Higher Order Naïve Bayes: A Novel Non-IID Approach to Text Classification

Murat Can Ganiz, Cibin George and William M. Pottenger, *Member, IEEE*

Abstract **The underlying assumption in traditional machine learning algorithms is that instances are I.I.D., Independent and Identically Distributed. These critical independence assumptions made in traditional machine learning algorithms prevent them from going beyond instance boundaries to exploit latent relations between features. In this article, we develop a general approach to supervised learning by leveraging higher-order dependencies between features. We introduce a novel Bayesian framework for classification termed Higher Order Naïve Bayes (HONB). Unlike approaches that assume data instances are independent, HONB leverages higher order relations between features across different instances. The approach is validated in the classification domain on widely-used benchmark datasets. Results obtained on several benchmark text corpora demonstrate that higher-order approaches achieve significant improvements in classification accuracy over the baseline methods, especially when training data is scarce. A complexity analysis also reveals that the space and time complexity of HONB compare favorably with existing approaches.**

Index Terms **Machine Learning, Statistical Relational Learning, Naïve Bayes, Text Classification, IID.**

I. INTRODUCTION

A well known problem in real-world applications of machine learning is that expert labeling of large amounts of data for training a classifier is prohibitively expensive. Often in practice, only a small amount of labeled data is available for training. In this case, however, making an adequate estimation of the model parameters of a classifier is challenging. Underlying this issue is the traditional assumption in machine learning algorithms that instances are I.I.D.: independent and identically distributed [1]. This assumption simplifies the underlying mathematics of statistical models, but in fact does not hold for many real world applications [2]. As a result of this assumption, models constructed under the I.I.D. assumption do not leverage connections between the attributes in different instances. A well-known example is market basket analysis, which forms sets of items that are purchased together in a given context such as a market basket. The advantage to this approach is that classification of a single instance of previously unseen

William M. Pottenger and Cibin George are with Rutgers University, Piscataway, NJ 08854 USA (telephone: (732) 445-5930, e-mail: drwmp@cs.rutgers.edu).

Murat Can Ganiz is with Doğuş University, Istanbul, Turkey (telephone: (+90 216) 5445555, e-mail: mcganiz@dogus.edu.tr).

data is possible because no additional context is needed to infer class membership [3]. However, such a context-free approach does not exploit valuable information about relationships between instances in the dataset [4].

Recently, a new domain termed *link mining* has emerged [2]. The focus of the research in this new domain is on datasets of instances that are explicitly linked, for example, by hyperlinks. Datasets with explicit links of this nature have been termed networked data [3]. One of the areas of algorithm research in this domain is link-based object classification, where the task is to classify instances connected to each other via a set of explicit links of this nature. Such approaches leverage both the attributes contained in the instances and the explicit links between instances, making the underlying assumption that labels of the linked objects tend to be correlated. Collective classification algorithms operate on networked data by jointly inferring class labels of a set of instances based on this underlying assumption. Several classification algorithms of this nature have been proposed that exploit the explicit links between instances (e.g., [5], [6], [7] and [8]). As noted, they utilize class labels of related instances when labeling a test instance and usually reduce classification error compared to flat (i.e., I.I.D.) classifiers. Models built in this way can be thought of as *second-order* because explicit links between instances are leveraged during model construction. Such second-order connections are based on explicit links between instances such as hyperlinks between web pages, citation links between scientific papers, etc.

In this work we focus on the development and analysis of a supervised machine learning algorithm that does not make the I.I.D. assumption, but instead moves beyond instance boundaries to exploit the latent information in higher-order co-occurrence paths between features within datasets. Unlike existing approaches, however, we do not rely on explicit links between instances when learning a model, but rather leverage implicit links. In what follows we define these implicit links.

We often refer to the terms "higher-order path," "higher-order link" or "higher-order co-occurrence" in this article. In order to understand the meaning of these terms, consider Fig. **1** below (reproduced from [9]). This figure depicts three documents, D1, D2 and D3, each containing two terms, or entities, represented by the letters A, B, C and D. Below the three documents there is a higher-order path that links entity A with entity D through B and C. This is a third-order path since three links, or "hops," connect A and D.

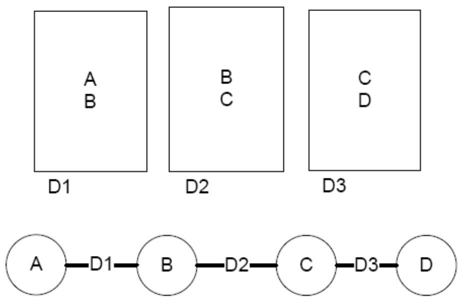


Fig. 1. Higher-order co-occurrence

A higher-order path can also be represented as a chain of co-occurrences of entities (attribute values, words, terms, etc.) in different records (instances, documents, etc.). Actually we can extract co-occurrence relations from virtually any dataset as long as there is a meaningful context of entities.

In general, we are interested in exploiting the latent information in such higher-order paths. We are motivated by the Latent Semantic Indexing (LSI) algorithm [10], which is a widely used technique in text mining and IR based on singular value decomposition matrix factoring. This approximation reduces the noise and sparsity of the original term-vector space and partially solves the synonymy problem. The authors note that the fundamental problem in IR is that existing methods try to match words of queries with words of documents, but these individual words do not provide reliable evidence for the conceptual content the user is seeking because there are many ways to express a concept and words usually have several meanings. Similar to the I.I.D. assumption in machine learning, in traditional vector-space models used in IR, documents are assumed to be independent since only documents that include query terms are retrieved and higher-order relations between words are not considered. However, LSI takes advantage of implicit higher-order (or latent) structure in the association of terms and documents. For example, [9] proved mathematically and demonstrated empirically that LSI is based on the use of higher order relations, in particular higher order co-occurrences. The authors also demonstrated that the retrieval performance of LSI is correlated with higher order relations. Higher-order relations in LSI capture "latent semantics" [9]. Our second motivation comes from the Statistical Relational Learning domain. The work in this domain focuses on the limitations of the I.I.D. assumption in traditional machine learning and as noted reveals that such a context-free approach does not exploit the available information about relationships between instances in the dataset [4].

Motivated by this prior work, we developed a novel classification algorithm termed HONB that leverages higher-order paths, which implicitly link instances in a dataset. Unlike approaches that assume instances are I.I.D., HONB leverages implicit co-occurrence relationships between attributes in different instances. Attributes (e.g., words in documents in text collections) are richly connected by such higher-order paths, and the generative model built by HONB exploits this rich connectivity pattern.

The power of HONB is most visible when we have only a small amount of training data. This is important because, for example, labeling documents by category or class is an expensive process and in many real world applications the amount of labeled data is far from adequate [11]. In order to demonstrate the utility of this approach we vary the amount of the input (i.e., labeled training data) in our experiments. Our results on several textual datasets show that when training data is scarce (i.e., a small number of labeled instances), HONB significantly reduces the generalization error by leveraging higher-order paths.

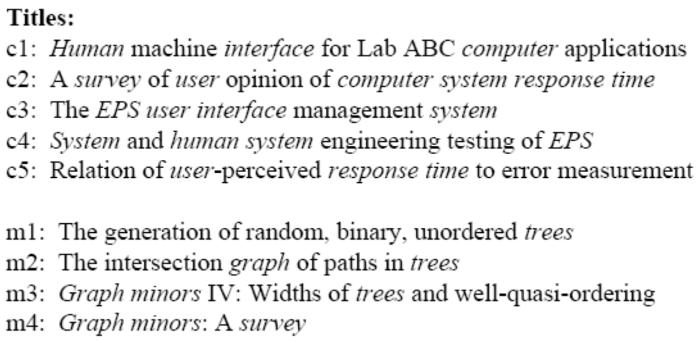
In what follows we first discuss background and related work, then present our approach to learning based on higher-order paths. We follow this with a section summarizing our results for widely-used text classification datasets. Next we discuss these results including aspects of future work, and finally draw conclusions.

II. Background and Related Work

A. *Higher-order Co-occurrences*

Higher-order co-occurrences play a key role in the effectiveness of systems used for information retrieval and text mining. One example is Literature Based Discovery (LBD), which employs second-order co-occurrence to discover connections between concepts (entities). A well-known example is the discovery of a novel migraine-magnesium connection in the medical domain. In [24] Swanson found that in the Medline database some terms co-occur frequently with “migraine” in article titles, e.g. “stress” and “calcium channel blockers.” They also discovered that “stress” co-occurs frequently with “magnesium” in other titles. As a result, they hypothesized a link between “migraine” and “magnesium,” and some clinical evidence has been obtained that supports this hypothesis. In LBD a second-order link of this nature is represented as A→B→C where in this example A is “migraine,” C is “magnesium” and B is one of several possible connecting terms such as “stress.”

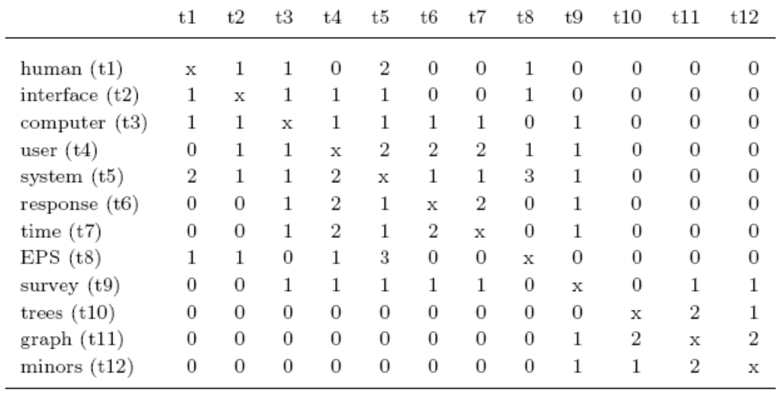
Another example is Latent Semantic Indexing (LSI), a well-known approach to information retrieval. Kontostathis and Pottenger [9] mathematically prove that LSI implicitly depends on higher-order co-occurrences. They also demonstrate empirically that higher-order co-occurrences play a key role in the effectiveness of systems based on LSI. LSI can reveal hidden or latent relationships among terms, as terms semantically similar lie closer to each other in the LSI vector space. This can be demonstrated using the LSI term-term co-occurrence matrix as the following example shows.



Titles:
c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Fig. 2. Example document collection (Deerwester et al., 1990)



	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
human (t1)	x	1	1	0	2	0	0	1	0	0	0	0
interface (t2)	1	x	1	1	1	0	0	1	0	0	0	0
computer (t3)	1	1	x	1	1	1	1	0	1	0	0	0
user (t4)	0	1	1	x	2	2	2	1	1	0	0	0
system (t5)	2	1	1	2	x	1	1	3	1	0	0	0
response (t6)	0	0	1	2	1	x	2	0	1	0	0	0
time (t7)	0	0	1	2	1	2	x	0	1	0	0	0
EPS (t8)	1	1	0	1	3	0	0	x	0	0	0	0
survey (t9)	0	0	1	1	1	1	1	0	x	0	1	1
trees (t10)	0	0	0	0	0	0	0	0	0	x	2	1
graph (t11)	0	0	0	0	0	0	0	0	1	2	x	2
minors (t12)	0	0	0	0	0	0	0	0	1	1	2	x

Fig. 3. Deerwester Term-to-Term Matrix (Adapted from Kontostathis & Pottenger [9])

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
human (t1)	x	0.54	0.56	0.94	1.69	0.58	0.58	0.84	0.32	-0.32	-0.34	-0.25
interface (t2)	0.54	x	0.52	0.87	1.50	0.55	0.55	0.73	0.35	-0.20	-0.19	-0.14
computer (t3)	0.56	0.52	x	1.09	1.67	0.75	0.75	0.77	0.63	0.15	0.27	0.20
user (t4)	0.94	0.87	1.09	x	2.79	1.25	1.25	1.28	1.04	0.23	0.42	0.31
system (t5)	1.69	1.50	1.67	2.79	x	1.81	1.81	2.30	1.20	-0.47	-0.39	-0.28
response (t6)	0.58	0.55	0.75	1.25	1.81	x	0.89	0.80	0.82	0.38	0.56	0.41
time (t7)	0.58	0.55	0.75	1.25	1.81	0.89	x	0.80	0.82	0.38	0.56	0.41
EPS (t8)	0.84	0.73	0.77	1.28	2.30	0.80	0.80	x	0.46	-0.41	-0.43	-0.31
survey (t9)	0.32	0.35	0.63	1.04	1.20	0.82	0.82	0.46	x	0.88	1.17	0.85
trees (t10)	-0.32	-0.20	0.15	0.23	-0.47	0.38	0.38	-0.41	0.88	x	1.96	1.43
graph (t11)	-0.34	-0.19	0.27	0.42	-0.39	0.56	0.56	-0.43	1.17	1.96	x	1.81
minors (t12)	-0.25	-0.14	0.20	0.31	-0.28	0.41	0.41	-0.31	0.85	1.43	1.81	x

Fig. 4. Deerwester Term-to-Term matrix truncated to two dimensions (Adapted from Kontostathis & Pottenger [9])

Let's consider a simple document collection given in Fig. 2 where document c1 has the words {human, interface} and c3 has {interface, user}. As can be seen from the co-occurrence matrix in Fig. 3, the terms "human" and "user" do not co-occur in this example collection. After applying LSI, however, the reduced representation co-occurrence matrix in Fig. 4 has a non-zero entry for "human" and "user" thus implying a similarity between the two terms. This is an example of second-order co-occurrence; in other words, there is a second-order path between "human" in c1 and "user" in c3 through "interface" (common to both c1 and c3). This second-order path implicitly links c1 to c3, violating the I.I.D. assumption. The results of experiments reported in [9] show that there is a strong correlation between second-order term co-occurrence, the values produced by the Singular Value Decomposition (SVD) algorithm used in LSI, and the performance of LSI measured in terms of F-measure , the harmonic mean of precision and recall. As noted, the authors also provide a mathematical analysis which proves that LSI does in fact depend on higher-order term co-occurrence.

There are several research efforts that use LSI in text classification [25], [26], [27], [28], [29]. Although LSI is essentially an unsupervised method, several authors have developed modifications to LSI such that it makes use of class labels during training [27], [28], [29]. Of these approaches [25] is particularly relevant to our work since the focus is also on small training sets. For example for the 20 newsgroups dataset the training set sizes range from one instance per class to 20 instances per class in this work. In addition, the algorithm employs a large number of unlabeled instances to aid in learning. The kNN classifier presented in [25] uses cosine similarity and combines the similarity scores of the 30 closest-neighbors using a noisy-OR operator.

A related issue in LSI is the choice of the truncation parameter k, which can be very important for supervised learning systems based on LSI. Previous work shows that values of k ranging from 100 to 300 give the best results [25]. However, this value is dependent on the size of the corpora. On the other hand, there are methods for choosing a ‘good’ number of singular values for the dimensionality reduction in LSI, some of which are discussed in [30]. Such methods can be used to build an automated LSI-based classifier which adapts the k value as the training set size changes.

B. Statistical Relational Learning

The vast majority of statistical machine learning algorithms operate on ‘flat’ data and traditionally assume that instances are independent and identically distributed (I.I.D.). As noted, however, this context-free approach does not exploit the available information about relationships between instances in the dataset [4]. In statistical relational learning, models operate on relational data that includes explicit links between instances (e.g., hyperlinks between web pages or citation links between scientific papers). These relations provide rich information that can be leveraged to improve classification accuracy because attributes of linked instances are often correlated, and links are more likely to exist between instances that have some commonality [2]. Given a set of test instances, relational models simultaneously label all instances in order to exploit the correlations between class labels of related instances. This is also called collective classification (or collective inference), and violates the traditional I.I.D. assumption. Several studies (e.g., [5], [6], [7], and [8]) have shown, however, that by making inferences about multiple data instances simultaneously, classification error can be significantly reduced [12].

In related work, Macskassy and Provost [3] formulate linked data as networked data and classify machine learning methods for networked data into two categories; within-network inference and across-network inference. Networked data is relational data where instances are interconnected such as web-pages or research papers. In our case, however, the dataset does not have to be innately relational. We extract relational information (i.e., links) from a standard machine learning dataset by using higher-order co-occurrences. These links are called higher-order paths. Each such higher-order path implies a relation between different records (instances).

C. Naïve Bayes

Naïve Bayes is a very popular algorithm due to its simplicity and efficiency, especially in the text mining domain. The primary reason for its simplicity and efficiency is the attribute independence assumption. Although Naïve Bayes assumes that attributes are independent, it nonetheless performs well in numerous domains, including on real world datasets. A detailed analysis of Naïve Bayes can be found in [13]. There has been a focused effort, however, to relax Naïve Bayes' attribute independence assumption while retaining its desirable simplicity and efficiency. Two techniques that exemplify this effort are Lazy Bayesian Rules (LBR) and Super-Parent TAN (SP-TAN). Both of these techniques have achieved remarkable accuracies, but at the cost of high computational overhead. SP-TAN, for example, has high computational complexity at training time and LBR has high computational complexity at classification time. The complexity of such approaches detracts from their usefulness as alternatives to Naïve Bayes [14]. A detailed analysis of these approaches can be found in [14].

III. Enumeration of Higher-order Paths

We focus on discovering higher-order association patterns in a labeled machine learning dataset based on relations between items, or entities. Entities can be attribute-value pairs in a standard machine learning dataset, words in a textual dataset, etc. A higher-order association is represented as a chain of co-occurrences of such entities in different instances. As noted we also refer to such associations as higher-order paths. Given a supervised learning dataset (i.e., labeled training data), we attempt to discover patterns in sets of higher-order associations that distinguish between the classes in the labeled data. In order to accomplish this we first need to enumerate all higher-order paths in a given class of instances. In this section we present our definitions for higher-order paths and data structures to represent and enumerate them. In the following section we present a theoretical framework for the closed-form (analytical) enumeration of higher-order paths.

Our definition of a higher-order path is similar to that found in graph theory, which states that given a non-empty graph G = (V, E) of the form V = $\{x_0, x_1, \dots, x_k\}$, E = $\{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$ 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. Such a path is often referred to by the natural sequence of its vertices $x_0x_1\dots x_k$ [15]. Our definition of a higher-order path differs from this in a couple of respects. First, vertices V = $\{e_0, e_1, \dots, e_k\}$ represent entities, and edges E = $\{r_0, r_1, \dots, r_m\}$ represent records, documents or instances. Finally and most importantly, in a higher-order path both vertices and edges must be distinct. We are interested in enumerating all such paths.

It is not straightforward, however, to represent higher-order paths in conventional graph structures. In order to use conventional graph structures and algorithms, we divided the above representation into two structures. First, we form a co-occurrence graph $G_c = (V, E)$ in which the vertices are the entities and there is an edge between two entities if they co-occur in one or more records. A path (length ≥ 2) extracted from G_c satisfies the first requirement of our higher-order path definition since the vertices in this path are distinct. The second requirement entails that records on a path must be distinct, and another data structure that contains lists of records for each edge is needed. We term this structure a *path group*. Note that the second requirement results in more intuitive paths. For example, with Latent Semantic Indexing (LSI) in view the latent semantics of $e_1 \sim R_1 \sim e_2 \sim R_2 \sim e_3$ is more clear than $e_1 \sim R_1 \sim e_2 \sim R_1 \sim e_3$. We are interested in paths of this nature that implicitly link entities through different records because they have the potential to reveal latent information. As noted previously, such implicit links between records violate the I.I.D. assumption typically made in traditional machine learning algorithms.

However, we are not necessarily looking for the shortest paths between two entities. In the second order path between e1 and e3 in Fig. 5 below, there is a shorter (first-order) path between e1 and e3 as well. Although we use particular order paths (e.g., only second-order or only third-order) in our algorithms, given our definition of a higher-order path we enumerate many more higher-order paths than just the shortest paths. These higher-order paths include latent connection information for number of different records.

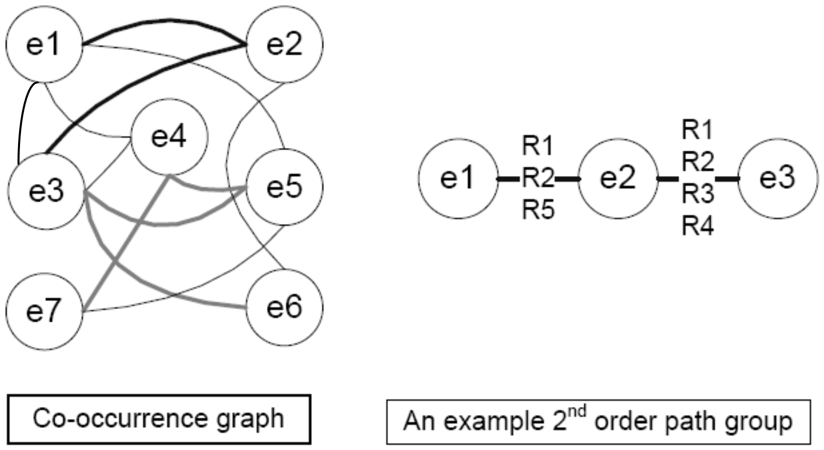


Fig. 5: Path Group structure based on a simple path in co-occurrence graph

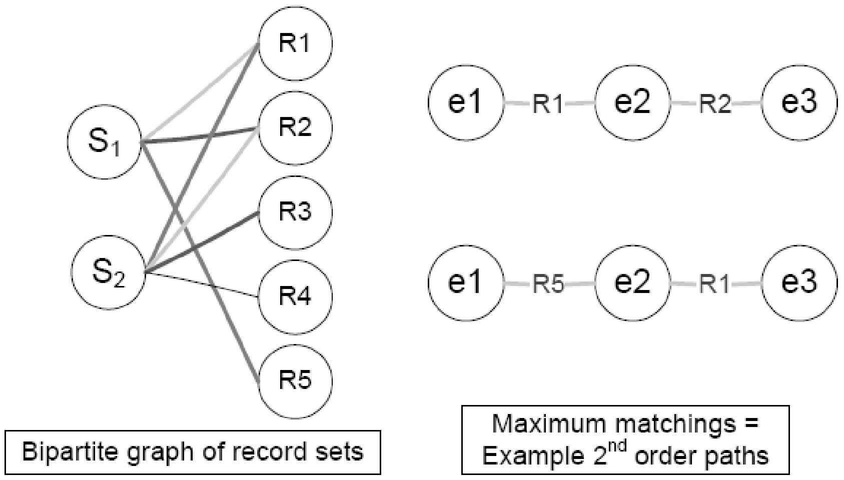


Fig. 6: Extracting/enumerating higher-order paths from path group structure

Using the path group representation it is possible to satisfy the second requirement of our higher-order path definition. In effect, we need to identify the systems of distinct representatives (SDRs) from the sets of records in a path group. An SDR of the sets S_0, …, S_{n-1} is defined as a sequence of n distinct elements r_0, …, r_{n-1} with $r_i \in S_i$, $0 \leq i \leq n-1$ [16]. Each distinct representative corresponds to a record in a higher order path. In order to enumerate all the distinct representatives in a given higher-order path, a bipartite graph $G_b = (V_1 \cup V_2, E)$ is formed such that V_1 represents the sets of records (S_0, S_1, …) in a given path group and V_2

represents the records themselves. A maximum matching with cardinality |V1| in this bipartite graph yields the SDR for a single higher-order path. All possible maximum matchings in the bipartite graph together comprise all the SDRs for all higher-order paths in the path group. This process is summarized in Fig. 5 and Fig. 6. In Fig. 5 we can see an example of the second-order path group (e_1-{1,2,5}-e_2-{1,2,3,4}-e_3) that is extracted from the co-occurrence graph G_c. This particular second-order path group includes two sets of records: S_0={1,2,5} and S_1={1,2,3,4}. S_0 corresponds to the records in which e_1 and e_2 co-occur, and S_1 is the set of records in which e_2 and e_3 co-occur. As noted, the path group may be composed of several higher-order paths. In Fig. 6, a bipartite graph G_b = (V_1 U V_2, E) is formed where V_1 represents the two sets of records and V_2 represents all records in these sets. Enumerating all maximum matchings in this graph yields all higher-order paths in the path group. The second diagram (depicted in Fig. 6) shows two examples of the many paths in this path group. In the first such path, edge labels r_1 and r_3 are records in S_0 and S_1, and the path corresponds to maximum matching in the bipartite graph.

Although this framework gives us the ability to algorithmically enumerate higher-order paths of any length, in fact we have developed closed forms for enumerating the systems of distinct representatives of record sets in second, third and fourth-order path groups. Our approach is motivated by the inclusion-exclusion principle in set theory [16].

Definition 1: An SDR of the sets S_0, …, S_{n-1} is defined as a sequence of n distinct elements r_0, …, r_{n-1} with $r_i \in S_i$, $0 \le i \le n-1$ [16]

As noted we wish to discover all of the systems of distinct representatives of n sets. This requires that we enumerate all possible combinations of *n* distinct records, each coming from one of these *n* sets. Motivated by the inclusion-exclusion principle, we have proven three theorems stating formulas for calculating the number of systems of distinct representatives of two, three and four sets respectively. Note that in what follows we use the notation ‘SDRs’ to refer to systems (plural) of distinct representatives (i.e., one or more higher-order paths).

Theorem 1: The number of SDRs for two sets A_1 and A_2 is:

$\#SDR(A_1, A_2) = \sum_{i|j\in\lambda(1,1)} S_i S_j - \sum_{ij\in\lambda(2)} S_{ij}$ *where* $S_I = |\bigcap_{i\in I} A_i|$

Proof of Theorem 1:

Each term can be represented by a partition of two which is shown as : λ(1,1) and λ(2). The first term represents and counts all sequences of representatives from A_1 and A_2 by multiplying the number of elements in each. In the simplest case consider that we have two disjoint sets A_1 and A_2, and $A_1 \cap A_2 = \emptyset$. Since there are no intersections, an element from A_1 can be combined with any element in A_2 to form a system of representatives and all the representatives in all the systems will be distinct. However, if $A_1 \cap A_2 \neq \emptyset$ then the elements in the intersection can represent both A_1 and A_2 because they belong to both sets. In this case it is possible to have a sequence with repeated or non-distinct representatives. This obviously violates the definition of a SDR and needs to be excluded from the total count. This kind of exception will occur for all elements in the intersection. Therefore we need to subtract the number of elements in the intersection, which corresponds to the second term in the closed-form formula.

Theorem 2: The number of SDRs for three sets A_1, A_2, and A_3 is:

$$\#SDR(A_1, A_2, A_3) = \sum_{i|j|k\in\lambda(1,1,1)} S_i S_j S_k - \sum_{ij|k\in\lambda(2,1)} S_{ij} S_k + 2 \sum_{ijk\in\lambda(3)} S_{ijk}$$

Proof of Theorem 2:

For three sets the proof is more involved. A SDR of these sets consists of three distinct representatives. This time we have three sets and we refer to the terms using a partition of three: λ(1,1,1), λ(2,1), and λ(3). We again start by taking the product of the number of elements in the sets. There are a total of $\sum_{i|j|k\in\lambda(1,1,1)} S_i S_j S_k = |A_1|\,|A_2|\,|A_3|$ of these sequences. Then in the second term we subtract the cases where two or more representatives are non-distinct or identical. In this step we are considering the two set intersections and since we have three sets there are *C(3,2)=3* combinations of them. For the sequences where all three elements are identical, we count them once in the first term and subtract them three times in the second term. As a result we need to add twice the number of these sequences back in to the sum to compensate for over counting.

Theorem 3: The number of SDRs for four sets $A_1, A_2, A_3,$ *and* A_4 *is:*

$$\#SDR(A_1, A_2, A_3, A_4) = \sum_{i|j|k|l\in\lambda(1,1,1,1)} S_i S_j S_k S_l - \sum_{ij|k|l\in\lambda(2,1,1)} S_{ij}\, S_k S_l + 2\sum_{ijk|l\in\lambda(3,1)} S_{ijk}\, S_l + \sum_{ij|kl\in\lambda(2,2)} S_{ij}\, S_{kl} - 6\sum_{ijkl\in\lambda(4)} S_{ijkl}$$

Proof of Theorem 3:

We refer to all possible sequence patterns by a partition of size four: λ(1,1,1,1), λ(2,1,1), λ(3,1), λ(2,2), λ(4) with respect to the formula above. Since λ(1,1,1,1) represents all the sequences obviously its coefficient will be one. In the second term we subtract λ(2,1,1); two set intersections which produce sequences with two or more non-distinct representatives. As with the previous theorem, the second term includes the sequences of λ(3,1), which are three set intersections that produce three or more non-distinct representatives. Thus we add twice λ(3,1) back into the sum in the third term. Similarly each summand of λ(2,2) occurs in two different summands of λ(2,1,1), so we add λ(2,2) once to compensate in the fourth term. Finally, λ(4) is a four set intersection which produces sequences with four non-distinct representatives which occur once in every summand of the previous terms. This means the formula is too large by $1 - C(4,2) + 2\,C(4,3) + 3 = 6$, so we need to subtract λ(4) six times in the fifth term.

These closed-form formulas (Theorems 1, 2 and 3) are used in enumerating the second, third and fourth-order paths in the path group structures exemplified in Fig. 6 above. When it is necessary to enumerate paths of order greater than four, we use the algorithmic approach outlined previously to enumerate all maximum matchings in the bipartite graph.

IV. HIGHER ORDER NAÏVE BAYES

Our approach is based on a Naïve Bayes algorithm that is applied to the problem of text classification. Naïve Bayes is commonly used in text classification because it is fast and easy to implement. Naïve Bayes is the simplest of Bayesian classifiers in that it assumes that all attributes of the examples are independent of each other given the context of the class [17]. Although this assumption does not hold for most real-world datasets, overall Naïve Bayes performs fairly well. In our case the relative simplicity of the Naïve Bayes classifier allows for detailed analysis of the effect of using higher-order paths.

It is important to note that the focus in this work is on the independence assumption between instances. Per the definition in the previous section, higher-order paths link attributes from different instances thereby implicitly linking different instances. In this way we violate the I.I.D. assumption on instances. For this reason our approach is similar to the work in link mining that leverages explicit links between instances during training. We *do not,* however, modify the Naïve Bayes algorithm to relax its assumption of independence between attributes. (As noted in the Background and Related Work section, this latter approach is taken in work such as [14].) In contrast, in our framework although priors for Naïve Bayes are estimated from higher-order paths we do not alter the Naïve Bayes algorithm itself. That is why we name our approach

Higher Order Naïve Bayes (HONB).

Naïve Bayes assumes that a document is generated by a parametric model. We use training data to calculate estimates of model parameters. By using these estimates with Bayes rule we can calculate the probability that a class generated a given test document. We classify the document in the most probable class in the usual way.

There are two different generative models commonly used for Naïve Bayes classification, the multivariate Bernoulli model and the multinomial model. A multinomial model is a unigram language model with integer term counts [17]. In this research we implemented a multivariate Bernoulli model (also known as a binary independence Naïve Bayes model). In this model an event is a document and is represented by a vector of binary attributes $X_1,\ldots, X_d$, which take values {0,1} indicating occurrence of terms in the document. The number of times a term occurs in a document is not captured. X_0 represents the class label of the document and takes values in $\{1,\ldots,C\}$ where C is the number of classes.

One can calculate the probability of an event (i.e., a document) given the probability of a document *d* belonging to class *C* using Bayes rule as follows:

$$(1) \quad P(c|d) = \frac{P(d|c)\,P(c)}{P(d)}.$$

In a multiclass classification scenario, one assigns document *d* to the class with the highest *P(c|d).* In addition, since *P(d)* is independent of the class it is not needed during classification and we have:

$$(2) \quad P(c|d) \alpha\, P(d|c)P(c).$$

A document consists of words $d \equiv \{w_1, w_2, \ldots, w_n\}\}$. Here we can understand the document to be the “event,” and the absence or presence of words to be attributes of the event [17]. The Naïve Bayes algorithm assumes that each word in a document is independent of others and independent of its position in the document. As a result, when calculating the probability of a document, the product of the probabilities of all the attribute values, including the probability of non-occurrence for terms that do not occur in the document, is taken:

$$(3)\quad P(d|c) = \prod_{w\in d} P(w\,|c) \prod\nolimits_{w\notin d, w\in W} (1 - P(w\,|c))$$

However, during classification performance is improved by not calculating the second product for each test document [18], so the equation is rewritten as:

$$(4)\quad P(d|c) = \prod_{w\in d} \frac{P(w|c)}{1 - P(w|c)} \prod\nolimits_{w\in W} (1 - P(w\,|c))$$

Here the second product is pre-computed and stored for each class.

To estimate a class probability of a document *p(c|d)*, the priors must be estimated from labeled training documents. Probabilities of all words in class *c* are estimated by:

$$(5)\quad \hat{P}(w|c) = \frac{n(c,w)}{n(c,d)}$$

where *n(c,w)* is the number of documents in class *c* including word *w* as in Equation [6] and *n(c,d)* is the total number of documents in class *c*.

$$(6)\quad n(c,w) = \sum_{d\in D_c} n(d,w)$$

Another parameter of the system is the class prior probability, which is estimated using Maximum Likelihood Estimation (MLE) as follows:

$$(7)\quad \hat{P}(c) = \frac{n(c,d)}{\sum_{c\in C} n(c,d)}$$

In Higher Order Naïve Bayes we form a co-occurrence graph for each class of labeled training documents and enumerate specific higher-order (e.g., second-order) paths. Our system parameters are the same as in the traditional multivariate Naïve Bayes model; however we estimate the parameters using the higher-order paths instead of documents. Similarly, probabilities of all words in class *c* are estimated by:

$$(8)\quad \hat{P}_{\sim}(w|c) = \frac{\tilde{n}(c,w)}{\tilde{n}(c,h)}$$

where *ň(c,w)* is the number of higher-order paths in class *c* including word *w* as in Equation 9, and *ň(c,h)* is the total number of higher-order paths in class *c*.

$$(9)\quad \tilde{n}(c,w) = \sum_{h \in H_c} \tilde{n}(h,w)$$

The class prior is estimated by dividing the number of higher-order paths in class *c* by the sum of the higher-order paths in all classes as shown in Equation 10:

$$(10)\quad \hat{P}_{\sim}(c) = \frac{\tilde{n}(c,h)}{\sum_{c \in C} \tilde{n}(c,h)}$$

Finally, by using the parameter estimates described above, we can estimate the class probability of a test document using:

$$(11)\quad \hat{P}(c|d)\ \alpha\ \hat{P}_{\sim}(c) \prod_{w \in d} \frac{\hat{P}_{\sim}(w|c)}{1-\hat{P}_{\sim}(w|c)} \prod\nolimits_{w \in W} (1-\hat{P}_{\sim}(w\,|c)).$$

By using the same parameters – e.g., words – Higher Order Naïve Bayes has the advantage that (just like Naïve Bayes) a single test instance can be classified without the need for additional context or a mapping function. However, by using higher-order paths instead of documents we also exploit the latent connections between documents within a given class. For both Naïve Bayes and Higher Order Naïve Bayes, we employ Laplace smoothing to avoid zero probabilities.

As noted above, although we do not assume that documents are I.I.D. during model construction, we still make the 'naïve' assumption that attributes are independent from each other during classification because we are using Naïve Bayes.

For all the experiments reported in this article we used second-order paths. This choice was based on the empirical observation that although models built from third-order paths show the same pattern of performance, the performance for second-order paths is slightly better. As noted previously a second-order path connects three different words (or entities) in two different documents (or records). For example, w_1~D_x~w_2~D_y~w_3 .

A. Complexity Analysis

During training, Naïve Bayes (NB) assembles a simple table of class probability estimates and a table of conditional attribute value probability estimates for each class. Therefore the space complexity is *O(cnv)*, where *c* is the number of classes, *n* is the number of attributes, and *v* is the average number of values per attribute [14]. In the case of text classification, the number of attribute values corresponds to the word dictionary size in both the multinomial and multivariate Naïve Bayesian models. As a result the space complexity can be defined as *O(cd)* where *d* is the dictionary size. Attribute value conditional probability estimates can be calculated by a simple scan through the data; thus the time complexity of the learning phase is *O(td)*, where *t* is the number of training examples and *d* is the dictionary size. During classification, classifying a single instance has time complexity *O(cd)* using the tables formed during training.

During training with Higher Order Naïve Bayes we first need to enumerate second-order paths before calculating probability estimates. As mentioned before, for each class in *k* classes we first form a co-occurrence graph and then enumerate all length two paths from this graph. All length two paths in a graph can be enumerated in time $O(n^3)$ (based on Stirling's approximation of $O(n^2)$ for the number of paths of length two [19][1]) where *n* is the number of nodes (words) in the co-occurrence graph of the words in the documents in a given class, and is bounded by *d*.

[1]An intuition for the $O(n^{(P+1)})$ bound for paths of length P can be seen by comparison to the generation of a tree for path traversal. The number of children formed is the total number of paths generated. Such a tree has a fan out of (*n*-1) at the first step (in the worst case of a clique, meaning a node can generate a path to every other node excluding itself), *n*-2 at the second step (excluding itself and the first vertex) and so on down to one for paths of all lengths up to P. Therefore the total number of children is *n*-1 * *n*-2 * ... * *n-P*, yielding $O(n^P)$ complexity for each of *n* vertices.

After this step we form path groups based on each of these paths. We enumerate higher-order paths from the path groups using a closed-form formula which in the worst case has time complexity $O(u \log(r))$ where u and r are the set sizes, both of which are bounded by t, the number of training examples (see Fig. 5 and 6). In our work with small samples of training data in text classification, $t << d$. We saw previously that there are $O(n^2)$ path groups, yielding a worst-case time complexity of $O(t \log(t) n^2)$ for path enumeration. In addition we saw that the time complexity of training for Naïve Bayes is $O(td)$, where d is equivalent to n in the worst case (when the documents in a class have the entire dictionary in them). As a result, during training HONB has an overall worst-case time complexity of $O(cn^3)$ where c is the number of classes and as noted n is bounded by d. This compares favorably with the time complexity for training in SP-TAN as well as the time complexity for classification in LBR, both of which require $O(tkn^3)$ where t is the number of training examples, c as before is the number of classes and n is the number of attributes (equivalent to the number of words in the dictionary of size d in a text classification application) [14].

In terms of space complexity, the only difference between HONB and NB is that in HONB we need to store the co-occurrence graph (e.g., as an adjacency list). In the worst case for a complete graph, we have $O(d^2/2)$ additional space complexity since this is a undirected graph. We don't need to store higher-order paths themselves since the attribute value statistics can be updated while enumerating paths. Finally, the time complexity of HONB is exactly the same as NB in the classification phase, which is $O(cd)$.

V. Experiments

A. Comparing HONB with Traditional Classifiers

In this section we present the results and analysis of our experiments comparing Higher Order

Naïve Bayes (HONB) with three traditional classifiers, Naïve Bayes (NB), Support Vector Machines (SVM) and an LSI-based kNN classifier (LSI kNN). NB is our baseline classifier. As stated by Chakrabarti [18] "SVMs are some of the most accurate classifiers for text; no other kind of classifier has been known to outperform it across the board over a large number of document collections." For example, SVMs usually perform better than Naïve Bayes. Therefore, we included SVM in our experiments as the state-of-the-art classifier in general in text classification domain. In addition to SVM we also included an LSI-based k-Nearest Neighborhood (kNN) classifier since in our prior work we proved that LSI implicitly uses higher-order co-occurrence paths [9]. Both SVM and LSI-based kNN classifiers were optimized to achieve the best performance on our datasets. Experiments were done using second-order paths for HONB. We have conducted additional experiments using third-order paths, but the results did not differ significantly. Therefore, given the lower execution time of second-order path enumeration, we chose to use second-order paths. For the experiments in this section we used four subsets of the 20 Newsgroups dataset that are also used in related work using word clusters for text classification (e.g., [11]). This related work is also focused on improving the efficiency of the classifiers when the training data is scarce, and these datasets provide an appropriate platform to demonstrate the usefulness of our algorithm. The description of these datasets is given in Table 1. The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned evenly across 20 different newsgroups. It is a commonly used benchmark dataset for text classification. We used the 18828 version of the dataset which has cross-postings (duplicates) removed and all headers filtered except for "From" and "Subject" headers[2]. The dataset was preprocessed using the Text Mining Infrastructure (TMI) [20]. We used the stemmer and stop word list embedded in the TMI, and filtered words that occurred in

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

less than three documents. Furthermore, for comparison purposes we employed Information Gain (IG) to rank the words in the dictionary and select the top-ranked 2000 words for each subset. To observe the scalability of HONB we also conducted experiments on several additional datasets: Cora [21], Citeseer [23] and WebKB [22]. For Cora, Citeseer and WebKB, we downloaded[3] and used the same processed versions of these datasets that were used in the experiments reported in [23]. The Cora dataset is composed of a number of scholarly research articles on machine learning, in all comprising seven classes: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory. In [23] Sen and Getoor chose documents such that each document is either cited or cites one of the other documents in the corpus. Stop words were removed, and words with document frequency less than 10 were also removed. The final corpus contains 2708 documents with a vocabulary of 1433 distinct words and 5429 links. Similarly, the Citeseer dataset is composed of a number of scholarly research articles from computer science domain and contains 3312 documents with a vocabulary of 3703 terms. The WebKB dataset is a hypertext dataset consisting of the web pages of the computer science departments of four different universities. There are five classes: Course, Student, Faculty, Project and Staff. Sen and Getoor [23] selected only those documents which either link to or are linked by at least one other document in the dataset and extracted a corpus of 877 documents. After stemming and stop word removal the dictionary size is 1703 distinct words, and there are 1608 links in this corpus.

Table 1: Datasets used in Experiments

Dataset	**Class Names**
COMP (5)	comp.graphics, comp.window.x, comp.sys.mac.harware, comp.os.ms-windows.misc, comp.sys.ibm.pc.harware
SCIENCE (4)	sci.crypt, sci.electronics, sci.med, sci.space
POLITICS (3)	talk.politics.mideast, talk.politics.guns, talk.politics.misc
RELIGION (3)	alt.atheism, talk.religion.misc, soc.religion.christian

[3] http://www.cs.umd.edu/~sen/lbc-proj/LBC.html

Citeseer (6)	AI, Agents, DB, HCI, IR, ML
Cora (6)	Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Theory
WebKB (5)	Student, Faculty, Course, Project, Staff

As mentioned we compare our results to those from three different traditional classifiers, Naïve Bayes, LSI kNN and SVM. All our datasets have binary word features which indicate only the occurrence or non-occurrence of a word in a document. We employed a Multivariate Binary event model in a Naïve Bayesian framework as described above. The SVM classifier we used is implemented in the R statistical toolkit [31] in package e1071, which is the R interface to the libsvm library [32]. For the comparison to SVM, we optimized parameters to obtain the best possible result including a range of experiments with different kernels (e.g., polynomial, radial basis function). In addition, when selecting the value of the soft margin cost parameter C for SVM, we considered the set $\{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$ of possible values. On every trial, we picked the smallest value of C which resulted in the highest accuracy obtained on the training set. During optimization of the parameters for the SVM runs we observed that the C-values were mostly 10^{-4} and 10^{-3}, and the RBF kernel performed the best about 60% of the time.

For the LSI kNN classifier, based on the related work in [25] we used the 25 nearest neighbors combined using the noisy-OR operator and cosine similarity. The LSI kNN classifier we employed is implemented in the LSA package, also in the R statistical toolkit. The truncation parameter k of LSI can be set using various recommender routines in this package. Similar to SVM we optimized the LSI kNN classifier by choosing the best performing recommender routine in this package, which was based on Kaiser-Criterium that retains the singular values larger than 1.0.

In order to simulate a real world scenario where only a few labeled data instances are available, in this set of experiments we selected 5% of the data for training from each dataset. In the 20 Newsgroups subsets, 5% corresponds to 25 documents per class. The remaining 475 documents per class were used as our test set. The other datasets, however, do not have an equal number of documents in each class so the number of instances selected varied depending on the distribution of instances. In all experiments we conducted eight random trials and averaged the results over these trials. The detailed results are depicted in Table 2. The performance is presented in terms of accuracy, and the number of classes per dataset is given in parentheses in the first column. Additionally, the largest absolute performance is highlighted in bold. Percent improvements are statistically significant at the five percent level unless marked in red italics, in which case the performance difference is not statistically significant.

For all datasets, HONB significantly outperformed the baseline NB classifier. The performance of HONB is most visible on the 20 Newsgroups datasets. On these datasets HONB outperforms all the other classifiers including state-of-the-art SVM and LSI kNN. The improvement of HONB over SVM is statistically significant for these 20 Newsgroups datasets. Similarly, HONB performs statistically significantly better than LSI kNN on three of the four datasets. Nonetheless, for the Citeseer, Cora and WebKB datasets, SVM is the best performing classifier. However, the difference in performance between SVM and HONB on the WebKB dataset is not statistically significant. Additionally, HONB statistically significantly outperforms LSI kNN on both the Cora and WebKB datasets.

These results suggest that under the conditions where training data is scarce, HONB is a significantly better classifier than NB. In addition, for five out of seven of these datasets HONB is comparable or better than SVM, which is as noted one of the best performing algorithms for text classification. Finally, for six of the seven datasets HONB is comparable or better than LSI kNN.

We contend that the main reason behind the superior performance of HONB is that it makes explicit use of the most valuable latent information that exists in higher-order paths. Just as prior research revealed for LSI [9], higher-order paths provide additional latent information. Unlike LSI, however, HONB explicitly leverages this higher-order information. This in fact was one of the motivating purposes for our research – from our prior work we knew LSI leveraged higher-order information, but wanted to get an answer to the question "How is this information best leveraged?" These results represent a significant step forward in answering this question.

In summary, in the case of text classification documents and words in a document set are richly connected by such higher-order paths. Higher-order paths as noted go beyond instance boundaries. In contrast, traditional classifiers only leverage relationships between attributes within instances (e.g., words within documents), or, as with the case of LSI kNN, are a black-box in terms of how they leverage latent information. When training data is scarce, these approaches do not exploit sufficient information from individual instances and thus do not obtain reliable parameter estimates. In contrast, HONB explicitly exploits the rich connectivity between words in different documents that belong to a given class. Consequently, HONB is able to obtain better parameter estimates and performs better.

Table 2: Comparison of HONB with Traditional Classifiers on 5% Training Data Samples

	Accuracy %				% improvement of HONB		
Dataset	NB	SVM	LSI kNN	HONB	over NB	over SVM	over LSI
COMP (5)	51.43	58.53	61.09	**65.06**	26.51	11.16	6.50
SCIENCE (4)	59.56	75.09	74.34	**84.32**	41.58	12.29	13.43
POLITICS (3)	67.86	77.36	82.15	**83.34**	22.82	7.73	***1.45***
RELIGION (3)	64.13	69.88	69.71	**74.18**	15.66	6.15	6.40
Citeseer (6)	31.30	**60.94**	60.61	51.52	64.62	-15.45	-15.0
Cora (6)	30.19	**55.31**	44.71	50.33	66.68	-9.00	12.6
WebKB (5)	54.82	**67.00**	48.58	64.06	16.86	***-4.39***	31.9

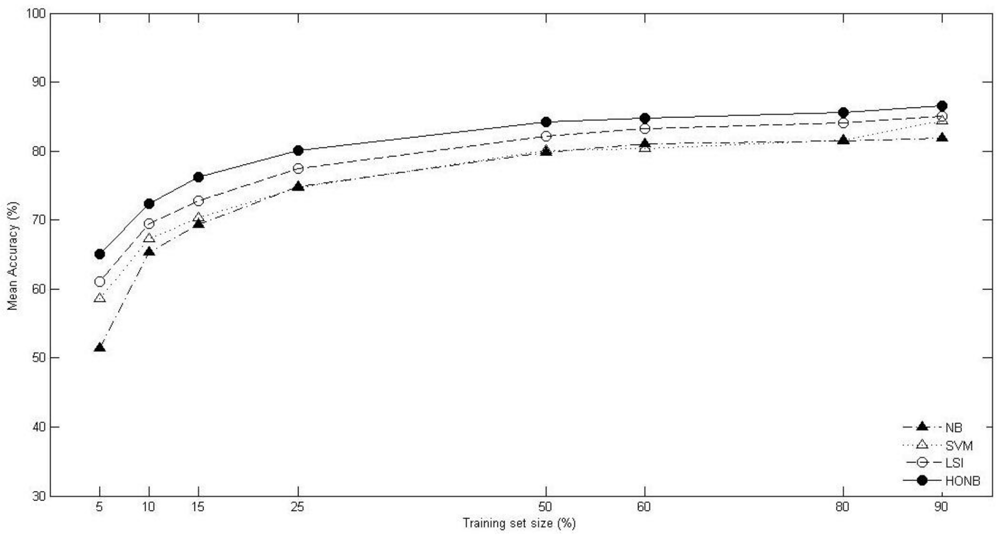
B. Evaluating HONB by Varying the Sparsity of Input

In the previous section we provided an analysis of HONB performance compared to the traditional classifiers NB, SVM, and LSI kNN. We observed that for a five percent subset of the training data, HONB outperforms the baseline Naïve Bayes classifier for all the datasets and outperforms the other traditional classifiers in most cases. In this section we drill down more deeply to explore the range of performance of HONB as a function of the scarcity of input training data. As before, we compare HONB's performance to traditional classifiers including NB, SVM, and LSI kNN. The summary of datasets is repeated for convenience in Table 3.

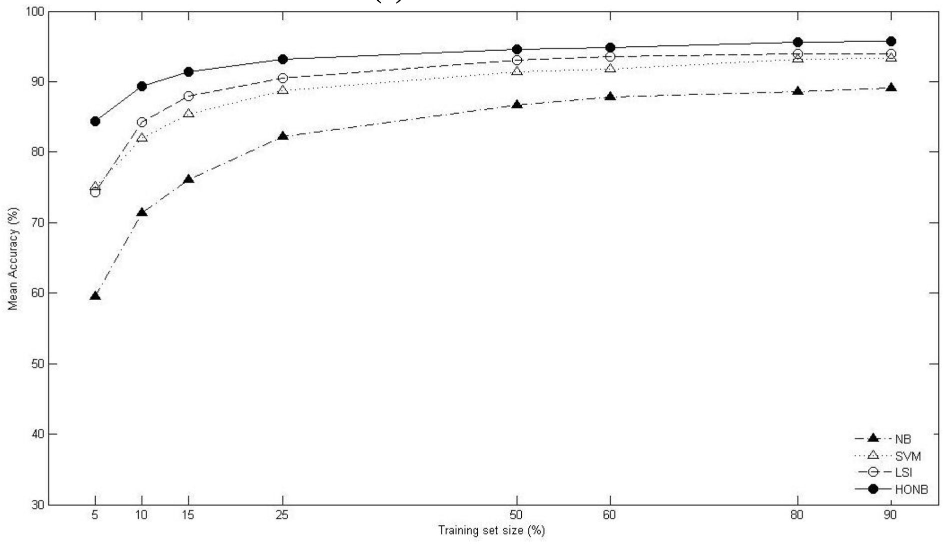
Table 3: Summary of Datasets Used in Experiments Varying Scarcity of Training Data

Dataset	Documents	Dictionary	Classes
COMPUTER	500	2000	5
SCIENCE	500	2000	4
POLITICS	500	2000	3
RELIGION	500	2000	3
Citeseer	3312	3703	6
Cora	2708	1433	7
WebKB	877	1703	5

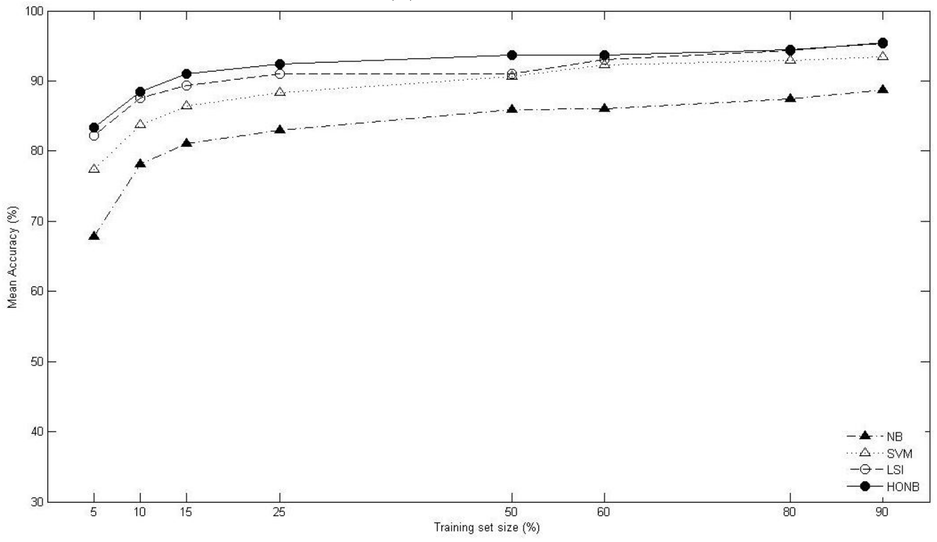
Although there are explicit links in some of these datasets, as noted previously HONB does not use them. We are only interested in modeling the textual data so that we can assess the impact of leveraging implicit higher-order information. For the following experiments we use the standard evaluation metric of accuracy. As mentioned, in the analysis in this section we vary the scarcity of the input (i.e., labeled training data) in the experiments in order to demonstrate the utility of our approach. Scarcity is measured in terms of the size of the input training set, where the number of documents per class ranges from 5% to 90%.



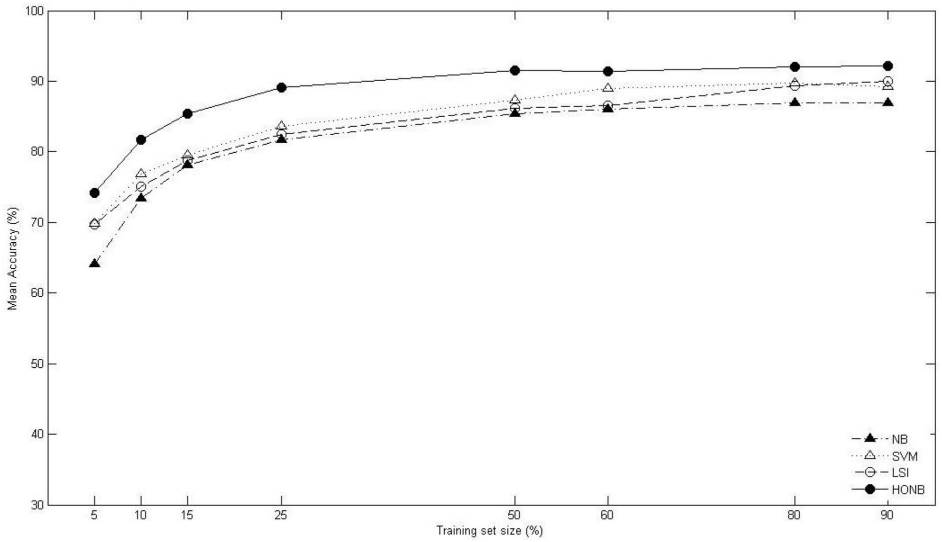
(a) COMPUTER



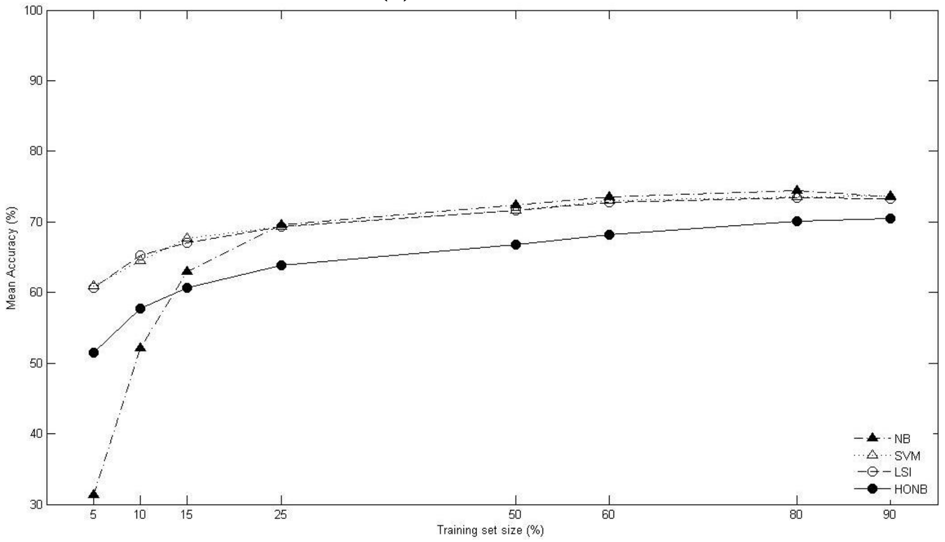
(b) SCIENCE



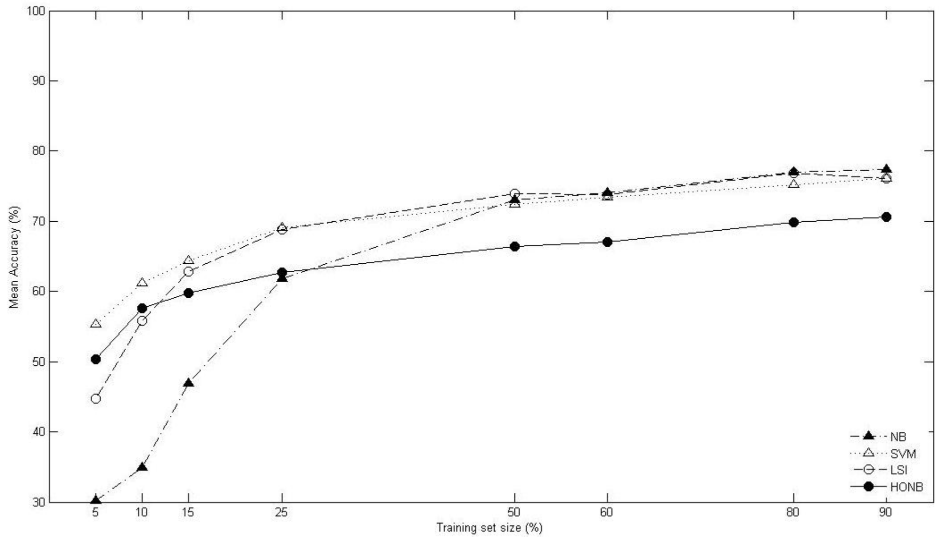
(c) POLITICS



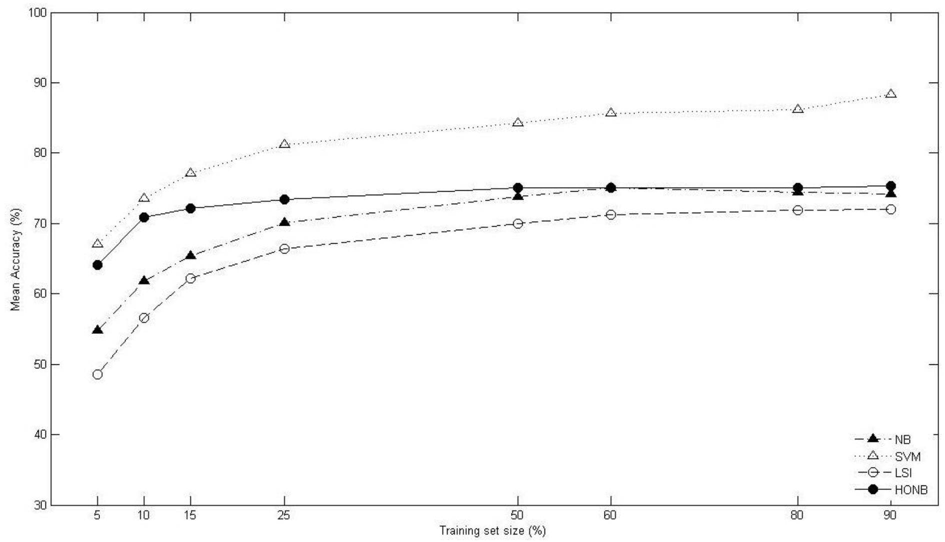
(d) RELIGION



(e) Citeseer



(f) Cora



(g) WebKB

Fig. 7: Classifier Accuracy from 5% to 90% of training set.

Fig. 7(a) through Fig. 7 (d) confirm that HONB consistently outperforms the baseline method, traditional NB, in terms of classification on the COMPUTER, SCIENCE, POLITICS, and RELIGION datasets. All these accuracy improvements of HONB over NB are statistically significant at the five percent level for all the percentages shown from 5% to 90%. HONB performs exceptionally well on 20 Newsgroups subsets, and in addition to outperforming the baseline classifier it also outperforms the state-of-the-art SVM classifier, as well as LSI kNN. Improvement over SVM is statistically significant at the five percent level for all the percentages above from 5% to 90%. Improvement over LSI kNN is also statistically significant at the five percent level except for the 80% and 90% samples on the COMPUTER dataset, the 5%, 10% and 60% to 90% samples for the POLITICS dataset, and the 90% sample for the RELIGION dataset.

A slightly different pattern of performance can be seen on Citeseer (Fig. 7 (e)) and Cora (Fig. 7 (f)) datasets. While classification accuracy of each method monotonically increased with increasing training set size, HONB drastically outperformed NB when training sets were small, reaching close to SVM and LSI kNN. Interestingly LSI kNN performs comparably with SVM for all the training set sizes, and after about 50% NB also achieves similar performance. For our last dataset, WebKB, we see a similar performance pattern as that of the 20 Newsgroups subsets in that HONB outperforms the baseline NB classifier for all training set percentages. However, here again improvement is statistically significant at the five percent level only up to the 50% sample size. In addition, although LSI kNN performs comparably with SVM on all the other datasets, SVM performs exceptionally well on this dataset and outperforms LSI kNN by a large margin. Nonetheless, the performance of HONB is close to SVM in the 5% to 10% sample size range, and the difference between the two is not significant at five percent. WebKB is a much smaller dataset than the others (see Table 3) resulting in many fewer training documents per class for each subset of the training data selected. Additionally, we should note that the class distribution of WebKB is quite different from that of the other datasets. One of the five classes (the student class) has many more documents than the others. Despite these differences HONB still achieves better performance compared to the baseline model. This highlights the robustness of HONB over different datasets with different class distributions.

These results are especially encouraging in that they suggest that higher-order methods were able to construct robust models even when training data was particularly scarce. We speculate that changes in the distribution of highly-discriminative terms across classes as a result of increasing training set size allowed NB to outperform HONB on the Citeseer and Cora datasets once the amount of training data reached a certain point. We are currently investigating this hypothesis and intend to report our findings as part of future work.

We conclude that the advantage of HONB, especially at the 5% to 10% sample size range, for all datasets is due to the lack of sufficient information in these small training sets to estimate the parameters for a traditional model. Simply put, there are too few documents per class. Even with a very small number of documents per class, however, HONB leverages a large number of second-order paths. These paths enable HONB to generalize even under conditions of extremely scarce input.

In order to verify the value of latent higher-order information, in an additional experiment we used what we termed *pure second-order paths.* In this case we enumerated only length two paths between entities (words) that did not co-occur elsewhere in the collection. This means we exploited *only* the higher-order latent information. Table 4 compares the accuracy of HONB vs. HONB with pure second-order paths. As before eight random samples were chosen for each newsgroup category and the average accuracies computed. As can be seen in Table 5, there is no statistically significant difference between the results.

We conclude that the performance improvement of Higher Order Naïve Bayes is due to the fact that the algorithm leverages the latent information present in higher-order paths. By exploiting this latent information we obtain significantly better results than traditional I.I.D. classifiers on several datasets.

Table 4: Accuracy of HONB vs. HONB with pure higher-order paths

Dataset	HONB (Avg.)	HONB pure (Avg.)	T-test p-value
COMP (5)	65.1	64.9	0.73
SCI (4)	84.3	84.2	0.82
POL (3)	83.3	83.6	0.76
REL (3)	74.2	74.5	0.88

Overall, these results indicate that when there is insufficient information for traditional Naïve Bayes to estimate parameters, Higher Order Naïve Bayes exploits the latent information in higher-order paths to achieve significantly better generalization.

VI. Discussion

By taking higher-order paths as our base unit of ‘semantics’ – our ‘documents’, if you will – we reveal the latent semantics that distinguish instances of different classes. These results also support our hypothesis that we can capture class-specific latent information in a generative classification model like Naïve Bayes under certain sparse data conditions. Indeed, for some applications all one can get is a very small number of labeled examples: precisely the conditions under which Higher Order Naïve Bayes performs well.

One of the limitations of Higher Order Naïve Bayes is the constraint to binary data, which indicates only the presence or absence of a word in a document. In consequence, our results may not generalize to multinomial data which includes word frequency information. However, one direction of our future work is to adapt Higher Order Naïve Bayes to use a multinomial model instead of a multivariate model. It may be possible to incorporate word frequency information in a higher-order path. For example, in the second order path $w_1 \sim D_x \sim w_2 \sim D_y \sim w_3$, the word frequencies in D_x and D_y could be used respectively for w_1 and w_3. w_2 poses a problem because it occurs in two documents: D_x and D_y, and we could choose to use either frequency or some combination. In any case, by leveraging word frequency information in higher-order paths it may be possible to exploit more information and obtain improvements on non-binary data sets. This is especially interesting given the observations of the difference in performance of multivariate vs. multinomial models in [17].

VII. Conclusions

In prior work [9], we gave a mathematical proof supported by empirical results of the dependence of LSI on higher-order paths. In this work, a general approach to leveraging higher-order dependencies for supervised learning was developed. We presented a novel classification method termed Higher Order Naïve Bayes.

Higher-order paths allow a classifier to operate on a much richer data representation than the conventional feature-vector form. This is especially important when working with small training sets where accurate parameter estimation becomes very challenging for traditional I.I.D. approaches [11]. Experimental results affirmed the value of leveraging higher-order paths on binary data, resulting in significant improvements in classification accuracies on benchmark text corpora across a wide range of training set sizes.

Based on these and other similar results, we conclude that for binary data sets, higher order paths provide valuable information that can improve model performance over traditional models. This work moves the field a step closer to understanding how higher-order information should be leveraged in a systematic way for greatest impact.

Acknowledgments

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

The authors also wish to thank colleague Shenzhi Li for her many insightful comments during the development of Higher Order Naïve Bayes. We also would like to thank the anonymous reviewers for their input; it improved this article significantly. Finally, co-author W. M. Pottenger gratefully acknowledges the continuing help of his Lord and Savior, Yeshua the Messiah (Jesus the Christ) in his life and work.

References

[1] B. Taskar, P. Abbeel, and D. Koller, , "Discriminative Probabilistic Models for Relational Data", In Proceedings of Uncertainty in Artificial Intelligence conference UAI02, Edmonton, Canada, 2002.

[2] L. Getoor, and C.P. Diehl, . "Link Mining: A Survey", SIGKDD Explorations. 7 2, 2005, pp. 3-12.

[3] S.A. Macskassy, F. and Provost, , "A brief survey of machine learning methods for classification in networked data and application to suspicion scoring", Workshop on Statistical Network Analysis at 23rd International Conference on Machine Learning, Pittsburg, PA, 2006..

[4] R. Angelova, and G. Weikum, "Graph-based Text Classification: Learn From Your Neighbors", SIGIR'06, Seattle, USA, 2006.

[5] S. Chakrabarti, S, Dom, B and Indyk, P., "Enhanced Hypertext Classification Using Hyper-Links", In Proceedings of ACM SIGMOD Conference. pp. 307-318,1998

[6] J. Neville, and D. Jensen ,"Iterative Classification in Relational Data", In Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data. pp. 13-20, 2000.

[7] B. Taskar, E. Segal, and D. Koller, "Probabilistic Classification and Clustering in Relational Data", 2001. In Proc. 17th International Joint Conference on Artificial Intelligence. pp. 870-878.

[8] Q. Lu, and L. Getoor, "Link-based Classification", Washington DC. U.S.A, 2003, Proceedings of the Twentieth international conference on machine learning (ICML-2003).

[9] A. Kontostathis and W.M. Pottenger, "A Framework for Understanding LSI Performance", Information Processing & Management. 42 1, 2006, pp. 56-73.

[10] S. Deerwester, S.T. Dumais, R. Harshman, "Indexing by latent semantic analysis", Journal of the American Society for Information Science, Vol. 41, 6, 1990, pp. 391-407.

[11] N. Slonim, and N. Tishby, "The Power of Word Clusters for Text Classification", 23rd European Colloquium on Information Retrieval Research, 2001.

[12] D. Jensen, J. Neville and B. Gallagher, "Why Collective Inference Improves Relational Classification", SIGKDD'04, 2004

[13] P. Domingos, and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple Bayesian classifier", 1996. In Proceedings of the Thirteenth International Conference on Machine Learning. pp. 105-112.

[14] G.I. Webb , J.R. Boughton, and Z. Wang, "Not So Naive Bayes: Aggregating One-Dependence Estimators", Machine Learning, Vol. 58, 1, 2005

[15] R. Diestel, "Graph Theory", Springer Press, , ISBN 0-387-95014-1, 2000.

[16] J.H. Van Lint, and R.M.A. Wilson, "A Course in Combinatorics", Cambridge University Press, ISBN: 0-521-42260-4, 1993.

[17] A.K. McCallum, and K. Nigam, "A comparison of event models for naive bayes text classification", In Working Notes of the ICMLAAAI Workshop on Learning for Text Categorization, 1998.

[18] S. Chakrabarti, "Mining the Web: Discovering Knowledge from Hypertext Data", Morgan Kaufmann, 2002.

[19] J. Stirling, "Methodus Differentialis" (p. 137), 1730.

[20] L.E. Holzman, T.A. Fisher, L.M. Galitsky, A. Kontostathis, W.M. Pottenger, "A Software Infrastructure for Research in Textual Data Mining", The International Journal on Artificial Intelligence Tools, Vol. 14(4), 2004, pp. 829-849.

[21] A.K. McCallum, K. Nigam, J. Rennie, K. Seymore, "Automating the Construction of Internet Portals with Machine Learning", Information Retrieval, pp. 127-163, 2000.

[22] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, K. Nigam, S. Slattery, "Learning to Extract Symbolic Knowledge from the World Wide Web", 1998. In Proceedings of 15th National Conference on Artificial Intelligence (AAAI-98).

[23] P. Sen and L. Getoor, "Link-based Classification", University of Maryland Technical Report, Number CS-TR-4858 , February 2007, 2007.

[24] D.R. Swanson, "Migraine and magnesium: eleven neglected connections", Perspectives in Biology and Medicine, 31(4), 526-557, 1998.

[25] S. Zelikovitz and H. Hirsh, "Using LSI for text classification in the presence of background text", In Proceedings of the Conference on Information and Knowledge Management, pg. 113–118, 2001.

[26] S. Zelikovitz and H. Hirsh, "Improving short-text classification using unlabeled background knowledge to assess document similarity", In Proceedings of the 17th International Conference on Machine Learning, pages 1183–1190, 2000.

[27] J. Sun, Z. Chen, H. Zeng, Y. Lu, C. Shi, W. Ma, "Supervised Latent Semantic Indexing for Document Categorization," Data Mining, IEEE International Conference on, pp. 535-538, Fourth IEEE International Conference on Data Mining (ICDM'04), 2004.

[28] T. Liu, Z. Chen, B. Zhang, W. Ma, G. Wu, "Improving text classification using local latent semantic indexing", Fourth IEEE International Conference on Data Mining, vol., no., pp. 162-169, 1-4 Nov. 2004

[29] S. Chakraborti, R. Mukras, R. Lothian, N. Wiratunga, S. Watt, and D. Harper, "Supervised Latent Semantic Indexing Using Adaptive Sprinkling", Proceedings of IJCAI, pages 1582–1587, 2007.

[30] F. Wild, C. Stahl, G. Stermsek, G. Neumann, Y. Penya, "Parameters Driving Effectiveness of Automated Essay Scoring with LSA", In Proceedings of the 9th CAA, pp.485-494, Loughborough, 2005

[31] F. Wild, "An LSA Package for R", In Mini-Proceedings of the 1st European Workshop on Latent Semantic Analysis in Technology-Enhanced Learning, Wild, Kalz, van Bruggen, and Koper (ed) , 11-12, March, 2007

[32] A. Karatzoglou, D. Meyer, K. Hornik, Support Vector Machines in R. Journal of Statistical Software, 2006, 15.