Evaluating Cost-Sensitive Unsolicited Bulk Email Categorization

José María Gómez Hidalgo
Depto. de Inteligencia Artificial
Universidad Europea – CEES
jmgomez@dinar.esi.uem.es

Abstract

In the recent years, Unsolicited Bulk Email has became an increasingly important problem, with a big economic impact. In this paper, we discuss cost-sensitive Text Categorization methods for UBE filtering. In concrete, we have evaluated a range Machine Learning methods for the task (C4.5, Naive Bayes, PART, Support Vector Machines and Rocchio), made cost sensitive through several methods (Threshold optimization, Weighting, and MetaCost). F or the evaluation, we have used the Receiver Operating Characteristic Convex Hull method, that best suits classification problems in which target conditions are not known, as it is the case. Our results do not show a dominant algorithm nor method for making algorithms cost-sensitive, but are the best reported on the test collection used, and approach real-world manual classifiers accuracy.

1 Introduction

Internet mail is an increasingly important medium of communication, with direct impact on human relations and businesses. However, it is also prone to misuse, specially by unethical direct marketing companies. Spam email, or more properly, Unsolicited Bulk Email (UBE), has been producing a considerable damage to Internet Service Providers, users and the whole Internet backbone. For instance, according to an study undertaken for the European Commission, Internet subscribers worldwide are wasting an estimated 10 billion euro a year just in connection costs due to UBE [13].

Several proposals have been made to alleviate the UBE problem, ranging from technical to regulatory and economic (see [6, 16] for an overview). Among the technical mechanisms proposed, filtering is specially promising and currently in use widely [16]. A popular approach is filtering email using the message features, in contrast to other filtering mechanisms like channels and aliasing [14, 12]. Most email clients allow users to manually build email filters. Also, mail processing systems include filtering capabilities at the server side, that are configured by their administrators by hand. Finally, some corporations have expert teams building UBE filters for other companies [4]. In general, these filters are costly produced and updated by users and administrators.

In the recent years, researchers are increasingly using Text Categorization (TC) techniques to automatically build UBE filters [3, 2, 1, 5, 8, 15, 18, 20, 25, 28, 29]. These approaches consist of building automatic UBE classifiers using Machine Learning algorithms trained on a collection of UBE and legitimate messages. Our aim is identifying the best Machine Learning (ML) algorithm for this task, that is, the one which produces the most accurate UBE classifier. Several learning algorithms have been evaluated in related work, but conclusions are hardly valid because of the two following reasons:

- Some tests performed in the literature (e.g. [18, 20, 25, 28]) have not taken into account that UBE classification is a cost-sensitive classification problem. If a UBE classifier decides that a message is legitimate and it is in fact UBE, this mistake will be better tolerated by the user than the opposite (assigning a legitimate to the UBE class). So, the first error is less expensive than the second, and in consequence, we face a problem with asymmetric misclassification costs.

- Other tests performed in the literature (e.g. [3, 2, 1, 5, 8, 29]) have considered asymmetric misclassification costs, but restricted to scenarios that do not correspond to real world conditions, that remain unknown.

In this paper, we address the evaluation of a representative sample of ML algorithms for the problem of inducing automatic UBE classifiers. The algorithms we have evaluated are the decision tree learner C4.5 [26], probabilistic method Naive Bayes [19], the rule learner PART [11], the relevance feedback Rocchio algorithm [27], and the kernel method called Support Vector Machines (SVM) [17]. Our evaluation has been performed using the Receiver Operating Characteristics Convex Hull (ROCCH) method [22, 9, 23]. This method is based on plotting Receiver Operating Characteristic (ROC) curves, that allow a visual comparison of ML algorithms regardless of the final operating conditions (real costs and class distributions, usually unknown in laboratory test conditions). The ROCCH method also allows to discard algorithms that can not be optimal for any real world conditions, and to identify the best available algorithm when real world conditions are finally assessed.

The ROCCH method is sensitive to the way ROC curves are plotted, which is in turn dependent on the method used to make a ML algorithm cost-sensitive. Among the several methods for making ML algorithms cost-sensitive available[1], many of them algorithm-dependent, we have focused on those that are applicable to most ML algorithms. Specifically, we have tested the Thresholding optimization method [33], suitable for algorithms that output numeric predictions (typically probabilities), and the algorithm-independent Weighting [32] and MetaCost [7] methods.

[1]See the bibliography maintained by Peter Turney at `http://extractor.iit.nrc.ca/bibliographies/cost-sensitive.html`.

The results obtained in our experiments have not shown a clear dominant algorithm nor method for cost-sensitivity. However, we can outline the following conclusions:

- The most often dominant algorithm is SVM, which it is also one of the best performing learning algorithms in the literature on TC and UBE categorization.
- To our surprise, the method for making ML algorithms cost-sensitive is Weighting. This method is equivalent to stratification by oversampling, which was proven worse in [7] for a large sample of common ML benchmark collections.
- The SVM algorithm trained with the Weighting method has produced best results ever reported for UBE categorization in the literature. The classifier produced in these conditions is able to detect a 91.7% of the UBE messages without misclassifying any legitimate messages. This makes the classifier valid for real-world conditions, in which the best ever reported results to our knowledge reached the 93.91% of the UBE messages detected [10].

The rest of the paper is organized as follows. First, UBE filtering is addressed as a Text categorization problem. Next, we discuss the problems of the evaluation methodology used in other's work, and the ROCCH method. After, the relation between this evaluation methodology and cost-sensitive learning is discussed. Next, we describe the experimental setup. Finally, the results of our experiments, its discussion, and the conclusions are presented.

2 UBE Categorization

On the recent years, several proposals have been made to address the UBE problem [6]. While economic and regulatory measures are difficult to implement, UBE filtering is actively been used by corporations, Internet Service Providers (ISPs) and individual email users.

Most filtering approaches rely on manually building and updating classification rules for email messages. The administration of such filters is costly task. However, many recent work on UBE filtering has addressed it as a Text Categorization problem. Automatic TC is the automatic classification of a set of documents into a set of predefined categories [35, 31]. In the case of UBE filtering, the documents are email messages and the categories are UBE and its complementary class, legitimate email. We will call this approach UBE Categorization.

In TC, the dominant approach is to apply Machine Learning and Information Retrieval techniques for, given a set of manually pre-classified documents, building an automatic classifier able to assign new documents to the categories. This approach has been followed by much recent work on UBE Categorization [3, 2, 1, 5, 8, 15, 18, 20, 25, 28, 29].

According to [31], the main issues when building an automatic classifier are the representation of text documents, dimensionality reduction, the selection of the ML algorithm, and the evaluation of the induced classifiers. We leave the latter issue for the next section.

2.1 Text Representation

The most often adopted representation of the messages is as term weight vectors, in the Vector Space Model (VSM) [30]. The terms are usually words and exceptionally trigrams (as in [20]. Quite often, terms are stemmed and filtered according to a stoplist. The weights in the vectors (that is, the attribute values) are frequently binary, perhaps because the most widely applied learning algorithm to the task is Naive Bayes (NB). Optionally, weights can be *tf* or *tf.idf* [8], or ad-hoc [15, 25]. The weight *tf* is defined as the number of times a term occurs in a message. The *tf.idf* weighting schema defines the weight of the ith term in the jth document as:

$$w_{ij} = tf_{ij} \cdot \log_2 \left(\frac{N}{df_i} \right)$$

Being tf_{ij} the number of times that the ith term occurs in the jth document, N the number of documents, and df_i the number of documents in which the ith term occurs.

2.2 Dimensionality Reduction

Dimensionality reduction is a required step because it improves efficiency and reduces overfitting. In the UBE categorization literature, terms are often selected with respect to their Information Gain (IG) scores [3, 2, 1, 8, 18, 28, 29], and sometimes according to ad-hoc metrics [15, 20]. Sometimes, term selection is not performed [5, 25]. When applied, the final number of terms is not standard (even in relation to the initial number of terms). Androutsopoulos et al. [3, 2] test several numbers of terms (from 50 to 700 in steps of 50) selected according to IG defined as:

$$IG(X, C) = \sum_{x=0,1;c=u,l} P(X = x, C = c).\log_2 \frac{P(X = x, C = c)}{P(X = x) \cdot P(C = c)}$$

Being u the UBE class and l the legitimate email class. Interestingly, IG is one of the best selection metrics for TC, as shown in [34].

2.3 Learning Algorithms

A wide variety of learning approaches have been applied to UBE Categorization, including Naive Bayes [3, 2, 1, 15, 18, 20, 25, 28, 29], Ripper [8, 20, 25], k-Nearest-Neighbors (kNN) [2, 15], boosted C4.5 [5, 8], Rocchio and linear Support Vector Machines (SVM) [8], C4.5 and PART [15], Genetic Algorithms [18], and

Stacking applied to Naive Bayes and kNN. The most often applied learner is the probability-based classifier Naive Bayes. Due to reasons we will discuss in the next section, cross-comparison is impossible at the moment; however, a detailed look at the results published in the literature let us reasonably expect SVM and boosted C4.5 be the top performing learning algorithms for UBE Categorization.

3 Evaluating UBE Categorization with the ROCCH Method

The two most relevant issues in TC effectiveness evaluation are the selection of the test collection and the evaluation metrics. There is an increasing agreement among TC researchers in relation to these decisions. The most often used TC test collection is Reuters[2], that includes 21578 news stories classified according to a set of 135 economic subject codes [31]. A number of learning algorithms have been tested and compared on Reuters [31, 35]. The most used performance metrics in TC evaluation are F_1 and the breakeven point (discussed below) [31, 35].

Evaluation is less standardized in UBE Categorization. Up to 11 different test collections have been used in different experiments, most of them kept private. The evaluation metrics include accuracy, miss and false alarm rates, recall and precision, and cost oriented measures like total cost ratio. Apart of this lack of uniformity, there are methodological mistakes that are worth to comment. In this section, we first discuss the topic of test collections, and after we turn to evaluation metrics. Finally we describe the ROCCH method, followed in our experiments.

3.1 Test Collections

A number of UBE test collections have been used in previous experiments. Only three of the collections used in previous experiments have been made publicly available for the research community: Ling-spam[3], PU1[4], and Spambase[5]. Ling-spam contains messages from the archives in the Linguist List and public UBE. PU1 contains (encoded) private email, which approximates a real usage scenario. As it comes encoded due to privacy issues, no real knowledge about the term of the messages is available, and the work on it is necessarily limited. Spambase messages have been donated by several users, and comes in a preprocessed fashion (48 terms plus 8 additional features extracted from the messages, ad-hoc weights). Again, the work with this collection is limited because no other number of terms can be extracted, and no other weighting schemas can be tested. Additionally, we would like to make the following remarks:

[2]Available at `http://www.research.att.com/~lewis/reuters21578.html`.
[3]Available at `http://www.iit.demokritos.gr/~ionandr/lingspam_public.tar.gz`.
[4]Available at `http://www.iit.demokritos.gr/~ionandr/pu1_encoded.tar.gz`.
[5]Available at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase`.

- No collection except Ling-spam and PU1 have been used in two or more papers. Ling-spam has been used in [2, 1], and PU1 has been used in [3, 5]. This makes results hardly comparable. Ling-spam is perhaps the most useful UBE Categorization test collection, because it is public, and comes in raw form. However, as the source of legitimate email is the Linguist List, it does not reflect a real usage scenario, and conclusions from UBE Categorization evaluation on this collection can only be limited.

- The percentage of UBE messages in the test collections is highly variable, ranging from 16,6% in [2, 1] to 88,2% in [28]. Moreover, none of the percentages is entirely valid because real world conditions are highly variable and unpredictable. For instance, the percentage of UBE in Ling-spam approximates that reported [6]. In this latest study, a 10% of UBE is reported for corporations, and only a 2% for ISPs. However, this numbers may have changed since 1997. More strikingly, America On Line was receiving around a 30% of UBE messages by the same dates. The amount of UBE may vary from ISP to ISP, from corporation to corporation, and from user to user.

From these facts, we obtain two conclusions: first, Ling-spam is the currently best (although not perfect) candidate for the evaluation of UBE Categorization; second, it is clear that UBE Categorization evaluation cannot depend on the amount of UBE of a collection – at least, evaluation metrics should be independent of class distribution in a UBE Categorization test collection.

3.2 Evaluation Metrics

The effectiveness of TC systems is measured in terms of the number of correct and wrong decisions. For simplicity, we will restrict ourselves to the problem of taking binary decisions about a single class, which is the case of a UBE Categorization system. Let us suppose that the TC system classifies a given number of documents. We can summarize the relationship between the system classifications and the correct judgments in a confusion matrix, like that shown in Table 1. Each entry in the table specifies the number of documents with the specified outcome. For the problem of categorizing UBE, we take UBE' as the positive class (+), and legitimate as the negative class (–). In this table, the key "tp" means "number of true positive decisions", and "tn", "fp" and "fn" refer to the number of "true negative", "false positive" and "false negative" decisions, respectively.

Most traditional TC evaluation metrics can be defined in terms of the entries of the confusion matrix. F_1 [31] is a measure that gives equal importance to recall and precision. Recall is defined as the proportion of class members assigned to a category by a classifier. Precision is defined as the proportion of correctly assigned documents to a category. Given a confusion matrix like the one shown in Table 1, recall (r), precision (p) and F_1 are computed using the following formulas:

	+	–
+	tp	fp
–	fn	tn

Table 1: A set of n classification decisions represented as a confusion matrix. The symbol "+" represents the concept "belonging to the (positive) class", and "–" represents the opposite. Columns represent the real classes, while rows represent the decisions taken by the system.

$$r = \frac{tp}{tp + fn} \quad p = \frac{tp}{tp + fp} \quad F_1 = \frac{2rp}{r + p}$$

Recall and precision metrics have been used in some of the works in UBE Categorization (e.g. [3, 2, 1, 15, 28, 29]. Other works make use of standard ML metrics, like accuracy and error [18, 20, 25]. Many of the papers in the literature take into account that not all kinds of classification mistakes have the same importance for a final user [3, 2, 1, 5, 8, 15, 29]. Intuitively, the error of classifying a legitimate message as UBE (a false positive) is far more dangerous than classifying a UBE message as legitimate (a false negative). This observation can be reexpressed as the cost of a false positive is greater than the cost of a false negative in the context of UBE Categorization. Misclassification costs are usually represented as a cost matrix in which the entry C(A,B) means the cost of taking a A decision when the correct decision is B, that is the cost of A given B (cost(A|B)). For instance, C(+,–) is the cost of a false positive decision (classifying legitimate email as UBE), and C(–,+) is the cost of a false negative decision).

The situation of unequal misclassification costs has been observed in many other ML domains, like fraud and oil spills detection [22]. The metric used for evaluating classification systems must reflect the asymmetry of misclassification costs. In the area of UBE Categorization, several cost-sensitive metrics have been defined, including Weighted Accuracy (WA), Weighted Error (WE), and Total Cost Ratio (TCR) (see e.g. [2]. Given a cost matrix, the cost ratio (CR) is defined as the cost of a false positive over the cost of a false negative. Given the confusion matrix for a classifier, the WA, WE and TCR of for the classifier are defined as:

$$WA = \frac{CR \cdot tn + tp}{CR \cdot (tn + fp) + (tp + fn)} \quad WE = \frac{CR \cdot fp + fn}{CR \cdot (tn + fp) + (tp + fn)}$$

$$TCR = \frac{tn + fp}{CR \cdot fp + fn}$$

The WA and WE metrics are version of the standard accuracy and error measures, that penalize those mistakes that are unpreferred. Taking the trivial rejecter that classifies every message as legitimate (equivalent to not using a filter) as a baseline, the TCR of a classifier represents to what extent is a classifier better than it. These metrics are less standard than others used in cost-sensitive classification, as Expected Cost, but to some extent they are equivalent. These metrics have been calculated for a variety of classifiers, in three scenarios corresponding to three CR values (1, 9 and 999) [3, 2, 1, 5, 15, 29].

The main problem presented in the literature on UBE cost-sensitive categorization is that the CR used do not correspond to real world conditios, which are unknown and may be highly variable. There is not evidence that a fp (classifying a legitimate message as UBE) is 9 nor 999 times worse than the opposite mistake. As class distributions, CR values may vary from user to user, from corporation to corporation, and from ISP to ISP. The evaluation methodology must take this fact into account. Fortunately, there are methods that allow to evaluate classifiers effectiveness when target (class distribution and CR) conditions are not known, as in UBE Categorization. In the next subsection, we introduce the ROCCH method for UBE Categorization.

3.3 The ROCCH Method

The Receiver Operating Characteristics (ROC) analysis is a method for evaluating and comparing a classifiers performance. It has been extensively used in signal detection, and introduced and extended by Provost and Fawcett in the Machine Learning community (see e.g. [22, 23]). In ROC analysis, instead of a single value of accuracy, a pair of values is recorded for different class and cost conditions a classifier is learned. The values recorded are the False Positive rate (FP) and the True Positive rate (TP), defined in terms of the confusion matrix as:

$$FP = \frac{fp}{fp + tn} \quad TP = \frac{tp}{tp + fn}$$

The TP rate is equivalent to the recall of the positive class, while the FP rate is equivalent to 1 less the recall of the negative class. Each (FP,TP) pair is plotted as a point in the ROC space. Most ML algorithms produce different classifiers in different class and cost conditions. For these algorithms, the conditions are varied to obtain a ROC curve. We will discuss how to get ROC curves by using methods for making ML algorithms cost-sensitive.

One point on a ROC diagram dominates another if it is above and to the left, i.e. has a higher TP and a lower FP. Dominance implies superior performance for a variety of commonly performance measures, including Expected Cost (and then WA and WE), recall and others [9, 23]. Given a set of ROC curves for several ML algorithms, the one which is closer to the left upper corner of the ROC space represents the best algorithm.

Dominance is rarely got when comparing ROC curves. Instead, it is possible to compute a range of conditions in which one ML algorithm will produce at least better results than the other algorithms. This is done through the ROC Convex Hull method, first presented in [22]. Concisely, given a set of (FP,TP) points, thus that do not lie on the upper convex hull correspond to suboptimal classifiers for any class and cost conditions. In consequence, given a ROC curve, only its upper convex hull can be optimal, and the rest of its points can be discarded. Also, for a set of ROC curves, only the fraction of each one that lies on the upper convex hull of them is retained, leading to a slope range in which the ML algorithm corresponding to the curve produces best performance classifiers.

The ROC analysis allows a visual comparison of the performance of a set of ML algorithms, regardless of the class and cost conditions [22]. This way, the decision of which is the best classifier or ML algorithm can be delayed until target (real world) conditions are known, and valuable information can be obtained at the same time. In the most advantageous case, one algorithm is dominant over the entire slope range. Usually, several ML algorithms will lead to classifiers that are optimal (among those tested) for different slope ranges, corresponding to different class and cost conditions.

Operatively, the ROCCH method consists of the following steps:

1. For each ML algorithm, obtain a ROC curve and plot it (or only its convex hull) on the ROC space.

2. Find the convex hull of the set of ROC curves previously plotted.

3. Find the range of slopes for which each ROC curve lies on the convex hull.

4. In case that target conditions are known, compute the corresponding slope value and output the best algorithm. In other case, output all ranges and best local algorithms or classifiers.

We have made use of the ROCCH method for evaluating a variety of ML algorithms for the problem of UBE Categorization. For more information on the ROCCH method, consult [22].

4 Cost-Sensitive UBE Categorization

One key issue in the ROCCH method is the way ROC curves are obtained for a ML algorithm. Many ML algorithms produce classifiers that output numeric predictions. For instance, probabilistic learning algorithms like Naive Bayes, decision tree and rule learners like C4.5 and Ripper, etc. are used to build classifiers whose output is a probability estimation of membership to the positive class. For these algorithms, the most popular method for plotting a ROC curve is threshold variation [22, 33]. The method works as follows: given a set of test instances (e.g. new email messages) and a classifier, the numeric output for each test instance is computed, and the instances are ordered according to the corresponding numeric prediction. Then, for each instance, a (FP,TP) is obtained as in recall-precision curves, that is, considering that instances before it are classified as positive and instances after it are classified as negative. Subsequent (FP,TP) points are then linked [33].

This method for plotting a ROC curve is closely related to a method for making algorithms cost-sensitive, that we will call the Threshold method. Once a numeric-prediction classifier has been produced using a set of pre-classified instances (the training set), one can compute a numeric threshold that optimizes cost on another set of pre-classified instances (the validation set). When new instances are to be classified, the numeric threshold for each of them determines if the instances are classified as positive (UBE) or negative (legitimate).

Instead of drawing a ROC curve through threshold variation, we can vary the class and cost conditions and obtain for each of them a (FP,TP) point using the Threshold method [33]. This view of ROC curve plotting allows to use other methods for making ML algorithms cost-sensitive. For instance, one can use techniques as Stratification or MetaCost applied to a ML algorithm for inducing a set of classifiers for a range of class and cost conditions, and then linking the obtained (FP,TP) points to form a ROC curve.

This is the basis of the method we have applied to obtain ROC curves for a range of methods for making ML algorithms cost-sensitive. In concrete, given a ML algorithm (like C4.5), a method for making algorithms cost-sensitive (like MetaCost), and a set of class and cost conditions (for instance, class distribution fixed as in Ling-spam, and cost ratios varying from 1/100 to 1 in steps of 1/10, and from 1 to 100 in steps of 10):

1. We obtain a (FP,TP) point for each of the class and cost conditions by ten-fold cross-validation.
2. We discard the points that do not belong to the convex hull.
3. And finally, we link the remaining (FP,TP) points.

It is important to note that this is not the method suggested in [24, 33] for plotting ROC curves. Instead, they propose to first obtain a ROC curve for each fold, and then averaging the obtained ROC curves. We instead obtain a (FP,TP) point for each fold, and the we average in the assumption that the results for each fold are actually estimating the same point in the ROC space.

5 Experimental Environment

In this section, we briefly review the experimental setup for the evaluation we have performed. First, we discuss the test data preparation. Then, we present the algorithms tested and their operating conditions. Finally, methods we have employed for making the learning algorithms cost-sensitive are described.

5.1 Test Data Preparation

Due to the reasons outlined above, we have selected the Ling-spam test collection for our evaluation process. This test collections consists of 2412 messages extracted from the Linguist mailing list archive, and 481 public UBE messages.

From the 4 versions available, we have used the one that comes with terms stemmed using the *morph* lematizer[6] and without the 100 most frequent words in the British National Corpus[7]

The collection has been processed using the Smart (v. 11) retrieval engine[8]. There were around 5000 different terms, among which we have selected the 500 top IG scoring. Each message has been represented as a term weight vector, in which the weights were binary, except for the Rocchio algorithm, which better suits *tf.idf* weights.

5.2 Machine Learning Algorithms

The algorithms tested in this study are:

- The C4.5 decision tree learner [26]. This algorithm induces decision trees by a top-down strategy, separating instances according to the values of high IG score attributes. It has been used with prunning activated.
- The Naive Bayes probabilistic schema [19], in which the posterior probability of an instance belonging to the positive class is computed from the prior probability, computed using Laplacean estimators.
- The rule learner PART [11], a rule learner that, unlike C4.5Rules nor Ripper, builds rules one at time without a global optimization process, by generating partial decision trees.
- The kernel method called Support Vector Machines (SVM) [17], which are maximum margin hyperplanes that attempt to separate training instances, built through Platt's sequential minimal optimization algorithm using polynomial kernels [21]. For efficiency, the computation has been restricted to the linear case.
- The relevance feedback Rocchio algorithm [27], which builds category vector prototypes by averaging on the positive and negative instances. The parameters β and γ, which control the impact of positive and negative instances, have been set to 4 and 1 respectively.

The implementation of the previous algorithms except Rocchio that we have used for this work is the one provided in the WEKA package[9]. This package, written in Java, includes the implementation of many popular ML algorithms, and preprocessing and evaluation capabilities. The Rocchio algorithm has been codified and added to the library.

[6]This stemmer comes as a part of the GATE package, available at `http://www.dcs.shef.ac.uk/research/groups/nlp`.

[7]These statistics are available at `ftp://ftp.itri.bton.ac.uk/pub/bnc`.

[8]Available at `ftp://ftp.cs.cornell.edu/pub/smart/`.

[9]Available at `http://www.cs.waikato.ac.nz/ml/weka/index.html`.

5.3 Cost-sensitive Methods

We have tested the three following methods for making algorithms cost-sensitive:

- The Threshold method [33], described above.
- The Weighting method [32], that consists of reweighting training instances according to the total cost assigned to each class. This method is equivalent to stratification by oversampling as described in [7]. The main idea is replicating instances of the most costly class, to force the ML algorithm to correctly classify that class instances.
- The MetaCost method [7], based on building an ensemble of classifiers using the bagging method, relabelling training instances according to cost distributions and the ensemble outcomes, and finally training a classifier on the modified training collection.

These methods have been used as implemented in the WEKA package, with the default parameter values. We have calculated ROC curves for hte five ML algorithms and the three cost-sensitivity method, excluding the combinations Rocchio + Weighting (since Rocchio works averaging, it makes no sense) and Rocchio + MetaCost (MetaCost is not siutable for stable learners). The points in the ROC curves have been obtained training a cost-sensitive classifier for a sample of cost conditions, keeping the class distribution unchanged. The cost ratios used were 1/1000, 1/900, ... 1/100, 1/90, ..., 1/20, 1/10, 1/5, 1, 5, 10, 20, ..., 90, 100, 200, ..., 900 and 1000, which leads to 43 points per ML algorithm and cost-sensitivity method combination.

6 Results and Interpretation

We first present the results for each algorithm alone, in order to show the effect of the methods for making it cost-sensitive. After that, we present the results for all the algorithm and cost-sensitivity method combination. Finally, our results are compared with other results in the literature, and with real-world results.

6.1 Cost-sensitive Methods

The results for the C4.5, Naive Bayes, PART and SVM algorithms are shown in the figure 1 and the table 2. As it can be seen, no clear winner (that is, no dominant) cost-sensitivity method can be found. The method that are most often optimal is Weighting, but it do not dominates other methods for any ML algorithm tested. Interestingly, in high slope ranges corresponding to extreme CR like those used in the bibliography (9 and 999), Weighting is the most frequent optimal method (except for PART).

C4.5			Naive Bayes		
Slope Range	(FP, TP)	Classifier	Slope Range	(FP, TP)	Classifier
[0.000,0.008]	(1.000,1.000)	C45MCi100	[0.000,0.003]	(1.000,1.000)	AllPos
[0.023,0.058]	(0.260,0.990)	C45THi020	[0.003,0.009]	(0.222,0.998)	NBTHi020
[0.058,0.101]	(0.191,0.986)	C45TH010	[0.009,0.017]	(0.110,0.997)	NBWE200
[0.101,1.833]	(0.033,0.970)	C45WE005	[0.017,0.381]	(0.050,0.996)	NBWE400
[1.833,2.467]	(0.027,0.959)	C45WE010	[0.381,0.480]	(0.029,0.988)	NBWE500
[2.467,18.00]	(0.012,0.922)	C45WE030	[0.480,16.00]	(0.004,0.976)	NBWE600
[18.00,36.30]	(0.010,0.886)	C45WE090	[16.00,472.0]	(0.002,0.944)	NBWE800
[36.300,∞]	(0.000,0.523)	C45WE1000	[472.00,∞]	(0.000,0.000)	AllNeg
PART			SVM		
Slope Range	(FP, TP)	Classifier	Slope Range	(FP, TP)	Classifier
[0.000,0.081]	(0.206,1.000)	PAMCi040	[0.000,0.008]	(0.239,1.000)	SVTHi005
[0.081,1.000]	(0.058,0.988)	PAWE005	[0.008,0.044]	(0.108,0.999)	SVWEi005
[1.000,1.913]	(0.031,0.961)	PAWE040	[0.044,0.538]	(0.040,0.996)	SVTH001
[1.913,107.0]	(0.008,0.917)	PAWE100	[0.538,1.316]	(0.027,0.989)	SVWE010
[107.0,116.5]	(0.006,0.703)	PAWE900	[1.316,5.875]	(0.008,0.964)	SVWE050
[116.50,∞]	(0.000,0.004)	PATH700	[5.875,∞]	(0.000,0.917)	SVWE200

Table 2: Results for the experiments. For each of the algorithms, the slope ranges, (FP,TP) and dominant classifiers are shown. The classifiers are codified as follows: first two letters correspond to the learning algorithm, next 2 letters correspond to the cost-sensitivity method, and the final number corresponds to the CR used (an "i" means 1/n instead of n, being n the CR). AllPos and AllNeg represent the trivial acceptor and rejector, respectively.

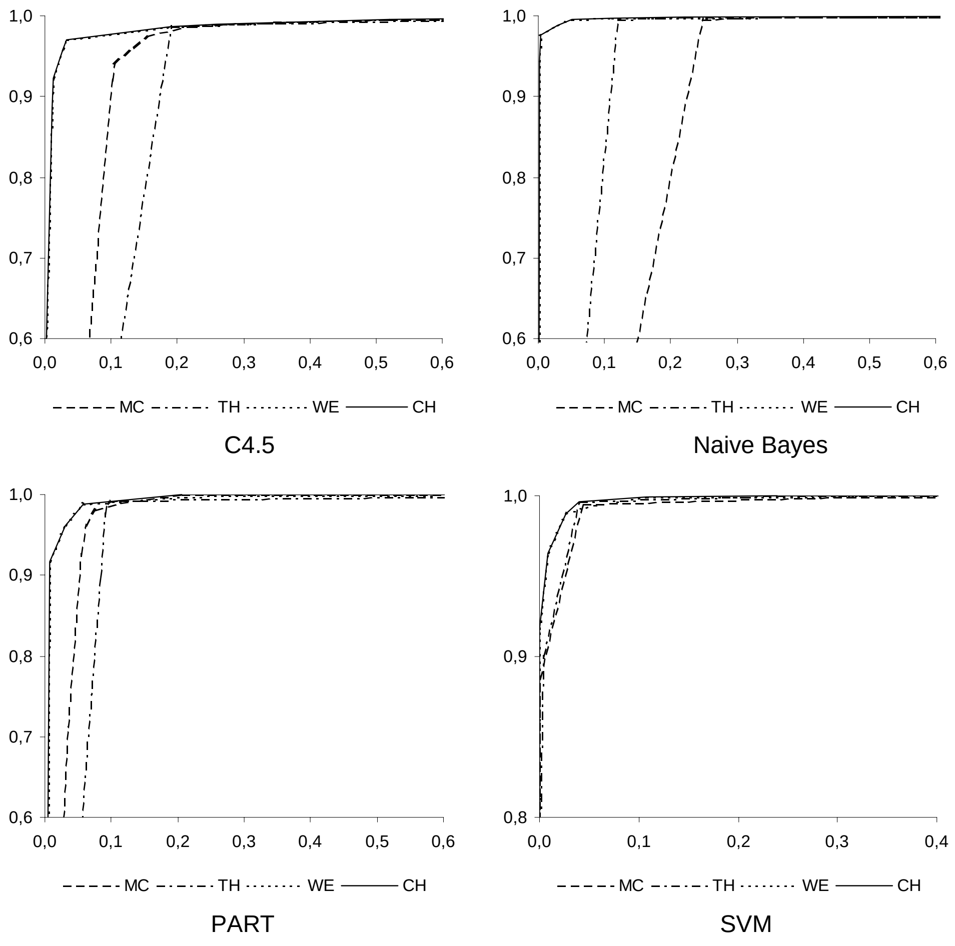

Figure 1: Results for the experiments. For each of the algorithms, the ROC curves corresponding to the cost-sensitivity methods are presented, and also the convex hull (CH).

Slope Range	(FP, TP) point	Classifier
[0.000,0.010]	(0.206,1.000)	PAMCi040
[0.010,0.044]	(0.108,0.999)	SVWEi005
[0.044,0.357]	(0.040,0.996)	SVTH001
[0.357,1.250]	(0.012,0.986)	ROTHi020
[1.250,14.750]	(0.004,0.976)	NBWE600
[14.750,∞]	(0.000,0.917)	SVWE200

Table 3: Summary of results for all algorithms and methods, following the codification in table 2.

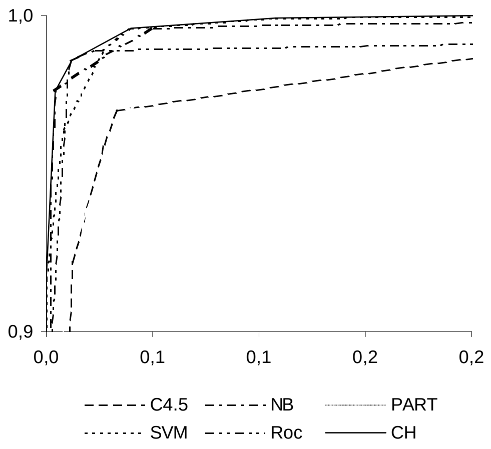

Figure 2: Results for the experiments. For each of the algorithms and cost-sensitivity methods, the ROC convex hull curves are presented (including the ROC curve for Rocchio + Thresholding), and also the convex hull of all of them.

6.2 Learning Algorithms

In figure 2, the ROCCH for each algorithm are presented, including the ROC curve for the Rocchio + Threshold combination. In the table 3, we present the best classifiers for the slope ranges shown. While no dominant algorithm is found, the best classifier for extreme cost conditions (those in which a user will not allow the system to filter out a legitimate message), the best classifier is SVM trained with the Weighting method for CR = 200.

6.3 Comparison with Related Work

In the UBE Categorization literature, three scenarios were introduced, corresponding to CR = 1, 9 and 999, and several cost-sensitive metrics have been caculated. In the table 4 we present the best results obtained in our experiments. Given the class distribution in Ling-spam (16.6% of UBE messages),

Cost Ratio	Slope	Best Classifier	R	P	WA	TCR
1	5.014	NBWE600	0.976	0.979	0.992	22.697
9	45.130	SVWE200	0,917	1.000	0.999	12.048
999	5009.538	SVWE200	0,917	1.000	0.999	12.048

Table 4: Best results of our experiments for the scenarios discussed in [3, 2, 1, 5, 15, 29]. Recall (R) and Precision (P) for the UBE class, Weighted Accuracy (WA) and Total Cost Ratio (TCR) are presented.

and the CR of each scenarios, the corresponding slopes are obtained, and the best classifier is shown accordingly. We have also calculated the recall and precion of the UBE class, the Weighted Accuracy and the Total Cost Ratio. To our knowledge, these are the best reported results for UBE Categorization on this benchmark collection [2, 1, 29].

The second and third rows in table 4 show performance numbers that approach best results reported in real-world studies [10], in which a recall of 0.939 is presented. This latter figure corresponds to a service in which a team of experts are manually building filtering rules 24 hours a day. Our results suggest that the process of building UBE filters may be nearly automatic using ML algorithms.

7 Conclusions and Future Work

In this paper, we have discussed the problem of UBE filtering as a problem of Text Categorization. Also, we have evaluated several Machine Learning algorithms made cost-sensitive throug several methods, on a public test collection. We have performed the evaluation using the ROCCH method, that specially suits classification problems in which target conditions are not known in advance. The results of our experiments show no clear dominant ML algorithm nor cost-sensitivity technique, but one of the classifiers produced was able to detect the 91.7% of the UBE messages, without discarding any legitimate messages.

The results of our experiments are promising, although we have not fully exploited the potential information in the email messages. Some papers in the literature suggest to consider other features than words in the messages, like the address of the sender, the number of capital letters, or some hand-crafted expressions like "win big money", to produce more accurate classifiers [15, 28]. In fact, what we propose is to follow a complete Knowledge Discovery process, identifying good candidates for message features, and producing accurate classifiers in the context of a real usage scenario. The best users for building such an scenario are probably ISPs, because they are the most affected users by the problem of UBE.

Acknowledgements

I would like to thank Manuel Maña and Enrique Puertas for their help with the experiments. I would also like to thank Tom Fawcett for his enlightening comments on the ROCCH method.

References

[1] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M.n van Someren, editors, *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000.

[2] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, , and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France, 2000.

[3] Ion Androutsopoulos, John Koutsias, Konstandinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 160–167, Athens, GR, 2000. ACM Press, New York, US.

[4] Brightmail, Inc. Brightmail solution suite technical white paper. Technical report, Brightmail, Inc., 2000.

[5] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001.

[6] Lorrie F. Cranor and Brian A. LaMacchia. Spam! *Comm. of the ACM*, 41(8), 1998.

[7] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proc. of the 5th International Conference on Knowledge Discovery and Data Mining*, 1999.

[8] Harris Drucker, Vladimir Vapnik, and Dongui Wu. Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[9] Chris Drummond and Robert C. Holte. Explicitly representing expected cost: An alternative to roc representation. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 198–207, 2000.

[10] eTesting Labs, Inc. Brightmail, inc. anti-spam service: Comparative performance test. Technical report, eTesting Labs, Inc., a Ziff Davis Media Inc. company, March 2001. Available at `http://etestinglabs.com`.

[11] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, pages 144–151. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[12] Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, and Alain Mayer. Consistent, yet anonymous, Web access with LPWA. *Communications of the ACM*, 42(2):42–47, 1999.

[13] Serge Gauthronet and Étienne Drouard. Unsolicited commercial communications and data protection. Technical report, Commission of the European Communities, 2001.

[14] Robert J. Hall. A countermeassure to duplicate-detecting anti-spam techniques. Technical Report TR 99.9.1, AT&T Labs Research, 1999.

[15] J.M. Gómez Hidalgo, M. Ma na López, and E. Puertas Sanz. Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of the Fourth Computational Natural Language Learning Workshop, CoNLL-2000*. Association for Computational Linguistics, 2000.

[16] Paul Hoffman and Dave Crocker. Unsolicited bulk email: Mechanisms for control. Technical Report Report UBE-SOL, IMCR-008, Internet Mail Consortium, 1998.

[17] Thorsten Joachims. A statistical learning model of text classification with support vector machines. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, New Orleans, US, 2001. ACM Press, New York, US.

[18] Hooman Katirai. Filtering junk e-mail: A performance comparison between genetic programming & naive bayes. Available at `http://citeseer.nj.nec.com/katirai99filtering.html`, 1999.

[19] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the “Lecture Notes in Computer Science” series, number 1398.

[20] Patrick Pantel and Dekang Lin. Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[21] John Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[22] Foster Provost and Tom Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[23] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning Journal*, 42(3):203–231, March 2001.

[24] Foster Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the 15th International Conference on Machine Learning*, pages 445–453, 1998.

[25] Jefferson Provost. Naive-bayes vs. rule-learning in classification of email. Technical report, Dept. of Computer Sciences at the U. of Texas at Austin, 1999.

[26] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[27] Joseph J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART retrieval system: experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, US, 1971.

[28] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[29] Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, US, 2001. Association for Computational Linguistics, Morristown, US.

[30] G. Salton. *Automating text processing: the transformation, analysis and retrieval of information by computer*. Addison-Wesley, 1989.

[31] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 2002. Forthcoming.

[32] K.M. Ting. Inducing cost-sensitive trees via instance weighting. In *Proceedings of The Second European Symposium on Principles of Data Mining and Knowledge Discovery, LNAI-1510*, pages 139–147, 1998.

[33] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[34] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 1997.

[35] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2), 1999.