_l \in \mathcal{X}^*$
Output: An abstraction hierarchy $\mathcal{T}$ over $\mathcal{S}$
Initialize $\mathcal{A} = \{a_1 : \{s_1\}, \cdots, a_N : \{s_N\}\}$, and $\mathcal{T} = \{a_1 : \{s_1\}, \cdots, a_N : \{s_N\}\}$
for $w = N+1$ to $2N-1$ do
  $(u_{min}, v_{min}) = \arg\min_{u,v \in \mathcal{A}} d_{\mathcal{D}}(a_u, a_v)$
  $a_w = a_{u_{min}} \cup a_{v_{min}}$
  $\mathcal{A} = \mathcal{A} \backslash \{a_{u_{min}}, a_{v_{min}}\} \cup \{a_w\}$
  $\mathcal{T} = \mathcal{T} \cup \{a_w\}$ s.t. $par(a_{u_{min}}) = a_w$, $par(a_{v_{min}}) = a_w$
end for
```

Context of an abstraction

Given a set $\mathcal{D} = \{\mathbf{x}_l\}$ of sequences, we define the context of a k-gram $s_j \in \mathcal{S}$, $j = 1, \cdots, N$, with respect to $\mathcal{D}$ as follows:

$$Context_{\mathcal{D}}(s_j) := [p(X_i|s_j), \#s_j] = \left[\left[\frac{\#[s_j, x_i]}{\sum_{x_i \in \mathcal{X}} \#[s_j, x_i]} \right]_{x_i \in \mathcal{X}}, \sum_{x_i \in \mathcal{X}} \#[s_j, x_i] \right]$$

That is, the context of a k-gram s_j wrt $\mathcal{D}$ is the conditional distribution of X_i given s_j, $p(X_i|s_j)$ estimated from $\mathcal{D}$, along with the frequency counts of the k-gram s_j in $\mathcal{D}$, $\#s_j$. Intuitively, $p(X_i|s_j)$ represents the context, while the frequency counts $\#s_j$ represent a weight that is used to aggregate the "contexts" of two or more k-grams appropriately.

More generally, we define the context of an abstraction $a_j = \{s_{j_1}, \cdots, s_{j_q}\}$, $a_j \in \mathcal{A}$, as:

$$Context_{\mathcal{D}}(a_j) := \left[\sum_{r=1}^{q} \pi_r \cdot p(X_i|s_{j_r}), \sum_{r=1}^{q} \#s_{j_r} \right] \text{ where } \pi_r := \frac{\#s_{j_r}}{\sum_{r=1}^{q} \#s_{j_r}}.$$

Note that if we "abstract out" the difference between all the k-grams in the abstraction a_j and replace their occurrences in $\mathcal{D}$ by a_j, then $\#a_j = \sum_{r=1}^{q} \#s_{j_r}$ and $\#[a_j, x_i] = \sum_{r=1}^{q} \#[s_{j_r}, x_i]$. Furthermore, for any $x_i \in \mathcal{X}$:

$$p(x_i|a_j) = \frac{\#[a_j, x_i]}{\#a_j} = \frac{\sum_{r=1}^{q} \#[s_{j_r}, x_i]}{\sum_{r=1}^{q} \#s_{j_r}} = \sum_{r=1}^{q} \frac{\#s_{j_r}}{\sum_{r=1}^{q} \#s_{j_r}} \frac{\#[s_{j_r}, x_i]}{\#s_{j_r}} = \sum_{r=1}^{q} \pi_r \cdot p(x_i|s_{j_r})$$

Hence, we have shown that $Context_{\mathcal{D}}(a_j) = [p(X_i|a_j), \#a_j]$. Next, we define a distance between contexts of two abstractions, motivated by ideas from information theory [Cover and Thomas, 1991].

Distance between abstractions

Our goal is to find an abstraction (or compression) A_i of the variable S_{i-1} and, at the same time, preserve the direct dependency between A_i and X_i as much as possible. One way to measure the dependency between two variables is to use mutual information [Cover and Thomas, 1991]. Hence, we want to construct an abstraction A_i of S_{i-1} such that the reduction in the mutual information between A_i and X_i, $I(A_i, X_i)$, is minimized at each step of Algorithm 3.

The reduction in mutual information between a node and its *parents* (in the graphical model corresponding to an AAMM) due to a single merge of Algorithm 3 can be calculated as follows: Let γ_m be an m-cut through the AH $\mathcal{T}$ and γ_{m-1} be the $(m-1)$-cut through $\mathcal{T}$ that results after one merge $\{a_u, a_v\} \rightarrow a_w$. Let $\mathcal{A}_m$ and $\mathcal{A}_{m-1}$ denote the set of abstractions corresponding to γ_m and γ_{m-1}, respectively. Furthermore, let π_u and π_v denote the prior probabilities of a_u and a_v in the set a_w, i.e. $\pi_u = \frac{p(a_u)}{p(a_u)+p(a_v)}$ and $\pi_v = \frac{p(a_v)}{p(a_u)+p(a_v)}$.

Proposition 1:

The reduction in the mutual information between each variable X_i and its parent, due to the above merge is given by $\delta I(\{a_u, a_v\}, a_w) = (p(a_u) + p(a_v)) \cdot JS_{\pi_u,\pi_v}(p(X_i|a_u), p(X_i|a_v)) \geq 0$, where $JS_{\pi_u,\pi_v}(p(X_i|a_u), p(X_i|a_v))$ represents the weighted Jensen-Shannon divergence between two probability distributions $p(X_i|a_u)$ and $p(X_i|a_v)$ with weights π_u and π_v, respectively [Lin, 1991].

We define the distance between two abstractions a_u and a_v in $\mathcal{D}$, denoted by $d_{\mathcal{D}}(a_u, a_v)$, as follows:

$$d_{\mathcal{D}}(a_u, a_v) \quad = \quad \delta I(\{a_u, a_v\}, a_w) \text{ where } a_w = \{a_u \cup a_v\}.$$

The effect of a single merge step of Algorithm 3 on the log likelihood of the data is given by the following proposition.

Proposition 2:

The reduction in the log likelihood of the data due to the merge $\{a_u, a_v\} \rightarrow a_w$ *is given by* $\delta LL(\{a_u, a_v\}, a_w) = M \cdot (p(a_u) + p(a_v)) \cdot JS_{\pi_u,\pi_v}(p(X_i|a_u), p(X_i|a_v)) \geq 0$*, where* M *is the cardinality of the* multiset *of* $(k+1)$*-grams. The likelihood function is given by* $L(\mathbf{x}, \mathbf{a}) = \prod_{|\mathcal{D}|} p(x_0 \cdots x_{k-1}) \cdot \prod_{i=k}^{n-1} p(x_i|a_i) = p(x_0 \cdots x_{k-1})^{\#[x_0 \cdots x_{k-1}]} \cdot \prod_{x_i \in \mathcal{X}, a_i \in \mathcal{A}_m} p(x_i|a_i)^{\#[a_i, x_i]}$. *(See* **Appendix** *section for the proof sketch of* **Propositions 1** *and* **2***.)*

Algorithm Analysis

Recall that $\mathcal{S} = \{s_1, \cdots, s_N\}$ is the set of unique k-grams in $\mathcal{D}$, $N = |\mathcal{S}|$, and that $\mathcal{A} = \{a_1, \cdots, a_m\}$ is the set of constructed abstractions, $m = |\mathcal{A}|$. Our algorithm is a *greedy* procedure that searches for the most "similar" two abstractions at each step based on the "similarity" between the probability distributions of an element in the alphabet $\mathcal{X}$ given the abstractions. The computation of $d_{\mathcal{D}}(a_u, a_v)$ takes $O(|\mathcal{X}|)$ time. At each step of the algorithm, for each $\mathcal{A}_w = \{a_1 : \mathcal{S}_1, \cdots, a_w : \mathcal{S}_w\}$, $w = N, \cdots, m+1$, there are $\frac{w(w-1)}{2}$ possible pairs of abstractions, and for each pair we need to compute $d_{\mathcal{D}}(a_u, a_v)$. The computational time can be reduced by a factor of N by initially computing the distances $d_{\mathcal{D}}(a_u, a_v)$ between any two abstractions a_u and a_v in $\mathcal{A}_N$ (the trivial set of abstractions) and then, at each step, updating only the distances between pairs containing $a_{u_{min}}$ and $a_{v_{min}}$. Thus, the time complexity of Algorithm 3 is $O(N^2|\mathcal{X}|)$.

3.2.3 Learning AAMM Parameters

The AAMM is a completely observable directed graphical model, i.e. there are no hidden variables in the model. Because the joint probability distribution of the model is written as a product of local conditional probabilities of each node given its parents, the parameters $\theta_{xa}^{(i)}$ can be estimated independently of each other using maximum likelihood estimation [Casella and Berger, 2002]. Learning AAMMs reduces to collecting the *sufficient statistics* $\#[s_j, x_i]$ from the set of training sequences, for all $s_j \in \mathcal{S}$ and $x_i \in \mathcal{X}$. Furthermore, for an abstraction $a_j = \{s_{j_1}, \cdots, s_{j_q}\}$, the *sufficient statistics* $\#[a_j, x_i]$ are $\sum_{r=1}^{q} \#[s_{j_r}, x_i]$. The marginal counts of s_j and a_j are obtained by summing out x_i. The parameter θ_s is estimated as in the standard Markov model. We used Laplace correction to avoid zero probabilities.

Given a new sequence $\mathbf{x} = x_0, \cdots, x_{n-1}$ and an m-cut γ_m through $\mathcal{T}$, the joint probability $p(\mathbf{x})$ can be computed as follows: parse the sequence from left to right. For each k-gram $x_{i-k}, \cdots, x_{i-1}$ find the abstraction $a_j \in \gamma_m$ it belongs to and retrieve the parameters associated with that abstraction. Multiply $p(x_i|a_j)$ for $i = k, \cdots, n-1$.

3.2.4 Using AAMMs for Classification

The AAMMs can be used for classification tasks by learning a model for each class and selecting the model with the highest posterior probability when classifying new data. In an AAMM, when the training data set is fully labeled, a class-specific AH can be constructed separately for each class. When the training data set is unlabeled, or only partially labeled, one AH can be constructed from the entire data set.

3.3 Experiments and Results

3.3.1 Experimental Design

We proceed to describe three sets of experiments that are designed to explore the effectiveness of AAMMs in generating simple and accurate sequence models.

In our first set of experiments, we compared the performance of AAMM with that of MM and Naïve Bayes on classification tasks in a setting where each sequence in the training data set is labeled with a class. In this case, we trained an AAMM for each class (based on the class-specific AH for that class).

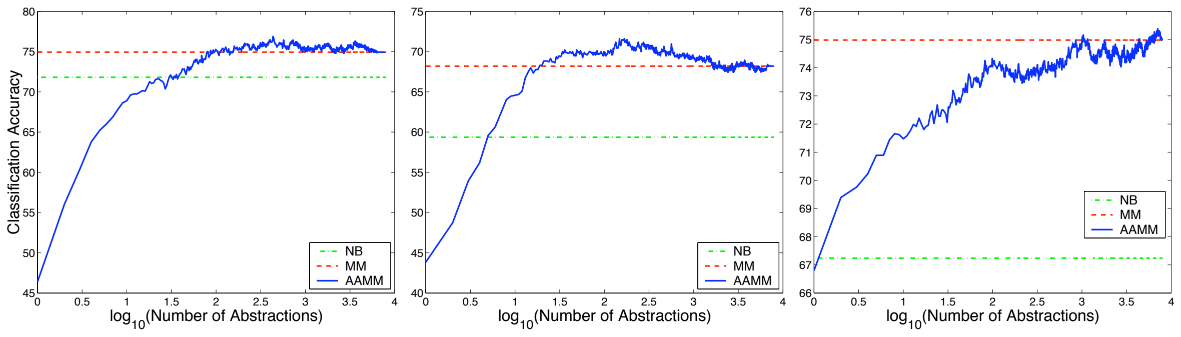
In our second set of experiments, we investigated the effect of *abstraction* on the performance of AAMMs in settings where only a subset of the training data is labeled and the rest are unlabeled. We performed experiments with 1%, 10%, 25%, and 100% of the training data set being used as labeled instances, with the rest being treated as unlabeled instances (by ignoring the class labels). To obtain the subsets of 1%, 10%, and 25% of labeled instances, we sampled instances, using a uniform distribution, from the training set. Because the class labels are unavailable for a large fraction of the training data, we constructed a single AH $\mathcal{T}$ from both labeled and unlabeled data and used it to train an AAMM for each class (from the labeled sequences).

Furthermore, in our third set of experiments, we investigated how the clustering algorithm affects the quality of the predictions made by an AAMM. In particular, we compared our AAMM clustering algorithm with agglomerative information bottleneck (AIB) introduced by Slonim and Tishby in [Slonim and Tishby, 1999]. The AIB was used to cluster words in a given text data set. However, it can be used to cluster k-grams. The primary difference between our AAMM clustering algorithm and AIB is in the similarity used to cluster k-grams together, i.e., in AAMM, the k-grams are clustered based on the similarity between the conditional distributions of X_i induced by k-grams, $p(X_i|k\text{-}gram)$, where X_i takes values in $\mathcal{X}$; in AIB, the k-grams are clustered based on the similarity between the class conditional distributions induced by k-grams, $p(Y|k\text{-}gram)$, where Y is the class variable that takes values in a set $\mathcal{Y}$ (the probabilities are estimated from the training data).

In this set of experiments, we constructed AHs from the sequences in the *training set* as follows: (i) a class-specific AH separately for each class (from sequences belonging to that class); (ii) a class-independent AH (from all training sequences); and (iii) an AH using the AIB (from all training sequences). In each case, because the class labels are available for all sequences, we trained an AAMM for each class (from sequences in that class).

We performed our experiments on the problem of predicting subcellular localization which is important in cell biology, because it can provide valuable information for predicting protein function and protein-protein interactions, among others. We show results of the experiments on three protein subcellular localization data sets: **psortNeg**, **plant**, and **non-plant**.

For all three data sets, we trained AAMMs for values of m that range from 1 to N, where m is the cardinality of the set of abstractions $\mathcal{A}_m$ used as "features" in the classification model, and N is the number of unique k-grams. We report the average classification accuracy obtained in a 5-fold cross-validation experiment as a function of m. Note that the x axis of all figures presented in the Results subsection shows the number of abstractions m on a logarithmic scale.



(a) psortNeg (b) plant (c) non-plant

Figure 3.3: Comparison of abstraction augmented Markov model (AAMM) with Markov model (MM) and Naïve Bayes (NB) on **psortNeg** (a), **plant** (b), and **non-plant** (c), respectively. The x axis shows the number of abstractions m, used as "features" in the classification model, on a logarithmic scale. In this experiment, an abstraction hierarchy $\mathcal{T}_j$ is constructed for each class $y_j \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible classes (class-specific abstraction hierarchies).

3.3.2 Data sets

The first data set used in our experiments, PSORTdb v.2.0[1] Gram-negative sequences, introduced in [Gardy *et al.*, 2003], 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 [Emanuelsson *et al.*, 2000]. 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).

[1]http://www.psort.org/dataset/datasetv2.html
[2]http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php

3.3.3 Results

We trained AAMMs and 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.

Comparison of AAMMs with MMs and NB

Figures 3.3(a), 3.3(b), and 3.3(c) show the results of the comparison of AAMMs with MMs on **psortNeg**, **plant**, and **non-plant** data sets, respectively. As can be seen in the figures, AAMM matches the performance of MM with substantially smaller number of abstractions. Specifically, the performance of MM trained using approximately 8000 3-grams is matched by that of AAMM trained using only 79, 19 and 855 abstractions on the **psortNeg**, **plant**, and **non-plant** data sets, respectively. On the **psortNeg** and **plant** data sets, AAMM outperforms MM over a broad range of choices of m. For example, with $m = 435$ on the **psortNeg** data set, AAMM achieves its highest accuracy of 76.87% as compared to 74.93% achieved by MM with $N = 7970$ unique k-grams. On the **plant** data set with only $m = 168$, AAMM achieves its highest accuracy of 71.59% as compared to MM which achieves an accuracy of 68.19% with $N = 7965$. Not surprisingly, with $m = N$ abstractions, the performance of AAMMs is the same as that of MMs, because the AAMM trained on N abstractions and MM are exactly the same models.

We conclude that AAMMs can match and, in some cases, exceed the performance of MMs using substantially (by one to three orders of magnitude) more compact classifiers. AAMMs could provide more robust estimates of model parameters than MMs, and hence, help avoid *overfitting*.

Figures 3.3(a), 3.3(b), and 3.3(c) also show the comparison of AAMMs with Naïve Bayes (NB) models trained using a "bag of letters" feature representation. As can be seen in the figures, except for a few values of m ($m < 18$, $m < 5$, and $m < 2$ on **psortNeg**, **plant**, and **non-plant**, respectively), AAMM outperforms NB (for any other choices of m). For example,

[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.

on the **plant** data set, with 20 features, NB achieves 59.36% accuracy as compared to 68.29% obtained using AAMM (the accuracy of MM is 68.19%). On the **non-plant** data set, with 20 features, NB achieves 67.23% accuracy, whereas AAMM and MM achieve 72.31% and 74.98% (respectively). Hence, both AAMM and MM (of order 3) are superior in performance to NB models (0^{th} order MM) for a large number of choices of m (for AAMM).

Evaluation of Abstraction in a Semi-supervised Setting

Figure 3.4 shows the results of the comparison of AAMMs with MMs using 3-grams for 1% (first row), 10% (second row), 25% (third row), and 100% (fourth row) of labeled data for all three data sets: **psortNeg** (left column), **plant** (center column), and **non-plant** (right column), respectively. Note that an MM is trained on the same fraction of labeled data as its AAMM counterpart. Also note that, unlike our previous experiment where all the data are labeled and, hence, a class-specific AH can be constructed for each class, in this experiment, a single class-independent AH can be constructed from all sequences in the training set because many of the training instances are unlabeled. As can be seen in the figure, in the cases where only 1% and 10% of the data are labeled, AAMMs substantially outperform MMs over a broad range of choices of m, on all of the three data sets. For example, on the 1% **plant** data set, with $m = 200$, the accuracy of AAMM is 38.72%, whereas that of MM is 30.53%. On the 10% **plant** data set, with $m = 560$, AAMM achieves its highest accuracy of 47.97%, as compared to that of MM which is 37.87%.

When we increased the number of labeled data to 25%, AAMMs still have a much higher performance than MMs for many choices of m on the **plant** and **non-plant** data sets, but become comparable in performance to MMs on the **psortNeg** data set. With $m = 2310$, on the 25% **plant** data set, AAMM achieves its highest accuracy of 56.80% as compared to 53.40% achieved by MM. On the other hand, on the 25% **psortNeg** data set, the highest accuracy of AAMM obtained with $m = 4035$ is 60.38% that is similar to the accuracy of MM which is 59.34%.

In the case where the entire data set is labeled (100%), the performance of AAMMs is

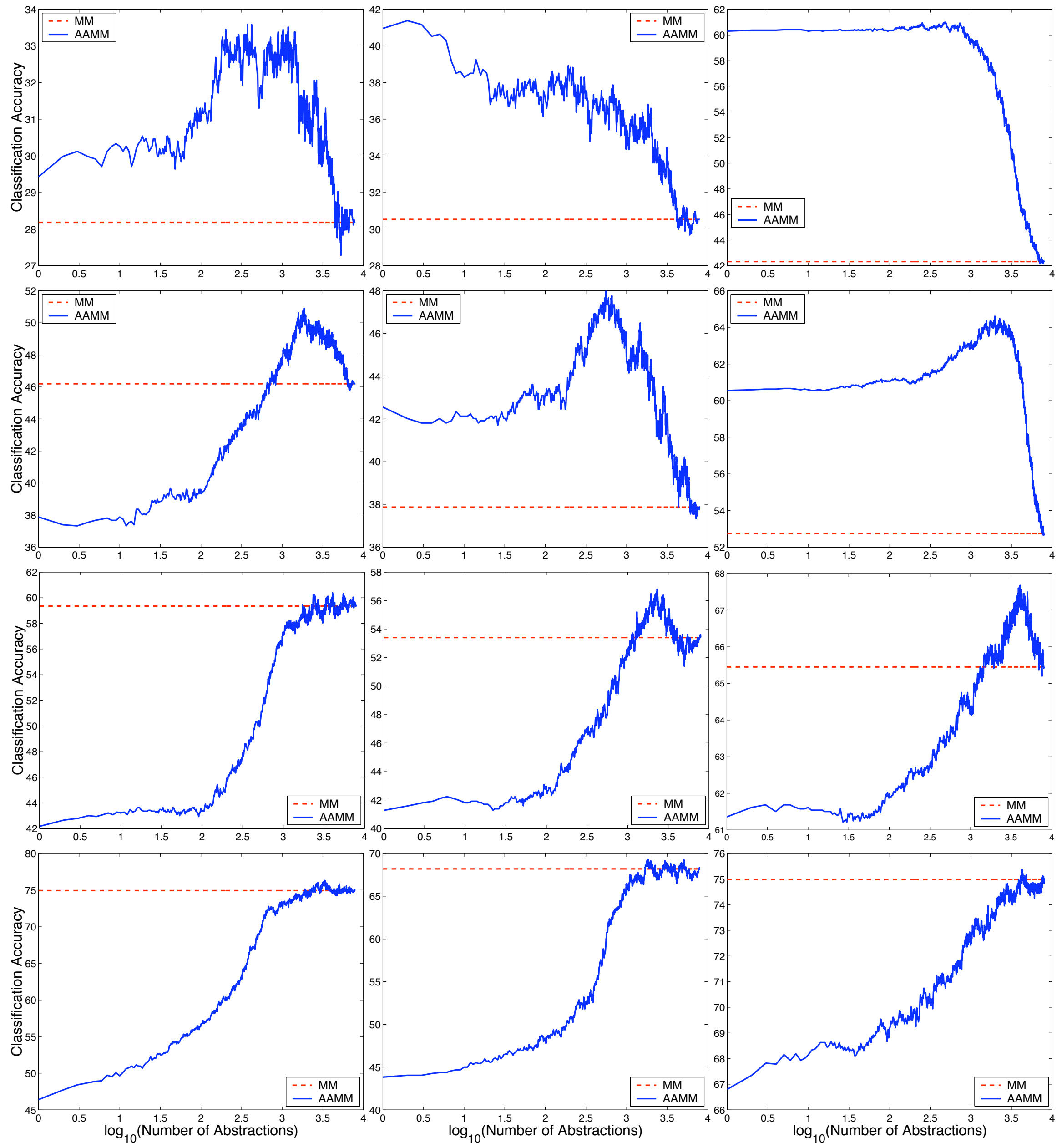
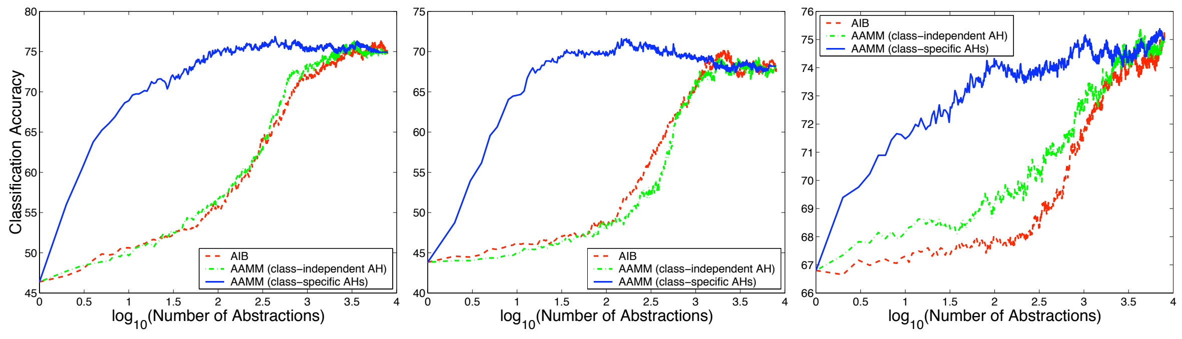


Figure 3.4: Comparison of AAMMs with MMs for 1% (first row), 10% (second row), 25% (third row), and 100% (fourth row) of labeled data for **psortNeg** (left column), **plant** (center column), and **non-plant** (right column), respectively. The x axis shows the number of abstractions m, used as "features" in the classification model, on a logarithmic scale. In this experiment, an abstraction hierarchy $\mathcal{T}$ is constructed from all training data, independent of the class variable (class-independent abstraction hierarchy).



(a) psortNeg (b) plant (c) non-plant

Figure 3.5: Comparison of the abstraction augmented Markov model clustering algorithm with the Agglomerative Information Bottleneck on **psortNeg** (a), **plant** (b), and **non-plant** (c), respectively.

comparable to that of MMs for choices of $m \geq 1255$, $m \geq 1095$, and $m \geq 1590$ on **psortNeg**, **plant**, and **non-plant** data sets, respectively. As an example, on the 100% **plant** data set, with $m = 1095$, the accuracy of AAMM is 67.34%, and that of MM is 68.19%.

We conclude that AAMMs can provide more accurate models compared to MMs in settings where there are relatively small amounts of labeled data, but rather plentiful unlabeled data that would be very expensive to be labeled. AAMMs incorporates information available in the unlabeled data, that would otherwise be ignored by MMs.

Comparison of AAMM clustering algorithm with Agglomerative Information Bottleneck

Figure 3.5 shows, on all three data sets, the results of the comparison of AAMMs based on (i) class-specific AHs, with one AH for each class, (ii) a single class-independent AH, and (iii) an AH produced using AIB [Slonim and Tishby, 1999]. As can be seen in the figure, the AAMMs based on the clustering algorithm proposed in this study substantially outperform the AAMMs based on the AIB clustering algorithm over a broad range of values of m. For example, on the **plant** data set, with $m = 100$, the accuracy of AAMM based on our clustering approach is 69.57%, whereas that of AIB-clustering based AAMM is 48.29%. As expected, AAMMs trained using class-specific abstraction hierarchies outperform AAMMs trained using a single

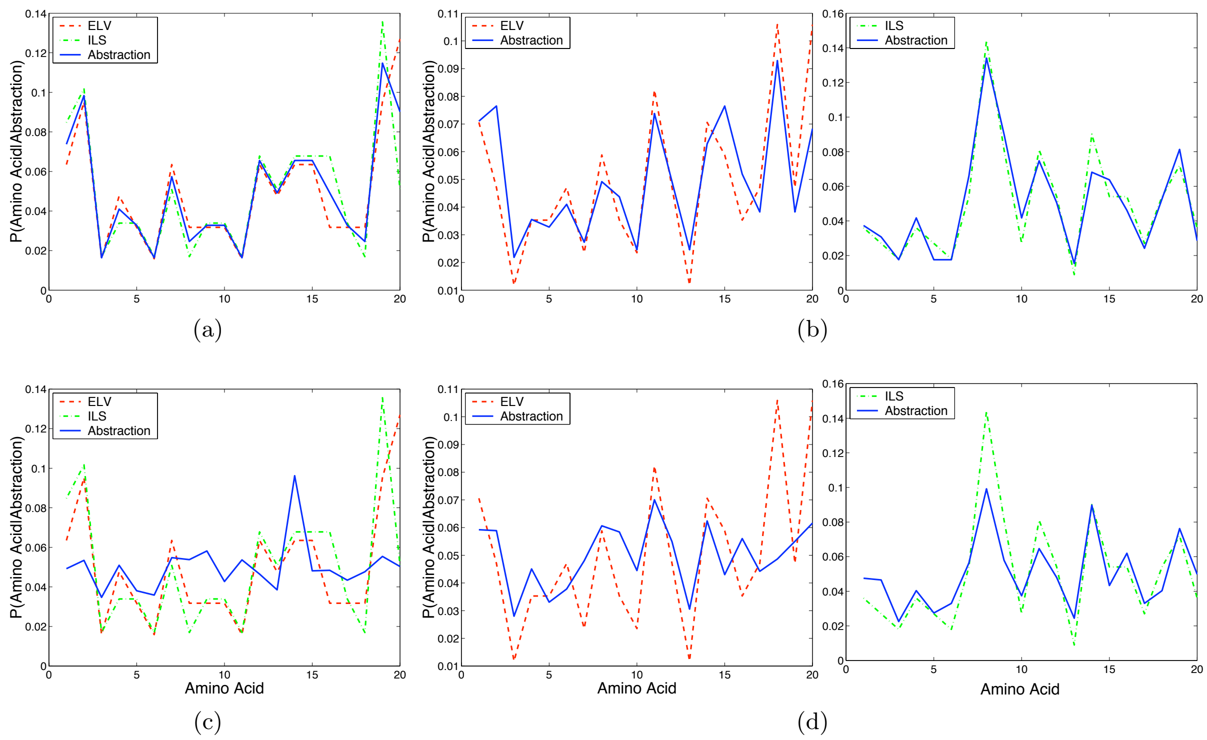


Figure 3.6: Amino acid distributions (on the **non-plant** data set) induced (a) by an abstraction, a_{681}, randomly chosen from the cut γ_{1000} through the AH $\mathcal{T}_0$ (corresponding to class 0), and by two 3-grams, "ELV" and "ILS", randomly chosen from a_{681}; (b) by "ELV" and "ILS" and by their abstractions, a_{652} and a_{76}, respectively, on the cut γ_{1000} through the AH $\mathcal{T}_1$ (corresponding to class 1); (c) by "ELV" and "ILS" and by their abstraction, a_2 on the cut γ_{10} through the AH $\mathcal{T}_0$; (d) by "ELV" and "ILS" and by their abstractions, a_5 and a_1, respectively, on the cut γ_{10} through the AH $\mathcal{T}_1$.

class-independent AH.

We conclude that organizing the k-grams in an AH based on the conditional distributions of the next letter in the sequence rather than based on the class conditional distributions induced by the k-grams produces more suitable AHs for AAMMs, and hence, better performing AAMMs.

3.4 Analysis of Abstractions

To gain some insight into the data organization generated by our clustering procedure, we took a closer look at the k-grams that are grouped into a particular abstraction. Figure 3.6(a) shows the amino acid distributions (on the **non-plant** data set) induced by an abstraction, a_{681}, randomly chosen from the cut γ_{1000} through the AH $\mathcal{T}_0$ (corresponding to class 0), and by two 3-grams, "ELV" and "ILS", randomly chosen from a_{681}. Figure 3.6(b) shows the amino acid distributions induced by "ELV" and "ILS" and by their corresponding abstractions, a_{652} and a_{76}, respectively, on the cut γ_{1000} through the AH $\mathcal{T}_1$ (corresponding to class 1). Not surprisingly, as can be seen in the figures, not only that "ELV" and "ILS" induce different amino acid distributions as compared to those that they induce in class 0, but also the amino acid distribution induced by "ELV" is different from that induced by "ILS" in class 1, and hence, they belong to different abstractions. Note that the amino acid distribution induced by an abstraction is very similar to those induced by individual (constituent) 3-grams (the average number of 3-grams per abstraction is 8). Hence, the individual 3-grams can be replaced by their abstraction without significant change in the accuracy of the resulting (AAMMs) classifiers. To avoid cluttered figures, we drew different plots for class 1.

Figures 3.6(c) and 3.6(d) show the amino acid distributions induced by "ELV" and "ILS" and by their abstractions on the cut γ_{10} through the AHs $\mathcal{T}_0$ and $\mathcal{T}_1$, respectively (the average number of 3-grams per abstraction is 799). As expected, the distribution induced by an abstraction become more dissimilar to those induced by individual 3-grams as we go from a finer to a coarser set of abstractions (e.g., 1000 to 10). We conclude that AAMM provides compact models at the expense of losing some information through abstraction. On the other hand, AAMM could potentially provide better estimates of model parameters as compared to MM (when the parameters are underestimated in the data). Consequently, AAMM can help avoid *overfitting*.

3.5 Summary and Discussion

We have presented abstraction augmented Markov models that simplify the data representation used by the standard Markov models. The results of our experiments on three protein subcellular localization data sets have shown that organizing the set of k-grams in a hierarchy using *abstraction* makes it possible to construct predictive models that have significantly smaller size (by 1 to 3 orders of magnitude) as compared to the size of the corresponding MM (which is exponential in k). While the abstraction reduces the model size, the performance of AAMM is similar, and in some cases better, than that of standard Markov models. Moreover, the results have shown that *abstraction* allows us to make better use of unlabeled training data.

3.5.1 Related Work

Segal et al. [Segal *et al.*, 2002] and McCallum et al. [McCallum *et al.*, 1998] have used abstraction hierarchies over *classes* to improve classification accuracy. Punera et al. [Punera *et al.*, 2006] have designed an algorithm to organize the classes in a given data set in an n-ary taxonomy. They have shown that n-ary tree based taxonomies provide a more "natural" way to organize classes in a hierarchy than the binary tree based taxonomies. Zhang et al have used abstraction hierarchies over *nominal variables* to build compact, yet accurate decision tree (AVT-DTL) [Zhang and Honavar, 2003] and Naive Bayes (AVT-NBL) [Zhang *et al.*, 2006] classifiers. Unlike AVT-DTL and AVT-NBL, AAMMs exploit abstraction hierarchies that compress entities (k-grams) based on conditional distributions of each element X_i in the sequence given its *parents*, rather than class conditional distributions. In related work, desJardins et al. [desJardins *et al.*, 2000] have explored the use of abstraction-based search (ABS) to compress conditional probability tables (CPTs) in Bayesian networks. Unlike ABS, AAMMs exploit abstraction to construct compact Markov models of sequence data.

3.5.2 Connection between AAMMs and HMMs

An AAMM can be simulated by an appropriately constructed Hidden Markov Model (HMM) [Rabiner, 1989], [Durbin *et al.*, 2004] where the number of hidden states is equal to the number of abstractions (i.e., clusters) in the AAMM. However, as a state corresponds to a cluster (i.e., a set of k-grams) over the observable variables, the state is not really "hidden". It can be derived in a feed-forward manner, thus not requiring a Backward Reasoning/Expectation step. This allows a "one pass through the data" learning procedure based on MAP learning

[Mitchell, 1997] once the set of clusters has been chosen. Unlike the learning procedure for HMMs (which involves the Expectation-Maximization (EM) algorithm), the AAMM learning procedure is not prone to local maxima, has lower variance as no uncertainty is inherited from the inference performed in the E step, and requires less time given that the set of clusters is already available (as shown below).

The overall time complexity of AAMM is $O(N^2 \cdot |\mathcal{X}| + N \cdot |\mathcal{D}| \cdot n)$, where N is the number of unique k-grams, $|\mathcal{X}|$ is the size of the alphabet, $|\mathcal{D}|$ is the size of the data set, and n is the maximum length of sequences in $|\mathcal{D}|$. The time complexity of the HMM EM training procedure for a fixed number of hidden states is $O(T \cdot |\mathcal{D}| \cdot n \cdot (|H|^2 + |H| \cdot |\mathcal{X}|))$, where $|H|$ is the number of hidden states and T is the number of EM iterations. Running this procedure for all possible number of hidden states (from 1 to N) requires an overall time of $O(N^2(N + |\mathcal{X}|) \cdot T \cdot |\mathcal{D}| \cdot n)$. By observing that $N <= |\mathcal{D}| \cdot n$, based on the above analysis, our algorithm requires at least a factor of $N \cdot T$ less time compared to the HMM training procedure. While algorithms that attempt to automatically determine the number of hidden states in HMMs (e.g., based on the Chinese Restaurant Process [Blei *et al.*, 2004]) can be used, not only that they come at additional costs vs. "vanilla" HMM, but they are also prone to the difficulties encountered by hidden variable models. Hence, although AAMMs are less expressive models than HMMs, they are easier to learn models compared to HMMs .

3.5.3 The relationship between AAMM clustering algorithm and Agglomerative Information Bottleneck

Slonim and Tishby proposed Agglomerative Information Bottleneck (AIB) [Slonim and Tishby, 1999] as a method to organize data in an AH based on the similarity of the class conditional distributions. Specifically, they have applied their approach to cluster words in a given text data set. However, it can also be used to cluster k-grams. Similar to the AAMM clustering algorithm, their algorithm starts with one word (or k-gram) per cluster and recursively merges pairs of sets of words (or k-grams), in a greedy fashion, to generate an AH. A merge that minimizes the loss in the mutual information between the intermediate clustering and the class variable is performed at each step of their algorithm. Unlike AIB, we cluster k-grams based on the similarity between the conditional distributions of each element X_i in the sequence given its *parents*, rather than the similarity between the class conditional distributions (as in AIB). That is, the AIB method captures the information that one variable (e.g., a word) contains about another variable (e.g., the topic of a document), while our method captures the information that a k-gram contains about the next element in a sequence, independent of the class.

3.5.4 AAMMs provide simple and accurate models

The performance of AAMMs is not always superior to that of MMs. However, AAMMs provide simpler models. The simplicity is achieved by losing some information through abstraction. The challenge is to trade off the complexity of the model against its predictive power. Not only that our model is simpler but it is also more interpretable from a biological point of view, i.e. the set of k-grams that form an abstraction can be seen as a sequence profile (e.g., Position Specific Scoring Matrix). Also, designing accurate models from relatively small amounts of labeled data is important in many applications that involve sequence data (e.g., biological applications) where unlabeled data is rather plentiful and obtaining labels for unlabeled data is very expensive.

While the abstraction hierarchies can be learned using any top-down or bottom-up clustering procedure, in this study we have used a bottom-up approach because it is simple, fast, and allows iterating through all cardinalities from N to 1.

3.5.5 Future directions

Some possible directions for further research include extensions of the model developed here for sequence classification to settings where the data instances have a much richer structure (e.g., multi-modal data consisting of images and text, and combination of audio and visual data).

3.6 Appendix

Lemma 1:

Let X and Z be two random variables such that Z can take on k possible values. Let $p\left(z_i\right)$ be the prior distribution of z_i and $p\left(x|z_i\right)$ be the conditional distribution of X given z_i for $i = 1, \cdots, k$. Then:

$$\begin{aligned} & JS_{p(z_1),\cdots,p(z_k)}\left(p(x|z_1),\cdots,p(x|z_k)\right) \\ &= H\left(\sum_{i=1}^{k} p(z_i)p(x|z_i)\right) - \sum_{i=1}^{k} p(z_i)H\left(p(x|z_i)\right) = I(X;Z) \end{aligned}$$

where $H(\cdot)$ is Shannon's entropy [Lin, 1991].

Proof of Proposition 1:

Without loss of generality, let us assume that the merge is $\{a_1, a_2\} \to a$.

Let $\delta I(\{a_1,a_2\},a) = I(A^{(\mathcal{A}_m)},Y) - I(A^{(\mathcal{A}_{m-1})},Y)$ denote the reduction in the mutual information $I(A;Y)$, where $A^{(\mathcal{A}_m)}$ represents the variable A that takes values in the set $\mathcal{A}_m = \{a_1, \cdots, a_m\}$. We use the above lemma. Hence,

$\delta I(\{a_1,a_2\},a)$

$$\begin{aligned} &= JS_{p(a_1),p(a_2),\cdots,p(a_m)}\left[p(y|a_1),p(y|a_2),\cdots,p(y|a_m)\right] - JS_{p(a),\cdots,p(a_m)}\left[p(y|a),\cdots,p(y|a_m)\right] \\ &= H\left(\sum_{i=1}^{m} p(a_i)p(y|a_i)\right) - \sum_{i=1}^{m} p(a_i)H\left(p(y|a_i)\right) - H\left(p(a)p(y|a) + \sum_{i=3}^{m} p(a_i)p(y|a_i)\right) \\ &\quad + p(a)H\left(p(y|a)\right) + \sum_{i=3}^{m} p(a_i)H\left(p(y|a_i)\right) \\ &= H\left(\sum_{i=1}^{m} p(a_i)p(y|a_i)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) - H\left(p(a)\frac{1}{p(a)}\sum_{i=1}^{2} p(y,a_i) + \sum_{i=3}^{m} p(a_i)p(y|a_i)\right) \\ &\quad + p(a)H\left(p(y|a)\right) \\ &= p(a)H\left(p(y|a)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) \\ &= p(a)H\left(\frac{1}{p(a)}\sum_{i=1}^{2} p(y,a_i)\right) - \sum_{i=1}^{2} p(a_i)H\left(p(y|a_i)\right) \\ &= p(a)\left(H\left(\sum_{i=1}^{2}\frac{p(a_i)}{p(a)}p(y|a_i)\right) - \sum_{i=1}^{2}\frac{p(a_i)}{p(a)}H\left(p(y|a_i)\right)\right) \end{aligned}$$

$$= \quad (p(a_1)+p(a_2)) \cdot JS_{\pi_1,\pi_2}(p(Y|a_1), p(Y|a_2)).$$

Proof of Proposition 2:

As in **Proposition 1**, let us assume, without loss of generality, that the merge is $\{a_1, a_2\} \rightarrow a$. Hence, $\#a = \#a_1 + \#a_2$. Furthermore, let $\pi_1 = \frac{p(a_1)}{p(a_1)+p(a_2)}$ and $\pi_2 = \frac{p(a_2)}{p(a_1)+p(a_2)}$.

$\delta LL(\{a_1, a_2\}, a)$

$$\begin{aligned}
&= LL(A^{(\mathcal{A}_m)}, X_i) - LL(A^{(\mathcal{A}_{m-1})}, X_i) \\
&= \sum_{x_i \in \mathcal{X}, a_j \in \mathcal{A}_m} \log p(x_i|a_j)^{\#[a_j,x_i]} - \sum_{x_i \in \mathcal{X}, a_j \in \mathcal{A}_{m-1}} \log p(x_i|a_j)^{\#[a_j,x_i]} \\
&= \sum_{x_i \in \mathcal{X}} \log p(x_i|a_1)^{\#[a_1,x_i]} + \sum_{x_i \in \mathcal{X}} \log p(x_i|a_2)^{\#[a_2,x_i]} - \sum_{x_i \in \mathcal{X}} \log p(x_i|a)^{\#[a,x_i]} \\
&= \sum_{x_i \in \mathcal{X}} \#[a_1, x_i] \cdot \log p(x_i|a_1) + \sum_{x_i \in \mathcal{X}} \#[a_2, x_i] \cdot \log p(x_i|a_2) \\
&\quad - \sum_{x_i \in \mathcal{X}} (\#[a_1, x_i] + \#[a_2, x_i]) \cdot \log p(x_i|a_1 \cup a_2) \\
&= -Mp(a) \sum_{x_i \in \mathcal{X}} \left(\sum_{j=1}^{2} \pi_j p(x_i|a_j) \right) \log \left(\sum_{j=1}^{2} \pi_j p(x_i|a_j) \right) \\
&\quad + M \cdot p(a) \left(\sum_{x_i \in \mathcal{X}} \sum_{j=1}^{2} \pi_j p(x_i|a_j) \log p(x_i|a_j) \right) \\
&= Mp(a) \left(H\left(\sum_{j=1}^{2} \pi_j p(x_i|a_j) \right) - \sum_{j=1}^{2} \pi_j H\left(p(x_i|a_j)\right) \right) \\
&= M \cdot ((p(a_1)+p(a_2)) \cdot JS_{\pi_1,\pi_2}(p(X_i|a_1), p(X_i|a_2))
\end{aligned}$$

where M is the cardinality of the *multiset* of $(k+1)$-grams. We have used that:

$$p(x_i|a_1 \cup a_2) = \pi_1 p(x_i|a_1) + \pi_2 p(x_i|a_2), \text{when } a_1 \cap a_2 = \phi.$$

and

$$\#[a_j, x_i] = p(a_j, x_i) \cdot M = p(x_i|a_j) \cdot p(a_j) \cdot M = p(x_i|a_j) \cdot \pi_j \cdot p(a) \cdot M$$

CHAPTER 4. LEARNING LINK-BASED CLASSIFIERS FROM ONTOLOGY-EXTENDED TEXTUAL DATA AND ITS EXTENSION TO DISTRIBUTED DATA

Real-world data mining applications call for effective strategies for learning predictive models from richly structured relational data. In this paper, we address the problem of learning classifiers from structured relational data that are annotated with relevant meta data. Specifically, we show how to learn classifiers at different levels of abstraction in a relational setting, where the structured relational data are organized in an abstraction hierarchy that describes the semantics of the content of the data. We show how to cope with some of the challenges presented by *partial specification* in the case of structured data, that unavoidably results from choosing a particular level of abstraction. Our solution to partial specification is based on a statistical method, called *shrinkage*. Furthermore, we extend the problem of learning link-based classifiers from ontology-extended textual data to the semantically heterogeneous, distributed, relational setting. We show under fairly general assumptions, how to exploit data sources annotated with relevant meta data in building predictive models (e.g., classifiers) from a collection of distributed relational data sources, without the need for a centralized data warehouse, while offering strong guarantees of *exactness* of the learned classifiers relative to their centralized relational learning counterparts. We present results of experiments in the case of learning link-based Naïve Bayes classifiers on a text classification task that (i) demonstrate that the choice of the level of abstraction can impact the performance of the resulting link-based classifiers and (ii) examine the effect of partially specified data.

4.1 Introduction

Advances in sensors, digital storage, computing and communications technologies have led to an exponential increase in the amount of on-line richly structured, relational data. Problems such as classifying web pages, filtering e-mail, annotating images, etc., have become very important. Machine learning algorithms [Mitchell, 1997], [Bishop, 2006] offer some of the most cost effective approaches to building predictive models (e.g., classifiers) in a broad range of applications in data mining.

Although the problem of learning classifiers from relational data sources has received much attention in the machine learning literature [Getoor *et al.*, 2002], [Neville *et al.*, 2003], [Taskar *et al.*, 2002], there has been limited exploration of techniques that use structured relational data sources that are annotated with relevant meta data. In this paper, we address the problem of learning classifiers from such data.

Representational commitments, i.e., the choice of features or attributes that are used to describe the data presented to a learner, and the level of detail at which they describe the data, can have a major impact on the difficulty of learning, and the accuracy, complexity, and comprehensibility of the learned predictive model [Valiant, 1984]. The representation has to be rich enough to capture distinctions that are relevant from the standpoint of learning, but not so rich as to make the task of learning infeasible due to overfitting.

Hence, we present an approach to learning classifiers at different *levels of abstraction* (or detail) when the data (attributes and classes) are organized in abstraction hierarchies. We adapt the link-based iterative classification algorithm introduced by Lu and Getoor [Lu and Getoor, 2003] and use it in conjunction with Naïve Bayes classifiers to illustrate our approach.

Next, we show how to cope with *partially specified data* that inevitably result from choosing a particular level of abstraction. Zhang et al. [Zhang and Honavar, 2003], [Zhang *et al.*, 2006] have previously addressed the problem of partially specified data in cases where data instances are described by nominal attributes. In this study, we deal with partial specification in the case of structured data, where, for example, a multinomial model is assumed as the underlying model that generated the data instances. We use a statistical approach, called *shrinkage*, to cope with partially specified data.

Furthermore, given the autonomous nature of the on-line data sources from the same domain, we show how to extend the approach to learning classifiers from a single data source to learning classifiers from a collection of autonomous data sources that are annotated with relevant meta data. Specifically, we present a principled approach to the problem of learning classifiers from a collection of semantically disparate *relational* data sources that do not rely on direct access to the data but instead can work with results of statistical queries against an integrated view.

We evaluated our approaches to learning classifiers from both structured relational data and structured relational, *distributed* data annotated with relevant meta data on the *Cora* data set, a standard relational benchmark data set of research articles and their citations [McCallum *et al.*, 2000] for which we manually constructed an abstraction hierarchy from the words in the data set. The task was to classify research articles based on their topics. The results of our experiments show that: (i) more abstract levels can yield better performing link-based classifiers, due to more robust estimates of model parameters (smaller number of parameters that need to be estimated from data); (ii) making use of partially specified data can improve classification performance; (iii) under fairly general assumptions, classifiers obtained in the distributed setting are identical to those obtained from a centralized, integrated data warehouse constructed from the collection of semantically disparate relational data sources and associated ontologies and mappings.

The rest of the paper is organized as follows: In Section 2, we introduce the framework necessary for learning classifiers from structured relational data annotated with relevant meta data. We present our approach to learning classifiers from such data in the case of link-based Naïve Bayes classifiers [Lu and Getoor, 2003] in Section 3. In Section 4, we discuss some of the challenges presented by partially specified data in our setting. Section 5 provides experimental results on a relational data set from the text categorization applications. In Section 6, we extend our approach to learning classifiers from structured relational data that are annotated with relevant meta data to a distributed setting. Section 7 concludes with summary, a brief

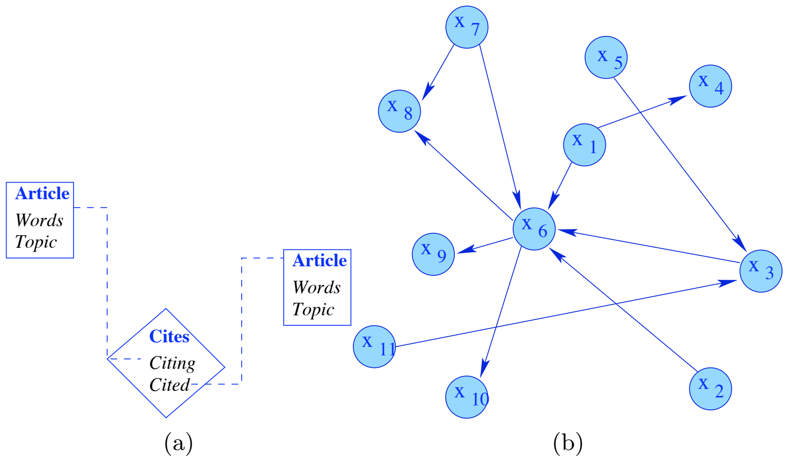


Figure 4.1: (a) A simple schema corresponding to a bibliographic domain (the figure is adapted from Getoor et al. (b) A sample instantiation graph corresponding to the schema in (a). The nodes x_i represent `Article` instances, i.e., $x_i \in$ `Article`, while the edges (x_i, x_j) represent `Cites` instances, i.e., $(x_i, x_j) \in$ `Cites`.

discussion of related work, and an outline of some directions for further research.

4.2 Ontology-Extended Structured Relational Data

A *schema* $\mathcal{S}$ of a structured relational data source describes a set of *concepts* $\mathcal{X} = \{X_1, \cdots, X_t\}$, and the *relations* between them, $\mathcal{R}(X_i, X_j)$. An instance of a concept X_i is a structured object, e.g. a string, a document, or an image. These instances can be described by *features* such as a set of attributes, a histogram of words, or features from spatially contiguous elements of an image. An attribute A of a concept X_i, denoted by $X_i.A$ takes values in a set $\mathcal{V}(X_i.A)$. A relation $R(X_i, X_j)$ corresponds to a set defined as $x_i.R = \{x_j \in X_j$ s.t. x_j is related to x_i through $R\}$, where x_i denotes an instance of the concept X_i. A tuple (x_i, x_j) is an instance of the relation R. A *data set* $\mathcal{D}$ that specifies a set of instances and the relations between them is an *instantiation* $\mathcal{I}(\mathcal{S})$ of a schema $\mathcal{S}$. It can be represented as an *instantiation graph* in which nodes denote instances and edges denote relations between instances [Getoor *et al.*, 2002].

Figure bibliographic domain which consists of the `Article` concept and the `Cites` relation. Figure 4.1b shows a sample instantiation graph corresponding to the relational schema in Figure 4.1a. The nodes x_i in the graph represent research articles (i.e., `Article` instances,

$x_i \in \texttt{Article}$) and the edges (x_i, x_j) represent the "citation" relation between articles (i.e., $(x_i, x_j) \in \texttt{Cites}$). Note that each instance can have a variable number of related instances and thus, a variable number of *features* [Neville *et al.*, 2003].

Assuming that we model a research article using the histogram of word occurrences, the concept `Article` is described by two attributes: `Article.`*Words* that denotes the word histogram of the article, and `Article.`*Topic*, a categorical attribute that denotes the topic of the article.

An ontology $\mathcal{O}$ associated with a structured relational data source $\mathcal{D}$ is given by a *content ontology* that describes the semantics of the content of the data (e.g., the values and relations between values that `Article.`*Words* can take in $\mathcal{D}$)[1]. Of particular interest are ontologies that specify *hierarchical* relations among values of attributes. *Isa* relations induce *abstraction hierarchies* (AHs) over the values of attributes.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ associated with an attribute $X_i.A$ is a rooted tree such that: (1) The tree $\mathcal{T}$ has exactly $n = |\mathcal{V}(X_i.A)|$ nodes such that the leaves correspond to the most specific values of $X_i.A$, and the internal nodes correspond to abstractions over the most specific values of $X_i.A$ (i.e., more abstract values of $X_i.A$); in particular, the root of $\mathcal{T}$ corresponds to the most abstract value of $X_i.A$; and (2) The edges of $\mathcal{T}$ represent partial order relations $\prec$ (e.g., isa relations) between their corresponding nodes.*

Definition 2 (m-Cut) *An m-cut (or level of abstraction) γ_m through the abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ satisfying the following properties: (1) For any leaf a_i, either $a_i \in \gamma_m$ or a_i is a descendant of a node $a_j \in \gamma_m$; and (2) For any two nodes $a_k, a_l \in \gamma_m$, a_k is neither a descendant nor an ancestor of a_l. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of the set of leaves.*

Figures 4.2a and 4.2b show two fragments of AHs over the values of the attributes `Article.`*Topic* and `Article.`*Words*, respectively, corresponding to the bibliographic domain. The set $\mathcal{V}$(`Article.`*Topic*) consists of {Artificial Intelligence (AI), Data Mining (DM), Machine Learning (ML), Natural Language Processing (NLP), Neural Networks (NN), Genetic Algorithms

[1]In a more general setting, the ontology $\mathcal{O}$ contains also a *structure ontology* that describes the semantics of the elements of a schema S (concepts and their attributes), in addition to the *content ontology*.

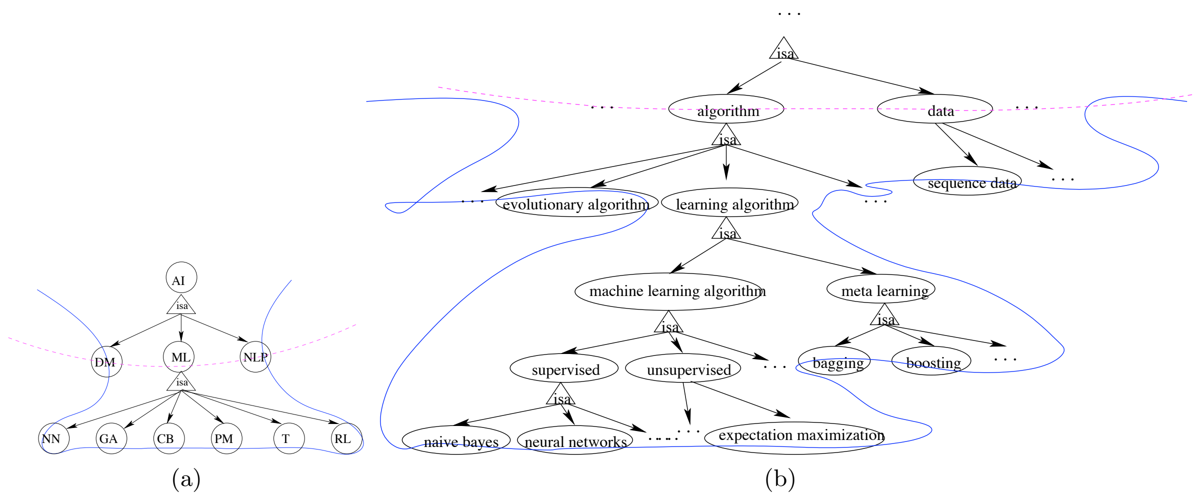


Figure 4.2: Two fragments of abstraction hierarchies (AHs) over the values of attributes `Article.`*Topic* and `Article.`*Words* corresponding to a bibliographic domain are shown in (a) and (b), respectively. The dash curves represent different cuts. The set {*DM, ML, NLP*} represents a cut or *level of abstraction* γ_3 in (a). {*DM, NN, GA, CB, PM, T, RL, NLP*} is a finer cut γ_8.

(GA), Case-Based (CB), Probabilistic Methods (PM), Theory (T), Reinforcement Learning $(RL)\}$. The subset of nodes $\{DM, ML, NLP\}$ represents a 3-cut γ_3 through the AH in 4.2a. The subset $\{DM, NN, GA, CB, PM, T, RL, NLP\}$ is a finer cut γ_8.

An ontology $\mathcal{O}$ associated with a structured relational data source $\mathcal{D}$ consists of a set of AHs $\{\mathcal{T}_1, \cdots, \mathcal{T}_l\}$, corresponding to the set of attributes, w.r.t. the *isa* relation. A global cut Γ through $\mathcal{O}$ consists of a set of cuts, one for each constituent AH, e.g., $\Gamma = \{\gamma_{Words}, \gamma_{Topic}\}$.

The *isa* relations between the nodes in an AH $\mathcal{T}_i$ associated with an attribute $X_i.A$ specify semantic relationships between the values of the attribute (more specific and more abstract values). Examples of such semantic relationships are equality, $x = y$, meaning that x and y are *equivalent* (or *synonyms*), and inclusion $x < y$, meaning that y *subsumes* x, or y is *more general* than x [Rajan *et al.*, 2005]. A subset of inclusion relationships between the values of `Article.`*Topic* is $\{NN < ML,\ AI > ML\}$.

Definition 3 (Ontology-Extended Structured Relational Data Source) *An ontology-extended structured relational data source (OESRDS) is defined as a tuple* $\{\mathcal{S}, \mathcal{D}, \mathcal{O}\}$*, where* $\mathcal{S}$ *represents the structured relational data source schema,* $\mathcal{D}$ *is an instantiation of* $\mathcal{S}$*, and* $\mathcal{O}$

represents the data source ontology [Caragea et al.*, 2007c].*

4.3 Learning Link-Based Classifiers from OESRDSs

We now proceed to describe an algorithm for learning classifiers from OESRDSs. We adapt the link-based iterative classification algorithm introduced by Lu and Getoor [Lu and Getoor, 2003] to the problem of learning classifiers from OESRDSs. We apply the resulting classifiers on the bibliographic domain where the task is to classify research articles based on their topics.

4.3.1 Link-based iterative classification

To label an instance, the iterative classification algorithm learns and exploits the distribution of links in the instantiation graph, in addition to the information available in the instance itself. This approach to learning from relational data is potentially more powerful than methods (e.g., influence propagation over relations [Taskar *et al.*, 2001]) that assume that related instances have similar labels.

An `Article` instance x_i is represented using the attribute `Article.`*Words*, denoted by $OA(x_i)$, and the link distribution of x_i, denoted by $LD(x_i)$ (to exploit the link patterns in classifying x_i). The object attribute $OA(x_i)$ holds x_i's word frequency counts. The link distribution $LD(x_i)$ holds the topic frequency counts computed from the set of objects that are linked to x_i in four different ways as follows:

- $InLink(x_i) = \{x_j \text{ s.t. } (x_j, x_i) \in \texttt{Cites}\}$,
- $OutLink(x_i) = \{x_j \text{ s.t. } (x_i, x_j) \in \texttt{Cites}\}$,
- $CoInLink(x_i) = \{x_j \text{ s.t. } x_j \neq x_i \text{ and } \exists x_k : (x_k, x_i) \in \texttt{Cites} \text{ and } (x_k, x_j) \in \texttt{Cites}\}$,
- $CoOutLink(x_i) = \{x_j \text{ s.t. } x_j \neq x_i \text{ and } \exists x_k : (x_i, x_k) \in \texttt{Cites} \text{ and } (x_j, x_k) \in \texttt{Cites}\}$.

The iterative classification algorithm consists of two steps, bootstrap and iteration [Lu and Getoor, 2003]:

Step 1: Using only the object attributes $OA(x_i)$, assign an initial topic to each article x_i in the test set.

Step 2: Using the object attributes $OA(x_i)$ and the link description $LD(x_i)$, iteratively assign a topic to each article x_i in the test set, until a termination criterion is met (e.g., either there are no changes to the topic assignments of articles or a certain number of iterations is reached). That is, for each article x_i:

1. Encode x_i using $OA(x_i)$ and $LD(x_i)$, based on the current assignments of linked articles;
2. Compute:

$$\hat{c}(x_i) = \arg\max_{c_j \in \mathbf{C}} P(c_j|OA(x_i)) \prod_{l \in \{In,Out,CI,CO\}} P(c_j|LD_l(x_i))$$

We use a Multinomial Naïve Bayes classifier [McCallum and Nigam, 1998]. The Naïve Bayes classifier makes the assumption that the attributes of each instance are conditionally independent given the class. Using Bayes rule and the independence assumption, the above probabilities can be replaced by:

$$P(c_j|OA(x_i)) = P(c_j) \prod_{v_i \in \mathcal{V}(OA(x_i))} P(v_i|c_j)$$

$$P(c_j|LD_l(x_i)) = P(c_j) \prod_{u_i \in \mathcal{V}(LD_l(x_i))} P(u_i|c_j)$$

Although this assumption may be violated in practice, empirical results show that Naïve Bayes classifier is competitive with state-of-the-art methods (including those that are computationally far more expensive than Naïve Bayes) on the document classification task [Mitchell, 1997], [Neville *et al.*, 2003].

4.3.2 Learning Link-Based Naïve Bayes Classifiers from OESRDSs

We note that the task of learning link-based Naïve Bayes classifiers reduces to estimating the probabilities $P(c_j)$, $P(v_i|c_j)$, and $P(u_i|c_j)$, for all class labels $c_j \in \mathbf{C}$, for all object attribute values $v_i \in \mathcal{V}(OA(x_i))$ and for all link description values $u_i \in \mathcal{V}(LD_l(x_i))$. These probabilities can be estimated from data using standard methods [Mitchell, 1997]. The resulting estimates constitute *sufficient statistics* for the parameters that specify a link-based Naïve Bayes classifier.

The task of learning link-based Naïve Bayes classifiers from an OESRDS $\{\mathcal{S}, \mathcal{D}, \mathcal{O}\}$ at a given level of abstraction Γ essentially entails estimating the relevant probabilities from the corresponding OESRDS.

We denote by $\sigma(v_i|c_j)$ the frequency counts of the value $v_i \in \mathcal{V}(OA(x_i))$, given the class label c_j; by $\sigma(u_i|c_j)$ the frequency counts of the value $u_i \in \mathcal{V}(LD_l(x_i))$, given the class label c_j; and by $\sigma(c_j)$ the frequency counts of the class label c_j, for a particular choice of a level of abstraction Γ in $\mathcal{O}$. The algorithm for learning a link-based Naïve Bayes classifier from an OESRDS works as follows:

1. Formulate statistical queries asking for the frequency counts $\sigma(v_i|c_j)$, $\sigma(u_i|c_j)$, and $\sigma(c_j)$, using the terms on the global cut Γ (γ_{Words} and γ_{Topic}).

2. Generate the link-based Naïve Bayes h_Γ corresponding to the cut Γ based on the computed frequency counts.

Choosing a level of abstraction Γ in $\mathcal{O}$ can result in data that are only *partially specified*. This can arise from *partially specified values* of an attribute[2]. In the next section, we define the partially specified values and provide a solution to the problem of dealing with partially specified data based on a statistical approach, called *shrinkage* [Stein, 1955], [James and Stein, 1961], [McCallum *et al.*, 1998].

4.4 Coping with Partially Specified Data

Definition 4 (Partially Specified Value): *An attribute value $v_i \in \mathcal{V}(X_i.A)$ in an AH $\mathcal{T}$ is partially specified (or under-specified) w.r.t. an attribute value $v_j \in \mathcal{V}(X_i.A)$ in the same AH if $v_i > v_j$; v_i is over-specified w.r.t. v_j if $v_i < v_j$; v_i is fully-specified w.r.t. v_j if $v_i = v_j$ [Zhang* et al.*, 2006].*

For example, given the AH in Figure 4.2a over the values of the attribute `Article`.*Topic*, the value ML is under-specified w.r.t. NN, since a machine learning article may be a neural network article, a reinforcement learning article, etc., but over-specified w.r.t. AI because each

[2]Partially specified data can also arise from partially specified schemas, i.e., when schema concepts are partially specified.

machine learning article is an artificial intelligence article. Furthermore, ML is fully-specified w.r.t. $MachineLearning$ (not shown in the figure).

The problem of learning from partially specified data (when only the attributes are partially specified) has been addressed by Zhang et al. [Zhang and Honavar, 2003], [Zhang *et al.*, 2006] in the setting where data instances are described by nominal attributes. This approach exploits the observation that the problem of learning from partially specified data is a natural generalization of the problem of learning from data in the presence of missing attribute values [Zhang *et al.*, 2006]. Hence, it is possible to adapt statistical approaches for dealing with missing data [Little and Rubin, 2002] to deal with partially specified data *under a variety of assumptions*, (e.g., the distribution of an under-specified attribute value is similar to that in another data source where the corresponding attribute is fully specified).

Learning from data instances (e.g., documents) where a multinomial model is assumed as the underlying generative model presents further complications: The "same" attribute can be over-specified in one place in the document and under-specified in another place in the document. To see this, consider the following paragraph taken from John C. Mallery's article "Semantic Content Analysis: A New Methodology for the RELATUS Natural Language Environment" [Mallery, 1991]:

"*Semantic content analysis differs from traditional computerized content analysis because it operates on the referentially integrated meaning representation of a text instead of a linear string of words. Rather than assessing the thematic orientation of texts based on the frequencies of word occurrences, this new methodology examines and interprets explicit knowledge representations of texts. [$\cdots$] Beyond semantic content analysis, lexical classification expands the referential performance because it provides a basic inference mechanism to extend indexation, semantically disambiguate word senses, and provide criteria for further deliberation in reference. [$\cdots$] The immediate political-analytic application of lexical classification is semantic content analysis.*"

In this paragraph, the term *semantic content analysis* is over-specified w.r.t. *lexical classification* which in turn is over-specified w.r.t. *methodology*; the term *this new methodology*

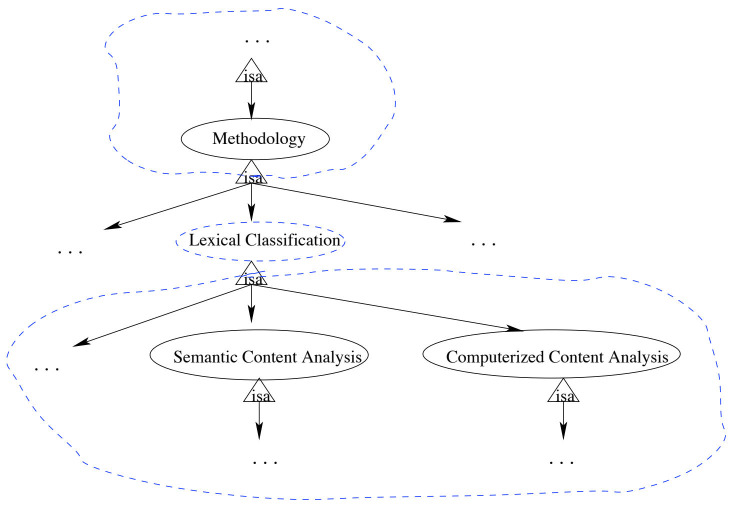


Figure 4.3: A fragment of an AH associated with the attribute *methodology*. The term *lexical classification* is at the same time under-specified (wrt *semantic content analysis*) and over-specified (wrt *methodology*) in the same document. Its cumulative frequency counts are computed from its own frequency counts, term frequency counts coming from all its descendants, and a percentage term frequency counts coming from all its ancestors. The participating nodes are dashed.

refers to the *semantic content analysis*.

The fact that a term can be at the same time over-specified and under-specified even in the same document (Figure 4.3), complicates the problem of dealing with partially specified data. Our solution to this problem is based on a statistical method, called *shrinkage* [Stein, 1955], [James and Stein, 1961], [McCallum *et al.*, 1998], that provides better estimates of a model parameters when the data is organized in an abstraction hierarchy (as defined in Section 2). Hence, we compute the counts for a term T_i in a document D as a summation (or *cumulative term frequency counts*) of the following three types of counts:

1. the term T_i frequency counts, i.e., the number of T_i occurrences in the document D;

2. the sum of term frequency counts coming from all its descendants T_k in the hierarchy, i.e., the term frequency counts of each node term T_k in the tree rooted at T_i (the term T_i is under-specified wrt any of its descendants);

3. a percentage term frequency counts coming from all its ancestors in the hierarchy, i.e., the percentage term frequency counts of each node term T_j in the tree on the path from

T_i to the root of the tree. Thus, the frequency counts of term T_j are distributed among all its descendants proportionally to their frequency counts (the term T_i is over-specified wrt any of its ancestors).

4.5 Experiments and Results

4.5.1 Experiments

The goal of the experiments is to explore the feasibility of our approach to learning link-based Naïve Bayes classifiers from ontology-extended structured relational data sources. We investigated: (i) the effect of learning classifiers at different levels of abstraction; (ii) the effect of partially specified data on the performance of our classifiers by comparing two approaches: using only term frequency counts and using cumulative term frequency counts.

We evaluated our approach on a subset of the Cora data set [McCallum *et al.*, 2000], that is a standard benchmark data set of research articles and their citations (which can be modeled as relations among the articles). The article topics are organized in a hierarchy with 73 leaves. We considered only the articles in Cora that are found on the Web and have topics in the topic hierarchy shown in Figure 4.2a, and that cite or are cited by at least one other article, so that there are no isolated nodes in our instantiation graph. Filtering the Cora data using these criteria yields a data set of 2469 articles and 8297 citations. We associate *abstraction hierarchies* (AHs) over the values of both attributes of the concept `Article`, i.e. `Article.`*Words* and `Article.`*Topic*.

The `Article.`*Topic* hierarchy is a subtree of the Cora topic hierarchy and contains only 8 leaves out of 73. These leaves along with their article numbers are shown in Table 4.1. The first six topics can be grouped into the more general term Machine Learning. Machine Learning, Data Mining, Natural Language Processing can be grouped into the more general term Artificial Intelligence (Figure 4.2a).

We designed the `Article.`*Words* abstraction hierarchy as follows: from the titles and abstracts of our collection of articles, we first removed punctuation and words that contain numbers, and then performed stemming and removal of stop words and words that occur less

Topic	NumArticles
Neural Networks (NN)	610
Genetic Algorithms (GA)	342
Case-Based (CB)	242
Probabilistic Methods (PM)	339
Theory (T)	325
Reinforcement Learning (ReL)	214
Data Mining (DM)	167
Natural Language Processing (NLP)	236

Table 4.1: Article topics along with their numbers in our subset of the Cora set.

than 10 times in the whole collection. From the remaining distinct terms we extracted 234 terms and manually organized them in an abstraction hierarchy using definitions in *Wikipedia* (available at http://en.wikipedia.org). Each node in the hierarchy corresponds to one of the 234 terms, and its associated term is more general w.r.t. any term from its descendants and more specific w.r.t. any term from its ancestors in the hierarchy.

In our experiments, we use four cuts, or *levels of abstraction*, through the abstraction hierarchy corresponding to the `Article`.*Words* attribute. These cuts are as follows:

- the most abstract level, i.e. the set of nodes corresponding to the children of the root form the first cut, denoted by **Cut 1**
- the second cut was obtained by replacing one node (randomly chosen) from **Cut 1** by its children; the resulting cut is denoted by **Cut 2**
- the most detailed level, i.e. the set of nodes corresponding to the leaves of the trees form the third cut, denoted by **Leaf Cut**
- a subset of nodes from the **Leaf Cut** was replaced by their parent, denoted by **Cut 3**.

4.5.2 Results

The effect of learning classifiers at different levels of abstraction. In our first set of experiments, we investigated the effect of learning classifiers at different *levels of abstraction*. We consider two tasks. In the first task suppose that we are interested in classifying computer

Level of Abstraction	Accuracy	Precision	Recall	F-Measure
Cut 1	0.86	0.80	0.47	0.51
Cut 2	0.86	0.83	0.46	0.51
Cut 3	**0.89**	**0.86**	**0.62**	**0.69**
Leaf Cut	**0.89**	0.84	0.61	0.68

Table 4.2: The classification results on the task of classifying articles into one of the three categories: *Data Mining*, *Machine Learning*, and *Natural Language Processing* for all four levels of abstraction considered: **Cut 1**, **Cut 2**, **Cut 3**, **Leaf Cut**.

science research articles into one of the three classes: *Data Mining*, *Machine Learning* and *Natural Language Processing*. Assume that we are given a cut in the AH corresponding to the `Article`.*Words* attribute. Sufficient statistics (corresponding to terms on the cut) are gathered so that the classifier can be trained.

The classification results for this task, for all four levels of abstraction, **Cut 1**, **Cut 2**, **Cut 3**, and **Leaf Cut**, are shown in Table 4.2. The performance measures of interest were estimated by averaging the performance of the classifier on the five runs of a cross-validation experiment. As can be seen from the table, classifiers trained at different levels of abstraction differ in their performance on the test data. Moving from a more general to a more specific level of abstraction does not necessarily improve the performance of the classifier because there may not be enough data to accurately estimate the classifier parameters. Similarly, moving from a more specific to a more general level of abstraction does not necessarily improve the performance since there may not be enough terms on the cut to discriminate between classes. As can be seen in Table 4.2, **Cut 3** yields the best performance among the four levels considered, although it is an abstraction of the **Leaf Cut**. This suggests the possibility of variants of the algorithm considered here that automatically search for an optimal level of abstraction using methods similar to those proposed in [Zhang and Honavar, 2003], [Zhang *et al.*, 2006].

In the second task suppose that we are interested in predicting whether the topic of a research article is *Neural Networks*. This requires finding a cut through the AH corresponding to the attribute `Article`.*Topic* that contains the term *Neural Networks*. The articles labeled

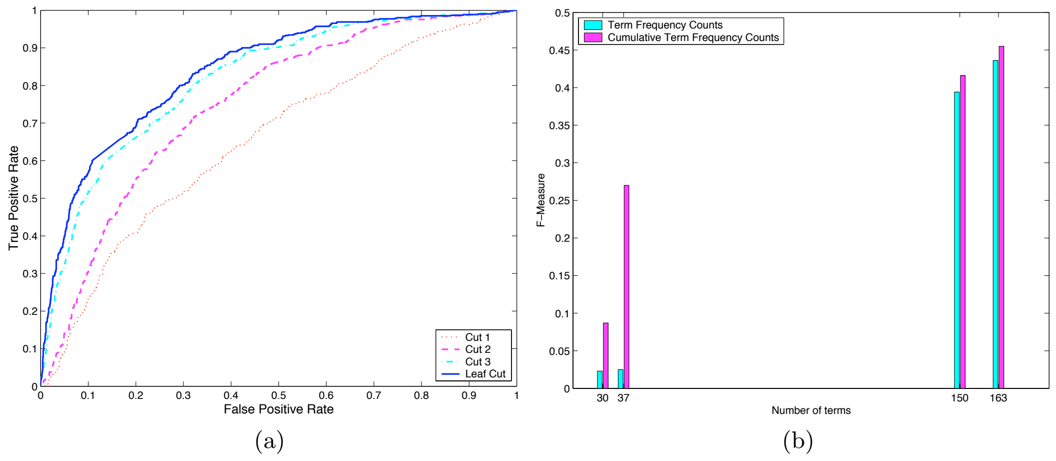


Figure 4.4: a) Comparison of the ROC curves of the link-based Naïve Bayes classifiers on the task of predicting whether a research article is *Neural Networks* for all four levels of abstraction considered in this study: **Cut 1**, **Cut 2**, **Cut 3**, and **Leaf Cut**. b) Comparison of the average F-Measure under two scenarious: the counts for each term are simply term frequency counts (shown in gray) and the counts for each term are cumulative term frequency counts (shown in black). The numbers on the x axes represent the number of terms on a particular cut.

with the term *Neural Networks* represent positive instances, while the rest represent negative instances.

Figure 4.4a shows the Receiver Operating Characteristic (ROC) curves on this binary classification task using the same four levels of abstraction as above. The ROC curves show the tradeoff between true positive and false positive predictions over their entire range of possible values. As can be seen from the figure, for any choice of the False Positive Rate, as we go from a coarser to a finer level of abstraction, the link-based Naïve Bayes classifier offers a higher True Positive Rate (Recall). The performance improvement is quite striking from **Cut 1** to **Cut 2**. However, the difference in performance between **Cut 3** and the **Leaf Cut** is rather small (and eventually levels off). Unlike the first task where the classifier trained based on **Cut 3** outperforms those trained based on the other cuts, in the second task the classifier trained based on the **Leaf Cut** outperforms the others. This can be explained by the fact that the number of parameters that need to be estimated is much smaller for this second task. As the number of parameters that need to be estimated increases, more and more data is required to obtain good estimates.

The effect of partially specified data on the performance of classifiers. In our second set of experiments, we investigated the effect of *partially specified data* on the performance of our classifiers.

Figure 4.4b compares the average F-Measure under two scenarios. In the first scenario, the counts for each term on the cut are simply the term frequency counts conditioned on the class attribute, i.e. the number of term occurrences in the collection of articles given the class attribute. In the second scenario, the counts for each term on the cut are the cumulative term frequency counts conditioned on the class attribute (see Section §4 for more details). As can be seen from the figure, taking into account the effect of partially specified data through the means of cumulative term frequency counts improves the average F-Measure for all four levels of abstraction considered in this study on the task of predicting whether the topic of an article is *Neural Networks*. We obtained similar results on the task of classifying articles into one of the three categories: *Data Mining*, *Machine Learning*, and *Natural Language Processing* (data not shown).

4.6 Extension to a Distributed Setting

4.6.1 Background and Motivation

Recent technological advances have led to a proliferation of autonomously operated, distributed data repositories in virtually every area of human endeavor. Many groups have developed approaches for querying semantically disparate sources [Levy, 2000], [Noy, 2004], [Doan and Halevy, 2005], [Calvanese *et al.*, 2005], for discovering semantic correspondences between ontologies [Kalfoglou and Schorlemmer, 2005], [Noy and Stuckenschmidt, 2005], and for learning from autonomous, semantically heterogeneous data [Caragea *et al.*, 2005]. One approach to learning from semantically disparate data sources is to first integrate the data from various sources into a warehouse based on semantics-preserving mappings between the data sources and a global integrated view, and then execute a standard learning algorithm on the resulting centralized, semantically homogeneous data. Given the autonomous nature of the data sources on the Web, and the diverse purposes for which the data are gathered, it is unlikely that a unique global view of the data that serves the needs of different users or communities of users under all scenarios exists. Moreover, in many application scenarios, it may be impossible to gather the data from different sources into a centralized warehouse because of restrictions on direct access to the data. This calls for approaches to learning from semantically disparate data that do not rely on direct access to the data but instead can work with results of statistical queries against an integrated view. We present a principled approach to the problem of learning classifiers from a collection of semantically disparate *relational* data sources in such a setting. We use link-based Naïve Bayes classifiers as an example to illustrate this approach. We show, under fairly general assumptions, that our approach is guaranteed to yield classifiers that are identical to those obtained from a centralized, integrated data warehouse constructed from the collection of semantically disparate relational data sources and associated ontologies and mappings. Experimental results using our implementation of link-based Naïve Bayes classifiers [Lu and Getoor, 2003], [Mitchell, 1997] for constructing text classifiers from text repositories based on related but disparate ontologies demonstrate the feasibility of the proposed approach.

4.6.2 Learning Classifiers from Semantically Heterogeneous Distributed Relational Data

Recall that An ontology $\mathcal{O}$ associated with a structured relational data source $\mathcal{D}$ is given by a *content ontology* that describes the semantics of the content of the data (e.g., the values and relations between values that `Article.`*Words* can take in $\mathcal{D}$). An ontology O in a distributed setting consists of a set of AHs $\{\mathcal{T}_1, \cdots, \mathcal{T}_l\}$, w.r.t. the *isa* relation (one AH for each data source). A *cut* (or *level of abstraction*) through an AH induces a partition of the set of leaves in that hierarchy. A *global cut* through an ontology consists of a set of cuts, one for each constituent AH.

A *mapping* ψ from a user ontology O_U to a data source ontology O_D (defining the semantics of two different views of the same domain) establishes semantic correspondences between the values of the attributes in O_U and the values of attributes in O_D. Examples of such semantic correspondences are equality, $x = y$ (i.e., x and y are *equivalent*), and inclusion

$x < y$ (i.e., y *subsumes* x, or y is *more general* than x) [Rajan *et al.*, 2005]. A subset of semantic correspondences between two AVHs corresponding to two ontologies O_U and O_D, $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, over the values of `Article.`*Topic* is $\{DM_{\mathcal{T}_1} = DataMining_{\mathcal{T}_2}, NN_{\mathcal{T}_1} < MachineLearning_{\mathcal{T}_2}, AI_{\mathcal{T}_1} > DataMining_{\mathcal{T}_2}\}$.

Definition: Let $\{\mathcal{S}_1, \mathcal{D}_1, \mathcal{O}_1\}, \cdots, \{\mathcal{S}_p, \mathcal{D}_p, \mathcal{O}_p\}$ be a set of OESRDSs. A user ontology O_U, together with a set of mappings $\{\psi_k | k = 1, \cdots, p\}$ from O_U to the data source ontologies $O_1, \cdots, O_p$ define a *user view* [Caragea *et al.*, 2007c], [Caragea *et al.*, 2005].

The user view *implicitly* specifies a user level of abstraction, corresponding to the leaf nodes of the hierarchies in O_U. The mappings ψ_k can be established manually or semi-automatically (e.g., using existing approaches to learning mappings between ontologies [Doan *et al.*, 2003]).

Learning Classifiers from a collection of OESRDSs We assume the existence of: (1) A collection of several related OESRDSs $\{\mathcal{S}_1, \mathcal{D}_1, \mathcal{O}_1\}, \cdots, \{\mathcal{S}_p, \mathcal{D}_p, \mathcal{O}_p\}$ for which the schemas and the ontologies are made *explicit* and the instances in the data sources are labeled according to some criterion of interest to a user (e.g., topic categories); (2) A user view, consisting of a user ontology O_U and a set of mappings ψ_k that relate O_U to $O_1, \cdots, O_p$; (3) A hypothesis class H (e.g., Bayesian classifiers) defined over an *instance space*; (4) A performance criterion P (e.g., accuracy on a classification task).

Under the above assumptions, learning classifiers from a collection of semantically heterogeneous OESRDSs can be formulated as follows: *the task of a learner L is to output a hypothesis $h \in H$ that optimizes P, via the mappings $\{\psi_k\}$.*

In this setting, the statistical query answering component of the algorithm poses a *statistical query* against the user view; decomposes the query into sub-queries and translates them into queries that can be answered by the individual data sources (based on the mappings from the user ontology to the data source ontologies); and assembles the answer to the original query from the answers returned by the individual data sources (Figure 4.5). Once a classifier has been learned, it can be used to classify data that is at the disposal of the user.

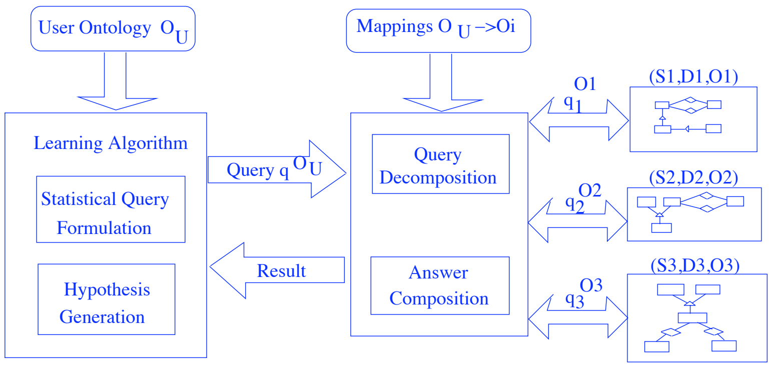


Figure 4.5: Learning classifiers from ontology-extended structured relational data sources.

Learning Link-Based Classifiers from a collection of OESRDs We now proceed to describe our algorithm for learning classifiers from a collection of OESRDSs. We adapt the link-based iterative algorithm introduced by Lu and Getoor [Lu and Getoor, 2003] to learning classifiers from OESRDSs (see Section 4.3 for more details on the algorithm).

Learning link-based Naïve Bayes classifiers reduces to estimating the probabilities $P(c_j)$, $P(v_i|c_j)$, and $P(u_i|c_j)$, for all classes $c_j \in \mathbf{C}$, for all object attribute values $v_i \in \mathcal{V}(OA(x_i))$ and for all link description values $u_i \in \mathcal{V}(LD_l(x_i))$ using standard methods [Mitchell, 1997] (see Section 4.3 for an explanation of the link description).

We denote by $\sigma(v_i|c_j)$ the frequency count of the value $v_i \in \mathcal{V}(OA(x_i))$, given the class c_j; by $\sigma(u_i|c_j)$ the frequency count of the value $u_i \in \mathcal{V}(LD_l(x_i))$, given the class c_j; and by $\sigma(c_j)$ the frequency count of the class c_j, in the user view. The algorithm for learning a link-based Naïve Bayes classifier from a set of related OESRDSs works as follows:

1. Select a global user cut Γ through the user ontology (AVHs). In particular, the user cut corresponds to the set of primitive values (i.e., leaves in AHs).

2. Apply the mappings ψ_k to find a cut Γ_k, corresponding to the user cut Γ, in each OESRDS $\{\mathcal{S}_k, \mathcal{D}_k, \mathcal{O}_k\}$.

3. Formulate statistical queries asking for the frequency counts $\sigma(v_i|c_j)$, $\sigma(u_i|c_j)$, and $\sigma(c_j)$, using terms in the user cut Γ.

4. Translate these queries into queries expressed in the ontology of each OESRDS $\{\mathcal{S}_k, \mathcal{D}_k, \mathcal{O}_k\}$, using terms in the cut Γ_k, and compute the local counts $\sigma_k(v_i|c_j)$, $\sigma_k(u_i|c_j)$, and $\sigma_k(c_j)$ from each OESRDS $\{\mathcal{S}_k, \mathcal{D}_k, \mathcal{O}_k\}$.

5. Send the local counts to the user and add them up to compute the global frequency counts $\sigma(v_i|c_j)$, $\sigma(u_i|c_j)$, and $\sigma(c_j)$.

6. Generate the link-based Naïve Bayes h_Γ corresponding to the cut Γ based on the global frequency counts.

4.6.3 Exactness

Definition: An algorithm $\mathcal{L}_{distributed}$ for learning from OESRDSs $\{\mathcal{S}_1, \mathcal{D}_1, \mathcal{O}_1\}, \cdots, \{\mathcal{S}_p, \mathcal{D}_p, \mathcal{O}_p\}$, via the mappings $\{\psi_k\}$, is *exact* wrt its centralized counterpart $\mathcal{L}_{centralized}$, if the hypothesis produced by $\mathcal{L}_{distributed}$ is identical to that produced by $\mathcal{L}_{centralized}$ from the data warehouse D that is constructed by integrating the data sources $\mathcal{D}_1, \cdots, \mathcal{D}_p$, according to the user view, via the same mappings $\{\psi_k\}$.

The *exactness* criterion defined above assumes that it is possible, in principle, to create an integrated data warehouse in the centralized setting. In practice, the data sources $\mathcal{D}_1, \cdots, \mathcal{D}_p$ might impose access constraints on the user. These constraints might prohibit the user from retrieving instance data from some of the data sources (e.g., due to restrictions on the queries that can be answered by the data source, bandwidth limitations, or privacy considerations), while allowing retrieval of answers to statistical queries against the data.

Note that the algorithm for learning a link-based Naïve Bayes classifier from OESRDSs using statistical queries is *exact* relative to the link-based Naïve Bayes classifier obtained by executing the standard algorithm on the data warehouse $\mathcal{D}$ obtained by integrating the set of OESRDSs $\{\mathcal{S}_1, \mathcal{D}_1, \mathcal{O}_1\}, \cdots, \{\mathcal{S}_p, \mathcal{D}_p, \mathcal{O}_p\}$ (using the same set of mappings $\{\psi_k\}$). This follows from the observation that $\sigma(v_i|c_j) = \sum_{i=1}^{k} \sigma_k(v_i|c_j) = \sigma_{\mathcal{D}}(v_i|c_j)$, $\sigma(u_i|c_j) = \sum_{i=1}^{k} \sigma_k(u_i|c_j) = \sigma_{\mathcal{D}}(u_i|c_j)$, $\sigma(c_j) = \sum_{i=1}^{k} \sigma_k(c_j) = \sigma_{\mathcal{D}}(c_j)$, when there is no overlap between the distributed sources. Note that dealing with duplication of instances between any two data sources requires establishing correspondences between individual instances [Parag and Domingos, 2004].

4.6.4 Experiments and Results

The goal of the experiments is to demonstrate the feasibility of the statistical query-based approach to learning link-based Naïve Bayes classifiers from a set of related, ontology-extended data sources from a *user's point of view.* We investigated: (i) the effect of learning classifiers at different levels of abstraction, by refining the user cut, starting with the most abstract cut; (ii) the effect of partially specified data on the performance of our classifiers by comparing two approaches: using only term frequency counts and using cumulative term frequency counts.

We evaluated our approach to learning classifiers from a collection of semantically disparate relational data sources on the same data set as for learning classifiers from a single relational data source. To simulate the distributed setting, we randomly partitioned the Cora dataset into two subsets, such that the class distribution in each subset is similar to the class distribution in the entire dataset (stratified partitioning). In our experiments, we used one-to-one, manually-defined mappings between the user ontology and the data source ontologies. Moreover, the same four cuts, or *levels of abstraction*, through the user hierarchy corresponding to the `Article`.*Words* attribute were also considered in the distributed setting.

As the results in the distributed setting are identical with the results in the centralized setting (i.e., the results in Section 4.5), we do not repeat them.

4.7 Summary and Discussion

We have described an algorithm for learning link-based Naïve Bayes classifiers from ontology-extended structured relational data sources. We have evaluated the resulting classifiers on a text categorization task for several choices of levels of abstraction. The results of our experiments show that more abstract levels can yield better performing classifiers. We have also addressed some of the unique challenges presented by partial specification of data which is unavoidable on text data.

The problem of learning classifiers from relational data sources has received much attention in the machine learning literature [Getoor *et al.*, 2002], [Neville *et al.*, 2003], [Taskar *et al.*, 2002]. In contrast to these methods, we have presented in this paper an algorithm for learning predictive models from relational data sources which is annotated with relevant meta data. Zhang et al. [Zhang and Honavar, 2003], [Zhang *et al.*, 2006] proposed an approach to learning classifiers from partially specified data over nominal attributes when data are stored in a *flat* table. McCallum et al. [McCallum *et al.*, 1998] have used a well-known statistical approach, called *shrinkage*, in a hierarchy of classes to improve classification accuracy. Inspired from this work, we have used *shrinkage* to handle partially specified text data where a multinomial model is assumed as the underlying generative model for the text documents.

Furthermore, we have described a general strategy for learning link-based Naïve Bayes classifiers [Lu and Getoor, 2003] from a collection of semantically disparate relational data sources. The proposed approach exploits mappings between a user ontology and data source ontologies to gather the necessary statistics for learning the classifiers. The resulting algorithms for learning link-based classifiers from semantically disparate relational data sources can be shown to be *provably exact* relative to their centralized counterparts under fairly general assumptions.

The problem of learning classifiers from a semantically homogeneous relational database has received much attention in the recent machine learning literature [Getoor *et al.*, 2002], [Neville *et al.*, 2003]. There is a large body of literature on learning predictive models from distributed data (see [Kargupta and Chan, 2000], [Caragea and Honavar, 2008] for surveys). Of particular interest in our setting is the work of Caragea et al [Caragea *et al.*, 2005] that introduced a general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the data sources. A basic strategy for coping with semantically disparate data was outlined in [Caragea *et al.*, 2005]. However, these works assumed that data are stored in a *flat* table.

4.7.1 Discussion

Due to the unavailability of data sources that are already annotated with relevant meta data, we performed experiments on only one data set, the relational Cora data set [McCallum *et al.*, 2000] . In our experiments, we manually associated an abstraction hierarchy over values of the attribute `Article.`*Words*. The hierarchy over the values of the attribute `Article.`*Topic* is provided by the Cora data set.

To simulate a distributed setting, the Cora data set was randomly partitioned into two data sets. We manually associated ontologies over values of the attributes `Article.`*Topic* and `Article.`*Words* for the two data sets and a users point of view in order to have control over the mappings. The mappings between the user ontology O_U and the data source ontologies O_1 and O_2 are also specied manually. Because of the difficulty of manually specifying mappings between two large ontologies, in our experiments, we used one-to-one mappings.

The algorithm presented here assumes a pre-specified *level of abstraction* defined by a global cut through the ontology. Our experiments have shown that the choice of the level of abstraction can impact the performance of the classifier. This suggests the possibility of improving the algorithm using a top down, iterative approach to refining the cut (see Figure 4.2b), starting with the most abstract cut in the abstraction hierarchy corresponding to the `Article.`*Words* attribute until an “optimal cut” and, thus, an optimal level of abstraction is identified for the learning task at hand. This strategy is similar to that adopted in [Zhang *et al.*, 2006] in learning Naïve Bayes classifiers from a single *flat* table, in the presence of attribute value taxonomies and partially specified data.

We investigated the effect of partially specified data on the performance of classifiers designed to predict the topic of a text document. Our experiments have shown that incorporating the effect of partially specified data through the means of cumulative term frequency counts (i.e., *shrinkage*) in computing the statistics used by the learning algorithm improves the performance of the resulting classifiers.

Some directions for future research include: implementation and experimental evaluation of a large class of algorithms for learning predictive models from structured relational data sources annotated with relevant meta data, including multi-modal data (e.g., images); exploring the effect of using different ontologies and mappings; the use of automated approaches to establish mappings between ontologies [Doan *et al.*, 2003]; the effect of degree of incompleteness of mappings; the effects of errors in mappings, etc.

CHAPTER 5. SUMMARY AND DISCUSSION

5.1 Thesis Summary

Topologically structured data present an important characteristic: they have a topology that reflects intrinsic dependencies between the constituent elements of the data. Examples of topologically structured data include: text data, image data, biological sequence data, social network data, etc. Hence, for each type of topologically structured data, we can define the notion of "neighborhood" for an element of interest. For example, in the case of biological sequence data, the neighborhood of an element (a residue) is given by the elements to its left and to its right in the sequence (called, a "window" around the residue of interest). In the case of image data, the neighborhood is given by the eight adjacent pixels in the image.

In this thesis, we present an abstraction-based approach that simplifies the data representation used by a learner on topologically structured data with applications to biological sequence and text classification tasks (when a multinomial model is assumed as the underlying generative model for sequence data). Our abstraction-based approach exploits the complementary strengths of *super-structuring* (constructing complex features by combining existing features) and *abstraction* (grouping similar features to generate more abstract features). Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (hence, it increases the complexity of the learned models) whereas *abstraction* helps reduce the number of model parameters by simplifying the data representation.

Although we have focused on super-structuring in the case of sequence data to enrich the data representation, and hence increase the performance of learned models, in general, any topology that reflects the interactions between elements of structured data can be used to guide super-structuring. For example, in the case of image data, this involves generating features from spatially contiguous elements of an image [Jolion and Rosenfeld, 1994], [Lazebnik *et al.*, 2006], [Ommer and Buhmann, 2007], [Cantoni and Petrosino, 2002], [Bosch *et al.*, 2007].

We evaluated our abstraction-based models on biological sequence and text data in settings where the sequences in a data set are *independent and identically distributed*, and where the sequences are *highly inter-connected*.

The results of our experiments show that:

- adapting data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those obtained using super-structuring alone (whose size grows exponentially with k). Moreover, the performance of such models is similar and, in some cases, better compared to the performance of models that use only super-structuring, i.e., more abstract levels can yield better performing models (which could be explained by *overfitting*) **(Chapters 2, 3 and 4)**.

- simplifying data representation using abstraction yields better performing models than those obtained by feature selection. On sequence classification tasks, selecting a subset of *super-structures* by feature selection can substantially reduce the number of model parameters. However, this approach of combining super-structuring and feature selection may not capture important sequence patterns (*motifs*, in the case of biological sequences) that are removed during the selection process **(Chapter 2)**.

- organizing *super-structures* (i.e., k-grams) in abstraction hierarchies based on the conditional distributions of the next letter in the sequence rather than based on the class conditional distributions induced by the k-grams produces more suitable abstraction hierarchies for abstraction-based Markov models, and hence, better performing models. Furthermore, this data organization procedure allows to incorporate information available in the unlabeled data, that would otherwise be ignored by standard Markov models **(Chapters 3)**.

- incorporating the effect of partially specified data using *shrinkage* when computing the sufficient statistics used by a learning algorithm improves the performance of the resulting classifiers on text data (a multinomial model is assumed as the underlying generative model for text data). In such settings, partially specified data inevitably result from choosing a level of abstraction **(Chapter 4)**.

5.2 Summary of Contributions and Future Directions

In what follows, we present the context of our work, summarize the contributions of this thesis, and outline some directions for further research. We start with the setting in which the sequences in a data set are *independent and identically distributed.*

5.2.1 Independent and identically distributed sequence data

5.2.1.1 Context of our work

Segal et al. [Segal *et al.*, 2002] and McCallum et al. [McCallum *et al.*, 1998] have used abstraction hierarchies over *classes* to improve classification accuracy. Slonim and colleagues [Slonim and Tishby, 1999], [Slonim *et al.*, 2001] have introduced the *information bottleneck* (IB) methods that perform model compression by identifying a set of *bottleneck variables.* In IB, the data in a given data set is organized in abstraction hierarchies based on the similarity of the class conditional distributions. Inspired from the IB methods, Punera et al. [Punera *et al.*, 2006] have designed an algorithm to organize *classes* in a given data set in an n-ary taxonomy. They have shown that n-ary tree based taxonomies provide a more "natural" way to organize classes in a hierarchy than the binary tree based taxonomies. Zhang et al. [Zhang *et al.*, 2006] have used abstraction hierarchies over *nominal variables* to build compact yet accurate classifiers. Silvescu [Silvescu, 2008] has introduced a method, called *combining super-structuring and abstraction* (SS-A), where abstraction hierarchies are learned from *sequence data.* Specifically, super-structuring is used to extract contiguous sub-sequences of a certain length k from sequence data, followed by abstraction that is used to organize them in an abstraction hierarchy. Silvescu [Silvescu, 2008] has trained Naïve Bayes classifiers to label biological sequences. This work is similar in spirit to the *agglomerative information bottleneck* (AIB) method of Slonim and Tishby [Slonim and Tishby, 1999]. However, AIB requires an iterative process to estimate parameters for the *bottleneck variables*, whereas SS-A requires only a simple weighted aggregation of the existing parameters.

5.2.1.2 Summary of contributions

- **We have extended SS-A to text classification (Chapter 2).** We found that:
  - The performance of classifiers trained using super-structuring representation was worse than that of classifiers trained using a "bag of words" feature representation which is consistent with the work of Dumais et al. [Dumais *et al.*, 1998].
  - We compared Naïve Bayes trained using an abstraction-based representation with Naïve Bayes trained using a "bag of words" representation, and with Naïve Bayes trained using feature selection-based representation. Consistent with the results obtained using SS-A, the abstraction-based representations result in better performing models than those based on feature selection, and in similar performing models compared to the "bag of words" representations.
  - Although feature selection and abstraction are seemingly antagonistic, better results can be obtained by using them sequentially, whereas feature selection is used to remove the irrelevant features, and abstraction is used to deal with feature redundancy.
- **We have explored Support Vector Machine classifiers in the SS-A framework on biological sequence prediction tasks (Chapter 2).** We found that:
  - The combination of super-structuring and abstraction results in better performing models than that of super-structuring and feature selection.
  - Unlike Naïve Bayes classifiers, SVMs trained on super-structuring followed by abstraction representation substantially outperform those trained on super-structuring

representation. Hence, abstraction can be seen as a *regularizer* for SVM classifiers which are sensitive to *overfitting*.

- **We have introduced and studied Abstraction Augmented Markov Models (AAMMs) (Chapter 3).** Specifically,
  - **we have shown how to represent and learn AAMMs.**
  - **we have designed a clustering algorithm that produces suitable abstraction hierarchies for AAMMs.**
  - Furthermore, our clustering algorithm allows to incorporate information available in the unlabeled data that are rather plentiful. Hence, **we have shown how to produce more accurate models compared to the standard Markov models in a *semi-supervised setting*.**
  - Although AAMMs are less expressive models than HMMs, **we have shown that they are easier to learn models compared to HMMs.**
- We found that:
  - Abstraction-based Markov models substantially outperform the standard Markov models in both supervised and semi-supervised settings.
  - The AAMM clustering algorithm helps produce more suited abstraction hierarchies than agglomerative information bottleneck for abstraction-based Markov models.

5.2.1.3 Future directions

As the immediate next steps, we plan to:

- Explore the combination of super-structuring, feature selection and abstraction on biological sequence data, whereas super-structuring helps enrich the data representation, feature selection helps remove the irrelevant feature, and abstraction helps handle redundancy in the features;

- Compare SS-A with more sophisticated feature selection methods;
- Explore Support Vector Machine classifiers in the context of AAMMs;
- Compare AAMMs with Hidden Markov Models [Durbin *et al.*, 2004] and Interpolated Markov Models [Jelinek and Mercer, 1980] on a broad range of sequence classification tasks, including text classification;
- Use abstraction to design Abstraction Augmented Hidden Markov Models (AAHMMs) and Abstraction Augmented Interpolated Markov Models (AAIMMs).

In related work, Liu et al. [Liu *et al.*, 2008] developed a system called *TEFE (Time-Efficient Feature Extraction)* to extract a "good" set of features from image data, and the goal was to trade off the test accuracy against the test time cost. TEFE starts by ranking all m features by their extraction time cost. Then, a collection of m SVM classifiers are trained in cascade such that the i^{th} SVM is trained on the best ranked i^{th} features. During testing, the best ranked feature is extracted from a test image. If the output returned by the 1^{st} SVM is higher than a user threshold θ, the prediction (made by the 1^{st} SVM) is returned. Otherwise the next feature is extracted from the image and the process is repeated until there are no more features available or a predefined amount of time t is exceeded.

Motivated by our findings that abstraction results in better performing models than feature selection, and the fact that image data is a special case of topologically structured data, we plan to:

- Extend our abstraction-based approach to image data.

In related work of Zhang et al. [Zhang *et al.*, 2006], the authors present an approach to searching for a compact classifier by trading off the complexity against the accuracy of the models. Similar to this approach, we plan to:

- Design a scoring function to determine the "best" cut in an abstraction hierarchy to trade off the complexity against the accuracy of the learned models.

Furthermore, as the quality of classifiers that use as input *abstract features* highly depends on the quality of the abstraction hierarchies, we plan to:

- Design well suited top-down clustering algorithms for building abstraction hierarchies.

Other directions for future research of the work in Chapter 2 and 3 may include:

- Extension of abstraction-based models to an unsupervised setting
- Comparison of abstraction-based models trained in a semi-supervised setting with those trained in a transductive setting

5.2.2 Relational sequence data

In a more general setting, sequence data is highly inter-connected, e.g. news articles *link* to related news articles, Wikipedia articles *link* to related internal or external articles. We refer to such data as relational or network data.

5.2.2.1 Context of our work and summary of contributions

The problem of learning classifiers from relational data sources has received much attention in the machine learning literature, e.g. [Getoor *et al.*, 2002], [Neville *et al.*, 2003], [Taskar *et al.*, 2002]. However, there has been limited exploration of techniques that use structured relational data sources that are annotated with relevant meta data.

- **We have described an Expectation-Maximization algorithm for learning link-based Naïve Bayes classifiers from relational data that are organized in abstraction hierarchies associated with both the input (i.e., text data) and the output (i.e., class variable) (Chapter 4).**

Choosing a particular level of abstraction inevitably results in partially specified data. Zhang et al. [Zhang and Honavar, 2003], [Zhang *et al.*, 2006] have previously proposed an approach to learning classifiers from partially specified data *over nominal attributes* when data are stored in a *flat* table. McCallum et al. [McCallum *et al.*, 1998] have used a well-known statistical approach, called *shrinkage*, in a hierarchy of classes to improve the statistical estimates of model parameters when the data at a node is sparse. The authors have shown that shrinkage improves classification accuracy. Shrinkage has also been used in language modeling [Jelinek and Mercer, 1980] for smoothing n-grams on speech recognition tasks.

- **We have used *shrinkage* to cope with partially specified text data where a multinomial model is assumed as the underlying generative model for the text documents (Chapter 4).**
- We found that:
  - more abstract levels can yield better performing classifiers
  - incorporating partially specified data using shrinkage yields better estimates of model parameters

Given the existence of massive autonomous distributed data sources that are gathered for diverse purposes, it is unlikely that a unique global view of the data that serves the needs of different users or communities of users under all scenarios exists. Moreover, it may be impossible to gather data from different sources into a centralized warehouse because of restrictions on direct access to the data. Kargupta and Chan [Kargupta and Chan, 2000] and Caragea et al. [Caragea *et al.*, 2005] (among others) have learned classifiers from semantically heterogeneous distributed data.

- **We have extended the approach introduced in [Caragea *et al.*, 2005] to the semantically heterogeneous, distributed, relational data setting (Chapter 4).** Specifically,
  - **we have described a strategy for learning link-based Naïve Bayes classifiers from a collection of semantically disparate relational data sources that exploits mappings between a user ontology and data source ontologies to gather necessary statistics for learning classifiers.**

  - The algorithm for learning link-based Naïve Bayes classifiers from such data using statistical queries is *exact* relative to link-based Naïve Bayes classifiers obtained in a centralized setting when there is no overlap between the distributed data sources.

5.2.2.2 Future directions

Some directions for future research of the work in Chapter 4 may include:

- application of the abstraction-based approach to annotate multi-modal data (e.g., images)
- exploration of approaches (e.g., approaches developed in earlier chapters) to automatically constructing abstraction hierarchies from data
- exploration of the effect of:
  - using different abstraction hierarchies and mappings in a distributed setting
  - the degree of incompleteness of mappings
  - errors in mappings
- usage of automated approaches to establish mappings between abstraction hierarchies.

BIBLIOGRAPHY

[Andorf *et al.*, 2004] C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar. Learning classifiers for assigning protein sequences to gene ontology functional families. In *Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004), India*, 2004.

[Baker and McCallum, 1998] D. Baker and A. McCallum. Distributional clustering of words for text classification. In *Proc. of SIGIR-98*, 1998.

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

[Bishop, 2006] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[Blei *et al.*, 2004] D. M. Blei, T. Griffiths, M. I. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS) 16*, 2004.

[Blom *et al.*, 2004] N. Blom, T. Sicheritz-Ponten, R. Gupt, S. Gammeltoft, and S. Brunak. Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence. *Proteomics*, 4(6):1633–49, 2004.

[Borgwardt *et al.*, 2005] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56, 2005.

[Bosch *et al.*, 2007] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408, New York, NY, USA, 2007. ACM.

[Burges, 1998] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Calvanese *et al.*, 2005] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. In *Proceedings of the 10th International Conference on Database Theory (ICDT 2005)*, volume 3363 of *LNCS*, pages 321–336. Springer, 2005.

[Cantoni and Petrosino, 2002] V. Cantoni and A. Petrosino. Neural recognition in a pyramidal structure. *Neural Networks, IEEE Transactions on*, 13(2):472–480, Mar 2002.

[Caragea and Honavar, 2008] D. Caragea and V. Honavar. Learning classifiers from distributed data sources. *Encyclopedia of Database Technologies and Applications*, 2008.

[Caragea and Honavar, 2009] C. Caragea and V. Honavar. *Machine Learning in Computational Biology.* Encyclopedia of Database Systems, (Raschid, L., Editor), Springer, In press., 2009.

[Caragea *et al.*, 2005] D. Caragea, J. Zhang, J. Bao, J. Pathak, and V. Honavar. Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous information sources. In *Proceedings of the International Conference on Algorithmic Learning Theory*, LNCS, pages 13–44, Singapore, 2005.

[Caragea *et al.*, 2007a] C. Caragea, J. Sinapov, D. Dobbs, and V. Honavar. Assessing the performance of macromolecular sequence classifiers. In *Proceedings of IEEE 7th International Conference on BioInformatics and BioEngineering (BIBE)*, pages 320–326, 2007.

[Caragea *et al.*, 2007b] C. Caragea, J. Sinapov, A. Silvescu, D. Dobbs, and V. Honavar. Glycosylation site prediction using ensembles of support vector machine classifiers. *BMC Bioinformatics*, 8(438):doi:10.1186/1471–2105–8–438, 2007.

[Caragea *et al.*, 2007c] D. Caragea, J. Bao, and V. Honavar. Learning relational bayesian classifiers on the semantic web. In *Proceedings of the IJCAI 2007 SWeCKa Workshop*, India, 2007.

[Caragea *et al.*, 2008] C. Caragea, J. Sinapov, D. Dobbs, and V. Honavar. Using global sequence similarity to enhance macromolecular sequence labeling. In *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 104–111, 2008.

[Caragea *et al.*, 2009a] C. Caragea, D. Caragea, and V. Honavar. Learning link-based classifiers from ontology-extended textual data. In *Proceedings of ICTAI-09*, 2009.

[Caragea *et al.*, 2009b] C. Caragea, D. Caragea, and V. Honavar. Learning link-based naive bayes classifiers from ontology-extended distributed data. In *Proceedings of ODBASE-09*, 2009.

[Caragea *et al.*, 2009c] C. Caragea, A. Silvescu, D. Caragea, and V. Honavar. Abstraction Augmented Markov Models. In *SIAM Conference on Data Mining, Under Review*, 2009.

[Caragea *et al.*, 2009d] C. Caragea, A. Silvescu, D. Caragea, and V. Honavar. Abstraction Augmented Markov Models. In *NIPS-09 Workshop on Machine Learning on Computational Biology*, 2009.

[Caragea *et al.*, 2009e] C. Caragea, J. Sinapov, D. Dobbs, and V. Honavar. Mixture of experts models to exploit global sequence similarity on biomolecular sequence labeling. *BMC Bioinformatics*, (doi:10.1186/1471-2105-10-S4-S4), 2009.

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

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

[Charniak, 1993] E. Charniak. *Statistical Language Learning, Cambridge: 1993.* MIT Press, 1993.

[Cover and Thomas, 1991] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[Craven *et al.*, 1998] Mark Craven, Dan Dipasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, and Kamal Nigam. Learning to extract symbolic knowledge from the world wide web. In *AAAI-98*, pages 509–516. AAAI Press, 1998.

[desJardins *et al.*, 2000] M. desJardins, L. Getoor, and D. Koller. Using feature hierarchies in bayesian network learning. In *In Proc. of SARA '02*, pages 260–270, Springer-Verlag, 2000.

[Doan and Halevy, 2005] A. Doan and A. Halevy. Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine, Special Issue on Semantic Integration*, 26(1):83–94, 2005.

[Doan *et al.*, 2003] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *VLDB Journal, Special Issue on the Semantic Web*, 2003.

[Duda *et al.*, 2001] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2001.

[Dumais *et al.*, 1998] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*, pages 148–155, 1998.

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

[Emanuelsson *et al.*, 2000] 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.

[Gardy *et al.*, 2003] J. L. Gardy, C. Spencer, K. Wang, M. Ester, G. E. Tusnady, I. Simon, S. Hua, K. deFays, C. Lambert, K. Nakai, and F. S.L. Brinkman. Psort-b: improving protein subcellular localization prediction for gram-negative bacteria. *Nucleic Acids Research*, 31(13):3613–17, 2003.

[Gavel and von Heijne, 1990] Y. Gavel and G. von Heijne. Sequence differences between glycosylated and non-glycosylated asn-x-thr/ser acceptor sites: implications for protein engineering. *Protein Engineering*, 3(5):433–442, 1990.

[Getoor *et al.*, 2002] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. *Journal of Machine Learning Research*, 3:679–707, December 2002.

[Getoor *et al.*, 2007] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[Globerson and Tishby, 2003] A. Globerson and N. Tishby. Sufficient dimentionality reduction. *Journal of Machine Learning Research*, 2003.

[Globerson *et al.*, 2007] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 2007.

[Guyon and Elisseeff, 2003] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[Hinton and Roweis, 2002] G. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 857–864, 2002.

[Honavar and Uhr, 1989a] V. Honavar and L. Uhr. Generation, local receptive fields, and global convergence improve perceptual learning in connectionist networks. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*, 1989.

[Honavar and Uhr, 1989b] V. Honavar and L. Uhr. Generation, local receptive fields, and global convergence improve perceptual learning in connectionist networks. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*, pages 180–185. San Mateo, CA: Morgan Kaufmann., 1989.

[Honavar, 1992] V. Honavar. Some biases for efficient learning of spatial, temporal, and spatio-temporal patterns. In *Proceedings of International Joint Conference on Neural Networks.*, 1992.

[James and Stein, 1961] W. James and C. Stein. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pages 361–379, University of California Press, 1961.

[Jelinek and Mercer, 1980] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. *Pattern Recognition in Practice*, pages 381–397, 1980.

[Joachims, 1997] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tf-idf for text categorization. In *ICML*, pages 143–151, 1997.

[Jolion and Rosenfeld, 1994] J.M. Jolion and A. Rosenfeld. *A Pyramid Framework for Early Vision: Multiresolutional Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[Jonyer *et al.*, 2004] I. Jonyer, L. Holder, and D. Cook. Mdl-based context-free graph grammar induction and applications. *International Journal of Artificial Intelligence Tools*, 13:45–64, 2004.

[Kalfoglou and Schorlemmer, 2005] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. In *Proceedings of Semantic Interoperability and Integration*, Dagstuhl, Germany, 2005.

[Kang *et al.*, 2004] D.-K. Kang, A. Silvescu, J. Zhang, and V. Honavar. Generation of attribute value taxonomies from data for accurate and compact classifier construction. In *Proceedings of ICDM 2004, Brighton, UK*, 2004.

[Kang *et al.*, 2005] D-K. Kang, J. Zhang, A. Silvescu, and V. Honavar. Multinomial event model based abstraction for sequence and text classification. In *In: Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA 2005)*, 2005.

[Kargupta and Chan, 2000] H. Kargupta and P. Chan. *Advances in Distributed and Parallel Knowledge Discovery.* AAAI/MIT, 2000.

[Koller and Sahami, 1996] D. Koller and M. Sahami. Toward optimal feature selection. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 284–292. Morgan Kaufmann Publishers, 1996.

[Koul *et al.*, 2008] N. Koul, V. Bahirwani, C. Caragea, D. Caragea, and V. Honavar. Using sufficient statistics to learn predictive models from massive data sets. In *Proceedings of the ACM/IEEE/WIC Conference on Web Intelligence (WI)*, Sydney, Australia, 2008.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings 18th International Conference on Machine Learning*, pages 282–289, 2001.

[Lazebnik *et al.*, 2006] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[Lee *et al.*, 2008] J.H. Lee, M. Hamilton, C. Gleeson, C. Caragea, P. Zaback, J. Sander, X. Lee, F. Wu, M. Terribilini, V. Honavar, and D. Dobbs. Striking similarities in diverse telomerase proteins revealed by combining structure prediction and machine learning approaches. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, volume 13, pages 501–512, 2008.

[Levy, 2000] A. Levy. Logic-based techniques in data integration. In *Logic-based artificial intelligence*, pages 575–595. Kluwer Academic Publishers, 2000.

[Lin, 1991] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

[Little and Rubin, 2002] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data.* Wiley, 2002.

[Liu and Motoda, 1998a] H. Liu and H. Motoda. *Feature Extraction, Construction and Selection.* Springer, 1998.

[Liu and Motoda, 1998b] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Springer, 1998.

[Liu *et al.*, 2008] Li-Ping Liu, Yang Yu, Yuan Jiang, and Zhi-Hua Zhou. TEFE: A time-efficient approach to feature extraction. In *Proceedings of ICDM-08*, pages 423–432, 2008.

[Lu and Getoor, 2003] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

[Mallery, 1991] J. C. Mallery. Semantic content analysis: A new methodology for the relatus natural language environment. 1991.

[Mardia *et al.*, 1979] K. V. Mardia, J. T. Kent, and J.M. Bibby. *Multivariate Analysis.* Academic Press, 1979.

[McCallum and Nigam, 1998] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[McCallum *et al.*, 1998] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In Jude W. Shavlik, editor, *Proceedings of ICML-98*, pages 359–367, Madison, US, 1998.

[McCallum *et al.*, 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the contruction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[Mika *et al.*, 1999] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11.* MIT Press, 1999.

[Mitchell, 1997] T. M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[Neville *et al.*, 2003] J. Neville, D. Jensen, and B. Gallagher. Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining*. IEEE Press, 2003.

[Noble and Ben-Hur, 2007] W. S. Noble and A. Ben-Hur. Integrating information for protein function prediction. *Bioinformatics - From Genomes to Therapies.*, 3:1297–1314, 2007.

[Noy and Stuckenschmidt, 2005] N. Noy and H. Stuckenschmidt. Ontology Alignment: An annotated Bibliography. In *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings, 2005.

[Noy, 2004] N.F. Noy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record, Special Issue on Semantic Integration*, 33(4), 2004.

[Ommer and Buhmann, 2007] B. Ommer and J.M. Buhmann. Learning the compositional nature of visual objects. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[Parag and Domingos, 2004] Parag and Pedro Domingos. Multi-relational record linkage. In *Proceedings of the KDD-2004 Workshop on Multi-Relational Data Mining*, Seattle, CA, 2004. ACM Press.

[Pazzani, 1996] M. Pazzani. *Searching for dependencies in Bayesian classifiers. In: Learning from data: Artificial intelligence and statistics.* Springer, 1996.

[Peng *et al.*, 2005] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans Pattern Anal Mach Intell*, 27(8):1226–1238, 2005.

[Perlich and Provost, 2003] Claudia Perlich and Foster Provost. Aggregation-based feature invention and relational concept classes. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 167–176, New York, NY, USA, 2003. ACM.

[Pietra *et al.*, 1997] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

[Piramuthu and Sikora, 2008] S. Piramuthu and R. T. Sikora. Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications*, 36:3401–3406, 2008.

[Punera *et al.*, 2006] K. Punera, S. Rajan, and J. Ghosh. Automatic construction of N-ary tree based taxonomies. *Data Mining Workshops, International Conference on*, pages 75–79, 2006.

[Qian and Sejnowski, 1988] N. Qian and T.J. Sejnowski. Predicting the secondary structure of globular proteins using neural networks models. *J. Molecular Biology*, 202:865–884, 1988.

[Rabiner, 1989] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989.

[Rajan *et al.*, 2005] S. Rajan, K. Punera, and J. Ghosh. A maximum likelihood framework for integrating taxonomies. In *Proceedings of AAAI, Pittsburgh, Pennsylvania, USA*, pages 856–861, 2005.

[Rendell and Seshu, 2007] L. Rendell and R. Seshu. Learning hard concepts through constructive induction: framework and rationale. *Computational Intelligence*, 6:247 – 270, 2007.

[Roweis and Saul, 2000] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding, 2000.

[Salzberg *et al.*, 1999] S. L. Salzberg, M. Pertea, A. L. Delcher, M. J. Gardner, and H. Tettelin. Interpolated markov models for eukaryotic gene finding. *Genomics*, 59:24–31, 1999.

[Segal *et al.*, 2002] E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In *Proc. of NIPS 2001*, pages 913–920, Vancouver, Canada, 2002.

[Shi and Malik, 2000] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Silvescu and Honavar, 2001] Adrian Silvescu and Vasant Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13, 2001.

[Silvescu *et al.*, 2004] A. Silvescu, C. Andorf, D. Dobbs, and V. Honavar. Inter-element dependency models for sequence classification. Technical report, Department of Computer Science, Iowa State University, 2004.

[Silvescu *et al.*, 2009] A. Silvescu, C. Caragea, and V. Honavar. Combining super-structuring and abstraction on sequence classification. In *Proceedings of International Conference on Data Mining*, 2009.

[Silvescu, 2008] A. Silvescu. Structural induction: towards automatic ontology elicitation, 2008.

[Slonim and Tishby, 1999] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems (NIPS-12)*, pages 617–623. MIT Press, 1999.

[Slonim *et al.*, 2001] N Slonim, N. Friedman, and T. Tishby. Agglomerative multivariate information bottleneck. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[Slonim *et al.*, 2002] N. Slonim, N. Friedman, and N. Tishby. Unsupervised document classification using sequential information maximization. In *Proc. of SIGIR*, pages 129–136, 2002.

[Srinivasan and King, 1999] A. Srinivasan and R. D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3:37–57, 1999.

[Stein, 1955] C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, pages 197–206, University of California Press, 1955.

[Sutton and McCallum, 2006] C. Sutton and A. McCallum. *An Introduction to Conditional Random Fields for Relational Learning.* MIT Press, 2006.

[Taskar *et al.*, 2001] B. Taskar, E. Segal, and D. Koller. Probabilistic supervised learning and clustering in relational data. In *Proceedings of IJCAI*, pages 870–876, Seattle, Washington, 2001.

[Taskar *et al.*, 2002] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, Edmonton, Canada, 2002.

[Taskar *et al.*, 2005] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Twenty-Second International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[Taskar, 2004] B. Taskar. Learning structured prediction models: A large margin approach., 2004.

[Tenenbaum *et al.*, 2000] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction, 2000.

[Terribilini *et al.*, 2006] M. Terribilini, J.-H. Lee, C. Yan, R. L. Jernigan, V. Honavar, and D Dobbs. Predicting rna-binding sites from amino acid sequence. *RNA Journal*, In Press, 2006.

[Towfic *et al.*, 2009] F. Towfic, C. Caragea, D. C. Gemperline, D. Dobbs, and V. Honavar. Struct-nb: Predicting protein-rna binding sites using structural features. *International Journal of Data Mining and Bioinformatics*, 2009.

[Uhr and Vossler, 1961] L. Uhr and C. Vossler. A pattern recognition program that generates, evaluates, and adjusts its own operators. In *IRE-AIEE-ACM '61 (Western): Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 555–569, New York, NY, USA, 1961. ACM.

[Uhr, 1972] L. Uhr. Layered "recognition cone" networks that preprocess, classify, and describe. *IEEE Transactions on Computers*, 21:758–768, 1972.

[Valiant, 1984] L.G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27:1134–1142, 1984.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, N.Y., 1998.

[Wang and McCallum, 2005] X. Wang and A. McCallum. A note on topical n-grams. Technical Report UM-CS-2005-071, University of Massachusetts, 2005.

[Yu and Liu, 2004] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, 2004.

[Zhang and Honavar, 2003] J. Zhang and V. Honavar. Learning decision tree classifiers from attribute-value taxonomies and partially specified data. In T. Fawcett and N. Mishra, editors, *Proceedings of the International Conference on Machine Learning*, pages 880–887, Washington, DC, 2003.

[Zhang *et al.*, 2006] 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.