Poyraz M, Kilimci ZH, Ganiz MC. Higher-order smoothing: A novel semantic smoothing method for text classification. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(3): 376–391 May 2014. DOI 10.1007/s11390-014-1437-6

Higher-Order Smoothing: A Novel Semantic Smoothing Method for Text Classification

Mitat Poyraz, Zeynep Hilal Kilimci, and Murat Can Ganiz*, *Member*, *IEEE*

Department of Computer Engineering, Dogus University, Istanbul 34722, Turkey

E-mail: {mpoyraz, hkilimci, mcganiz}@dogus.edu.tr

Received September 1, 2013; revised March 11, 2014.

Abstract It is known 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". These findings have inspired a novel Bayesian framework for classification named Higher-Order Naive Bayes (HONB), which was introduced previously, that can explicitly make use of these higher-order relations. In this paper, 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 the HONB which allows semantics in higher-order paths to be exploited. We take the concept one step further in HOS and exploit the relationships between instances of different classes. As a result, we move beyond not only instance boundaries, but also class boundaries to exploit the latent information in higher-order paths. This approach improves the parameter estimation when dealing with insufficient labeled data. 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

1 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, entities and their attributes are highly interconnected either by explicit or implicit relations. Consequently, the IID approach does not fully make use of valuable information about these relationships. There are several studies which exploit explicit link information in order to overcome the shortcomings of IID approach[1-4]. However, the use of explicit links has a significant drawback. In order to classify a single instance, an additional context needs to be provided. On the other hand, higher-order learning approach addresses this drawback by exploiting the implicit relations during parameter estimation in the training phase. In general, it is a statistical relational learning framework, which allows supervised and unsupervised algorithms to leverage relationships between attributes and instances. This approach makes use of implicit link information[5-9]. Using implicit link information within data provides a richer data representation. In real world applications, it is often rather difficult and usually expensive to obtain labelled data. Therefore it is of great importance for classification systems to perform reasonably well using a small amount of labelled data. Using implicit links is known to be effective especially when we have limited labelled data. A novel Bayesian framework for classification named Higher-Order Naive 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[6-8]. These implicit links are defined as higher-order paths. Higher-order paths of this kind richly connect attributes or features such as terms in documents of a text collection. HONB exploits this rich connectivity[6].

In the present study, we follow the same practice of exploiting implicit link information via developing a no-

Regular Paper

This work was supported in part by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant No. 111E239. Points of views in this document are those of the authors and do not necessarily represent the official position or policies of the TÜBİTAK.

A preliminary version of this paper was published in the Proceedings of ICDM 2012.

*Corresponding Author

vel semantic smoothing method for Naive Bayes (NB). We call it Higher-Order Smoothing (HOS). HOS is built upon a novel graph-based data representation, which was inspired from the data representation in HONB. Moreover, we improve the concept by taking it one step further to exploit the relationships between instances of different classes in HOS. Thus HOS moves beyond the instance boundaries and the class boundaries to fully exploit the latent information in higher-order paths.

We have performed extensive experiments by varying the size of the training set in order to simulate real world settings and compare our algorithm with several existing smoothing methods and other classification algorithms. Our results on several benchmark datasets show that HOS significantly increases the performance of NB. Furthermore, it can even outperform support vector machines (SVM) on some datasets.

The rest of the article is organized as follows. In Section 2, we briefly review the related work. In Section 3, we provide background information on NB, smoothing methods, higher-order data representations and algorithms. Based on these backgrounds, we present our approach in Section 4. Subsequently, our experimental setup is described in Section 5, while our results are presented in Section 6. These are then followed by discussion in Section 7. Finally, in Section 8, we provide conclusion remarks and future work directions.

2 Related Work

The Latent Semantic Indexing (LSI) algorithm[10] is a widely used technique in text mining and information retrieval. 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 LSI-based classifiers. Among these, in [12] the authors propose an LSI-based k-Nearest Neighborhood (LSI k-NN) algorithm in a semi-supervised setting for short text classification. This is one of the straightforward uses of LSI in text classification. In this study, authors use the k-Nearest Neighborhood (k-NN) algorithm which is based on calculating distance or similarities between training instances and a test instance in the transformed LSI space. They set the number of neighbors to 30 and use the noisy-or operator. A similar approach is used in a supervised setting to build an LSI-based k-NN algorithm as one of the baseline algorithms in [7]. In this study, the number of neighbors is set to 25, and the dimension parameter (k) of the LSI algorithm is optimized.

The LSI algorithm and its ability to reveal latent semantics make one of our primary motivations. However, there are several difficulties in using LSI for text classification; it is a highly complex, unsupervised, black box algorithm. In our study we attempt to use higher-order paths explicitly in a supervised setting. Our second motivation stems from the studies in link mining that utilize explicit links[4]. Several studies in this domain have shown that significant improvements can be achieved by classifying multiple instances collectively[1-3]. 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 several studies, which exploit implicit link information in order to improve the performance of machine learning algorithms[5-7]. Using the implicit link information within data provides a richer data representation and it is shown to be effective especially under the scarce training data conditions.

A novel Bayesian framework named Higher-Order Naive Bayes (HONB) has been introduced in [6-7]. HONB employs a graph-based data representation and leverages the co-occurrence relations between attribute values across different instances. These implicit links are called higher-order paths. Such higher-order paths richly connect attributes or features such as the terms in the documents of a text collection. HONB exploits this rich connectivity[7]. 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 has led to the development of Higher-Order Support Vector Machines (HOSVM) algorithm. The higher-order learning framework is a statistical relational learning framework. It consists of several supervised and unsupervised machine learning algorithms in which relationships between different instances are leveraged via higher-order paths[8-9,13].

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 an higher-order path that links term A with term D through B and C. This is a third-order path

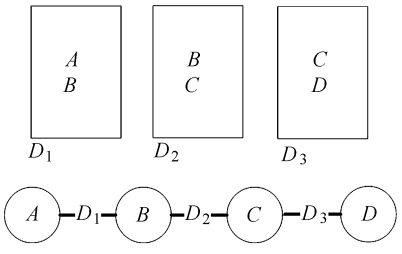


Fig.1. Higher-order co-occurrence[11].

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 relationships such as synonymy.

It is well known that NB and SVM are the most popular algorithms in text classification. There are two commonly referred event models in NB 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 are represented as "1" and "0", respectively. On the other hand, the multinomial model is a unigram language model with integer term counts. Thus, each class can be defined as a multinomial distribution[14]. McCallum and Nigam[14] compared multivariate Bernoulli and multinomial models on several different datasets. Their experimental results confirm that the multivariate Bernoulli event model shows better performance at smaller vocabulary sizes, whereas the multinomial model generally performs well with large vocabulary sizes[14].

In another study[15], the authors propose a multivariate Poisson Naive Bayes text classification model with weight-enhancing method to improve performances on rare categories. Their experiments show that, the proposed model is a good alternative to traditional Naive Bayes classifier because it allows more reasonable parameter estimation.

Majority of the text classification studies which use the Naive Bayes algorithm, employ multinomial model based on the well-known contribution by McCallum and Nigam[14]. It is generally assumed that the term frequency data is a richer representation of documents than the binary word occurrences.

However, there are several studies in the literature that report contradictory results about the benefit of using term frequencies. For instance, MNB is shown to perform better with binary data in a number of cases such as spam detection[16-17]. Similarly, our results on several datasets also reveal contradictory results and they show that the binary term frequencies and the multivariate binary models can outperform algorithms operating on term frequency data on several cases.

In general, the NB parameter estimation drastically suffers from the presence of sparsity due to the very large number of parameters to estimate in the text classification. The number of parameters corresponds to $(|V||C|+|C|)$ where V denotes the dictionary and C denotes the set of class labels[18]. The sparsity of the text data increases the importance of smoothing methods since they are intended to distribute a certain amount of probability mass to the previously unseen events.

Most of the studies on NB text classification use the Laplace smoothing by default. Only a few studies attempt to use different smoothing methods. For instance, the authors of [19] used multinomial model with several different smoothing techniques which origin from the statistical language modeling field and are generally used with n-gram language models. These include absolute discounting with unigram backing-off and absolute discounting with unigram interpolation. They stated that the absolute discounting with unigram interpolation gives better results than the Laplace smoothing. They also used document length normalization. In [20], the authors 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. A semantic smoothing method based on the extraction of topic signatures was proposed in [22]. The topic signatures consist of multi-word phrases such as the n-grams or collocations that are extracted from the training corpus. After having the topic signatures and multiword phrases extracted, the authors used them in a semantic smoothing background collection model to smooth and map the topic signatures. They demonstrated that when the training data is small, the NB classifier with semantic smoothing outperforms the NB with background smoothing (Jelinek-Mercer) and the NB with Laplace smoothing.

SVM is a popular large margin classifier. This machine learning method aims to find a decision boundary that separates points into two classes by maximizing the margin[23]. SVM can project data points to a higher dimensional space by using kernel techniques so that the data points become linearly separable. There are several kernels that can be used with the SVM algorithm. The linear kernel is known to perform well on the text classification since many text categorization problems are linearly separable[23]. We include the SVM results on both binary and term frequency data in our experiments for comparing our method with a well-accepted text classifier.

There are only a small number of studies that employ tripartite graph in text categorization. One of these[24] proposed a new clustering algorithm to automatically mine hierarchical taxonomy from the dataset in order to take advantage of hierarchical classifier. The method is called Consistent Bipartite Spectral Graph Co-Partitioning (CBSGC) algorithm, which is based on generalized singular value decomposition. According to the study, document-term and category-document bipartite graphs reflect only partial information of the

data. Therefore, the approach co-clusters the category, document, and term into a tripartite graph in order to leverage the complementary information contained in them. The experiments show that, CBSGC discovers a reasonable hierarchical taxonomy and as a result it improves the classification accuracy.

A recent graph-based study[25] introduces the concept of distance graphs representation of documents instead of traditional vector space representation. It shows that the distance graphs provide additional information about the ordering and distance between terms compared with the vector space representation. This new representation is similar to our higher-order approach in creating a graph representation of words and the higher-order paths between terms. We can observe that they are creating a directed graph representation with self-loops, which indicates the frequency and the order of words in the context of a document. On the other hand, our approach is based on an undirected graph representation of words in the context of the training set of documents. In order to illustrate the advantages of the distance graphs, the authors of [25] conducted experiments on three well-known datasets such as the 20 Newsgroups, Reuters-21578, and WebKB. They also investigated different applications including the classification, clustering, and similarity search. They leveraged the distance graph models with three different classification algorithms and two different clustering algorithms. They also tested the cosine similarity with the use of the vector-space models, including unigram, bigram, and trigram, as well as the distance graph models of different orders ranging from one to four. They showed that the distance graphs perform better than the vector space models for the three datasets.

The authors of a recent paper[26] focused on question classification on fine-grained taxonomies. They presented a novel algorithm named as DC2. DC2 aims to improve the question classification performance and it is based on the divergence of the probability distributions of terms (Poisson distribution). The technique was applied to a large corpora based on class related fragments for question classification in English and Spanish and compared with several supervised methods including SVM.

Nguyen *et al.*[27] focused the properties of the CFC (class feature centroid) classifier especially on the bias toward the large classes in class imbalance problems. In order to overcome this bias in skewed class distributions, the authors proposed an improvement. Instead of weighting terms by presence or absence in the classes, they used the Kullback-Leibler (KL) divergence measure between the pairs of class conditional term probabilities for binary class datasets. They used Jensen-Shannon (JS) divergence for multiclass datasets. They evaluated their technique on three binary datasets (movie review, sentiment polarity, and multi-domain sentiment) and two multiclass datasets (Reuters-21578 and 20 Newsgroups). Additionally, they developed a word cloud visualization approach to highlight the important class specific words that are selected by their term weighting method. They noted that these words are not ranked high by unsupervised term weighting. This approach is similar to our approach in a way that HOS actually gives more emphasis and weight to the terms that have more relations with class labels compared with HONB.

3 Background

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

3.1 Naive Bayes Event Models

There are two generative event models that are commonly used with NB for text classification. The first and the less popular one is the 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. The class conditional document probability in (1) is calculated by using the individual class conditional term probabilities of the document in (2)[28]:

$$P(d|c_j) = \prod_{w \in d} \frac{P(w_i|c_j)}{1 - P(w_i|c_j)} \prod_{w \in W} (1 - P(w_i|c_j)), \tag{1}$$

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

In (2), the class conditional term probability corresponds to 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 (2) are introduced according to Laplace's rule of succession in order to avoid the zero probability problem. The Laplace smoothing adds a pseudo count to every word count. The main disadvantage of Laplace is giving too much probability mass to previously unseen events[29].

The second NB event model is the multinomial model (MNB) which can make use of term frequencies.

Suppose term w_i occurs $n(d, w_i)$ times in document d, which is said to have length $\ell_d = \sum_{w_i} n(d, w_i)$. With this definition the class conditional document probability is calculated as in (3)

$$P(d|c_j) = P(L = \ell_d|c_j)P(d|\ell_d, c_j) = P(L = \ell_d|c_j) \binom{\ell_d}{\{n(d, w_i)\}} \prod_{w_i \in d} \theta_t^{n(d,w_i)}. \quad (3)$$

On the other hand, the class conditional term probabilities are estimated using (4).

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

where $|W|$ is the vocabulary size (i.e., the total number of words)[28].

Because of the sparsity in the 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 previously unseen terms. This process is generally known as smoothing. The most common smoothing method in NB is the Laplace smoothing. Formulas of the NB event models in (2) and (4) include the Laplace smoothing. In the next subsection, we will provide details of a more advanced smoothing method which performs well especially on MVNB.

3.2 Jelinek-Mercer Smoothing

In the Jelinek-Mercer smoothing method, the maximum estimate is interpolated with the smoothed lower order distribution[22]. This is achieved by the linear combination of maximum likelihood estimate (5) with the collection model (6) as shown in (7). In (6), D represents the whole training set, including documents from all the classes.

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

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

$$P(w_i|c_j) = (1 - \beta) \times P_{ml}(w_i|c_j) + \beta \times P(w_i|D). \quad (7)$$

3.3 Higher-Order Data Representation

The graph-based data representation we built upon is initially used in [7]. It is noted that the definition of a higher-order path is similar to the path definition in graph theory, which states that given a non-empty graph $G = (V, E)$ of the form $V = \{x_0, x_1, \ldots, x_k\}$, $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$ and given that the nodes in V are 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. It builds a bipartite graph $G = ((V_D, V_W), E)$ from a set of D documents. In this graph, vertices in 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].

3.4 Higher-Order Naive Bayes

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

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

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

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

4 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[7]. However, in HONB, higher-order paths are extracted in the context

of a particular class. Therefore, we cannot exploit the relations between the terms and the documents in the different classes.

In this study, 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 moving beyond not only 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 the 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 $Class = \{course, faculty, project, staff, student\}$. We add these four class labels as new terms (i.e., columns to our document by term matrix). We call them "class terms". Each of these class terms 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 V_c correspond to class terms or in other words class labels. Fig.2 shows a tripartite graph which represents relationship between terms, class labels, and documents. Similar to the previous higher-order data representation with bipartite graph, a higher-order path in dataset D can be considered as a chain sub-graph of $\widehat{G}$. However, we are interested in such chain sub-graphs which start with a term vertex from V_W, span 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 a more dense data representation than the traditional vector space representation. This makes the basis of our improved smoothing algorithm.

Let us 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 bold dotted lines and it corresponds to a second-order path. In this example let us 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 do not occur in the 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].

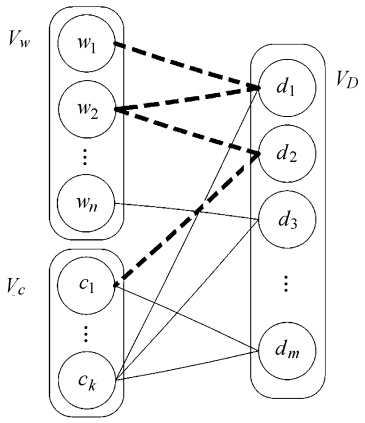


Fig.2. Data representation for higher-order paths using a tripartite graph.

Based on this representation and modified definition of higher-order paths we can formulate the HOS. Let $\partial(w_i, c_j)$ denote the number of higher-order paths that are 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 the class term c_j 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 (10). Although HOS has the potential to estimate parameters for terms that do not occur in the documents of a class but occurs in other classes in training data, there can be terms that occur only in test instances. In order to avoid zero probability problems in these cases, we apply Laplace smoothing in (10). Class priors are calculated according to multivariate Bernoulli model using documents.

$$P(w_i|c_j) = \frac{1 + \partial(w_i, c_j)}{2 + \Phi(D)}. \tag{10}$$

We noticed 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 orders of paths. (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 the best.

$$P(w_i|c_j) = (1-\beta) \times P_{fo}(w_i|c_j) + \beta \times P_{so}(w_i|c_j), \quad (11)$$

where *fo* means fist-order and *so* means second-order.

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 described in more details in [8].

Algorithm 1. Enumerating Second-Order Paths for HOS
Input: Boolean document-term data matrix $\boldsymbol{X} = ||X_d^t||$
Output: $\boldsymbol{O}_2$: matrix which stores the number of second-order paths in data matrix $\boldsymbol{X}$

1. Initialize vector $\boldsymbol{l} = (l^1, \ldots, l^n)$, which will store class labels of $\boldsymbol{X}$
2. **for** each row i in data matrix $\boldsymbol{X}$
   $l^i = ||X_i^{t-1}||$
3. Initialize class labels binary matrix $\boldsymbol{C}_{lb} = ||C_{lb_d}^c||$ which will represent each class value as binary where c is the number of classes in $\boldsymbol{X}$
4. **for** each row i in $\boldsymbol{C}_{lb}$ matrix
   for each column $\boldsymbol{c}$ in $\boldsymbol{C}_{lb}$ matrix
   if l^i is equal to j
   Set $\boldsymbol{C}_{lb}$ equal to $\boldsymbol{1}$
5. Compute matrix $\boldsymbol{X}_{clb} = ||X_{clb_d}^{t+c}||$ by appending binary class valued matrix $\boldsymbol{C}_{lb}$ to $\boldsymbol{X}$
6. Compute first-order co-occurrence matrix $\boldsymbol{O}_1 = \boldsymbol{X}_{clb}^{\mathrm{T}} \boldsymbol{X}_{clb}$
7. Compute second-order co-occurrence matrix $\boldsymbol{O}_2 = \boldsymbol{O}_1 \boldsymbol{O}_1$
8. **for** each row i in first-order co-occurrence matrix $\boldsymbol{O}_1$
   for each column j in first-order co-occurrence matrix $\boldsymbol{O}_1$
   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 the same
   $s = O_2(i,j) - (O_1(i,j) \times (O_1(i,i) + O_1(j,j)))$
   Update the element of second-order co-occurrence matrix, $O_2(i,j) = O_2(i,j) + s$
10. Return $\boldsymbol{O}_2$

In the algorithm above, at first, class labels are removed from the document by term data matrix and stored in a vector. Following this a binary class labels matrix is built. In the binary class labels matrix rows represent class values of documents as binary vectors where the index position of the class label is 1 and the other positions 0. Afterwards, the binary class labels matrix is combined with original data matrix $\boldsymbol{X}$. This results in a new matrix called class-binarized data matrix ($\boldsymbol{X}_{clb}$), which stores the input data matrix and its binary class values. We use $\boldsymbol{X}_{clb}$ 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 $\boldsymbol{X}_{clb}$ (term by document matrix) and $\boldsymbol{X}_{clb}$ (document by term matrix). The 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. The scalar s represents the number of paths like t_1, d_1, t_2, d_1, t_3, where both document vertices (d_1 in the example above) are the same. We filter these types of paths by subtracting s from the total paths between two terms.

5 Experimental Setup

In order to analyze the performance of our algorithm for text classification, we use seven datasets including several widely used benchmark datasets. The first one is a variant of the 20 Newsgroups① dataset. It is called 20News-18828 and it has fewer documents than the original 20 Newsgroup dataset since duplicate postings are removed. Additionally posting headers are deleted except "From" and "Subject".

Our second dataset is the WebKB② dataset, which includes web pages collected from computer science departments of different universities. There are seven categories, namely 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.

The third dataset is 1150Haber dataset which consists of 1 150 news articles in five categories namely economy, magazine, health, politics and sport collected from Turkish online newspapers[30].

①http://people.csail.mit.edu/people/jrennie/20Newsgroups, Mar. 2014.
②http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/, Mar. 2014.

In addition to these three major datasets we use four additional datasets especially in normalization experiments. Among these, Milliyet4c1k dataset includes documents from the columns of a popular Turkish newspaper called Milliyet[③] from 2002 to 2011. It contains four categories and 1 000 documents for each category. The categories of this dataset are dünya (world), ekonomi (economy), siyaset (politics), and spor (sports).

Our next Turkish dataset is named as Hurriyet6c1k and it includes news documents from 2010 to 2011 published in Hürriyet[④] which is a mainstream Turkish newspaper. It contains six categories and 1 000 documents for each category. Categories in this dataset are dünya (world), ekonomi (economy), gündem (current), spor (sports), siyaset (politics), and yaşam (life).

We use setimes-tr and setimes-en datasets collected from SETimes[⑤] website which publishes news about Southeast Europe in English and several other languages including Turkish. It is important to note that setimes-en and setimes-tr includes exactly the same news content but in different languages. Therefore it can be considered as an aligned corpus at document level. This type of bi-lingual datasets allows us to observe performance of text classification algorithms on the content written in different languages. This is important because even the basic data statistics in Table 1 and Table 2 illustrate interesting differences between Turkish and English document sets, such as the vocabulary sizes and average (Avg.) and standard deviation (StdDev.) of term lengths.

Table 1. Descriptions of the Datasets Without Preprocessing

Dataset	$\|C\|$	$\|D\|$	$\|V\|$
20News-18828	20	18 828	50 570
WebKB4	4	4 199	16 116
1150Haber	5	1 150	11 038
Hurriyet6c1k	6	6 000	24 228
Milliyet4c1k	4	4 000	39 174
Setimes-tr	7	7 000	16 456
Setimes-en	7	7 000	10 373

We particularly choose datasets in different languages in order to observe efficiency of the higher-order algorithms on different languages. Similar to LSI, we expect higher-order paths based algorithms HONB and HOS to perform well on a different language without any need of tuning. More information about dataset 1150Haber and text classification on Turkish documents can be found in [31].

Table 2. Data Characteristics of the Datasets Without Preprocessing

Dataset	Number of Terms in Documents		Term Length		Document
	Avg.	StdDev.	Avg.	StdDev.	Sparsity
20News-18828	47.24	44.41	5.35	2.61	0.98
WebKB4	81.68	49.95	5.34	2.97	0.96
1150Haber	57.23	37.37	5.81	3.22	0.97
Hurriyet6c1k	50.47	27.41	5.78	3.20	0.97
Milliyet4c1k	137.82	42.80	5.74	3.14	0.93
Setimes-tr	65.86	36.88	5.93	3.29	0.97
Setimes-en	53.28	29.24	5.92	2.88	0.97

Hurriyet6c1k, Milliyet4c1k, setimes-tr, and setimes-tr are compiled by our research group and can be obtained for research purposes from our web site[⑥].

We use a wide variety of datasets with different features. 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 has highly skewed class distribution.

We employ commonly used preprocessing methods such as stop word removal, stemming, and attribute selection. We also filter infrequent terms whose document frequency is less than three. However, statistics given in Table 1 are acquired from the datasets before applying stemming, stop word filtering, and attribute selection in order to understand their nature especially in terms of the vocabulary sizes. Descriptions of the datasets, under these conditions are given in Table 1 including the number of classes ($|C|$), number of documents ($|D|$), and the vocabulary size ($|V|$).

As can be seen from Algorithm 1, the complexity of the higher-order path enumeration algorithm is proportional to the number of terms or in other words the vocabulary size. In order to 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. Stop word lists and Snowball stemmer have 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 they are known to improve the performance of traditional vector space

③http://www.milliyet.com.tr, Mar. 2014.

④http://www.hurriyet.com.tr, Mar. 2014.

⑤http://www.setimes.com, Mar. 2014.

⑥http://dmrl.ce.dogus.edu.tr/datasets.html, Mar. 2014.

classifiers. Please note that HOS is expected to work well when the data is sparse and these preprocessing operations reduce sparsity. For that reason, we are actually giving a considerable advantage to our baseline classifiers (NB and SVM).

As mentioned before we vary the training set size by using the following percentages of the data for training: 1%, 5%, 10%, 30%, 50%, 70%, and 90%, and the rest for testing. These percentages are indicated with "TS" 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 sets, we employ stratified sampling. This approach is similar to [14] and [32] where they use 80% of the data for training and 20% for testing.

Our datasets include term frequencies (TF). However, higher-order paths based classifiers HONB and HOS can intuitively work with binary data. This simplifies the path counting since a binary term-by-term matrix can be translated into an un-weighted and undirected graph. As a result, we form a binary version of each dataset by converting term frequencies to binary occurrence values in order to enumerate higher-order paths. We do not consider word weighting schemas such as TFIDF since our main focus is on binary data as mentioned above.

We employ up to second-order paths based on the experimental results of previous studies[6-7,11].

Since we use binary term occurrences, 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 called Jelinek-Mercer (JM).

It is not easy to use HOS in a multinomial Naive Bayes setting since it changes the nature of our tripartite graph structure which is undirected and unweighted. The choice of using the term frequencies transforms it to a weighted graph. In this graph an edge between a term and document is weighted by the occurrence frequency of the term in that document. Consequently, counting the number of paths that satisfies our path definition is not straightforward anymore. Furthermore, it is not clear how to calculate weights in higher-order path structure since a relation edge between two terms includes two different term frequency values. Solving these challenges is out of the context of this paper. Yet, we attempt to use HOS with Multinomial Naive Bayes (MNB+HOS) on term frequency data in a straightforward way by omitting the step 8 of Algorithm 1. This means that we are relaxing one of the restrictions on path definition. These initial experiments are intended for comparison purposes with other algorithms such as SVM and MNB. Additionally, we also apply several normalizations in order to see the effect of them in its performance. For JM and HOS, we fix β to 0.5 experimentally since for majority of our datasets and training set size percentages this value performs the best.

Furthermore, we compare our results with HONB and the state-of-the-art text classifier SVM and LSI-based k-Nearest Neighborhood algorithm (LSI k-NN) using the similar approach in [7]. Similarly, the number of neighbors is set to 25 and the k parameter of LSI is automatically optimized for each dataset and training set percentage by using the Kaiser approach. SVM, which operates binary term occurrences datasets, is indicated as "SVM binary" and the one that uses term frequency data is simply indicated as "SVM" in the results. "SVM binary" alias should not be confused with concept of the binary SVM which performs a binary classification of two classes. Instead both "SVM binary" and "SVM" are multiclass SVMs. We use linear kernel in SVM since it is known to perform well in text classification. This is also consistent with the results of our initial experiments. 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 pick the smallest value of C, which results in the highest accuracy. We observe that the best performing value for C is usually 1 or 10^{-1} depending on the dataset when the training data is small (e.g., up to 10%) and 10^{-2} when training data increases (e.g., after 10%).

Several studies emphasize the importance of different varieties of normalizations such as document-length normalization in improving Naive Bayes performance[32-34]. Thus, we conduct several experiments to analyze the performance of HOS by incorporating document length and matrix level normalizations for the first-order and second-order term matrices. The document normalization is applied by dividing the term frequencies to their maximum value in the document vector.

We observe that the values in the second-order term matrix are much larger than those in the first-order term matrix. As a result, we apply min-max normalization to each matrix.

6 Experimental Results

We use average accuracy values of 10 random trial experiments with varying training set size percentages. Accuracy is our main evaluation metric and it is aug-

mented by standard deviations in the result tables below. Additionally, we employ 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). However, we report these values only for 80% training data level due to the length restrictions. Additionally, we observe that $F1$ and AUC values are similar to the accuracy values.

There are several approaches in evaluating machine learning algorithms[35]. In addition to the evaluation metrics, we 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, if the accuracy values of HOS are higher than the values of our baselines (MVNB and MVNB+JM), they are indicated with bold font. Additionally, if the difference between the results of HOS and our baselines are statistically significant according to t-test, it is indicated by a dot (•) near by the accuracy value. This is similar to the representation in [36].

6.1 Binary Model Results

Our experiments show that HOS demonstrates remarkable performance on 20 Newsgroups dataset. This can be clearly seen in Table 3 showing the performance of the 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. The performance improvement is especially visible at low training set size levels. It is important to note that 20 Newsgroups dataset is one of the most commonly used datasets in text mining domain.

Table 3. Accuracy and Standard Deviations of Algorithms on 20 Newsgroups Dataset with Varying Training Set Size

TS	MVNB	MVNB+JM	MVNB+HOS	HONB
1	24.77 ± 2.49•	48.01 ± 1.37	$\mathbf{42.92 \pm 3.61}$	44.09 ± 2.04
5	55.68 ± 1.26•	69.10 ± 0.68	$\mathbf{65.81 \pm 1.57}$	64.65 ± 0.92
10	65.01 ± 1.57•	72.95 ± 1.42•	$\mathbf{76.70 \pm 0.79}$	69.93 ± 0.62
30	72.83 ± 0.74•	75.66 ± 0.63•	$\mathbf{81.97 \pm 0.33}$	76.12 ± 0.38
50	75.11 ± 0.58•	76.64 ± 0.68•	$\mathbf{83.06 \pm 0.29}$	78.53 ± 0.37
70	75.65 ± 0.64•	76.81 ± 0.67•	$\mathbf{83.33 \pm 0.54}$	79.92 ± 0.34
90	76.21 ± 1.18•	76.50 ± 1.02•	$\mathbf{83.26 \pm 0.84}$	80.11 ± 0.65

Table 4 shows the performance of HOS on WebKB4 dataset. Although it is 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 the MVNB+JM smoothing starting from 30% training set level.

Table 4. Accuracy and Standard Deviations of Algorithms on WebKB4 Dataset with Varying Training Set Size

TS	MVNB	MVNB+JM	MVNB+HOS	HONB
1	44.48 ± 1.03	69.96 ± 3.15	30.08 ± 6.56	70.58 ± 3.80
5	68.17 ± 2.49	79.33 ± 2.15	61.15 ± 6.51	77.68 ± 2.94
10	74.46 ± 1.36•	80.76 ± 1.54	$\mathbf{77.71 \pm 2.33}$	80.83 ± 1.35
30	81.53 ± 1.05•	83.02 ± 0.92•	$\mathbf{85.24 \pm 0.75}$	86.83 ± 0.58
50	82.57 ± 0.83•	82.81 ± 0.81•	$\mathbf{86.08 \pm 0.55}$	87.64 ± 0.75
70	83.53 ± 0.98•	83.19 ± 1.08•	$\mathbf{87.01 \pm 0.87}$	88.53 ± 0.75
90	84.17 ± 2.10•	83.41 ± 1.61•	$\mathbf{87.01 \pm 1.20}$	88.36 ± 1.42

Interestingly, HONB performs slightly well than HOS on this dataset. We attribute the better performance of HONB to the skewed class distribution of the dataset. This is one of the main differences of the WebKB4 dataset from other datasets.

The performance of HOS on the 1150Haber dataset, which can be seen in Table 5, is similar to its performance on the 20 Newsgroups dataset. HOS statistically significantly outperforms the baseline algorithms: starting from 10% training set level for MVNB and starting from 30% level for MVNB+JM. HONB and HOS show a very similar performance on this dataset with the exception of small training set levels (i.e., up to 30%) where the HONB performs better. This may be attributed to the much larger number of paths generated by HONB compared with HOS since 1150Haber is our smallest dataset including 230 news documents per class. After 30% level the differences between accuracies of HONB and HOS are not statistically significant. It is interesting to observe similar behavior of HOS in datasets with different properties in different languages. For instance, 20 Newsgroups dataset contains highly noisy and informal use of language in newsgroups pos-

Table 5. Accuracy and Standard Deviations of Algorithms on 1150Haber Dataset with Varying Training Set Size

TS	MVNB	MVNB+JM	MVNB+HOS	HONB
1	35.70 ± 7.64	48.40 ± 5.04	32.09 ± 11.1	30.32 ± 12.7
5	65.06 ± 12.6	81.01 ± 6.95	$\mathbf{67.00 \pm 11.9}$	88.25 ± 0.93
10	72.95 ± 3.83•	86.01 ± 2.03	$\mathbf{83.13 \pm 4.12}$	91.61 ± 0.85
30	87.64 ± 1.14•	91.49 ± 0.71•	$\mathbf{93.79 \pm 0.31}$	94.20 ± 0.59
50	88.73 ± 0.65•	91.10 ± 0.63•	$\mathbf{94.42 \pm 0.42}$	94.73 ± 0.57
70	89.97 ± 0.88•	91.39 ± 0.83 •	$\mathbf{95.01 \pm 0.85}$	95.30 ± 0.96
90	90.78 ± 2.73•	91.48 ± 2.42•	$\mathbf{94.35 \pm 2.14}$	95.22 ± 1.75

tings in English. On the other hand 1150Haber includes relatively more formal use of language in mainstream newspaper column articles in Turkish.

Improvements are most visible in 20 Newsgroups dataset. HOS improves upon MVNB and SVM about 17% in terms of accuracy at 10% training set size (TS) level on the 20 Newsgroups dataset. We can observe improvements in all training set size levels on this dataset.

In addition to the detailed accuracy results on several different training set levels, we present the results of additional evaluation metrics at the 80% training set level. We choose this training set level since it is commonly used in random trial experiments[14,32].

Table 6 shows F-measure ($F1$) performance of the algorithms at 80% training set level. The similar trend of outperforming the baseline algorithms can also be seen in here. HOS statistically significantly outperforms both of the baseline classifiers (MVNB and MVNB+HOS) for all datasets. The performance improvements are especially visible on 20 Newsgroups and 1150Haber datasets.

Table 6. F-Measure Performance of the Algorithms at 80% Training Set Level

Algorithm	20 News-18828	WebKB4	1150Haber
HONB	79.96 ± 0.75	88.34 ± 0.97	95.92 ± 1.60
MVNB+HOS	$\mathbf{83.02 \pm 0.72}$	$\mathbf{85.32 \pm 1.74}$	$\mathbf{94.95 \pm 1.84}$
MVNB	$76.41 \pm 0.59\bullet$	$82.80 \pm 1.23\bullet$	$89.79 \pm 2.40\bullet$
MVNB+JM	$77.39 \pm 0.81\bullet$	$82.43 \pm 1.31\bullet$	$90.96 \pm 2.50\bullet$
SVM Binary	$78.81 \pm 0.81\bullet$	91.28 ± 1.28	$92.06 \pm 2.72\bullet$

Table 7 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 7. AUC Performance of the Algorithms at 80% Training Set Level

Algorithm	20 News-18828	WebKB4	1150Haber
HONB	98.18 ± 0.07	97.58 ± 0.27	99.57 ± 0.24
MVNB+HOS	$\mathbf{98.57 \pm 0.09}$	$\mathbf{96.90 \pm 0.46}$	$\mathbf{99.56 \pm 0.25}$
MVNB	$97.67 \pm 0.17\bullet$	$96.17 \pm 0.51\bullet$	99.25 ± 0.38
MVNB+JM	$97.74 \pm 0.19\bullet$	$96.19 \pm 0.54\bullet$	99.43 ± 0.31
SVM Binary	$88.69 \pm 0.38\bullet$	$94.35 \pm 0.67\bullet$	$95.03 \pm 1.70\bullet$

Fig.3 compares MVNB+HOS with two state-of-the-art classifiers: SVM and LSI k-NN. As we can see from the figure MVNB+HOS outperforms both SVM and LSI k-NN by a wide margin on 1150Haber binary term occurrence dataset. Due to the extremely high complexity of LSI k-NN algorithm we could obtain results from the training set percentage level 1% to 90% only for 1150Haber dataset. We conduct experiments for other datasets up to a certain TS level (e.g. up to ts50 for WebKB4, milliyet4k1c, hurriyet6k1c, and up to ts10 for 20 Newsgroups, setimes-tr and setimes-en). However, these results are consistent with the 1150Haber results showing that MVNB+HOS perform better than LSI k-NN in general.

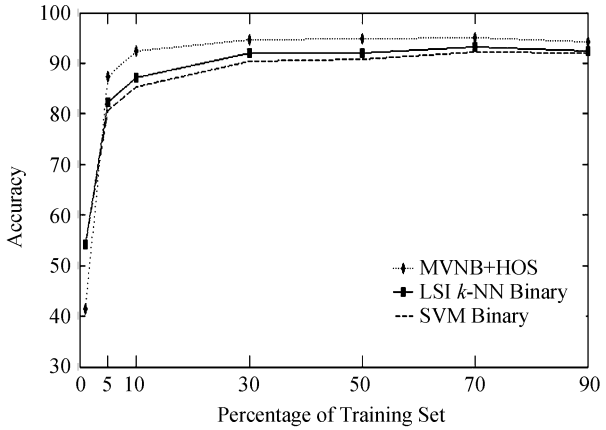


Fig.3. Accuracy comparison of MVNB+HOS, LSI k-NN and SVM algorithms on 1150Haber binary dataset.

6.2 Normalization Experiments on Multinomial and Binary Models

In this subsection we report the results of a multinomial version of HOS which uses the term frequencies in a multinomial Naive Bayes setting. These results are presented in Tables 8∼21. We use multinomial data to advance the current higher-order learning framework, which works on binary data, so it can make use of term frequencies. Multinomial models of HOS at this very basic level show promises on small training set percentages. However, it requires more work to establish a better framework.

Additionally we provide the results for several normalization approaches for both multivariate binary (MVNB+HOS) and multinomial models (MNB+HOS).

Several studies emphasize the importance of different varieties of normalizations such as document-length normalization in improving the performance of NB[32-34]. Thus, we conduct several experiments to analyze HOS by incorporating document length normalization and matrix level normalizations for first-order and second-order term matrices.

In Table 8, Table 10, and Table 12 we can see the results of the normalization for the multivariate binary HOS (MVNB+HOS) and compare it with the SVM algorithm which is applied to the binary word occurrence data. In Table 8, Table 9, and Table 12 for MNB+HOS

Table 8. Accuracy and Standard Deviations of Algorithms on 20News-18828 Binary Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	17.02 ± 4.96	$\mathbf{36.47 \pm 5.11}$	39.02 ± 1.79
5	$\mathbf{65.81 \pm 1.57}$	61.34 ± 2.16	$58.32 \pm 1.19\bullet$
10	$\mathbf{76.70 \pm 0.79}$	65.03 ± 2.72	$66.76 \pm 0.81\bullet$
30	$\mathbf{81.97 \pm 0.33}$	68.53 ± 1.56	$74.24 \pm 0.72\bullet$
50	$\mathbf{83.06 \pm 0.29}$	69.98 ± 0.86	$77.31 \pm 0.70\bullet$
70	$\mathbf{83.33 \pm 0.54}$	70.47 ± 0.56	$79.98 \pm 0.45\bullet$
90	$\mathbf{83.26 \pm 0.84}$	70.72 ± 0.68	$81.35 \pm 1.17\bullet$

Table 9. Accuracy and Standard Deviations of Algorithms on 20News-18828 TF Dataset with Varying Training Set Size

TS	MNB+HOS$^{(1)}$	MNB+HOs$^{(3)}$	SVM
1	54.33 ± 1.41	$\mathbf{56.59 \pm 1.43}$	32.65 ± 1.75
5	73.15 ± 0.75	$\mathbf{74.05 \pm 0.55}$	56.16 ± 1.11
10	76.92 ± 0.90	$\mathbf{77.45 \pm 0.31}$	65.15 ± 0.61
30	80.17 ± 0.51	79.47 ± 0.29	75.99 ± 0.61
50	81.11 ± 0.34	80.09 ± 0.20	79.35 ± 0.34
70	81.57 ± 0.40	80.28 ± 0.41	81.53 ± 0.32
90	81.45 ± 0.81	80.63 ± 0.92	82.38 ± 1.15

Table 10. Accuracy and Standard Deviations of Algorithms on WebKB4 Binary Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	27.04 ± 12.28	$\mathbf{59.84 \pm 9.26}$	$68.42 \pm 3.17\bullet$
5	61.15 ± 6.51	$\mathbf{75.66 \pm 2.17}$	$81.59 \pm 2.35\bullet$
10	77.71 ± 1.33	$\mathbf{80.10 \pm 1.61}$	$84.54 \pm 1.33\bullet$
30	$\mathbf{85.24 \pm 0.75}$	81.11 ± 1.22	$89.38 \pm 1.03\bullet$
50	$\mathbf{86.08 \pm 0.55}$	81.08 ± 0.67	$90.41 \pm 0.63\bullet$
70	$\mathbf{87.01 \pm 0.87}$	82.07 ± 0.94	$91.62 \pm 0.83\bullet$
90	$\mathbf{87.01 \pm 1.20}$	81.25 ± 1.73	$92.08 \pm 1.01\bullet$

Table 11. Accuracy and Standard Deviations of Algorithms on WebKB4 TF Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(3)}$	SVM
1	69.54 ± 2.86	66.36 ± 3.63	60.57 ± 1.82
5	76.52 ± 5.00	$\mathbf{79.89 \pm 0.82}$	79.01 ± 1.33
10	78.04 ± 5.17	$\mathbf{82.53 \pm 1.18}$	83.48 ± 1.14
30	83.04 ± 2.35	$\mathbf{84.37 \pm 0.70}$	89.43 ± 0.55
50	82.84 ± 1.07	$\mathbf{84.05 \pm 0.69}$	91.04 ± 0.47
70	83.45 ± 0.98	$\mathbf{84.28 \pm 0.97}$	91.69 ± 0.72
90	83.45 ± 1.62	$\mathbf{83.95 \pm 1.57}$	92.20 ± 1.00

Table 12. Accuracy and Standard Deviations of Algorithms on 1150Haber Binary Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	26.97 ± 8.26	$\mathbf{41.49 \pm 10.66}$	54.04 ± 4.93
5	49.26 ± 24.16	$\mathbf{87.26 \pm 3.18}$	$80.73 \pm 2.56\bullet$
10	61.60 ± 11.80	$\mathbf{92.41 \pm 0.59}$	$85.36 \pm 1.02\bullet$
30	92.99 ± 0.49	$\mathbf{94.59 \pm 0.43}$	$90.36 \pm 1.37\bullet$
50	94.26 ± 0.70	94.92 ± 0.75	$90.85 \pm 0.88\bullet$
70	95.01 ± 0.85	95.10 ± 0.89	$92.17 \pm 1.22\bullet$
90	94.34 ± 2.13	94.26 ± 2.05	$91.91 \pm 1.92\bullet$

or MVNB+HOS we can see that the performance of HOS can exceed the performance of SVM. The performance improvement is visible starting from 5% training set level for both 20 News-18828 and 1150Haber, which can be seen in Table 8 and Table 12 respectively. It is important to note that the c parameter of SVM is optimized in these results.

For our additional datasets of Hurriyet6c1k, Milliyet4c1k, setimes-tr, and setimes-en, we observe slightly different patterns for SVM. MVNB+HOS usually outperforms SVM on small training set levels.

In this subsection, HOS$^{(1)}$ indicates no normalization, HOS$^{(2)}$ shows term matrix level normalization, and HOS$^{(3)}$ indicates both term matrix and document normalizations.

In general, term matrix normalizations that are indicated with HOS$^{(2)}$ do not improve the performance of MVNB+HOS as we can see in Table 8, Table 10, Table 12, Table 14, Table 16, Table 18, and Table 20. They

Table 13. Accuracy and Standard Deviations of Algorithms with Multinomial Model on 1150Haber Dataset with Varying Training Set Size

TS	MNB+HOS$^{(1)}$	MNB+HOS$^{(3)}$	SVM
1	$\mathbf{59.35 \pm 4.99}$	54.21 ± 3.33	38.92 ± 3.03
5	88.58 ± 1.08	$\mathbf{89.40 \pm 1.22}$	67.47 ± 4.24
10	92.17 ± 0.40	$\mathbf{92.65 \pm 0.35}$	76.27 ± 2.71
30	94.24 ± 0.48	$\mathbf{94.32 \pm 0.58}$	87.39 ± 1.21
50	$\mathbf{94.55 \pm 0.86}$	94.46 ± 0.97	89.55 ± 1.12
70	94.98 ± 1.06	$\mathbf{95.21 \pm 1.12}$	90.55 ± 1.49
90	$\mathbf{94.52 \pm 2.46}$	94.17 ± 2.78	90.78 ± 1.93

Table 14. Accuracy and Standard Deviations of Algorithms with Binary Model on Hurriyet6c1k Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	39.08 ± 11.83	$\mathbf{60.13 \pm 1.63}$	$52.73 \pm 1.49\bullet$
5	70.74 ± 1.12	$\mathbf{72.03 \pm 0.57}$	$64.56 \pm 1.35\bullet$
10	74.28 ± 0.68	74.64 ± 0.70	$68.84 \pm 0.47\bullet$
30	76.20 ± 0.36	76.11 ± 0.37	$73.85 \pm 0.77\bullet$
50	76.86 ± 0.46	76.48 ± 0.49	75.60 ± 0.56
70	76.70 ± 0.59	76.36 ± 0.47	76.67 ± 0.60
90	77.25 ± 1.72	76.86 ± 1.69	77.56 ± 2.10

Table 15. Accuracy and Standard Deviations of Algorithms with Multinomial Model on Hurriyet6c1k Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(3)}$	SVM
1	58.45 ± 1.72	$\mathbf{60.18 \pm 1.76}$	40.47 ± 6.22
5	70.71 ± 0.81	$\mathbf{71.27 \pm 0.48}$	60.63 ± 2.25
10	73.57 ± 0.51	$\mathbf{73.96 \pm 0.68}$	66.30 ± 0.94
30	$\mathbf{75.82 \pm 0.50}$	$\mathbf{75.66 \pm 0.38}$	73.05 ± 0.53
50	$\mathbf{76.46 \pm 0.70}$	76.19 ± 0.77	75.34 ± 0.53
70	$\mathbf{76.38 \pm 0.60}$	75.86 ± 0.61	76.43 ± 0.63
90	$\mathbf{77.01 \pm 1.78}$	75.86 ± 1.89	77.96 ± 1.48

Table 16. Accuracy and Standard Deviations of Algorithms on Binary Milliyet4c1k Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	68.43 ± 7.48	$\mathbf{82.34 \pm 1.14}$	$76.01 \pm 3.13\bullet$
5	86.48 ± 0.51	$\mathbf{86.95 \pm 0.37}$	$83.93 \pm 1.06\bullet$
10	87.55 ± 0.45	$\mathbf{87.65 \pm 0.47}$	85.51 ± 0.76
30	$\mathbf{87.92 \pm 0.33}$	87.77 ± 0.40	88.76 ± 1.00
50	$\mathbf{88.11 \pm 0.69}$	87.95 ± 0.72	90.04 ± 0.74
70	$\mathbf{88.03 \pm 0.70}$	87.81 ± 0.79	90.55 ± 1.05
90	$\mathbf{87.47 \pm 1.55}$	87.37 ± 1.51	90.70 ± 0.90

Table 17. Accuracy and Standard Deviations of Algorithms with Multinomial Model on Milliyet4c1k Dataset with Varying Training Set Size

TS	MNB+HOS$^{(1)}$	MNB+HOS$^{(3)}$	SVM
1	80.40 ± 2.10	$\mathbf{81.21 \pm 1.26}$	72.76 ± 3.35
5	85.61 ± 0.53	85.99 ± 0.55	84.05 ± 0.71
10	86.77 ± 0.47	86.86 ± 0.46	87.12 ± 0.65
30	87.11 ± 0.44	86.97 ± 0.51	90.60 ± 0.47
50	87.27 ± 0.50	87.16 ± 0.57	92.06 ± 0.38
70	87.13 ± 0.87	86.95 ± 0.84	92.14 ± 0.67
90	86.52 ± 1.62	86.45 ± 1.66	93.45 ± 1.01

Table 18. Accuracy and Standard Deviations of Algorithms on Binary Setimes-Tr Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	24.03 ± 6.92	$\mathbf{38.92 \pm 2.28}$	$38.32 \pm 1.34\bullet$
5	48.02 ± 1.23	$\mathbf{50.01 \pm 0.91}$	49.98 ± 1.20
10	51.42 ± 0.49	$\mathbf{51.87 \pm 0.40}$	53.80 ± 0.43
30	$\mathbf{54.01 \pm 0.34}$	53.72 ± 0.44	56.90 ± 0.48
50	$\mathbf{54.08 \pm 0.62}$	53.73 ± 0.56	57.31 ± 0.45
70	$\mathbf{54.00 \pm 0.84}$	53.61 ± 0.81	57.55 ± 0.65
90	$\mathbf{54.84 \pm 2.32}$	54.31 ± 2.09	57.87 ± 1.85

Table 19. Accuracy and Standard Deviations of Algorithms with Multinomial Model on Setimes-Tr Dataset with Varying Training Set Size

TS	MNB+HOS$^{(1)}$	MNB+HOS$^{(3)}$	SVM
1	$\mathbf{40.82 \pm 1.85}$	40.55 ± 2.12	36.41 ± 1.66
5	$\mathbf{50.62 \pm 1.03}$	50.47 ± 1.03	48.96 ± 1.29
10	$\mathbf{52.19 \pm 0.56}$	52.28 ± 0.51	53.04 ± 0.32
30	$\mathbf{54.24 \pm 0.43}$	53.88 ± 0.43	55.60 ± 0.55
50	$\mathbf{54.16 \pm 0.49}$	53.58 ± 0.46	55.73 ± 0.52
70	$\mathbf{54.28 \pm 0.66}$	53.87 ± 0.79	55.61 ± 0.93
90	$\mathbf{55.50 \pm 2.31}$	54.22 ± 2.17	56.55 ± 1.36

only increases the performance at very low training set percentage levels such as 1% to 10% but degrade the performance above these levels.

However, the picture is quite different in terms of the normalizations applied to the multinomial HOS (MNB+HOS). In general, MNB+HOS performs better when matrix normalization and document normalization are applied together (MNB+HOS$^{(3)}$). This is especially the case for smaller training set percentages.

Table 20. Accuracy and Standard Deviations of Algorithms on Binary Setimes-En Dataset with Varying Training Set Size

TS	MVNB+HOS$^{(1)}$	MVNB+HOS$^{(2)}$	SVM Binary
1	26.56 ± 3.26	$\mathbf{38.85 \pm 1.48}$	39.20 ± 1.91
5	48.04 ± 1.32	$\mathbf{49.44 \pm 1.25}$	51.35 ± 0.90
10	51.05 ± 0.76	$\mathbf{51.36 \pm 0.74}$	54.63 ± 0.56
30	$\mathbf{53.78 \pm 0.59}$	53.33 ± 0.61	57.67 ± 0.53
50	$\mathbf{53.97 \pm 0.51}$	53.19 ± 0.51	$58.42 \pm 0.73\bullet$
70	$\mathbf{54.03 \pm 1.00}$	52.95 ± 0.92	$58.44 \pm 0.50\bullet$
90	$\mathbf{54.38 \pm 2.30}$	53.60 ± 2.17	$58.80 \pm 1.62\bullet$

Interestingly, if matrix normalization is applied alone, it can seriously degrade the performance of MNB+HOS. Therefore we omit the results of MNB+HOS$^{(2)}$, term matrix level normalization from result tables.

In terms of comparison between MNB+HOS and SVM on term frequency data, we observe a similar pattern to MVNB+HOS and SVM on binary term occurrence data. For 20News-18828 and 1150Haber MNB+HOS outperforms SVM by a wide margin especially in small training set percentages as can be seen in Table 9 and Table 13. A similar performance difference in small training set percentages can be seen in Table 15 for Hurriyet6c1k dataset. However, we observe an opposite pattern in WebKB4, setimes, and Milliyet4c1k datasets in Table 11, Table 17, Table 19, and Table 21. In these datasets SVM performs better than Naive Bayes models.

Table 21. Accuracy and Standard Deviations of Algorithms on Setimes-En TF Dataset with Varying Training Set Size

TS	MNB+HOS$^{(1)}$	MNB+HOS$^{(3)}$	SVM
1	40.36 ± 1.06	$\mathbf{40.67 \pm 1.05}$	37.61 ± 2.34
5	$\mathbf{50.54 \pm 1.36}$	50.38 ± 1.17	50.52 ± 1.24
10	$\mathbf{52.55 \pm 0.77}$	52.25 ± 0.46	53.91 ± 0.64
30	$\mathbf{54.65 \pm 0.37}$	53.45 ± 0.46	56.97 ± 0.38
50	$\mathbf{54.72 \pm 0.58}$	53.76 ± 0.57	57.14 ± 0.58
70	$\mathbf{54.62 \pm 0.69}$	53.48 ± 0.81	57.40 ± 0.51
90	$\mathbf{55.47 \pm 2.62}$	54.71 ± 2.04	57.61 ± 1.90

Interestingly, the performance of SVM that operates on binary data (indicated as "SVM binary") is very close to the performance of SVM that operates on term frequency data.

As mentioned before, we optimize the parameters of SVM. Interestingly, we have varying results for the best c value. It is hard to draw a general conclusion for the best c value. The c value which yields best accuracies drastically changes from dataset to dataset and even for training set percentages. Furthermore, using binary term occurrence data or term frequency data changes the best c value.

On the 1150Haber dataset, the best results are obtained with MVNB+HOS$^{(2)}$ at small training size levels up to 50% as can be seen from Table 12. Furthermore,

the SVM on binary data exceeds the SVM on term frequency data at almost at every training set size level.

The normalizations applied to MNB+HOS does very little or no effect on the performance in setimes-en, setimes-tr, Milliyet4c1k experiments. In general, normalization methods on MNB+HOS do make a very little difference (about 1% or 2%) on small training set percentages such as 1%, 5%, and 10%.

In addition to these results, we also compare our method with an LSI-based classification algorithm on a term frequency dataset. In Fig.4 we can see that MNB+HOS outperforms both SVM and LSI k-NN by a wide margin on 1150Haber term frequency dataset.

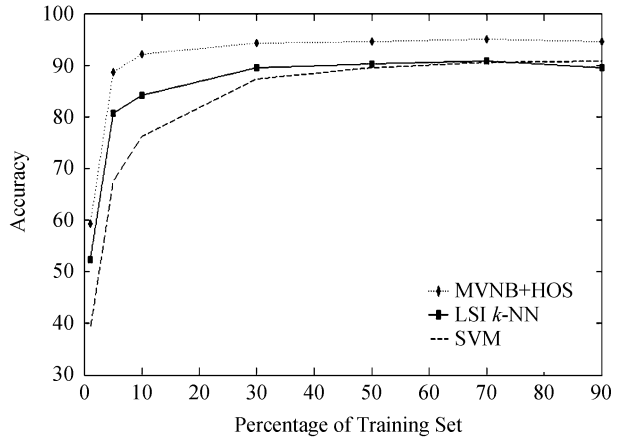


Fig.4. Accuracy comparison of MNB+HOS, LSI k-NN and SVM algorithms on 1150Haber term frequency dataset.

7 Discussion

The use of higher-order paths for estimation of conditional term probabilities have been discussed in [7-8]. It is observed that highly discriminative terms exhibit much stronger influence on classification by HONB than by NB[8]. Additionally, HONB tends to place more emphasis on the presence of terms in a document being classified[7-8]. Since HOS is based on the higher-order paths, it enjoys some of these benefits. However, in HOS we are enumerating much fewer 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 provides a higher performance on small training set levels (i.e., 1% and 5%) compared with HOS. Yet, HOS quickly catches up about at 10% level and outperforms HONB especially on the 20News-18828 dataset. This particular dataset has a relatively large number of classes (having 20 classes compared with 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 since the higher-order paths must end with a class label. It would be interesting to observe the performance of HOS on real world datasets with much larger number of categories.

Results on 1150Haber, Hurriyet6c1k, Milliyet4c1k, and setimes-tr datasets suggest that HOS may also perform well on different languages than English without additional language specific tuning. This is similar to LSI. This can be an important advantage for HOS compared with the natural language processing based semantic methods such as [21].

In terms of training time complexity, an $O(n^2(m+n))$ algorithm is given in previous studies for obtaining amounts of higher-order paths in a dataset with m instances and n dimensions[6,8]. In the training phase, HONB forms a document by term matrix $(m \times n)$ for each class, and uses this algorithm to obtain amounts of higher-order paths between all terms.

Our approach, which is given in Algorithm 1, suggests using the same principles; therefore, it has similar computational complexity. When we examine Algorithm 1 more closely, the computational complexity is dominated by computation matrices $\boldsymbol{O}_1$ and $\boldsymbol{O}_2$, therefore the complexity is $O(mn^2 + n^3)$.

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

Extensive experiments demonstrate that normalization methods can increase the performance of HOS especially on binary data (MVNB+HOS). In general, matrix normalizations can improve the performance of MVNB+HOS on small training set percentages. On the other hand, multinomial model shows similar pattern almost on every dataset. It is either no normalization or the combination of matrix and document normalization leads to the best results of MNB+HOS.

In order to understand the reasons for the better performance of HOS over its baseline, we find some correlations between the average number of distinct words in documents and document sparsity measured using the number of zeroes in document vector of document by term matrix. We speculate that HOS and other algorithms of higher-order learning framework work better when both the average number of terms in documents are around 50 to 80 and average document sparsity is relatively high (95%∼97%). In addition to these, we argue that the relatively high number of classes in a large dataset such as 20News-18828 dataset boost the performance of HOS.

It is also interesting to observe in our results that on several datasets the algorithms working on binary term occurrence data can indeed outperform algorithms operating on term frequency data in several cases. This is

similar to the findings of [16-17]. For instance, SVM working on binary data can exceed the performance of the SVM that is working on the same dataset but with term frequencies. This phenomenon can be seen in the WebKB4, setimes-en, setimes-tr, Hurriyet6c1k, and 1150Haber datasets. The difference is more visible in the Turkish datasets and higher training set levels. On the contrary, MNB almost always demonstrates better performance than MVNB. It is important to note that this is not the case for MVNB+HOS. MVNB+HOS can exceed the performance of MNB and MNB+HOS in several cases.

There are several studies on text mining and information retrieval that employ term weightings. For instance, the authors of [37] experimented with different term weighting functions in the context of LSA performance. It would be interesting to apply weighting methods in HOS and analyze their influence as future work.

8 Conclusions

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 Naive Bayes (HONB) was introduced in [7]. HONB can explicitly make use of these higher-order co-occurrence relations.

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

We performed extensive experiments on several benchmark textual datasets and compared MVNB+HOS with several different state-of-the-art classifiers. MVNB+HOS significantly outperforms the baseline classifier of Naive Bayes using different smoothing methods including Laplace smoothing and Jelinek-Mercer smoothing, in all datasets under different training data conditions. When we compared HOS with a similar semantic classifier of LSI k-NN which is based on LSI, we could see that HOS outperforms LSI k-NN by a wide margin in 1150Haber dataset. Furthermore, it even outperforms SVM by a wide margin in the well-known 20Newsgroup, 1150Haber, and Hurriyet6c1k datasets. Our results demonstrate the value of HOS as a semantic smoothing algorithm.

Acknowledgements Authors wish to thank anonymous reviewers for their valuable feedback.

References

[1] Taskar B, Abbeel P, Koller D. Discriminative probabilistic models for relational data. In *Proc. the 18th Conf. Uncertainty in Artificial Intelligence*, August 2002, pp.485-492.

[2] Chakrabarti S, Dom B, Indyk P. Enhanced hypertext categorization using hyperlinks. In *Proc. International Conference on Management of Data*, June 1998, pp.307-318.

[3] Neville J, Jensen D. Iterative classification in relational data. In *Proc. AAAI 2000 Workshop on Learning Statistical Models from Relational Data*, July 2000, pp.13-20.

[4] Getoor L, Diehl C P. Link mining: A survey. *ACM SIGKDD Explorations Newsletter*, 2005, 7(2): 3-12.

[5] Ganiz M C, Kanitkar S, Chuah M C, Pottenger W M. Detection of interdomain routing anomalies based on higher-order path analysis. In *Proc. the 6th IEEE International Conference on Data Mining*, December 2006, pp.874-879.

[6] Ganiz M C, Lytkin N, Pottenger W M. Leveraging higher order dependencies between features for text classification. In *Proc. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, September 2009, pp.375-390.

[7] Ganiz M C, George C, Pottenger W M. Higher order Naive Bayes: A novel non-IID approach to text classification. *IEEE Trans. Knowledge and Data Engineering*, 2011, 23(7): 1022-1034.

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

[9] Edwards A, Pottenger W M. Higher order Q-Learning. In *Proc. IEEE Symp. Adaptive Dynamic Programming and Reinforcement Learning*, April 2011, pp.128-134.

[10] Deerwester S C, Dumais S T, Landauer T K *et al.* Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990, 41(6): 391-407.

[11] Kontostathis A, Pottenger W M. A framework for understanding latent semantic indexing (LSI) performance. *Journal of the Information Processing and Management*, 2006, 42(1): 56-73.

[12] Sarah Z, Hirsh H. Transductive LSI for short text classification problems. In *Proc. the 17th International Florida Artificial Intelligence Research Society Conference*, May 2004, pp.556-561.

[13] Li S, Wu T, Pottenger W M. Distributed higher order association rule mining using information extracted from textual data. *SIGKDD Explorations Newsletter — Natural Language Processing and Text Mining*, 2005, 7(1): 26-35.

[14] McCallum A, Nigam K. A comparison of event models for Naive Bayes text classification. In *Proc. AAAI 1998 Workshop on Learning for Text Categorization*, July 1998, pp.41-48.

[15] Kim S B, Han K S, Rim H C, Myaeng S H. Some effective techniques for naive Bayes text classification. *IEEE Trans. Knowl. Data Eng.*, 2006, 18(11): 1457-1466.

[16] Schneider K M. On word frequency information and negative evidence in Naive Bayes text classification. In *Proc. Int. Conf. Advances in Natural Language Processing*, October 2004, pp.474-485.

[17] Metsis V, Androutsopoulos I, Paliouras G. Spam filtering with Naive Bayes — Which Naive Bayes?. In *Proc. Conference on Email and Anti-Spam*, July 2006.

[18] McCallum A, Nigam K. Text classification by bootstrapping with keywords, EM and shrinkage. In *Proc. ACL 1999 Workshop for the Unsupervised Learning in Natural Language Processing*, June 1999, pp.52-58.

[19] Juan A, Ney H. Reversing and smoothing the multinomial Naive Bayes text classifier. In *Proc. International Workshop on Pattern Recognition in Information Systems*, April 2002, pp.200-212.

[20] Peng F, Schuurmans D, Wang S. Augmenting naive Bayes classifiers with statistical language models. *Information Retrieval*, 2004, 7(3/4): 317-345.

[21] Zhou X, Zhang X, Hu X. Semantic smoothing for Bayesian text classification with small training data. In *Proc. International Conference on Data Mining*, April 2008, pp.289-300.

[22] Chen S F, Goodman J. An empirical study of smoothing techniques for language modeling. In *Proc. the 34th Annual Meeting on Association for Computational Linguistics*, June 1996, pp.310-318

[23] Joachims T. Text categorization with support vector machines: Learning with many relevant features. In *Proc. the 10th European Conf. Machine Learning*, Apr. 1998, pp.137-142.

[24] Gao B, Liu T, Feng G, Qin T, Cheng Q, Ma W. Hierarchical taxonomy preparation for text categorization using consistent bipartite spectral graph co-partitioning. *IEEE Transactions on Knowledge and Data Engineering*, 2005, 17(9): 1263-1273.

[25] Aggarwal C C, Zhao P. Towards graphical models for text processing. *Knowledge and Information Systems*, 2013, 36(1): 1-21.

[26] Tomás D, Vicedo J L. Minimally supervised question classification on fine-grained taxonomies. *Knowledge and Information Systems*, 2013, 36(2): 303-334.

[27] Nguyen T T, Chang K, Hui S C. Supervised term weighting centroid-based classifiers for text categorization. *Knowledge and Information Systems*, 2013, 35(1): 61-85

[28] Chakrabarti S. Supervised learning. In *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, 2002, pp.148-151.

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

[30] AmasyalıM F, Beken A. Measurement of Turkish word semantic similarity and text categorization application. In *Proc. IEEE Signal Processing and Communications Applications Conference*, April 2009. (in Turkish)

[31] Torunoğlu D, Çakırman E, Ganiz M C *et al.* Analysis of preprocessing methods on classification of Turkish texts. In *Proc. International Symposium on Innovations in Intelligent Systems and Applications*, June 2011, pp.112-118.

[32] Rennie J D, Shih L, Teevan J, Karger D R. Tackling the poor assumptions of Naive Bayes text classifiers. In *Proc. ICML2003*, August 2003, pp.616-623.

[33] Eyheramendy S, Lewis D D, Madigan D. On the Naive Bayes model for text categorization. In *Proc. the 9th International Workshop on Artificial Intelligence and Statistics*, January 2003, pp.332-339.

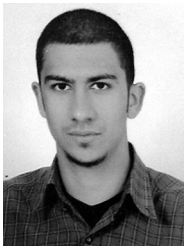
[34] Kolcz A, Yih W. Raising the baseline for high-precision text classifiers. In *Proc. the 13th Int. Conf. Knowledge Discovery and Data Mining*, August 2007, pp.400-409.

[35] Japkowicz N, Shah M. Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, 2011.

[36] Su J, Shirab J S, Matwin S. Large scale text classification using semi-supervised multinomial Naive Bayes. In *Proc. the 28th Int. Conf. Machine Learning*, June 2011, pp.97-104.

[37] Nakov P, Popova A, Mateev P. Weight functions impact on LSA performance. In *Proc. the EuroConference Recent Advances in Natural Language Processing*, September 2001, pp.187-193.

[38] Poyraz M, Kilimci Z H, Ganiz M C. A novel semantic smoothing method based on higher order paths for text classification. In *Proc. IEEE Int. Conf. Data Mining*, Dec. 2012, pp.615-624.



Mitat Poyraz is a business intelligence consultant at QlikView Turkey. Before joining the QlikView Turkey, he was a research assistant and Masters student at Computer Engineering Department of Dogus University, Istanbul, Turkey. His research interests are textual data mining, machine learning algorithms, and semantic smoothing approaches.

Zeynep Hilal Kilimci is a Ph.D. student and research assistant at Computer Engineering Department of Dogus University, Istanbul, Turkey. Her research interests are data management, data mining, text mining, and data warehousing.

Murat Can Ganiz received the Ph.D. degree in computer science from the Lehigh University, in 2008. He is currently an assistant professor in Computer Engineering Department of Dogus University, Turkey, which he joined in 2009. His research interests include machine learning, text mining, sentiment analysis and opinion mining, data mining, link analysis, and social network analysis. His company, VeriUs Technology, Inc., creates innovative data analytics, sentiment analysis and opinion mining technology. He is a member of the IEEE.