# BNS Feature Scaling: An Improved Representation over TF·IDF for SVM Text Classification

George Forman
Hewlett-Packard Labs
Palo Alto, CA, USA
ghforman@hpl.hp.com

## ABSTRACT
In the realm of machine learning for text classification, TF·IDF is the most widely used representation for real-valued feature vectors. However, IDF is oblivious to the training class labels and naturally scales some features inappropriately. We replace IDF with Bi-Normal Separation (BNS), which has been previously found to be excellent at ranking words for feature selection filtering. Empirical evaluation on a benchmark of 237 binary text classification tasks shows substantially better accuracy and F-measure for a Support Vector Machine (SVM) by using BNS scaling. A wide variety of other feature representations were later tested and found inferior, as well as binary features with no scaling. Moreover, BNS scaling yielded better performance without feature selection, obviating the need for feature selection.

## Categories and Subject Descriptors
H.3.3 [**Information Search & Retrieval**]: Information filtering; I.5.2 [**Pattern Recognition**]: Design methodology, *feature evaluation and selection.*

## General Terms
Algorithms, Performance, Experimentation.

## Keywords
Text classification, machine learning, feature selection, feature weighting, Support Vector Machine, TF*IDF text representation.

## 1. INTRODUCTION
Text classification via machine learning is at the heart of effective document categorization, personalization, news filtering, and information routing. State-of-the-art classification accuracy can be achieved by applying a linear Support Vector Machine (SVM) to a 'bag-of-words' representation of the text, where each unique word in the training corpus becomes a separate feature [1][9][10]. The numerical feature value for a given word/term is often represented by its term frequency TF in the given text multiplied by its inverse document frequency (IDF) in the entire corpus—the ubiquitous 'TF·IDF' representation. IDF is commonly taken to be log( # documents ÷ # documents containing the term). By multiplying by IDF, the common functional words such as 'of' and 'can' are devalued relative to the uncommon words that are more likely topic-specific indicators.

Although TF·IDF is widely used in text classification, it is oblivious to the class labels in the training set, which can lead to inappropriate scaling for some features. Consider a toy example: a word X occurs in 80% of the positive training cases and another word Y occurs in only 3% of the positive cases—suppose neither occurs among the negative training cases. IDF gives a super-linear boost to words with lower frequency of occurrence. But in this case the more common word is a much stronger predictor. A specific, real example is illustrated later in Section 5.1.

Filter methods for feature selection have developed a variety of metrics that do a good job of correctly ranking the predictive value of different features, e.g. Bi-Normal Separation (BNS) [4].

In this paper we improve on the state-of-the-art by using the BNS feature scoring metric in a new way: to scale the magnitude of the feature values. That is, we compute the BNS score for each feature and use TF·BNS for each feature value, replacing IDF. This increases the effect of important words on the kernel distance computations for the SVM. We show that this simple idea substantially improves SVM accuracy and F-measure on a benchmark of 237 binary text classification tasks, especially when TF is restricted to be binary. For comparison, we also tested a dozen other metrics and found that BNS scaling performed best.

BNS feature selection was previously shown to substantially improve text classification [4]. One of the difficulties of feature selection, however, is in deciding the optimal number of features to use. The new method of BNS scaling offers to simplify the process, because it consistently performed best by using *all* features. This has an intuitive appeal of not 'throwing away' information. For those situations where the volume of data must be reduced for computational scalability at the cost of classification accuracy, we recommend a hybrid that uses Information Gain for feature selection and BNS for feature scaling, based on our empirical study.

### 1.1 Related Work and Scope
In this space of classification research, some work addresses binary classification [4][9][13], as in *information filtering*, e.g. separating spam from good email. Other work addresses multi-class classification [17], e.g. *routing* or classifying a document into one of many categories. Our focus here is on binary classification, but we expect the results to generalize. Binary tasks are an important sub-problem in most multi-class classification methods, which decompose the 1-of-n problem by pitting each class against the others. Finally, we note that the problem n-of-m multi-class classification, e.g. *topic recognition*, is addressed by m independent binary classifiers.


*CIKM'08*, October 26–30, 2008, Napa Valley, California, USA.

There is a large research literature in feature selection metrics to filter words for text classification. The goal is often to improve accuracy, but in some papers it is to preserve accuracy as much as possible as the number of features is reduced in order to decrease computational workload. In this work our goal is simply to maximize classification performance, as measured by accuracy and F-measure. However, in the process, we developed a method that satisfies both goals.

As a side note, in non-text domains, there has been a lot of work that takes a collection of features with widely disparate ranges and normalizes or discretizes them to make them palatable for induction algorithms. These concerns led to the common practice of normalizing the feature space for SVMs. Instead, we modify the normalization phase as an additional opportunity to condition the data for learning. Finally, there are many references to 'feature scaling' or 'feature weighting' in the literature that simply refer to variations on normalization. For example, word counts are sometimes scaled so that long documents appear to have a uniform number of words as short documents. In contrast, this work scales the feature ranges based on the supervised labels. The closest references to this sort of feature weighting in the literature are in lazy learning (case-based learning), where the goal is to learn an appropriate distance function so that the nearest neighbors of a novel case suggest appropriate class labels [15]. Such methods constitute iterative wrapper searches to adjust the feature weights of a (linear) distance metric. By contrast, in our work, we let the SVM perform the inductive learning, and we simply condition the input feature vectors with a single, fast pass over the training set. Any of the feature scoring metrics and IDF can be computed in a single linear scan the training corpus.

# 2. METHODS

Here we briefly define the feature scoring metrics and how they are used for feature selection and/or for feature scaling.

## 2.1 Feature Selection via Filtering

Filtering methods for feature selection evaluate each feature independently via a chosen scoring metric. Then, some number of top-scoring features is selected for input to the induction algorithm. One can either specify the *number* of top-ranked features to select, or equivalently, specify a particular score *threshold*, which is particular to the feature scoring metric being used. In order to compare different scoring metrics on a common x-axis scale, the former is often preferable in research.

## 2.2 Feature Scaling

The key idea of this paper is to use a feature scoring metric to make the numeric range greater for more predictive features, just as IDF attempts to do in TF·IDF. This affects the dot-product distance between cases as evaluated by the linear SVM kernel [9]. For example, if the BNS score is 2.1 for the word feature 'free' in a spam classification task, then its Boolean presence or absence in a document would be represented as either 0 or 2.1, rather than 0 or 1. A less predictive word 'cat' with a BNS score of 0.3 would have the smaller range 0 or 0.3, and therefore have less effect on the kernel distance computations. This basic idea can be applied to any scoring metric. Furthermore, it can also be applied to non-binary features, e.g. to scale term frequency TF counts: TF·BNS.

Additionally, feature scaling may be used in conjunction with feature selection. The scoring metric used for feature selection may be different than the metric used for feature scaling. The best method found by our experiments is such a hybrid: IG used for feature selection and BNS for feature scaling.

## 2.3 Feature Scoring Metrics

The primary feature scoring metrics we use in this paper are defined as follows.

| | |
|---|---|
| Bi-Normal Separation (BNS): | $\lvert F^{-1}(tpr) - F^{-1}(fpr) \rvert$ |
| Inverse Document Freq (IDF): | $\log( (pos+neg) \div (tp+fp) )$ |
| Log Odds Ratio [13]: | $\log( (tp \cdot tn) \div (fp \cdot fn) )$ |
| Information Gain (IG): | $H(data) - H(data \mid word)$ |
| | $= H(pos,neg) - (P(word)\, H(tp,fp) + (1-P(word))\, H(fn,tn) )$ |

where

pos = number of positive training cases, typically minority,
neg = number of negative training cases,
tp = number of positive training cases containing word,
fp = number of negative training cases containing word,
fn = pos – tp,
tn = neg – fp,
true positive rate *tpr*=P(word | positive class) = tp/pos,
false positive rate *fpr*=P(word | negative class) = fp/neg,
P(word) = (tp+fp) / (pos+neg),
entropy $H(x,y) = -nln(x/(x+y)) - nln(y/(x+y))$,
$nln(x) = x \log_2 x$, and
$F^{-1}$ is the inverse Normal cumulative distribution function, as commonly available from statistical tables.

Note that these are computed using binary word features, i.e. many occurrences of a word in a single document only count toward one tp or fp count. Information Gain and BNS are naturally symmetric with respect to positively and negatively correlated features. Log Odds Ratio, however, assigns a very low score to a strongly predictive feature that occurs in almost all negative cases but in none of the positive cases. To rectify this, for any negatively correlated features we reverse the meaning of a word occurrence to be a word non-occurrence, i.e. $tp \leftrightarrow fn$ and $fp \leftrightarrow tn$. This solution improves a number of feature selection metrics that otherwise ignore strong negative features [4].

As usual, there are some nuances to converting these straightforward mathematical definitions to robust code. For example, Log Odds Ratio is undefined if tp, tn, fp, or fn is zero. To avoid this, we substitute 0.5 for any zero count, which has the desirable property that even if some of the variables are zero, the function remains responsive to the magnitude of the other variables. Likewise, for IG we define nln(x) = 0, whenever x = 0. Finally, in the BNS function, the inverse Normal goes to infinity at zero or one; hence, we limit *tpr* and *fpr* to the range [0.0005, 1-0.0005]. Laplace smoothing is a more common method to avoid these extreme probabilities, but it damages the maximum likelihood estimate, and it loses the good performance of BNS by devaluing many valuable negative features in favor of very rare positive features [4]. Alternately and perhaps preferably, one could substitute a fractional count if tp or fp is exactly zero; this may work better for extremely large training sets. We used a fixed limit because we used a finite size lookup table for the inverse Normal function, generated by Gnuplot's invnorm() function and transferred to our Java code, since this standard statistical function is not available in the Java math libraries.

# 3. EXPERIMENT DESIGN

Our experiments consider a wide variety of text feature representations. For each potential feature scoring metric, we consider using it as a scale factor on TF feature counts and separately as a scale factor on binary features. We also consider plain binary features, as well as raw, unscaled TF features. In Section 4.6, we also combine scaling with feature selection.

## 3.1 Benchmark Datasets

The benchmark text classification tasks we use are drawn from Reuters-21578 (re), OHSUMED abstracts (oh), the Los Angeles Times (la), the Foreign Broadcast Information Service (fbis), the Web ACE project (wap), and various TREC competitions (tr, new3), originally compiled by Han and Karypis [7]. See Table 1, which includes the total vocabulary size and the average number of words present per document. There are 19 multi-class datasets and their class sizes vary widely. For these experiments we consider each class vs. all others within that dataset, yielding 237 binary text classification tasks, which are representative of the size and difficulty of many of the industrial text classification tasks we face at HP Labs.

Although the tasks are not entirely independent of each other because of some overlap in their negative classes, they are much more independent than studies that consider only a single source of documents with a common negative class, e.g. using just Reuters. We have made the processed TF feature vectors available for other researchers at the WEKA web site [16].

**Table 1. Benchmark text classification datasets**

| Dataset | Classes | Docs | Words | Words/Doc |
|---|---|---|---|---|
| fbis | 17 | 2463 | 2000 | 160 |
| la1 | 6 | 3204 | 31472 | 151 |
| la2 | 6 | 3075 | 31472 | 148 |
| new3 | 44 | 9558 | 26833 | 235 |
| oh0 | 10 | 1003 | 3182 | 53 |
| oh10 | 10 | 918 | 3012 | 56 |
| oh15 | 10 | 1050 | 3238 | 59 |
| oh5 | 10 | 913 | 3100 | 54 |
| ohscal | 10 | 11162 | 11465 | 60 |
| re0 | 13 | 1504 | 2886 | 52 |
| re1 | 25 | 1657 | 3758 | 53 |
| tr11 | 9 | 414 | 6429 | 282 |
| tr12 | 8 | 313 | 5804 | 274 |
| tr21 | 6 | 336 | 7902 | 470 |
| tr23 | 6 | 204 | 5832 | 385 |
| tr31 | 7 | 927 | 10128 | 269 |
| tr41 | 10 | 878 | 7454 | 195 |
| tr45 | 10 | 690 | 8261 | 281 |
| wap | 20 | 1560 | 8460 | 141 |

## 3.2 Evaluation Measures

We evaluate performance based on two standard performance measures: accuracy for its historical standard in machine learning research and F-measure for its improved sensitivity in the common information retrieval situation where positives are rare. F-measure is the harmonic average of precision & recall, where precision = true positives ÷ predicted positives, and recall = true positives ÷ all positives in ground truth. To achieve good F-measure requires the classifier have good precision *and* good recall on the positive class, whereas good accuracy can be achieved simply by predicting all test cases as negative, if the positive class is rare.

For each of the 237 binary text classification tasks, the performance of the learning algorithm is measured via standard 4-fold stratified cross-validation, repeated with eight different random split seeds (kept consistent for each representation tested). Note that all feature selection and scaling is determined from only the training folds within cross-validation, to avoid leaking information from the test set.

For any given feature representation under study, we obtain 237 x 8 performance measurements. To distill these into one summary number, we use *macro-averaging*, weighting each of the measurements equally (as opposed to *micro-averaging*, which would weight them according to the number of documents that happen be available in each task).

## 3.3 Induction Algorithms

We use the linear SVM implementation provided by the WEKA library v3.4 [16]. In this experiment, we use its default behavior, rather than attempting to optimize each of its many parameters. In particular, its default complexity constant is one. We had to disable its feature space normalization feature, otherwise it re-scales all feature values to the range [0,1]. Thus, BNS feature scaling causes the more important features to have a greater effect on the SVM kernel dot product than less predictive features.

This kernel-centric explanation of the effect is specific to kernel-based classifiers such as the SVM, but we found that the BNS scaling also helps some other classification models that are not based on kernels. We repeated the experiment protocol with the multinomial Naïve Bayes classifier, which has been shown superior to its simple *binomial* form for text classification [11]. (Note that feature scaling would have no effect on the classic binomial formulation, which converts each feature to binary nominal values. The multinomial formulation uses word counts, which BNS scaling effectively increases for the more important features.) We found that while BNS scaling does benefit the multinomial Naïve Bayes classifier, its best performance came from using plain binary features with BNS feature selection, and even then it had substantially worse accuracy than SVM—consistent with much of the literature. Thus, we report the Naïve Bayes results only in a tech report [2]. Henceforth we just consider SVM models.

# 4. EMPIRICAL RESULTS

Initially we focus on the BNS feature scoring metric (the initial impetus for this work) and contrast it with IDF or no scaling, since they are most common. Later in Section 4.5 we compare against many other feature scoring metrics that we considered later.

## 4.1 Accuracy & F-measure

Figure 1 shows the average accuracy and F-measure for six different text representations, including error bars which extend above and below each point by one standard error, stdev÷sqrt(N). Binary features and term frequency (TF) counts are considered for each of three feature scaling methods: IDF, BNS, and no scaling. The two most common text representations, TF·IDF and binary features, both performed significantly worse than BNS scaling of binary features (labeled just 'BNS'). We expected this from the intuitive motivation in the introduction. BNS scaling leverages

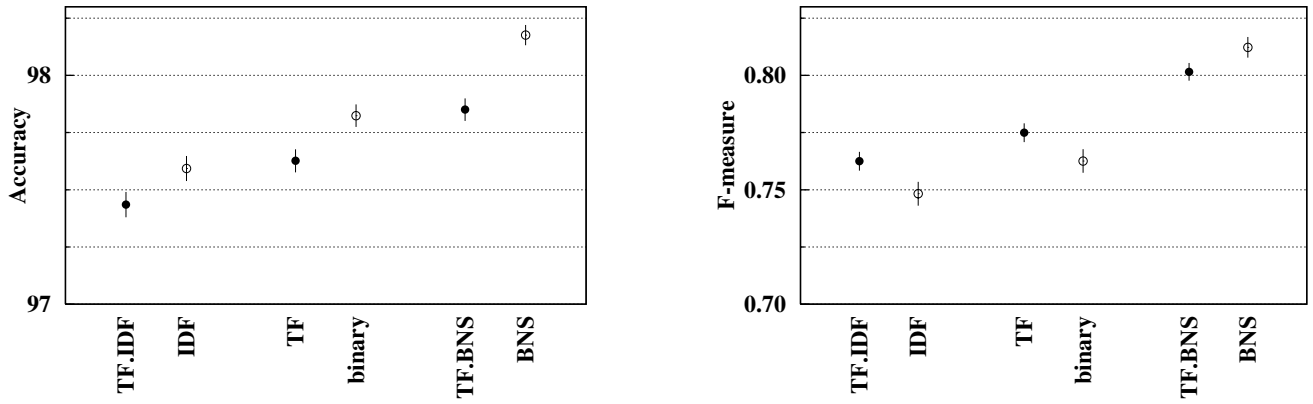


**Figure 1. SVM accuracy (left) and F-measure (right) for six different text feature representations, including standard error bars.**

the class labels to improve the distance evaluations of the SVM kernel. Using binary features, as opposed to term frequencies, consistently improves accuracy; binary features do not appear to have a consistent effect on F-measure, but we elucidate this next. Overall, the BNS-scaled binary features provided a 1% improvement over TF·IDF for accuracy and 7% for F-measure, both with no question of statistical significance by their small error bars in proportion to the differences.

## 4.2 Precision vs. Recall Analysis

Good F-measure requires a balance between precision and recall. It can be informative to see the tradeoffs these different methods make between precision and recall at the classifier's natural classification threshold. (Alternately, one could consider full precision-recall curves by sweeping through all decision thresholds, but this considers many decision points for which the classification algorithm was not attempting to optimize and makes it difficult to compare six methods.) Figure 2 shows Precision vs. Recall for the six methods considered in the previous figure, including standard error bars in both dimensions. The x- and y-axis ranges are identical to make it clear that the SVM precision typically exceeded its recall.

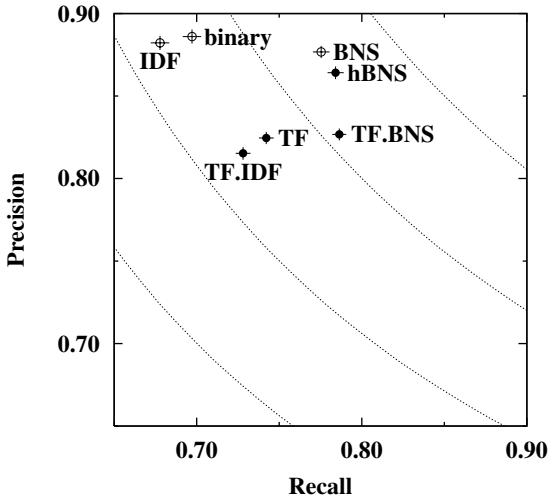


**Figure 2. Precision & Recall for six text representations. The thin dotted lines show isoclines of equal F-measure.**

Here we can see consistent effects. Using binary features yields a large improvement in precision compared with TF features. Likewise, BNS scaling results in a large improvement in recall compared with IDF or no scaling. (We will explain the underlying cause of this in the discussion section, as well as the additional data point labeled 'hBNS.') Together, these two effects make for a substantial increase in F-measure. The dotted curved lines in the background show points of equal F-measure, indicating the gradient to obtain better F-measure performance.

Since the positive class is typically small for text classification, the best improvement in accuracy is by improving precision, not recall—hence, the consistent benefit of binary features on accuracy we observed in Figure 1. (Note: for applications where the best precision is sought regardless of recall, one may prefer instead to optimize the precision in the top N predicted positives.)

## 4.3 The Effect of Class Distribution

The importance of BNS scaling increases as the class distribution becomes more skewed. To illustrate this, we have binned together those benchmark tasks that have between 0% and 5% positive cases, 5%—10% positives, and so forth. See Figure 3, which shows macro-average F-measure for each bin and again includes standard error bars. When the minority class is relatively balanced (20%–25%), the benefit of BNS scaling vs. no scaling or IDF scaling of binary features is small. As the percentage of positives decreases to a small minority, the classification

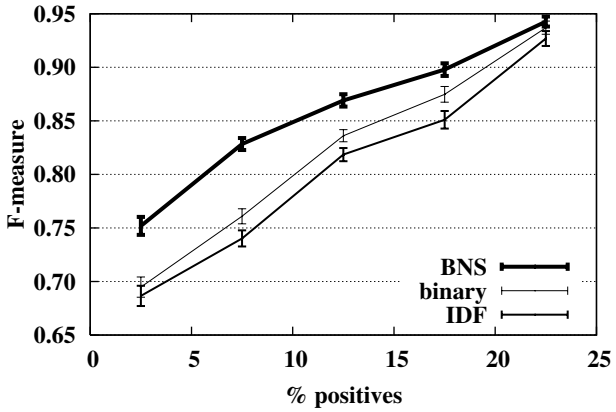


**Figure 3. The benefit of BNS feature scaling grows as the percentage of positive cases becomes small.**

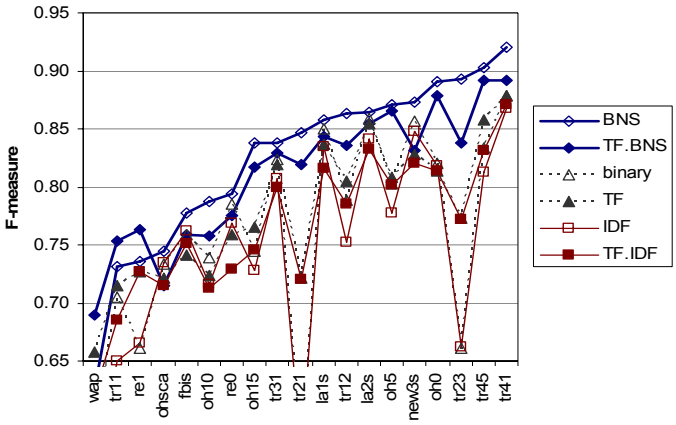


**Figure 4. BNS feature scaling typically dominated other methods for the 19 datasets.**

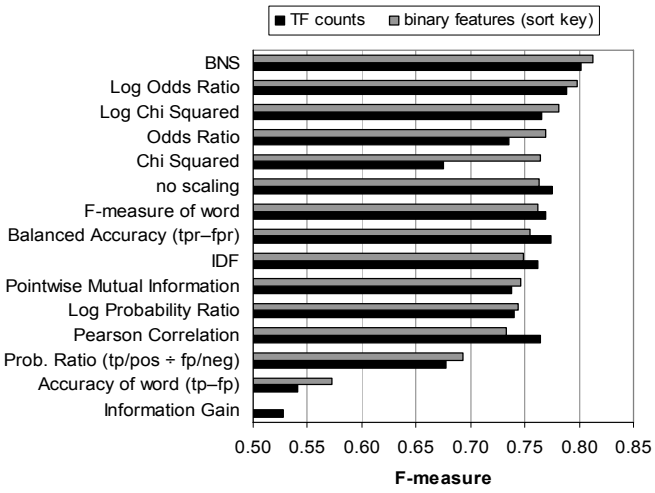


**Figure 5. F-measure for 30 different text representations.**

performance naturally drops for all methods, but the decline is substantially less if BNS scaling is used. In other words, the benefit of BNS scaling is greater for highly imbalanced classification problems. Each successive bin contains roughly half as many benchmark tasks, indicating that the performance under high skew is the more common operating range for text classification problems.

## 4.4 Per-Dataset Views

Next we show the macro-average F-measure for each of the 19 datasets separately. Although the graph is cluttered, Figure 4 shows clearly the near-complete dominance of BNS scaling of binary features (labeled 'BNS'). To aid the visualization, we have sorted the classes by their BNS performance. BNS did not dominate for three of the most difficult datasets, where instead TF·BNS performed best. Note that the performance of some of the methods can be quite erratic, e.g. for datasets tr21 and tr23 the methods IDF and binary performed extraordinarily poorly.

## 4.5 Comparing Many Other Scoring Metrics

At first, we tested the feature scaling idea with only the BNS metric and the Information Gain metric, because these two were the leaders for feature selection in our prior study [4]. But this begs the question of whether some other feature scoring metric might perform yet better. To resolve this question, we tested all the feature selection metrics of our prior study, where their many formulae can be found. Note that asymmetric functions, such as the Probability Ratio [13], have been adjusted to symmetrically value strong features that happen to be correlated with word absence rather than word presence. To these we added Pearson Correlation and Pointwise Mutual Information [10]. Altogether, fifteen choices for feature scaling (all supervised except for IDF and 'no scaling') are paired both with TF counts and separately with binary features in Figure 5. Although some future metric may be devised that exceeds BNS, the present evidence indicates that BNS with binary features is statistically significantly better—it passes a paired two-tailed t-test against the runner-up with $p<1\%$ (even at N=75 trials, for readers who may not consider the benchmark problems as independent). Many of the methods, including IDF, performed worse than using no scaling.

We note that the runner up was Log Odds Ratio. (We included its formula in Section 2.3, partly because it is extremely easy for researchers and practitioners to adopt in their code, whereas BNS requires statistical tables not available in common software libraries.) Our prior study showed that Odds Ratio has a preference surface that is similar to BNS. For feature selection, the dynamic range of the scoring metric is immaterial—only the relative ranking of features is important. But here, we find that the logarithm of Odds Ratio did a better job of conditioning the feature vectors for the linear SVM kernel than using raw Odds Ratio, which has a very large range. Similarly for Chi-Squared and Probability Ratio.[1]

## 4.6 Feature Selection & Scaling Combined

Finally, we consider the question of whether performance might further be improved by its use in conjunction with feature selection, since feature selection has been shown beneficial, esp. with BNS and IG [4]. In Figure 6 we vary the number of features selected, with and without BNS scaling on binary features (all performance figures were worse for TF features, so they are not shown in order to reduce visual clutter). The rightmost point on the logarithmic x-axis is not to scale, but indicates that all features were used within each dataset, i.e. no feature selection. The overall best performance for both accuracy and F-measure is obtained using BNS scaling with all features. Without BNS scaling, the best available performance is obtained by selecting a subset of features via BNS. (The lack of improvement with IG feature selection and certain other methods has led some researchers in the past to conclude that SVM does not benefit from feature selection.)

With the improvement of BNS feature scaling, we observe that feature selection by either method only produced a degradation in accuracy and F-measure. Interestingly, the least degradation occurs with a hybrid where BNS is used for scaling the features

[1] Whereas most experiment points were evaluated in seconds or minutes, an occasional classification task paired with Odds Ratio, Probability Ratio or Chi-Squared (each without the logarithm) took many hours to train. We eventually terminated these, and exclude them from the average. We suppose their huge dynamic range made a highly discontinuous search space for the SVM optimization.

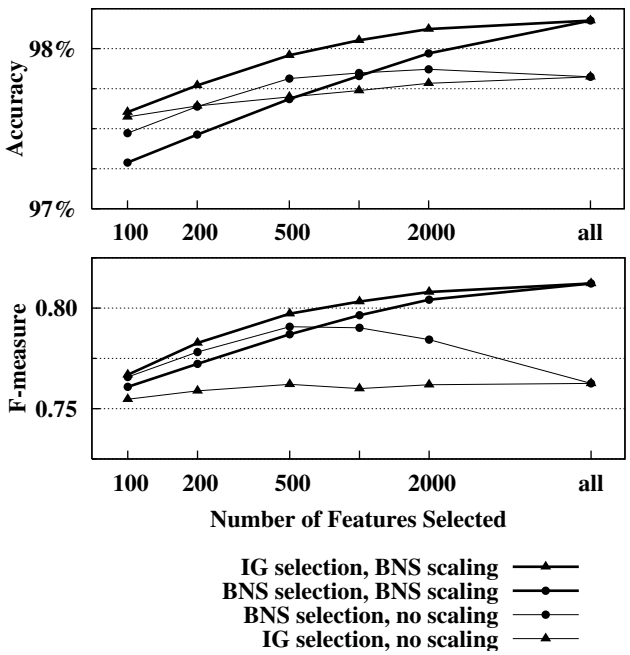


**Figure 6. Accuracy (top) and F-measure (bottom) as we vary the number of features selected by BNS or IG. Binary fx.**

but IG is used for selecting the features. These results also prove that the reason feature scaling works is not because it simply forces the worst features to zero scaling; if it were so, then BNS feature selection would show similar behavior with and without BNS scaling.

Note that this is an average result—it may certainly be that for some datasets a smaller number of features can improve performance. Nonetheless, for a data miner looking to simplify their process or the number of knobs that have to be set on their software, simply using all features with BNS scaling is likely to be a good choice. This is also important when the size of the available training set is small. Trying to determine the best number of features to select by many trials of cross-validation on a very small dataset is likely to result in overfitting to the training set, rather than truly tuning the parameter to reduce the generalization error on unseen cases.

# 5. Discussion

Given that there are so many papers on feature selection that attempt to discard 'useless' features that are unpredictive, extremely rare, or redundant, why do we find the best performance uses all the features? We offer this intuition for text classification, at least for topic identification. Given a topic of interest, it is highly improbable that a random word from the dictionary, say 'fee,' is equally common within the topic as well as in the negative class—this would happen only for a relatively small set of stopwords or functional words. As for redundancy, only if a set of words are *perfectly* correlated—always appearing together or not at all—do they add no information beyond a single member of the set. We know that many uncorrelated weak predictors can combine to form a strong predictor, e.g. boosting.

Although we advocate using BNS scaling with all features in general, there are certainly classification tasks that are best performed with only a few. For example, Gabilovich and Markovitch [6] studied a set of tasks where the topic is defined by only a few keywords, such as 'Denver, Colorado.' For such tasks, there are many words in the dictionary that are equally likely in and out of the topic. Such words truly add nothing, and they find a benefit in accuracy by removing them.

We acknowledge there are situations where training with the complete vocabulary is out of the question with respect to modern computational resources. In these cases, the hybrid method may do nicely in cutting computational cost while minimizing loss in performance. That said, we suspect the need arises rarely in actual practice. Even for a corpus with a huge vocabulary, the words appearing in the training set are likely not too great—the more common problem is that the training set is small. And for a classification task of economic worth, one can often afford to train for a week if the improved accuracy pays for the relatively inexpensive machine cycles. The greatest need to trim computation may be in bioinformatics, or perhaps in machine learning research, where we perform hundreds of analyses and cross-validations under conference deadlines.

## 5.1 An Example Task Illustrated

Let's consider a specific binary classification task: oh10 class 6 (OHSUMED class 'Italy'), which has 60 positives against 990 negatives—5.7% positives. On this task, BNS scaling with binary features obtains 0.77 F-measure, whereas TF·IDF gets 0.54 on average. From Table 1, we see that the oh10 documents tend to be short, so term frequencies will tend to be small and unlikely better than just binary feature values.

Figure 7 shows the distribution of the specific words in the tp vs. fp space. As is common in text classification, there are a few words that occur commonly in both classes (e.g. 'patient' and 'result'), and many words that occur rarely (unlabeled and overlapping points near the origin). Observe that the positive class has two highly predictive words: 'italy' and 'italian.'

The diagonal grid lines show the isoclines of equal IDF value, with the points nearest the origin being weighted highest. Unfortunately, such points are valued by IDF much more highly than 'italy' or 'italian.' These essential words are ranked equally with a plethora of non-predictive words that occur in ~3% of positive cases as well as ~3% of negative cases.

Figure 8 shows the same, but here the curved grid lines show the isoclines of equal BNS value, with the greatest values given to the top left and bottom right corners, naturally. As desired, the words 'italy' and 'italian' are given the highest BNS weights. Likewise, a fairly high weight is given to the word 'cost' on the x-axis, which occurred in 130 negative cases (13% fpr) and zero positive cases. Our previous study on feature selection showed that BNS's strength is in leveraging many frequent words in this region to improve recall, especially when positives are scarce. Referring back to Figure 7, we see that IDF gives a much lower weight to these negative features, precisely because they do occur more frequently. Now, consider what happens when the positive class becomes increasingly rarer: IDF will further prefer positive features because they are rare, and the region of good 'recall' features along the x-axis is downplayed. Thus, we can see why BNS is increasingly better than IDF for greater class imbalance, as observed in Section 4.3.

With this understanding about positively and negatively correlated features, we consider again Figure 2 where we saw that binary features improved precision and BNS scaling improved recall. The best F-measure was obtained by using binary features

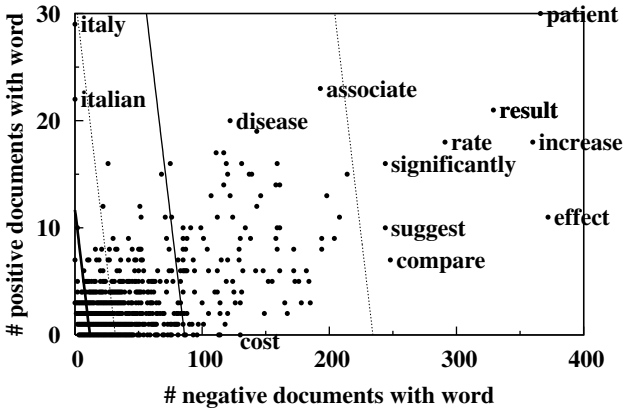


**Figure 7. IDF evaluation of word features for oh10 class 6.**

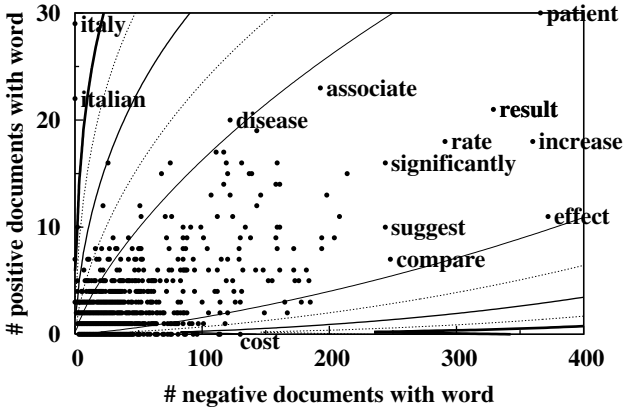


**Figure 8. BNS evaluation of same.**

with BNS scaling. But since recall was slightly better with TF.BNS features, perhaps we could further improve performance by developing a hybrid method that uses TF.BNS for negatively correlated features and binary features with BNS scaling for positive features. That is, for each word of the training set, we determine whether it is positively or negatively correlated with the positive class, and we select a binary representation for positively correlated features, otherwise its term frequency TF count is used; this integer is then scaled via the BNS evaluation of the feature. This hybrid method, labeled 'hBNS' in Figure 2, shows recall as good as the best method, but a slight decrease in precision from BNS. Overall, its F-measure was insignificantly better than just using binary features with BNS scaling. Of course, when the goal is accuracy, or when precision is preferred to recall, then it is better to simply use all binary features.

## 5.2 Non-Language Features

The techniques that have been developed for text classification are used for all sorts of unstructured domain problems, not just natural language text. Hence, for some tasks, there will be features that occur with very high probability in one class or the other, making the IDF assumption even less pertinent. For example, in classifying technical support documents, sometimes there are unique strings or identifiers that can be a substantial aid to the classifier, e.g. the word or n-gram term 'OS=WinXP' appearing in a technical document is an extremely strong indicator for an MSWindows category.

Furthermore, even traditional natural language topic identification tasks are beginning to have additional features included that do not follow the Zipfian properties of regular words. For example, Gabrilovich and Markovitch augment the feature vector with the opinion of many topical classifiers previously trained on freely available training sets [5]. Likewise, we have developed a technique to cope with concept drift that augments the feature vector with the opinions of many previous classifiers for the same source of data [3]. In fact, it was this purpose that originally motivated BNS scaling, since otherwise strongly predictive TIX features were not highly influential on the final classifier. In the extreme, when the true class label was revealed as a pseudo-word feature in an experiment on Reuters2000 ECAT classification, an SVM with binary features achieved only 0.22 F-measure, but 0.78 with BNS scaling of those features.

## 5.3 Practical Implementation

On a practical note for implementers, we find the inverse Normal function is not available in standard math libraries for Java or C++. It is available in Gnuplot as invnorm(), and in Microsoft Excel as NORMINV(), so we easily constructed a large lookup table to evaluate the function in Java. We included the formula for the runner-up method, Log Odds Ratio, because it is extremely simple to program without statistical tables.

For our research harness using WEKA, each feature is represented by a floating-point double, which occupies 8 bytes of memory. This is flexible and allows each feature to be scaled independently. But since the best results reported indicate that binary features are optimal, each feature can be reduced to a single bit of storage (64x savings), and the feature scaling parameters can be kept in a array of floating-point numbers that grows only with the number of features, not with the number of cases in the dataset. These scaling factors would be taken into account when computing the SVM kernel. This memory savings could be important for unusually large problems, such as training on all 800,000 cases of Reuters 2000 with all features. Such a feat is made practical by Joachim's new SVM training algorithm whose runtime scales linearly for sparse datasets [8].

# 6. CONCLUSION

We have presented a new method that showed a substantial gain in performance for SVM on a large—albeit specific—benchmark of text classification problems. Data mining practitioners may welcome it because it showed consistent, nontrivial gains, and moreover because it is very easy to implement compared to many reported machine learning advances, such as boosting or iterative SVM feature elimination.

Future work potentially includes searching for superior scaling functions besides BNS, combining scaling with other induction algorithms, applying scaling for 1-of-m multi-class classification, and testing on other benchmark datasets, such as the new Reuters corpus or genomic data. In particular, it would be interesting to test and extend these ideas on a benchmark corpus of long documents, where word counts may play a more significant role. Finally, given enough computer time, one could also optimize the SVM parameters to maximize F-measure under each test condition.

# 7. ACKNOWLEDGMENTS

This work benefited from discussions with Bin Zhang and Ira Cohen. We appreciate the large number of compute cycles made

available by Eric Anderson and the HP Labs Utility Datacenter. We are grateful also to Ian Witten and his team for the WEKA machine learning library [16], which greatly facilitates research.

## 8. REFERENCES

[1] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. Inductive Learning Algorithms and Representations for Text Categorization. In *Proc. of the 17th Intl. Conf. on Information and Knowledge Management* (CIKM, Maryland):148-155, 1998.

[2] Forman, G. BNS Scaling: A Complement to Feature Selection for SVM Text Classification. Hewlett-Packard Labs Tech Report HPL-2006-19, 2006.

[3] Forman, G. Tackling concept drift by temporal inductive transfer. In *Proc. of the 29th Int'l ACM Conf. on Research and Development in Information Retrieval* (SIGIR, Seattle):252-259, 2006.

[4] Forman, G. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Special Issue on Variable and Feature Selection, Journal of Machine Learning Research*, 3(Mar):1289-1305, 2003.

[5] Gabrilovich, E., and Markovitch, S. Feature Generation for Text Categorization Using World Knowledge. In *Proc. of the 19th Intl. Joint Conference for Artificial Intelligence* (IJCAI, Edinburgh), 2005.

[6] Gabilovich, E. and Markovitch, S. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5. In *Proc. of the 21st Intl. Conf. on Machine Learning* (ICML), 2004.

[7] Han, E. and Karypis, G. Centroid-Based Document Classification: Analysis & Experimental Results. In *Proc. of the 4th European Conf. on the Principles of Data Mining and Knowledge Discovery* (PKDD): 424-431, 2000.

[8] Joachims, T. Training Linear SVMs in Linear Time. In *Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (KDD, Philly):217–226, 2006.

[9] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of the 10th European Conf. on Machine Learning* (ECML, Berlin):137-142, 1998.

[10] Manning, C. and Schütze, H. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[11] McCallum, A. and Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. Workshop on Learning for Text Categorization, In *the 15th National Conf. on Artificial Intelligence* (AAAI), 1998.

[12] Mladenic, D., Brank, J., Grobelnik, M. and Milic-Frayling, N. Feature selection using linear classifier weights: interaction with classification models. In *Proc. of the 27th ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR,Sheffield):234-241, 2004.

[13] Mladenic, D. and Grobelnik, M. Feature Selection for Unbalanced Class Distribution and Naïve Bayes. In *Proc. of the 16th Intl. Conf. on Machine Learning* (ICML):258-267, 1999.

[14] Rogati, M. and Yang, Y. High-performing feature selection for text classification. In *Proc. of the 11th Intl. Conf. on Information and Knowledge Management* (CIKM, Virginia):659-661, 2002.

[15] Wettschereck, D., Aha, D. W., and Mohri, T. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review 11, 1-5, 273-314, 1997.

[16] Witten, I. and Frank, E., *Data mining: Practical machine learning tools and techniques (2nd edition),* Morgan Kaufmann, 2005. http://www.cs.waikato.ac.nz/~ml/weka

[17] Yang, Y. and Pedersen, J. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of the Intl. Conf. on Machine Learning* (ICML):412-420, 1997.