DIMACS Technical Report 2009-16
June 2009

Leveraging Higher Order Dependencies Between Features for Text Classification

by

Murat C. Ganiz
Department of Computer Science
Lehigh University
DIMACS
Rutgers University

Nikita I. Lytkin
Department of Computer Science
Rutgers University

William M. Pottenger
DIMACS and Department of Computer Science
Rutgers University

ABSTRACT

Traditional machine learning methods only consider relationships between feature values within individual data instances while disregarding the dependencies that link features across instances. In this work, we develop a general approach to supervised learning by leveraging higher-order dependencies between features. We introduce a novel Bayesian framework for classification named Higher Order Naive Bayes (HONB). Unlike approaches that assume data instances are independent, HONB leverages co-occurrence relations between feature values across different instances. Additionally, we generalize our framework by developing a novel data-driven space transformation that allows any classifier operating in vector spaces to take advantage of these higher-order co-occurrence relations. We also develop a $O\left((m+n)n^2\right)$ time algorithm for obtaining the counts of higher-order paths in boolean data. This algorithm improves over the $O\left(m^2n^3\right)$ complexity of a straight-forward path counting algorithm. Results obtained on several benchmark text corpora demonstrate that higher-order approaches achieve significant improvements in classification accuracy over the baseline (first-order) methods.

Key words: machine learning, text classification, higher order learning, statistical relational learning, higher order naive bayes, higher order support vector machine

1 Introduction

A well known problem in real-world applications of machine learning methods is that expert labeling of large amounts of data for training a classifier is prohibitively expensive. Often in practice, only a small amount of labeled data is available for training. Traditional methods of classification treat individual data instances independently. In this case, however, a small training set makes an adequate estimation of the model parameters of a classifier very challenging. Prior work has demonstrated the value of leveraging explicit [1, 12, 18] as well as implicit [4, 7] link information within data in order to provide a richer data representation for model estimation.

In Sect. 4.1, we build on a graph-based data representation from our prior work [4] and introduce a novel Bayesian framework for classification named Higher Order Naive Bayes (HONB). Unlike approaches that assume data instances are independent, HONB leverages co-occurrence relations between feature values across different instances. We term these implicit co-occurrence relations higher-order paths. Features (e.g., words in documents of a text collection) are richly connected by such higher-order paths, and a model built by HONB exploits this rich connectivity.

We further generalize our framework in Sect. 4.2 by developing a novel data-driven space transformation that allows any classifier operating in vector spaces to take advantage of relational dependencies captured by higher-order paths between features.

In Sect. 4.3, we develop a $O\left((m+n)n^2\right)$ time algorithm for obtaining the counts of higher-order paths in boolean data. This algorithm improves over the $O\left(m^2n^3\right)$ complexity of a straight-forward path counting algorithm also given in Sect. 4.3.

We evaluate the proposed methods[1] on several benchmark text corpora across a wide range of training set sizes in Sect. 5. In that section, we also draw comparisons with the results reported in [17], where a word clustering approach was proposed for dealing with small and sparse training data for text classification.

The paper is organized as follows. Related work is discussed in Sect. 2. In Sect. 3, we provide the necessary background for development of the proposed methods described in Sect. 4. Experimental results are presented in Sect. 5. A discussion of the effects of leveraging higher-order dependencies for text classification is provided in Sect. 6. Concluding remarks are made in Sect. 7.

2 Related Work

Our motivation for using higher-order dependencies for classification stems from advances in the areas of link mining [5] and information retrieval. In addition to (or sometimes instead of) using the more traditional data representation by feature vectors characterizing each data instance independently of the others, link-based approaches [18, 11, 13] to collective

[1] "Systems and Methods for Data Transformation for Supervised Machine Learning," M.C. Ganiz, N.I. Lytkin and W.M. Pottenger, U.S. Patent Pending 61/185255

classification leverage explicit dependencies, or links, within networked data [13]. Several studies [1, 12, 18] have shown that collective classification can achieve significant reductions in classification errors by performing inferences about multiple data instances simultaneously. However, such methods are context-dependent and are therefore not designed to classify single data instances. This restriction limits the domain of applicability of link-based classifiers.

In this work, we propose classification methods that leverage higher-order dependencies in the form of implicit links between instances and features of the training data. Unlike collective classifiers, methods presented in this work maintain the ability to classify single data instances without requiring any additional context information.

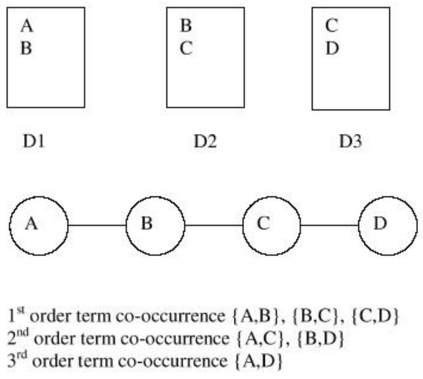

A
B
B
C
C
D
D1
D2
D3
A
B
C
D
1st order term co-occurrence {A,B}, {B,C}, {C,D}
2nd order term co-occurrence {A,C}, {B,D}
3rd order term co-occurrence {A,D}

Figure 1: Higher-order co-occurrences [7]

In [7], we gave a mathematical proof supported by empirical results of the dependence of Latent Semantic Indexing (LSI) [2], a technique often used in text mining and information retrieval, on higher-order relations and in particular on higher-order term co-occurrences expressed via higher-order paths. A higher-order path is exemplified in Fig. 1 (reproduced from [7]) in the context of text data. Figure 1 depicts three documents, D1, D2 and D3, each containing two terms represented by the letters A, B, C and D. Shown below the three documents in Fig. 1 is a higher-order path that links term A with term D through B and C. This path contains three edges and is therefore referred to as a third-order path.

Higher-order relations play an important role in many other systems for text mining and information retrieval. In [10], higher-order associations were used for rule mining in textual data. Higher-order co-occurrences were used in [3] for solving a component of the problem of lexical choice, which identifies synonyms in a given context. In another effort, [21] used second-order co-occurrences for improving the runtime performance of LSI. Higher-order co-occurrences have also been used in other applications including word sense disambiguation [15] and stemming [20].

It should be noted that our framework extends beyond the textual domain. In other words, instances D1, D2 and D3 from Fig. 1 need not be text documents – they may be records in a database, or instances in a labeled training dataset. Likewise, features A, B, C, etc. need not be terms – they may be values in a database record, or feature-value pairs in a data instance.

In fact, in [4] we have successfully used higher-order paths for capturing dependencies amongst data instances when modeling time series data for anomaly detection in the Border Gateway Protocol, which constitutes the backbone of the Internet's routing infrastructure. The approach presented in this work builds on the data representation introduced in our prior work [4] and also described in Sect. 3.2.

3 Background

In this section we review parts of the Bayesian learning theory (Sect. 3.1) and the data representation (Sect. 3.2) underlying the development of approaches presented in Sect. 4. Although the methods discussed in this work are not limited to a particular application domain, here we restrict our attention to textual data. We assume the input space to be an n-dimensional binary vector space where each document is represented by a vector whose non-zero coordinates correspond to terms present in the document.

Formally, let $W = \{w_1, \ldots, w_n\}$ denote the set of binary features that correspond to terms in the vocabulary. Any document d can therefore be represented by an n-dimensional binary vector $w(d) = (w_1(d), \ldots, w_n(d))$, where

$$w_i(d) = \begin{cases} 1, & \text{if document } d \text{ contains term } w_i \\ 0, & \text{otherwise.} \end{cases}$$

For convenience of further exposition, we will simply write d to denote the corresponding vector $w(d)$.

3.1 Bayesian Learning Theory

Given a document d and two classes c_1 and c_2, the Bayes discriminant function can be written as

$$f(d) = \log \frac{P(c_1|d)}{P(c_2|d)} = \log \frac{P(d|c_1)P(c_1)}{P(d|c_2)P(c_2)}, \tag{1}$$

where

$$P(d|c_j) = P(w_1(d), \ldots, w_n(d)|c_j),\ \forall j \in \{1, 2\}, \tag{2}$$

is the conditional likelihood of document d belonging to class c_j, and $P(c_j)$ is the prior probability of class c_j.

By assumption of mutual independence of terms given a class, the conditional likelihood (2) becomes

$$P(w_1(d), \ldots, w_n(d)|c_j) = \prod_{i=1}^{n} P(w_i(d)|c_j), \tag{3}$$

where $P(w_i(\cdot)|c_j)$ denotes the conditional probability mass function of term w_i in class c_j. The Bayes discriminant function (1) therefore becomes

$$\begin{aligned} f(d) \quad = \quad & \sum_{i:w_i(d)=1} \log \tfrac{P(w_i|c_1)}{P(w_i|c_2)} + \\ & \sum_{i:w_i(d)=0} \log \tfrac{1-P(w_i|c_1)}{1-P(w_i|c_2)} + \log \tfrac{P(c_1)}{P(c_2)}, \end{aligned} \tag{4}$$

where $P(w_i|c_j)$ denotes the conditional probability of occurrence of term w_i in documents of class c_j, i.e., $P(w_i|c_j) = P(w_i(\cdot) = 1|c_j)$. The log likelihood ratios in (4) are, essentially, term weighting factors that attain large absolute values for terms that are strong discriminators between a pair of classes.

This framework is not limited to binary classification problems. The Bayes discriminant function for a general K-class classification task can be written as

$$g(d) = \arg \max_{j=1,\ldots,K} P(c_j|d). \tag{5}$$

Given a set of class labels $C = \{c_1, \ldots, c_K\}$ and the corresponding (training) set D_j of documents representing class c_j for each $j \in \{1, \ldots, K\}$, conditional probabilities $P(w_i|c_j)$ are estimated by

$$P(w_i|c_j) = \frac{1 + \sum\limits_{d \in D_j} w_i(d)}{2 + |D_j|}, \tag{6}$$

which is the ratio of the number of documents that contain term w_i in class c_j to the total number of documents in class c_j. The constants in numerator and denominator in (6) are introduced according to Laplace's rule of succession in order to avoid zero-probability terms.

The prior class probabilities $P(c_j)$ are estimated by

$$P(c_j) = \frac{|D_j|}{\sum\limits_{k=1}^{K} |D_k|}. \tag{7}$$

Together, equations (3) and (5–7) comprise the well-known Naive Bayes classifier.

3.2 Higher Order Data Representation

The data representation on which we build in the following section was initially used in our prior work [4] for anomaly detection in time series data. In the representation employed herein, a set D of documents is considered as a bipartite graph $G = (V_D \cup V_W, E)$. Vertices in V_D correspond to documents, while vertices in V_W correspond to terms. Two vertices $d \in V_D$ and $w \in V_W$ are connected by an edge $(d, w) \in E$ iff document d contains the term w.

A higher-order path in dataset D is a chain subgraph of G. For example, a chain w_i—d_l—w_k—d_r—w_j, which we will denote by $(w_i, d_l, w_k, d_r, w_j)$, encodes the fact that document $d_l \in D$ contains terms w_i and w_k, while document $d_r \in D$ contains terms w_k and w_j. The order of a path is determined by the number of document vertices the path spans. Thus, path $(w_i, d_l, w_k, d_r, w_j)$ captures a second-order co-occurrence between terms w_i and w_j, realized through term w_k shared by distinct documents d_l and d_r.

Higher-order paths simultaneously capture term co-occurrences within documents as well as term sharing patterns across documents, and in doing so provide a much richer data representation than the traditional feature vector form. As we will see in Sect. 5, the richness of representation becomes crucial when the small size of a training dataset prohibits traditional methods from accurately estimating model parameters.

4 Approach

In this section we present two novel methods of leveraging higher-order dependencies between features for supervised machine learning. In Sect. 4.1, we build on the Naive Bayes classifier by introducing a higher-order classifier termed Higher Order Naive Bayes. Our approach, however, can be generalized beyond the probabilistic machine learning methods. In Sect. 4.2, we introduce a data-driven space transformation that allows any learner that operates in vector spaces to leverage higher-order dependencies in data. An efficient path counting algorithm is given in Sect. 4.3.

4.1 Higher Order Naive Bayes

Higher-order paths as defined in Sect. 3.2 allow us to extract rich relational information between features in a dataset. We incorporate this information into a Bayesian learning framework by modifying the parameter estimation equations (6) and (7) to depend on the counts of higher-order paths as follows.

Let $\varphi(w_i, D)$ denote the number of higher-order paths that contain term w_i given the dataset D, and let $\Phi(D)$ denote the total number of higher-order paths in D. The parameter estimation equations of the proposed Higher Order Naive Bayes classifier are:

$$\hat{P}(w_i|c_j) = \frac{1 + \varphi(w_i, D_j)}{2 + \Phi(D_j)}, \tag{8}$$

and

$$\hat{P}(c_j) = \frac{\Phi(D_j)}{\sum\limits_{k=1}^{K} \Phi(D_k)}. \tag{9}$$

4.2 Leveraging Higher Order Dependencies by a Vector Space-Based Classifier

In this section, we present a novel data transformation that allows any classifier operating in vector spaces to take advantage of higher-order dependencies between features. We describe our approach for the case of binary classification. This, however, does not limit the applicability of the proposed approach, because numerous methods for multi-class classification based on binary classifiers have been proposed (see [14] for an overview). The proposed data transformation proceeds as follows.

Given two sets D_j and D_k of (training) documents from classes c_j and c_k, resp., the class conditional term probabilities (8) are computed. Let us denote the corresponding conditional log likelihood ratios as

$$\phi_i^{(1)} = \log \frac{\hat{P}(w_i|c_j)}{\hat{P}(w_i|c_k)}, \tag{10}$$

and

$$\phi_i^{(0)} = \log \frac{1 - \hat{P}(w_i|c_j)}{1 - \hat{P}(w_i|c_k)}. \tag{11}$$

Each binary document vector $d = (d_1, \ldots, d_n)$, $d \in D_j \cup D_k$, is then transformed into a real vector $\hat{d} = (\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$, where

$$\hat{d}_i = \begin{cases} \frac{\phi_i^{(1)}}{\sqrt{|\phi_i^{(1)}|}}, & \text{if } d_i = 1, \phi_i^{(1)} \neq 0 \\ \frac{\phi_i^{(0)}}{\sqrt{|\phi_i^{(0)}|}}, & \text{if } d_i = 0, \phi_i^{(0)} \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Finally, the resulting dataset $\hat{D}_j \cup \hat{D}_k$ is used as input for training a binary classifier for classes c_j and c_k.

Data transformation (12) assigns weights that are high in absolute values for highly discriminative terms present in a document. The normalizing factors[2] in (12) moderate the spread of values of each feature in order to allow less discriminative terms to retain a certain level of influence over the classification. This level of influence depends on the discriminative power of a term as measured by (10) and (11). An illustration of the effect of the normalizing factors can be found in Sect. 6.

4.3 Algorithms for Counting Second Order Paths

For convenience of exposition, in this subsection we denote features by small letters and instances by capital letters. Thus, for a given $m \times n$ data matrix $X = ||x_L^i||$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, x_L^i denotes the value of feature i in data instance L.

The number of second-order paths for each feature in a boolean dataset X with m objects and n features can be obtained in $O(m^2 n^3)$ time by Algorithm 1. This algorithm is trivial to implement and requires no additional memory space beyond the $O(mn)$ space needed to store the dataset X.

However, the $O(m^2 n^3)$ computational complexity can be reduced to $O((m+n)n^2)$ if enough memory is available to store two additional symmetric $n \times n$ matrices, which we denote by A and A^2. Matrix $A = X^T X$ holds the number of first-order paths between every pair of features, and is used to compute the matrix $A^2 = AA$. The ij-th element $a_{ij}^{(2)}$ of matrix A^2 holds the upper bound on the number of second-order paths where features i and j are the two end vertices. In order to obtain the exact number of such second-order paths, the value $a_{ij}^{(2)}$ must be corrected by first subtracting the number of paths where one of the features appears more than once (e.g. (i, L, i, M, j)) and then subtracting the number of paths where the same object vertex appears twice (e.g. (i, L, k, L, j)). The first correction can be accomplished by setting to zero all the diagonal elements of matrix A prior

[2]It is possible to omit the normalizing factors in (12). However, we have found experimentally that the normalized transformation, on average, yields slightly higher classification accuracies.

Algorithm 1: Count Second-Order Paths

```
   Input: Boolean $m \times n$ matrix $X = ||x^i_L||$
   Output: Vector $p = (p^1, \ldots, p^n)$, whose $i$-th coordinate $p^i$ equals the number of
           second-order paths that contain feature $i = 1, \ldots, n$
   Output: The total number $t$ of second-order paths
1  Initialize vector $p$ to be a zero vector: $p^i := 0$, $i = 1, \ldots, n$
2  Initialize the path counter $t := 0$
3  for $i \in \{1, \ldots, n\}$ do
4      for $k \in \{1, \ldots, n\} \setminus \{i\}$ do
5          for $j \in \{1, \ldots, n\} \setminus \{i, k\}$ do
6              for $L \in \{1, \ldots, m-1\}$ do
7                  for $M \in \{L+1, \ldots, m\}$ do
8                      $a := x^i_L x^k_L x^k_M x^j_M$
9                      $p^i := p^i + a$
10                     $p^k := p^k + a$
11                     $p^j := p^j + a$
12                     $t := t + a$
                   end
               end
           end
       end
   end
13 return $(p, t)$
```

to computing A^2. Matrix A can also be used for obtaining for each feature i, the count of second-order paths where feature i is the middle vertex (e.g. (k, L, i, M, j)).

Algorithm 2 implements this faster, but more memory consuming approach. The two correction steps mentioned above are implemented by steps 6 and 12-14. Computation of matrices A and A^2 in steps 5 and 8, respectively, of Algorithm 2 takes $O(mn^2 + n^3)$ time. The loop in step 9 takes $O(mn^2)$ due to step 12, which iterates over the data instances. The computational complexity of Algorithm 2 is dominated by computation of matrices A and A^2, and is therefore $O\left((m+n)n^2\right)$. Because of its lower computational complexity, Algorithm 2 was used in all the experiments reported in this work.

5 Experimental Results

Experimental evaluation was carried out on six widely-used text corpora. Four of these datasets were RELIGION, SCIENCE, POLITICS and COMP subsets of the 20 News Groups (20NG) [9] benchmark data. These particular subsets were selected to allow us to draw comparisons of our results with the related work [17], which will be discussed later in this section. Our preprocessing procedures closely followed those in [17]. First, all cross-postings in the 20NG data were removed. Then, for each dataset we performed stop word removal, stemming and removal of all terms that occurred in fewer than three documents in the dataset. The remaining terms were ranked by Information Gain. The top 2000 terms were selected. Finally, 500 documents were sampled at random from each class to comprise the 20NG datasets used in our experiments.

The other two datasets, Citeseer and Cora, are collections of scholarly research articles preprocessed by [16]. The Citeseer dataset contained 3312 documents with a vocabulary of 3703 terms. The Cora dataset comprised of 2708 documents with a vocabulary of 1433 terms. A summary description of all six datasets is provided in Table 1.

Naive Bayes (NB) was used as the baseline for evaluation of the proposed Higher Order Naive Bayes (HONB) classifier. Support Vector Machine (SVM) [19] was chosen as the base classifier for evaluation of the data transformation proposed in Sect. 4.2. SVM with the linear kernel has been shown [6] to perform well on text classification problems. The linear kernel allows us to observe the direct impact of leveraging higher-order dependencies, without any additional data transformations as performed implicitly by other kernel functions. Multi-class problems were addressed using the “one-against-one” classification scheme [8]. Under this scheme, a binary SVM classifier is constructed for every pair of classes. A data instance is then classified by each binary classifier and the final classification is determined by the majority vote over the assigned class labels. We refer to a SVM classifier constructed on the transformed data as Higher Order SVM (HOSVM). Experiments described in this section were done using second-order paths for HONB and HOSVM. We have conducted additional experiments using third-order paths, but the results did not differ significantly.

Figure 2 shows mean classification accuracies obtained by varying training set size from 5% up to 60%. For each training set size, eight trials were performed. On each trial, a set of documents were randomly sampled from each class for training, while the rest were used for

Algorithm 2: Count Second-Order Paths

```
   Input: Boolean $m \times n$ matrix $X = ||x_L^i||$
   Output: For each feature $i = 1, \ldots, n$, output the number of second-order paths that
           include feature $i$
   Output: The total number of second-order paths in $X$
1  Initialize vector $p = (p^1, \ldots, p^n)^T$ of per-feature path counts to be a zero vector
   $p^i := 0,\ i = 1, \ldots, n$
2  Initialize scalar $t$, which will store the total number of second-order paths
   $t := 0$
3  Compute vector $l = (l^1, \ldots, l^m)^T$ of $\ell_1$ norms of object (row) vectors of $X$
   $l^L = \sum_{i=1}^{n} x_L^i,\ L = 1, \ldots, m$
4  Compute vector $r = (r^1, \ldots, r^m)^T$ of numbers of pairs of non-zero features within each
   data instance with one feature removed
   $r^L = \frac{1}{2}(l^L - 1)(l^L - 2),\ L = 1, \ldots, m$
5  Compute the first-order feature co-occurrence matrix $A = ||a_{ij}||$
   $A := X^T X$
6  Set all diagonal elements $a_{ii}$ of $A$ to zero
   $\mathrm{diag}(A) := 0$
7  Compute vector $c = (c^1, \ldots, c^n)^T$ of squared column sums of $A$
   $c^i = \left( \sum_{j=1}^{n} a_{ji} \right)^2,\ i = 1, \ldots, n$
8  Compute the second-order feature co-occurrence matrix $A^2 = ||a_{ij}^{(2)}||$
   $A^2 := AA$
9  for $i = 1, \ldots, n$ do
10     if $i < n$ then
11         for $j = i + 1, \ldots, n$ do
12             Compute the number $s$ of paths $(i, L, k, L, j)$, $k \in \{1, \ldots, n\} \setminus \{i, j\}$, where
               feature $i$ $(j)$ is an end vertex and where both object vertices are the same
               $s = \sum_{L=1}^{m} x_L^i x_L^j \left(l^L - 2\right)$
13             $p^i := p^i + a_{ij}^{(2)}$ - s
14             $p^j := p^j + a_{ij}^{(2)}$ - s
15             $t := t + a_{ij}^{(2)}$ - s
           end
       end
16     Add to $p^i$ the number of paths $(k, L, i, M, j)$, $k, j \neq i$, $k \neq j$, $L \neq M$, where feature
       $i$ is the middle vertex
       Let $b = (b^1, \ldots, b^n)^T$ be a vector of cumulative sums of elements of the $i$-th column
       $a_{\cdot i}$ of matrix A, i.e., $b^h = \sum_{g=1}^{h} a_{gi},\ h = 1, \ldots, n$
       $p^i := p^i + c^i - a_{\cdot i}^T b - r^T x^i$
   end
17 return $(p, t)$
```

Table 1: Six datasets used in the experiments

Dataset	Classes
RELIGION (3)	alt.atheism, soc.religion.christian, talk.religion.misc
SCIENCE (4)	sci.crypt, sci.electronics, sci.med, sci.space
POLITICS (3)	talk.politics.guns, talk.politics.mideast, talk.politics.misc
COMP (5)	comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac-hardware, comp.windows.x
Citeseer (6)	AI, Agents, DB, HCI, IR, ML
Cora (6)	Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Theory

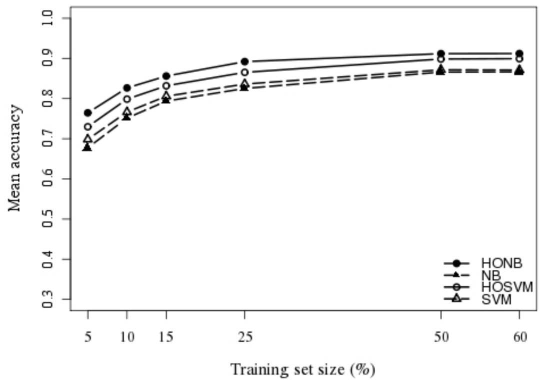
testing. On every trial, all terms that did not appear in any of the training documents were disregarded. The classifiers were then trained in the corresponding subspace of the original term space.

As can be seen from Figs. 2(a)–2(d), HONB and HOSVM[3] consistently outperformed first-order classifiers NB and SVM[3], resp., on the RELIGION, SCIENCE, POLITICS and COMP datasets. All of these accuracy improvements were statistically significant at the 5% level for (HONB, NB) pair of classifiers and in 83% of the cases for the (HOSVM, SVM) pair.

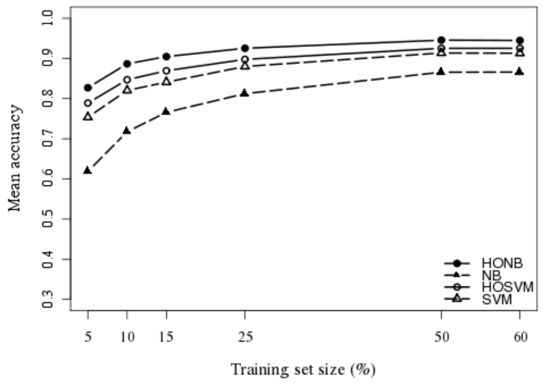
A slightly different pattern of performance can be observed on the Citeseer (Fig. 2(e)) and Cora (Fig. 2(f)) datasets. While classification accuracy of each method monotonically increased with increasing training set size, HONB drastically outperformed NB when training sets were small, reaching close to accuracy levels of SVM-based methods. HOSVM performed 1.5-2% better than SVM 66% of the time, but never worse overall. These results are especially encouraging in that they suggest that higher-order methods were able to construct robust models even when training data was particularly scarce. We speculate that changes in the distribution of highly-discriminative terms across classes as a result of increasing training set size allowed NB to outperform HONB on the Citeseer and Cora datasets once the amount of training data reached a certain point. We are currently investigating this hypothesis and intend to report our findings as part of our future work.

Consistent and statistically significant accuracy improvements attained by higher-order classifiers on small training sets led us to explore this aspect further. In order to simulate a real-world scenario where only a few labeled data instances are available, and to illustrate accuracy improvements in a setting comparable with the related work [17], we focused our attention on 5% training samples. In case of the 20NG datasets, for instance, this corresponded to training on 25 documents per class and testing on the other 475 documents per

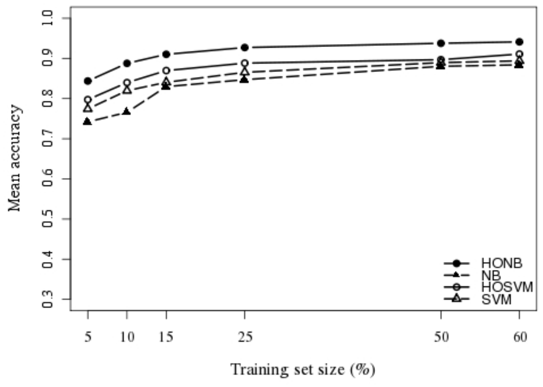
[3]When selecting the value of the soft margin cost parameter C for SVM, we considered the set $\{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$ of possible values. On every trial, we picked the smallest value of C which resulted in the highest accuracy obtained on the *training* set.



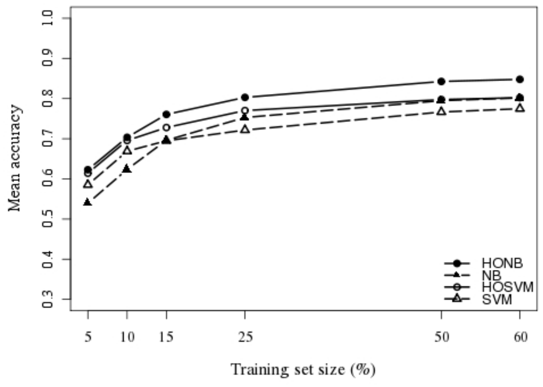
(a) RELIGION



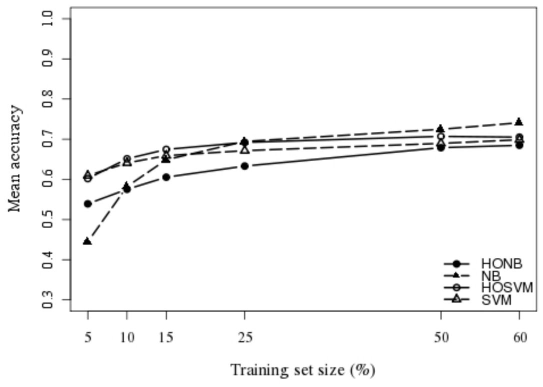
(b) SCIENCE



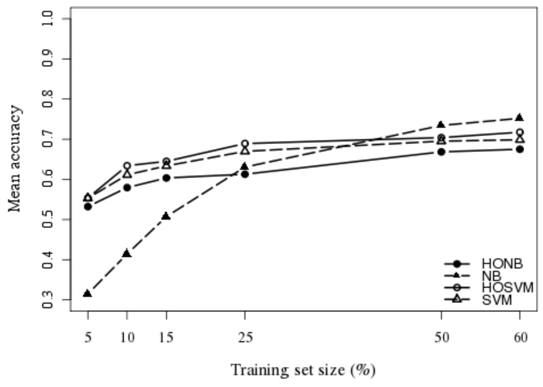
(c) POLITICS



(d) COMP



(e) Citeseer



(f) Cora

Figure 2: Scalability across training set size

class. Classification accuracies averaged over eight trials are reported in Table 2. Highest accuracies for pairs (NB, HONB) and (SVM, HOSVM) of classifiers are highlighted in bold. The corresponding standard deviations are also reported in Table 2.

Table 2: Mean classification accuracies

	NB		HONB		SVM		HOSVM	
Dataset	Acc.	St. dev.	Acc.	St. dev.	Acc.	St. dev.	Acc.	St. dev.
RELIGION (3)	0.66	0.051	**0.741**	0.03	0.699	0.022	**0.723**	0.023
SCIENCE (4)	0.632	0.071	**0.833**	0.043	0.751	0.029	**0.792**	0.039
POLITICS (3)	0.73	0.042	**0.832**	0.01	0.763	0.03	**0.793**	0.047
COMP (5)	0.539	0.051	**0.623**	0.036	0.585	0.022	**0.614**	0.031
Citeseer (6)	0.444	0.025	**0.539**	0.015	**0.609**	0.01	0.602	0.01
Cora (6)	0.315	0.006	**0.532**	0.021	0.553	0.021	**0.554**	0.026

The obtained results indicate that leveraging higher-order dependencies in the RELIGION, SCIENCE, POLITICS and COMP datasets lead to significant improvements in classification accuracies of both Bayesian and SVM-based approaches. The improvements of HONB over NB and of HOSVM over SVM on those datasets are statistically significant at the 5% level. The only exception is the (HOSVM, SVM) pair on the POLITICS dataset. Although the difference in SVM and HOSVM accuracies on the POLITICS dataset was significant at level $\alpha = 0.158$, HOSVM outperformed SVM on seven out of eight trials on that data by an average of 3%.

HONB performed particularly well on the 20NG data, outperforming NB by 11.7% and SVM by 5.8% on average. HONB also outperformed HOSVM by an average of 2.7%, although the differences in accuracy between the two classifiers were not statistically significant at the 5% level. HOSVM consistently outperformed SVM by an average of 3.1%.

On the Citeseer dataset, HONB outperformed NB by about 9% (significant at the 5% level), while HOSVM and SVM performed at the same level. On the Cora dataset, NB's average classification accuracy was 31.5%, which is the lowest accuracy across all datasets and classifiers considered in this work. Such low accuracy and almost zero standard deviation (Table 2) resulted from NB assigning all test documents to the majority class Neural_Networks. By exploiting valuable higher-order dependencies, HONB attained approximately 53% average accuracy, which is not significantly different at the 5% level from the roughly 55% accuracy of SVM and HOSVM.

In another set of experiments, we expanded our reach to compare the performance of higher-order classifiers to other approaches that also attempt to achieve better generalization performance on sparse input data. These related research efforts use clustering as a dimensionality reduction technique. In general they cluster words into groups and these groups are used as features in text classification. The authors of [17] observed that one advantage of using word clusters can be seen when word statistics (i.e., NB parameter estimates) are relatively hard to estimate. In order to demonstrate this, they conducted experiments with small numbers of labeled training documents from the subsets listed in Table 1 of the 20NG data. To provide a more objective comparison, we focused on the relative improvement over respective base models. In [17] the authors used multinomial NB (mNB) as their base model and improved on the performance of this model using word clusters as features instead of just words. The latter algorithm is indicated as NBWC in Table 3. Similarly, we used binomial NB as our base model and improved on the performance of this model using higher-order paths instead of individual documents to estimate parameters.

Table 3: Mean classification accuracies reported by [17]

Dataset	mNB	NBWC
RELIGION (3)	0.525	0.553
SCIENCE (4)	0.65	0.725
POLITICS (3)	0.62	0.67
COMP (5)	0.473	0.508

Tables 2 and 3 show that compared to NBWC, HONB achieved much better performance than the corresponding base model for all datasets (4.7% average difference between NBWC and mNB versus 11.7% average difference between HONB and NB). Additionally, our results are statistically significant.

In order to further verify the value of leveraging higher-order dependencies within the data, additional experiments were conducted. In the first experiment, when estimating the conditional probabilities (8) and (9), we used only the "pure" second-order paths, i.e., paths that involved terms which did not co-occur together in any single document of the training set. The results of these experiments are reported in Table 4 comparing performance of HONB with that of the pure HONB. The differences in performance of these classifiers were minimal and not statistically significant.

Table 4: Mean classification accuracies of a pure higher-order classifier

Dataset	HONB	pure HONB
RELIGION (3)	0.741	0.745
SCIENCE (4)	0.833	0.842
POLITICS (3)	0.832	0.836
COMP (5)	0.623	0.649

In the second experiment, prior to training an SVM classifier with the linear kernel, a data transformation analogous to (12) was performed using the first-order conditional term probabilities (6) instead of higher-order probabilities (8). The resulting approach is referred to as NBSVM. Mean classification accuracies attained by NBSVM[3] are shown in Table 5. Comparison of Tables 2 and 5 makes it clear that NBSVM performed worse than both SVM and HOSVM. These results indicate that taking advantage of higher-order dependencies was indeed crucial for achieving the performance improvements attained by HONB and HOSVM.

Table 5: Mean classification accuracies of NBSVM

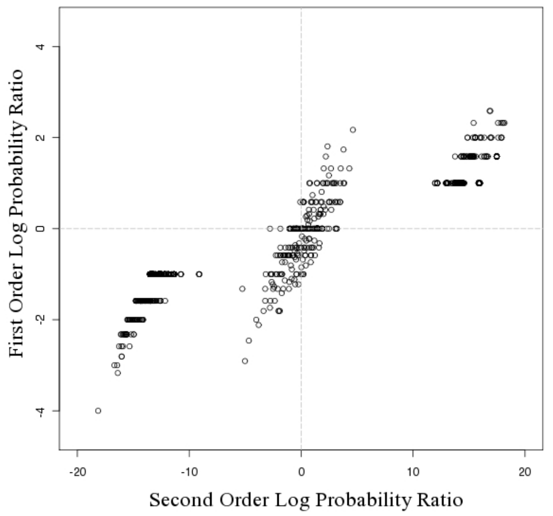
Dataset	NBSVM	HOSVM
RELIGION (3)	0.678	0.723
SCIENCE (4)	0.745	0.792
POLITICS (3)	0.759	0.793
COMP (5)	0.576	0.614

We have also conducted experiments with the Radial Basis Function (RBF) kernel for the HOSVM and SVM classifiers. The results were consistent with the findings of [6]. Namely, there were no significant differences between classification accuracies attained with the linear kernel and those attained with the RBF kernel.

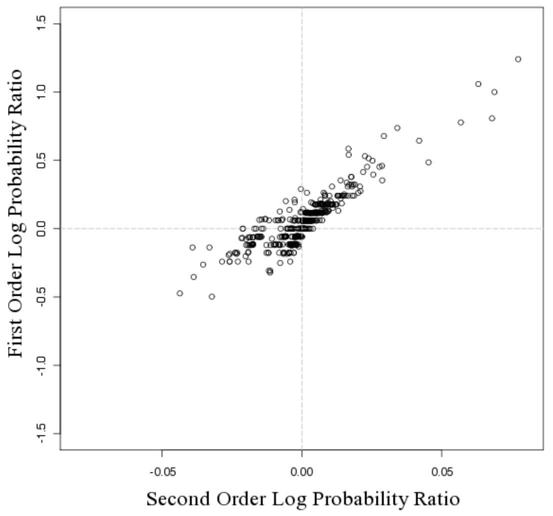
In summary, higher-order approaches HONB and HOSVM outperformed their first-order counterparts. Consistent performance improvements were observed on the 20NG data across a wide range of training set sizes. On the Citeseer and Cora datasets, HONB outperformed NB when training set sizes were below 10% and 25%, respectively. Under the conditions of small (5%) training samples from the Citeseer dataset, HOSVM performed the same as SVM, which outperformed HONB by 7%. HONB, SVM and HOSVM produced very similar results on the Cora dataset. It is interesting to note that while HOSVM scaled better across datasets, HONB did not require any parameter tuning and performed exceptionally well on the 20NG data.

6 Discussion

In order to gain a deeper understanding of the effect of using higher-order paths for estimation of conditional term probabilities (8), let us consider Fig. 3 generated based on one of the 5% (25 documents per class) training samples from the RELIGION dataset using two of its classes, “alt.atheism” and “soc.religion.christian”. For every term w_i, Fig. 3(a) presents a plot of the conditional log probability ratio (10) obtained from higher-order probabilities (8) (horizontal axis) versus the log ratio obtained from first-order probabilities (6) (vertical axis). Notice the differences in scales of values on the axes of Fig. 3(a): $[-20, 20]$ for higher-order log ratios versus $[-4, 4]$ for first-order log ratios. Additionally, three distinct groups of terms appeared as a result of using higher-order paths for estimating the model parameters. We found that terms that fell into the right (left) most group are highly-discriminative terms that appeared in documents of only one of the classes in the *training* set. Figure 3(a) reveals that due to drastic difference in scales of values of higher- and first-order log ratios, highly-discriminative terms exert much stronger influence on classification in HONB than in NB. In other words, highly-discriminative terms contribute more heavily to the Bayesian discriminant function (4) under HONB than under NB.



(a)



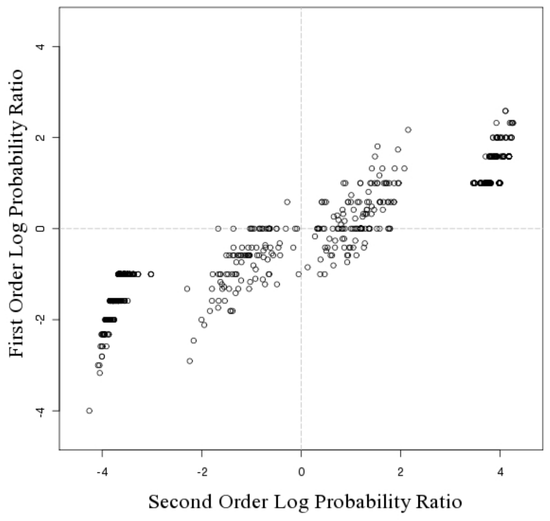
(b)

Figure 3: Conditional log probability ratios obtained from higher-order probabilities (8) (horizontal axes) versus the log ratios obtained from first-order probabilities (6) (vertical axes) on "alt.atheism" and "soc.religion.christian" classes of one of the 5% (25 documents per class) training samples from the RELIGION dataset.

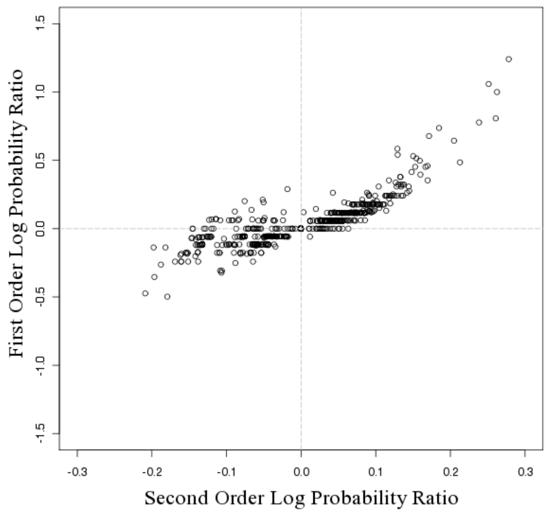
Similarly, Fig. 3(b) shows a plot of the conditional log probability ratios (11) of non-occurrence of a term in a HONB model versus a NB model. An important feature in this figure is the one order of magnitude difference in values on the axes: $[-0.1, 0.1]$ for higher-order log ratios on the horizontal axis versus $[-1.5, 1.5]$ for first-order log ratios on the vertical axis. Together, Figs. 3(a) and 3(b) indicate that while both NB and HONB take into account presence of terms as well as their absence, HONB tends to place more emphasis on the presence of terms in a document being classified.

As was noted in Sect. 4.2, the normalizing factors in (12) were introduced in order to allow terms that may appear in multiple classes, but are still good discriminators as measured by the log likelihood ratios (10) and (11), to have a non-negligible impact during document classification by HOSVM. The effect of these normalizing factors can be seen by comparing Figs. 3(a) and 3(b) with Figs. 4(a) and 4(b).

On the vertical axes in Figs. 4(a) and 4(b) are plotted the same first-order conditional log probability ratios as in Figs. 3(a) and 3(b), respectively. Plotted on the horizontal axes of Figs. 4(a) and 4(b) are the higher-order conditional log probability ratios shown on the horizontal axes of Figs. 3(a) and 3(b), respectively, and normalized as in (12). Note the change in scales of the horizontal axes once normalization has been applied: $[-20, 20]$ before normalization (Fig. 3(a)) versus $[-4, 4]$ after, and $[-0.1, 0.1]$ before normalization (Fig. 3(b)) versus $[-0.3, 0.3]$ after. It is this change in scales coupled with the increased spread of the middle group of terms along the horizontal axis that allowed good discriminator terms



(a)



(b)

Figure 4: Conditional log probability ratios obtained from higher-order probabilities (8) and normalized as in (12) for use by HOSVM (horizontal axes) versus the log ratios obtained from first-order probabilities (6) (vertical axes) on the same dataset as in Fig. 3.

from the middle group in Fig. 3(a) to increase their relative influence during document classification by HOSVM.

Although it is trivial to identify strongly discriminative features in a given training set, the question remains of how to weight those features for pattern classification. Methods proposed in this work address this question by leveraging higher-order dependencies between features.

7 Conclusions and Future Work

In prior work [7], we gave a mathematical proof supported by empirical results of the dependence of LSI on higher-order paths. In this work, a general approach to leveraging higher-order dependencies for supervised learning was developed. We presented a novel classification method termed Higher Order Naive Bayes (HONB). We further generalized our framework by developing a new data transformation that allows any classifier operating in vector spaces to take advantage of the rich relational information captured by higher-order paths. We also developed a $O\left((m+n)n^2\right)$ time algorithm for obtaining the counts of higher-order paths in boolean data. This algorithm improves over the $O\left(m^2n^3\right)$ complexity of a straight-forward path counting algorithm.

Higher-order paths allow a classifier to operate on a much richer data representation than the conventional feature vector form. This is especially important when working with small and sparse training sets where accurate parameter estimation of traditional (first-order) models becomes very challenging [17]. Experimental results affirmed the value of leveraging higher-order paths, resulting in significant improvements in classification accuracies on benchmark text corpora across a wide range of training set sizes. In addition, we compared our results with those reported in [17], where term clusters were used (instead of terms) as features for classification by Naive Bayes (NBWC). Our experiments demonstrated that HONB consistently achieved better performance over the baseline Naive Bayes (NB) classifier than did NBWC.

In ongoing efforts, we are developing methods for leveraging higher-order dependencies in various domains including nuclear detection, as well as in large datasets. We are also investigating extensions of the proposed framework to a number of other supervised learning approaches, and to unsupervised machine learning.

Acknowledgments

The authors wish to thank Rutgers University, the National Science Foundation and the Department of Homeland Security. This material is based upon work partially supported by the National Science Foundation under Grant Numbers 0703698 and 0712139 as well as by the Department of Homeland Security Science & Technology Directorate. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the Department of Homeland Security or Rutgers University.

The authors also wish to thank several colleagues at DIMACS, Rutgers University, including Dr. Ilya B. Muchnik for valuable discussions regarding the higher-order path counting algorithm, Dr. Alexey V. Nefyodov for helpful suggestions on the presentation of this work and Shenzhi Li for her many insightful comments during the development of the Higher Order Learning framework.

We are also grateful for the help of other co-workers, family members and friends. Co-author W. M. Pottenger also gratefully acknowledges the continuing help of his Lord and Savior, Yeshua the Messiah (Jesus the Christ) in his life and work.

References

[1] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27(2):307–318, 1998.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[3] P. Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. In *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 507–509, 1997.

[4] M. C. Ganiz, S. Kanitkar, M. C. Chuah, and W. M. Pottenger. Detection of interdomain routing anomalies based on higher-order path analysis. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 874–879, Washington, DC, USA, 2006. IEEE Computer Society.

[5] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7(2):3–12, 2005.

[6] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[7] A. Kontostathis and W. M. Pottenger. A framework for understanding latent semantic indexing (LSI) performance. *Inf. Process. Manage.*, 42(1):56–73, 2006.

[8] U. H.-G. Kreßel. Pairwise classification and support vector machines. In *Advances in kernel methods: support vector learning*, pages 255–268, Cambridge, MA, USA, 1999. MIT Press.

[9] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[10] S. Li, T. Wu, and W. M. Pottenger. Distributed higher order association rule mining using information extracted from textual data. *SIGKDD Explor. Newsl.*, 7(1):26–35, 2005.

[11] Q. Lu and L. Getoor. Link-based classification. In T. Fawcett and N. Mishra, editors, *ICML*, pages 496–503. AAAI Press, 2003.

[12] J. Neville and D. Jensen. Iterative classification in relational data. In *In Proc. AAAI*, pages 13–20. AAAI Press, 2000.

[13] J. Neville and D. Jensen. Dependency networks for relational data. *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 170–177, Nov. 2004.

[14] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[15] H. Schütze. Automatic word sense discrimination. *Comput. Linguist.*, 24(1):97–123, 1998.

[16] P. Sen and L. Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, February 2007.

[17] N. Slonim and N. Tishby. The power of word clusters for text classification. In *In 23rd European Colloquium on Information Retrieval Research*, 2001.

[18] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–878, 2001.

[19] V. Vapnik. *Statistical Learning Theory.* John Wiley, 1998.

[20] J. Xu and W. B. Croft. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst.*, 16(1):61–81, 1998.

[21] X. Zhang, M. W. Berry, and P. Raghavan. Level search schemes for information filtering and retrieval. *Inf. Process. Manage.*, 37(2):313–334, 2001.