Quantifying Trends Accurately Despite Classifier Error and Class Imbalance

George Forman
Hewlett-Packard Labs
1501 Page Mill Road
Palo Alto, CA 94304
ghforman@hpl.hp.com

ABSTRACT

This paper promotes a new task for supervised machine learning research: *quantification*—the pursuit of learning methods for accurately estimating the class distribution of a test set, with no concern for predictions on individual cases. A variant for *cost quantification* addresses the need to total up costs according to categories predicted by imperfect classifiers. These tasks cover a large and important family of applications that measure trends over time.

The paper establishes a research methodology, and uses it to evaluate several proposed methods that involve selecting the classification threshold in a way that would spoil the accuracy of individual classifications. In empirical tests, Median Sweep methods show outstanding ability to estimate the class distribution, despite wide disparity in testing and training conditions. The paper addresses shifting class priors and costs, but not concept drift in general.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: *decision support.*
I.5 [**Pattern Recognition**]: Design Methodology, *classifier design and evaluation*

General Terms

Algorithms, Measurement, Design.

Keywords

classification, quantification, cost quantification, text mining.

1. INTRODUCTION

Tracking trends over time constitutes a very large family of business and scientific applications, e.g. monitoring the prevalence of hepatitis B. If the cases are labeled accurately, either by humans or by supervised machine learning, then a simple histogram of the class labels gives accurate counts. But commonly such classifiers have some degree of error that leads, perhaps surprisingly, to a large and *systematic* bias in the counts.

There is a tremendous literature in machine learning that focuses on optimizing the correctness of individual predictions (accuracy, F-measure, etc.). While useful, it is not sufficient for this large family of applications whose objective is different: accurate estimation of the histogram counts. Interestingly, it does not matter that the count may involve both false positives and false negatives, as long as they balance one another well to cancel each other out. This brings about two challenging and valuable tasks for ongoing research:

1. The *quantification* task for machine learning: given a labeled training set, induce a *quantifier* that takes an unlabeled test set as input and returns its best estimate of the class distribution.

At first, this task may seem almost trivial, but experience proves otherwise. The ubiquitous practices of random sampling and cross-validation in machine learning research completely hide the problem, since the training class distribution is chosen to match that of testing. This has perhaps suppressed recognition of this important research setting, as well as its more complicated testing methodology, which we address in Section 3 to evaluate the quantification methods proposed in Section 2.

2. The *cost quantification* task for machine learning: given a labeled training set, induce a *cost quantifier* that takes an unlabeled test set as input and returns its best estimate of the total cost associated with each class, as determined by a cost attribute on each record (which in some settings may have missing values).

For example, consider the task of estimating the total time spent (labor cost) by technical support agents dealing with calls related to problem X each month. One solution is to train a binary

Table 1. Parameters varied in the experimental comparison

P = 10...100	Positives in training set
N = 100...1000	Negatives in training set
p = 1...95%	Percent positives in *test* set
Benchmark =	25 binary text classification tasks, x 10 splits
Learning Algorithms:	
SVM	linear Support Vector Machine
NB	multinomial Naive Bayes
Performance Metrics:	
Abs.Err	|estimated p – actual p|
Bias	estimated p – actual p
CE	normalized Cross-Entropy
Methods:	
CC	Classify & Count
AC	Adjusted CC, plus variants with selected thresholds
MS	Median Sweep of AC at all thresholds
MM	Mixture Model

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

classifier to recognize problem X, and for each monthly batch of call logs, return the sum of the cost values of all cases predicted positive. Partly due to reasons stated above, this is a poor approach. We address this and propose methods in Section 6.3.

A flip-side benefit of quantification technology is that to obtain an equivalent accuracy of the count, a much less accurate classifier can be used. This enables some applications of machine learning where otherwise its classification accuracy would be unacceptable. This can also result in savings in the labor cost to develop labeled datasets to train classifiers. At first blush, this savings may not seem substantial. However, at Hewlett-Packard we train thousands of classifiers to analyze technical support logs to track the many different types of support issues that arise for our many different product lines [3]. As a concrete example: monitoring for an increase in the incidence rate of cracked screens on HP iPAQ handheld products. Further, the training needs are ongoing because of concept drift, as well as the introduction of product lines and new support issues. So labor savings from quantification technology continues to accumulate over time.

We finish this section with a discussion of related work, before we introduce and then test methods for quantification in the subsequent sections. In this paper, without much loss of generality, we primarily focus on two-class tasks in order to simplify the exposition and resolve a key sub-problem for the multi-class setting (Section 6.2). In this binary setting we can speak of estimating the number of *positives* in a test set.

1.1 Related Work

Most supervised machine learning research attempts to optimize the correctness of individual classifications in one way or another. Hence, each classification can be judged independently, and the success of a method can be judged by its accuracy, error rate or F-measure averaged over a large benchmark of tasks. In contrast, quantification produces a *single* output for a whole batch of items. These outputs must be aggregated over many batches in a large benchmark to evaluate methods. Hence, the research methodology is unusual.

It bears a superficial resemblance to the batching used in research for probability estimating classifiers. For example, if a meteorologist predicts the chance of rain at 20% on certain days of the year, we would like it to rain on 20% of those days. The correctness of a prediction on a single day cannot be judged. Some methods even require examination of the entire test set before producing any output. However, probability estimation, like traditional classification, continues to make individual predictions on each item, and judge them in aggregate. By contrast, quantification makes a single prediction based on an entire batch—a single scalar for two-class tasks. This batching requirement calls for a different research methodology (section 3).

Intuitively, not having to make individual predictions should make the estimation task easier. An insurance company can estimate how many cars will have accidents next year, but cannot predict which ones. The nature of the uncertainty is shifted from the individual cases to the aggregate count.

Regarding probability estimation: one obvious idea for a quantification method is to induce a classifier that outputs calibrated probability estimates, and then to sum these probabilities over the test set to estimate the count for each class. This has an intuitive advantage over simply counting discrete predictions made by a traditional classifier, which loses information about the uncertainty of individual predictions. Nonetheless, this obvious method is ill-posed: the calibration depends critically on the class distribution of the *training* set, which does not generally match that of the test set in quantification (cf. it always matches under cross-validation).

Estimating the class distribution of a target dataset is not new. But existing work in machine learning estimates the test class distribution *in order to* adjust the classification threshold [e.g. 1,9,10]. Again, the objective metric in such research has been the correctness of the individual classifications. To our knowledge, ours is the first work to empirically compare and determine machine learning methods that excel in estimating the class distribution. This paper extends our recent publication [4] with superior methods, as well as a more focused experiment protocol.

Of course, once accurate and robust methods are established for estimating the distribution, they can be used as a subroutine for the traditional purposes of calibrating probability estimating classifiers, or optimizing the classification decision threshold to minimize cost, e.g. in ROC analysis [1].

As a side note, there is *unsupervised* work in tracking shifting topic distributions [e.g. 7,8]. It naturally has uncalibrated cluster boundaries, having no bearing on *supervised* quantification.

2. QUANTIFICATION METHODS

As a strawman method, consider simply learning a state-of-the-art binary classifier from the training set, and counting the number of items of the test set for which it predicts positive. We call this simple method Classify & Count (CC). The observed count of positives from the classifier will include true positives TP and false positives FP. Ideally, we would like to adjust the observed count somehow for the false positives and false negatives. By the following characterization, we derive such a quantifier, the Adjusted Count (AC) method [4]:

	Classifier Prediction:	
Actual Class:	Pos	Neg
Positives	TP = *tpr* * Positives	FN
Negatives	FP = *fpr* * Negatives = *fpr* * (total - Positives)	TN

where *tpr* is the *true positive rate* of the classifier, P(predict + | actual pos), and *fpr* is its *false positive rate*, P(predict + | actual neg). The observed count of positives is then:

$$\text{Pos} = tpr * \text{Positives} + fpr * (\text{total} - \text{Positives})$$
$$= (tpr - fpr) * \text{Positives} + fpr * \text{total}$$

Solving for the actual number of Positives, we get:

$$\text{Positives} = (\text{Pos} - fpr * \text{total}) \;/\; (tpr - fpr)$$

Finally, dividing both sides by the total, we express the equation in terms of percentages of positives:

$$\text{adjusted estimate } p' = \frac{(\textit{observed \% positives}) - fpr}{tpr - fpr} \qquad (1)$$

It remains only to estimate the *fpr* and *tpr* characteristics of the classifier, which is accomplished via standard cross-validation on the training set. These characteristics are independent of the training class distribution because they treat positives and negatives separately. Since these estimates may deviate somewhat

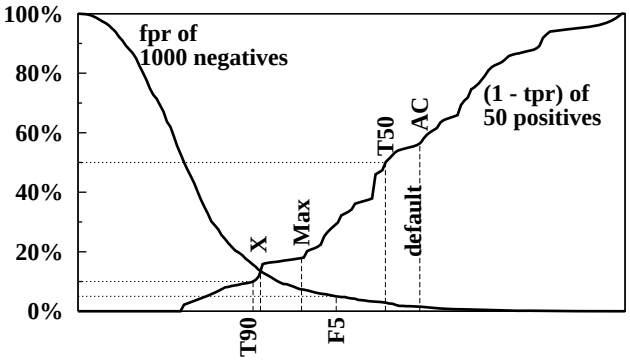

Figure 1. Various threshold selection policies.

from the actual *tpr*,*fpr* rates in testing, we must clip the output of formula (1) to the range 0% to 100% in order to avoid occasional infeasible estimates. In summary, the AC method trains a binary classifier, estimates its *tpr*,*fpr* characteristics via cross-validation on the training set, and then when applied to a given test set, adjusts its quantification estimate by formula (1).

2.1 Class Imbalance Problem

The AC method estimates the true class distribution well in many situations, but its performance degrades severely if the training class distribution is highly imbalanced (e.g. Figure 8 in [4]). For example, with P=50 positive training cases and N=1000 negative training cases, the induced classifier is very conservative about voting positive. If the positive class is rare enough, it will simply vote negative always, i.e. tpr=0%. While this may result in optimal *classification accuracy* (for the distribution in the training set), it is useless for estimating the distribution of the test set. Backing off from this extreme, consider a classifier that is very conservative but not entirely negative. Its true positive rate *tpr* would be very low, and its false positive rate *fpr* would be non-zero. This forces the denominator (*tpr* – *fpr*) of formula (1) to a small range, making the quotient highly sensitive to any error in the estimate of *tpr* and *fpr*, giving bad estimates under imbalance.

Unfortunately, high class imbalance is pandemic, esp. to our business applications, and so the need to operate well under these conditions is important. A well known workaround is simply to disregard many cases in the majority class of the training set to bring it back into balance. But this throws away information. We can do better. Having investigated the reason for the degradation, we have devised new methods that are resilient under class imbalance and take advantage of any surfeit of negative training cases, which are often freely available.

2.2 Imbalance Tolerant Methods

The solution to the class imbalance problem lies in recognizing that accuracy on individual classifications is not important to our objective. We will select a different decision threshold for the classifier that will provide better estimates via formula (1), although the threshold may be completely inappropriate for maximizing classification accuracy. Specifically, we will admit many more false positives to avoid a threshold in the tails of the curve, where estimates of *tpr*,*fpr* are poorer.

The remaining question is what policy to use for selecting a threshold. To consider the possibilities, we use the illustration in Figure 1. The x-axis represents the spectrum of thresholds, i.e. the scores generated by the raw classifier. They are uncalibrated and may take any range, e.g. the probability output by Naïve Bayes, which is notorious for its poor probability calibration, or the signed distance from the separating hyperplane of an SVM (calibrating the SVM output via a fitted logistic regression model would have no effect on the methods, since they do not use the magnitude of the x-axis except as a decision threshold).

The descending curve shows the false positive rate *fpr* and the ascending curve shows the inverse of the true positive rate (*false negative rate* = 1 – *tpr*). The inversion of *tpr* is visually useful to see the tradeoff with *fpr*. For a perfect classifier, there would be perfect separation between these two curves (this never occurred in our benchmark tasks). These example curves represent an SVM classifier whose natural threshold of zero delivers 92% classification accuracy for an imbalanced training set having 50 positives and 1000 negatives. Because negatives abound, SVM naturally optimized for a very low false positive rate, even at the cost of a 'few' misclassified positives (28 of 50). This explains its poor F-measure of 50%.

The basic AC method uses the classifier's default threshold, which will be far in the *fpr* tail if positives are rare in training. An intuitively better threshold policy is where the two curves cross, where fpr = 1-tpr (labeled X in Figure 1). This "X" method nicely avoids the tails of both curves. Considering the earlier discussion of small denominators, another likely policy is where the denominator is maximized: method Max = argmax(tpr-fpr). More traditionally, a Neyman-Pearson criterion would select the threshold at a particular true positive rate (method T90 = 90%, T50 = 50%), or false positive rate (F5 = 5%). We tested all these threshold policies and others.

2.3 Median Sweep (MS)

All the threshold selection methods above run the risk that the *tpr*,*fpr* estimates from cross-validation *at their chosen threshold* do not happen to match the actual rates encountered on the test set. For this reason, we consider an additional approach: obtain an estimate at *every* threshold, and return a mean or median of these estimates. Median is preferred, as it is less sensitive to outliers. Specifically, the Median Sweep (MS) method computes the distribution estimate via formula (1) for *all* thresholds, and returns the median. Considering the earlier discussion about high sensitivity when the denominator becomes small, we also evaluate a variant (MS2) that considers only thresholds where the denominator (*tpr*-*fpr*) is greater than ¼.

2.4 Mixture Model (MM)

For comparison, we also include a robust quantification method from our earlier work, which is based on completely different principles. Due to space limitations, we cannot describe the method fully here, but we refer readers to [4]. In short, it models the distribution of raw classifier scores generated on the test set as a mixture of two distributions: the classifier scores on the training negatives, and those on the training positives—as determined by cross-validation. An optimization step determines the mixture that results in the best fit, and then it returns this as the estimated % positive. This method is surprisingly robust with even as few as 10 training positives (!). While it does not apparently suffer from the class imbalance problem, we shall see that the Median Sweep methods often surpass it.

3. EXPERIMENT METHODOLOGY

As mentioned in the introduction, quantification research necessitates a substantially different experiment methodology. In particular, the class distribution must be varied independently and dramatically between the training and testing sets. Further, since each batch produces only a single estimate, we must test on many batches and aggregate measurements to identify trends. The ideal quantification method will generate accurate estimates, despite wide variation in training and testing conditions.

To vary the *training* conditions, we randomly select P=10…100 positive training cases, and N=100 or 1000 negative training cases from the benchmark dataset at hand. These sizes are selected to cover common operating ranges of interest to our business applications, and are reasonable for many other situations. The larger number of negatives represents a common multi-class case where we consider one class at a time against many others that each has 10…100 example cases.

To vary the *testing* conditions, we select as many positives and negatives from the remaining benchmark dataset such that the percent positives matches our target p. For a specific dataset having 864 positives and 10298 negatives, we first take 100 positives and 1000 negatives out for training (subsetting these for varied training situations), leaving 764 positives and 9298 negatives. When targeting p=20% positive, we test with all 764 positives and a random subset of 3056 negatives; for p=1%, we use a random subset of 94 positives against all 9298 negatives. Our business datasets often have >100,000 cases to quantify, but are not publishable and rarely labeled with ground truth.

Our prior study measured performance on test sets ranging from p=5%…95% positive, stepping by 5%. While reasonable scientifically, this does not focus on the area of interest for the business problems we face: 1…20% positives is a more challenging and more important range in which to estimate well. Furthermore, we speculate that this range is preferable to study in general. Four loose arguments: (a) For >50% positive, one might simply reverse the meaning of positive and negative. (b) A common type of quantification task has many mutually exclusive classes, therefore most classes are moderately rare in order to sum to 100%. (c) For ~20…80% positive, class imbalance is not a problem, and classifiers tend to operate better in this region. (d) Finally, to get tight confidence intervals when estimating the percent positive, e.g. by manual testing, many more cases must be examined if positives are rare—so, the labor savings of automatic quantification is much greater in the tails.

3.1 Error Metrics

A natural error metric is the estimated percent positives minus the actual percent positives. By averaging across conditions, we can determine whether a method has a positive or negative bias. But even a method that guesses 5 percentage points too high or too low equally often will have zero bias. For this reason, *absolute* error is a more useful measure. But it is unsatisfactory in this way: estimating 41% when the ground truth is 45% is not nearly as 'bad' as estimating 1% when the ground truth is 5%. For this reason, cross-entropy is often used as an error measure. To be able to average across different test class distributions, however, it needs to be normalized so that a perfect estimate always yields zero error. Hence, we use normalized cross-entropy, defined as:

$$\text{normCE}(p,q) = \text{CE}(p,q) - \text{CE}(p,p) \quad (2)$$

$$\text{CE}(p,q) = -p \log_2(q) - (1-p) \log_2(1-q)$$

Table 2. Benchmark classification tasks.

#	Dataset	Class	Positives	Negatives	Total
1	fbis	3	387	2076	2463
2	fbis	7	506	1957	2463
3	fbis	10	358	2105	2463
4	la1	0	354	2850	3204
5	la1	1	555	2649	3204
6	la1	2	341	2863	3204
7	la1	3	943	2261	3204
8	la1	4	273	2931	3204
9	la1	5	738	2466	3204
10	la2	0	375	2700	3075
11	la2	1	487	2588	3075
12	la2	2	301	2774	3075
13	la2	3	905	2170	3075
14	la2	4	248	2827	3075
15	la2	5	759	2316	3075
16	ohscal	0	1159	10003	11162
17	ohscal	1	709	10453	11162
18	ohscal	2	764	10398	11162
19	ohscal	3	1001	10161	11162
20	ohscal	4	864	10298	11162
21	ohscal	5	1621	9541	11162
22	ohscal	6	1037	10125	11162
23	ohscal	7	1297	9865	11162
24	ohscal	8	1450	9712	11162
25	ohscal	9	1260	9902	11162

where q is the estimate of the actual percent positives p in testing. Since cross-entropy goes to infinity as q goes to 0% or 100%, we back off any estimate in these situations by half a count out of the entire test set. Matching our intuition, this back-off will increasingly penalize a method for estimating zero positives for larger test sets: it is worse to mistakenly estimate zero positives among thousands of test cases than among ten.

3.2 Datasets

The benchmark text classification tasks are drawn from OHSUMED abstracts (ohscal), the Los Angeles Times (la), and the Foreign Broadcast Information Service (fbis) [6]; the feature vectors are publicly available for download from the Journal of Machine Learning Research [5]. See Table 2. For this suite of experiments, we consider the binary classification task of one class vs. all others. Only 25 of the 229 potential binary tasks suited our needs, because we required a large number of positives and negatives for this study. (Our prior study went up to 2000 training negatives, but this depletes the supply of negatives for testing, leaving only 21 suitable binary tasks.) F-measure for these tasks averages in the mid-70's.

3.3 Learning Algorithms

We use the linear Support Vector Machine (SVM) implementation provided by the WEKA library v3.4 [11]. We also repeated the experiment with the multinomial Naïve Bayes classifier, which has respectable performance in the text domain.

The adjusted count methods and the mixture model all require cross-validation on the *training* set to generate the distribution of scores for positives and negatives, in order to characterize *tpr* and *fpr*. We chose 50-fold stratified cross-validation. (Note that if there are fewer than 50 positives in the training set, it feels

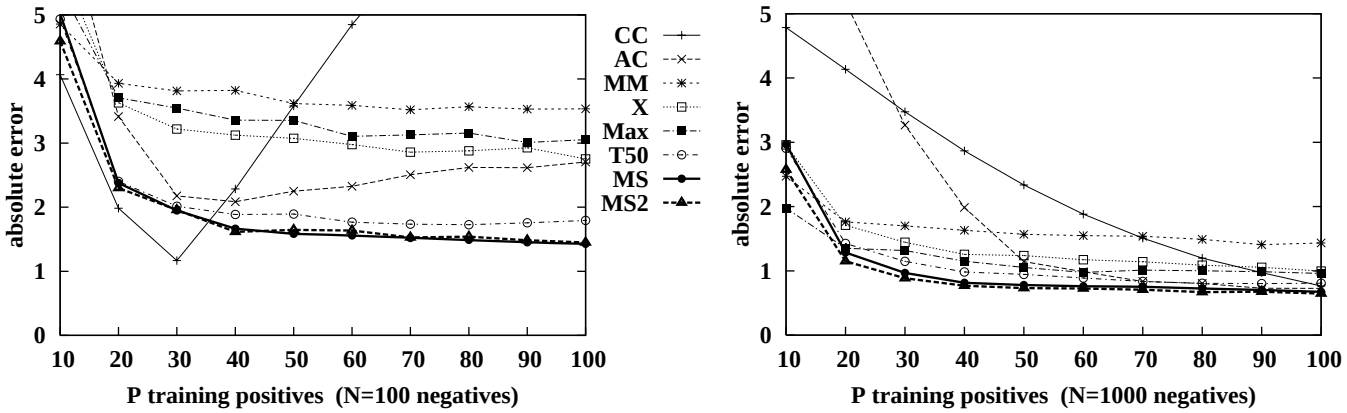

Figure 2. Absolute error for target p=5% positives only.

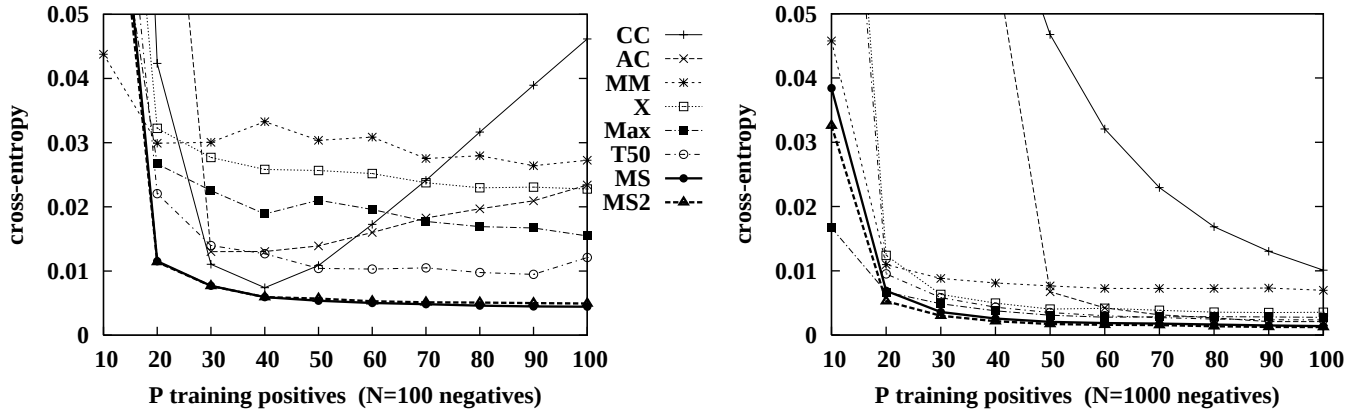

Figure 3. Cross-entropy for targets p=1…20% averaged.

undesirable that some test folds will contain no positives. But we found only degradation trying 10- or min(50,P,N)-folds instead.)

A great deal of computer time can be saved by sharing the computation that is common to all methods: (a) the 50-fold cross validation on the training set, (b) training the final classifier on the training set, and (c) applying that classifier to the test set to generate scores for the positives and for the negatives. Different subsets of these scores can then be used to evaluate various quantification methods under different test class distributions. Sharing steps 1–3 reduced the overall computational load of this experiment by a factor of ~200. As it was, the experiment consumed several days on hundreds of fast 64-bit CPUs in the HP Labs Utility Datacenter. The complete experiment protocol is listed in pseudo-code in the tech report version of this paper.

4. EXPERIMENT RESULTS

Given the large number of parameters in this experiment, we break down the results into sections where we hold some conditions constant as we vary others. For each figure, we will have a pair of graphs: N=100 training negatives on the left, and N=1000 training negatives on the right; we take care that the y-axis range is identical for easy comparison. Every data point represents an average performance over the 25 benchmark text classification tasks times 10 random splits. Except where stated otherwise, we focus on the SVM base classifier.

4.1 Varied training, fixed target p=5%

We begin by examining how resilient the various quantification methods are to wide variations in the training set, while we hold the *test* conditions fixed at p=5% positive. Figure 2 shows the accuracy of the quantifications, as measured by absolute error from the 5% target. Overall we see the Median Sweep methods dominate (MS and MS2, bold lines). Note the absolute scale: the MS methods estimated on average within two percent given only P=30 positives and N=100 negatives, or within one percent given P=30 & N=1000 (e.g. estimating 6% positive when the ground truth is 5%). Note the performance with P=50 is nearly as good as at P=100. When labor costs are involved to manually label training cases, this can amount to significant savings.

In the graph for N=100 training negatives (left), the simple Classify & Count (CC) method achieved the lowest absolute error, but only for very specific training conditions. We seek methods with *consistently* good predictions, despite training variations.

Next consider N=1000 negatives (right): CC appears competitive when given a large enough set of training positives, but this is illusory. Deeper examination reveals that for smaller P it underestimates and for greater P it progressively overestimates. The training class prior is simply moving the decision threshold, resulting in more positive predictions, not better quantification. By contrast, the basic Adjusted Count (AC) method does

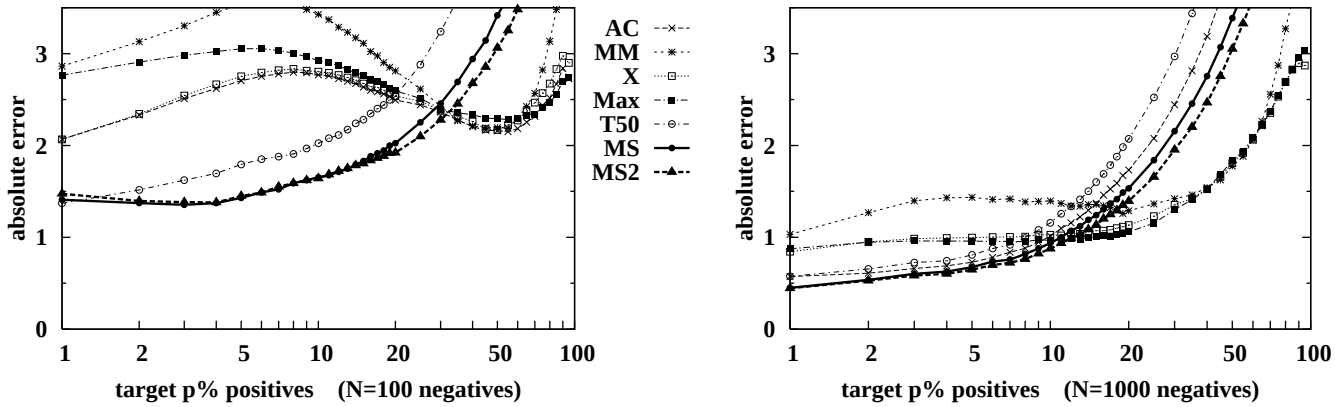

Figure 4. Absolute error for targets p=1…90% individually. P=100 training positives.

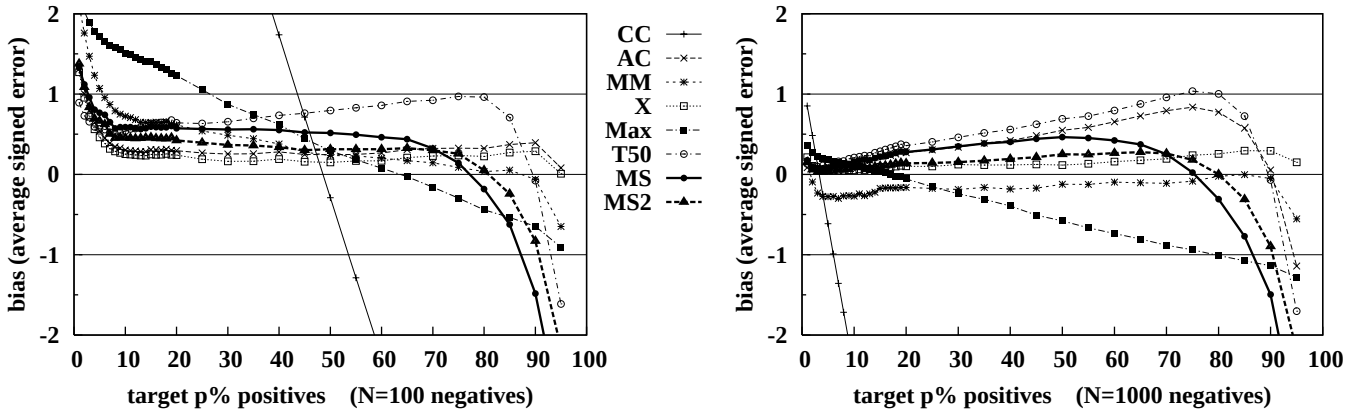

Figure 5. Bias. Average signed error for targets p=1..90% individually. P=100 training positives.

converge to better quantification once sufficient training positives are available. With P<40 & N=1000, however, the class imbalance is so great that the AC method gets severe error. Our prior work highlighted the remarkable stability of the Mixture Model (MM) method even with great imbalance. MM is dominated, however, by many of the methods in this paper designed to address the class imbalance problem. Interestingly, even *without* class imbalance (P≈N=100 at left), the MS and T50 (*tpr*=50%) methods surpass prior methods. We return to this point in the discussion.

(A few mediocre methods were omitted from the graphs to improve readability. The tech report version of this paper available at HP has a color appendix containing all the results.)

4.2 Varied training, varied target

The analysis above was for a single target of p=5% positives. Does the strong result for Median Sweep generalize for p=1…20% positives? To check this, we average the performance over this whole range. As discussed in the methodology section, to average different targets together, we must not simply use absolute error, but rather normalized cross-entropy. See Figure 3 for these results. Though the y-axis scale is now different than Figure 2, the rankings are qualitatively similar. Median Sweep continues to dominate on average.

To determine whether Median Sweep dominates for all target distributions, we expanded the study up to 95% test positives. Instead of averaging over this range, we spread out p=1..95% along the x-axis in Figure 4; but to keep the focus on the lower region, we used a log-scale x-axis. Since we are not averaging results across different values of p, the y-axis shows absolute error, which is more intuitive than cross-entropy. To view the results in two dimensions, we must hold some other variable constant: we fix P=100 training positives, where performance is relatively insensitive to changes in P.

In the low p range, Median Sweep methods excel. In the higher range, two methods excel consistently: Max (maximize *tpr* - *fpr*) or X (where *fpr* and 1-*tpr* cross). But in the following analysis we shall see that the Max method suffers from systematic bias.

Finally, the absolute error grows substantially for all methods as p approaches 95%. But this is not especially concerning: (a) it is a rare operating range, (b) if positives are so prevalent, it is more likely that the classifier would be trained with a positive majority rather than a negative majority as we have done here, and (c) in order to experiment at 95% positives, we end up with very few test negatives, due to the shortage of positive training cases. For example, if only 380 positives are available for testing, we end up with merely 20 test negatives in order to achieve p=95%. So, the experimental results have naturally higher variance in this region. If one needed to research this region more effectively, larger benchmark datasets are called for, or else the meaning of positives and negatives might be reversed.

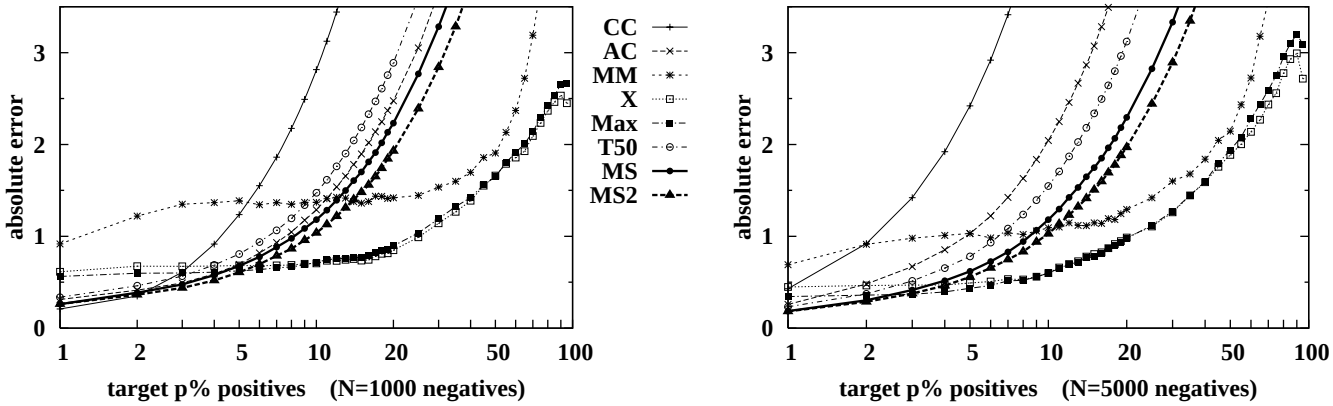

Figure 6. Like Figure 4, but for the held-out dataset, and greater training class imbalance. P=50 training positives.

4.3 Bias Analysis

Next we analyze the other component of accuracy—bias—by presenting the average signed error for each method. Figure 5 shows the bias under varied training (left vs. right) and testing conditions (p% positives on x-axis). We abandoned the log-scale here in order to show the strongly linear bias of two methods: Max and Classify & Count. For the classifier trained with 50% positives (P=N=100, at left), the CC method progressively overestimates when p<50%, and underestimates when p>50%, as expected. When trained with 9% positives (P=100, N=1000, at right), this balance point is shifted accordingly, but not proportionately—it is unbiased only at p=3% instead of 9%. We have seen this behavior consistently: SVM exaggerates the training class imbalance in testing. Although the ubiquitous suggestion is to bias the SVM cost penalty C, it proves ineffective and has been better addressed recently by Wu and Chang [12].

It is surprising that the Max method, being an Adjusted Count variant, also exhibits a linear bias, albeit to a lesser degree than CC. This means that the Max method consistently finds thresholds such that the *tpr* and *fpr* characterization does not hold well for the test set. All the other methods have a relatively stable bias over a wide range.

We see greatly increasing bias at the tails, which is expected: If a method's estimates vary by a few percent and p is close to zero, any infeasible negative estimates are adjusted up to 0%, resulting in a positive bias. As we approach p=95% positives, the even greater negative bias is similarly caused by clipping estimates which have greater variance, as shown previously.

4.4 Failure Analysis

Although an induced classifier should learn to separate cases well enough that its true positive rate *tpr* is greater than its false positive rate *fpr*, they nonetheless fail sometimes. This usually happens under great class imbalance in training. For example, for one of the ten splits on one of the tasks trained with P=10 and N=100, the induced classifier's natural threshold gave *tpr=fpr=0.* It learned to classify everything as negative, which results in a troublesome zero denominator in the adjusted count method. The commonness of this problem was part of the impetus for this research: *tpr* was less than or equal to *fpr* in 623 of 10,000 cases for AC. In progressively decreasing occurrence of failure, we have: AC, T90, F5, F10, T50 and X. The Max method never experienced a failure, exactly because it seeks to maximize the denominator. Naturally, this is a non-problem for either CC or the Mixture Model.

4.5 Naïve Bayes vs. SVM

We do not present graphs for the Naïve Bayes classifier because every quantification method under every training/testing condition performed substantially better on average with SVM as the base classifier. It is well established for text classification that SVM usually obtains better accuracy than Naïve Bayes. But our finding further suggests its *tpr* and *fpr* characteristics may be more stable as well.

4.6 Greater Imbalance, Held-Out Dataset

Given the consistent performance of the Median Sweep methods, we would like to ensure their continued performance in situations with even greater training class imbalance (~1%), such as we face in practice. This study so far has been limited to N=1000 training positives, in order to have 25 benchmark tasks for study. Although we could increase class imbalance by simply reducing P, this results in degenerate classifiers. Instead, we would like to consider a greater number of negatives. In addition, we want to validate these results against other classification problems.

For these two purposes, we held back a dataset: new3 from [6]. It has 9558 cases partitioned into 44 classes. We repeated our study on its 17 classes that have at least 200 positives, setting aside 5000 negatives for training. Figure 6 shows these results for P=50 and N=5000 negatives (~1% positives in *training*, right) and for a subset of N=1000 negatives (~5%, left). The log-scale x-axis shows p, and the y-axis shows average absolute error. Although perhaps uninteresting for its similar results to Figure 4 with N=1000, it is encouraging that the conclusions generalize to held-out tasks and to greater training imbalance. The Median Sweep methods continue to estimate well for low p; they have <1% absolute error for p<=10% in both graphs of Figure 6 and for N=1000 in Figure 4. The competitive methods Max and X have somewhat improved performance for this hold-out benchmark, and now slightly beat MS for p as low as ~5%.

5. DISCUSSION

Observe in Figure 4 that the curves cluster into two shapes: concave upward (MS,MS2,T50) and S-curve (Max,X,MM). Interestingly, the AC method under high imbalance (N=1000 and

all of Figure 6) belongs to the concave upward group, but under balanced training (N=100 in Figure 4) belongs to the S-curve group. As discussed previously, the AC method under high imbalance uses thresholds with many false negatives, i.e. closer to T50 in the concave upward group. (Recall that T50 selects the threshold at *tpr*=50%.) But under more balanced conditions, AC uses thresholds closer to the X crossover point in Figure 1, which results in the S-curve grouping. Looking now at MS, its consistent grouping with T50 suggests it may be using many estimates derived from *tpr* rates nearer to 50% than near the cross-over point X.

We set out to address a problem that occurred under class imbalance, and we ended up discovering methods that estimate substantially better even under balanced training, e.g. Median Sweep and T50. (See P~N=100 in Figures 2-4.) Since the adjusted count formula is unchanged from the basic AC method, this implies T50 selects a threshold for which *tpr*,*fpr* characterization on the training set is more reliable than the default threshold. This may provide a clue to the separate problem of estimating *tpr*,*fpr* well for other purposes.

We believe the reason that Median Sweep works so well is that instead of relying on the accuracy of a single *tpr*,*fpr* estimate, it takes in information from all the estimates, of which many are likely close. In some sense, it has the advantage of bootstrapping, without the heavy computational cost of repeating the 50-fold cross-validation for many different random samplings of the available training set.

Until now we have referred to the Median Sweep methods together. Overall they perform very similarly, which we should expect since the median is very insensitive to outlier clipping, unlike the mean. Even so, the MS2 variant—which only considers estimates that come from larger, more stable denominators—shows a slight benefit, particularly in lower bias and over a broader range. This validates the idea that small denominators result in poorer estimates. Putting MS2 into production runs the risk that on some tasks there may no estimates with a sufficiently large denominator, although this never happened in our benchmark tasks. At the very least, it could fall back to MS in such cases. Further research may develop a more robust outlier clipping method that could improve Median Sweep methods.

One motivation mentioned in the introduction for quantification research is reduced training effort to obtain a given level of accuracy. To illustrate this, note in the right-hand graph of Figure 3 that Median Sweep methods with P=20 positives achieve similar accuracy to AC with P=50. But for the basic Classify & Count method, additional training does not lead to an accurate, unbiased quantifier. Furthermore, in the left-hand graph of Figure 3 we see that additional training examples mislead AC. The point is this: quantification research is essential because accurate estimates cannot be achieved by simple methods like CC or AC just by providing more training data (unlike active learning research where all methods produce the same classification accuracy given enough training cases).

Although we are pleased to have reduced the absolute error of the estimate to less than 1% in many situations, we need to quantify increasingly rare events, where the bias and the *relative* error both grow. To conduct experiments in the tail of the distribution requires much larger labeled datasets made available for research.

6. EXTENSIONS

The implications of this work extend to trending over time, multi-class quantification, and quantification of costs, which we describe in sequence.

6.1 Trending

Measuring trends over time was listed as a primary motivation, but so far we have only discussed quantifying the class distribution of a single test set. In order to apply this technology for trending, the cases are partitioned into discrete bins, e.g. daily or monthly groupings, and the quantification is performed independently on each batch to obtain accurate estimates. These may then be plotted together in one graph, optionally with a fitted trend line to project into the future where no cases are yet available. As is typical with such applications, if there are too many bins for the volume of data, the counts in each bin become small and noisy. The quantification methods we describe are intended to work on large batches. They will produce noisy estimates given only a handful of items. For more accurate quantification in these situations, we have used a sliding window technique to aggregate cases from adjacent bins into each batch. At the same time, this provides smoothing like a moving average, e.g. to smooth over weekend-effects.

Note that this work addresses changes in the class distribution but not general concept drift, where the definition of what counts as positive may gradually or suddenly change arbitrarily [2]. When trending over time, concept drift is often implicated, and can be difficult to cope with. Ideally the quantifier used on each bin is given a training set appropriate to the class concept in effect for that bin. Naturally this can be hard to determine, and requires ongoing training data.

Regardless of concept drift, if additional training cases become available later—e.g. some cases are labeled in a new monthly batch of data—it is best to redo the trend quantification over all bins. The additional training data may improve the quantifier's estimates on the old bins as well. If instead one applies the improved quantifier only to the new batch of data, this estimate should not be appended to pre-existing trend lines. To do so would compare estimates that are not calibrated to one another.

6.2 Multi-Class Quantification

In our use of quantification, we usually want to track the trends for many classes, e.g. different types of technical support issues. Since most customers call with a single problem, these classes are competing, and therefore may be treated as a 1-of-n multi-class problem. On the other hand, occasionally multiple issues do apply to a single case. If there were a rampant trend of coinciding issues, we would not want to have designed the system to be blind to it. Hence, we quantify each issue independently, i.e. as an m-of-n multi-class topic recognition task.

Nonetheless, there are situations where the 1-of-n multi-class setting is called for. To treat this, one should not simply apply a multi-class classifier to the dataset. If some classes are much rarer than others either in the training set *or* in the test set, the test set counts predicted for those classes may be very rare. The adjusted count method applied then to each class will not lead to good multi-class estimates.

Instead we recommend performing independent quantifications for each class vs. all others, and then normalizing the estimates so

they sum to 100%. In this way, each independent quantification compensates for imperfect classification and for class imbalance.

6.3 Cost Quantification

Simply estimating the *number* of cases belonging to a category may not correlate with importance. A relatively uncommon issue having a high cost can be more important to delve into than a more frequent issue having low cost.

If the average cost per positive case C^+ is known in advance, it can simply be multiplied into the quantifier's estimate to obtain the total cost of the positive cases. More commonly C^+ is not known, and we must analyze the cost attribute attached to each case, e.g. the parts & labor cost to repair each problem. Consider a rare subclass of repairs whose costs climbed substantially in the new month of data.

6.3.1 Cost Quantification Methods

Classify & Total: The obvious solution, akin to CC, is to train a classifier and total the cost attribute associated with all cases predicted positive. But unless that classifier is perfectly accurate, it will result in poor and systematically biased cost estimates.

Grossed-Up Total: The next obvious solution is to perform the total as above, but then to adjust it up or down according to a factor f determined by quantification. If the binary classifier predicted 502 positives and the quantifier estimates 598.3 positives, then the cost total would be multiplied by f=598.3/502. But this method suffers from similar problems as AC: it runs the risk that the binary classifier may select zero or very few cases to include in the total, if positives happen to be rare in its training *or* test sets. Else if positives were overly common in the training set, then the induced liberal classifier will include in its total the costs of many negatives, polluting the result. This pollution occurs even at the perfect ratio if there are some false positives.

Conservative Average * Quantifier (CAQ): We can reduce the false-positive pollution by selecting the classifier's decision threshold to be more conservative—a classic precision-recall tradeoff. Using a smaller set of highly precise predictions, we can average their costs to estimate C^+, and then multiply it by the estimated size of the class from a quantifier. Ideally we would like a threshold with 100% precision, but often there is no such threshold. Furthermore, a highly conservative threshold based on precision may predict only a few cases as positive, esp. if positives are rare. Given too few items to average over, the variance of the C^+ estimate will be large, giving a poor overall cost estimate. To avoid this problem, one might instead always take the top, say, 100 most strongly predicted positives for the average. But this cannot ensure high precision—some test sets might have only 60 positives.

Precision-Corrected Average * Quantifier (PCAQ): Despite decreased precision, there is pressure to use a less conservative threshold for the reasons above, and also because at high precision/low recall the classifier's precision characterization from cross-validating the training set has high variance. In perfect analogy to the first part of this paper, we select a classification threshold with worse precision, but having more stable characterization as well as providing a sufficient number of predicted positives to average over. We then adjust the average according to a simple equation that accounts for the false-positive pollution:

$$\text{precision-corrected average } C^+ = \frac{(1-q)\,C_t - (1-P_t)\,C_{all}}{P_t - q} \qquad (3)$$

where q is a quantifier's estimate of the percentage of positives in the test set, P_t is an estimate of the precision at a given threshold t, C_t is the average cost of all cases predicted positive up to the threshold t, and C_{all} is the average cost of all cases. The derivation is in the appendix of the tech report version of this paper available online. The remaining design decision is which threshold to use—for example, the T50 or X thresholds shown in Figure 1. We suggest avoiding Max, given our earlier bias discussion about its choosing thresholds with poor *tpr*,*fpr* characterization.

Median Sweep PCAQ: Rather than use a single threshold and hope that its precision characterization is accurate, we may sweep over many thresholds and select the median of the many PCAQ estimates of C^+. This has some of the benefit of bootstrapping without the computational cost. Just as the MS2 method excludes estimates that are likely to have high variance, a profitable variant on this method might exclude estimates from thresholds where (a) the number of predicted positives falls below some minimum, e.g. 30, (b) the confidence interval of the estimated C^+ is overly wide, and/or (c) the precision estimate P_t was calculated from fewer than, say, 30 training cases predicted positive in cross-validation.

Mixture Model Average * Quantifier (MMAQ): Finally, rather than try to determine an estimate at each threshold, we can model the shape of the C_t curve over all thresholds as the mixture

$$C_t = P_t\,C^+ + (1-P_t)\,C^- \qquad (4)$$

where C^- is the average cost of a negative case (which is also unknown). This method estimates C^+ (and C^-) via linear regression of the points generated at many different thresholds. The same thresholds omitted by Median Sweep can be omitted here as well, in order to eliminate some outliers that may have a strong effect on the linear regression. Alternately, one may use regression techniques that are less sensitive to outliers, e.g. that optimize for L1-norm instead of mean *squared* error.

6.3.2 Evaluation

We found MMAQ outperformed CAQ in a small test. The next logical research step is to evaluate all these methods against one another. Unfortunately, any such empirical experiment depends strongly on the cost distribution for positives vs. the cost distribution for negatives (including their relative variances), in addition to variation in the training set makeup and the test class distribution. Besides its being a high dimensional experiment, we must first have a large publishable benchmark with costs of reasonable interest to a family of applications. This is an open invitation to the research community.

6.3.3 Missing Costs

In some settings, especially those in worldwide enterprises, cost values may be missing or detectably invalid for some cases. Given that most of the above methods begin by estimating the average cost for positives C^+, such cases with missing cost may simply be omitted from the analysis. That is, the estimate of C^+ is determined by the subset of cases having valid cost values, and the count is estimated by a quantifier run over all the cases. This can be effective if the data are *missing at random* (MAR).

However, if the MAR-assumption does not hold, the missing values should first be imputed by a regression predictor.

6.3.4 Cost-Confounded Prediction

The methods above implicitly assume that the cost of positive cases is not correlated with the prediction strength of the base classifier. As an assurance, one may check the correlation between cost and the classifier scores over the positive cases of the training set. If the classifier predicts the most expensive positives strongest, then the methods above, esp. CAQ, will overestimate badly. Negative correlation results in underestimates. (This problem also arises if the classifier's scores have substantial correlation with cost for *negative* cases.)

To avoid these problems, we recommend the cost attribute not be given as a predictive feature to the classifier. If the average cost for the positive class C^+ is similar to the overall average, then this attribute will generally be non-predictive. But in the interesting case where it is substantially different from the background, this feature may be strongly predictive, e.g. a rare but relatively expensive subclass. In this case, it is tempting to provide cost as a predictive feature to improve the classifier. But it is better not to: the methods are explicitly designed to function despite imperfect classifiers.

7. CONCLUSION

It is fortunate that quantification can be made to compensate for the inaccuracy of a classifier, yielding substantially more precise and less biased estimates. This requires only small amounts of training data, which can reduce labor costs compared with having to train highly accurate classifiers. These factors can lead to greater acceptance of machine learning technology for business use. We have been pushing machine learning within our company for years, but have never before experienced the business pull we find for quantification [3]. To data mining researchers who wish to apply and develop advanced prediction models, this comes as some surprise, since the task seems so simple—at least on the surface.

Though the Median Sweep, Max and X methods all show great improvement over prior technology, they are surely not the last word. Future work will involve research further down the tail toward greater class imbalance. Chemists easily talk about *parts per million*, but machine learning is currently nowhere near up to the task. To research the tail will require very large benchmark datasets, ideally publishable ones for repeatability and experimentation by others. Studying high class imbalance requires that the data set labels not have mistakes, for the conclusions are more sensitive to any noise in the answer key. Ideally, such a dataset would include individual costs to support research in *cost quantification*. The most effective methods may depend strongly on the characteristics of the data, so hopefully such a dataset would suit a popular family of applications. Other research directions are in multi-class methods, possibly including class hierarchies, or quantification under various constraints, such as having less tolerance for *under*estimating the size or cost of a subclass, as motivated by some business applications. Finally, trending over time naturally introduces concept drift, which is a challenging but important area for research.

8. ACKNOWLEDGMENTS

I wish to thank my colleagues Jaap Suermondt, Evan Kirshenbaum, Jim Stinger, Tom Tripp, and Farzana Wyde for their contributions in conceiving and developing this application. Thanks also to Bin Zhang for pre-reviewing this paper.

9. REFERENCES

[1] Fawcett, T. ROC graphs: notes and practical considerations for data mining researchers. Hewlett-Packard Labs, Tech Report HPL-2003-4, 2003. www.hpl.hp.com/techreports

[2] Fawcett, T. and Flach, P. A response to Webb and Ting's 'On the application of ROC analysis to predict classification performance under varying class distributions.' *Machine Learning*, 58(1):33-38, 2005.

[3] Forman, G., Kirshenbaum, E., and Suermondt, J. Pragmatic text mining: minimizing human effort to quantify many issues in call logs. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD, Philadelphia), 2006.

[4] Forman, G. Counting positives accurately despite inaccurate classification. In *Proc. of the 16th European Conf. on Machine Learning* (ECML, Porto):564-575, 2005.

[5] Forman, G. An extensive empirical study of feature selection metrics for text classification. *J. of Machine Learning Research*, 3(Mar):1289-1305, 2003.

[6] 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.

[7] Havre, S., Hetzler, E., Whitney, P., and Nowell, L. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9-20, 2002.

[8] Mei, Q. and Zhai, C. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Data Mining* (KDD, Chicago): 198-207, 2005.

[9] Saerens, M., Latinne, P., and Decaestecker, C. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.

[10] Vucetic, S. and Obradovic, Z. Classification on data with biased class distribution. In *Proc. of the 12th European Conf. on Machine Learning* (ECML, Freiburg):527-538, 2001.

[11] Witten, I. and Frank, E., *Data mining: Practical machine learning tools and techniques (2nd edition),* Morgan Kaufmann, San Francisco, CA, 2005.

[12] Wu, G. and Chang, E. KBA: kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. on Knowledge and Data Engineering*, 17(6):786-795, 2005.