Evaluating Cost-Sensitive Unsolicited Bulk Email Categorization

José M^{a} Gómez Hidalgo[1], Enrique Puertas Sanz[1], Manuel J. Maña López[2]

[1]Departamento de Inteligencia Artificial – Universidad Europea - CEES – 28670 Villaviciosa de Odón – Madrid – Spain – {jmgomez, epuertas}@dinar.esi.uem.es

[2]Departamento de Informática – Universidad de Vigo – 32004 Campus As Lagoas s/n – Ourense – Spain – mjlopez@uvigo.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, Rocchio and KNN), made cost sensitive through several methods (Threshold Optimization, Instance 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.

Keywords: Text Categorization, Cost-sensitive Learning, Machine Learning, Unsolicited Bulk Email, Receiver Operating Characteristic Convex Hull, C4.5, Naive Bayes, PART, Support Vector Machines, Rocchio, KNN, Threshold Optimization, Instance Weighting, MetaCost.

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 (Gauthronet and Drouard, 2001).

Several proposals have been made to alleviate the UBE problem, ranging from technical to regulatory and economic (Cranor and LaMacchia, 1998). Among the technical mechanisms proposed, filtering is specially promising and currently in use widely. A popular approach is filtering email using the message features, in contrast to other filtering mechanisms like channels and aliasing. 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 (Brightmail, Inc., 2000). 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 (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Carreras and Márquez, 2001; Drucker et al., 1999; Gómez et al., 2000; Katirai, 1999; Pantel and Lin, 1998; Provost, 1999; Sahami et al., 1998; Sakkis et al., 2001). 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. (Katirai, 1999; Pantel and Lin, 1998; Provost, 1999; Sahami et al., 1998)) 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. (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Carreras and Márquez, 2001; Drucker et al., 1999; Sakkis et al., 2001)) have considered asymmetric misclassification costs, but restricted to scenarios that do not correspond to real world conditions, this is, how tolerant the final user will be to misclassification, 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, probabilistic method Naive Bayes, the rule learner PART, the relevance feedback Rocchio algorithm (Rocchio, 1971), and the kernel method called Support Vector Machines (SVM). Our evaluation has been performed using the Receiver Operating Characteristics Convex Hull (ROCCH) method (Provost and Fawcett, 2001). 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 (Witten and Frank, 1999), suitable for algorithms that output numeric predictions (typically probabilities), and the algorithm-independent Instance Weighting and MetaCost (Domingos, 1999) methods. 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.

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

- To our surprise, the method for making ML algorithms cost-sensitive is Instance Weighting. This method is equivalent to stratification by oversampling, which was proven worse in (Domingos, 1999) for a large sample of common ML benchmark collections.
- The SVM algorithm trained with the Instance 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 (eTesting Labs, Inc., 2001).

The rest of the paper is organized as follows. First, we present the categorization environment, including the data collection and preparation. After, we give an overview of the algorithms and cost-sensitive method tested in this work. In the next section, a discussion of the evaluation approach is presented, focused on the ROCCH method. Following this, we describe the results of our experiment, and we conclude with our conclusions and future work.

2. Text Collection Selection and Preparation

As we have stated in the introduction, we address UBE detection as Text Categorization. The main points in a TC system are text representation and dimmensionality reduction, the selection of the learning algorithms, and the evaluation of the induced classifiers (Sebastiani, 2002).

2.1. Test Collection

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[2] , PU1[3] , and Spambase[4] . Ling-spam contains messages from the archives in the Linguist List and public UBE. PU1 contains 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 form (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.

No collection except Ling-spam and PU1 have been used in two or more papers. Ling-spam has been used in (Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a), and PU1 has been used in (Androutsopoulos et al., 2000b; Carreras and Márquez, 2001). 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. We must also note that the percentage of UBE messages in the test collections is highly variable, ranging from 16,6% in (Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a) to 88,2% in (Sahami et al., 1998). 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 (Cranor and

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

LaMacchia, 1998). In this latest study, a 10% of UBE is reported for corporations, and only a 2% for ISPs. The amount of UBE may vary from ISP to ISP, from corporation to corporation, etc.

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

2.2. Text Representation

The most often adopted representation of the messages in the literature is as term weight vectors, in the Vector Space Model (VSM) (Salton, 1989). The terms are usually words and exceptionally trigrams (as in (Pantel and Lin, 1998). 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* (Drucker et al., 1999), or ad-hoc (Gómez et al., 2000; Provost, 1999). 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.

There are 4 versions of the Ling-spam test collection. We have used the one that comes with terms stemmed using the *morph* lematizer and without the 100 most frequent words in the British National Corpus. The collection has been processed using the Smart (v.11) retrieval engine[5].

2.3. 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 (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Drucker et al., 1999; Katirai, 1999; Sahami et al., 1998; Sakkis et al., 2001), and sometimes according to ad-hoc metrics (Gómez et al., 2000; Pantel and Lin, 1998). Sometimes, term selection is not performed (Carreras and Márquez, 2001; Provost, 1999). When applied, the final number of terms is not standard (even in relation to the initial number of terms). Androutsopoulos et al. (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c) test several numbers of terms (from 50 to 700 in steps of 50) selected according to IG defined as:

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

Being u the UBE class and l the legitimate email class. The original Ling-spam test collection had around 5000 different terms, and we have selected the 500 top IG scoring. Each message has been

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

represented as a term weight vector, in which the weights were binary, except for the Rocchio algorithm, which better suits *tf·idf* weights.

3. Learning cost sensitive UBE classifiers

3.1. Learning Algorithms

A wide variety of learning approaches have been applied to UBE Categorization, including Naive Bayes (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Gómez et al., 2000; Katirai, 1999; Pantel and Lin, 1998; Provost, 1999; Sahami et al., 1998; Sakkis et al., 2001), Ripper (Drucker et al., 1999; Pantel and Lin, 1998; Provost, 1999), k-Nearest-Neighbors (KNN) (Androutsopoulos et al., 2000c; Gómez et al., 2000), boosted C4.5 (Carreras and Márquez, 2001; Drucker et al., 1999), Rocchio and linear Support Vector Machines (SVM) (Drucker et al., 1999), C4.5 and PART (Gómez et al., 2000), Genetic Algorithms (Katirai, 1999), and Stacking applied to Naive Bayes and KNN. The most often applied learner is the probability-based classifier Naive Bayes. Due to several reasons, 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. The algorithms tested in this study are (Witten and Frank, 1999):

- The C4.5 decision tree learner which 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, in which the posterior probability of an instance belonging to a class is computed from the prior probability, using Laplacean estimators.
- The rule learner PART 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), which are maximum margin hyperplanes that attempt to separate training instances, built through Platt's sequential minimal optimization algorithm using polynomial kernels. For efficiency, the computation has been restricted to the linear case.
- The Rocchio algorithm (Rocchio, 1971), 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 KNN algorithm, an instance based classifier that classifies new instances by comparing them to the K nearest neighbours in the training collection. We have used K=10

The implementation of the previous algorithms except Rocchio that we have used for this work is the one provided in the WEKA package[6]. 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.

3.2. Making algorithms cost-sensitive

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

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

- The Threshold method (Witten and Frank, 1999), described below.
- The Instance Weighting method, 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 (Domingos, 1999). 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 (Domingos, 1999), 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 WEKA, with default parameter values.

4. 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, that includes 21578 news stories classified according to a set of 135 economic subject codes (Sebastiani, 2002). A number of learning algorithms have been tested and compared on Reuters (Sebastiani, 2002; Yang, 1999). The most used performance metrics in TC evaluation are F_1 and the breakeven point (discussed below) (Sebastiani, 2002; Yang, 1999). Evaluation is less standardized in UBE Categorization. The evaluation metrics include accuracy, miss and false alarm rates, recall and precision, and cost oriented measures like total cost ratio. In this section, we first discuss the topic of evaluation metrics, and then we describe the ROCCH method, followed in our experiments.

4.1. 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. 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 (–). 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 (Sebastiani, 2002) 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. F_1 is an average of recall and precision. Recall and precision metrics have been used in some of the works in UBE Categorization (e.g. (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Gómez et al., 2000; Sahami et al., 1998; Sakkis et al., 2001)). Other works make use of standard ML metrics, like accuracy and error (Katirai, 1999; Pantel and Lin, 1998; Provost, 1999). 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 (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Carreras and Márquez, 2001; Drucker et al., 1999; Gómez et al., 2000; Sakkis et al., 2001). 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).

The situation of unequal misclassification costs has been observed in many other ML domains, like fraud and oil spills detection (Provost and Fawcett, 2001). 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. (Androutsopoulos et al., 2000c)). 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:

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

The WA and WE metrics are version of the standard accuracy and error measures, that penalize those mistakes that are dispreferred. 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 equivalent. These metrics have been calculated for a variety of classifiers, in three scenarios corresponding to three CR values (1, 9 and 999) (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000c; Androutsopoulos et al., 2000a; Carreras and Márquez, 2001; Gómez et al., 2000; Sakkis et al., 2001).

The main problem presented in the literature on UBE cost-sensitive categorization is that the CR used do not correspond to real world conditions, 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.

4.2. 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 (Provost and Fawcett, 2001). 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.

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 (Provost and Fawcett, 2001). 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 (Provost and Fawcett, 1997). 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 (Provost and Fawcett, 1997). 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 includes the following steps:

1. For each ML algorithm, obtain a ROC curve and plot it 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.

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 (Provost and Fawcett, 2001; Witten and Frank, 1999): 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, 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. 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).

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 1: 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 two letters correspond to the cost-sensitivity method, and the number corresponds to the CR (an "i" means 1/n instead of n). AllPos and AllNeg represent the trivial acceptor and rejector, respectively.

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 (Witten and Frank, 1999). 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): first we obtain a (FP,TP) point for each of the class and cost conditions by ten-fold cross-validation; then we discard the points that do not belong to the convex hull; and finally, we link the remaining (FP,TP) points. 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, leading to 43 points per curve.

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 2: Summary of results for all algorithms and methods, following the codification in table 1.

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 3: Best results of our experiments for the scenarios discussed in the literature. Recall (R) and Precision (P) for the UBE class, Weighted Accuracy (WA) and Total Cost Ratio (TCR) are presented.

5. 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. We have calculated ROC curves for the six ML algorithms and the three cost-sensitivity methods, excluding the combinations Rocchio + Instance Weighting (since Rocchio works averaging, it makes no sense) and Rocchio + MetaCost (MetaCost is not siutable for stable learners). Also we have not applied Instance Weighting nor MetaCost to KNN.

5.1. Cost-sensitive Methods

The results for the C4.5, Naive Bayes, PART and SVM algorithms are shown in the table 1. 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), Instance Weighting is the most frequent optimal method (except for PART).

5.2. Learning Algorithms

In figure 1, the ROCCH for each algorithm are presented, including the ROC curves for the Rocchio + Threshold and KNN + Threshold combinations. In the table 2, 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.

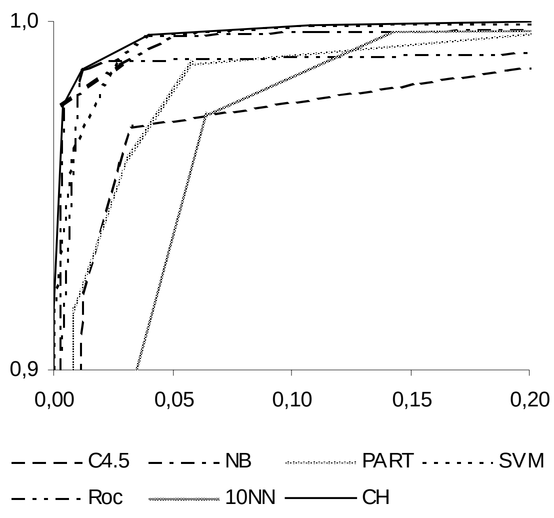

Figure 1: Results for the experiments. For each of the algorithms and cost-sensitivity methods, the ROC convex hull curves are presented, along with the convex hull of all of them.

5.3. 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 calculated. In the table 3 we present the best results obtained in our experiments. Given the class distribution in Ling-spam (16.6% of UBE messages), 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 precision 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. The second and third rows in table 3 show performance numbers that approach best results reported in real-world studies (eTesting Labs, Inc., 2001), 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.

6. 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 through 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 (Gómez et al., 2000; Sahami et al., 1998). 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.

References

Androutsopoulos I., Koutsias J., Chandrinos K., Paliouras G., and Spyropoulos C. (2000a). An evaluation of naive bayesian anti-spam filtering. *ECML 2000*.

Androutsopoulos I., Koutsias J., Chandrinos K. V., and Spyropoulos C. D. (2000b). An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. *SIGIR-00*.

Androutsopoulos I., Paliouras G., Karkaletsis V., Sakkis G., Spyropoulos C., and Stamatopoulos P. (2000c). Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. *PKDD 2000.*

Brightmail, Inc. (2000). Brightmail solution suite technical white paper. Brightmail, Inc.

Carreras X. and Márquez L. (2001). Boosting trees for anti-spam email filtering. *RANLP-2001.*

Cranor L. F. and LaMacchia B. A. (1998). Spam! *Comm. of the ACM*, 41(8).

Domingos P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proc. of the 5th International Conference on Knowledge Discovery and Data Mining*.

Drucker H., Vapnik V., and Wu D. (1999). Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Neural Networks*.

eTesting Labs, Inc. (2001). Brightmail, inc. anti-spam service: Comparative performance test. Technical report, eTesting Labs, Inc., a Ziff Davis Media Inc. company.

Gauthronet S. and Drouard E. (2001). Unsolicited commercial communications and data protection. Technical report, Commission of the European Communities.

Gómez Hidalgo J.M., Maña López M., and Puertas Sanz E. (2000). Combining text and heuristics for cost-sensitive spam filtering. *CoNLL-2000*.

Katirai H. (1999). Filtering junk e-mail: A performance comparison between genetic programming & naive bayes. Available at http://citeseer.nj.nec.com/katirai99filtering.html.

Pantel P. and Lin D. (1998). Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the AAAI Workshop.*

Provost F. and Fawcett T. (2001). Robust classification for imprecise environments. *Machine Learning Journal*, 42(3):203–231.

Provost F. (1999). Naive-bayes vs. rule-learning in classification of email. Technical report, Dept. of Computer Sciences at the U. of Texas at Austin.

Rocchio J. J. (1971). Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*.

Sahami M., Dumais S., Heckerman D., and Horvitz E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop.*

Sakkis G., Androutsopoulos I., Paliouras G., Karkaletsis V., Spyropoulos C. D., and Stamatopou-los P. (2001). Stacking classifiers for anti-spam filtering of e-mail. *EMNLP-01*.

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

Sebastiani F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*. Forthcoming.

Witten I. H. and Frank E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Yang Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval Journal.*