A Comprehensive Empirical Comparison of Modern Supervised Classification and Feature Selection Methods for Text Categorization

Yindalon Aphinyanaphongs and Lawrence D. Fu
Center for Health Informatics and Bioinformatics, New York University Langone Medical Center, 227 East 30th Street, New York, NY, 10016 and Department of Medicine, New York University School of Medicine, 550 First Avenue, New York, NY, 10016. E-mail: {yin.a, lawrence.fu}@nyumc.org

Zhiguo Li, Eric R. Peskin, and Efstratios Efstathiadis
Center for Health Informatics and Bioinformatics, New York University Langone Medical Center, 227 East 30th Street, New York, NY, 10016. E-mail: {zhiguo.li, eric.peskin, efstratios.efstathiadis}@nyumc.org

Constantin F. Aliferis
Center for Health Informatics and Bioinformatics, New York University Langone Medical Center, 227 East 30th Street, New York, NY, 10016, Department of Pathology, New York University School of Medicine, 550 First Avenue, New York, NY, 10016, and Department of Biostatistics, Vanderbilt University, 1211 Medical Center Drive, Nashville, TN, 37232. E-mail: constantin.aliferis@nyumc.org

Alexander Statnikov
Center for Health Informatics and Bioinformatics, New York University Langone Medical Center, 227 East 30th Street, New York, NY, 10016 and Department of Medicine, New York University School of Medicine, 550 First Avenue, New York, NY, 10016. E-mail: alexander.statnikov@med.nyu.edu

An important aspect to performing text categorization is selecting appropriate supervised classification and feature selection methods. A comprehensive benchmark is needed to inform best practices in this broad application field. Previous benchmarks have evaluated performance for a few supervised classification and feature selection methods and limited ways to optimize them. The present work updates prior benchmarks by increasing the number of classifiers and feature selection methods order of magnitude, including adding recently developed, state-of-the-art methods. Specifically, this study used 229 text categorization data sets/tasks, and evaluated 28 classification methods (both well-established and proprietary/commercial) and 19 feature selection methods according to 4 classification performance metrics. We report several key findings that will be helpful in establishing best methodological practices for text categorization.

Received April 25, 2013; revised July 26, 2013; accepted August 11, 2013

 DOI: 10.1002/asi.23110

Introduction

Text categorization, the process of automatically classifying text documents into sets of predefined labels/classes, has become a popular area in machine learning and information retrieval (Joachims, 2002). Text categorization can be applied in any domain utilizing text. A prototypical example is content filtering for e-mail and websites. Spam e-mail detection models label incoming e-mail as spam or not spam (Androutsopoulos, Koutsias, Chandrinos, & Spyropoulos, 2000). Text categorization models also can identify spam websites or objectionable sites from a search engine index (Croft, Metzler, & Strohman, 2010). Another text categorization example is sentiment analysis (Melville, Gryc, & Lawrence, 2009). In this case, a model identifies opinions about movies or politics from websites. Text categorization also is commonly used in biomedicine (Zweigenbaum, Demner-Fushman, Yu, & Cohen, 2007). For example, patients with heart failure were automatically identified using unstructured text in the electronic medical record (Pakhomov et al., 2007). There are multiple successful applications of text categorization to filtering of

biomedical literature (Aphinyanaphongs & Aliferis, 2006; Aphinyanaphongs, Tsamardinos, Statnikov, Hardin, & Aliferis, 2005).

A very common, if not typical, workflow for text categorization contains several steps:

1. Input textual documents.
2. Parse the documents into tokens (e.g., words).
3. Build a document token matrix.
4. Represent the tokens as binary occurrences or a weighting such as term frequency–inverse document frequency (Manning, Raghavan, & Schütze, 2008).
5. Optionally apply a feature selection algorithm.
6. Apply a classifier to the document token matrix.
7. Evaluate performance of the classifier.

Depending on the task, each of these steps influences classification performance. Performance may vary by encoding different parts of a document (Manning et al., 2008), applying different tokenization rules (Hassler & Fliedl, 2006), varying token weighting (Leopold & Kindermann, 2002), applying various feature selection algorithms, and applying different classifiers (Forman, 2003; Genkin, Lewis, & Madigan, 2007; Yang & Pedersen, 1997).

Performing text categorization requires choosing appropriate methods for each step in the text categorization framework. The selected methods can significantly affect model performance depending on data set characteristics such as the sample size, number of input features, sparsity/distribution of the data, and distribution of the target/response variable, along with other factors. Currently, there are no theory-backed formulas for choosing optimal methods based on information about the data set, which is why researchers choose these methods based on good empirical performance on similar data sets and learning tasks. Comprehensive empirical benchmarks based on a wide variety of methods and data sets are needed to guide researchers in choosing optimal methods for their field. In addition to text categorization, empirical benchmarks are contributing to the development of guidelines for classification in many domains such as cancer classification from microarray gene expression data (Dudoit, Fridlyand, & Speed, 2002; Statnikov, Aliferis, Tsamardinos, Hardin, & Levy, 2005; Statnikov, Wang, & Aliferis, 2008), predicting survival from microarray gene expression data (Bovelstad et al., 2007), cancer classification from mass spectrometry data (Wu et al., 2003), classification from DNA methylation data (Zhuang, Widschwendter, & Teschendorff, 2012), classification from microbiomic data (Statnikov et al., 2013), hand-written digit recognition (Bottou et al., 1994; LeCun et al., 1995), music genre classification (Li, Ogihara, & Li, 2003), and clinical decision making (Harper, 2005).

This work provides a comprehensive benchmark of feature selection and classification algorithms for text categorization. Several prior studies have compared feature selection and classification methods for text categorization (Forman, 2003; Garnes, 2009; Genkin et al., 2007; Yang & Pedersen, 1997); however, these studies have focused on a small number of algorithms and did not include recently developed, state-of-the-art approaches. The main contribution of the present study is that it revisits and extends previous text categorization benchmarks of feature selection and classification methods to provide a more expanded and modern comparative assessment of methods (Table 1). In total, we used 229 data sets/tasks for text categorization, 28 classification methods (both well-established and proprietary/commercial ones such as the Google Prediction API), 19 feature selection methods, and four classification-performance metrics in the design of 10-times repeated, fivefold cross-validation.

TABLE 1. Comparison of the current study with prior work with respect to the number of feature selection and classification methods used.

Study	No. of feature selection methods	No. of classification methods
Yang & Pedersen (1997)	5[a]	2
Forman (2003)	12[a]	1
Genkin et al. (2007)	3[a]	3
Garnes (2009)	17[a]	2
Current study	19	28

Note. [a]Only limited feature selection has been performed, whereby features were ranked by a univariate metric (that takes into account only one token), and their performance was assessed by a classifier for multiple feature subset sizes. However, no single feature set has been selected and consecutively evaluated in independent testing data.

Methods

Data sets and Corpora

Table 2 lists corpora and data sets/tasks used in the study. Overall, we used 229 data sets/tasks that originate from 20 corpora. Corpora in this study were used in the seminal benchmarking work by Forman (2003) and a more recent Text Retrieval Conference (TREC) 2010 challenge (http://trec.nist.gov/). Each document in the corpus contains a classification label, which was assigned and verified by manual review. In each corpus, we constructed multiple data sets for binary classification of documents into each class (positive) versus the rest (negative). We removed all data sets that had less than 10 documents in the positive class to ensure that we have at least two positive documents in each of the five folds of cross-validation.

Data-Preparatory Steps

We applied several data-preparatory steps before applying feature selection and classification methods. First, we removed words that appear below a cutoff threshold. Second, we applied the Porter stemming algorithm, which reduces words to their base form (Porter, 1980). For example, "smoking," "smoker," and "smoked" all refer to the single concept of "smoke." Third, we used a stop word

DOI: 10.1002/asi

TABLE 2. Corpora and data sets/tasks used in the study.

Corpus	Source	No. of documents	No. of classes	No. of positive documents for each class
cora36	whizbang.com	1,800	36	50 (each)
fbis	TREC	2,463	17	38 43 46 46 46 48 65 92 94 121 125 139 190 358 387 506
la1	TREC	3,204	6	273 341 354 555 738 943
la2	TREC	3,075	6	248 301 375 487 759 905
oh0	OHSUMED	1,003	10	51 56 57 55 71 76 115 136
oh5	OHSUMED	918	10	59 61 61 72 74 85 93 120 144 149
oh10	OHSUMED	1,050	10	52 60 61 70 87 116 126 148 165 165
oh15	OHSUMED	913	10	53 56 56 66 69 98 98 106 154 157
ohscal	OHSUMED	11,162	10	709 764 864 1001 1037 1159 1260 1297 1450 1621
re0	Reuters-21578	1,504	13	11 15 16 20 37 38 39 42 60 80 219 319 608
re1	Reuters-21578	1,657	25	10 13 15 17 18 18 19 19 20 20 27 31 31 32 37 42 48 50 60 87 99 106 137 330 371
tr11	TREC 1990s	414	8	11 20 21 29 52 69 74 132
tr12	TREC 1990s	313	7	29 29 30 34 35 54 93
tr21	TREC 1990s	336	4	16 35 41 231
tr23	TREC 1990s	204	5	11 15 36 45 91
tr31	TREC 1990s	927	6	21 63 111 151 227 352
tr41	TREC 1990s	878	9	18 26 33 35 83 95 162 174 243
tr45	TREC 1990s	690	10	14 18 36 47 63 67 75 82 128 160
trnew	TREC 2010	8,245	8	19 59 67 80 168 230 333 1006
wap	WebACE	1,560	19	11 13 15 18 33 35 37 40 44 54 65 76 91 91 97 130 168 196 341

list to remove words that do not carry any semantic value (http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T43/), such as "the," "a," "each," "for," and so on. Finally, we converted the word frequencies in each document to the term-frequency/inverse document frequency (tf–idf) weighting, which is typically used for representing text data for machine learning algorithms (Manning et al., 2008).

These four steps were applied in each data set, excluding cora36 because we did not have access to the raw text data. For all data sets from a study by Forman (2003), we applied a cutoff threshold of three documents (i.e., terms appearing in fewer than three documents were removed), the Porter stemming algorithm, and stop word removal. Then, we converted the word frequencies to tf–idf form. For TREC 2010 data sets, we applied a cutoff of five documents, the Porter stemming algorithm, and stop word removal, and then weighted the word frequencies according to tf–idf.

Classification Methods

We used 27 previously published, supervised classification algorithms from the following 12 algorithmic families: standard support vector machines (SVMs); SVMs weighting by class prior probabilities; SVMs for optimizing multivariate performance measures, transductive SVMs; L2-regularized L1-loss SVMs, L1-regularized L2-loss SVMs, L1-regularized logistic regression, L2-regularized logistic regression, kernel ridge regression, naïve Bayes, Bayesian logistic regression, and AdaBoostM1. These classification algorithms were chosen because of their appropriateness for high-dimensional text data and the extensive number of successful applications to text categorization.

In addition to the aforementioned methods, we also evaluated a proprietary classification algorithm from the Google Prediction API Version 1.3 (https://developers.google.com/prediction/), which is gaining popularity in commercial text categorization applications on the web (https://groups.google.com/forum/?fromgroups=#!forum/prediction-api-discuss). Since the Google Prediction API limits training data size to 250 MB (in specialized nonsparse CSV format), this method was run on only 199 of 229 data sets, satisfying this size requirements.

The description of all employed classification algorithms and software implementations is provided in Table 3.

Classifier Parameters

We selected parameters for the classification algorithms by the nested cross-validation procedure that is described in the following subsection. We also included classifiers with default parameters for comparison purposes. Table 4 describes the parameter values for each classifier.

Model/Parameter Selection nd Performance Estimation

For model/parameter selection and performance estimation, we used nested repeated fivefold cross-validation procedure (Braga-Neto & Dougherty, 2004; Kohavi, 1995; Scheffer, 1999; Statnikov, Tsamardinos, Dosbayev, & Aliferis, 2005). The inner loop of cross-validation was used to determine the best parameters of the classifier (i.e., values of parameters yielding the best classification performance for the validation data set). The outer loop of cross-validation was used for estimating the classification

DOI: 10.1002/asi

TABLE 3. Supervised classification methods used in the study, high-level description of parameters and software implementations (see the Appendix for a review of the basic principles of these methods).

Method	Description (implementation)	References
svm1_libsvm	Standard SVMs: linear kernel, default penalty parameter *C* (*libsvm*)	Chang & Lin, 2011; Fan, Chen, & Lin, 2005; Vapnik, 1998
svm2_libsvm	Standard SVMs: linear kernel, penalty parameter *C* selected by cross-validation (*libsvm*)	
svm3_libsvm	Standard SVMs: polynomial kernel, penalty parameter *C* and kernel degree *q* selected by cross-validation (l*ibsvm*)	
svm1_weight	SVMs with penalty weighting by class prior probabilities: linear kernel, default penalty parameter *C* (*libsvm*)	
svm2_weight	SVMs with penalty weighting by class prior probabilities: linear kernel, penalty parameter *C* selected by cross-validation (*libsvm*)	
svm3_weight	SVMs with penalty weighting by class prior probabilities: polynomial kernel, penalty parameter *C* and kernel degree *q* selected by cross-validation (*libsvm*)	
svm1_light	Standard SVMs: linear kernel, default penalty parameter *C* (*svmlight*)	Joachims, 2002; Vapnik, 1998
svm2_light	Standard SVMs: linear kernel, penalty parameter *C* selected by cross-validation (*svmlight*)	
svm3_light	Standard SVMs: polynomial kernel, penalty parameter *C* and kernel degree *q* selected by cross-validation (*svmlight*)	
svm4_light	Standard SVMs: linear kernel, fixed penalty parameter *C* (*svmlight*)	
svm1_perf	SVMs for optimizing multivariate performance measures: optimized for AUC, linear kernel, default penalty parameter *C* (*svmperf*)	Joachims, 2005; Vapnik, 1998
svm2_perf	SVMs for optimizing multivariate performance measures: optimized for AUC, linear kernel, penalty parameter *C* selected by cross-validation (*svmperf*)	
svm3_perf	SVMs for optimizing multivariate performance measures: optimized for AUC, polynomial kernel, penalty parameter *C* and kernel degree *q* selected by cross-validation (*svmperf*)	
svm4_perf	SVMs for optimizing multivariate performance measures: optimized for AUC, linear kernel, fixed penalty parameter *C* (*svmperf*)	
svm1_trans	Transductive SVMs: linear kernel, default penalty parameter *C* (*svmlight*)	Joachims, 1999, 2002; Vapnik, 1998
svm2_trans	Transductive SVMs: linear kernel, penalty parameter *C* selected by cross-validation (*svmlight*)	
svm3_trans	Transductive SVMs: polynomial kernel, penalty parameter *C* and kernel degree *q* selected by cross-validation (*svmlight*)	
svm4_trans	Transductive SVMs: linear kernel, fixed penalty parameter *C* (*svmlight*)	
svm1_l2l1	L2-regularized L1-loss SVMs: linear kernel, default penalty parameter *C* (*liblinear*)	Fan, Chang, Hsieh, Wang, & Lin, 2008; Vapnik, 1998
svm2_l2l1	L2-regularized L1-loss SVMs: linear kernel, penalty parameter *C* selected by cross-validation (*liblinear*)	
svm_l1l2	L1-regularized L2-loss SVMs: linear kernel, penalty parameter *C* selected by cross-validation (*liblinear*)	
lr1	L1-regularized logistic regression: penalty parameter *C* selected by cross-validation (*liblinear*)	Fan et al., 2008; Lin, Weng, & Keerthi, 2008
lr2	L2-regularized logistic regression: penalty parameter *C* selected by cross-validation (*liblinear*)	
krr	Kernel ridge regression: polynomial kernel, ridge parameter selected by cross-validation (*clop*)	Guyon, 2005; Guyon et al., 2006; Hastie, Tibshirani, & Friedman, 2001
nb	Naïve Bayes: using multinomial class conditional distribution (*Matlab Statistics Toolbox*)	Mitchell, 1997
blr	Bayesian logistic regression: Gaussian priors, variance parameter selected by cross-validation (*bbr*)	Genkin, Lewis, & Madigan, 2004; Genkin et al., 2007
adaboostm1	AdaBoostM1: using classification trees as base learners (*Matlab Statistics Toolbox*)	Freund & Schapire, 1996
google	Google Prediction API: Proprietary Google Prediction API for pattern matching and machine learning (http://developers.google.com/prediction/)	–

performance of the model that was built using the previously found best parameters by testing with an *independent set of samples*. To account for variance in performance estimation, we repeated this entire process (nested fivefold cross-validation) for 10 different splits of the data into five cross-validation testing sets and averaged the results (Braga-Neto & Dougherty, 2004).

Feature Selection Methods

The number of input features in text data is typically very large because of the number of unique terms in the corpus, and the number of features also increases with the number of documents in the corpus. Feature selection is often performed to remove irrelevant or redundant features before training of the classification algorithm. The benefits of feature selection include facilitating data visualization and data understanding, improving computational efficiency by reducing the amount of computation and storage during model training and application, increasing prediction performance, and reducing the required amount of training data (Forman, 2003; Guyon & Elisseeff, 2003).

We used 19 feature selection algorithms (including no feature selection) from the following three broad algorithmic families: causal graph-based feature selection by Generalized Local Learning methodology (GLL-PC

DOI: 10.1002/asi

TABLE 4. Parameters of supervised classification methods used in the study.

Method	Parameter	Value(s)
svm1_libsvm	*C* (error penalty)	1
svm2_libsvm	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm3_libsvm	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
	q (polynomial degree)	optimized over (1, 2, 3)
svm1_weight	*C* (error penalty)	1
svm2_weight	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm3_weight	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
	q (polynomial degree)	optimized over (1, 2, 3)
svm1_light	*C* (error penalty)	$\frac{1}{N\left(\sum\sqrt{X^T X}\right)^2}$ where *X* is training data and *N* is number of samples in the training data
svm2_light	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm3_light	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
	q (polynomial degree)	optimized over (1, 2, 3)
svm4_light	*C* (error penalty)	1
svm1_perf	*C* (error penalty)	0.01
svm2_perf	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm3_perf	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
	q (polynomial degree)	optimized over (1, 2, 3)
svm4_perf	*C* (error penalty)	*N*/100, where is number of samples in the training data
svm1_trans	*C* (error penalty)	$\frac{1}{N\left(\sum\sqrt{X^T X}\right)^2}$ where *X* is training data and *N* is number of samples in the training data
svm2_trans	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm3_trans	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
	q (polynomial degree)	optimized over (1, 2, 3)
svm4_trans	*C* (error penalty)	1
svm1_l2l1	*C* (error penalty)	1
svm2_l2l1	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
svm_l1l2	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
lr1	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
lr2	*C* (error penalty)	optimized over (0.01, 0.1, 1, 10, 100)
Krr	*ridge*	optimized over (10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1)
	q (polynomial degree)	optimized over (1, 2, 3)
Nb	*prior*	use empirical priors from training data
Blr	*prior*	Gaussian
adaboostm1	*nlearn* (no. of learners)	100
google	–	–

algorithm), support vector machine-based recursive feature elimination (SVM-RFE), and backward wrapping based on univariate association of features. Each of the 19 algorithms was applied with all classifiers. The feature selection algorithms and software implementations are described in Table 5. These feature selection algorithms were chosen because of their extensive and successful applications to various text categorization tasks as well as general data analytics. We emphasize that all feature selection methods were applied during cross-validation utilizing only the training data and splitting it into smaller training and validation sets, as necessary. This ensures integrity of the model performance estimation by protecting against overfitting.

Classification Performance Metrics

We used the area under the receiver operating characteristic (ROC) curve (AUC) as the primary classification performance metric. The ROC curve is the plot of sensitivity versus 1-specificity for a range of threshold values on the outputs/predictions of the classification algorithms (Fawcett, 2003). AUC ranges from 0 to 1, where AUC = 1 corresponds to perfectly correct classification of documents, AUC = 0.5 corresponds to classification by chance, and AUC = 0 corresponds to an inverted classification. We chose AUC as the primary classification performance metric because it is insensitive to unbalanced class prior probabilities, it is computed over the range of sensitivity-specificity tradeoffs at various classifier output thresholds, and it is more discriminative than are metrics such as accuracy (proportion of correct classifications), F-measure, precision, and recall (Ling, Huang, & Zhang, 2003a, 2003b).

To facilitate comparison with prior literature on text categorization, we also used precision, recall, and the F-measure. Precision is the fraction of retrieved (classified) relevant documents that are relevant. Precision is sensitive to unbalanced class prior probabilities (Fawcett, 2003). Recall is the fraction of relevant documents retrieved (classified) by algorithms. Recall is the same as sensitivity, and it is not sensitive to unbalanced class prior probabilities (Fawcett, 2003). The F-measure is defined as 2(recall × precision)/(recall + precision). This metric is sensitive to unbalanced class prior probabilities (Fawcett, 2003), and also is known as the F_1 measure because recall and precision are equally weighted.

Statistical Comparisons

To test whether the differences in performance (average classification AUC, precision, recall, F-measure, or number of selected features) between the algorithms are nonrandom, we used a permutation test, adapted from Menke and Martinez (2004). An example of this test for the AUC performance metric is shown in Figure A1 in the Appendix. For comparison of two algorithms *X* and *Y*, the test involves the following steps:

DOI: 10.1002/asi

TABLE 5. Feature selection methods used in the study (see the Appendix for a review of the basic principles of these methods and additional information about their parameters).

Method	Description (implementation)	References
all	Using no feature selection	–
gll_g_k1	GLL-PC: G^2 test, $\alpha = 0.05$, *max-k* = 1 (*Causal Explorer*)	Aliferis, Statnikov, Tsamardinos, Mani, & Koutsoukos, 2010a, 2010b; Aphinyanaphongs & Aliferis, 2006; Aphinyanaphongs et al., 2005
gll_g_k2	GLL-PC: G^2 test, $\alpha = 0.05$, *max-k* = 2 (*Causal Explorer*)	
gll_g_k3	GLL-PC: G^2 test, $\alpha = 0.05$, *max-k* = 3 (*Causal Explorer*)	
gll_z_k1	GLL-PC: Fisher's *Z* test, $\alpha = 0.05$, *max-k* = 1 (*Causal Explorer*)	
gll_z_k2	GLL-PC: Fisher's *Z* test, $\alpha = 0.05$, *max-k* = 2 (*Causal Explorer*)	
gll_z_k3	GLL-PC: Fisher's *Z* test, $\alpha = 0.05$, *max-k* = 3 (*Causal Explorer*)	
svm_rfe1	SVM-RFE: with statistical comparison (*internal implementation on top of libsvm*)	Guyon, Weston, Barnhill, & Vapnik, 2002
svm_rfe2	SVM-RFE: without statistical comparison (*internal implementation on top of libsvm*)	
u1_auc	Backward wrapping based on univariate association of features: using area under ROC curve (AUC) to measure association; with statistical comparison (*internal implementation on top of libsvm*)	Fawcett, 2003; Guyon & Elisseeff, 2003
u2_auc	Backward wrapping based on univariate association of features: using area under ROC curve (AUC) to measure association; without statistical comparison (*internal implementation on top of libsvm*)	
u1_bs	Backward wrapping based on univariate association of features: using Bi-Normal separation to measure association; with statistical comparison (*internal implementation on top of libsvm and Matlab Statistics Toolbox*)	Forman, 2003; Guyon & Elisseeff, 2003
u2_bs	Backward wrapping based on univariate association of features: using Bi-Normal separation to measure association; without statistical comparison (*internal implementation on top of libsvm and Matlab Statistics Toolbox*)	
u1_ig	Backward wrapping based on univariate association of features: using information gain to measure association; with statistical comparison (*internal implementation on top of libsvm*)	
u2_ig	Backward wrapping based on univariate association of features: using information gain to measure association; without statistical comparison (*internal implementation on top of libsvm*)	
u1_df	Backward wrapping based on univariate association of features: using document frequency to measure association; with statistical comparison (*internal implementation on top of libsvm*)	
u2_df	Backward wrapping based on univariate association of features: using document frequency to measure association; without statistical comparison (*internal implementation on top of libsvm*)	
u1_x2	Backward wrapping based on univariate association of features: using χ^2 test to measure association; with statistical comparison (*internal implementation on top of libsvm and Matlab Statistics Toolbox*)	
u2_x2	Backward wrapping based on univariate association of features: using χ^2 test to measure association; without statistical comparison (*internal implementation on top of libsvm and Matlab Statistics Toolbox*)	

1. Define the null hypothesis (H_0) to be: The average performance (across all data sets and cross-validation testing sets) of algorithms *X* and *Y* is the same. Compute the absolute value of the observed average differences in performance of algorithms *X* and *Y* ($Y\left(\hat{\Delta}_{XY}\right)$).
2. Repeatedly randomly rearrange the performance values of algorithms *X* and *Y* (independently for each data set and cross-validation testing set), and compute the absolute value of the average differences in performance of algorithms *X* and *Y* in permuted data. Repeat the steps for 20,000 permutations to obtain the null distribution of Δ_{XY}, the estimator of the true unknown absolute value of the average differences in performance of the two algorithms.
3. Compute the cumulative probability (*p*) of Δ_{XY} being greater than or equal to the observed difference $\hat{\Delta}_{XY}$ over all permutations.

This process was repeated for each of the 10 data splits into five cross-validation folds, and the *p* values were averaged. If the resulting *p* value was smaller than .05, we rejected H_0 and concluded that the data support that algorithms *X* and *Y* do not have the same performance, and this difference is not due to sampling error. The procedure was run separately for each performance metric.

Finally, when we performed a comparison of multiple methods, we adjusted all *p* values for multiple comparisons following the method of Benjamini and Hochberg (1995) and Benjamini and Yekutieli (2001).

Computing Resources and Infrastructure

For this project, we used the Asclepius High Performance Computing Cluster at the NYU Center for Health Informatics and Bioinformatics consisting of ~1,000 latest Intel x86 processing cores, 4 TB of RAM distributed among the cluster's compute nodes, and 584 TB of disk storage. To make the computations feasible, we divided the problem into many independent jobs, each implemented in Matlab, R, C/C++, or both. The completely independent nature of the jobs enabled linear speedup. We typically used 100 cores of the cluster at a time. The project required 50 core-years of computation and completed in roughly 6 months of elapsed time.

Experiments with Google Prediction API were performed using the infrastructure of Google and scripting files to automate the entire process. Once the data had been prepared in special format, they were uploaded to Google

DOI: 10.1002/asi

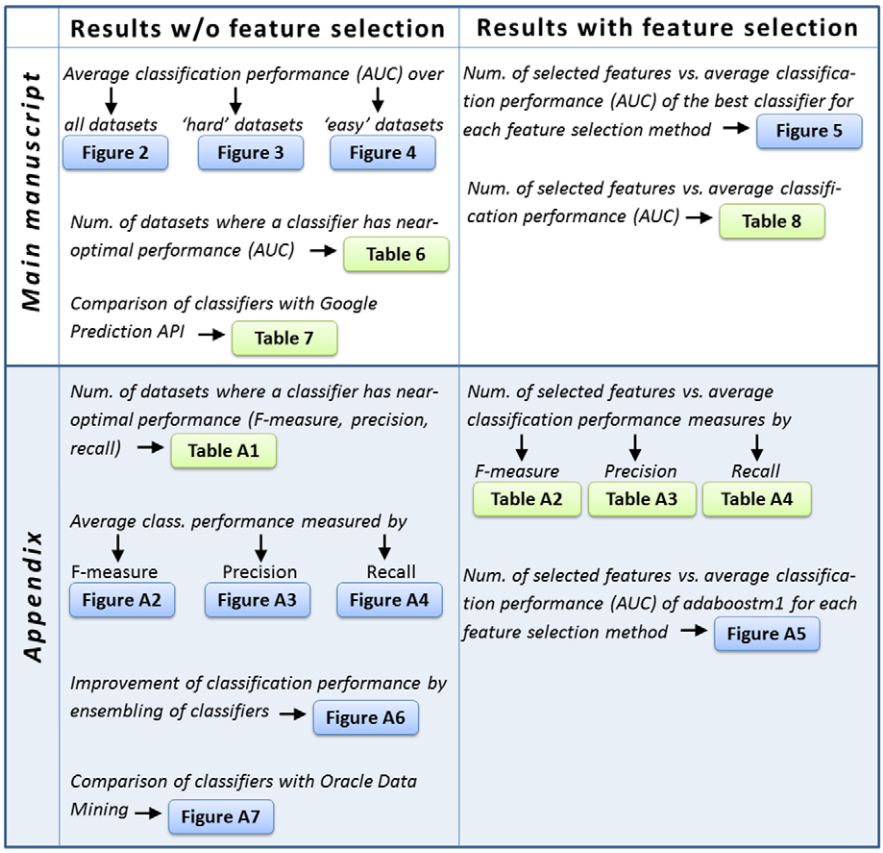

FIG. 1. A road map of results obtained in this work. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Cloud Storage. We then trained a model with the Google Prediction API and ran classification queries following our cross-validation design. While working with Google infrastructure, we used the OAuth 2.0 scheme for authorization.

Results

Figure 1 shows a road map of figures and tables that contain results of this work. A discussion of key findings with references to specific figures and tables is provided next.

Results Without Feature Selection

To facilitate interpretation of results, we operationally defined "easy" data sets as the ones with the highest AUC (top 25%), on average, over all classifiers. We defined "hard" data sets as the ones with the lowest AUC (bottom 25%), on average, over all classifiers. There are 57 easy and 57 hard data sets.

Our analysis of the classification performance of classifiers without feature selection starts with Figures 2 to 4. Each bar in each figure shows mean classification performance measured by the AUC with the 20th to 80th percentile across all data sets, the 57 hard data sets, and the 57 easy data sets, respectively. In each figure, we show with dark bars the top-performing classifier and any other classifier which has statistically indistinguishable classification performance.

Among all data sets (Figure 2) and the hard data sets (Figure 3), the best performing classifiers by AUC are AdaBoostM1 and support vector machines optimized for AUC with a linear kernel and a penalty parameter selected by cross-validation (*svm_perf2*). Among the easy data sets (Figure 4), mean AUC performance is relatively stable across half of the classifiers. Again, AdaBoostM1 has the highest AUC performance, although the best performing support vector machines are statistically distinguishable from AdaBoostM1. The same comparisons also show that naïve Bayes is the worst performing classification method, on average, with respect to the AUC metric. For completeness, we depict mean classifier performance across all data sets for F-measure, precision, and recall (Figures A2–A4 in the Appendix).

In Table 6, we provide a granular quantitative view of classifier performance. For each classification method, we show in how many data sets the method achieved an AUC within 0.01 of the best performing method (i.e., it is "near-optimal"). For example, naïve Bayes (*nb*) performed within 0.01 of the top-performing classifier in 54 of 229 data sets, in 28 of 57 easy data sets, and in only 2 of 57 hard data sets. Consistent with our previous analysis of Figures 2 to 4, AdaBoostM1 has higher AUC performance, as compared to

DOI: 10.1002/asi

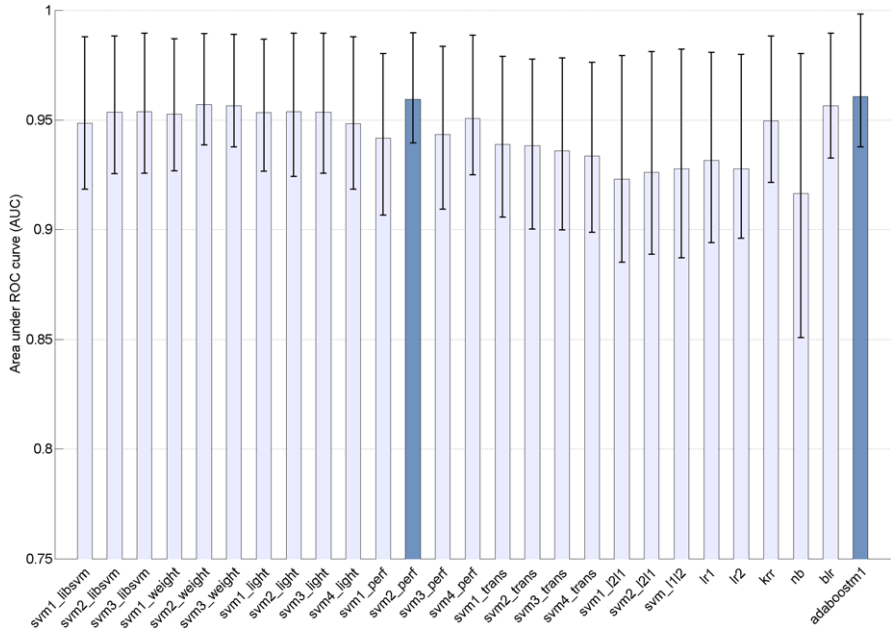

FIG. 2. Mean and 20th to 80th percentile interval of the area under the curve computed over all 229 data sets in the study. Dark bars correspond to classifiers that are statistically indistinguishable from the best performing one (adaboostm1); light bars correspond to all other classifiers. The *y*-axis is magnified for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

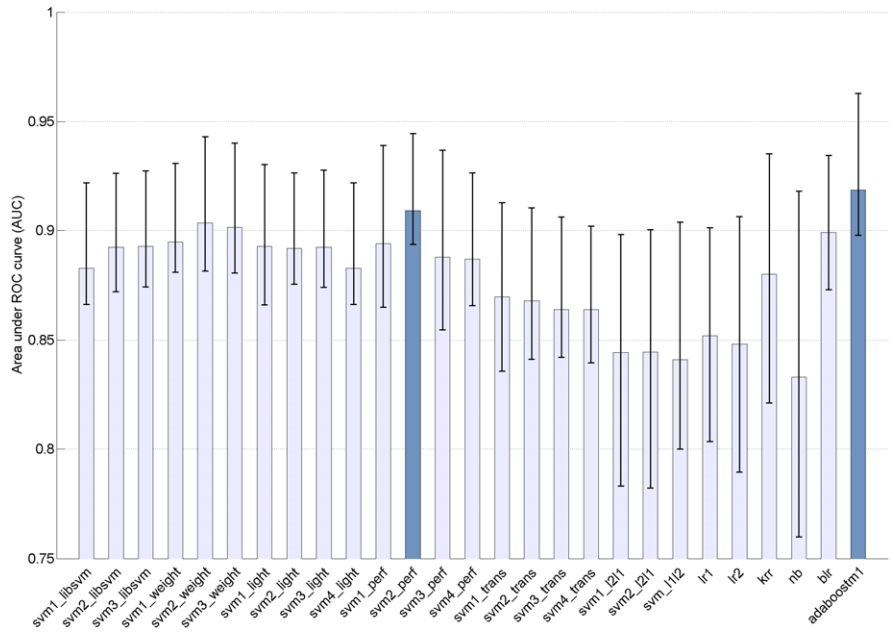

FIG. 3. Mean and 20th to 80th percentile interval of the area under the curve computed over 57 "hard" data sets in the study. Dark bars correspond to classifiers that are statistically indistinguishable from the best performing one (adaboostm1); light bars correspond to all other classifiers. The *y*-axis is magnified for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

other classifiers. AdaBoostM1 performed within 0.01 of the best performing classifier in 160 of 229 data sets, in 53 of 57 easy data sets, and in 34 of 57 hard data sets. On the other hand, transductive SVMs (*svm_trans1, svm_trans2, svm_trans3, and svm_trans4*) perform worse than other methods for the same comparisons. This is an unexpected finding because transductive classifiers have, in theory, an advantage over conventional classifiers by having access to unlabeled testing data. Also note that there is no single classifier that performs within 0.01 of the best performing classifier, either in all 229 data sets or even in all 57 easy or 57 hard data sets. For completeness, we extend this analysis

DOI: 10.1002/asi

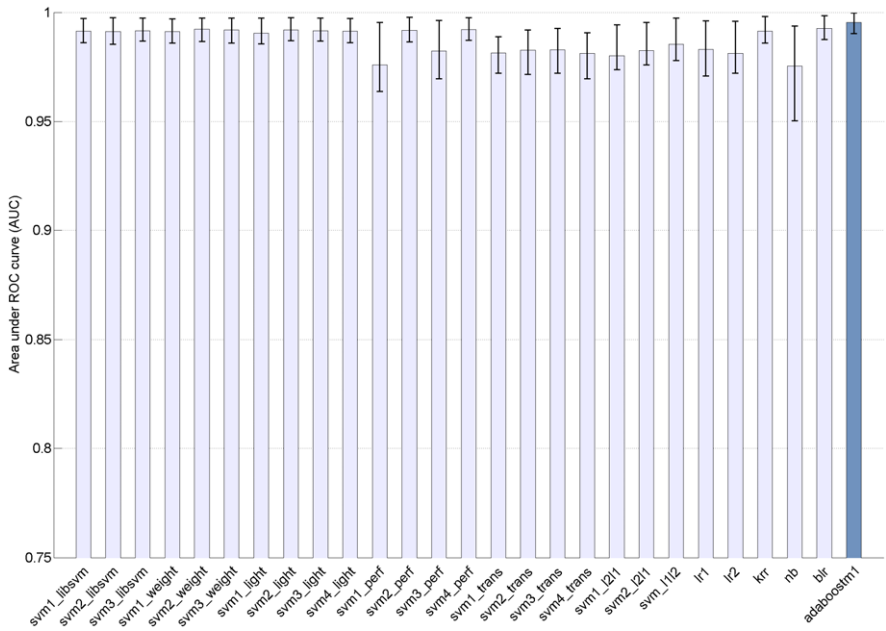

FIG. 4. Mean and 20th to 80th percentile interval of the area under the curve computed over 57 "easy" data sets in the study. There are no classifiers that are statistically indistinguishable from the best performing one (adaboostm1), shown with the dark bar. The *y*-axis is magnified for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE 6. Number of data sets when a classification method achieved area under ROC curve (AUC) within 0.01 of the nominally best performing method (i.e., it is "near-optimal").

Method	*n* data sets for top AUC (over all 229)	*n* data sets for top AUC (over "easy" 57)	*n* data sets for top AUC (over "hard" 57)
svm1_libsvm	76	48	2
svm2_libsvm	101	46	7
svm3_libsvm	105	49	8
svm1_weight	86	46	6
svm2_weight	116	50	16
svm3_weight	115	49	15
svm1_light	103	42	8
svm2_light	107	50	8
svm3_light	105	49	8
svm4_light	76	48	2
svm1_perf	73	27	9
svm2_perf	130	49	18
svm3_perf	79	29	12
svm4_perf	84	49	2
svm1_trans	35	12	1
svm2_trans	40	21	2
svm3_trans	35	23	1
svm4_trans	23	19	1
svm1_l2l1	41	32	2
svm2_l2l1	44	32	1
svm_l1l2	54	37	2
lr1	50	30	1
lr2	53	29	3
krr	100	44	8
nb	54	28	2
blr	116	49	6
adaboostm1	160	53	34

TABLE 7. Comparison of classifiers with Google Prediction API.

Performance metric	Result of Google	Result of the best classifier (for each performance metric)	No. of classifiers with better performance than Google
Area under ROC curve	0.909	0.962	27
F-measure	0.497	0.736	17
Precision	0.588	0.850	23
Recall	0.460	0.936	21
	averaged over 199 data sets where Google runs		total no. of classifiers = 28

to the F-measure, precision, and recall (see Table A1 in the Appendix).

In Table 7, we show the Google Prediction API average performance on all feasible data sets.[1] The AUC of Google Prediction API is 0.909, which is inferior to 27 other classifiers. The F-measure is 0.497, which is worse than 17 other classifiers. Precision (0.588) and recall (0.460) perform worse than do 23 and 21 other classifiers, respectively.

Results with Feature Selection

Our next analysis shows results with feature selection applied before training of the classifier. Table 8 summarizes the mean AUC performance over all data sets for various

[1]Google implements size limits on the input data sets to the Prediction API. Thus, we ran the Prediction API for 199 of 229 data sets.

DOI: 10.1002/asi

DOI: 10.1002/asi

TABLE 8. Area under ROC curve for combinations of classifiers and feature selection methods, averaged over all 229 data sets.[a]

	all	gll_g_k1	gll_g_k2	gll_g_k3	gll_z_k1	gll_z_k2	gll_z_k3	svm_rfe1	svm_rfe2	u1_auc	u2_auc	u1_bs	u2_bs	u1_ig	u2_ig	u1_df	u2_df	u1_x2	u2_x2
svm1_libsvm	**0.948**	0.945	0.926	0.913	0.942	0.942	0.934	0.890	0.940	0.853	0.927	0.897	0.947	0.871	0.940	0.841	0.939	0.879	0.939
svm2_libsvm	**0.954**	0.944	0.928	0.909	0.944	0.940	0.934	0.890	0.943	0.860	0.929	0.901	0.947	0.875	0.939	0.843	0.942	0.935	0.937
svm3_libsvm	**0.954**	0.947	0.933	0.928	0.945	0.941	0.936	0.890	0.943	0.891	0.940	0.908	**0.949**	0.897	0.947	0.854	0.945	0.945	0.945
svm1_weight	**0.953**	**0.955**	0.944	0.937	0.948	0.943	0.936	0.897	0.944	0.907	0.943	0.917	**0.952**	0.907	**0.950**	0.834	0.935	0.902	0.947
svm2_weight	**0.957**	**0.955**	0.946	0.939	**0.950**	0.945	0.939	0.897	0.945	0.910	0.947	0.918	**0.953**	0.910	**0.954**	0.857	0.944	0.904	**0.950**
svm3_weight	**0.956**	**0.954**	0.945	0.937	**0.950**	0.945	0.938	0.896	0.945	0.910	0.947	0.917	**0.953**	0.909	**0.953**	0.859	0.945	0.902	**0.949**
svm1_light	**0.953**	0.940	0.919	0.905	0.937	0.936	0.932	0.868	0.936	0.822	0.929	0.873	0.944	0.839	0.932	0.841	0.943	0.839	0.930
svm2_light	**0.954**	0.948	0.930	0.917	0.947	0.942	0.938	0.890	0.943	0.861	0.936	0.901	**0.950**	0.880	0.943	0.844	0.944	0.885	0.943
svm3_light	**0.954**	0.947	0.935	0.926	0.945	0.941	0.935	0.889	0.943	0.890	0.940	0.906	**0.949**	0.896	0.947	0.851	0.944	0.894	0.945
svm4_light	**0.948**	0.945	0.926	0.910	0.942	0.941	0.935	0.888	0.940	0.856	0.930	0.893	0.945	0.874	0.940	0.838	0.939	0.877	0.939
svm1_perf	0.942	0.943	0.937	0.932	0.933	0.932	0.927	0.893	0.937	0.900	0.927	0.916	0.947	0.902	0.937	0.803	0.909	0.899	0.936
svm2_perf	**0.959**	**0.955**	0.944	0.936	**0.949**	0.944	0.937	0.897	0.947	0.907	0.945	0.918	**0.954**	0.907	**0.952**	0.844	0.944	0.902	**0.950**
svm3_perf	0.943	**0.951**	0.942	0.936	0.946	0.941	0.935	0.896	0.941	0.904	0.937	0.915	**0.949**	0.906	0.946	0.833	0.925	0.901	0.942
svm4_perf	**0.951**	**0.952**	0.946	0.938	0.943	0.944	0.938	0.883	0.941	0.892	0.940	0.899	0.946	0.886	0.947	0.850	0.941	0.883	0.942
svm1_trans	0.939	0.934	0.915	0.893	0.932	0.930	0.928	0.863	0.929	0.828	0.925	0.878	0.940	0.843	0.926	0.813	0.927	0.839	0.924
svm2_trans	0.938	0.941	0.926	0.911	0.937	0.933	0.929	0.882	0.936	0.863	0.927	0.898	0.945	0.872	0.936	0.829	0.934	0.872	0.934
svm3_trans	0.936	0.940	0.921	0.906	0.937	0.930	0.926	0.876	0.933	0.876	0.930	0.897	0.943	0.870	0.930	0.839	0.931	0.867	0.929
svm4_trans	0.933	0.941	0.919	0.902	0.940	0.938	0.933	0.880	0.934	0.852	0.924	0.888	0.941	0.859	0.933	0.807	0.926	0.865	0.932
svm1_l2l1	0.923	0.814	0.760	0.740	0.744	0.727	0.727	0.794	0.877	0.653	0.785	0.818	0.891	0.714	0.826	0.765	0.906	0.734	0.826
svm2_l2l1	0.926	0.847	0.801	0.783	0.779	0.766	0.768	0.807	0.888	0.682	0.828	0.835	0.905	0.740	0.856	0.816	0.919	0.761	0.857
svm_l1l2	0.928	0.863	0.820	0.803	0.785	0.778	0.776	0.818	0.888	0.730	0.843	0.856	0.906	0.773	0.859	0.832	0.921	0.774	0.851
lr1	0.932	0.866	0.824	0.809	0.789	0.782	0.781	0.817	0.890	0.739	0.845	0.854	0.909	0.780	0.865	0.836	0.927	0.777	0.856
lr2	0.928	0.860	0.809	0.788	0.796	0.781	0.780	0.815	0.894	0.692	0.838	0.837	0.906	0.746	0.861	0.823	0.922	0.765	0.864
krr	**0.950**	0.942	0.933	0.924	0.933	0.930	0.924	0.886	0.937	0.898	0.936	0.909	0.945	0.895	0.940	0.849	0.938	0.932	0.939
nb	0.916	0.936	0.887	0.855	0.897	0.869	0.847	0.701	0.886	0.673	0.875	0.770	0.919	0.647	0.875	0.858	0.931	0.619	0.834
blr	**0.956**	**0.951**	0.940	0.933	**0.950**	0.946	0.939	0.894	0.944	0.908	0.947	0.913	**0.950**	0.908	**0.952**	0.858	0.946	0.902	**0.949**
adaboostm1	**0.961**	**0.956**	0.945	0.938	**0.949**	0.941	0.935	0.892	0.946	0.910	**0.950**	0.913	**0.952**	0.907	**0.952**	0.877	**0.954**	0.900	**0.949**
Proportion of selected features	100.0%	1.6%	0.4%	0.2%	1.6%	0.7%	0.5%	0.5%	12.6%	4.1%	33.5%	1.5%	24.9%	0.3%	16.0%	5.1%	44.5%	0.8%	19.2%
No. of selected features	7,549	114	31	14	109	49	30	50	873	917	3,570	190	2,222	24	1,384	326	2,923	73	1,827

Note. [a]Cells with dark blue font and highlighting correspond to the top-10% combinations of classifiers and feature selection methods according to AUC; cells with light grey font correspond to the bottom-50% combinations of classifiers and feature selection methods according to AUC. The proportion of features was individually computed in each data set and then averaged over all data sets. [Table can be viewed in color in the online issue, which is available at wileyonlinelibrary.com.]

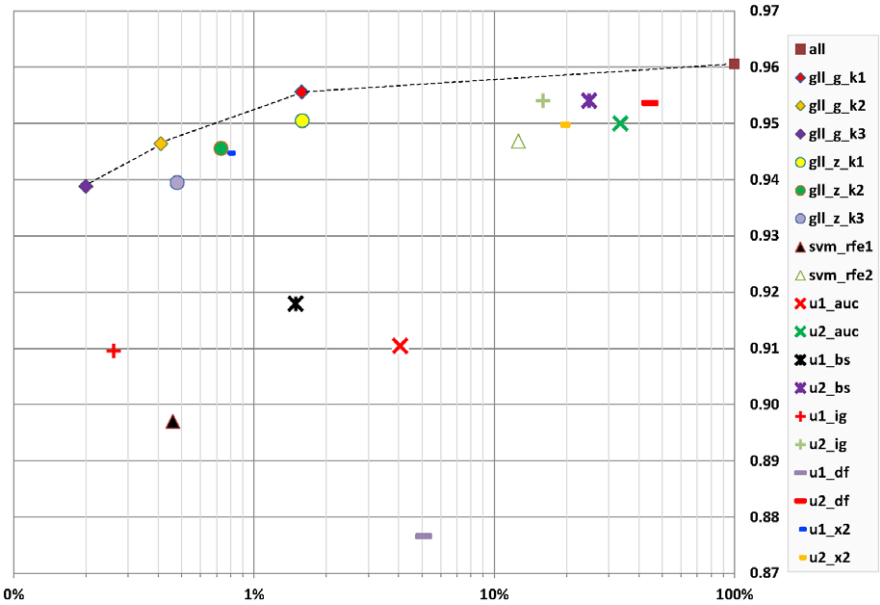

FIG. 5. Area under the curve (AUC) versus proportion of features for various feature selection methods (on average, over all 229 data sets in the study). Reported AUC is a result of the best classifier (i.e., the classifier with the maximum AUC, on average, over all data sets) for each feature selection method (see Table 8 for detailed results). AUC axis is magnified, and a proportion of the feature axis is shown in log scale for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

classifiers and feature selection algorithms. Cells with dark blue font and highlighting correspond to the top-performing 10% classifier/feature selection combinations. Cells with light grey font represent the bottom-performing 50% combinations. Cells with normal font represent combinations performing in 50 to 90% intervals. The bottom of Table 8 shows the mean proportion and absolute number of selected features for each method. The proportion of features was computed individually in each data set and then averaged over all data sets. We can make several general observations based on the results in Table 8:

- Using all features (*all*) consistently produces high and the nominally best AUC performance for the majority of classifiers.
- For naïve Bayes (*nb*) and some variants of SVMs, feature selection improves AUC performance.
- The following classifiers perform in the top 10% for at least 6 of 19 options for feature selection: AdaBoostM1, Bayesian logistic regression (*blr*), linear SVM optimized for AUC (*svm2_perf*), and SVMs with penalty weighting by class prior probabilities (*svm2_weight and svm3_weight*).
- L2-regularized L1-loss SVMs (*svm1-l2l1* and *svm2-l2l1*), regardless of the feature selection method, perform classification in the bottom 50%. In addition to the previously mentioned classifiers, the following classification methods perform in the bottom 50% for the majority of feature selection methods and never achieve classification in the top-10%: naïve Bayes (*nb*), L1/L2 regularized logistic regression (*lr1* and *lr2*), L1-regularized L2-loss SVMs (*svm_l1l2*), and transductive SVMs (*svm1_trans* and *svm4_trans*).
- Among feature selection methods based on univariate association, binormal separation without statistical comparison of the nested feature subsets (*u2_bs*) is the best technique for the majority of classifiers in terms of mean AUC. This method reduces the proportion of features by 75.1%, which is similar to the findings of Forman (2003).
- Univariate feature selection methods with statistical comparison of the nested feature subsets (with names beginning with "*u1*") and SVM-RFE also with statistical comparison (*svm_rfe1*) generally perform classification in the bottom 50%. On the other hand, the aforementioned methods perform more significant reduction of the proportion of features (by 94.9–99.7%) than do similar methods without statistical comparison (by 55.5–87.4%).
- GLL-PC with G^2 test and the *max-k* parameter set to 1 (*gll_g_k1*) reduces by almost two orders of magnitude the input feature space while maintaining near-optimal AUC performance.

For completeness, we show similar results for the F-measure, precision, and recall in Tables A2 to A4 in the Appendix.

In Figure 5, we visualize the results of Table 8 and show the relationship between the proportion of selected features and AUC performance. Each point in the figure denotes a feature selection method and indicates its highest achieved AUC over all classifiers. The dashed line shows a Pareto frontier. Feature selection methods on the Pareto frontier are such that no other method selects smaller feature sets while achieving higher AUC averaged over all data sets. As in Table 8, using all features exhibits the highest AUC performance. GLL-PC algorithms reduce the number of features markedly while maintaining near-optimal AUC performance. We see that with GLL-PC with G^2 test and the *max-k* parameter set to 1 (*gll_g_k1*), depicted by the red diamond,

DOI: 10.1002/asi

we maintain near-optimal AUC performance with only 1.6% of the original number of features (114/7,549 features, on average). Furthermore, using the same algorithm with the *max-k* parameter set to 3 (*gll_g_k3*), depicted by the purple diamond, we may opt for extreme aggressiveness in feature selection to 0.2% of the original number features (14/7,549 features, on average) and lose only 0.02 AUC. The closest other algorithm in feature reduction, a univariate method with information gain and statistical comparison of the nested feature subsets (*u1_ig*), selects 0.3% of the original number of features (24/7,549 features, on average), but we lose more than 0.05 AUC from the empirically best performance.

While Figure 5 shows the feature selection versus classification-performance tradeoff of the best performing classifier, on average, over all data sets, Figure A5 in the Appendix focuses on a specific classifier, AdaBoostM1. The findings are the same as those for Figure 5.

Discussion

Performing text categorization involves a number of steps: creating a document corpus by specifying input features and labels, preprocessing the data, converting documents into a representation suitable for learning, performing feature selection, training a classifier, and evaluating its performance. This benchmark focused only on a critical portion of this pipeline: feature selection and classification. Since the classification performance depends on all steps in the pipeline, we hypothesize that optimizing other steps may further improve classification performance. For example, we represented documents with a simple bag-of-words approach and used tf–idf weighting. More sophisticated text-representation techniques (Hassler & Fliedl, 2006) and other weighting schemes may improve classification performance (Leopold & Kindermann, 2002).

Our benchmark focused on the AUC classification-performance metric and, to the extent possible, optimized all methods for this metric. We chose to emphasize AUC because it is insensitive to unbalanced distribution of classes and does not require a single threshold on the continuous classifier outputs (e.g., distance from the separating hyperplane for SVMs) to convert them to binary classifications. The latter dichotomization removes information about the range of sensitivity-specificity tradeoffs at various classifier output thresholds, which is essential to take into account. In addition to AUC, we briefly reported results for the F-measure, precision, and recall, which are often used in the field of text categorization. While we have performed limited optimization of methods for these metrics (e.g., used cross-validation to perform model selection by evaluating various models with the metric of interest), further optimizations are possible, but were not explored here due to computationally prohibitive running time of experiments. These include, for example, optimizing SVMs for the F-measure, precision, and recall using the method that was used for AUC optimization in this study (Joachims, 2005).

While our work included the ensemble classification method AdaBoostM1, it is natural to ask whether building an ensemble over all tested classifiers yields a significantly improved classification performance. Performing such an experiment following the cross-validation design of our study is extremely computationally expensive because we would need to reserve an independent sample for evaluation of the ensemble classifier. Nevertheless, we can obtain an estimate of the best-case ensemble performance by analyzing the previously derived ROC curves for all classifiers and computing their Pareto frontier ROC curve and the AUC (Lévesque, Durand, Gagné, & Sabourin, 2012). Using this method, we obtained an estimate for ensembling over the tested classifiers of AUC 0.986 while the best individual classifier for each data set achieved an AUC of 0.974, and AdaBoostM1 achieved an AUC of 0.961, on average, over all 229 data sets. Therefore, there is no significant benefit in using ensemble methods to improve classification performance, on average, over all data sets. However, in a small number of data sets (37 of 229 data sets), there is potential of improving statistically significantly classification performance by ensembling compared to the best individual classifier for each data set (see Figure A6 in the Appendix).

In addition to Google Prediction API, which was relatively straightforward to use for the design of our study (and, as a result, it was applied to the majority of data sets), we also have performed limited experiments with two more commercial offerings: Oracle Data Mining (http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html), and IBM SPSS Modeler (http://www-01.ibm.com/software/analytics/spss/products/modeler/). Because these commercial solutions cannot handle many of the data sets in our collection or have other technical issues that prevent a complete, side-to-side comparison with the other methods, we treat these experiments as supplementary and present the corresponding results in the Appendix.

While there are several benchmarks of feature selection and classification methods for text categorization (Forman, 2003; Garnes, 2009; Genkin et al., 2007; Yang & Pedersen, 1997), the current work has contributed to prior research in the following ways:

- The literature on text categorization comparative performance has focused on a *small number of classifiers*. For example, Forman (2003) limited his benchmark to one classifier: a linear SVM. Yang and Pedersen (1997) used k-nearest neighbors and linear least squares fit mapping. Genkin et al. (2007) used lasso logistic regression along with ridge logistic regression and SVMs. Garnes (2009) used naïve Bayes (*nb*) and SVMs. Classification performance of text categorization depends on a classifier and a feature selection method and their interaction. We designed our benchmark to include a large number of classification methods and apply them in combination with a large number of feature selection methods.
- Prior benchmarks used only *univariate feature selection methods*. This benchmark includes state-of-the-art, multivariate feature selection methods that were not available or used previously.

DOI: 10.1002/asi

- Previous benchmarks performed limited versions of feature selection. For example, Forman (2003) evaluated nested subsets of the top-1,000 ranked features (instead of selecting a single feature subset). Since the number of relevant features is an unknown parameter that can significantly affect performance and changes from data set to data set, we used feature selection methods and model selection such that the optimal number of features was determined automatically.
- Previous benchmarks by Forman (2003), Genkin et al. (2007), and Garnes (2009) used only performance metrics that relied on a single threshold to dichotomize continuous outputs of the classifier into binary class labels (e.g., recall, precision, and F-measure). Using a single threshold is not informative because it does not address all the possible trade-offs between sensitivity and specificity, which is why we used an AUC that considers various thresholds for dichotomization of classifier outputs.

A major aim of this work was to present empirical results to facilitate development of best methodological practices for text categorization. We believe that rigorous benchmarking studies are the best way to perform large-scale evaluations of methods.

Conclusion

This work described is, to our knowledge, the largest benchmarking study of supervised classification and feature selection methods for text categorization. Our key findings are the following:

- A small number of classification methods outperforms others, on average, over all data sets. AdaBoostM1, followed by the linear SVM optimized for AUC, SVMs with penalty weighting by class prior probabilities, and Bayesian logistic regression have the best average classification performance measured by AUC.
- Naïve Bayes, L2-regularized L1-loss SVMs, L1-regularized L2-loss SVMs, L1/L2 regularized logistic regression, and transductive SVMs perform worse than do other methods with respect to AUC, on average, over all data sets.
- A commercial offering for text categorization, Google Prediction API, has inferior average classification performance compared to most well-established machine learning methods. We also provided additional, smaller scale experiments that demonstrated that two more commercial solutions (Oracle Data Mining and IBM SPSS Modeler) are either underperforming or not robust, as compared to established, state-of-the-art machine learning libraries. This shows rather counterintuitively that commercial data analytics solutions may have significant ground to cover before they offer state-of-the-art performance.
- Even though some classification methods outperform others, on average, there is no single classifier that has uniformly optimal or near-optimal performance in all tested data sets. The best classifier, AdaBoostM1, has near-optimal AUC performance in only 70% of data sets. This suggests that the choice of classification method in the text categorization pipeline has to be optimized for the specific task/data set of interest.
- Causal graph-based feature selection by Generalized Local Learning allows reducing dimensionality of the input feature space by almost two orders of magnitude, without compromising classification performance. Other feature selection methods either have much worse classification performance or select significantly more features. These results are consistent with recent results in other domains (Aliferis, Statnikov, Tsamardinos, Mani, & Koutsoukos, 2010a, 2010b).

The findings of this study should be of interest both to practitioners and method developers in the field and, hopefully, will be of use in establishing best methodological practices for text categorization.

Acknowledgments

We thank George Forman for making available data sets from his study (2003). The first three authors contributed equally to this article.

References

Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X.D. (2010a). Local causal and Markov Blanket Induction for causal discovery and feature selection for classification. Part I: Algorithms and empirical evaluation. Journal of Machine Learning Research, 11, 171–234.

Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X.D. (2010b). Local causal and Markov Blanket Induction for causal discovery and feature selection for classification. Part II: Analysis and extensions. Journal of Machine Learning Research, 11, 235–284.

Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., & Spyropoulos, C.D. (2000). An experimental comparison of naïve Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Proceedings of the 23rd annual International Association for Computing Machinery Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval (pp 160–167). New York, NY: Association for Computing Machinery (ACM).

Aphinyanaphongs, Y., & Aliferis, C.F. (2006). Prospective validation of text categorization models for identifying high-quality content-specific articles in MEDLINE. In Proceedings of the Annual American Medical Informatics Association (AMIA) 2006 Symposium (pp. 6–10). Bethesda, MD: American Medical Informatics Association (AMIA).

Aphinyanaphongs, Y., Tsamardinos, I., Statnikov, A., Hardin, D., & Aliferis, C.F. (2005). Text categorization models for high-quality article retrieval in internal medicine. Journal of the American Medical Informatics Association, 12(2), 207–216.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society: Series B (Methodological), 57(1), 289–300.

Benjamini, Y., & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. Annals of Statistics, 29(4), 1165–1188.

Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Jackel, L.D., . . . Simard, P. (1994). Comparison of classifier methods: A case study in handwritten digit recognition. In Proceedings of the 12th International Association of Pattern Recognition International Conference on Pattern Recognition, Vol. 2. Conference B: Computer Vision & Image Processing (pp. 77–82). Washington, DC: IEEE Computer Society Press.

Bovelstad, H.M., Nygard, S., Storvold, H.L., Aldrin, M., Borgan, O., Frigessi, A., & Lingjaerde, O.C. (2007). Predicting survival from microarray data—A comparative study. Bioinformatics, 23(16), 2080–2087.

Braga-Neto, U.M., & Dougherty, E.R. (2004). Is cross-validation valid for small-sample microarray classification? Bioinformatics, 20(3), 374–380.

Chang, C.C., & Lin, C.J. (2011). LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(3), 27.

DOI: 10.1002/asi

Croft, W.B., Metzler, D., & Strohman, T. (2010). Search engines: Information retrieval in practice. Boston, MA: Addison-Wesley.

Dudoit, S., Fridlyand, J., & Speed, T.P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association, 97(457), 77–88.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., & Lin, C.J. (2008). LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9, 1871–1874.

Fan, R.E., Chen, P.H., & Lin, C.J. (2005). Working set selection using second order information for training support vector machines. Journal of Machine Learning Research, 6(1889), 1918.

Fawcett, T. (2003). ROC Graphs: Notes and practical considerations for researchers (Report No. HPL-2003-4). Palo Alto, CA: HP Laboratories.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research, 3, 1289–1305.

Freund, Y., & Schapire, R.E. (1996, July). Experiments with a new boosting algorithm. Paper presented at the International Conference on Machine Learning (ICML), Bari, Italy.

Garnes, Ø.L. (2009). Feature selection for text categorisation (Unpublished master's thesis). Norwegian University of Science and Technology, Department of Computer Science, Trondheim, Norway.

Genkin, A., Lewis, D.D., & Madigan, D. (2004). Large-scale Bayesian logistic regression for text categorization. Piscataway, NJ: Rutgers University, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS).

Genkin, A., Lewis, D.D., & Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. Technometrics, 49(3), 291–304.

Guyon, I. (2005). Kernel ridge regression tutorial. Retrieved from http://clopinet.com/isabelle/Projects/ETH/KernelRidge.pdf

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3(1), 1157–1182.

Guyon, I., Li, J., Mader, T., Pletscher, P.A., Schneider, G., & Uhr, M. (2006). Feature selection with the CLOP package. Retrieved from http://clopinet.com/isabelle/Projects/ETH/TM-fextract-class.pdf

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. Machine Learning, 46(1), 389–422.

Harper, P.R. (2005). A review and comparison of classification algorithms for medical decision making. Health Policy, 71(3), 315–331. doi:10.1016/j.healthpol.2004.05.002

Hassler, G.F.M., & Fliedl, G. (2006). Text preparation through extended tokenization. Data Mining VII: Data, Text and Web Mining and their Business Applications, 37, 13–21.

Hastie, T., Tibshirani, R., & Friedman, J.H. (2001). The elements of statistical learning: Data mining, inference, and prediction. New York, NY: Springer.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 200–209). San Francisco, CA: Morgan Kaufmann.

Joachims, T. (2002). Learning to classify text using support vector machines. Boston, MA: Kluwer Academic.

Joachims, T. (2005). A support vector method for multivariate performance measures. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 377–384). San Francisco, CA: Morgan Kaufmann.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence, 2, 1137–1145.

LeCun, Y., Jackel, L.D., Bottou, L., Brunot, A., Cortes, C., Denker, J.S., . . . Sackinger, E. (1995, October). Comparison of learning algorithms for handwritten digit recognition. Paper presented at the International Conference on Artificial Neural Networks, Cambridge, UK.

Leopold, E., & Kindermann, J. (2002). Text categorization with support vector machines. How to represent texts in input space? Machine Learning, 46(1), 423–444.

Lévesque, J.-C., Durand, A., Gagné, C., & Sabourin, R. (2012). Multi-objective evolutionary optimization for generating ensembles of classifiers in the ROC space. In Proceedings of the 14th International Conference on Genetic and Evolutionary Computation (pp. 879–886). New York, NY: ACM.

Li, T., Ogihara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. In Proceedings of the 26th annual International Association for Computing Machinery Special Interest Group on Information Retrieval Conference on Research and Development in Informaion Retrieval (pp. 282–289). New York, NY: ACM.

Lin, C.J., Weng, R.C., & Keerthi, S.S. (2008). Trust region Newton method for logistic regression. Journal of Machine Learning Research, 9, 627–650.

Ling, C.X., Huang, J., & Zhang, H. (2003a). AUC: a better measure than accuracy in comparing learning algorithms. In Advances in Artificial Intelligence (pp. 329–341). Springer Berlin Heidelberg.

Ling, C.X., Huang, J., & Zhang, H. (2003b). AUC: A statistically consistent and more discriminating measure than accuracy. Proceedings of the 18th International Joint Conference of Artificial Intelligence (IJCAI) (pp. 519–524). San Francisco, CA: Morgan Kaufmann.

Manning, C.D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. New York, NY: Cambridge University Press.

Melville, P., Gryc, W., & Lawrence, R.D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification. In Proceedings of the 15th Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining International Conference on Knowledge Discovery and Data Mining (pp. 1275–1284). New York, NY: ACM.

Menke, J., & Martinez, T.R. (2004). Using permutations instead of student's t distribution for p-values in paired-difference algorithm comparisons. Proceedings of Institute of Electrical and Electronics Engineers International Joint Conference on Neural Networks, 2, 1331–1335.

Mitchell, T. (1997). Machine learning. New York, NY: McGraw-Hill.

Pakhomov, S., Weston, S.A., Jacobsen, S.J., Chute, C.G., Meverden, R., & Roger, V.L. (2007). Electronic medical records for clinical research: Application to the identification of heart failure. American Journal of Managed Care, 13(6, Pt. 1), 281–288.

Porter, M.F. (1980). An algorithm for suffix stripping. Program: Electronic Library and Information Systems, 14(3), 130–137.

Scheffer, T. (1999). Error estimation and model selection (Unpublished doctoral thesis). Technischen Universitãt Berlin, School of Computer Science.

Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., & Levy, S. (2005). A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics, 21(5), 631–643.

Statnikov, A., Henaff, M., Narendra, V., Konganti, K., Li, Z., Yang, L., . . . Alekseyenko, A.V. (2013). A comprehensive evaluation of multicategory classification methods for microbiomic data. Microbiome, 1(1), 11.

Statnikov, A., Tsamardinos, I., Dosbayev, Y., & Aliferis, C.F. (2005). GEMS: A system for automated cancer diagnosis and biomarker discovery from microarray gene expression data. International Journal of Medical Informatics, 74(7–8), 491–503.

Statnikov, A., Wang, L., & Aliferis, C.F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics, 9, 319.

Vapnik, V.N. (1998). Statistical learning theory. New York, NY: Wiley.

Wu, B., Abbott, T., Fishman, D., McMurray, W., Mor, G., Stone, K., . . . Zhao, H. (2003). Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. Bioinformatics, 19(13), 1636–1643.

Yang, Y., & Pedersen, J.O. (1997). A comparative study on feature selection in text categorization. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 412–420). San Francisco, CA: Morgan Kaufmann.

Zhuang, J., Widschwendter, M., & Teschendorff, A.E. (2012). A comparison of feature selection and classification methods in DNA methylation studies using the Illumina Infinium platform. BMC Bioinformatics, 13, 59. doi:10.1186/1471-2105-13-59

Zweigenbaum, P., Demner-Fushman, D., Yu, H., & Cohen, K.B. (2007). Frontiers of biomedical text mining: Current progress. Briefings in Bioinformatics, 8(5), 358–375. doi:10.1093/bib/bbm045

Appendix

Basic Principles of Supervised Classification Methods Used in the Study

Standard SVMs. The underlying idea of support vector machine (SVM) classifiers is to calculate a maximal margin (i.e., gap) hyperplane separating two classes of the data. To learn nonlinearly separable functions, the data are implicitly mapped to a higher dimensional space by means of a kernel function, where a separating hyperplane is found. New samples are classified according to the side of the hyperplane to which they belong.

SVMs with penalty weighting. Weighted SVMs is a variant of the standard SVMs. This method assigns different weights to different samples, such that the training algorithm learns the decision surface according to the relative importance of samples. This is especially useful when the data contain extremely unbalanced classes. In such situations, one may prefer to assign higher weights to samples from a rarer class (i.e., class with fewer samples).

SVMs for optimizing multivariate performance measures. This method is a variant of the standard SVMs, except it uses a different loss function, designed to optimize a performance metric of interest.

Transductive SVMs. In addition to employing the learning mechanism of standard SVMs, transductive SVMs seek to utilize unlabeled testing data during the training process. Compared with standard SVMs, this is a much harder learning problem because training transductive SVMs requires solving the (partly) combinatorial optimization problem of assigning testing data labels. An approximate solution can be found using a form of local search: beginning with a labeling of the testing data based on classification of an inductive SVM, and then improving the solution by switching the labels of testing samples so that the objective function is optimized.

L2-regularized L1-loss and L1-regularized L2-Loss SVMs. These are variants of the standard SVMs that use a

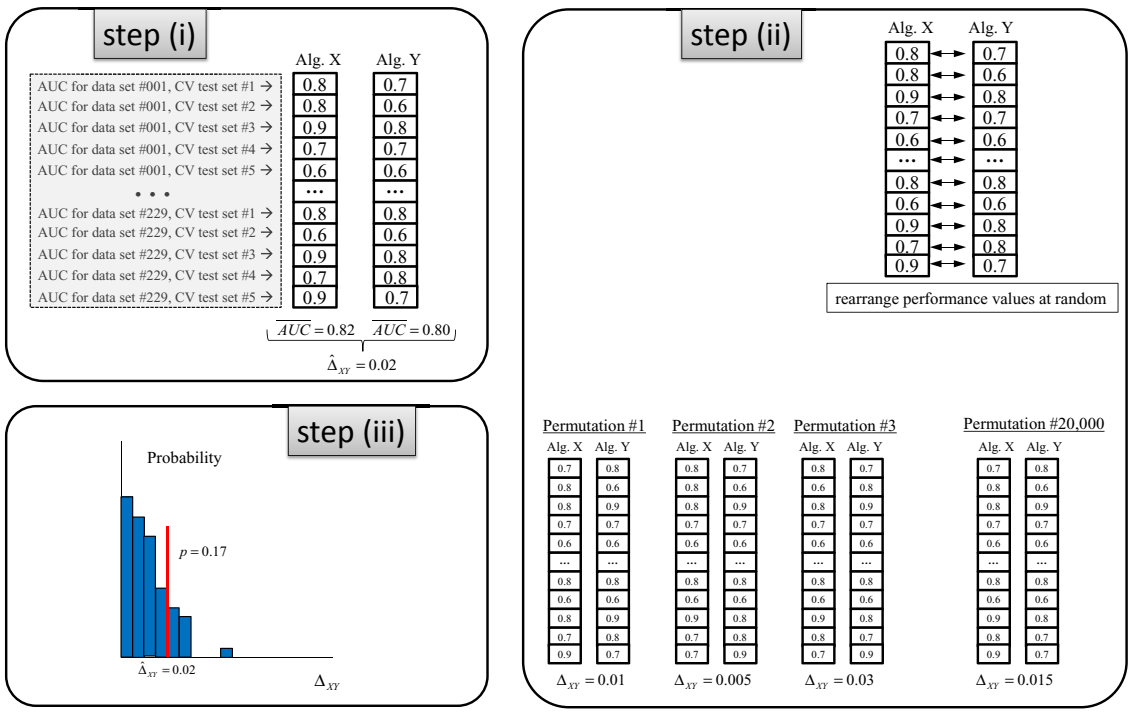

FIG. A1. Illustration of the statistical comparison procedure (steps of the procedure are described in the article.) [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

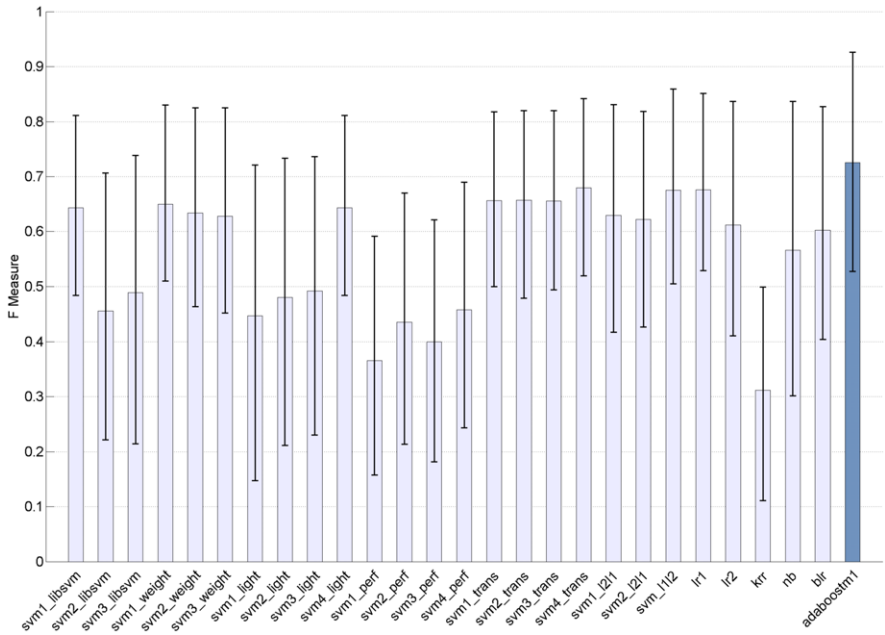

FIG. A2. Mean and 20th to 80th percentile interval of F-measure computed over all 229 data sets in the study. There are no classifiers that are statistically indistinguishable from the best performing one (adaboostm1), shown with the dark bar. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

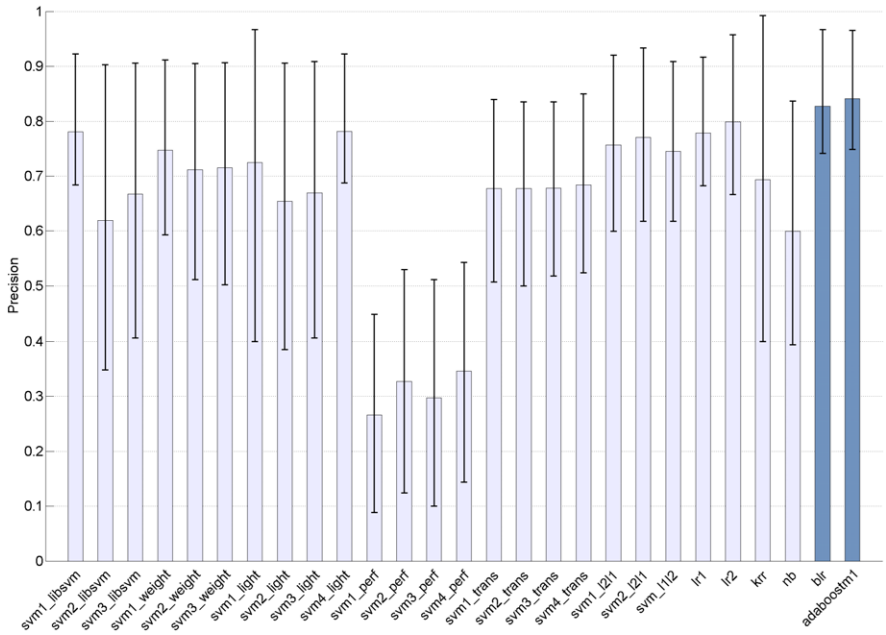

FIG. A3. Mean and 20th to 80th percentile interval of precision computed over all 229 data sets in the study. Dark bars correspond to classifiers that are statistically indistinguishable from the best performing one (adaboostm1); light bars correspond to all other classifiers. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

different training algorithm and use different regularization/penalty and/or loss functions. These methodologies train by a greedy strategy (coordinate descent method implemented in the LIBLINEAR program (Fan, Chang, Hsieh, Wang, & Lin, 2008)), which is faster as compared to the training of standard SVMs, especially in high-dimensional data sets. Standard SVMs have L2-regularization that favors solutions with relatively small coefficients, but does not discard any features (which is equivalent to applying Gaussian prior on the coefficients). L1-regularization more uniformly shrinks the weights

DOI: 10.1002/asi

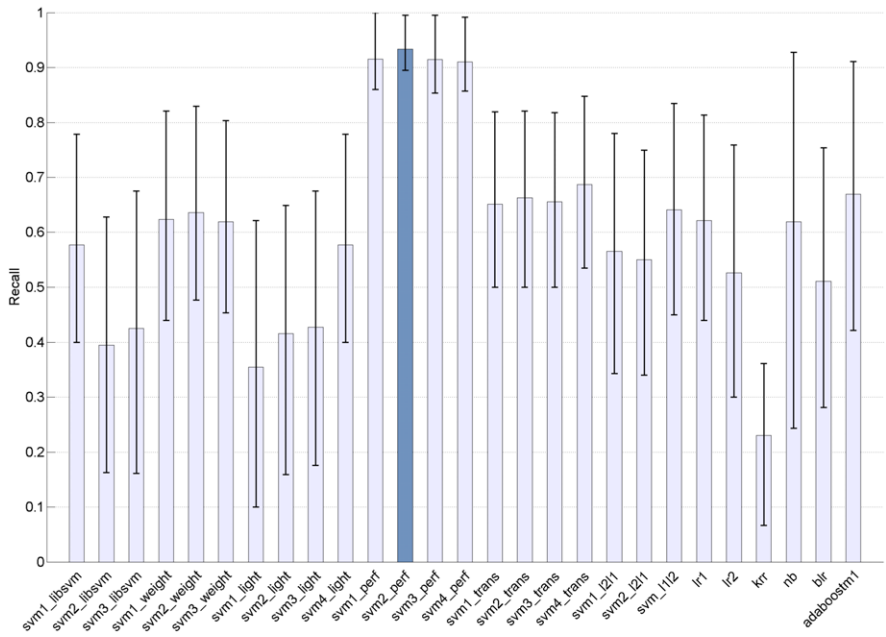

FIG. A4. Mean and 20th to 80th percentile interval of recall computed over all 229 data sets in the study. There are no classifiers that are statistically indistinguishable from the best performing one (svm2_perf), shown with the dark bar. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

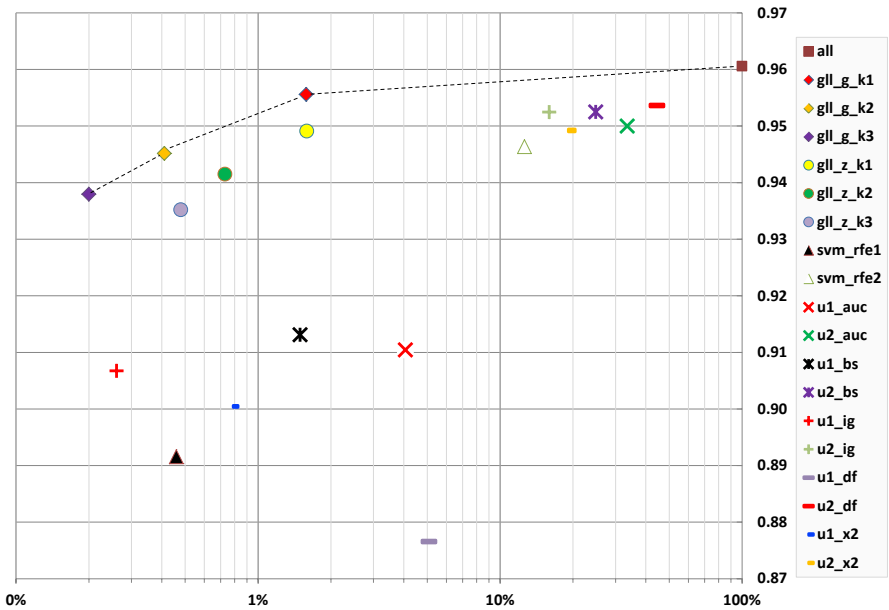

FIG. A5. Area under the receiver operating characteristic (ROC) curve (AUC) of adaboostm1 versus proportion of features for various feature selection methods (on average, over all 229 data sets in the study). The AUC axis is magnified, and a proportion of the feature axis is shown in log scale for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

of features and can set weights to zero, effectively performing embedded feature selection (which is equivalent to applying Laplace prior on the coefficients). While standard SVMs use L1-loss (hinge loss), L2-loss (squared hinge-loss) also is used in the field.

L1-regularized and L2-regularized logistic regression. These are implementations of logistic regression with different regularization/penalty functions. The implications of L1 and L2 regularization on solutions are described in the previous paragraph. These methods use training by a greedy

DOI: 10.1002/asi

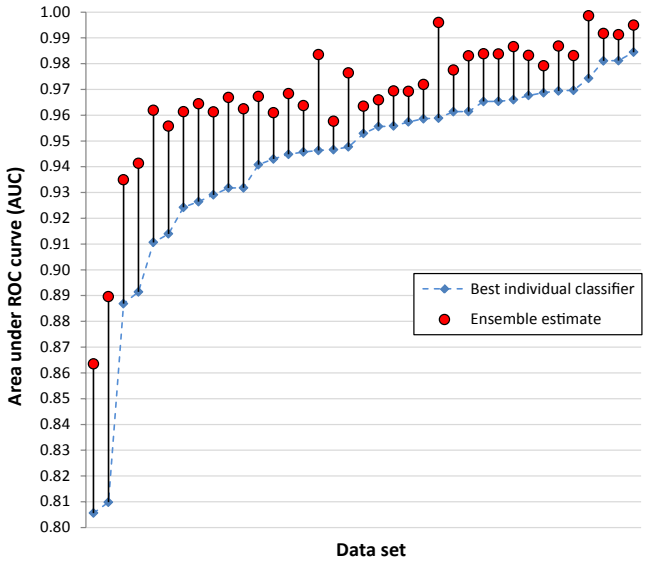

FIG. A6. Area under the receiver operating characteristic (ROC) curve (AUC) of the best performing individual classifier for each data set and AUC estimate of the ensemble method. The results are shown for only 37 of 229 data sets, where ensembling leads to a statistically significant improvement of AUC > 0.01 compared to the best individual classifier. Data sets are sorted in ascending order based on performance of the best individual classifier. The AUC axis is magnified for ease of visualization. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

strategy (coordinate descent method implemented in the LIBLINEAR program [Fan et al., 2008]), which is suitable for high-dimensional data.

Kernel ridge regression. Kernel ridge regression (KRR) adds kernel trick to the ridge regression. Ridge regression is multiple linear regression with regularization by L2 penalty. KRR and standard SVMs are similar in dealing with nonlinearity (by kernel trick) and model regularization (by L2 penalty); the difference lies in the loss function. Standard SVMs use a hinge loss function while KRR uses a mean squared error loss function.

Naïve bayes. This classifier works by direct application of the Bayesian theorem. It assumes that the probability distribution of a feature is independent of another feature, given class labels.

Bayesian logistic regression. This implementation of logistic regression is very similar to the L1-regularized logistic regression described earlier. The key difference lies in implementation of the training algorithm (cyclic coordinate descent optimization).

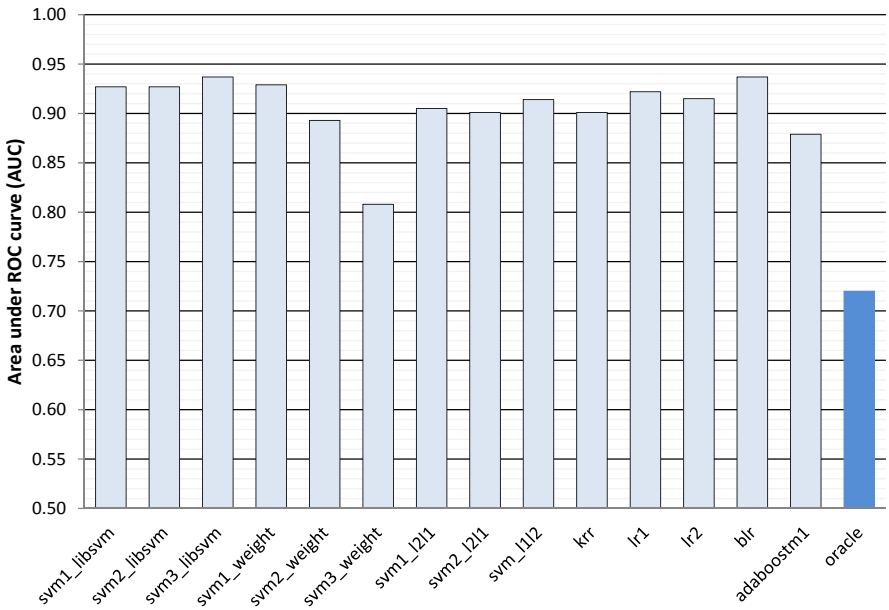

FIG. A7. Area under the receiver operating characteristic (ROC) curve (AUC) of several classification methods trained on the *trnew* data set (TREC 2010; Topic 200) with 8,245 documents and validated in the independent set of 685,592 documents. Since the main purpose here was comparison with Oracle of the best performing classification methods, we considered only a subset of classifiers. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

DOI: 10.1002/asi

AdaBoostM1. This is an ensemble learning algorithm that uses the idea of adaptive boosting. It learns by iteratively combining a large number of "weak" learners (in our case, decision stumps). During each round of training, a new weak learner is added to the ensemble, and a weighting vector is adjusted to focus on data samples that were misclassified in previous rounds.

Basic Principles of Feature Selection Methods Used in the Study

Generalized Local Learning (GLL-PC). This feature selection method is based on a theoretical framework that ties together feature selection and causal graphical models and provides theoretical guarantees for the optimality of the selected feature set (i.e., minimal predictor number and maximal predictivity). GLL-PC outputs features in the Markov boundary of the response, which is the minimal set of features, conditioned on which the response is rendered statistically independent of the remaining features in the data set. Under certain assumptions about the learner and the loss function, Markov boundary is the solution to the features selection problem; that is, it is the minimal set of features with optimal predictive performance for the current data set and response. Furthermore, in faithful distributions, GLL-PC identifies exactly the local causal neighborhood of the response which consists of all its direct causes and direct effects. The parameter *max-k* denotes maximum size of the conditioning set to establish that the response is statistically independent of other features.

Support vector machine-based recursive feature elimination (SVM-RFE). This algorithm involves iteratively fitting linear SVM classification models (in the training data) by (a) discarding 20% of features at each iteration with the small impact on classification and (b) selecting the subset of features that participate in the best performing classification model (as typically assessed in the validation data). Two variants of SVM-RFE have been utilized: one that outputs the subset of features in the nominally best performing classification model (denoted as "without statistical comparison" in Table 5) and another one that outputs the subset of features in a classification model with the smallest number of features that has performance statistically indistinguishable from that of the nominally best performing one (denoted as "with statistical comparison" in Table 5). The latter variant typically results in smaller feature sets. To perform statistical comparison of performance estimates (area under curve, or AUC) in SVM-RFE, we used DeLong, DeLong, and Clarke-Pearson's (1988) test at $\alpha = 0.05$.

Backward wrapping based on univariate association of features. This procedure first ranks all features according to univariate association with the response, and then performs backward elimination using linear SVM classifier (fit on the training set and evaluated on the validation set) to determine the best performing, smallest nested subset of features. The nested subsets of features are defined by removing 20% of the lowest ranking features at each iteration. Similarly to SVM-RFE, this method can either return the smallest nested subset with the nominally best performance (denoted as "without statistical comparison" in Table 5) or the smallest nested subset with performance statistically indistinguishable from that of the nominally best one according to Delong et al.'s (1988) test at $\alpha = 0.05$ (denoted as "with statistical comparison" in Table 5).

Exploring Other Commercial Offerings for Text Categorization

We used Version 11g of Oracle Data Mining (ODM) (for Oracle Version 11.1.07), which implements a number of classifiers and feature selection methods to analyze data within Oracle databases (Milenova, Yarmus, & Campos, 2005). A detailed list of methods implemented in ODM is given in http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/toc.htm Use of ODM in the design of our study requires setting up multiple Oracle databases and loading our data sets one by one as well as programming cross-validation functionality. This is a technically challenging task in the setting of Oracle and cannot be easily automated; that is why we performed an experiment with a single data set *trnew* (TREC 2010, Topic 200; see Table 2; and Cormack, Grossman, Hedin, & Oard, 2010), for which we have obtained a large independent validation data set with 685,592 documents. An SVM model was trained with ODM on the entire *trnew* data set with 8,245 documents using the default parameters and applied to the independent validation data set. The AUC was 0.720, which is worse than the performance of all other classifiers tested in this data set (see Figure A7 in the Appendix). The poor relative performance of SVM ODM, especially as compared to the standard SVM implementations, can be in part explained by ODM's model-selection procedure. Rather than exhaustively evaluating all parameters by cross-validation and selecting an empirically best one (which was the case for other SVM implementations), ODM performs a more computationally inexpensive, but highly heuristic, selection of parameters (Cormack et al., 2010).

We used Version 14.1 of IBM SPSS Modeler, which is a software for data mining and predictive modeling. It implements many machine learning algorithms, such as Bayesian networks, Classification and Regression Trees (CART), linear regression, logistic regression, SVMs, and factor analysis. In addition, the software provides capability for auto-modeling; that is, automatically selecting an algorithm and parameterizations suitable for the task at hand. We have randomly selected 10 data sets and applied IBM SPSS Modeler to them using either graphics user

DOI: 10.1002/asi

interface or API, and using either auto-modeler or SVM classifiers. IBM SPSS Modeler was not able to complete analysis within 5 days of single central processing unit (CPU) time in either of these 40 experimental setups. On the other hand, libSVM implementation of SVM classifiers used in our study yielded results within <5 min for each of the same data sets on the same CPU. As a verification test, we also applied IBM SPSS Modeler to a few smaller data sets with 10 to 200 features, and the software successfully completed the analysis within a few minutes for each data set. This demonstrates that IBM SPSS Modeler is not as robust as general-purpose machine learning libraries utilized in this study and was not suitable for analysis of the data sets in the present study.

TABLE A1. Number of data sets when a classification method achieved F-measure, precision, and recall within 0.01 of the nominally best performing method (i.e., it is "near-optimal").

Method	*n* data sets for top F-measure (over all 229)	*n* data sets for top precision (over all 229)	*n* data sets for top recall (over all 229)
svm1_libsvm	1	10	2
svm2_libsvm	1	16	0
svm3_libsvm	0	13	1
svm1_weight	11	9	2
svm2_weight	13	7	2
svm3_weight	12	4	3
svm1_light	1	41	2
svm2_light	0	12	1
svm3_light	0	14	1
svm4_light	1	10	2
svm1_perf	1	0	143
svm2_perf	6	1	131
svm3_perf	3	0	105
svm4_perf	1	0	95
svm1_trans	32	5	1
svm2_trans	26	4	2
svm3_trans	22	2	1
svm4_trans	21	3	1
svm1_l2l1	7	10	2
svm2_l2l1	9	15	0
svm_l1l2	24	12	1
lr1	24	16	1
lr2	8	51	0
krr	0	77	2
nb	17	8	12
blr	7	57	2
adaboostm1	122	76	19

DOI: 10.1002/asi

TABLE A2. F-measure for combinations of classifiers and feature selection methods, averaged over all 229 data sets.[a]

	all	gll_g_k1	gll_g_k2	gll_g_k3	gll_z_k1	gll_z_k2	gll_z_k3	svm_rfe1	svm_rfe2	u1_auc	u2_auc	u1_bs	u2_bs	u1_ig	u2_ig	u1_df	u2_df	u1_x2	u2_x2
svm1_libsvm	**0.643**	0.595	0.527	0.494	0.604	0.597	0.586	0.508	0.618	0.385	0.565	0.449	0.589	0.420	0.580	0.302	0.582	0.456	0.590
svm2_libsvm	0.456	0.430	0.421	0.387	0.423	0.428	0.424	0.263	0.401	0.209	0.490	0.241	0.499	0.235	0.513	0.270	0.489	0.498	0.502
svm3_libsvm	0.489	0.523	0.508	0.486	0.472	0.499	0.498	0.345	0.484	0.260	0.450	0.290	0.516	0.295	0.498	0.303	0.473	0.471	0.475
svm1_weight	**0.650**	0.629	0.572	0.551	0.632	0.611	0.597	0.545	0.632	0.467	0.597	0.511	0.632	0.488	0.614	0.381	0.600	0.505	0.621
svm2_weight	0.634	0.572	0.505	0.465	0.571	0.560	0.547	0.363	0.557	0.336	0.580	0.309	0.551	0.332	0.575	0.417	0.631	0.347	0.558
svm3_weight	0.628	0.603	0.552	0.525	0.587	0.591	0.586	0.394	0.566	0.363	0.584	0.347	0.573	0.357	0.592	0.434	0.628	0.373	0.572
svm1_light	0.447	**0.669**	**0.658**	**0.646**	0.633	0.640	0.638	0.565	0.613	0.454	0.564	0.516	0.573	0.499	0.597	0.244	0.418	0.531	0.587
svm2_light	0.480	0.468	0.431	0.396	0.432	0.462	0.467	0.307	0.459	0.194	0.422	0.209	0.478	0.220	0.452	0.268	0.468	0.236	0.441
svm3_light	0.492	0.517	0.490	0.473	0.473	0.497	0.496	0.338	0.475	0.261	0.450	0.253	0.508	0.289	0.497	0.302	0.475	0.293	0.477
svm4_light	**0.644**	0.596	0.524	0.485	0.604	0.596	0.585	0.494	0.612	0.382	0.562	0.446	0.586	0.414	0.577	0.301	0.582	0.443	0.587
svm1_perf	0.365	0.361	0.446	0.488	0.281	0.346	0.384	0.558	0.414	0.457	0.308	0.587	0.503	0.531	0.370	0.247	0.327	0.553	0.393
svm2_perf	0.435	0.409	0.466	0.498	0.315	0.364	0.396	0.569	0.463	0.465	0.347	0.596	0.550	0.540	0.418	0.300	0.414	0.561	0.435
svm3_perf	0.399	0.410	0.472	0.505	0.317	0.369	0.405	0.572	0.454	0.465	0.339	0.599	0.543	0.542	0.410	0.294	0.383	0.562	0.422
svm4_perf	0.458	0.461	0.493	0.516	0.357	0.394	0.417	0.571	0.488	0.463	0.382	0.589	0.569	0.535	0.454	0.335	0.444	0.555	0.465
svm1_trans	**0.656**	**0.683**	0.639	0.607	0.612	0.588	0.581	0.493	**0.643**	0.430	0.641	0.488	**0.664**	0.439	0.642	0.383	0.629	0.442	0.600
svm2_trans	**0.657**	0.543	0.442	0.397	0.521	0.465	0.445	0.319	0.533	0.248	0.538	0.288	0.546	0.245	0.526	0.359	0.637	0.248	0.497
svm3_trans	**0.655**	0.636	0.546	0.492	0.576	0.531	0.506	0.340	0.568	0.292	0.583	0.343	0.610	0.288	0.573	0.435	**0.651**	0.279	0.531
svm4_trans	**0.680**	**0.649**	0.550	0.494	0.615	0.576	0.553	0.451	0.620	0.396	0.610	0.449	0.631	0.403	0.606	0.383	**0.645**	0.400	0.583
svm1_l2l1	0.629	0.614	0.600	0.589	0.523	0.540	0.543	0.608	0.620	0.506	0.551	0.595	0.641	0.561	0.585	0.309	0.569	0.585	0.587
svm2_l2l1	0.622	**0.652**	0.629	0.618	0.548	0.567	0.566	0.615	0.625	0.521	0.580	0.611	**0.659**	0.575	0.610	0.391	0.603	0.596	0.597
svm_l1l2	**0.675**	**0.665**	0.631	0.615	0.548	0.558	0.565	0.609	**0.659**	0.519	0.614	0.619	**0.679**	0.564	0.638	0.432	**0.656**	0.575	0.630
lr1	**0.676**	**0.665**	0.631	0.610	0.547	0.556	0.561	0.609	**0.660**	0.505	0.614	0.615	**0.679**	0.556	**0.642**	0.439	**0.661**	0.566	0.631
lr2	0.612	**0.655**	0.628	0.616	0.539	0.556	0.559	0.618	0.628	0.525	0.580	0.611	**0.654**	0.578	0.611	0.398	0.605	0.596	0.598
krr	0.312	0.277	0.305	0.306	0.290	0.330	0.347	0.301	0.306	0.276	0.312	0.224	0.227	0.286	0.299	0.162	0.296	0.307	0.308
nb	0.566	**0.666**	0.549	0.459	0.597	0.519	0.464	0.247	0.574	0.176	0.504	0.360	0.623	0.185	0.533	0.397	0.575	0.174	0.480
blr	0.602	**0.685**	**0.670**	**0.656**	0.631	0.640	0.641	0.630	**0.656**	0.547	0.621	0.601	**0.666**	0.585	**0.648**	0.368	0.591	0.596	0.639
adaboostm1	**0.725**	**0.706**	**0.693**	**0.684**	**0.670**	**0.659**	**0.651**	0.622	**0.694**	0.598	**0.689**	0.614	**0.697**	0.624	**0.694**	0.460	**0.699**	0.621	**0.692**
Proportion of selected features	100.0%	1.6%	0.4%	0.2%	1.6%	0.7%	0.5%	0.5%	12.6%	4.1%	33.5%	1.5%	24.9%	0.3%	16.0%	5.1%	44.5%	0.8%	19.2%
No. of selected features	7,549	114	31	14	109	49	30	50	873	917	3,570	190	2,222	24	1,384	326	2,923	73	1,827

Note. [a]Cells with dark blue font and highlighting correspond to the top 10% combinations of classifiers and feature selection methods according to F-measure; cells with light grey font correspond to the bottom 50% combinations of classifiers and feature selection methods according to F-measure. The proportion of features was individually computed in each data set and then averaged over all data sets. [Table can be viewed in color in the online issue, which is available at wileyonlinelibrary.com.]

DOI: 10.1002/asi

TABLE A3. Precision for combinations of classifiers and feature selection methods, averaged over all 229 data sets.[a]

	all	gll_g_k1	gll_g_k2	gll_g_k3	gll_z_k1	gll_z_k2	gll_z_k3	svm_rfe1	svm_rfe2	u1_auc	u2_auc	u1_bs	u2_bs	u1_ig	u2_ig	u1_df	u2_df	u1_x2	u2_x2
svm1_libsvm	**0.781**	**0.785**	0.750	0.734	0.754	**0.764**	**0.763**	0.734	**0.761**	0.603	0.744	0.690	**0.762**	0.678	**0.760**	0.445	0.732	0.706	**0.762**
svm2_libsvm	0.619	0.564	0.542	0.495	0.543	0.535	0.535	0.353	0.513	0.301	0.605	0.333	0.608	0.333	0.625	0.375	0.631	0.610	0.613
svm3_libsvm	0.667	0.675	0.657	0.646	0.652	0.661	0.648	0.460	0.621	0.386	0.627	0.412	0.651	0.425	0.651	0.432	0.638	0.635	0.637
svm1_weight	0.747	0.629	0.613	0.614	0.612	0.594	0.588	0.661	0.693	0.563	0.651	0.641	0.708	0.605	0.662	0.338	0.637	0.632	0.681
svm2_weight	0.711	0.551	0.474	0.436	0.567	0.543	0.523	0.360	0.576	0.312	0.589	0.318	0.579	0.310	0.587	0.367	0.668	0.340	0.579
svm3_weight	0.715	0.600	0.530	0.505	0.624	0.604	0.588	0.399	0.606	0.341	0.620	0.362	0.614	0.343	0.616	0.400	0.678	0.382	0.611
svm1_light	0.725	**0.798**	**0.782**	**0.772**	0.745	0.741	0.733	0.687	**0.758**	0.594	0.747	0.673	**0.765**	0.650	**0.764**	0.423	0.679	0.679	0.752
svm2_light	0.654	0.618	0.552	0.522	0.578	0.595	0.602	0.408	0.584	0.296	0.567	0.315	0.603	0.325	0.591	0.390	0.637	0.347	0.584
svm3_light	0.669	0.670	0.630	0.623	0.658	0.657	0.649	0.448	0.608	0.392	0.633	0.368	0.637	0.424	0.652	0.434	0.644	0.442	0.641
svm4_light	**0.781**	**0.786**	0.748	0.725	0.754	**0.762**	**0.765**	0.708	0.756	0.589	0.742	0.674	**0.758**	0.653	0.756	0.444	0.732	0.682	**0.757**
svm1_perf	0.266	0.262	0.347	0.390	0.191	0.245	0.281	0.504	0.323	0.380	0.224	0.541	0.409	0.460	0.273	0.180	0.237	0.487	0.296
svm2_perf	0.327	0.306	0.366	0.400	0.221	0.262	0.291	0.515	0.363	0.388	0.255	0.551	0.453	0.468	0.313	0.233	0.312	0.495	0.330
svm3_perf	0.297	0.306	0.371	0.406	0.223	0.266	0.299	0.518	0.357	0.386	0.248	0.555	0.449	0.470	0.306	0.227	0.285	0.495	0.320
svm4_perf	0.346	0.353	0.390	0.415	0.257	0.288	0.310	0.516	0.386	0.384	0.280	0.546	0.479	0.461	0.344	0.262	0.338	0.487	0.354
svm1_trans	0.677	0.753	**0.765**	**0.766**	0.732	0.735	0.744	0.681	0.714	0.598	0.713	0.642	0.724	0.643	0.726	0.418	0.648	0.666	0.718
svm2_trans	0.678	0.624	0.571	0.537	0.654	0.620	0.611	0.433	0.595	0.341	0.600	0.354	0.599	0.354	0.617	0.397	0.661	0.369	0.601
svm3_trans	0.678	0.719	0.702	0.680	0.713	0.706	0.695	0.474	0.643	0.430	0.665	0.440	0.668	0.442	0.669	0.464	0.669	0.457	0.658
svm4_trans	0.684	0.733	0.722	0.699	0.728	0.739	0.741	0.645	0.694	0.556	0.689	0.602	0.694	0.594	0.699	0.412	0.661	0.619	0.702
svm1_l2l1	0.756	0.648	0.632	0.619	0.508	0.524	0.523	0.628	0.648	0.528	0.589	0.658	0.714	0.589	0.628	0.427	0.692	0.603	0.613
svm2_l2l1	**0.771**	0.665	0.637	0.621	0.517	0.534	0.529	0.622	0.651	0.530	0.603	0.664	0.727	0.585	0.635	0.467	0.732	0.599	0.606
svm_l1l2	0.745	0.684	0.639	0.623	0.546	0.546	0.551	0.619	0.676	0.530	0.631	0.666	0.725	0.579	0.663	0.490	0.712	0.591	0.648
lr1	**0.778**	0.688	0.643	0.618	0.550	0.545	0.549	0.623	0.689	0.515	0.639	0.669	0.742	0.570	0.678	0.495	0.743	0.582	0.658
lr2	**0.799**	0.663	0.625	0.611	0.500	0.509	0.509	0.619	0.665	0.530	0.615	0.662	0.744	0.584	0.646	0.487	**0.768**	0.592	0.619
krr	0.694	0.587	0.601	0.586	0.599	0.630	0.635	0.620	0.644	0.584	0.661	0.502	0.506	0.606	0.624	0.413	0.665	0.642	0.643
nb	0.600	0.691	0.658	0.581	0.679	0.646	0.632	0.362	0.628	0.250	0.582	0.425	0.639	0.259	0.574	0.403	0.565	0.273	0.553
blr	**0.827**	**0.801**	**0.776**	**0.769**	0.756	0.749	0.741	0.754	**0.786**	0.715	**0.782**	**0.768**	**0.816**	0.747	**0.792**	0.527	**0.796**	0.755	**0.791**
adaboostm1	**0.841**	**0.817**	**0.800**	**0.795**	**0.797**	**0.789**	**0.783**	**0.763**	**0.819**	0.710	**0.802**	0.739	**0.815**	0.744	**0.809**	0.584	**0.814**	0.755	**0.812**
Proportion of selected features	100.0%	1.6%	0.4%	0.2%	1.6%	0.7%	0.5%	0.5%	12.6%	4.1%	33.5%	1.5%	24.9%	0.3%	16.0%	5.1%	44.5%	0.8%	19.2%
No. of selected features	7,549	114	31	14	109	49	30	50	873	917	3,570	190	2,222	24	1,384	326	2,923	73	1,827

Note. [a]Cells with dark blue font and highlighting correspond to the top 10% combinations of classifiers and feature selection methods according to precision; cells with light grey font correspond to the bottom 50% combinations of classifiers and feature selection methods according to precision. The proportion of features was individually computed in each data set and then averaged over all data sets. [Table can be viewed in color in the online issue, which is available at wileyonlinelibrary.com.]

TABLE A4. Recall for combinations of classifiers and feature selection methods, averaged over all 229 data sets.[a]

	all	gll_g_k1	gll_g_k2	gll_g_k3	gll_z_k1	gll_z_k2	gll_z_k3	svm_rfe1	svm_rfe2	u1_auc	u2_auc	u1_bs	u2_bs	u1_ig	u2_ig	u1_df	u2_df	u1_x2	**u2_x2**
svm1_libsvm	0.577	0.510	0.438	0.405	0.532	0.519	0.505	0.419	0.548	0.311	0.489	0.363	0.514	0.335	0.501	0.252	0.513	0.370	0.514
svm2_libsvm	0.395	0.379	0.373	0.342	0.373	0.383	0.377	0.228	0.357	0.177	0.442	0.206	0.455	0.200	0.467	0.237	0.437	0.453	0.457
svm3_libsvm	0.425	0.461	0.449	0.428	0.404	0.436	0.437	0.300	0.427	0.217	0.391	0.248	0.463	0.249	0.437	0.260	0.414	0.410	0.414
svm1_weight	0.624	0.705	0.641	0.614	0.710	0.701	0.687	0.549	0.635	0.533	0.632	0.554	0.650	0.525	0.648	0.666	0.665	0.517	0.638
svm2_weight	0.636	0.725	0.687	0.662	0.696	0.691	0.690	0.580	0.653	0.605	0.677	0.549	0.664	0.597	0.676	0.697	0.681	0.577	0.652
svm3_weight	0.619	0.713	0.706	0.690	0.650	0.666	0.675	0.594	0.639	0.615	0.649	0.576	0.662	0.606	0.672	0.671	0.662	0.586	0.642
svm1_light	0.355	0.605	0.602	0.593	0.579	0.590	0.590	0.556	0.562	0.472	0.506	0.501	0.503	0.506	0.540	0.193	0.334	0.519	0.530
svm2_light	0.416	0.411	0.383	0.351	0.376	0.407	0.415	0.275	0.409	0.168	0.370	0.179	0.431	0.189	0.399	0.231	0.407	0.206	0.386
svm3_light	0.427	0.455	0.433	0.418	0.404	0.434	0.435	0.295	0.420	0.219	0.391	0.218	0.458	0.245	0.436	0.261	0.417	0.245	0.415
svm4_light	0.577	0.513	0.441	0.404	0.532	0.518	0.504	0.427	0.549	0.328	0.491	0.377	0.516	0.353	0.502	0.255	0.513	0.377	0.516
svm1_perf	**0.916**	**0.954**	**0.927**	**0.912**	**0.963**	**0.944**	**0.924**	0.827	**0.927**	**0.901**	**0.965**	0.814	**0.907**	0.869	**0.942**	0.737	0.879	0.857	**0.934**
svm2_perf	**0.933**	**0.953**	**0.926**	**0.912**	**0.965**	**0.946**	**0.926**	0.825	**0.926**	**0.898**	**0.961**	0.812	**0.902**	0.868	**0.944**	0.766	**0.910**	0.855	**0.940**
svm3_perf	**0.915**	**0.947**	**0.921**	**0.908**	**0.962**	**0.942**	**0.922**	0.824	**0.919**	**0.895**	**0.958**	0.806	**0.900**	0.863	**0.940**	0.754	0.885	0.852	**0.934**
svm4_perf	**0.910**	**0.933**	**0.922**	**0.909**	**0.943**	**0.937**	**0.922**	0.806	**0.904**	0.875	**0.936**	0.781	0.876	0.838	**0.926**	0.741	0.890	0.827	**0.918**
svm1_trans	0.651	0.644	0.583	0.543	0.552	0.517	0.503	0.447	0.615	0.424	0.613	0.467	0.647	0.419	0.613	0.379	0.626	0.398	0.567
svm2_trans	0.663	0.533	0.434	0.393	0.483	0.424	0.399	0.345	0.536	0.315	0.550	0.355	0.560	0.298	0.511	0.372	0.638	0.287	0.483
svm3_trans	0.656	0.601	0.485	0.425	0.521	0.460	0.431	0.299	0.541	0.254	0.552	0.325	0.597	0.250	0.538	0.438	0.651	0.235	0.494
svm4_trans	0.687	0.611	0.504	0.458	0.558	0.502	0.474	0.460	0.608	0.452	0.600	0.482	0.629	0.443	0.584	0.385	0.644	0.427	0.558
svm1_l2l1	0.565	0.613	0.615	0.610	0.579	0.598	0.606	0.642	0.640	0.536	0.556	0.598	0.618	0.591	0.593	0.269	0.508	0.621	0.616
svm2_l2l1	0.550	0.669	0.668	0.665	0.632	0.649	0.660	0.661	0.652	0.567	0.611	0.622	0.643	0.623	0.644	0.360	0.543	0.653	0.658
svm_l1l2	0.641	0.672	0.652	0.635	0.584	0.602	0.611	0.639	0.676	0.549	0.635	0.626	0.666	0.589	0.648	0.412	0.630	0.597	0.648
lr1	0.622	0.669	0.647	0.631	0.569	0.593	0.601	0.634	0.667	0.534	0.625	0.619	0.657	0.580	0.643	0.415	0.617	0.586	0.639
lr2	0.526	0.677	0.672	0.666	0.643	0.660	0.669	0.667	0.651	0.574	0.607	0.621	0.629	0.629	0.641	0.357	0.530	0.657	0.655
krr	0.230	0.207	0.228	0.229	0.216	0.250	0.264	0.223	0.229	0.202	0.232	0.164	0.170	0.207	0.224	0.113	0.217	0.228	0.230
nb	0.619	0.696	0.532	0.427	0.588	0.486	0.421	0.214	0.581	0.156	0.508	0.346	0.669	0.179	0.580	0.454	0.696	0.157	0.508
blr	0.511	0.627	0.616	0.602	0.569	0.585	0.589	0.569	0.595	0.475	0.547	0.526	0.598	0.512	0.582	0.310	0.507	0.525	0.569
adaboostm1	0.670	0.652	0.642	0.631	0.611	0.598	0.590	0.565	0.636	0.551	0.634	0.566	0.643	0.575	0.639	0.409	0.642	0.569	0.636
Proportion of selected features	100.0%	1.6%	0.4%	0.2%	1.6%	0.7%	0.5%	0.5%	12.6%	4.1%	33.5%	1.5%	24.9%	0.3%	16.0%	5.1%	44.5%	0.8%	19.2%
No. of selected features	7,549	114	31	14	109	49	30	50	873	917	3,570	190	2,222	24	1,384	326	2,923	73	1,827

Note. [a]Cells with dark blue font and highlighting correspond to the top 10% combinations of classifiers and feature selection methods according to recall; cells with light grey font correspond to the bottom 50% combinations of classifiers and feature selection methods according to recall. The proportion of features was individually computed in each data set and then averaged over all data sets. [Table can be viewed in color in the online issue, which is available at wileyonlinelibrary.com.]

DOI: 10.1002/asi

References

Cormack, G.V., Grossman, M.R., Hedin, B., & Oard, D.W. (2010). Overview of the TREC 2010 legal track. In Proceedings of the 19th Text Retrieval Conference (pp 1–45). Gaitherburg, MD: National Institute of Standards and Technology.

DeLong, E.R., DeLong, D.M., & Clarke-Pearson, D.L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. Biometrics, 44(3), 837–845.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., & Lin, C.J. (2008). LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9, 1871–1874.

Milenova, B.L., Yarmus, J.S., & Campos, M.M. (2005). SVM in oracle database 10g: Removing the barriers to widespread adoption of support vector machines. In Proceedings of the 31st International Conference on Very Large Data Bases (pp. 1152–1163). New York, NY: Association for Computing Machinery (ACM).

DOI: 10.1002/asi