Scalable Association-based Text Classification

Dimitris Meretakis [1] Dimitris Fragoudis [2] Hongjun Lu [1] Spiros Likothanassis [2]

meretaks@cs.ust.hk dfragoud@ceid.upatras.gr luhj@cs.ust.hk likothan@cti.gr

[1] Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong, China

[2] Computer Engineering & Informatics Department
University of Patras
Rio - Patras, Greece

ABSTRACT

Naïve Bayes (NB) classifier has long been considered a core methodology in text classification mainly due to its simplicity and computational efficiency. There is an increasing need however for methods that can achieve higher classification accuracy while maintaining the ability to process large document collections. In this paper we examine text categorization methods from a perspective that considers the tradeoff between accuracy and scalability to large data sets and large feature sizes. We start from the observation that Support Vector Machines, one of the best text categorization methods cannot scale up to handle the large document collections involved in many real word problems. We then consider bayesian extensions to NB that achieve higher accuracy by relaxing its strong independence assumptions. Our experimental results show that LB, an association-based lazy classifier can achieve a good tradeoff between high classification accuracy and scalability to large document collections and large feature sizes.

Keywords

Statistical/probabilistic models, machine learning and IR, text categorization, text data mining

1. INTRODUCTION

Text classification is the supervised learning task of assigning natural language text documents to one or more predefined categories or classes according to their content. While it is a task with a history of almost forty years of research [13], it has recently attracted an increasing amount of attention due to the ever-expanding amount of text documents available in digital form in the World Wide Web, electronic mail, net news and digital libraries. Its applications span a number of areas including sorting email, filtering junk email, cataloguing news articles, providing relevance feedback and reorganizing large document collections.

While the quality of classification is the primal goal of competitive methods, the explosive growth in the amounts of textual data collected and stored introduces the need for methods that can scale up to handle large textual databases. In contrast to relational data mining where scalability is a main issue, the trade-off between quality and scalability has been largely ignored by a number of studies for text classification. Strikingly, the Naïve Bayes (NB) classifier [7] probably the most well studied and longest used method for text classification and Information Retrieval [13] attributes its success and longevity not only to its accuracy but also to its simplicity and to its low computational requirements. It scales up linearly both with the number of features and with the number of documents.

Although NB was used in the majority of supervised IR for a long time [13], a number of recent studies have shown that alternative, more complex algorithms often outperform NB in terms of classification performance. Such algorithms originate from a variety of research areas such as Nearest Neighbor [14], neural networks [18], regression [19], rule-induction [1],[6], and Support Vector Machines [11]. The accuracy improvements are often significant in a variety of real-world data sets (see also [20] for a comparative study of many algorithms).

Support Vector Machines (SVM) have been recognized as one of the most effective text classification methods. Unfortunately they cannot be applied to large volumes of data since their training time is quadratic to the number of training examples [12]. In this work we investigate whether extensions of NB can achieve higher accuracy performance without significant sacrifices in terms of scalability.

A careful examination of successful text classification methods at some level of abstraction suggests that their efficiency is to a large extend attributed on some common properties that NB does not posses and of which we focus on the following two ones: First, most methods deviate from the independence assumption and capture relationships among sets of words. The relaxation of the independence assumptions allows more complex models to be built that are expected to capture more aspects of the reality. Second, a number of methods focus on the use of context, that seems to be very important in text classification [6]. Context-insensitive classifiers build global models of the data, where the influence of each word to the final decision is independent of the presence/absence of other words in the same document. In natural languages, however, the meaning of words is clearly affected by the context in which they are used. The word “adopt” for example has a totally different meaning in the presence of word “child” than in the presence of the word “strategy”. Context-sensitive learning methods aim at higher performance by capturing such context-specific properties of the data. It is noteworthy that these two

properties are *orthogonal* i.e. a classifier may or may not posses one independently of the other.

Extensions of NB along these directions have been shown to yield higher classification accuracy but often at the cost of excessive computational requirements [8]. In this paper we focus on the use of two such extensions, namely Tree Augmented Naïve Bayes classifier (TAN) [9] and LB or Large Bayes Classifier [17], [15]. TAN is a global Bayesian classifier that relaxes the independence assumptions of NB by taking into account dependencies between pairs of features. LB is a very competitive local/lazy Bayesian classifier constructed from associations among feature-values, also called itemsets. Our experimental results show that LB with appropriate text representation can achieve a very good tradeoff between accuracy and scalability.

The rest of the paper is structured as follows. Section 2 contains our classification framework and discusses alternative forms of text representation. Section 3 provides an overview of NB and its two extensions TAN and LB as well as SVMs while Section 4 describes our experimental evaluation and discussion. Our conclusions are stated in Section 5.

2. TEXT REPRESENTATION AND CLASSIFICATION FRAMEWORK

For simplicity we chose the *multivariate Bernoulli model* or *binary independence event model* [16] as the basis for the representation of text documents. Documents are represented as vectors of binary features and each feature corresponds to the presence or absence of a single word. This model does not capture a large amount of information. More specifically both the order of words and the number of their appearances in a document are ignored; documents are represented as "bags of words". An alternative representation is the *multinomial model* where the number of occurrences of the words in a document is also captured. The latter provides more information and indeed it is reported to achieve higher performance for NB in a recent study over many data sets [16]. Its main drawback is that it assumes independence between the occurrences of the same word in a document, an assumption that is regularly violated in real text documents. For our experiments we chose the multivariate-Bernoulli model for the following reasons:

i. Relaxing the independence assumptions is simpler if the multivariate Bernoulli model is used [16]. This is particularly true when adapting algorithms from the relational machine learning paradigm, where the simple feature-vector representation is dominant. We are currently experimenting with the multinomial model and other models when the independence assumptions are relaxed.

ii. The multinomial model is usually more accurate with a large vocabulary. Although NB scales up linearly to the number of features, the performance of LB and TAN degrades faster as the number of features increases. Moreover [16] report that in the Reuters-21578 corpus a small feature size achieves highest performance (and the two models perform similar). Similarly, [8] report some of the highest accuracy results for SVMs on the Reuters-21578 dataset (which we use for our experiments) using only 300 features and the multivariate Bernoulli model.

The training database is a set of N documents $D=\{d_1, d_2, \ldots, d_N\}$. A vocabulary $V=\{w_1, w_2, \ldots w_n\}$ contains all n words used as features. This is constructed in the preprocessing stage. Each document d_j is then represented as a binary feature vector $d_j=<W_1,W_2,\ldots,W_n>$ where the value of feature W_l is one if word w_l is present in d_j and zero otherwise. Finally each document is labeled with none, one or more from a set of M classes $C=\{c_1,c_2,\ldots,c_M\}$. The data set is therefore represented as a relational N×n table (if we ignore the classes). This *relational* representation is very common for traditional machine learning algorithms, it is however inefficient in text classification since the number of features can be very large. A large improvement in space requirements is achieved if the data set is stored as a *transactional database* [2]. This requires that words are considered as *items* and each document is stored as a transaction consisting of the items it contains. Each transaction therefore contains exactly as much items as the words that appear in the corresponding document, which can be orders of magnitude less than the vocabulary size [1]. It is not clear to us how TAN can be extended to handle transactional databases. However, LB that has its roots in association mining inherently uses this representation and therefore can be easily applied. This is at the cost of ignoring information about words that do not appear in a document. On the other side transforming the relational database in to a transactional one allows absence of words to be captured by LB also. This is discussed more later.

3. AN OVERVIEW OF THE LEARNING METHODS

3.1 Naïve Bayes Classifier

The basic Naïve Bayes model assumes that the probability of appearance/absence for a word w_j is independent from the presence/absence of any other word w_k given that the class is known. Although unrealistic, this assumption allows the easy computation of the conditional probability of the class c_k given a document d_i as follows:

$$P(c_k \mid d_i) = \frac{P(c_k)\cdot P(d_i \mid c_k)}{P(d_i)} = \frac{P(c_k)\cdot \sum_{j=1}^{n} P(W_j \mid c_k)}{P(d_i)}$$

Therefore NB only needs to learn the marginal probability of the class $P(c_k)$ and the conditional probabilities of presence/absence of each word w_j given the class $P(W_j|c_k)$. This can easily be done in one pass over the data keeping in memory $2\cdot n\cdot M$ counters for the measurement of occurrence of each word under each class label. Its computational requirements are therefore minimal since it requires time linear both to the number of documents and to the number of features. New examples are also classified in time linear to the number of features.

3.2 TAN

TAN [9] extends NB taking into account additional dependencies between features. It employs a modified version of a method proposed in [5] to learn a restricted, tree-structured Bayesian Network that captures only the most important dependencies between pairs of features. The result of the learning phase is a Bayesian Network structure where each feature is connected to exactly another one and the class. Forcing the creation of a tree-structured network and incorporating in the model only pair-wise

[1] Note the similarity of this representation with the one used by RIPPER [6] where it is called "learning with set-valued features" (see also [4]).

dependencies allows the relaxing of the strong independence assumptions of NB and at the same time prevents the creation of complex Bayesian Network structures that are computationally more expensive often without paying in terms of classification accuracy.

In the learning phase TAN first measures the degree of dependence between each pair W_j and W_k by calculating their Conditional Mutual Information given the class C:

$$CMI(W_j, W_k \mid C) = \sum_{c, w_j, w_k} P(c, w_j, w_k) \cdot \log \frac{P(w_j, w_k \mid c,)}{P(w_j \mid c) \cdot P(w_k \mid c)}$$

The value of the *CMI* quantifies the degree of dependence of two words given the class. The highest the value of $CMI(W_j, W_k \mid C)$ the more the independence assumption is violated. If W_j and W_k are conditionally independent given the class *CMI* becomes zero. Subsequently, TAN forms a complete weighted graph where each node corresponds to a word and the weight of the edge (W_j ,W_k) is $CMI(W_j, W_k \mid C)$. A maximum spanning tree algorithm is then applied to the graph resulting to the tree-structured Bayesian Network model that captures the strongest pair-wise dependencies.

All $n \cdot (n-1)/2$ such computations, where n is the vocabulary size, can be performed with one pass over the data, provided that the $n \cdot (n-1)/2$ probability tables can fit in main memory. This requires bookkeeping during the learning phase of a total of $4 \cdot M \cdot n \cdot (n-1)/2$ counters for the measurement of all pair-wise co-occurrences (or co-absences) under each class label. During classification the Bayesian network is consulted and an estimation of the probability of each class is computed. The document is labeled after the most probable class. Learning time is linear to the number of examples but quadratic to the number of features. New examples are classified in time linear to the number of features.

3.3 LB

The main idea behind Large Bayes (LB) classifier [17], [15] is the use of association mining for classification. For LB each document is a transaction consisting of a set of items that express the presence or absence of specific words. The learning phase of LB employs an association miner [2] to discover associations among items, also called itemsets. Each such itemset l is labeled with the observed probabilities for its joint occurrence with each class label $P(l,c_i)$. The association mining procedure discovers itemsets that are frequent enough in the training data and are also interesting. The interestingness of an itemset l is defined in terms of the error when estimating $P(l,c_i)$ using subsets of l. Let l be an itemset of size $|l|$ and l_j , l_k be two ($|l|$-1)-itemsets obtained from l by omitting the j^{th} and k^{th} item respectively. Itemsets l_j, l_k can be used to produce an estimate $P_{j,k}(l, c_i)$ of $P(l, c_i)$:

$$P_{est}(l, c_i) = P_{j,k}(l, c_i) = \frac{P(l_j, c_i) \cdot P(l_k, c_i)}{P(l_j \cap l_k, c_i)}$$

The interestingness of l is directly related to the quality of this approximation, and in [15] we use chi-square tests to quantify it (N is the number of documents in the training set):

$$\chi^2 = \sum_{i=1}^{M} \frac{\left(P(l, c_i) \cdot N - P_{est}(l, c_i) \cdot N\right)^2}{P_{est}(l, c_i) \cdot N} = \sum_{i=1}^{M} \frac{\left(P(l, c_i) - P_{est}(l, c_i)\right)^2}{P_{est}(l, c_i)} * N$$

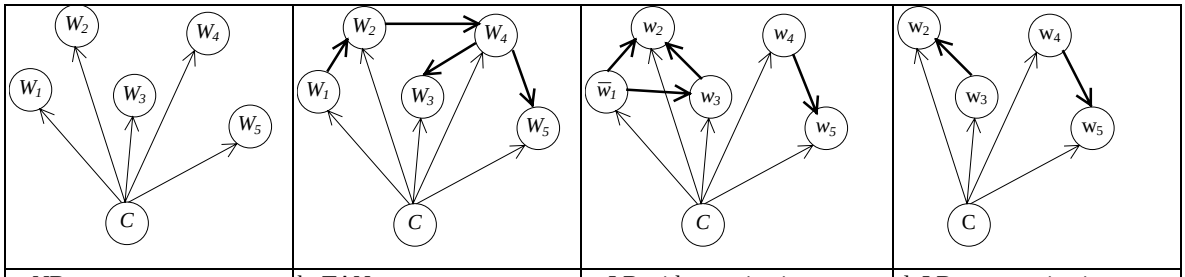
Itemsets with high interestingness values cannot be approximated by subsets and are therefore useful; all others are discarded. LB does not build any classification model during the learning phase; the discovered itemsets are the model.

To classify a new document d, LB first selects a *local border* consisting of all the longest itemsets that contain items from d and builds a local probabilistic model using these itemsets. This model is then evaluated to yield the probabilities for each class and the most probable class is assigned to d. This lazy approach allows LB to construct models that take into account the context imposed by each document under consideration. The constructed model and the associated independence assumptions only hold for the particular document. In the extreme case were only itemsets with one item are used LB reduces exactly to NB. The learning time of LB is tied to the performance of the association miner used for the discovery of the itemsets. Apriori-based association mining methods scale up linear with the number of examples performing several passes over the data but exponentially in the worst case with the number of features. The chi-square tests of LB, however, impose very tight constraints keeping low the number of itemsets discovered and consequently the learning time. More recent methods [10] however promise big computational improvements over Apriori-based ones and can be "plugged in" LB to perform the mining task more efficiently.

Note that if a transactional representation is used LB can capture information only about the presence of words in a document. However, if supplied with the relational representation both the presence and absence of words are taken into account (absent words are considered distinct items). As discussed in Section 4 there is significant difference in the behavior of the former version of LB which we will call LB+ and the latter one which we will name LB+-.

3.4 An illustrative example

In order to illustrate the properties of the three algorithms consider the following toy example. A database of documents is available and the vocabulary consists of 5 words $V=\{w_1,\ldots,w_5\}$. In the training phase NB and TAN will build the global models of figures 1a and 1b respectively (W_j are the word features that can take values 0 or 1). Note that in addition to the class dependencies induced by NB, TAN enhances the dependency graph with a tree of dependencies among pairs of features (the bold arrows). LB does not build any model but instead discovers a set of itemsets such as $\{\overline{w}_1, w_2, w_3\}$ (standing for $\{W_1=0, W_2=1, W_3=1\}$) and $\{w_4, w_5\}$.



a. NB	b. TAN	c. LB with negative items	d. LB, no negative items

Figure 1. Global (NB and TAN) and local (LB) models built from the three classifiers in the example of Section 3.4 for the classification of a document that contains words w_2, w_3, w_4 and w_5 but not w_1.

Assume now a query to classify a document *d* that contains words w_2, w_3, w_4 and w_5 but not w_1. TAN and NB will estimate $P(d,c_i)$ using the induced models and will assign the class with the highest probability:

a.NB:

$$P(d,c_i) = P(c_i)P(\overline{w}_1 \mid c_i)P(w_2 \mid c_i)P(w_3 \mid c_i)P(w_4 \mid c_i)P(w_5 \mid c_i)$$

b.TAN:

$$P(d,c_i) = P(c_i)P(\overline{w}_1, w_2 \mid c_i)P(w_4 \mid w_2, c_i)P(w_3 \mid w_4, c_i)P(w_5 \mid w_4, c_i)$$

LB will first select the longest itemsets that are subsets of the query document *d* and then it will build a local model in order to derive an estimation for $P(d,c_i)$. If negative items (i.e. absence of words) are used the model of figure 1c is constructed, otherwise item $\overline{w}_1$ is ignored and the model of figure 1d is created. Computation of $P(d,c_i)$ in the two cases is as follows:

c. LB,with negative items:

$$P(d,c_i) = P(c_i)P(\overline{w}_1, w_2, w_3 \mid c_i)P(w_4 \mid c_i)P(w_5 \mid w_4, c_i)$$

d. LB, no negative items:

$$P(d,c_i) = P(c_i)P(w_2, w_3 \mid c_i)P(w_4, w_5 \mid c_i)$$

3.5 Support Vector Machines

Support Vector Machines (SVM) are a relatively new approach for binary classification problems that stems from statistical learning theory. In their simplest form *linear SVMs* treat examples as points in a high-dimensional space. During training, their goal is to find hyperplanes that maximize the margin between positive and negative examples. The task is treated as a quadratic optimization problem and the resulting hyperplanes are used to classify new examples. SVMs are conceptually not related to the methods described above. Nevertheless, they have recently been shown to be highly effective for the task of text classification achieving the highest classification scores in standard text collections. While classification time is linear to the number of features, a major issue on the application of SVMs on large datasets is that their running time is quadratic in the number of training examples. In our experimental section we study the behavior of SVMs both in terms of accuracy and in terms of scalability using *SVMlight* package [11], an efficient implementation that has been used in several text classification studies.

4. EXPERIMENTAL EVALUATION

The goal of this section is to study the performance of the methods above along two directions:

I. Quality of classification measured in terms of recall and precision.

II. Scalability in the number of features and the number of documents.

We used the Reuters-21578 Distribution 1.0 that is available from [3]. This data set consists of 21,578 stories from the 1987 Reuters newswire, each one pre-assigned to one or more of a list of 135 topics. We used the '*Mod Apte'* training-test split that contains 9,603 training and 3,299 test examples and considered the 93 categories that are assigned to least one example in the training and testing data. All words were converted to lower case, punctuation marks were removed numbers were ignored and words from a common list of stop-words were removed. Finally a feature selection step was applied. For each class C we created a local dictionary consisting of the *K* words w_t with the highest average mutual information with *C*:

$$MI(C;W_t) = H(C) - H(C \mid W_t) = \sum_{c \in C} \sum_{w_t \in \{0,1\}} P(c, w_t) \log \frac{P(c, w_t)}{P(c) \cdot P(w_t)}$$

The classification task requires that a document may be assigned in none, one or more categories. We followed standard practice and treated the problem as a series of binary classification problems where the task is to decide whether the document belongs to a specific class or not. This procedure is repeated for all classes and the set of "on" classes is assigned to the document.

We evaluated performance using the standard Information Retrieval measures *recall* and *precision* defined as:

$$recall = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive examples}}$$

$$precision = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}}$$

To combine recall and precision with a single-value metric that can be used to derive a total order on the classifiers we use the F_1 measure, defined as follows:

$$F_1(recall, precision) = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

The algorithms were optimized to yield maximum F_1 scores using a validation test consisting of the last 2456 training stories. Algorithms were trained on the first 7147 documents, optimized on the 2456 validation documents and finally tested on the 3299 test stories.

An interesting particularity of the data that influences the interpretation of the results is that the category distribution is extremely skewed. There are 93 categories in total but as Figure 2 shows only a few categories appear relatively frequently in the training set. Since the performance of many classifiers appears to vary significantly with the number of positive examples in a class [20] we obtained separate results for the 10 largest classes (appearing more than 150 times each in the data) and for all the classes together.

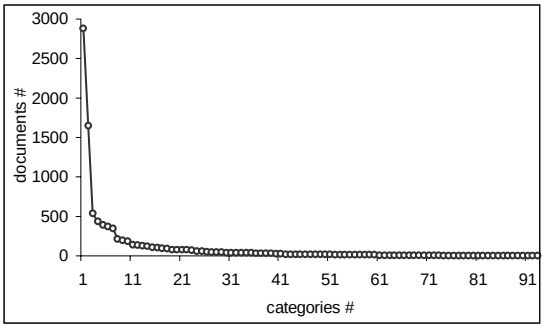


Figure 2. Category frequency distribution in Reuters-21578.

4.1 Classification quality

Classification accuracy results are presented in Table 1, which presents the recall, precision and F1 measures of the five algorithms on all classes for vocabulary sizes of 20, 50, 100, 200, 300 and 400 words. In order to summarize the recall, precision and F_1 measures over a set of different classes we used *micro-averaging*, that is cell-wise summing up of the confusion matrices over all the classes and then applying the measures on the resulting confusion matrix. The feature set size has different influence on each algorithm so the feature set size used for each result is also reported.

	10 most frequent categories				All 93 categories			
	m-recall	m-precision	m-F_1	|V|	m-recall	m-precision	m-F_1	|V|
NB	0.840	0.813	0.826	100	0.712	0.798	0.752	20
TAN	0.841	0.876	0.858	200	0.765	0.815	0.789	20
LB+	0.850	0.882	0.866	100	0.760	0.82	0.790	50
LB+-	0.877	0.871	0.874	300	0.783	0.814	0.798	50
SVM	0.861	0.916	0.888	300	0.767	0.904	0.830	100

Table 1. Best micro-averaged performance of classifiers and the corresponding vocabulary size on the ten largest categories (left) and on all 93 (right).

The effect of the feature size on the classification accuracy is shown in Figure 3 that plots the value of the F_1 score as a function of the feature set size. Table 1 and Figure 3 confirm results from other studies showing that SVMs are the most accurate methods for text classification. SVMs achieve higher performance both in the 10 most frequent classes and for the 93 classes in total. However the difference from the other methods is not as big as in other studies. NB sets the base level for comparison and is easily beaten by all other methods, however LB+ and LB+- offer consistently higher scores than TAN with LB+- being very close to SVMs.

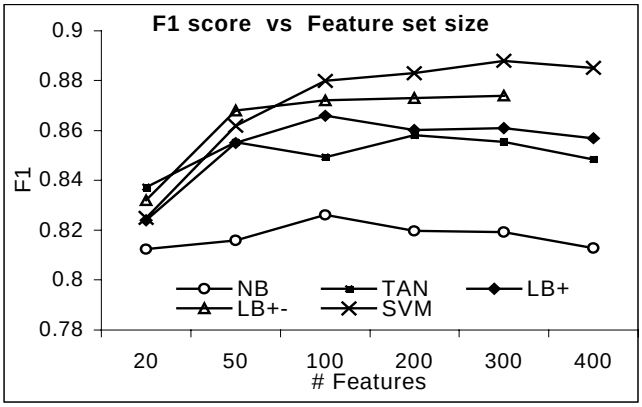


Figure 3. F_1 scores for NB, TAN, LB+, LB+- and SVMs as a function of the vocabulary size.

It is interesting to note that the information available to LB+ is less than that available for the other methods as it only uses information about the presence of words in a document. This is more apparent by a comparison between LB+ and LB+- where the only difference is that LB+- also considers information about the absence of words. Since the two methods differ only in the input representation we conclude that information about absence of words may also have predictive value. This is however at a very expensive price in terms of computational effort needed to handle it as shown in the next section.

We attribute the improvement of the both LB versions over TAN to a) the use of local models that capture context specific properties of the data and b) the capture of dependencies among more than two words (dependencies of up to five words were used by LB in the experiments above). The benefits are so big that even LB+ outperforms TAN, although it uses a poorer representation model. We also measured the relative contribution of factors a) and b) above to the performance gains. We choose LB+- and restricted it to discover itemsets with 2 items only thus eliminating the influence of factor b). The F_1 score fell but remained above that of TAN therefore we believe that the strength of LB comes mainly from its context-sensitive properties.

4.2 Scalability

In Figure 4 we show the training time by SVM and LB+ (on a 400MHz PentiumII WinNT PC) when applied in databases of up to nine times the size of the original one to learn the most common class ("earn"). The high accuracy of SVMs comes not for free. Their main drawback is that they require training time quadratic to the number of examples. In contrast, all other methods scale up linear with the data set size. This is an inherent property of SVMs that is caused by their attempt to treat the learning as a quadratic optimization problem and hinders their application in big datasets.

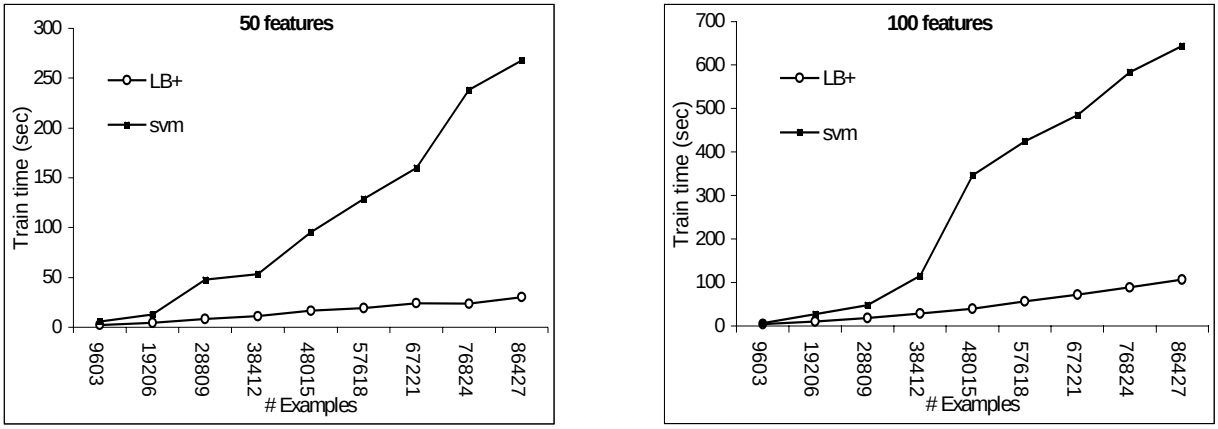


Figure 4. Training time of LB+ and SVM for the biggest class ("earn") and for multiples of the original dataset when the feature set size is 50 (left) and 100 (right) features.

Scalability over the feature set size has remained a major problem for most data mining/machine learning applications. Figure 5 shows the time to train and test the algorithms for the class "earn" as the feature set varies from 20 to 400 words. SVMs and NB scale up linearly with the number of features in both the learning and testing phase. TAN's training time increases quadraticaly with the feature size but its testing time is linear in the number of features. The graphs also show the significant differences in the performance of LB for the two different representation models. In principle the training time of LB using an Apriori-based itemset discovery engine is exponential in the number of features. Therefore, when both presence and absence of words is considered the combinatorial explosion prevents LB+- from using more than a few hundred features. However when only word presence is taken into account the input space is very sparse since most documents contain few of words from the vocabulary only. With these conditions LB+ scales up very well into many hundreds of features. This is particularly important in comparison with TAN because LB+ outperforms TAN in terms of accuracy and also scales up much better when the dimensionality increases.

An interesting finding of our experiments is that LB particularly benefits from stemming. In our datasets we did not use stemming and this caused the discovery of many unnecessary itemsets. For example the words "japan" and "japanese" are among the most informative words for class trade. The probability of co-occurrence of these two words in a document labeled as "trade" is very high as they are not independent. Therefore LB will consider the itemset {japan, japanese} as interesting. Also if a word x is associated with "japan" to form itemset {"japan",x} the duplicate itemset {"japanese", x"} will most likely also be identified. This causes unnecessary increase in the size and the complexity of the model without apparent improvement in the predictive power and could have been avoided if stemming was used.

LB is also the only algorithm in this study that does not scale up linearly during the testing phase. This is the price to be paid because LB builds a different model for each classification query. Observations for LB are similar as for the training phase with LB+- scaling up poorly whereas LB+ can scale up reasonably well. We believe that classification speed of LB can speed up if special data structures for storing the itemsets are used or if some caching method is applied. This is the focus of our ongoing work.

5. CONCLUSIONS

There are considerable tradeoffs in choosing an appropriate classification method for a given task. As already shown in previous studies SVM is definitely the most effective in terms of classification performance. Its inability to handle large data sets, however, raises the question for methods that can handle the large document collections encountered in many real problems. In this paper we considered Bayesian extensions of NB as alternatives. Global methods such as Bayesian Networks provide good results but scale up very poorly as the number of features increases. On

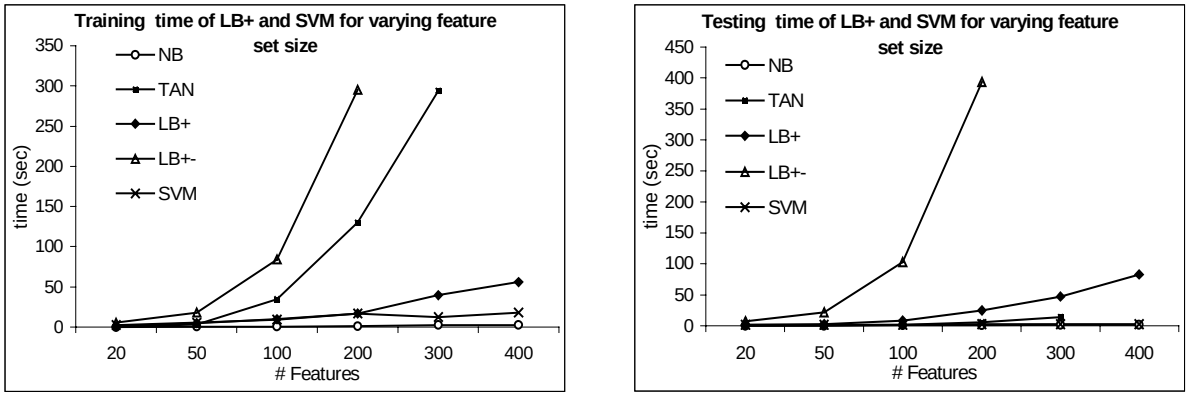


Figure 5. Train (left) and test (right) time for class "earn" for constant database size (9603 examples) and varying feature size from 20 to 400 features.

the other hand LB that follows a lazy probabilistic approach is a promising alternative. LB+- that uses a dense data representation capturing information about the presence and absence of words provides very competitive results but does not scale up to large feature spaces. Its alternative LB+ sacrifices the use of word absence information without loosing much of quality to gain significant improvement in terms of speed and scalability. We conclude that LB+ using this sparse data representation provides a good alternative that achieves high accuracy, scales up linear in the number of documents and handles large vocabulary sizes.

There are several research directions that we are currently exploring. We are interested in applying more complex document representation models for Bayesian approaches. Accuracy of NB is shown to substantially improve when the multinomial model is used. Our goal it to investigate if extensions of NB will gain similar gains. The higher performance of LB+- compared to LB+ raises the question whether the use of word absence information can be used selectively so as to yield accuracy gains without significantly decreasing the performance. Finally we are working on the integration of a faster association mining engine with LB. Results like those reported in [10] are promising and a successful integration with LB would translate to faster running times and better scalability when the dimensionality is increased.

6. ACKNOWLEDGMENTS

The third author's work is partially supported by grants from the National 973 project of China (No. G1998030414) and from the Research Grant Council of the Hong Kong SAR, China (AOE97/98.EG05). Our thanks to Rachna Kapur for proofreading and helping us with the experiments.

7. REFERENCES

[1] C. Apte, F. Damerau, and S. M. Weiss, “Automated learning of Decision Rules for Text Categorization”, *ACM TOIS*, Vol.12, No.3, pp.233 – 251, 1994.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", *VLDB-94*, 1994.

[3] S. D. Bay, *The UCI KDD Archive.* [http://kdd.ics.uci.edu], University of California, Dept. of Information & Computer Science, 1999.

[4] William Cohen, “Learning with set-valued features”, *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96),* 1996.

[5] C.K. Chow and C.N. Liu, “Approximating discrete probability distributions with dependence trees”, *IEEE Trans. on Information Theory,14*, 462-467, 1968.

[6] W. Cohen and Y. Singer, “Context-sensitive learning methods for text categorization”, *ACM TOIS*, Vol.17, No.2, pp.141 – 173, 1999.

[7] R.Duda, P.Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[8] S. Dumais, J. Platt, D. Heckerman and M. Sahami, “ Inductive Learning Algorithms and Representations for Text Categorization”, 7th ACM CIKM Conference , 1998.

[9] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers*"*, *Machine Learning*, *29*, 131-163, 1997.

[10] J.Han, J.Pei and Y.Yin, “Mining Frequent Patterns without Candidate Generation”,ACM SIGMOD,2000.

[11] T. Joachims, “Text Categoriation with Support Vector Machines: Learning with many Relevant Features”, in ECML’98, 1998.

[12] T. Joachims, “Making Large Scale SVM-Learning Practical”, in *Advances in Kernel Methods Support Vector Learning*, MIT Press, 1999.

[13] David D. Lewis, “Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval”, in *ECML-98*, 1998.

[14] B. Masand, G. Linoff, and D. Waltz, “Classifying news stories using memory based reasoning”, *15th ACM SIGIR Conference*, pp. 59-64, 1992.

[15] D. Meretakis, H. Lu and B. Wuthrich, "A study on the performance of Large Bayes Classifier". ECML-2000, Barcelona, Spain, 2000.

[16] A.McCallum and K.Nigam. “A Comparison of Event Models for Naive Bayes Text Classification”, *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48, AAAI Press, 1998

[17] D. Meretakis and B. Wüthrich, "Extending Naive Bayes Classifiers Using Long Itemsets" , *5th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, (KDD-99)*, San Diego, USA, 1999.

[18] T.H.Ng, W.B.Goh, and K.L. Low, “Feature selection, perceptron learning and a usability case study for text categorization”, *20th ACM SIGIR Conference*, 1997.

[19] Y. Yang and C. G. Chute, “An example-based mapping method for text categorization and retrieval”, *ACM TOIS*, Vol. 12, No 3, pp 252-277, 1994.

[20] Y. Yang and X. Liu, “An re-examination of text categorization”, *22nd ACM SIGIR Conference*, 1999.