Maximizing Text-Mining Performance

Sholom M. Weiss, Chidanand Apte, Fred J. Damerau, David E. Johnson, Frank J. Oles, Thilo Goetz, and Thomas Hampp, IBM T.J. Watson Research Center

THE AUTHORS' ADAPTIVE-RESAMPLING APPROACH SURPASSES PREVIOUS DECISION-TREE PERFORMANCE AND VALIDATES THE EFFECTIVENESS OF SMALL, POOLED LOCAL DICTIONARIES. THEY DEMONSTRATE THEIR APPROACH USING THE REUTERS-21578 BENCHMARK DATA AND A REAL-WORLD CUSTOMER E-MAIL ROUTING SYSTEM.

WITH THE ADVENT OF CENTRALized data warehouses, where data might be stored as electronic documents or as text fields in databases, text mining has increased in importance and economic value. One important goal in text mining is automatic classification of electronic documents. Computer programs scan text in a document and apply a model that assigns the document to one or more prespecified topics. Researchers have used benchmark data, such as the Reuters-21578 test collection, to measure advances in automated text categorization. Conventional methods such as decision trees have had competitive, but not optimal, predictive performance.

Using the Reuters collection, we show that adaptive resampling techniques can improve decision-tree performance and that relatively small, pooled local dictionaries are effective. We've applied these techniques to online banking applications to enhance automated e-mail routing.

Text categorization

Many automated prediction methods exist for extracting patterns from sample cases.[1] These patterns can be used to classify new cases. In text mining, specifically text categorization, the raw cases are individual documents. We can transform these cases into a standard model of features and classes. The cases are encoded in terms of features in some numerical form, requiring a transformation from text to numbers. For each case, we take a uniform set of measurements on the features. A dictionary is compiled from the collection of training documents. We measure the frequencies of occurrence of dictionary words in each document. Prediction methods look at samples of documents with known topics and attempt to find patterns for generalized rules that can be applied to new unclassified documents.

We can describe sample cases in terms of dictionary words or phrases found in the documents. Each case consists of the values of a single article's features; these values could either be Boolean (indicating whether the feature does or does not appear in the text) or numerical (some function of the frequency of occurrence in the processed text). We also label each case to indicate the classification of the article it represents. Our objective is to compute decision criteria that distinguish between text categories. Figure 1 illustrates the organization of necessary tasks.

Given data classified using a standard numerical encoding, we can apply dozens of different data-mining methods. These methods come from many fields, including statistics, machine learning, and information retrieval. We're interested in the numerous secondary characteristics of collecting and applying the dictionary words, including stemmed or unstemmed words and binary (true or false) occurrence or word counts.

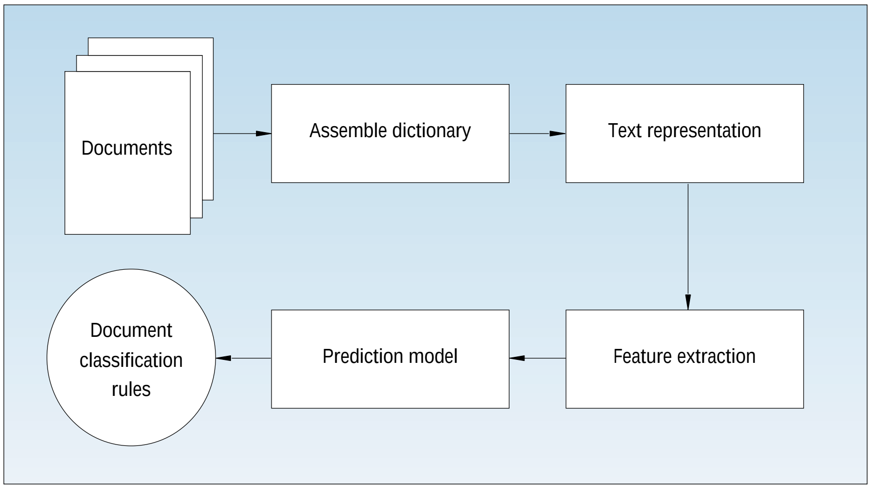


Figure 1. An overview of automated document classification.

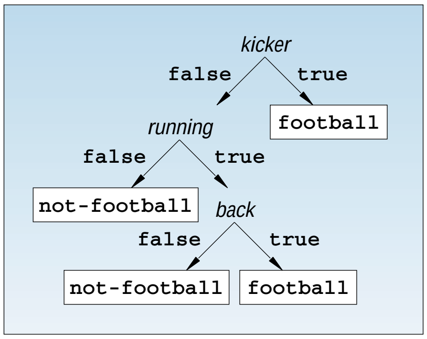


Figure 2. A text-categorization decision tree for classifying document topics as football or not-football stories.

We can check a document for specific words' presence or absence, or for their frequency of occurrence. When compiling the dictionary, we can stem, or map, the words into a common root. For example, we can map plural words into their singular form.

A more critical dictionary characteristic is size. How many words should it hold? (For more information, see the sidebar.) This question's answer depends on many factors, including the prediction method and the available computing resources.

Adaptive resampling

Methods for high-performance text categorization have often employed a universal dictionary of all words in the complete collection of documents. However, methods such as decision trees[2] are more accurate with smaller dictionaries than a universal dictionary, and training is simplified.

Decision trees can perfectly separate training cases by forming a covering tree with true-or-false rules that partition the cases into topic and not-the-topic groups. Figure 2 shows a simplified binary decision tree for classifying document topics as football or not-football stories. All decisions are true or false.

In practice, decision-tree predictive per-

Dictionaries, large or small?

The history of text-categorization approaches covers a relatively short period. Many solutions still employ the original *linear-scoring model*, where you induce a single set of weights for each class from the training documents. The model forms a universal dictionary of all stemmed words in the document collection. Some methods use a feature-selection technique to select a smaller subset of the stemmed words relevant to a particular topic.

Early research with feature selection achieved good results with only a few dozen words per topic. Theoretically, feature selection could be a complex calculation, but in practice, it uses relatively simple computations.[1,2] Applying decision trees improved results modestly. However, decision trees suffered from overfitting to the data. Despite techniques for limiting this phenomenon, empirical experimentation has shown that performance degrades when the dictionary grows too large. Both nearest-neighbor methods on the restricted dictionary and neural nets were competitive with or superior to decision trees. However, both methods incurred large computational costs—for training neural nets and for testing in the nearest neighbor.

Chidanand Apte and his colleagues introduced a *local dictionary*,[3] where the dictionary of words comprises a class's most frequently occurring words. This procedure eliminates computationally expensive feature selection. Decision rules improved performance by several percentage points and had the advantage of a transparent solution that could be readily explained to humans.

Since then, using a universal dictionary of all words in the document collection furthered improved performance. The classical linear methods from statistics usually have severe computational problems in high dimensions. Yet, new training variations have emerged that can find a simple scoring solution with a large dictionary.[4] Researchers have applied nearest-neighbor methods with a universal dictionary, and a variation that learns thresholds to classify multiple topics in parallel does very well.[5] Support vectors, a new method that fits linear, polynomial, or radial basis functions, also does very well.[6] However, these new methods operate in very high dimensions, and some of the training procedures are computationally expensive optimization techniques. Despite performance gains with universal dictionaries, doubts remain about their practical application.

References

1. D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," *Proc. Speech and Natural language Workshop*, Morgan Kaufmann, San Francisco, 1992, pp. 212–217.

2. Y. Yang, "A Comparative Study on Feature Selection in Text Categorization," *Proc. Int'l Machine Learning Conf.*, Morgan Kaufmann, San Francisco, 1997.

3. C. Apte, F. Damerau, and S. Weiss, "Automated Learning of Decision Rules for Text Categorization," *ACM Trans. Information Systems*, Vol. 12, No. 3, July 1994, pp. 233–251.

4. I. Dagan, Y. Karov, and D. Roth, "Mistake-Driven Learning in Text Categorization," *Proc. Second Conf. Empirical Methods in Natural Language Processing*, Assoc. for Computational Linguistics, New Brunswick, N.J., 1997.

5. Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," to be published in *J. Information Retrieval*, May 1999.

6. T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, tech. report, Univ. of Dortmund, Dept. of Informatics, Dortmund, Germany, 1997.

formance is not as strong as in the training cases. This phenomenon is often described as *overfitting*, where the tree is too specialized to the training data. Various techniques can prune the tree to reduce the training accuracy with a corresponding increase in predictive performance on new test documents. With very large dictionaries, overfitting is more likely because many words can separate the classes based only on random occurrences of combinations. This has led to the observation that smaller dictionaries can help reduce overfitting, hence improving predictive performance. Yet, reducing the dictionary size might throw out useful words that could potentially improve performance.

From a statistical perspective, overfitting decision trees is another way of saying that the variance is too high. Thus, the error observed on the training data could vary greatly on new test data. Researchers have observed that you can greatly reduce the variance by inducing multiple decision trees from the same data.[3] Giving a vote to each of the multiple decision trees and picking the answer with the most votes determine the final answer on a new case. For example, if we induce 10 decision trees, they are all applied to a new case. If six trees vote class 1, and four trees vote not class 1, then the topic for class 1 is assigned.

As the number of induced decision trees increases, the variance gradually decreases until a plateau is reached. In practice, multiple trees usually do better than a single tree, often far better. Ten trees are typically sufficient to model most of the expected gains.

How do we generate the trees? No change is made to a standard tree induction method. The difference is the sampling method. In *bagging*,[3] the simplest approach, a sample of size n is taken with replacement from the original set of n documents. Some documents will be repeated in the sample; others might not occur. The expected proportion of unique documents in any given sample is 63.2%. Thus, you can generate many samples, induce a decision tree from each sample, and then vote the results.

Adaptive resampling, an alternative method of sampling, usually performs better than bagging. Instead of sampling all cases randomly, so that each document has a $1/n$ chance of being drawn from the sample, adaptive resampling uses an incremental approach to random selection. The objective is to increase the odds of sampling documents that previously induced trees have erroneously classified.

We can specify two characteristic variations for improving performance using multiple trees:

- weighted or unweighted multiple-tree voting and
- different formulas for estimating the likelihood of errors for individual cases.

All techniques for optimizing performance by adaptive resampling vote multiple trees. Some algorithms use weighted voting, which gives some trees more weight than others. The original *boosting* algorithm used weighted voting and an explicit formula for updating the likelihood of sampling each case.[4] However, unweighted voting and resampling with any technique that greatly increases the likelihood of including erroneous cases can also greatly improve accuracy.[3] Thus, we do not need to restrict boosting to adaptive term weighting[5,6] or to adaptive weighting of different classifiers while combining their results. Rather, boosting can also work effectively for adaptive weighting of training examples, as our results will show.

SMALLER DICTIONARIES CAN HELP REDUCE OVERFITTING, HENCE IMPROVING PREDICTIVE PERFORMANCE.

The theory of adaptive resampling is still under active investigation.[7] A key principle is to iteratively induce new classifiers by increasing the weight of erroneously classified cases in the next sample. You can generate a new sample either randomly or by simply weighting cases for classifiers that can use weighted cases. Researchers have reported strong results using standard classifiers such as decision trees and extremely simplistic classifiers such as trees that are restricted to a few terminal nodes.

We follow the simpler approach of unweighted multiple-tree voting. We maintain a table of the cumulative errors on the training cases for the current trees. For example, suppose we have three trees. When we apply each tree independently, the cumulative number of errors for case 1 is three; for case 2 it is zero. The sampling procedure is adaptive and will greatly increase the likelihood of drawing case 1 in the next random sample drawn for the fourth tree.

As long as the likelihood of selecting erroneous cases is greatly increased, the exact function for increasing the weighting of error cases is not critical to improving performance.[3] Let $e(i)$ be the cumulative number of errors for case i and all independent trees. Then, the following pseudoprobability function suffices to randomly sample case i with likelihood $pr(i)$:[1]

$$pr(i) = \left(1+e(i)^3\right) \Big/ \sum_j \left(1+e(j)^3\right).$$

Adaptive resampling applies to many learning methods. However, resampling is particularly advantageous combined with decision trees because:

- Decision tree algorithms are relatively efficient in high dimensions.
- Decision trees have a larger variance than other methods such as nearest neighbor or neural nets. When the cases are slightly changed, possibly by duplicating or deleting a few cases, the induced trees can differ greatly.

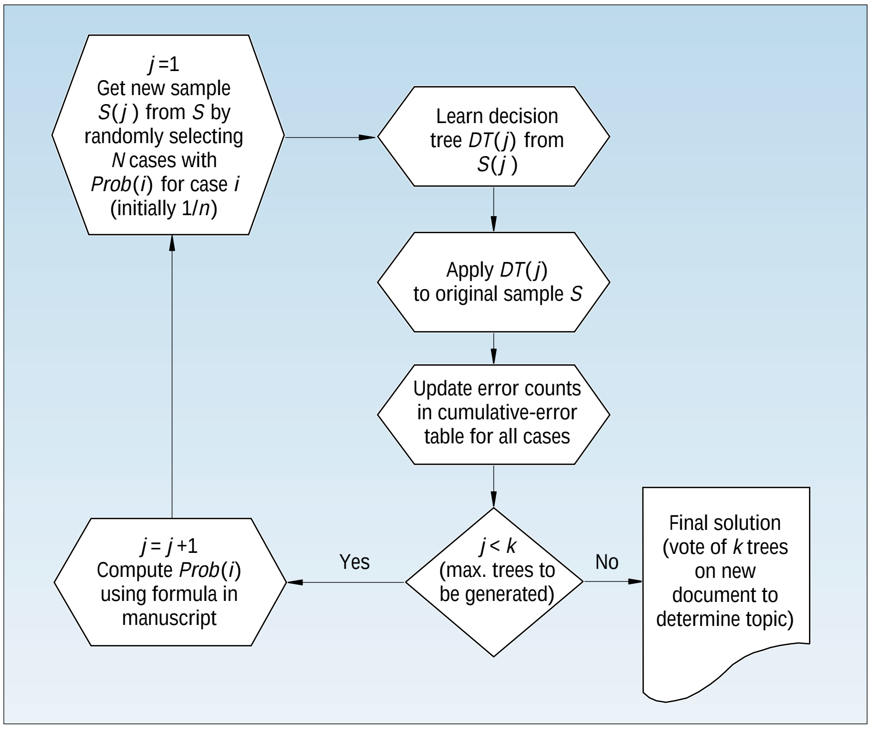
Figure 3 outlines the process flow of our system's adaptive-resampling mechanism. We have applied adaptive resampling in this form to many applications.The binary (true-or-false) trees are slightly pruned by a statistical test with a default setting. (Our default implementation uses a significance level of 1.5 standard errors. We performed no experiments at a different level.) Unpruned trees are only slightly less effective. We do not tune the tree parameters. The more trees the better, but we have observed that it usually is not worth going beyond 100 trees. This approach's ultimate test is to examine its results on real applications.

The Reuters-21578 collection

With 9,603 training documents and 3,299 test documents, our Reuters-21578 benchmark's variation corresponds to the Mod-Apte organization. We used 95 topics—all those with at least two training cases. Two training cases are the minimal number for finding patterns and self-testing. Topics that have only one training case occur 11 times in the test cases. Some researchers have removed topics that do not have at least one training and one test case. That removes topics that have no test cases but also many training cases. With approximately 3,500 instances of topics assigned to test cases, these slight differences in topic selection have a negligible effect on overall predictive performance. For further details, see the Readme file for the Reuters text distribution at http://www.research.att.com/~lewis.

The standard evaluation criterion for the Reuters benchmark is the *breakeven* point, at which precision equals recall. Given a binary-classification problem of topic versus not-topic, recall is the ratio of the correct topic cases to the total topic cases. Precision is the ratio of correct topic cases to total predicted topic cases. If we add the three individual components of these measures over all topics, we can compute an overall performance measure. For the Reuters application using decision trees, recall typically is less than precision. Therefore, we need some artificial means to find a breakeven point. To be fair, the measure must entail some global adjustment to the learning program, such as changing the decision threshold for all topics to a value different than that for minimum error. Extracted from the summary in Thorsten Joachim's report for the University of Dortmund,[8] Table 1 lists the reported breakeven results of controlled experiments for the Reuters-21578 collection. (The results for rule induction exactly repeat the procedures Chidanand Apte and his colleagues used on the Reuters-21578 collection.[9])

Using complex optimization procedures and a universal dictionary, the support-vector method emerges as the best performer. All these methods require selections among various parameters that are not necessarily obvious or stable. For example, the nearest-neighbor results are for 30 neighbors. Results for other methods—run on an earlier version of the Reuters collection—suggest that they might come within a percent or two of the support-vector method's performance. For example, a parallel nearest-neighbor method that uses local thresholds or an optimal linear least square method with a universal dictionary perform around the 85% level.[10] We use defaults for our decision trees, train for a prespecified number of trees, and reserve no cases for tuning. We strictly base the number of induced trees on resources. We expect that with 10 trees, results will improve greatly over a single tree and leave a relatively small margin for improvement. With more than 10 trees, results will gradually increase in accuracy until they converge to the limit of predictive performance at 100 trees.



j =1 Get new sample $S(j)$ from S by randomly selecting N cases with $Prob(i)$ for case i (initially $1/n$)
Learn decision tree $DT(j)$ from $S(j)$
Apply $DT(j)$ to original sample S
Update error counts in cumulative-error table for all cases
$j < k$ (max. trees to be generated)
Yes
No
$j = j+1$ Compute $Prob(i)$ using formula in manuscript
Final solution (vote of k trees on new document to determine topic)

Figure 3. Adaptive-resampling-based text categorization.

For voted answers, such as an ensemble of decision trees, we can use a relatively simple procedure to find the breakeven point. Voting decision trees minimizes error. For a binary-classification application, a majority of votes wins. To get a breakeven point, you can slightly alter the number of votes needed to win. For example, you could have 25 voting trees, which requires a boundary of 13 to win. Instead, you could assign a boundary of 11 as the global breakeven point. Using this procedure, we can find an almost exact breakeven point. If necessary, we can use linear interpolation between the crossover number of votes, which moves from greater to lesser recall versus precision.

Voting an ensemble of decision trees is effective for applications where the trees differ markedly. The dictionary size will effect the variance. The larger the dictionary, the more likely that results will vary from training to test documents. Thus, a small local dictionary would not appear appropriate or adequate for maximizing document-classification performance.

An alternative to the universal dictionary is a pooled local dictionary. Local dictionaries of size k are generated for each topic and are then pooled into a single dictionary. If k is large enough, the pooled local dictionary is identical to the universal dictionary. If k is relatively small, the pooled dictionary is substantially smaller than the universal dictionary.

Predictive performance

We examined all topics with two or more training cases in the Reuters collection. Following removal of stopwords, the data presented to the learning programs (that is, decision-tree procedures) consisted solely of unnormalized frequency counts of unstemmed words. Words occurring in a headline received one extra count.[9]

Table 2 summarizes the experiment's results. We performed most experiments with the 2,133-word dictionary, for which a single tree induced for each topic had a breakeven point just below 80%.

In addition to the number of voted trees and their breakeven percentage, the table

Table 1. Reported breakeven points for the Reuters collection.

Prediction method	Performance breakeven
Naive Bayes (linear)	**73.4**
Rocchio (linear)	**78.7**
Decision tree C4.5	**78.9**
***K* nearest neighbor**	**82.0**
Rule induction	**82.0**
Support vector (radial basis function)	**86.3**

Table 2. Results of voting multiple decision trees using pooled local dictionaries.

Number of trees	Words trained	Words applied	Mean tree size	Breakeven point
10	2,133	1,156	14.6	85.9
25	2,133	1,453	16.6	87.0
50	2,133	1,553	17.2	87.7
100	2,133	1,667	17.2	87.8
50	3,878	2,276	16.7	87.7
50	7,595	2,799	15.9	87.1

Table 3. Recall and precision for the 10 most frequent topics.

Topic	Recall	Precision
Earnings	99.0	96.6
Acquisitions	96.1	95.3
Money-fx	77.1	75.8
Grain	95.3	91.6
Crude	92.6	85.0
Trade	81.2	70.4
Interest	64.9	83.3
Ship	76.4	86.1
Wheat	97.2	83.1
Corn	98.2	82.1

summarizes the mean tree size and the number of words in the dictionary for training and for applying the trees to new documents. The 2,133 training words were the pooled dictionaries of the top 150 words for all topics. Once the top 150 words were found for a topic, we removed the stopwords. We also found and entered the top 500 hundred words for the unused topics into the 2,133-word dictionary, adding approximately 50 new words.

100 trees achieves the best performance but is only marginally better than 50 trees. This is consistent with the experience of others who have evaluated adaptive-resampling-based decision trees on nontextual applications.[3] Most of the gains are achieved with 10 trees. Researchers reported that the performance of relatively unpruned trees generally converges to a minimum error with approximately 100 trees. Although the overall error is relatively small at 87.8%, more than 50% of the cases are from the two most frequent topics, for which classification results are very strong. Table 3 lists the recall and precision for the 10 most frequent topics. Predictive performance, especially recall, degrades markedly as a topic's prevalence decreases.

We also performed two experiments with larger dictionaries. In one experiment, the number of words collected for each local dictionary was doubled to 300 and pooled, resulting in a 3,878-word dictionary. In a second experiment using a 7,595-word dictionary, we merged the pooled local 2,133-word dictionary with a universal dictionary of all words with a global frequency of five or more. In neither case do results improve over the simple 2,133-word dictionary.

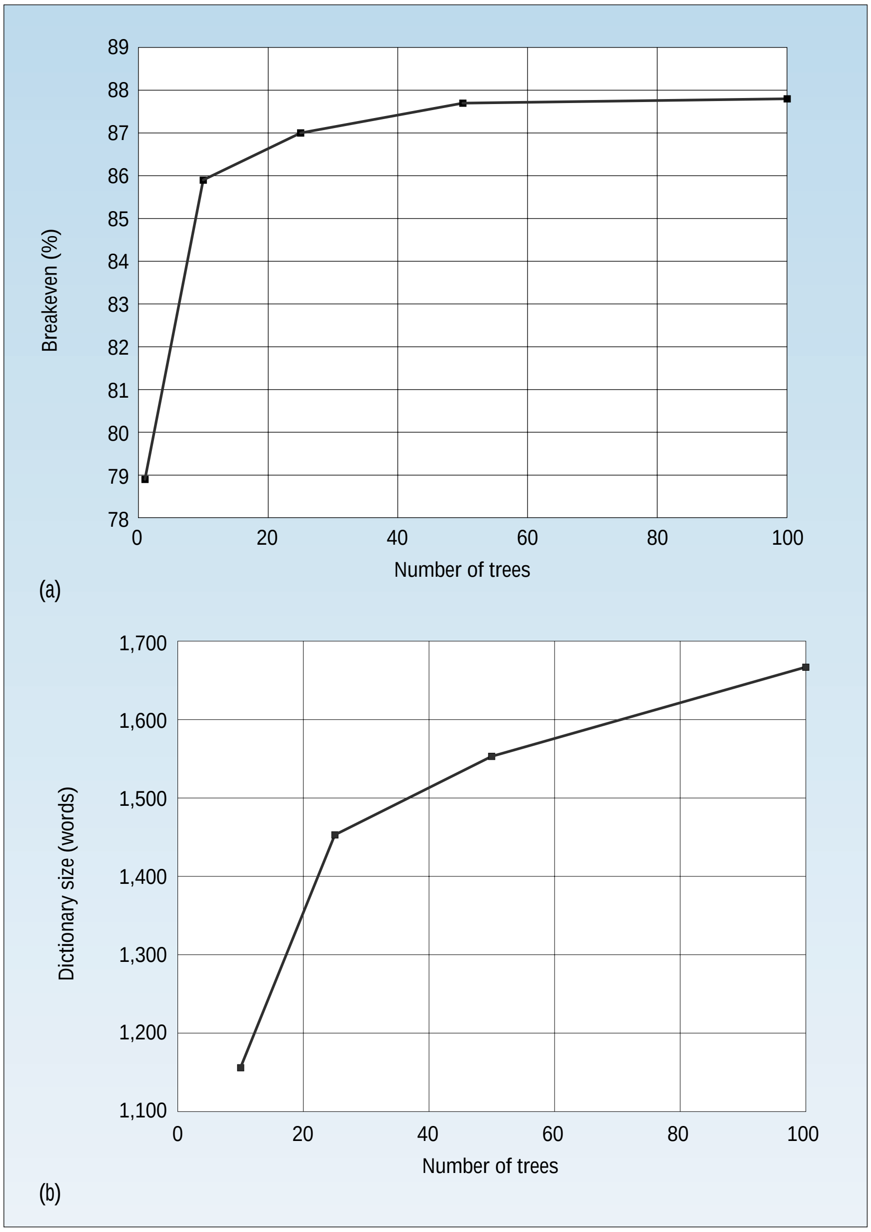


Figure 4. The effect of the number of voting trees on (a) performance and (b) dictionary size.

Discussion. Voting an ensemble of decision trees demonstrated a better performance than competitive text-mining methods. We can approximate a measure of significance of breakeven results by using the standard error, 0.6%, of 3,299 test cases with an error rate of 0.15. Two standard errors, a typical measure of significance, are a 1.2% difference in performance for two classifiers. The results are likely to be much more significant than this measure because we perform multiple experiments over many topics. The results are highly scalable, with the greatest gains found by inducing just 10 trees per topic.

The best previously reported results were achieved with universal dictionaries and somewhat complex prediction methods. For example, the support-vector result relies on tuning parameters and complex optimizations, which might be unstable for future applications. The cost of matching new documents is relatively high for nearest-neighbor methods that use large dictionaries and large training sets. To some extent, parallel computing and document clustering can alleviate these costs.

Adaptive resampling requires no special estimation or tuning of individual parameters. Multiple trees are generated and voted. Also, our results demonstrate that pooled local dictionaries suffice to capture the key words in a much smaller dictionary than the universal dictionary, which also might not be a realistic option in many other applications. Our experiments achieved their results with only a 2,133-word training dictionary. The runtime dictionary contained fewer than 1,200 words for all topics when voted with 10 trees. As illustrated in Figures 4a and 4b, an increase in the number of trees increases predictive performance, but also increases the size of the applied dictionary.

Although many trees are evaluated at runtime, the results for the Reuters experiments show that the average tree is quite small, with approximately only 16 terminal nodes. Most of the topics have relatively small trees, corresponding to the low prevalence of many topics. The induced trees are binary, and assuming a balanced tree, the approximate number of logical comparisons is $\log_2$ of the number of terminal nodes. This suggests that the number of logical comparisons is negligible even for 50 trees per topic and is much more modest than even a linear classifier using a full universal dictionary.

An advantage of voted trees is the relative ease with which you can trade off precision and recall. To make comparisons, we employed an artificial breakeven point standard. However, for voted trees, we can adjust these trade-offs simply by varying the number of votes needed to win.

A weakness of this approach is the lack of clarity or explanation of decisions relative to a single decision tree or rule-based solution. Performance between the transparent solution for decision rules and the voted decision trees differs by 6%. However, voting as few as five trees will surpass the weaker result. Voting a considerable number of trees will result in superior performance with little difficulty. However, the decision criteria will not be transparently expressed.

In our experiments, we employed full trees with only mild pruning. Alternatively, you could induce many very small trees.[7] The support-vector model continues to show promise with new techniques for much quicker training than described in earlier studies.[11]

Real-world testing

The Reuters collection is—although extensively studied and now quite well understood—a somewhat artificial collection. It was collected long ago to construct classification rules. Also, two categories, "acquisitions" and "earnings," account for approximately 55% of the data. A data set with a more even distribution of cases across categories would be more challenging.

To evaluate the utility of resampling and other techniques in an operational situation, we used a collection of electronic mail that had been sent to a leading bank, NationsBank. The bank considers this proprietary data, and we can not release it to the research community.

Upon receipt, human coders first assigned each piece of incoming mail to an application area and then to a category within that application. The assigned category determined where the mail was routed. We focus on the area of online banking. The top five categories in this area account for approximately 70% of the total online banking documents. Because we have found that categories that have too few documents are often not suitable for training a classifier, we limited the training and testing to those categories occurring 50 or more times—a total of 6,352 documents. Eighteen classification categories meet this criterion, which have only 22 documents assigned to more than one category. We randomly split the collection into two sets: 80% for training and 20% for testing.

We trained classifiers on the data's training portion using two different decision tree algorithms: Sprint, which is part of the IBM Intelligent Miner toolkit,[12] and the DMSK (Data-Miner Software Kit) decision tree and resampling algorithm used in the Reuters experiments.[1] Also, we trained a classifier based on the multiplicative update learning algorithm.[13]

Table 4 shows the results of these runs in terms of the precision and recall achieved on the test data set—that is, the 20% reserved from the total document collection. We give results for each of the five most frequent categories and list an overall measure for all 20 categories. To consider the different number of cases in each category, we compute an overall (microaverage) measure by summing the components of the recall and precision measures over all 20 binary classifications. Precision and recall for all methods are in the 80% to 90% range overall. For the five most frequent categories, both precision and recall are mostly close to or over 90%. These categories account for 81.4% of the test documents. This means that for four out of five documents, we could send that piece of mail to the correct queue for disposition approximately 90% of the time.

Depending on the penalty for sending an erroneous reply, you might envision automatically answering an incoming e-mail 80% of the time. However, because one of the categories in the top six is "close account," it seems unlikely that an automatic reply even at a 90% correct level would be acceptable. Conversely, routing to the appropriate location for a reply appears to be practical. Many documents in the "copy of check" category have a stereotypical text, which makes the classification problem simpler. Some do have free text. Decision trees seem to take advantage of the high uniformity, but the multiplicative update algorithm does not.

Unlike the Reuters newswire application, in the banking domain, the classes are essentially mutually exclusive. Almost always, e-mail is routed to a single queue for a response.

Table 4. Precision and recall for the top five home-banking categories.

Category	Sprint	DMSK tree	DMSK ensemble	Multiplicative update learning algorithm
Copy of check	1.000 / 0.923	0.991 / 0.979	0.994 / 0.944	0.881 / 0.988
Change of address	0.972 / 0.935	0.937 / 0.969	0.965 / 0.952	0.948 / 0.939
Check order reorder	0.825 / 0.902	0.845 / 0.986	0.860 / 0.933	0.835 / 0.898
Close account	0.899 / 0.861	0.846 / 0.917	0.909 / 0.833	0.956 / 0.915
Copy of last statement	0.881 / 0.897	0.867 / 0.897	0.879 / 0.879	0.824 / 0.913
All 20 categories	0.903 / 0.812	0.841 / 0.841	0.894 / 0.894	0.846 / 0.869

In such applications, the multiple-tree approach can particularly shine because multiclass classification need not be decomposed into the binary classification of class versus not-class. For applications where multiple classes can be assigned to a single new text document, complexity grows proportionally to the number of classes, and an alternative, such as a parallel nearest neighbor, becomes increasingly attractive.

REAL BUSINESS SITUATIONS EXIST where automatic document routing is sufficiently accurate to be useful. After all, human routers are not perfect themselves. At the moment, however, no way exists to characterize these likely successes a priori. Currently, you can only tell through experimentation whether a particular application lends itself to these techniques. With experience gained from the deployment of real-world categorization systems, we can expect further advances in automation and performance. ■

References

1. S. Weiss and N. Indurkhya, *Predictive Data Mining: A Practical Guide*, Morgan Kaufmann, San Francisco, 1998.
2. C. Apte and S. Weiss, "Data Mining with Decision Trees and Decision Rules," *Future Generation Computer Systems*, Vol. 13, Nos. 2–3, Nov. 1997, pp. 197–210.
3. L. Breiman, "Bagging Predictors," *Machine Learning*, Vol. 24, No. 2, 1996, pp. 123–140.
4. Y. Freund and R. Schapire, "Experiments with a New Boosting Algorithm," *Proc. Int'l Machine Learning Conf.*, Morgan Kaufmann, San Francisco, 1996, pp. 148–156.
5. R. Schapire, Y. Singer, and A. Singhal, "Boosting and Rocchio Applied to Text Filtering," *Proc. ACM Special Interest Group on Information Retrieval '98*, Morgan Kaufmann, San Francisco, 1998.
6. R.E. Schapire and Y. Singer, *BoosTexter: A Boosting-Based System for Text Categorization*, to be published in *Machine Learning*, 1999.
7. J. Friedman, T. Hastie, and R. Tibshirani, *Additive Logistic Regression: A Statistical View of Boosting*, tech. report, Statistics Dept., Stanford Univ., Stanford, Calif., 1998; http://www-stat.stanford.edu/~tibs.
8. T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, tech. report, Univ. of Dortmund, Dept. of Informatics, Dortmund, Germany, 1997.
9. C. Apte, F. Damerau, and S. Weiss, "Automated Learning of Decision Rules for Text Categorization," *ACM Trans. Information Systems*, Vol. 12, No. 3, July 1994, pp. 233–251.
10. Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," to be published in *J. Information Retrieval*, May 1999.
11. S. Dumais et al., *Inductive Learning Algorithms and Representations for Text Catgeorization*, tech. report, Microsoft Research, Redmond, Wash., 1998.
12. *Intelligent Miner User's Guide*, Tech. Report SH12-6213-01, Software Solutions Division, IBM, 1997.
13. I. Dagan, Y. Karov, and D. Roth, "Mistake-Driven Learning in Text Categorization," *Proc. Second Conf. Empirical Methods in Natural Language Processing*, Assoc. for Computational Linguistics, New Brunswick, N.J., 1997.

Sholom M. Weiss is a research staff member at the IBM T.J. Watson Labs. His research interests involve innovative methods of data mining. He is an author and coauthor of many publications on artificial intelligence and machine learning, including *Predictive Data Mining: A Practical Guide* (Morgan Kaufmann, 1997). He is a fellow of the AAAI and serves on numerous editorial boards. Contact him at the IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; sholom@us.ibm.com.

Chidanand Apte is a research staff member at the IBM T.J. Watson Research Center, where he manages the Data Abstraction Research group. His technical interests include data-mining algorithms and systems, and solutions for business intelligence. He holds a PhD in computer science from Rutgers University. He is a senior member of the IEEE and a member of the AAAI. Contact him at the IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; apte@us.ibm.com.

Fred J. Damerau is on the research staff at the IBM T.J. Watson Research Center. He is also an adjunct professor at Pace University. His technical interests include natural-language processing, artificial intelligence, machine learning, and information retrieval. His work focuses on text categorization and related issues. He received his BA in mathematics from Cornell University and his MA and PhD in linguistics from Yale University. He is a member of the ACM, Association of Computational Linguistics, and AAAI. Contact him at the IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY, 10598; damerau@us.ibm.com.

David E. Johnson is a research staff member and the manager of the natural-language understanding group in the Mathematical Sciences department at the IBM T.J. Watson Research Center. His research focuses on machine-learning-based text categorization and text analysis, with a particular interest in systems that automatically process e-mail. He received his PhD in theoretical linguistics from the University of Illinois at Urbana-Champaign. He is a past president of the Association for the Mathematics of Language and has served on the editorial boards of several journals, including *Computational Linguistics* and *Machine Translation*. He is a member of the Association of Computational Linguistics. Contact him at the Mathematical Sciences Dept., IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; dejohns@us.ibm.com.

Frank J. Oles is a research staff member at the IBM T.J. Watson Research Center. He is working on applying machine learning to text categorization. Previously, he worked on knowledge representation with applications to medical informatics and on the mathematical foundations of programming-language semantics. He holds a PhD in computer and information science from Syracuse University. Contact him at the IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; oles@watson.ibm.com.

Thilo Goetz is a research associate and staff programmer at the IBM T.J. Watson Research Center. His interests include natural-language processing and machine learning. He holds an MA in computational linguistics from the University of Tübingen, Germany. Contact him at the IBM T.J. Watson Research Center, Natural Language Understanding Group, P.O. Box 218, Yorktown Heights, NY 10598; goetz@watson.ibm.com.

Thomas Hampp is a software engineer for Software Solutions Development, IBM Germany. He is working on text-mining development. His main technical interest is the architecture and design of natural-language processing systems. Previously, he worked at the IBM T.J. Watson Research Lab, focusing on text-classification systems. He has an MA in computational linguistics from the University of Tübingen, Germany. He is a member of the ACL. Contact him at IBM Germany, Software Solutions Development (SWSD), DMS Text Mining Development, Hanns Klemm Str. 45, 71034 Böblingen, Germany.