2012 IEEE 12th International Conference on Data Mining

A Novel Semantic Smoothing Method based on Higher Order Paths for Text Classification

Mitat Poyraz, Zeynep Hilal Kilimci, Murat Can Ganiz
Computer Engineering Dept.
Doğuş University
Acıbadem, Kadıköy, 34722, Istanbul, Turkey
{mpoyraz, hkilimci, mcganiz }@dogus.edu.tr

***Abstract*—It has been shown that Latent Semantic Indexing (LSI) takes advantage of implicit higher-order (or latent) structure in the association of terms and documents. Higher-order relations in LSI capture "latent semantics". Inspired by this, a novel Bayesian framework for classification named Higher Order Naïve Bayes (HONB), which can explicitly make use of these higher-order relations, has been introduced previously. We present a novel semantic smoothing method named Higher Order Smoothing (HOS) for the Naive Bayes algorithm. HOS is built on a similar graph based data representation of HONB which allows semantics in higher-order paths to be exploited. Additionally, we take the concept one step further in HOS and exploited the relationships between instances of different classes in order to improve the parameter estimation when dealing with insufficient labeled data. As a result, we have not only been able to move beyond instance boundaries, but also class boundaries to exploit the latent information in higher-order paths. The results of our extensive experiments demonstrate the value of HOS on several benchmark datasets.**

Keywords-Naive Bayes; Semantic Smoothing; Higher Order Naive Bayes;Higher Order Smoothing;Text Classification

I. INTRODUCTION

Traditional machine learning algorithms assume that instances are independent and identically distributed (IID) [1]. This assumption simplifies the underlying mathematics of statistical models and allows the classification of a single instance. However in real world datasets, instances and attributes are highly interconnected. Consequently, the IID approach does not fully make use of valuable information about relationships within a dataset [4]. There are several studies which exploit explicit link information in order to overcome the shortcomings of IID approach [1], [2], [3], [4]. However, the use of explicit links has a significant drawback; in order to classify a single instance, an additional context needs to be provided. There is another approach which encounter this drawback, known as higher-order learning. It is a statistical relational learning framework which allows supervised and unsupervised algorithms to leverage relationships between different instances of the same class [9]. This approach makes use of implicit link information [5], [6], [7]. Using implicit link information within data provides a richer data representation. It is difficult and usually expensive to obtain labeled data in real world applications. Using implicit links is known to be effective especially when we have limited labeled data. In one of these studies, a novel Bayesian framework for classification named Higher Order Naïve Bayes (HONB) has been introduced [6], [7]. HONB is built on a graph based data representation which leverages implicit higher-order links between attribute values across different instances [6], [7], [8]. These implicit links are defined as higher-order paths. Attributes or features such as terms in documents of a text collection are richly connected by higher order paths of this kind. HONB exploits this rich connectivity [6].

In this study, we follow the same practice of exploiting implicit link information by developing a novel semantic smoothing method for Naïve Bayes (NB). We call it Higher Order Smoothing (HOS). HOS is built on novel graph based data representation which is inspired from the data representation of HONB. However in HOS, we take the concept one step further and exploit the relationships between instances of different classes. This approach improves the parameter estimation in the face of sparse data conditions by reducing the sparsity. As a result, we move beyond instance boundaries and class boundaries as well to exploit the latent information in higher-order paths.

We perform extensive experiments by varying the size of the training set in order to simulate real world settings and compare our algorithm with different smoothing methods and other algorithms. Our results on several benchmark datasets show that HOS significantly boosts the performance of Naïve Bayes (NB) and on some datasets it even outperforms Support Vector Machines (SVM).

The rest of the article is organized as follows: in Section II we briefly review related work. In Section III we provide background information on NB, smoothing methods, higher-order data representations and algorithms. Based on this background, we present our approach in detail in Section IV. Next, we describe our experiment setup in Section V and present our results in Section VI. This is followed by discussion in Section VII. Finally, we provide conclusion remarks and future work directions in Section VIII.

II. RELATED WORK

At the very basic level, we are motivated by the Latent Semantic Indexing (LSI) algorithm [10], which is a widely used technique in text mining and IR. It has been shown that LSI takes advantage of implicit higher-order (or latent) structure in the association of terms and documents. Higher-order relations in LSI capture "latent semantics" [11]. There are several disadvantages of using LSI in classification. It is

1550-4786/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDM.2012.109

IEEE computer society

a highly complex, unsupervised, black box algorithm. A second motivation stems from the studies in link mining which utilize explicit links [4]. Several studies in this domain have shown that significant improvements can be achieved by classifying multiple instances collectively [1], [2], [3]. However, use of explicit links requires an additional context for classification of a single instance. This limitation restricts the applicability of these algorithms. There are also several studies which exploit implicit link information in order to improve the performance of machine learning models [5], [6], [7]. Using implicit link information within data provides a richer data representation and it is shown to be effective especially under the scarce training data conditions. In one of these a novel Bayesian framework for classification named Higher Order Naïve Bayes (HONB) is introduced [6], [7]. HONB employs a graph based data representation and leverages co-occurrence relations between attribute values across different instances. These implicit links are named as higher-order paths. Attributes or features such as terms in documents of a text collection are richly connected by such higher order paths. HONB exploits this rich connectivity [6]. Furthermore, this framework is generalized by developing a novel data driven space transformation that allows vector space classifiers to take advantage of relational dependencies captured by higher-order paths between features [6]. This led to the development of Higher Order Support Vector Machines (HOSVM) algorithm. Higher-order learning which is a statistical relational learning framework consist of several supervised and unsupervised machine learning algorithms in which relationships between different instances are leveraged via higher order paths [8], [9], [12].

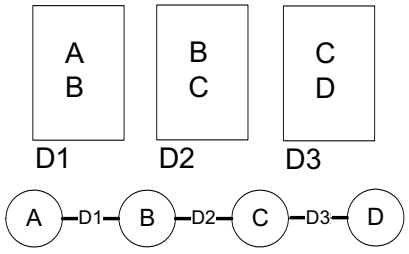

Figure 1. Higher Order co-occurrence [11]

A higher order path is shown in fig. 1 (reproduced from [11]). This figure depicts three documents, D1, D2 and D3, each containing two terms represented by the letters *A*, *B*, *C* and *D*. Below the three documents there is a higher-order path that links term *A* with term *D* through *B* and *C*. This is a third-order path since three links, or "hops," connect *A* and *D*. Similarly, there is a second order path between *A* and *C* through *B*. *A* co-occurs with *B* in document D1, and *B* co-occurs with *C* in document D2. Even if terms *A* and *C* never co-occur in any of the documents in a corpus, the regularity of these second order paths may reveal latent semantic relationship such as synonymy.

There are two commonly referred event models in Naïve Bayes for text categorization; multivariate Bernoulli (MVNB) and multinomial models (MNB). The first one is also known as binary independence model. In this model presence and absence of the terms is represented respectively "1" and "0". On the other hand in multinomial model is a unigram language model with integer term counts. Thus, each class can be defined as a multinomial distribution. Multinomial model is actually a unigram language model [13]. McCallum and Nigam [13], compared multivariate Bernoulli and multinomial model on several different data sets. Their experimental results show that the multivariate Bernoulli event model represents better performance at smaller vocabulary sizes, whereas the multinomial model generally performs well with large vocabulary sizes. Most of the studies about Naïve Bayes text classification employs multinomial model based on the recommendation of the well-known paper of McCallum and Nigam [13]. However, there are some interesting studies using binary data. For instance, MNB is shown to perform better with binary data in some cases such as spam detection [14], [15]. In another study [28], Kim et al., propose a multivariate Poisson Naïve Bayes text classification model with weight-enhancing method to improve performances on rare categories. Their experiments show that, this model is a good alternative to traditional Naïve Bayes classifier because it allows more reasonable parameter estimation.

In general, NB parameter estimation drastically suffer from sparse data because it has very large number of parameters to estimate in text classification problems. Number of parameters is ($|V|\,|C| + |C|$) where V denotes the dictionary and C denotes the set of class labels [16]. Most of the studies on NB text classification employ Laplace smoothing by default. There are a few studies that attempt to use different smoothing methods. For instance Juan and Ney [17] use multinomial model with several different smoothing techniques which origin from statistical language modeling field and generally used with n-gram language models. These include absolute discounting with unigram backing-off and absolute discounting with unigram interpolation. They state that absolute discounting with unigram interpolation gives better results than Laplace smoothing. They also consider document length normalization. Peng et al. [18] augment NB with n-grams and advanced smoothing methods from language modeling domain such as linear interpolation, absolute smoothing, Good-Turing smoothing, and Witten–Bell smoothing.

In [20] authors propose a semantic smoothing method based on the extraction of topic signatures. Topic signatures correspond to multi-word phrases such as n-grams or collocations that are extracted from the training corpus. After having topic signatures and multiword phrases they used them in semantic smoothing background collection model to smooth and map the topic signatures. They demonstrate that when the training data is small, the NB classifier with semantic smoothing outperforms better than NB with background smoothing (Jelinek-Mercer) and Laplace smoothing.

Support Vector Machines (SVM) is a popular large margin classifier. This machine learning method aims to find a decision boundary that separates points into two

classes thereby maximizing margin [27]. SVM projects data points into a higher dimensional space so that the data points become linearly separable by using kernel techniques. There are several kernels that can be used SVM algorithm. Linear kernel is known to perform well on text classification since most text categorization problems are linearly separable [27]. We include SVM results in our experiments for comparison reasons.

There is limited number of studies which employs tripartite graph in text categorization. In one of these, authors propose a novel clustering algorithm to automatically mine hierarchical taxonomy from the data set in order to take advantage of hierarchical classifier. Their method is called consistent bipartite spectral graph co-partitioning (CBSGC) algorithm, which is based on generalized singular value decomposition. According to the authors, document-term and category-document bipartite graphs reflect only partial information of data corpus. Therefore, in their approach, they co-cluster the category, document and term into a tripartite graph in order to leverage the complementary information contained in them. Their experiments show that, CBSGC discover reasonable hierarchical taxonomy and improves the classification accuracy [29].

III. BACKGROUND

In this section we review the Naïve Bayes event models and data representations. Although our method is not restricted to a particular application domain we focus on textual data.

A. Naïve Bayes Event Models

There are two generative event models that are commonly used with Naïve Bayes (NB) for text classification. First and the less popular one is multivariate Bernoulli event model which is also known as binary independence NB model (MVNB). In this model, documents are considered as events and they are represented a vector of binary attributes indicating occurrence of terms in the document. 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 $j \in \{1,\ldots,K\}$. the probability that a document in class c_j will mention term w_i. With this definition [21],

$$P(d \mid c_j) = \prod_{w \in d} \frac{P(w_i \mid c_j)}{1 - P(w_i \mid c_j)} \prod_{w \in W} \left(1 - P(w_i \mid c_j)\right) \qquad (1)$$

Conditional probabilities $P(w_i \mid c_j)$ are estimated by

$$\phi_{c_j, w_i} = P(w_i \mid c_j) = \frac{1 + \sum_{d \in D_j}^{|D|} w_i(d)}{2 + |D_j|} \qquad (2)$$

which is 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 Eq. 2 are introduced according to Laplace's rule of succession in order to avoid zero-probability terms [6]. Laplace smoothing adds a pseudo count to every word count. The main disadvantage of Laplace is to give too much probability mass to previously unseen events [30].

Second NB event model is multinomial model (MNB) which can make use of term frequencies. Let term w_i occur $n(d, w_i)$ times in document d, which is said to have length $\ell_d = \sum_{w_i} n(d, w_i)$. With this definition

$$\begin{aligned} P(d \mid c_j) &= P(L = \ell_d \mid c_j) P(d \mid \ell_d, c_j) \\ &= P(L = \ell_d \mid c_j) \binom{\ell_d}{\{n(d, w_i)\}} \prod_{w_i \in d} \theta_t^{n(d, w_i)} \end{aligned} \qquad (3)$$

Class conditional term probabilities are estimated using eq.4.

$$P(w_i \mid c_j) = \frac{1 + \sum_{d \in D_j}^{|D|} n(d, w_i)}{|W| + \sum_{d \in D_j, w_i \in d} n(d, w_i)} \qquad (4)$$

where $|W|$ is vocabulary (total number of words) [21].

Because of sparsity in training data, missing terms (unseen events) in the document can cause "zero probability problem" in NB. To eliminate this, we need to distribute some probability mass to unseen terms. This process is known as smoothing. The most common smoothing method in NB is Laplace smoothing. Formulas of the NB event models in Eq. 2 and Eq. 4 already included Laplace smoothing. In the next section, we provide details of a more advanced smoothing method which perform well especially on MVNB.

B. Jelinek-Mercer Smoothing

In Jelinek-Mercer smoothing method, the maximum estimate is interpolated with the smoothed lower-order distribution [20]. This is achieved by linear combination of maximum likelihood estimate (Eq.5) with the collection model (Eq.6) as shown in Eq.7. In Eq.6, D represents the whole training set, including the documents from all classes.

$$P_{ml}(w_i \mid c_j) = \frac{\sum_{d \in D_j}^{|D|} w_i(d)}{|D_j|} \qquad (5)$$

$$P(w_i \mid D) = \frac{\sum_{d}^{|D|} w_i(d)}{|D|} \qquad (6)$$

$$P(w_i \mid c_j) = (1 - \beta) \times P_{ml}(w_i \mid c_j) + \beta \times P(w_i \mid D) \qquad (7)$$

C. Higher Order Data Representation

Data representation we built on is initially used in [7]. In this study, it is indicated that definition of a higher-order path is similar to the one in graph theory, which states that given a non-empty graph $G = (V, E)$ of the form $V = \{x_0, x_1, \ldots, x_k\}$, $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$ with nodes x_i distinct, two vertices x_i and x_k are linked by a path P where the number of edges in P is its length.

A different approach is given in [6] by using a bipartite graph. In this approach a bipartite graph $G = ((V_D,V_W),E)$ is built from a set of D documents for a better representation. In this graph, vertices V_D correspond to documents and vertices in V_W correspond to terms. "There is an edge (d,w) between two vertices where $d \in V_D$ and $w \in V_W$ iff word w occurs in document d. In this representation, a higher-order path in dataset D can be considered as a chain subgraph of G. For example a chain w_i—d_l—w_k—d_r—w_j which is also denoted as $(w_i, d_l, w_k, d_r, w_j)$ is a second-order path since it spans through two different document vertices. 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" [6].

D. Higher Order Naïve Bayes

Rich relational information between terms and documents can be exploited by using higher-order paths. In Higher Order Naïve Bayes (HONB) this valuable information is integrated into multivariate Bernoulli Naïve Bayes algorithm (MNVB) by estimating parameters from higher-order paths instead of documents [6]. Formulation of parameter estimates are given in Eq.8 and Eq.9 which are taken from [6].

$$P(w_i \mid c_j) = \frac{1 + \varphi(w_i, D_j)}{2 + \phi(D_j)} \tag{8}$$

$$P(c_j) = \frac{\phi(D_j)}{\sum_{k=1}^{K} \phi(D_k)} \tag{9}$$

The number of higher-order paths containing term w_i given the set of documents that belongs c_j is represented by $\varphi(w_i, D_j)$. On the other hand, $\phi(D_j)$ denote the total number of higher-order paths extracted from the documents of c_j. Eq. 8 includes the Laplace smoothing on order to avoid zero probability problem for the terms that don't exist in c_j.

IV. Approach

In this section we present a novel semantic smoothing method called Higher Order Smoothing (HOS) by following the same approach of exploiting implicit link information. HOS is built on a graph-based data representation from the previous algorithms in higher-order learning framework such as HONB [6], [7]. However, in HONB, higher order paths are extracted in the context of a class. Therefore we cannot exploit relations between terms and documents in different classes.

In HOS we take the concept one step further and exploit the relationships between instances of different classes in order to improve the parameter estimation. As a result, we are not only moving beyond document boundaries but also class boundaries to exploit the latent semantic information in higher-order co-occurrence paths between terms. We accomplish this by extracting higher order paths from the whole training set including all classes of documents. Our aim is to reduce sparsity especially in the face of insufficient labeled data conditions.

In order to do so, we first convert the nominal class attribute to a number of binary attributes each representing a class label. For instance, in WebKb4 dataset 'Class' attribute has the following set of values C = {course, faculty, project, staff, student}. We add these four class labels as new terms (i.e. columns to our document by term matrix). We call them "class labels". Each of these labels indicates if the given document belongs to a particular class or not.

After this transformation, we slightly modify the higher-order data representation by characterizing a set of D documents, their terms and class labels as a tripartite graph. In this tripartite graph $\hat{G}=((V_W,V_C,V_D),E)$, vertices in V_D correspond to documents, vertices in V_W correspond to terms, and finally vertices in Vc correspond to class terms or in other words class labels. Fig. 2, shows such a tripartite graph which represents relationship between terms, class labels, and documents. Similarly, to previous higher-order data representation with bipartite graph, a higher order path in dataset D can be considered as a chain subgraph of $\hat{G}$. However, we are interested in such chain subgraphs that start with a term vertex from V_W, spans through different document vertices in V_D, and terminate with a class term vertex in V_C. w_i—d_s—w_k—d_r—c_j is such a chain which we denote by $(w_i, d_s, w_k, d_r, c_j)$. This chain corresponds to a second-order path since it spans through two document vertices. These paths have potential to cross class boundaries and capture latent semantics. We enumerate higher-order paths between all the terms in the training set and the class terms. These higher order paths capture the term co-occurrences within a class of documents as well as term relation patterns across classes. As a result, they provide more dense data representation than the traditional vector space. This is the basis of our smoothing algorithm.

Let's consider w_1—d_1—w_2—d_2—c_1 which is an example chain in the tripartite graph given in Fig. 2. This chain is indicated with red bold lines and it corresponds to a second-order path. In this example let's assume that w_1 never occurs in the documents of c_1. We still can estimate parameter value of w_1 for c_1 using such paths. This is achieved by intermediate terms such as w_k that co-occurs with w_i (given w_k occurs in the documents of c_j). As can be seen from the example, this new data representation and the new definition of higher order paths allow us to calculate class conditional

probabilities for some of the terms that don't occur in documents of a particular class. This framework serves as a semantic smoothing method for estimating model parameters of previously unseen terms given the fact that higher order paths reveal latent semantics [11].

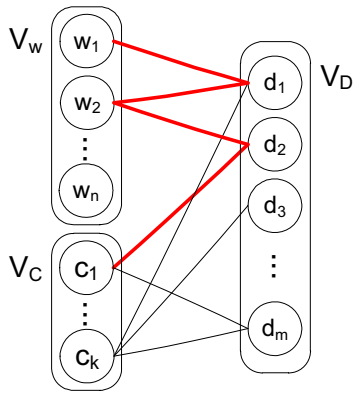

Figure 2. Data representation for HO paths using tripartite graph

Based on this representation and modified definition of higher-order paths we can formulate HOS. Let $\partial(w_i,c_j)$ denote the number of higher-order paths that is between term w_i and class label c_j in the dataset D, and $\Phi(D)$ denote the total number of higher order paths between all terms and all class terms in D. Please note that D represents all documents from all classes. This is one of the important differences between the formulation of HONB and HOS. The parameter estimation equation of the proposed HOS is given in Eq.10. Although HOS has the potential to estimate parameters for terms that don't occur in the documents of a class but occurs in other classes in training data, there can be terms that occur only in test set. In order to avoid zero probability problems in these cases, we apply Laplace smoothing in Eq.10. Class priors are calculated according to multivariate Bernoulli model using documents.

$$P(w_i \mid c_j) = \frac{1+\partial(w_i,c_j)}{2+\phi(D)} \qquad (10)$$

We recognize that different orders of paths may have different contribution to semantics and provide even richer data representation. Similar to the linear interpolation (a.k.a. Jelinek-Mercer) we can combine estimates calculated from different order of paths. Eq.11 shows the linear combination of first-order paths (just co-occurrences) with second-order paths. We use this formulation in our experiments. We set β to 0.5 experimentally since for majority of our datasets and training set size percentages this value performs best.

$$P(w_i \mid c_j) = (1-\beta) \times P_{fo}(w_i \mid c_j) + \beta \times P_{so}(w_i \mid c_j) \qquad (11)$$

The overall process of extracting second-order paths for HOS is described in Algorithm 1. It is based on the enumeration algorithm proposed in [6] and which is described in detail in [8].

Algorithm 1 : Enumerating second-order paths for HOS

Input **:** Boolean document-term data matrix $X = \| X_d^t \|$

*Output***:** O_2 matrix which stores the number of second-order paths in data matrix X

1. Initialize vector $l = (l^1,\ldots,l^n)$, which will store class labels of X
2. **for** each row i in data matrix X
   2a. $l^i = \| X_i^{t-1} \|$
3. Initialize class labels binary matrix $C_{lb} = \| C_{lb\,d}^{\,c} \|$ which will represent each class value as binary where c is the number of classes in X
4. **for** each row i in C_{lb} matrix
   4a. **for** each column c in C_{lb} matrix
      4b. **if** l^i is equal to j
         4c. set $C_{lb}(i,j)$ equal to 1
5. Compute matrix $X_{clb} = \| X_{clb\,d}^{\,t+c} \|$ by appending binary class valued matrix C_{lb} to X
6. Compute first-order co-occurrence matrix $O_1 = X_{clb}{}^T X_{clb}$
7. Compute second-order co-occurrence matrix $O_2 = O_1 O_1$
8. **for** each row i in first-order co-occurrence matrix O_1
   8a. **for** each column j in first-order co-occurrence matrix O_1
      8b. Compute scalar s, to eliminate paths in the form of t_1,d_1,t_2,d_1,t_3, where both document vertices (d_1) are same
      s = O_2(i,j) – (O_1(i,j) * (O_1(i,i) + O_1(j,j)))
      8c. Update the element of second-order co-occurrence matrix, O_2(i,j) = O_2(i,j) + s
10. Return O_2

In the algorithm above, at first, class labels are removed from the document by term data matrix and stored in a vector. Following this binary class labels matrix is built. In the binary class labels matrix rows represents class value of documents as binary vectors where the index position of the class label has 1 and other positions has 0. Afterwards, binary class labels matrix is combined with original data matrix X. This results in a new matrix called class-binarized data matrix (X_{clb}) which stores the input data matrix and its binary class values. We use X_{clb} to calculate the first and second order paths. The matrix which represents the co-occurrence relations of terms (first order paths) is calculated by multiplying transpose of X_{clb} (term by document matrix) and X_{clb} (document by term matrix). Second order paths matrix is calculated by multiplying first order paths by itself. Although this matrix includes the number of second

order paths between terms (including binary class labels), we are interested in certain type of paths according to the path definition in [7]. This definition does not allow repeated terms or documents in the context of a higher-order path in order to exploit latent relations between different terms occurring in different documents. Therefore, a scalar *s* if computed in order to eliminate paths t_1, d_1, t_2, d_1, t_3 , where both document vertices (d_1) are same and second order paths matrix is updated using this scalar value.

V. Experiment Setup

In order to analyze the performance of our algorithm for text classification, we use three widely used benchmark datasets. First one is a variant of 20 Newsgroups[1] dataset. It is called 20News-18828 and it has fewer documents from the original 20 Newsgroup dataset since duplicates postings are removed. Additionally for each posting headers are deleted except "From" and "Subject" headers. Our second dataset is the WebKB[2] dataset which includes web pages collected from computer science departments of different universities. There are seven categories which are student, faculty, staff, course, project, department and other. We use four class version of the WebKB dataset which is used in [13]. This dataset is named as WebKB4. Third dataset is 1150Haber dataset which consists of 1150 news articles in five categories namely economy, magazine, health, politics and sport collected from Turkish online newspapers [22]. We particularly choose a dataset in different language in order to observe efficiency of higher-order algorithms in different languages. Similar to LSI, we expect higher-order paths based algorithms HONB and HOS to perform well on different languages without any need for tuning. More information about this data set and text classification on Turkish documents can be found in [23]. One of the most important differences between WebKB4 and other two datasets is the class distribution. While 20News-18828 and 1150Haber have almost equal number of documents per class, WebKB4 have highly skewed class distribution. For the statistics given in Table 1, we apply no stemming or stop word filtering. We only filter infrequent terms whose document frequency is less than three. Descriptions of the datasets, under these conditions are given in Table 1 including number of classes (|C|), number of documents (|D|) and the vocabulary size (|V|).

Table 1. Descriptions of the datasets with no preprocessing

Data Set	|C|	|D|	|V|
20News-18828	20	18,828	50,570
WebKB4	4	4,199	16,116
1150Haber	5	1150	11,038

[1] http://people.csail.mit.edu/people/jrennie/20Newsgroups

[2] http://www.cs.cmu.edu/~textlearning

As can be seen from Algorithm 1, complexity of the higher-order path enumeration algorithm is proportional to the number of terms. In order avoid unnecessary complexity and to finish experiments on time we reduce the dictionary size of all three datasets by applying stop word filtering and stemming using Snowball stemmer which has implementations for both English and Turkish. Finally, dictionary sizes are fixed to 2,000 by selecting the most informative terms using Information Gain feature selection method. All of these preprocessing operations are widely applied in the literature and it has been known that they usually improve the performance of traditional vector space classifiers. For that reason, we are actually giving a considerable advantage to our baseline classifier NB and SVM. Please note that HOS is expected to work well when the data is very sparse. In fact, these preprocessing operations reduce sparsity. As mentioned before we vary the training set size by using following percentages of the data for training and the rest for testing: 1%, 5%, 10%, 30%, 50%, 70%, 80% and 90%. These percentages are indicated with "ts" prefix to avoid confusion with accuracy percentages. We take class distributions into consideration while doing so. We run algorithms on 10 random splits for each of the training set percentages and report average of these 10 results augmented by standard deviations. While splitting data into training and test set, we employ stratified sampling. This approach is similar to [13] and [24] where they use 80% of the data for training and 20% for test.

Our dataset include term frequencies (tf). However, higher-order paths based classifiers HONB and HOS currently can only work with binary data. Therefore they convert term frequencies to binary values in order to enumerate higher-order paths. We use up to second-order paths based on the experiment results of previous studies [6], [7]. Since we use binary data, our baseline classifier is multivariate Bernoulli NB (MVBN) with Laplace smoothing. This is indicated as MVNB in the results. We also employ more advanced smoothing method with MVNB which is Jelinek-Mercer (JM). Furthermore, we compare our results with HONB and state of the art text classifier SVM. We used linear kernel in SVM since it has been known to perform well in text classification. Additionally, we optimize soft margin cost parameter C by using the set of $\{10^{-3}, \ldots, 1, 10^{1}, \ldots, 10^{3}\}$ of possible values. We picked the smallest value of C which resulted in the highest accuracy. We observed that C-value is usually 1 when the training data is small (e.g. up to 10%) and it is usually 10^{-2} when training data increase (e.g. after 10%) with the exception of 1150Haber which is our smallest dataset. In 1150Haber, best performing C value is 1 in all training set percentages except 90%.

VI. Experiment Results

We use average accuracy values of 10 random trial experiments with varying training set size percentages. These accuracy results are our main evaluation metric and

they are augmented by standard deviations in the result tables below. Additionally, we employed other commonly employed evaluation metrics in order to evaluate the effectiveness of HOS from a wider perspective. These include F-measure (F1) and the area under the ROC curve (AUC) metrics. However, we report these values only for 80% training data level due to length restrictions. On the other hand, we observe that F1 and AUC exhibit similar patterns. We also provide statistical significance tests in several places by using Student's t-Test. This is especially useful when accuracy values of different algorithms are close to each other. We use $\alpha = 0.05$ significance level and consider the difference is statistically significant if the probability associated with Student's t-Test is lower. In the following result tables, accuracy values of HOS are indicated with bold font if they are higher than the values of our baseline MVNB classifier.

Our experiments show that HOS demonstrate remarkable performance on 20 Newsgroups dataset. This can be clearly seen in Table 2 showing the performance of algorithms in different training set size conditions. HOS statistically significantly outperforms our baseline classifier MVNB (with default Laplace smoothing) by a wide margin in all training set percentages. Moreover, HOS statistically significantly outperforms all other algorithms including NB with Jelinek-Mercer smoothing (MVNB+JM) and HONB. Although it is not statistically significant at 90% training set size, HOS outperforms SVM for all training set percentages. SVM is one of the best performing algorithms in text categorization domain. The performance improvement is especially visible at low ts levels. It is important to note that 20 newsgroups is one of the most commonly used datasets in text mining domain.

Table 2. Accuracy and standard deviations of algorithms on 20 Newsgroups dataset with varying training set size.

ts	MVNB	MVNB+JM	HOS	HONB	SVM
1	24.77±2.49	48.01±1.37	42.92±3.61	44.09±2.04	32.65±1.75
5	55.68±1.26	69.10±0.68	65.81±1.57	64.65±0.92	56.16±1.11
10	65.01±1.57	72.95±1.42	**76.70±0.79**	69.93±0.62	65.15±0.61
30	72.83±0.74	75.66±0.63	**81.97±0.33**	76.12±0.38	75.99±0.61
50	75.11±0.58	76.64±0.68	**83.06±0.29**	78.53±0.37	79.35±0.34
70	75.65±0.64	76.81±0.67	**83.33±0.54**	79.92±0.34	81.53±0.32
80	76.29±0.58	77.01±0.71	**83.59±0.41**	80.49±0.50	82.07±0.46
90	76.21±1.18	76.50±1.02	**83.26±0.84**	80.11±0.65	82.38±1.15

Table 3 shows the performance of HOS on WebKB4 dataset. Although not as visible as 20 Newsgroups dataset, HOS still outperforms our baseline MVNB starting from 10% training set level. All these performance improvements are statistically significant. Additionally, HOS statistically significantly outperforms MVNB with JM smoothing starting from 30% level. Interestingly, HONB performs slightly better than HOS on this dataset. On the other hand SVM is significantly the best performing algorithm. We attribute the better performance of HONB and especially SVM to the skewed class distribution of the dataset. This the main difference of WebKB dataset from our other datasets.

Table 3. Accuracy and standard deviations of algorithms on WebKB4 dataset with varying training set size.

ts	MVNB	MVNB+JM	HOS	HONB	SVM
1	44.48±1.03	69.96±3.15	30.08±6.56	70.58±3.80	60.57±1.82
5	68.17±2.49	79.33±2.15	61.15±6.51	77.68±2.94	79.01±1.33
10	74.46±1.36	80.76±1.54	**77.71±2.33**	80.83±1.35	83.48±1.14
30	81.53±1.05	83.02±0.92	**85.24±0.75**	86.83±0.58	89.43±0.55
50	82.57±0.83	82.81±0.81	**86.08±0.55**	87.64±0.75	91.04±0.47
70	83.53±0.98	83.19±1.08	**87.01±0.87**	88.53±0.75	91.69±0.72
80	83.14±1.17	82.85±1.23	**86.47±1.25**	88.79±0.85	91.78±0.64
90	84.17±2.10	83.41±1.61	**87.01±1.20**	88.36±1.42	92.20±1.00

The performance of HOS on 1150Haber dataset, which can be seen in Table 4, is somewhat similar to 20 Newsgroups. HOS statistically significantly outperforms baseline MVNB starting from 10% and MVNB with JM smoothing from 30% level. HONB and HOS show a very similar performance on this dataset with the exception of small training set sizes (i.e. up to 30%) where HONB performs better. This may be attributed to the much larger number of paths generated by HONB compare to the HOS since 1150haber is our smallest dataset including 230 news documents per class. After 30% the differences between accuracies of HONB and HOS are not statistically significant. Similar to the 20 newsgroups dataset, HOS statistically significantly outperform SVM starting from 10% level. It is interesting to observe similar behavior of HOS in datasets with different properties in different languages. 20 newsgroups dataset contains highly noisy and informal use of language in newsgroups postings in English. On the other hand 1150haber includes relatively more formal use of language in mainstream newspaper column articles in Turkish.

Table 4. Accuracy and standard deviations of algorithms on 1150Haber dataset with varying training set size.

ts	MVNB	MVNB+JM	HOS	HONB	SVM
1	35.70±7.64	48.40±5.04	32.09±11.1	30.32±12.7	38.92±3.03
5	65.06±12.6	81.01±6.95	**67.00±11.9**	88.25±0.93	67.47±4.24
10	72.95±3.83	86.01±2.03	**83.13±4.12**	91.61±0.85	76.27±2.71
30	87.64±1.14	91.49±0.71	**93.79±0.31**	94.20±0.59	87.39±1.21
50	88.73±0.65	91.10±0.63	**94.42±0.42**	94.73±0.57	89.55±1.12
70	89.97±0.88	91.39±0.83	**95.01±0.85**	95.30±0.96	90.55±1.49
80	89.70±2.40	90.83±2.50	**94.96±1.84**	95.91±1.60	91.91±2.39
90	90.78±2.73	91.48±2.42	**94.35±2.14**	95.22±1.75	90.78±1.93

In order to quantify the level of the performance improvement over other algorithms, we define the following performance gain for the accuracy.

$$gain_{HOS} = \frac{p_{HOS} - p_X}{p_X} \quad \textbf{(12)}$$

where p_{HOS} is the Higher Order Smoothing algorithm's accuracy result and p_X stands for the result of the other algorithms (MVNB, MVNB+JM, HONB, or SVM). We present performance improvements of HOS over other algorithms on Table 5, 6 and Table 7. Improvements are most visible in 20 Newsgroups dataset. In Table 5, we can see that HOS improves upon MVNB and SVM about 17% in terms of accuracy at 10% training set size (ts) level on 20 Newsgroups dataset. We can observe improvements in all training set size levels on this dataset. Table 6 shows the accuracy gains of HOS on other algorithms including MVNB with Laplace, MVNB with Jelinek-Mercer, and SVM on WebKB4 dataset. Table 7 show accuracy gains on different training set size levels of 1150haber dataset.

Table 5. Performance improvement of HOS over other methods on 20 Newsgroups dataset with varying training set size.

ts	MVNB	MVNB+JM	HONB	SVM
10	17.98	5.14	9.68	17.73
30	12.55	8.34	7.69	7.87
50	10.58	8.38	5.77	4.68
70	10.15	8.49	4.27	2.21
80	9.57	8.54	3.85	1.85
90	9.25	8.84	3.93	1.07

Table 6. Performance improvement of HOS over other methods on WebKB4 dataset with varying training set size.

ts	MVNB	MVNB+JM	HONB	SVM
10	4.36	-3.78	-3.86	-6.91
30	4.55	2.67	-1.83	-4.69
50	4.25	3.95	-1.78	-5.45
70	4.17	4.59	-1.72	-5.10
80	4.01	4.37	-2.61	-5.79
90	3.37	4.32	-1.53	-5.63

Table 7. Performance improvement of HOS over other methods on 1150Haber dataset with varying training set size.

ts	MVNB	MVNB+JM	HONB	SVM
10	13.95	-3.35	-9.26	8.99
30	7.02	2.51	-0.44	7.32
50	6.41	3.64	-0.33	5.44
70	5.60	3.96	-0.30	4.93
80	5.86	4.55	-0.99	3.32
90	3.93	3.14	-0.91	3.93

We present the results of more evaluation metrics at the 80% training set level. This percentage is commonly used in random trial experiments [13], [24]. Table 8 shows F-measure (F1) performance of algorithms at 80% training set level. Similar trend can also be seen in here. HOS outperforms baseline MVNB for all the datasets. Table 9 presents AUC values of the algorithms in this training set percentage level. Again, HOS outperforms baseline MVNB for all the datasets. One interesting observation from this table is the results of algorithms on WebKB4 dataset. Although SVM is by far the best performing algorithm in this dataset in terms of accuracy, it has been outperformed by HOS in terms of AUC.

Table 8. F-measure performance of algorithms at 80% training set level.

ALGORITHM	20NEWS-18828	WEBKB4	1150HABER
HONB	79.96 ± 0.75	88.34 ± 0.97	95.92 ± 1.60
HOS	**83.02 ± 0.72**	**85.32 ± 1.74**	**94.95 ± 1.84**
MVNB	76.41 ± 0.59	82.80 ± 1.23	89.79 ± 2.40
MVNB+JM	77.39 ± 0.81	82.43 ± 1.31	90.96 ± 2.50
SVM	82.02 ± 0.47	90.81 ± 1.21	91.92 ± 2.39

Table 9. AUC performance of algorithms at 80% training set level.

ALGORITHM	20NEWS-18828	WEBKB4	1150HABER
HONB	98.18 ± 0.07	97.58 ± 0.27	99.57 ± 0.24
HOS	**98.57 ± 0.09**	**96.90 ± 0.46**	**99.56 ± 0.25**
MVNB	97.67 ± 0.17	96.17 ± 0.51	99.25 ± 0.38
MVNB+JM	97.74 ± 0.19	96.19 ± 0.54	99.43 ± 0.31
SVM	90.32 ± 0.25	93.41 ± 0.72	94.95 ± 1.50

VII. DISCUSSION

The use of higher-order paths for estimation of conditional term probabilities have been discussed in [7] and [8]. It is observed that highly-discriminative terms exhibit much stronger influence on classification by HONB than by NB. Additionally, HONB tends to place more emphasis on the presence of terms in a document being classified [7]. Since HOS is based on higher-order paths, it enjoys some of these benefits. However, in HOS we are enumerating much fewer number of higher-order paths because paths need to end with a class label. Therefore, we have less data to extract patterns from. As a result, HONB as a higher performance on small training set sizes compare to HOS especially at 1% and 5% levels. Yet, HOS quickly catches up about at 10% level and outperforms HONB especially in 20 Newsgroups dataset. This dataset has relatively large number of classes (having 20 classes compare to six classes in WebKB4, and five classes in 1150Haber). With 20 class labels, we can extract much stronger patterns from higher-order paths using HOS. It would be interesting to observe the performance of HOS on real world datasets with much larger number of categories.

Results on 1150haber dataset suggest that HONB and HOS may also perform well on different languages than English without additional language specific tuning. This property is similar to the LSI. This can be an important advantage for HOS compare to the natural language processing based semantic methods such as [19].

In terms of training time complexity, an $O\left(n^2(m+n)\right)$ algorithm is given in previous studies for obtaining counts of higher-order paths in a dataset with m instances and n dimensions [6], [8]. In training phase, HONB forms a document by term matrix (m x n) for each class, and use this algorithm to obtain counts of higher-order paths between all terms. Our approach which is given in Algorithm 1, suggests using the same principles; therefore, it has the same computational complexity. When we examine Algorithm 1 more closely, the computation of matrices O_1 and O_2 in steps 6 and 7, respectively, takes $O\left(mn^2+n^3\right)$ time. The loops in step 4 and 8, iterates over C_{lb} and O_1 matrices, respectively, and takes $O\left(n^2\right)$ time. The computational complexity of Algorithm 1 is dominated by computation matrices O_1 and O_2, therefore the complexity is $O\left(mn^2+n^3\right)$.

However, given the fact that we have based our computations on a single document by term matrix and we have used the paths ending only with class labels, we are enumerating much fewer numbers of paths. So in practice, HOS runs faster than HONB. Both HOS and HONB share the low classification time complexity of Naïve Bayes.

VIII. CONCLUSIONS AND FUTURE WORK

It has been shown that LSI takes advantage of implicit higher-order (or latent) structure in the association of terms and documents. Higher-order co-occurrence relations in LSI capture "latent semantics" [11]. Motivated by this, a novel Bayesian framework for classification named Higher Order Naïve Bayes (HONB) is introduced [6], [7]. HONB can explicitly make use of these higher-order co-occurrence relations in the context of a class.

We present a novel semantic smoothing method named Higher Order Smoothing (HOS) for Naive Bayes algorithm. HOS is built on a novel graph based data representation which allows us to exploit semantic information in higher-order paths. HOS exploits the relationships between instances of different classes in order to improve the parameter estimation in the face of sparse data. As a result, we do not only move beyond instance boundaries but also class boundaries to exploit the latent information in higher-order co-occurrence paths.

We have performed extensive experiments on several benchmark textual datasets and compared HOS with several different state of the art classifiers. HOS significantly outperforms the baseline classifier Naive Bayes using different smoothing methods including Laplace smoothing and Jelinek-Mercer smoothing, in all datasets under different training data conditions. Furthermore, it even outperforms Support Vector Machines (SVM) by a wide margin in the well-known 20 Newsgroups dataset. Our results demonstrate the value of HOS as a semantic smoothing algorithm.

As future work, we are planning to perform more detailed analysis in order to understand the reasons for the improved performance of HOS. Additionally, we would like to get insights about under which conditions and what type of datasets HOS performs well. We are also planning to advance the current higher-order learning framework, which works on binary data, so it can make use of term frequencies or weighted term counts such as tf-idf. Several studies emphasize the importance of different varieties of normalizations such as document-length normalization in improving Naive Bayes performance [24], [25], [26]. Thus, we as well would like to analyze HOS by incorporating document length and weight normalization in the future.

ACKNOWLEDGMENT

This work was supported in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 111E239. Points of view in this document are those of the authors and do not necessarily represent the

official position or policies of the TÜBİTAK. Authors wish to thank Dilara Torunoğlu and Işıl Çoşkun for their help.

REFERENCES

[1] B. Taskar, P. Abbeel, and D. Koller, "Discriminative Probabilistic Models for Relational Data," Proc. Uncertainty in Artificial Intelligence (UAI02), Morgan Kaufmann 2002, August 2002, pp. 485-492.

[2] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," Proc. International Conference on Management of Data (ACM SIGMOD), ACM Press, June 1998, pp. 307-318.

[3] J. Neville, and D. Jensen, "Iterative classification in relational data," Proc. AAAI 2000 Workshop on Learning Statististical Models from Relational Data (LSR), AAAI Press 2000, July 2000, pp. 13-20.

[4] L. Getoor, and C. P. Diehl, "Link Mining: A Survey," Proc. International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD), ACM Press, August 2005, pp. 3-12.

[5] M. Ganiz, W. M. Pottenger, S. Kanitkar, and M. C. Chuah, "Detection of Interdomain Routing Anomalies Based on Higher-Order Path Analysis," Proc. IEEE International Conference on Data Mining (ICDM), IEEE Computer Society 2006, December 2006, pp. 874-879.

[6] M. Ganiz, N. Lytkin, and W. M. Pottenger, "Leveraging Higher Order Dependencies between Features for Text Classification," Proc. European Conference on Machine Learning and Pronciples and Practice of Knowledge Discovery in Databases (ECML/PKDD), Springer, September 2009, pp. 375-390.

[7] M. Ganiz, C. George, and W. M. Pottenger, "Higher Order Naïve Bayes: A Novel Non-IID Approach to Text Classification," IEEE Transactions on Knowledge and Data Engineering, vol. 23, July 2011, pp. 1022-1034.

[8] N. Lytkin, "Variance-based clustering methods and higher order data transformations and their applications," Ph.D. Thesis, Rutgers University, NJ, U.S.A. (2009)

[9] A. Edwards, and W. M. Pottenger, "Higher order Q-Learning," Proc. Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), IEEE Press 2011, vol. 1, April 2011, pp.128-134.

[10] S. Deerwester, S. T. Dumais, and R. Harshman, "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science, vol. 41, December 1990, pp. 391-407.

[11] A. Kontostathis, and W. M. Pottenger, "A Framework for Understanding LSI Performance", Journal of the Information Processing and Management, vol. 42, January 2006, pp. 56-73.

[12] S. Li, T. Wu, and W. M. Pottenger, "Distributed higher order association rule mining using information extracted from textual data," in SIGKDD Explorations Newsletter, vol. 7, June 2005, pp. 26-35.

[13] A. McCallum, and K. Nigam, "Comparison of Event Models for Naive Bayes Text Classification," Proc. AAAI 1998 Workshop on Learning for Text Categorization, AAAI Press 1998, July 1998, pp. 41-48.

[14] K. M. Schneider, "On Word Frequency Information and Negative Evidence in Naive Bayes Text Classification," Proc. International Conference on Advances in Natural Language Processing (EsTAL), Springer 2004, October 2004, pp. 474–485.

[15] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam Filtering with Naive Bayes – Which Naive Bayes?," Proc.Conference on Email and Anti-Spam (CEAS), July 2006.

[16] A. McCallum, and K. Nigam, "Text classification by bootstrapping with keywords, EM and shrinkage," in Working Notes of ACL 1999 Workshop for the Unsupervised Learning in Natural Language Processing (ACL), June 1999, pp. 52-58.

[17] A. Juan, and H. Ney, "Reversing and Smoothing the Multinomial Naive Bayes Text Classifier," Proc. Internatinal Workshop on Pattern Recognition in Information Systems (PRIS), ICEIS Press 2002, April 2002, pp. 200–212.

[18] F. Peng, D. Schuurmans, and S. Wang, "Augmenting Naive Bayes Classifiers with Statistical Language Models," Information Retrieval, vol. 7, January 2004, pp. 317–345, 2004.

[19] X. Zhou, X. Zhang, and X. Hu, "Semantic smoothing for Bayesian text classification with small training data", Proc. International Conference on Data Mining (SIAM), April 2008, pp. 289–300.

[20] S. F. Chen, and J. Goodman, "An Emprical Study of Smoothing Techniques for Language Modeling," Technical Report, Harvard University Center for Research in Computing Technology, 1998.

[21] S. Chakrabarti, "Mining the Web: Discovering Knowledge from Hypertext Data,", Morgan Kaufmann Publishers, 2002, pp. 148-151.

[22] M. F. Amasyalı, and A. Beken, "Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi Ve Metin Sınıflandırmada Kullanılması", Proc IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SIU), IEEE Press, April 2009, pp.

[23] D. Torunoğlu, E. Çakırman, M. C. Ganiz, S. Akyokuş, M. Z. Gürbüz, "Analysis of Preprocessing Methods on Classification of Turkish Texts," Proc. International Symposium on Innovations in Intelligent Systems and Applications (INISTA), IEEE Press, June 2011, pp. 112-118.

[24] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," Proc. International Conference on Machine Learning (ICML), AAAI Press 2003, August 2003, pp. 616-623.

[25] S. Eyheramendy, D. D. Lewis, and D. Madigan, "On the Naive Bayes Model for Text Categorization," Proc. International Workshop on Artificial Intelligence and Statistics (AISTATS), , January 2003, pp.332-339.

[26] A. Kolcz, and W. Yih, "Raising the Baseline for High-Precision Text Classifiers," Proc. International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD), ACM Press, August 2007, pp.400-409.

[27] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," Proc. European Conference on Machine Learning (ECML), Springer, April 1998, pp.137-142.

[28] S.-B. Kim, K.-S. Han, H.-C. Rim, and S.-H. Myaeng. "Some Effective Techniques for Naive Bayes Text Classification," IEEE Trans. Knowl. Data Eng., 18(11):1457–1466, 2006.

[29] B. Gao, T. Liu, G. Feng, T. Qin, Q. Cheng, and W. Ma, "Hierarchical Taxonomy Preparation for Text Categorization Using Consistent Bipartite Spectral Graph Co-partitioning," IEEE Transactions on Knowledge and Data Engineering, vol.17, no. 9, pp. 1263-1273, September 2005.

[30] Manning, C. D. and H. Schütze, "Foundations of Statistical Natural Language Processing". MIT Press, Cambridge, MA, 1999.