Classifying Scientific Publications Using Abstract Features

Cornelia Caragea[1], Adrian Silvescu[2], Saurabh Kataria[1], Doina Caragea[3], and Prasenjit Mitra[1]
[1]Information Sciences and Technology, Pennsylvania State University
[2]Naviance Inc., Oakland, CA
[3]Computer and Information Sciences, Kansas State University
{ccaragea,skataria,pmitra}@ist.psu.edu
silvescu@gmail.com, dcaragea@ksu.edu

Abstract

With the exponential increase in the number of documents available online, e.g., news articles, weblogs, scientific documents, *effective* and *efficient* classification methods are required in order to deliver the appropriate information to specific users or groups. The performance of document classifiers critically depends, among other things, on the choice of the feature representation. The commonly used "bag of words" representation can result in a large number of features. Feature abstraction helps reduce a classifier input size by learning an abstraction hierarchy over the set of words. A cut through the hierarchy specifies a compressed model, where the nodes on the cut represent *abstract features*. In this paper, we compare feature abstraction with two other methods for dimensionality reduction, i.e., feature selection and Latent Dirichlet Allocation (LDA). Experimental results on two data sets of scientific publications show that classifiers trained using *abstract features* significantly outperform those trained using features that have the highest average mutual information with the class, and those trained using the topic distribution and topic words output by LDA. Furthermore, we propose an approach to automatic identification of a cut in order to trade off the complexity of classifiers against their performance. Our results demonstrate the feasibility of the proposed approach.

Introduction

Recent technological advances as well as the popularity of Web 2.0 applications have resulted in large amounts of online text data, e.g. news articles, weblogs, and scientific documents. These data require *effective* and *efficient* methods for classification to facilitate retrieval of content that is tailored to the interests of specific individuals or groups. Machine learning offers a promising approach to the design of algorithms for training computer programs to accurately classify text data. However, the choice of appropriate features used to encode such data is crucial for the performance and the complexity of the learning algorithms (Valiant 1984). The "bag of words" representation, commonly used for text classification, usually results in high dimensional input spaces. The high dimensionality increases the risk of *overfitting* (poor generalization), especially when model parameters are estimated from small labeled training sets. Reducing the number of features to an appropriate size can help minimize *overfitting* by the learning algorithms.

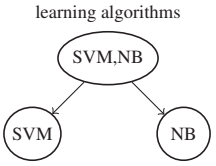

Figure 1: Example of an abstract feature, namely "learning algorithms", which is more general than the features "SVM" (Support Vector Machines) and "NB" (Naïve Bayes). The abstract feature is identified by $\{SVM, NB\}$.

Feature selection methods (Guyon and Elisseeff 2003), (Fleuret 2004), reduce the number of features by selecting a subset of the available features based on some chosen criteria. In particular, feature selection by average mutual information (MI) (Yang and Pederson 1997) selects the top words that have the highest average MI with the class.

Topic models, such as Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) and Probabilistic Latent Semantic Analysis (PLSA) (Hofmann 1999), are unsupervised, generative models of documents, which have been successfully used to perform dimensionality reduction for text classification (Wei and Croft 2009). They are designed to uncover hidden *topics*, i.e., clusters of semantically related words that co-occur in the documents. LDA models each document in a corpus as a mixture of topics (drawn from a conjugate Dirichlet prior), and each topic as a distribution over words in the vocabulary. The topic distribution of a document can be seen as a lower dimensional representation, compared to that in the original input space. The union of words with high probability in each topic can be seen as a set of discriminative features for the documents.

Dimensionality reduction can also be achieved using feature abstraction (Baker and McCallum 1998), (Silvescu, Caragea, and Honavar 2009), (Kang et al. 2005). Feature abstraction methods are data organization techniques, designed to reduce a model input size by grouping "similar" features into clusters of features. Each cluster is identified by an *abstract feature*. An example of an abstract feature is

shown in Figure 1. Feature abstraction effectively reduces the number of parameters of standard model by one to three orders of magnitude without sacrificing classification accuracy (Baker and McCallum 1998), (Silvescu, Caragea, and Honavar 2009). Specifically, it learns an abstraction hierarchy over the set of features using hierarchical agglomerative clustering (Duda, Hart, and Stork 2001), based on the Jensen-Shannon divergence (Lin 1991). A *cut* or level of abstraction through the resulting abstraction hierarchy specifies a compressed model, where the nodes on the cut are used as "features" in the classification model. *Abstraction acts as a regularizer that helps minimize overfitting (through parameter smoothing) when the training set is limited in size*, i.e., it improves the statistical estimates of complex models by reducing the number of parameters, and hence, can yield more accurate models compared to "bag of words" models.

In this paper, we address two main questions. The first question is: how does feature abstraction compare with two other dimensionality reduction methods, i.e., feature selection by average MI and LDA? We compare feature abstraction as described in (Silvescu, Caragea, and Honavar 2009) with feature selection by average MI (Yang and Pederson 1997) and LDA (Blei, Ng, and Jordan 2003) on scientific document classification. Given that both feature abstraction and LDA identify clusters of words in documents, i.e., abstractions and topics, respectively, we analyze the interpretability of the resulting abstractions by directly comparing them with the topics produced by LDA.

The second question that we address is how to identify a "good" cut automatically through an abstraction hierarchy, i.e., a cut which results in simple and accurate classifiers? We extend the work of (Silvescu, Caragea, and Honavar 2009) to address its shortcomings. Specifically, we propose the use of a scoring function to search for a "good" cut, in a greedy, top-down fashion, to trade off the complexity of a classifier against its performance. To penalize complex classifiers, we evaluate two different "information criteria", namely the *Akaike Information Criterion* (Akaike 1974) and the *Bayesian Information Criterion* (Schwarz 1978). Note that the procedure does not necessarily find the optimal cut through the abstraction hierarchy, however, we experimentally show that we obtain a "good" trade-off between the complexity and the accuracy of the classifiers.

The results of our experiments on two data sets of scientific publications show that: (i) feature abstraction can yield significantly more accurate models than those trained using the "bag of words", those trained using features that have the highest average MI with the class, and those trained using the topic distribution and topic words output by LDA (for example, the abstraction-based classifiers show 25%-43% error reduction over the "bag of words" models); (ii) the proposed scoring functions are capable of identifying a "good" cut, which can produce classifiers that use significantly smaller number of features and, at the same time, are more accurate than those using the "bag of words" approach.

Related Work

In this section, we discuss the most relevant works to our study. Information Bottleneck (IB) and its variants (Tishby, Pereira, and Bialek 1999), (Slonim and Tishby 1999), (Pereira, Tishby, and Lee 1993) are distributional clustering based approaches, designed to find a compression of a variable X while preserving as much information as possible about a target (or relevance) variable Y. Baker and McCallum (1998) applied distributional clustering (Pereira, Tishby, and Lee 1993) to reduce the dimensionality of the feature space for document classification tasks. They compared the distributional clustering based approach with the Latent Semantic Indexing (Deerwester et al. 1990), and feature selection (Yang and Pederson 1997), and found that their approach can significantly reduce a model input size for a small decrease in the accuracy of the "bag of words" models. Slonim and Tishby (2001) compared feature abstraction by *bottom-up* agglomerative IB with feature selection on text documents, whereas Bekkerman et al. (2003) compared feature abstraction by *top-down* IB with feature selection on binary and multi-label text classification tasks. Both studies found that using feature abstraction yields significant improvement in classification performance over the "bag of words" models. Unlike the above works, we compare the (*bottom-up*) feature abstraction with LDA on *multi-class* text documents, in addition to feature selection.

Depending upon the application needs, the basic LDA framework (Blei, Ng, and Jordan 2003) has been extended by various research efforts. For example, the "bag of word" assumption in LDA is relaxed to the "bag of n-grams" such that the *topics* are distributions over n-grams (Wang and McCallum 2005). DiscLDA (Lacoste-Julien, Sha, and Jordan 2009) is a variant of LDA aimed at finding useful low-dimensional document representations, but also taking into account class label information in deriving these representations, i.e. DiscLDA is a *supervised* variant of the *unsupervised* LDA. Hierarchical Topic Models (Blei et al. 2004) aim at organizing the hidden topics in the data into a hierarchy. In this study, we compare feature abstraction (under the "bag of words" assumption) with "vanilla" LDA; a comparison among the other methods will be pursued in the future.

desJardins, Getoor, and Koller (2000) explored the use of abstraction-based search to compress conditional probability tables in Bayesian networks. Furthermore, Kang et al. (2005) automatically constructed word taxonomies from text data and used *Minimum Description Length* (Mitchell 1997) to search for a "good" cut through the word taxonomy. For a given cut, the authors explored the expansion of each node on the cut. In contrast, we use Akaike and Bayesian Information Criteria, and explore the cuts in a predefined order, specifically the order in which the nodes were added to the abstraction hierarchy, hence, making the procedure more efficient than that of Kang et al. (2005).

Feature Representations

We describe three dimensionality reduction methods for text classification: one based on feature selection by average mutual information (Yang and Pederson 1997), one based on topic models (Blei, Ng, and Jordan 2003), and another one based on the feature abstraction approach of (Silvescu, Caragea, and Honavar 2009). In what follows, we denote by $\mathcal{D} = \{(\mathbf{w}_l, c_l)\}_{l=1,\cdots,N}$ a data set of documents $\mathbf{w}_l$ and their

associated class labels c_l; by $\mathcal{V}$ the set of words of size n; and by m the desired reduced space dimensionality ($m \ll n$).

Feature Selection

Feature selection is performed by selecting a subset $\mathcal{F}$ of features (i.e., words) from $\mathcal{V}$, $\mathcal{F} \subseteq \mathcal{V}$ such that $|\mathcal{F}| = m$. The features are ranked according to a scoring function f and the top m best ranked features are selected.

As the scoring function, we used the average mutual information (Cover and Thomas 1991) between the class variable, denoted by C, which takes values in the set of the available classes $\mathcal{C}$, and the random variable over the absence or presence of a word w_i in a document, denoted by W_i, which takes values 0 or 1, respectively. The average mutual information is defined as follows:

$$\begin{aligned} I(C, W_i) &= H(C) - H(C|W_i) \qquad (1) \\ &= \sum_{C \in \mathcal{C}} \sum_{W_i \in \{0,1\}} p(C, W_i) \log \frac{p(C, W_i)}{p(C)p(W_i)} \end{aligned}$$

where $H(C)$ is the entropy of the class variable, and $H(C|W_i)$ is the entropy of the class variable conditioned on the absence or presence of word w_i. The probability $p(C, W_i)$ is estimated from counts gathered from $\mathcal{D}$ and $p(C)$ and $p(W_i)$ are obtained by marginalizing $p(C, W_i)$.

The feature selection representation. Let $(\mathbf{w}_l, c_l)$ be an instance in $\mathcal{D}$. The "bag of words" representation of $(\mathbf{w}_l, c_l)$ is given by: $([w_1 : \#w_1^l], \cdots, [w_n : \#w_n^l], c_l)$, where $[w_i : \#w_i^l]$, for $i = 1, \cdots, n$, represents the frequency counts of the word w_i in the document $\mathbf{w}_l$. Given the selected set of features $\mathcal{F} = \{w_{i_1}, \cdots, w_{i_m}\}$, the instance $(\mathbf{w}_l, c_l)$ is transformed into $([w_{i_1} : \#w_{i_1}^l], \cdots, [w_{i_m} : \#w_{i_m}^l], c_l)$.

Topic Models

Latent Dirichlet Allocation (LDA) models each document in a corpus as a mixture of topics, and each topic as a distribution over words in the vocabulary (Blei, Ng, and Jordan 2003). Specifically, the generative process for a document d is as follows: (i) sample a distribution over topics θ from a Dirichlet(α) prior, $\theta \sim Dir(\alpha)$; (ii) for each word w in d, choose a topic z according to θ, $z \sim Multinomial(\theta)$; (iii) sample a word w from a multinomial distribution over words ϕ, corresponding to the chosen topic z, $w \sim Multinomial(\phi(z))$.

The topic distribution representation. Let the number of topics be set to m. An instance $(\mathbf{w}_l, c_l)$ is transformed into $(\theta_1^l, \cdots, \theta_m^l, c_l)$, where θ_i^l is the probability of topic i in the document $\mathbf{w}_l$.

The topical words representation. The topics found by LDA correspond to clusters of semantically related words that occur together in the collection of documents. The union of words with high probability in each topic can be seen as a set of discriminative features for the documents. Let $\mathcal{W} = \{w_{i_1}, \cdots, w_{i_k}\}$ be the set of words with high probability in each of the m topics. Then, the instance $(\mathbf{w}_l, c_l)$ is transformed into $([w_{i_1} : \#w_{i_1}^l], \cdots, [w_{i_k} : \#w_{i_k}^l], c_l)$.

Feature Abstraction

Feature abstraction reduces a classifier input size by finding clusters of "similar" features, called *abstract features*.

Constructing abstract features. The algorithm for constructing abstract features (or abstractions) is based on hierarchical agglomerative clustering (Duda, Hart, and Stork 2001). The *input* is a data set $\mathcal{D} = \{(\mathbf{w}_l, c_l)\}_{l=1,\cdots,N}$ of text documents $\mathbf{w}_l$ and their class labels c_l. The *output* is an abstraction hierarchy $\mathcal{T}$ over the set of words $\mathcal{V}$ of size n, extracted from $\mathcal{D}$. An *abstraction hierarchy* (AH) over $\mathcal{V}$ is defined as a rooted tree such that the leaf nodes correspond to the words in $\mathcal{V}$ and the internal nodes correspond to abstractions or clusters of words. An m-cut γ_m through $\mathcal{T}$ is a set of m nodes, which forms a partition of $\mathcal{V}$, i.e., $\gamma_m = \{a_1 : \mathcal{V}_1, \cdots, a_m : \mathcal{V}_m\}$, where a_i denotes the i^{th} abstraction and $\mathcal{V}_i$ denotes the subset of words that are clustered together into a_i based on some similarity measure.

The algorithm initializes each abstraction a_i with a word w_i in $\mathcal{V}$, and recursively merges pairs of abstractions that are most "similar" to each other. The AH $\mathcal{T}$, which is stored in a last-in-first-out (LIFO) stack, is returned after $n - 1$ steps. Note that $\mathcal{T}$ is constructed in a bottom-up fashion.

Two abstractions are considered to be "similar" if they occur within similar class contexts. The *class context of a word* w_i in $\mathcal{V}$ is defined as the conditional probability distribution $p(C|w_i)$ of the class variable C given the word w_i, which is estimated from $\mathcal{D}$ as follows:

$$\hat{p}(C|w_i) = \left[\frac{\#[w_i, c_j]}{\sum_{c_j \in \mathcal{C}} \#[w_i, c_j]} \right]_{c_j \in \mathcal{C}}, \qquad (2)$$

where $\#[w_i, c_j]$ represents the number of times w_i co-occurs with c_j in $\mathcal{D}$. Furthermore, the *class context of an abstraction* $a = \{w_1, \cdots, w_{|a|}\}$, i.e., $p(C|a)$, is obtained using a weighted aggregation of the contexts of its constituent words. That is,

$$\hat{p}(C|a) = \sum_{t=1}^{|a|} \frac{\#w_t}{\sum_{t=1}^{|a|} \#w_t} \cdot \hat{p}(C|w_t). \qquad (3)$$

The distance between two abstractions a_i and a_j, denoted by $d(a_i, a_j)$, is defined as the weighted Jensen-Shannon divergence (Lin 1991) between their class contexts:

$$d(a_i, a_j) = (p(a_i) + p(a_j)) JS_{\pi_i, \pi_j}(p(C|a_i), p(C|a_j)), \quad (4)$$

where $\pi_i = \frac{p(a_i)}{p(a_i)+p(a_j)}$ and $\pi_j = \frac{p(a_j)}{p(a_i)+p(a_j)}$. The abstractions a_i and a_j with the smallest distance between their contexts are merged into a_k, at each step (π_i and π_j represent the prior probabilities of a_i and a_j in a_k).

Let A be a random variable that takes values in the set of abstractions γ_m. Silvescu, Caragea, and Honavar (2009) have shown that the reduction in mutual information between A and C due to the merger $\{a_i, a_j\} \rightarrow a_k$ is given by $d(a_i, a_j)$. The choice of the distance d induces an ordering over the cuts in $\mathcal{T}$, with the smallest reduction in mutual information from one cut, γ_m, to another, γ_{m-1}, where γ_{m-1} differs from γ_m only by the nodes a_i and a_j which are "abstracted" into a_k (see Figure 2). For a given m, the cut γ_m

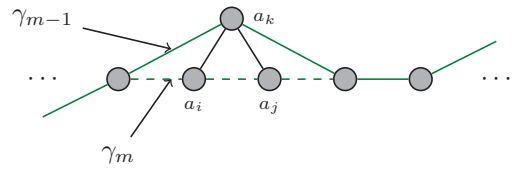

Figure 2: Example of two consecutive cuts, γ_m and γ_{m-1}. For any choice of m, the cut γ_{m-1} differs from γ_m only by the nodes a_i and a_j, which are "abstracted" into a_k. The rest of the nodes are the same on both γ_{m-1} and γ_m.

Algorithm 1 Identifying a "good" cut

Input: The training set $\mathcal{D}$, an abstraction hierarchy $\mathcal{T}$, a hypothesis class $\mathcal{H}$, and a scoring function f.
Output: A cut γ_m.
$i = 1$
$\gamma = \gamma_i$ // Initialize γ to the root of $\mathcal{T}$.
Train $h^{(i)}$ on $\mathcal{D}_{\gamma_i}$
repeat
  $i = i + 1$
  $\gamma = \gamma_i$ // Top-down search in $\mathcal{T}$.
  Train $h^{(i)}$ on $\mathcal{D}_{\gamma_i}$
until $f(h^{(i)}|\mathcal{D}) < f(h^{(i-1)}|\mathcal{D})$
return γ_{i-1} // $m = i - 1$

is *uniquely* obtained by removing $m-1$ elements from the top of the last-in-first-out (LIFO) stack. Hence, the number of cuts considered is linear in the number n of words in the vocabulary, ranging from γ_1, the root of $\mathcal{T}$, to γ_n, the leaves of $\mathcal{T}$, while γ_m, $1 < m < n$, is obtained by removing the $m-1$ nodes that were added last to $\mathcal{T}$.

The abstract feature representation. For a given choice of m such that $m \ll n$, let $\gamma_m = \{a_1 : \mathcal{V}_1, \cdots, a_m : \mathcal{V}_m\}$ be the corresponding set of abstractions. Then $(\mathbf{w}_l, c_l)$ can be represented, in a transformed data set $\mathcal{D}_{\gamma_m}$, by using m abstract features as follows: $(\mathbf{w}_l, c_l) = ([a_1 : \#a_1^l], \cdots, [a_m : \#a_m^l]\}, c_l)$, where $\#a_i^l = \sum_{w_q \in \mathcal{V}_i} \#w_q^l$, $\forall 1 \leq i \leq m$. Note that this corresponds to "abstracting out" the difference between the words w_q, which are clustered together into a_i.

Next we present our extension of the abstraction-based approach of (Silvescu, Caragea, and Honavar 2009) to identify a "good" cut through the learned abstraction hierarchy.

Identifying a "Good" Cut

Feature abstraction allows reducing dimensionality and, at the same time, could potentially improve the statistical estimates of complex models by reducing the number of parameters that need to be estimated from the data, hence minimizing the risk of *overfitting* through parameter smoothing (Baker and McCallum 1998), (Silvescu, Caragea, and Honavar 2009). However, while abstraction provides simpler models compared to the standard ones, the simplicity is achieved at the risk of some information loss, which prevents the best-possible classification. To trade off the complexity of a model against its classification accuracy, we propose the use of a scoring function to guide a top-down search for a "good" cut through an AH $\mathcal{T}$, i.e., a cut which results in simple and accurate classifiers.

The procedure for identifying a "good" cut through $\mathcal{T}$ is shown in Algorithm 1 and is performed in a top-down fashion. Given a training set $\mathcal{D}$, an AH $\mathcal{T}$, a hypothesis class $\mathcal{H}$, and a scoring function f, the algorithm initializes the cut γ_1 to the root of $\mathcal{T}$. The set $\mathcal{D}$ is then transformed into $\mathcal{D}_{\gamma_1}$, in which instances in $\mathcal{D}$ are encoded using the abstractions on γ_1, and a classifier $h^{(1)}$ is learned on $\mathcal{D}_{\gamma_1}$. The algorithm iterates through the cuts γ_i of $\mathcal{T}$, in order, from γ_2 towards γ_n, until the value of the scoring function corresponding to the current classifier, $f(h^{(i)}|\mathcal{D})$, becomes smaller than that of the previous classifier, $f(h^{(i-1)}|\mathcal{D})$.

Next we formally introduce our scoring function to perform model selection. Given the logarithm of the posterior distribution $\log p(h^{(i)}|\mathcal{D})$, as follows:

$$\log p(h^{(i)}|\mathcal{D}) \propto \log p(\mathcal{D}|h^{(i)}) + \log p(h^{(i)}), \tag{5}$$

the uncertainty in the choice of a model is expressed in terms of the prior probability distribution $p(h^{(i)})$. When $p(h^{(i)})$ is not the uniform distribution, Eq. 5 is referred to as a *penalized log-likelihood* (Jordan 2003). The addition of the penalty term $\log p(h^{(i)})$ compensates for the effect of *overfitting* of complex models, and is used to improve the parameter estimates in settings where the available training labeled data are small (Jordan 2003).

We evaluate two "information criteria", which differ from each other by the penalty term $\log p(h^{(i)})$. Specifically, we use the *Akaike Information Criterion, or AIC* (Akaike 1974) and the *Bayesian Information Criterion, or BIC* (Schwarz 1978). AIC and BIC are given as follows (Bishop 2006):

$$f(h^{(i)}|\mathcal{D}) = \log p(\mathcal{D}|h^{(i)}) - M,$$

and

$$f(h^{(i)}|\mathcal{D}) = \log p(\mathcal{D}|h^{(i)}) - \frac{1}{2} M \log N,$$

respectively, where $\log p(\mathcal{D}|h^{(i)})$ is the conditional log-likelihood of the data given a hypothesis $h^{(i)}$, M is the number of model parameters, N is the number of examples in $\mathcal{D}$. BIC penalizes model complexity more stringently than AIC.

To perform model selection, we followed the procedure described in (Bishop 2006) (section 3.4). Thus, the models are compared directly on the training data, with no requirement for a validation set, which is especially useful in small sample settings.

Algorithm Analysis. As mentioned above, the total number of cuts that are considered is n. Hence, the computational complexity of Algorithm 1, in the worst case, is $O(nt)$, where t is the time taken to train a classifier. The time t is dependent on the type of the classifier used.

Experiments and Results

We compared the abstraction-based approach with feature selection by average mutual information and LDA and evaluated the quality of the scoring function used to identify a "good" cut through an abstraction hierarchy on two data sets

of scientific publications: Cora, a standard benchmark data set of research articles (McCallum et al. 2000) and the CiteSeer digital library (Giles, Bollacker, and Lawrence 1998).

For both data sets, we used the same subsets[1] of research articles as in (Lu and Getoor 2003). However, we did not use the links between the articles. **Cora** consists of 3191 machine learning articles that are found on the Web, with each article being categorized into one of seven classes. **CiteSeer** consists of a labeled subset (3186 publications) of the CiteSeer digital library, with each publication being categorized into one of six classes.

We followed the same preprocessing procedure as in (Lu and Getoor 2003) to prune the vocabulary sizes. Specifically, we performed stemming and removed stop words and words with document frequency less than 10 in each collection of articles. The vocabulary sizes of Cora and CiteSeer, denoted by n, are 1864 and 1908, respectively.

Experimental Design

Our experiments are designed to explore the following questions: (i) How effective are *abstract features*, i.e., clusters of words, compared with features, i.e., words that have the highest average mutual information with the class variable, and topics and topic words, output by LDA (Blei, Ng, and Jordan 2003), in generating a correct classification of scientific publications? (ii) How effective is *abstraction* in minimizing *overfitting* through parameter smoothing? and (iii) How good are our proposed scoring functions in identifying a "good" cut through the abstraction hierarchy $\mathcal{T}$ associated with the training data $\mathcal{D}$?

To answer these questions, we compared the performance of classifiers trained on the Cora and the CiteSeer data sets using abstract features with that of their corresponding counterparts trained using a "bag of words" representation, words with the highest average mutual information with the class variable, the topic distribution obtained by using LDA[2], and the top 20 words extracted from each topic of a set of topics by using LDA (the feature representations are detailed below). Furthermore, we evaluated both the Akaike and the Bayesian Information Criteria in identifying a "good" cut.

The feature representations for each article are as follows:

- a "bag of words" representation, i.e., all words in the vocabulary (BoW);
- a bag of m words chosen using feature selection by average mutual information (FS);
- a bag of m abstractions over all words in the vocabulary (FA) (Silvescu, Caragea, and Honavar 2009);
- the distribution of m topics, obtained by using LDA (TD) (Blei, Ng, and Jordan 2003);
- a bag of topic words output by LDA as the top 20 words for each of the m topics above (TW) (Blei, Ng, and Jordan 2003).

In our experiments, we used Support Vector Machine (SVM) and Logistic Regression (LR)[3] classifiers, and report the average classification accuracy obtained in a 5-fold cross-validation experiment. Note that LR was also used by Lu and Getoor (2003) to classify articles into categories. We performed a paired t test to check the statistical significance of our results (Dietterich 1998).

Results

Figures 3(a) and 3(b) show the results of the comparison of FA with BoW, FS, and TD on Cora and CiteSeer, using SVM (top) and LR (bottom), respectively. For a choice of the number of abstractions m, the same number of features is chosen for FS and the same number of topics is extracted by LDA for TD. Figures 3(c) and 3(d) show the results of the comparison of FA with BoW, FS, and TW, on both data sets, using SVM (top) and LR (bottom), respectively. Here, the number of topics m is chosen first and for each topic, the top 20 words are extracted. Let l denote the number of words after removing duplicates. These l words are used by TW. Correspondingly, l abstractions are chosen for FA, and l features are selected for FS. The goal of this experiment is to show how the classifiers' performance varies while varying the number of abstractions, from 1 to n. Note that results for TD and TW are shown on different plots, as the number of features varies from one another.

Comparison of FA with BoW. For both data sets, the performance of SVM and LR classifiers trained on BoW is matched by that of their counterparts trained on FA using only a small number of abstractions, e.g., $m = 6$ on Cora using SVM (see Figure 3). Both SVM and LR trained on FA significantly outperform those trained on BoW for a wide range of values of m ($p < 0.05$), on both data sets. For example, on Cora, with $m = 150$ abstractions, SVM achieves an accuracy of 79.88% using FA as compared to 64.86% achieved by SVM using BoW, which gives 43% reduction in classification error. Hence, abstraction can help minimize *overfitting*.

Comparison of FA with FS. As can be seen from Figure 3, FA significantly outperforms FS for most of the choices of the number of features, on both data sets ($p < 0.05$). For example, on Cora, with only $m = 100$ features, SVM using FA achieves an accuracy of 79.53% whereas SVM using FS achieves an accuracy of 68.53%, which gives 35% reduction in classification error. Moreover, the performance of FS is significantly better than that of TD ($p < 0.05$) for any choice of the number of features for both data sets (see Figures 3(a) and 3(b)), and worse than that of TW for many choices of the number of features again for both data sets (see Figures 3(c) and 3(d)).

Comparison of FA with TD and TW. Figures 3(a) and 3(b) show that, for any choice of the number of features, both SVM and LR trained using FA significantly outperform

[1] Available at http://www.cs.umd.edu/projects/linqs/projects/lbc/.

[2] We have used the MALLET implementation of LDA. The hyperparameters α and β are set to $50/T$ and 0.01, respectively, based on previous research (Steyvers and Griffiths 2006), which found that these values work well with many text collections (T represents the number of topics). We used LDA to extract the topic distribution on the training set and to output a model, and then used the model to infer the topic distribution on the test set.

[3] We have used the WEKA implementation of LR and SVM with the default parameters.

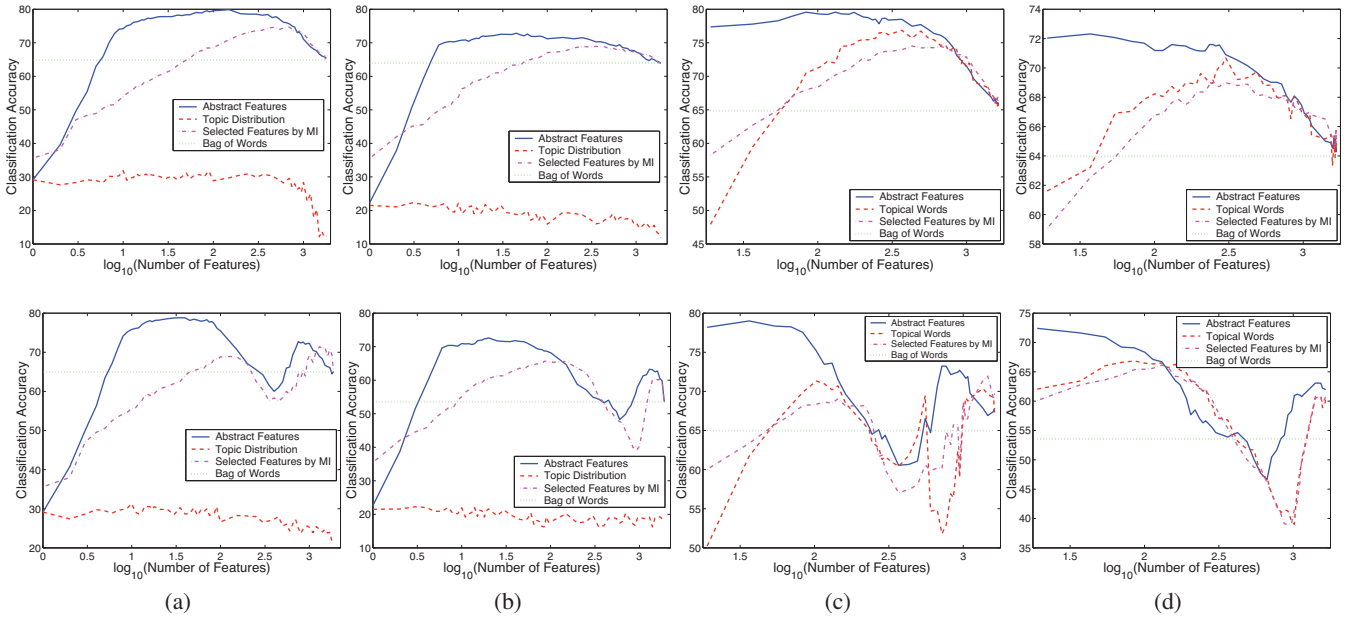

Figure 3: Comparison of feature abstraction with the "bag of words", selected features by mutual information (MI), and topic distribution on: (a) Cora and (b) CiteSeer, using Support Vector Machine (top) and Logistic Regression (bottom), respectively. Comparison of feature abstraction with the "bag of words", selected features by MI, and topic words on: (c) Cora and (d) CiteSeer, using Support Vector Machine (top) and Logistic Regression (bottom), respectively.

those trained using TD for the same number of features ($p < 0.05$), and in fact, the classifiers' performance using TD is very poor, on both Cora and CiteSeer. One reason for this may be that the articles in the data sets are very short in length, i.e., they contain only title and abstract, and for some articles, the content consists only of a few words. Hence, the resulting topic distribution is not discriminative for the class, especially for large number of topics. However, the performance of classifiers trained using TW is better than that of classifiers trained using TD (as shown in Figure 3).

As can be seen in Figures 3(c) and 3(d), both classifiers trained on FA significantly outperform those trained on TW ($p < 0.05$) when the number of features is small. As the number of features increases, the performance of classifiers using TW is similar or slightly better than that of classifiers using FA (however, the difference is not statistically significant). When the number of topics is relatively small, the corresponding topic words result in good classification performance, and as we increase the number of topics, the performance of TW-based classifiers decreases.

FA achieves the highest overall accuracy, e.g., on Cora (SVM), FA achieves near 80% with 150 abstractions, while the next is near 77% achieved by TW with 492 topic words.

Identifying a "good" cut. Table 1 shows the number of iterations m taken by Algorithm 1 to automatically identify a "good" cut through the abstraction hierarchy, learned on the training set, using scoring functions based on Akaike (AIC) and Bayesian Information Criteria (BIC). Note that m represents the size of the identified cut γ_m as well. The table also shows the classification accuracy (shown as a percentage) of SVM (top) and LR (bottom) trained using as features the m abstractions on the identified cut γ_m, on both Cora and CiteSeer, respectively. The results are shown for each of the 5 folds of cross-validation as the size of the chosen cut differs from one fold to another. The "Average" column shows the average cut size and average accuracy after the 5 runs of cross-validation.

As can be seen in the table, the algorithm converges very fast, e.g., on average, after 20 iterations (out of 1908) on the CiteSeer dataset, using BIC. As expected, BIC (highlighted in bold) results in faster convergence compared to AIC and, hence, produces simpler models than AIC, without sacrificing classification performance. The last column of Table 1 (i.e., FA) corresponds to the peaks in Figure 3 using FA, and represents the "best" accuracies observed on the test set in 5-fold cross-validation experiments (when the number of abstractions m is varied from 1 to n).

Again as can be seen in the table, the average accuracy (after the 5 runs of cross-validation), generally is *slightly* worse than the "best" accuracy observed on the test set when the number of abstractions m is varied (i.e., the values in the last column). This provides evidence that the proposed scoring functions help identify a "good" cut (or a "good" trade-off between the complexity and the accuracy of the classifiers), without the need to iterate through the entire abstraction hierarchy (i.e., 1864 and 1908 iterations corresponding to the total number of cuts considered on Cora and CiteSeer, respectively). BIC penalizes more stringently the model complexity than AIC, as shown by the smaller number of abstractions m used by BIC compared to AIC. Note that the "best" accuracy reported in the last column of the table is obtained by using the same number of abstractions on

Data Set		Fold 1		Fold 2		Fold 3		Fold 4		Fold 5		Average		FA	
		m	Acc	m	Acc	m	Acc	m	Acc	m	Acc	m	Acc	m	"Best" Acc
CiteSeer	(AIC)	30	73.66	27	68.13	26	72.84	32	75.66	57	75.35	34	73.13	45	72.85
	(BIC)	**20**	**70.21**	**16**	**66.71**	**14**	**72.68**	**31**	**75.66**	**18**	**74.56**	**20**	**71.97**		
Cora	(AIC)	46	79.34	19	77.27	20	76.64	31	78.52	29	76.95	29	77.74	150	79.88
	(BIC)	**23**	**79.18**	**16**	**76.01**	**20**	**76.64**	**31**	**78.52**	**25**	**77.74**	**23**	**77.62**		
CiteSeer	(AIC)	84	68.65	27	67.50	65	68.91	89	71.58	66	72.84	66	69.89	20	72.63
	(BIC)	**24**	**73.04**	**18**	**68.91**	**14**	**72.52**	**32**	**74.25**	**35**	**73.15**	**25**	**72.37**		
Cora	(AIC)	36	80.28	46	78.68	39	78.68	31	78.36	49	77.58	40	78.72	35	78.81
	(BIC)	**23**	**79.18**	**16**	**76.80**	**39**	**78.68**	**31**	**78.36**	**29**	**77.89**	**28**	**78.18**		

Table 1: The number of iterations m taken by Algorithm 1 to automatically identify a "good" cut through an abstraction hierarchy using scoring functions based on Akaike (AIC) and Bayesian Information Criteria (BIC) as well as the classification accuracy (shown as a percentage) of SVM (top) and LR (bottom) trained using, as features, the m abstractions on the identified cut, on Cora and CiteSeer, respectively. The results are shown for each of the 5 folds of cross-validation, along with their averages. As expected, BIC (highlighted in bold) results in faster convergence compared to AIC and, hence, produces simpler models compared to AIC, without sacrificing classification performance. The last column (i.e., FA) represents the "best" accuracies observed on the test set in 5-fold cross-validation experiments (when the number of abstractions m is varied), which correspond to the peaks in Figure 3 using FA.

Topic 34	Abstraction 24	Topic 15	Abstraction 28	Topic 21	Abstraction 48
scheme	evolutionary	parameter	**estimation**	**search**	solver
crossover	population	regression	hmm	problem	climbing
diverse	**recombination**	**likelihood**	**parametric**	heuristic	strategy
number	coevolutionary	maximum	**likelihood**	algorithm	hillclimbing
parent	**landscape**	density	prior	instance	error
common	phenotype	**parametric**	probabilistic	good	**greedy**
point	chromosome	**asymptotic**	**nonparametric**	space	optimum
mutation	**mutation**	statistic	mixture	collect	emerge
characteristic	coevolution	observable	boltzmann	**greedy**	hill
recombination	genotype	**estimation**	dirichlet	exhaustive	**search**
competition	evolve	confidence	**asymptotic**	peak	engine
landscape	evolution	**nonparametric**	exponential	examine	discovery
previous	dilemma	interval	assumption	emploit	restriction
classic	prey	smooth	probable	suitable	opponent
preserve	optimal	kernel	hazard	augment	body

Table 2: Comparison of three topics with three abstractions (out of 100) on Cora. The topics and the abstractions are represented by their top 15 words. The words that are colored and highlighted in bold, appear in both the LDA topic and the FA abstraction.

each fold, which explains why it can be lower than the accuracy obtained when m is estimated using scoring functions (e.g., on CiteSeer using SVM and AIC).

Furthermore, compared to the classifiers obtained using the "bag of words" approach, the proposed scoring functions are capable of identifying a "good" cut, which produces classifiers that use significantly smaller number of features and, at the same time, are more accurate. For example, on Cora using SVM, BIC finds (on average) 23 nodes (after 23 iterations) which yield an accuracy of 77.62%, compared to 64.86% achieved by BoW using 1864 features. Lu and Getoor (2003) reported accuracies of 67.4% and 60.7% on Cora and CiteSeer, respectively, (however, in 3-fold cross-validation experiments) using a regularized LR on BoW, which are outperformed by our LR on FA using both AIC and BIC scoring functions (see Table 1).

Analysis of Abstract Features. As both feature abstraction and LDA identify clusters of words in documents, i.e., abstraction and topics, respectively, in Table 2, we show the direct comparison of three *topics* (out of 100) extracted by LDA with three *abstract features* (or abstractions) constructed by feature abstraction, on Cora. We randomly chose the abstractions and identified the LDA topics that roughly match them. As can be seen in the table, abstract features correspond to clusters of correlated words, which can be as interpretable as the LDA topics. In the table, we highlighted the words that overlap in an abstract feature and the corresponding topic.

Conclusion

In this paper, we presented an application of feature abstraction to text classification for simplifying the complexity of the learned models and improving their classification performance. We compared feature abstraction with feature selection by average mutual information and with Latent Dirichlet Allocation. Feature abstraction involves learning an ab-

straction hierarchy over the words in the vocabulary, i.e., finding clusters of words that are most "similar" to each other, measured by the similarity of the class distributions that the words induce. We further proposed scoring functions based on two different information criteria to automatically identify a "good" cut through the resulting abstraction hierarchy. The identified cut provides a "good" trade-off between the complexity and the accuracy of the classifiers.

Our results have shown that: abstract features can produce higher-performance models compared to the "bag of words", top-ranked features by mutual information with the class variable, and the topic distribution and topic words output by LDA; the proposed scoring functions are able to identify a "good" cut, which results in simple and accurate classifiers. In future work, it would be interesting to investigate how to combine link features (e.g., corresponding to the "cite" link), abstract them, and combine them with abstract features to improve classification performance. It would also be interesting to compare feature abstraction with other LDA variants such as DiscLDA.

References

Akaike, H. 1974. A new look at statistical model identification. *IEEE Transactions on Automatic Control* 19:716–723.

Baker, L. D., and McCallum, A. K. 1998. Distributional clustering of words for text classification. In *Proc. of ACM SIGIR*, 96–103. ACM Press.

Bekkerman, R.; El-Yaniv, R.; Tishby, N.; and Winter, Y. 2003. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Res.* 3:1183–1208.

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

Blei, D. M.; Griffiths, T. L.; Jordan, M. I.; and Tenenbaum, J. B. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Proc. of NIPS*. MIT Press.

Blei, D.; Ng, A.; and Jordan, M. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

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

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Soc. for Inf. Science* 41(6):391–407.

desJardins, M.; Getoor, L.; and Koller, D. 2000. Using feature hierarchies in bayesian network learning. In *Proc. of SARA '02*, 260–270.

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

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Fleuret, F. 2004. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.* 5:1531–1555.

Giles, C. L.; Bollacker, K. D.; and Lawrence, S. 1998. Citeseer: an automatic citation indexing system. In *International Conference on Digital Libraries*, 89–98. ACM Press.

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

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*.

Jordan, M. I. 2003. *An Introduction to Probabilistic Graphical Models*.

Kang, D.-K.; Zhang, J.; Silvescu, A.; and Honavar, V. 2005. Multinomial event model based abstraction for sequence and text classification. In *Proc. of SARA '05*.

Lacoste-Julien, S.; Sha, F.; and Jordan, M. I. 2009. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Proc. of NIPS*, volume 21.

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

Lu, Q., and Getoor, L. 2003. Link-based classification. In *Proc. of ICML '03*.

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

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

Pereira, F.; Tishby, N.; and Lee, L. 1993. Distributional clustering of english words. In *Proc. of ACL*, 183–190.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6:461–464.

Silvescu, A.; Caragea, C.; and Honavar, V. 2009. Combining super-structuring and abstraction on sequence classification. In *Proc. of ICDM '09*, 986–991.

Slonim, N., and Tishby, N. 1999. Agglomerative information bottleneck. In *Proc. of NIPS-12*, 617–623. MIT Press.

Slonim, N., and Tishby, N. 2001. The power of word clusters for text classification. In *European Colloquium on IR Res.*

Steyvers, M., and Griffiths, T. L. 2006. Probabilistic topic models.

Tishby, N.; Pereira, F. C.; and Bialek, W. 1999. The information bottleneck method. In *Invited paper to The 37th Conf. on Communication, Control, and Computing*, 368–377.

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

Wang, X., and McCallum, A. 2005. A note on topical n-grams. Technical report, University of Massachusetts.

Wei, X., and Croft, W. 2009. Lda-based document models for ad-hoc retrieval. In *Proc. of ACM SIGIR*, 178–185.

Yang, Y., and Pederson, J. O. 1997. Feature selection in statistical learning of text categorization. In *Proc. of ICML*, 412–420.