5-2007

Learning to classify e-mail

Irena KOPRINSKA

Josiah POON

James CLARK

Jason Yuk Hin CHAN
Singapore Management University, jasonchan@smu.edu.sg

Citation

KOPRINSKA, Irena; POON, Josiah; CLARK, James; and CHAN, Jason Yuk Hin. Learning to classify e-mail. (2007). *Information Sciences*. 177, (10), 2167-2187.
Available at: https://ink.library.smu.edu.sg/sis_research/7703

Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 177 (2007) 2167–2187

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Learning to classify e-mail

Irena Koprinska [*], Josiah Poon, James Clark, Jason Chan

School of Information Technologies, The University of Sydney, Sydney, Australia

Received 9 March 2006; received in revised form 31 August 2006; accepted 12 December 2006

Abstract

In this paper we study supervised and semi-supervised classification of e-mails. We consider two tasks: filing e-mails into folders and spam e-mail filtering. Firstly, in a supervised learning setting, we investigate the use of random forest for automatic e-mail filing into folders and spam e-mail filtering. We show that random forest is a good choice for these tasks as it runs fast on large and high dimensional databases, is easy to tune and is highly accurate, outperforming popular algorithms such as decision trees, support vector machines and naïve Bayes. We introduce a new accurate feature selector with linear time complexity. Secondly, we examine the applicability of the semi-supervised co-training paradigm for spam e-mail filtering by employing random forests, support vector machines, decision tree and naïve Bayes as base classifiers. The study shows that a classifier trained on a small set of labelled examples can be successfully boosted using unlabelled examples to accuracy rate of only 5% lower than a classifier trained on all labelled examples. We investigate the performance of co-training with one natural feature split and show that in the domain of spam e-mail filtering it can be as competitive as co-training with two natural feature splits.

Keywords: E-mail classification into folders; Spam e-mail filtering; Random forest; Co-training; Machine learning

1. Introduction

E-mail is one of the most popular, fastest and cheapest means of communication. It has become a part of everyday life for millions of people, changing the way we work and collaborate [43]. E-mail is not only used to support conversation but also as a task manager, document delivery system and archive [13,43]. The downside of this success is the constantly growing volume of e-mail we receive. To deal with this problem and facilitate efficient retrieval when needed, many people categorize e-mails and file them into folders. For more information about the cognitive aspects of information organization and retrieval in e-mail see [13]. Prioritizing e-mail according to its importance, and possibly sending the most important messages to a mobile device, is another desirable characteristic of an e-mail management system. Finally, the rate of spam e-mail is also rapidly growing. MessageLabs report that in October 2005 spam e-mails were 68% of all e-mails [28]. Some of the spam e-mails are

[*] Corresponding author. Tel.: +61 2 9351 3764; fax: +61 2 9351 3838.
E-mail addresses: irena@it.usyd.edu.au (I. Koprinska), josiah@it.usyd.edu.au (J. Poon), jclark@it.usyd.edu.au (J. Clark), jchan3@it.usyd.edu.au (J. Chan).

0020-0255/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2006.12.005

unsolicited commercial and get-rich messages, others can contain offensive material. Spam e-mails can also clog up the e-mail system by filling-up the server disk space when sent to many users from the same organization.

E-mail packages typically allow the user to hand construct keyword-based rules to automatically file e-mails into folders and filter spam messages. However, most users do not create such rules as they find it difficult to use the software or simply avoid customizing it [8]. In addition, manually constructing a set of robust rules is a difficult task as users are constantly creating, deleting, and reorganizing their folders. Even if the folders remain the same, the nature of the e-mails within a folder may well change over time. The characteristics of the spam e-mail (e.g. topics, frequent terms) also change over time as the spammers constantly invent new strategies to deceive filters. Hence, the rules must be constantly tuned by the user which is a time consuming and error-prone process. A system that can automatically learn how to classify e-mails into a set of folders is highly desirable.

In this paper we consider two e-mail classification tasks: automatic e-mail filing into folders and spam e-mail filtering. These tasks are formulated as supervised and semi-supervised machine learning problems. In a supervised setting, given a supervision in the form of a set of labelled training examples (e.g. e-mails labelled as belonging to different folders such as work, teaching etc, or as spam and non-spam), the goal is to build a classifier, that is then used to predict the category of an unseen incoming e-mail. In a semi-supervised setting, the goal is the same but the learning is achieved with less supervison, i.e. using a small set of labelled e-mails, by taking advantage of the easily available unlabelled e-mails. This is motivated by the need to reduce the large number of manually labelled examples that are required to build accurate classifiers using supervised learning. We consider a specific semi-supervised paradigm, called co-training.

The contribution of this paper is as follows:

- In both supervised and semi-supervised setting for e-mail classification we study the application of Random Forest (RF), which is one of the recent ensemble techniques. RF is particularly suitable for classifying text documents as it is fast, easy to tune, and can handle large feature sets. We compare the learning behaviour of RF and well established algorithms such as Support Vector Machines (SVM), Naïve Bayes (NB) and Decision Trees (DT), and show that RF outperforms them in both settings.
- We introduce a new feature selector TFV and show that it outperforms the widely used and computationally more expensive IG.
- We study the portability of anti-spam filter across users.
- We conduct a large scale algorithm performance evaluation on the benchmark spam filtering corpora LingSpam and PU1. We compare RF, DT, SVM, NB, stacking and boosted DTs using the same version of the corpora, the same pre-processing and optimising the parameters of all algorithms. The last four algorithms have been previously applied but using different pre-processing and on different versions of the corpora which did not allow fair comparison.
- We investigate the performance of semi-supervised co-training using RF, SVM, NB and DT as base classifiers. We show empirically that co-training can be successfully used to learn from a small number of labelled examples in the domain of spam filtering.
- Conventional semi-supervised co-training requires the dataset to be described by two disjoint natural feature sets. Most datasets have a single set of features which limits the applicability of co-training. We show that if there is high data redundancy as in the domain of spam e-mail filtering, co-training with random feature split is as competitive as co-training with natural feature split.

The paper is organised as follows. Section 2 discusses supervised classification of e-mails into folders and spam filtering. Section 3 investigates the semi-supervised co-training classification for spam e-mail filtering. Section 4 concludes the paper.

2. Supervised learning for e-mail classification

2.1. Problem statement

E-mail classification (e.g. filing e-mails into folders, spam filtering) is a supervised learning problems. It can be formally stated as follows. Given a training set of labelled e-mail documents $D_{\text{train}} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$, where d_i is an e-mail document from a document set D and c_i is the label chosen from a predefined set of categories C, the goal is to induce a hypothesis (classifier) $h : D \rightarrow C$ that can correctly classify new, unseen e-mail documents $D_{\text{test}}, D_{\text{test}} \not\subset D_{\text{train}}$. In the first task, C consists of the labels of the folders. In the second task, C contains two labels: spam and non-spam (legitimate).

2.2. Previous work

E-mail classification has been an active area of research. Cohen [11] developed a propositional learning algorithm RIPPER to induce "keyword-spotting rules" for filing e-mails into folders. The multi-class problem was transformed into several binary problems by considering one folder versus all the others. The comparison with the traditional information retrieval method Rocchio indicated similar accuracies. Cohen argued that keyword spotting rules are more useful as they are easier to understand, modify and can be used in conjunction with user-constructed rules.

Bayesian approaches are the most widely used in text categorization and e-mail classification. They allow quick training and classification and can be easily extended to incremental learning. While rule-based approaches make binary decisions, probabilistic techniques provide a degree of confidence of the classification, which is an advantage, especially for cost-sensitive evaluation. Sahami et al. [37] applied NB for spam e-mail filtering using bag of words representation of the e-mail corpora and binary encoding. The performance improved by the incorporation of hand-crafted phrases (e.g. "FREE!", "be over 21") and domain-specific features such as the domain type of the sender and the percentage of non-alphabetical characters in the subject. Rennie's iFile [35] uses NB to file e-mails into folders and suggest the three most suitable folders for each message. The system applies stemming, removes stop words and uses document frequency threshold as feature selector. SpamCop [31] is a system for spam e-mail filtering also based on NB. Both stemming and a dynamically created stop word list are used. The authors investigated the effect of the training data size, different rations of spam and non-spam e-mails, use of trigrams instead of words and also show that SpamCop outperforms Ripper. Provost's experiments [33] also confirmed that NB outperforms Ripper in terms of classification accuracy on both filing e-mail into folders and spam filtering.

MailCat [40] uses a nearest-neighbor (k-NN) technique and tf-idf representation to file e-mails into folders. K-NN supports incremental learning but requires significant time for classification of new e-mails. Androutsopoulos et al. [4] found that Naïve Bayes and a k-NN technique called TiMBL clearly outperform the keyword-based spam filter of Outlook 2000 on the LingSpam corpora.

Ensembles of classifiers were also used for spam filtering. Sakkis et al. [38] combined a NB and k-NN by stacking and found that the ensemble achieved better performance. Carreras et al. [8] showed that boosted trees outperformed decision trees, NB and k-NN. Rios and Zha [36] applied RF for spam detection on time indexed data using a combination of text and meta data features. For low false positive spam rates, RF was shown to be overall comparable with SVM in classification accuracy.

We extend previous research on supervised e-mail classification by: (1) applying RF for both filing e-mails into folders and filtering spam e-mails, comparing RF with a number of state-of-the-art classifiers and showing that it is a very good choice in terms of accuracy, running and classification time, and simplicity to tune, (2) introducing a new feature selector that is accurate and computationally efficient, (3) studying the portability of an anti-spam filter across different users, and (4) comparing the performance of a large number of algorithms on the benchmark corpora for spam filtering using the same version of the data and the same pre-processing.

2.3. Classification algorithms

2.3.1. RF

An ensemble of classifiers combines the decisions of several classifiers in some way in an attempt to obtain better results than the individual members. Recent research has shown that an effective ensemble consists of individual classifiers which are highly correct and disagree as much as possible on the input data predictions [19,25]. Thus, methods for creating ensembles focus on generating disagreement amongst the ensemble members, before combining their decisions with majority voting. Some ensemble methods exploit the diversity within the classification algorithm, e.g. training neural network classifiers with different topologies, different parameters, different training algorithms or different initial weights, and then combining them. Other ensemble methods alter the training data for each ensemble member by using different pre-processing methods, introducing noise or using different data sources. Typical examples of this approach are the widely used bagging [7] and boosting [17] that generate disagreement amongst its individual members by randomly or iteratively changing the training sets. A third group of approaches generate diversity by using different feature vectors for each ensemble member, e.g. different sub-sets of the original features [20].

RF [6] is a recently introduced ensemble approach that combines decision trees [34]. The diversity in the individual trees of RF is generated by both altering the data set using bagging and selecting random input features. The RF algorithm is summarized in Table 1.

If n is the number of training examples and m is the number of features (attributes) in the original training data, the training data for each of the t ensemble members is first generated by randomly selecting n instances from the training data with replacement to form the bootstrapped samples. Then, for each data sample, a DT is grown. When growing a typical DT, splits on all available attributes for a given node will be considered and the best one will be selected based on performance indexes such as Gini index or Gain ratio. In RF, only a small number $k(k \ll m)$ of randomly selected features, available at the node, are searched. The number of features k is kept fixed but for each split a new random set of features of size k is selected. Each tree is fully grown and not pruned as opposite to the standard DT algorithm where pruning is typically applied. To classify a new example, it is propagated through all t trees and the decision is taken by majority voting. In essence, RF is a bagging ensemble of random trees.

The predictive performance of RF depends on the strength of the individual trees and their correlation with each other. A tree with high strength has a low classification error. Ideally we would like the trees to be less correlated and highly accurate. As the trees become less accurate or correlated, the RF's performance decays. The low level of correlation is achieved by using bagging and random feature selection which inject randomness in the RF algorithm and it generates dissimilar, and thus, low-correlated, trees. The strength and correlation are also dependant on the number of features k. As k increases, both the correlation among the trees and their accuracy tend to increase. As a trade-off, the value of k is typically set to $k = \log_2 m + 1$ [6].

RF are able to overcome one of the biggest disadvantages of single decision trees – instability. Very often small changes in the data result in a very different tree and big changes in predictions. The reason for this is the hierarchical process of tree building – the errors made in the splits close to the root are carried down the entire tree. RF reduces this variance by averaging the results of many decision trees.

It has been shown empirically that RF outperform single DT [41] in terms of classification accuracy. RF are typically faster than standard DT as they consider less number of features when selecting an attribute and do not involve pruning. RF has also been shown to run much faster and give comparable accuracy results to the highly successful AdaBoost ensemble algorithm [6]. Breiman also proved that RF does not overfit. The ability to run efficiently on large data sets and produce accurate results makes RF a very attractive algorithm for text categorization.

Table 1
RF algorithm

Given:

n – number of training examples, m – number of all features, k – number of features to be used in the ensemble, t – number of ensemble members

Create Random Forest (RF) of *t* trees:

For each of t iteration

1. Bagging
   Sample n instances with replacement from training data
2. Random feature selection
   Grow decision tree without pruning. At each step select the best feature to split on by considering only k randomly selected features and calculating the Gini index

Classification:

Apply the new example to each of the t decision trees starting from the root. Assign it to the class corresponding to the leaf. Combine the decisions of the individual trees by majority voting

RF should not be confused with Decision Forest (DF) that are a random-subspace approach. DF generates diversity among the trees by selecting different feature sub-sets for each ensemble member. The training data for each tree is created by pseudo-randomly selecting k features and then growing a DT using only these features. More details about our implementation and application of DF to e-mail classification can be found in [23].

In our experiments the RF's parameters were set to $k = 9$ randomly selected features and $t = 10$ trees.

2.3.2. Algorithms used for comparison

We compare the performance of RF with DT, SVM and NB classifiers. The purpose of the comparison with DT is to test if the ensemble of random, unpruned trees, using only some of the features, is able to outperform a single pruned decision tree, using all features. SVM was chosen as one of the best performing classifiers in text categorization. NB is the most frequently used classifier in e-mail classification. Below is a brief description of the algorithms we used and how we optimised their parameters. We used their Weka implementations [2].

SVM [42] is a very popular machine learning technique. It finds the maximum margin hyperplane between two classes using the training data and applying an optimization technique. The decision boundary is defined by a sub-set of the training data, the so-called support vectors. A multi-class problem is transformed into multiple binary sub-problems. SVM has shown good generalization performance on many classification problems, including text categorization [14]. What makes it suitable for text categorization is its ability to handle high dimensional features [21]. The Weka's implementation of SVM is based on the Platt's sequential minimization optimization algorithm for training of the classifier [32].

DT is the most popular inductive learning algorithm [34]. The nodes of the tree correspond to attribute tests, the links – to attribute values and the leaves – to the classes. To induce a DT, the most important attribute (according to IG) is selected and placed at the root; one branch is made for each possible attribute value. This divides the examples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all instances at a node have the same classification, in which case a leaf is created. To classify an example we start at the root of the tree and follow the path corresponding to the example's values until a leaf node is reached and the classification is obtained. To prevent overtraining DTs are typically pruned.

NB [29] is a simple but highly effective Bayesian learner. It uses the training data to estimate the probability that an example belongs to a particular class. It assumes that attribute values are independent given the class. Although this assumption clearly has no basis in many learning situations including text categorization, Naïve Bayes can produce very good results.

We run some preliminary experiments to choose the parameters of the classifiers. The following setting were used in all experiments: SVM – the default options and polynomial kernel of the SMO classifier; DT and NB – default options of the J48 and Naïve Bayes classifiers.

2.4. Data and pre-processing

2.4.1. E-mail corpora

2.4.1.1. Filing into folders. We used the e-mail corpus collected by Crawford et al. [12] that contains messages from four different users (U1-4). In addition, we also used the e-mail of one of the authors (U5), collected over one year. All users kept folders and e-mails that contained sensitive private information. The characteristics of the corpus are given in Table 2.

2.4.1.2. Spam filtering. We used three corpora: LingSpam, PU1 and U5Spam, as shown in Table 3. The first two are publicly available [1] while the third one is a subset of the U5 corpus. LingSpam [4] was created by mixing spam e-mails received by a user with legitimate e-mails sent to the Linguist mailing list. PU1 [3] consists of e-mails received by a user over 3 years. A representative dataset was constructed taking into account the e-mails received by regular correspondents and removing duplicate e-mails received on the same day. The corpus was encrypted for privacy reasons by replacing each token with a number.

Table 2
Filing into folders corpus – statistics

User	# E-mails	# Folders	Folder size (min–max # e-mails)
U1	545	7	10–326
U2	423	6	4–418
U3	888	11	14–206
U4	926	19	4–507
U5	982	6	42–337

Table 3
Spam filtering corpora – statistics

Corpus	# E-mails	# Spam e-mails	# Legitimate e-mails
PU1	1099	481	618
LingSpam	2893	481	2412
U5Spam	982	82	900

An important characteristic of the LingSpam corpus is that the topics of the legitimate e-mails are linguistically specific, although probably more varied than one might expect [4] as they contain, for example, job, conference, software and other announcements. In contrast, the topics of the legitimate e-mails in U5Spam and PU1 are not topic specific and more diverse. This implies that it should be easier to learn to discriminate spam and legitimate e-mail on the LingSpam corpus than on PU1 and U5Spam.

2.4.2. Pre-processing of the corpora

The first step in the process of constructing an automatic classifier is the transformation of the e-mails into a format suitable for the classification algorithms. We have developed a generic architecture for text categorization, called LINGER [10]. It supports the *bag of words* representation which is the most commonly used in text categorization. All unique terms (tokens, e.g. words, special symbols, numbers etc.) in the entire training corpus are identified and each of them is treated as a single feature. A feature selection mechanism is applied to choose the most important terms and reduce dimensionality. Each document is then represented by a vector that contains a normalized weighting for every selected term, which represents the importance of that term in the document.

2.4.2.1. Term extraction in the U1-5 corpora. In addition to the body of the e-mail, the following headers were parsed and tokenized: *Sender* (*From*, *Reply-to*), *Recipient* (*To, CC, Bcc*) and *Subject*. Attachments are considered part of the body and are processed in their original format (binary, text, html). All these fields were treated equally and a single bag of words was created for each e-mail. The following symbols were used as token delimiters and then discarded: <> ()[]/\ | −_#% ∧ &∗, . :; @ ∼ ‘+ = ‴. Three other symbols (!, $, ?) were used as delimiters and then kept as they often appear in spam e-mail. No stemming and stop word list were applied.

Next, words that only appear once in each corpus were discarded. Finally, words longer than 20 characters were removed from the e-mail's body. Such long words are usually strings in binary attachments. As a result, the initial number of unique terms is reduced from about 9000 to 1000 for each corpus.

2.4.2.2. Term extraction in LingSpam and PU1. The processing of LingSpam and PU1 is described in [4,3]. Only the *Body* and *Subject* were used, discarding the other headers, attachments and HTML tags. This may be a disadvantage as the *Sender* fields, attachments and HTML tags provide useful information for the classification. For example, Graham [18] claims that the use of information from the headers is probably the main reason why his Bayesian filter outperformed the one used in SpamCop [31]. Both LingSpam and PU1 are available in four versions depending on whether stemming and stop word list were used. Our previous experiments [10] have shown similar results on all four versions. In this paper we report results on the *lemm* version (only stemming used) as it is the most widely used version.

2.4.2.3. Features, weighting and normalization. Feature selection is an important step in text categorization as text documents have a large number of terms. Removing the less informative and noisy terms reduces the computational cost and improves the classification performance. The features are ranked according to the feature selection mechanism and those with value higher than a threshold are selected. LINGER supports two feature selectors: Information Gain (IG) and Term Frequency Variance (TFV).

IG [27] is the most popular and one of the most successful feature selection techniques used in text categorization. Given a set of possible categories $C = \{c_1, \ldots, c_k\}$, the IG of a feature f is defined as

$$\mathrm{IG}(f) = -\sum_{i=1}^{k} P(c_i)\log P(c_i) + P(f)\sum_{i=1}^{k} P(c_i|f)\log P(c_i|f) + P(\bar{f})\sum_{i=1}^{k} P(c_i|\bar{f})\log P(c_i|\bar{f})$$

It measures how much knowing if f is present or absent in a document helps us to predict the category. The importance of the feature is measured globally as the computation is done for each feature across all categories. IG has quadratic time complexity.

TFV is feature selector that we have developed. Like IG, it is category dependant. For each term f we compute the term frequency (tf) in each category and then calculate the variance as

$$\mathrm{TFV}(f) = \sum_{i=1}^{k}[\mathrm{tf}(f, c_i) - \mathrm{mean\_tf}(f)]^2$$

The normalising factor from the standard variance formula is ignored, as our goal is to rank features and select the ones with the highest score. Features with high variance across categories are considered informative and are selected. For example, terms that occur predominantly in some of the categories will have high variance and terms that occur in all categories will have low variance. TFV can be seen as an improvement of the document frequency, which is the simplest method for feature reduction. For each term in the training corpora, document frequency counts the number of documents in which the term occurs and selects features with frequency above a predefined threshold. The assumption is that rare terms are not informative for category prediction. Yang and Pedersen [44] compared several feature selectors and found that document frequency is comparable to the best performing techniques (IG and χ^2 for feature reduction up to 90%. They concluded that document frequency is not just an ad hoc approach but a reliable measure for selecting informative features. However, document frequency is category independent and will simply select terms with high DF no matter of their distribution in the categories. TFV addresses this problem by not selecting terms with high document frequency if they appear frequently in each category, i.e. are not discriminating. Both TFV and document frequency are highly scalable as they have linear complexity.

LINGER incorporates the three most popular *feature weighting* mechanisms: (1) binary, (2) term frequency and (3) term frequency, inverse document frequency (*tf-idf*). In the first method weights are either 0 or 1 denoting absence or presence of the term in the e-mail. In the term frequency method the weights correspond to the number of times the feature occurs in the document. Term frequency weights are more informative than the binary weights, e.g. knowing that "money" occurs 10 times in an e-mail is a better indicator that the e-mail is spam than knowing that "money" simply occurs in this e-mail. The third method assigns higher weights to features that occur frequently, but also balances this by reducing the weight if a feature appears in many documents. In [10], we compared these three weighting schemes, normalised at different levels (e-mail, category and corpora). Term frequency normalised at category level was found to perform best and this is the feature weighting used in this paper.

2.5. Performance measures

To evaluate performance we calculate *accuracy (A), recall (R), precision (P)* and *F1 measure*. Table 4 shows the definition of these measures in terms of the two-way contingency table for a binary classifier.

Accuracy is the most commonly used measure in machine learning. Precision, recall and their combination, the F1 measure, are the most popular criteria used in text categorization. In the multi class task of general mail classification, *macro-averaging* [39] was used – precision, recall and the F1 measure were first calculated for each class and the results were then averaged.

Table 4
Contingency table for binary classifier

E-mails	# Assigned to folder c_i	# Not assigned to folder c_i
# From folder c_i	tp	fn
# Not from folder c_i	fp	tn
$A = \frac{tp+tn}{tp+tn+fp+fn}, P = \frac{tp}{tp+fp}, R = \frac{tp}{tp+fn}, F1 = \frac{2PR}{P+R}$		

In the spam filtering experiments we calculated accuracy, *spam recall (SR), spam precision (SP)* and *spam F1 measure (SF1)*. SR is the proportion of spam e-mails in the test set that are classified as spam, i.e. the spam e-mails that the filter manages to block. It measures the effectiveness of the filter. SP is the proportion of e-mails in the test data classified as spam that are truly spam, i.e. it measures filter's protection (and overprotection) ability.

Discarding a legitimate e-mail is of greatest concern to most users than classifying a spam message as legitimate. This means that high SP is particularly important. As suggested in [3], blocking a legitimate message is considered a bigger error, λ times more costly, than non-blocking a spam message. To make accuracy sensitive to this cost, *weighted accuracy (WA)* was defined: when a legitimate message is misclassified, this counts as λ errors. Similarly, when legitimate message is correctly classified, this also counts as λ successes. More formally, WA is defined as follows:

$$\mathrm{WA} = \frac{\lambda n_{l \to l} + n_{s \to s}}{\lambda N_l + N_s}$$

where N_l is the number of legitimate e-mails, N_s is the number of smap e-mails, $n_{\mathrm{A} \to \mathrm{B}}$ is the number of e-mails belonging to class A classified as belonging to class B, $\mathrm{A}, \mathrm{B} \in \{l, s\}$, l – legitimate, s – spam e-mails.

We also follow the proposed three cost scenarios [3]:

(a) no cost considered ($\lambda = 1$), e.g. flagging messages that are predicted as spam but not blocking them;
(b) semi-automatic scenario for moderately accurate filter ($\lambda = 9$), e.g. notifying senders about their blocked e-mails and asking them to re-send the e-mail following special instructions that will allow the e-mail to pass the filter. The higher value of λ reflects the extra cost of re-sending the e-mail that is considered to be the same as manually deleting nine spam e-mails;
(c) completely automatic scenario for a highly accurate filter ($\lambda = 999$), e.g. removing blocked messages. The very high value of λ reflects the recovery cost when a legitimate e-mail was misclassified as spam and deleted.

It should be noted that there is no research justification for using exactly $\lambda = 9$ or 999. These values depend on the human perception of the associated cost. We used the same values for the reasons of consistency and comparison with previous research.

For evaluation of all results we used *stratified ten-fold cross validation* which is known to provide a good estimate of the generalization error of a classifier. Each corpus is split into 10 non-overlapping segments of equal size. Each segment is stratified, i.e. contains approximately the same number of examples from each class. A classifier is trained 10 times, each time using a version of the data in which one of the segments is omitted (training data). Each trained classifier is then tested on the data from the segment which was not used during training. The results are averaged over the 10 classifiers to obtain an overall error estimate.

2.6. Results and discussion

In all experiments IG and TFV were used as feature selectors and the best scoring 256 features were chosen. *Tf* weighting with category (folder) level normalization was applied.

2.6.1. Filing into folders

2.6.1.1. Overall performance. The classification results are summarized in Table 5. They vary significantly across the users, e.g. the accuracy is between 81% and 96%, and the F1 results are between 42% and 94%

Table 5
Performance of RF on filing e-mails into folders (10 trees, 9 features)

Feature selection	A (%)	R (%)	P (%)	F1 [%]
U1				
RF-TFV	92.11	80.91	89.05	84.21
RF-IG	91.01	77.30	92.29	83.19
U2				
RF-TFV	93.24	70.35	87.95	76.27
RF-IG	90.09	62.63	84.74	67.15
U3				
RF-TFV	81.32	56.53	77.73	58.83
RF-IG	68.32	43.76	68.56	46.16
U4				
RF-TFV	81.97	41.98	48.44	42.56
RF-IG	71.16	31.32	44.06	33.85
U5				
RF-TFV	96.03	92.99	95.62	94.16
RF-IG	93.69	90.94	94.63	92.58

(for the better feature selector TFV). This variation is due to the different classification styles. While U1 and U5 categorize e-mails based on the topic and sender (e.g. colleagues, friends, MachineLearningCourse, thesis), U3 do this based on the action performed (e.g. Delete, Keep, Read & Keep, ToActOn) while U2 and U4 use all strategies – based on the topic, sender and action performed. Thus, some mailboxes of U2 and U4 do not contain semantically similar e-mails but e-mails grouped by action and time, which highly complicate learning. It is interesting to note the relatively high accuracy but low F1 score for U3 and U4. This is due to the imbalanced class distribution. In both cases there is a big folder Read & Delete that contains 1/4 and 1/2 of all e-mails, respectively. Although these e-mails are semantically different, they are classified surprisingly well based on the sender. This highlights the importance of the headers other than *Subject*. As a result, it is classified very accurately based on sender. Another difficulty is the large ratio of the number of folders over the number of e-mails for U2 and U4 which also makes learning difficult as some folders do not contain enough training examples.

To summarise, the results show that the problem of e-mail filing into folders is quite different than the standard text categorization problem. Firstly, we cannot expect that an automatic text-based e-mail system will be helpful for users who file e-mails based on criteria different than content or sender. Thus, an automatic system may perform very well on some users and very badly on others. Secondly, the classification task is highly imbalanced. Some folders contain a small number of e-mails as user activities and interests change over time, while others keep growing. Thirdly, the content of the big folders may change over time and requires addressing the concept drift, as more examples will not lead to better classification. Finally, some of the e-mails can be classified in more than one folder. A good strategy to address this issue was proposed by Rennie [35] who calculates the probabilities for assigning an e-mail to each folder and recommends the three highly ranked folders. The results show that the correct folder is among the three top folders with very high accuracy. An additional advantage is that the users are in control to take the final decision, while the system is assisting them by simplifying the decision making.

2.6.1.2. Comparison with other classifiers. Fig. 1 shows a comparison of RF with DT, SVM and NB classifiers. RF outperforms the other algorithms in 36 out of the 40 cases. DT is the second best algorithm followed by SVM. DTs have been overlooked in the area of text categorization. A big advantage over the other classifiers is that a DT can be converted in set of rules explaining the classification rule. NB is the worst performing algorithm which confirms previous research that NB is a popular but not the best algorithm for text categorization [39].

It should be noted that the total number of features *f* that are seen by RF during learning is much smaller than the number of features seen by the other classifiers. The number of features visited by a tree from RF can

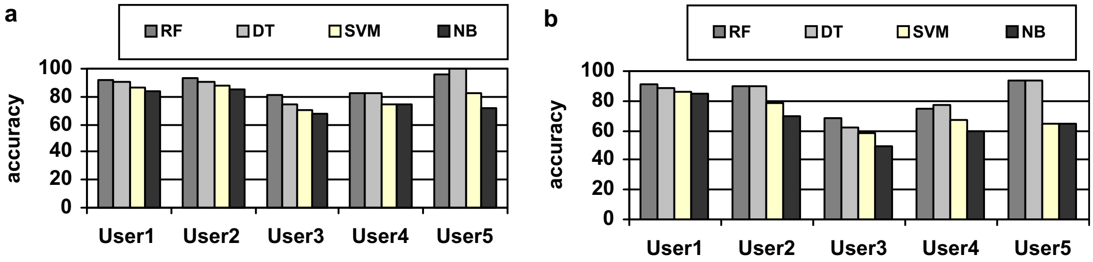

Fig. 1. Classification accuracy [%] on filing e-mails into folders with (a) TFV and (b) IG as feature selectors.

be estimated as $f = N\left(1 - \left(1 - \frac{k}{N}\right)^S\right)$ [24], where N is the total number of features (256 in our case), S is the number of tree nodes and k is the size of the randomly selected features (10 in our experiments). Assuming that $S = 10$, this means that a tree from RF will see only 84 features in comparison to 256 by the other classifiers. This saving is more dramatic for higher N and smaller tree complexity S.

The time taken to build the classifiers is shown in Table 6. It is an important consideration as classifiers must be kept up to date and this requires re-training. The fastest algorithm was NB (although it was the less accurate), the slowest was SVM, almost 10 times slower than NB. RF was fast enough – it built a classifier for 4.71 s on average. It can also be estimated that a tree from RF is built n/k times faster than an unpruned DT which was confirmed in our experiments. The classification time is another important consideration in real-world systems, and all of the algorithms showed good results.

Another important advantage of RF is the small number of tuneable parameters – number of trees and number of random features. In contrast, SVM is a very complex algorithm, with many parameters to adjust. DT requires pruning and smoothing parameters, NB – smoothing parameters.

2.6.1.3. Effect of the feature selector. An important observation is that the simpler feature selection mechanism TFV was more effective than IG. As Fig. 1 shows, the classification accuracy using TFV was better than IG for all users and classifiers (40 combinations in total) except for two cases – SVM on U1 (tie) and NB on U1 (IG better with 0.37%). This result is also confirmed by the F1 scores averaged across all users, see Table 7.

A comparison of the top 100 terms with highest IG and TFV scores shows that in both cases the best scoring features are: (1) terms that are frequent in legitimate, linguistic oriented e-mails, such as "language", "university", "linguistic", "conference", "abstract", "workshop", "theory"; (2) terms that are frequent in

Table 6
Time (s) to build the classifier for TFV (results for IG are similar)

	RF	DT	SVM	NB
U1	3.54	0.79	7.60	0.11
U2	3.68	1.84	16.05	0.21
U3	3.29	1.39	6.69	0.10
U4	6.79	3.63	48.88	0.23
U5	6.26	0.68	9.26	0.32
Average	4.71	1.66	17.70	0.19

Table 7
F1 results (%) averaged across all users for the two feature selectors (TFV and IG)

Classifier	TFV	IG
RF	71.20	64.58
DT	74.59	67.15
SVM	65.92	52.56
NB	64.91	53.85

spam e-mail but not in legitimate e-mails, such as “free”, “money”, “you”, “!”, “$”. TFV rates higher and selects more punctuation and special symbols such as “_”, “*”, “:”, “(“, “)”, “+”, and also action verbs such as “can”, “have”, “make”, “will”, “send”. Sahami et al. [37] also found that spam e-mails contain a larger proportion of non-alphanumeric characters.

2.6.2. Spam filtering

In this section we investigate the performance of RF for spam e-mail classification, compare it with other classifiers, study the effect of the two feature selectors TFV and IG and also the portability of an anti-spam filter across users.

2.6.2.1. Overall performance. The classification results for the *lemm* versions of LingSpam and PU1 are given in Table 8. By comparing WA and SF1 scores, we can see that overall RF is the best classifier, obtaining the most consistent results on the two corpora for both IG and TFV. When IG was used as feature selector, RF achived the best WA for both LingSpam and PU1, and the three cost-sensitive scenarios. When TFV was used as feature selector, the best WA for both corpora and three scenarios was achieved by DT and SVM which classified all examples correctly. In this case, RF came second misclassifying only two non-spam e-mails as spams on LingSpam, and six e-mails on PU1 (four non-spams as spams and two spams as non-spams). The second best classifier is DT. The worst classifier is NB falling behind the winner in terms of SF1 with 10–14% on LingSpam and 12–15% on PU1. NB (and also to the lesser extent SVM-IG) obtained high SP at the expense of lower SR, i.e. misclassified more spams than non-spams. We should keep in mind that for spam filtering SP is more important than SR. Again, overall, TFV was found to be better feature selector than IG.

As expected, it was easier to discriminate between spam and legitimate e-mails in LingSpam because of the topic-specific nature of its legitimate e-mails. A baseline accuracy can be calculated as classifying all test examples to the majority class in the training set (also called ZeroR classifier). The baselines are 83.37% for LingSpam (very high) and 56.23% for PU1. All algorithms perform above the baseline which means that there is no indication for overtraining.

The cost sensitive evaluation on LingSpam shows that all algorithms were insensitive to the higher weight of the spam misclassification. The reason for this is the ceiling effect in LingSpam – very few examples were misclassified. In PU1, WA for $\lambda = 9$ and 999 does not change significantly or goes up in comparison to $\lambda = 1$

Table 8
Performance of RF, DT, SVM and NB on spam filtering (%)

	$\lambda = 1$				$\lambda = 9$	$\lambda = 999$
	WA	SP	SR	SF1	WA	WA
LingSpam						
RF-IG	99.31	98.30	97.50	97.90	99.62	99.67
DT-IG	98.48	95.60	95.20	95.40	99.04	99.13
SVM-IG	96.85	99.00	81.90	89.60	99.45	99.83
NB-IG	96.13	92.80	83.20	87.70	98.38	98.71
RF-TFV	99.94	99.60	100	99.80	99.92	99.92
DT-TFV	100	100	100	100	100	100
SVM-TFV	100	100	100	100	100	100
NB-TFV	95.61	91.70	80.90	86.00	98.17	98.55
PU1						
RF-IG	98.27	97.9	98.1	98.00	98.36	98.38
DT-IG	92.81	90.90	92.90	91.90	92.74	92.72
SVM-IG	93.18	95.70	88.40	91.90	96.24	96.92
NB-IG	86.81	94.00	74.60	83.20	94.54	96.26
RF-TFV	99.45	99.20	99.60	99.40	99.37	99.35
DT-TFV	100	100	100	100	100	100
SVM-TFV	100	100	100	100	100	100
NB-TFV	89.81	97.40	78.80	87.10	96.82	98.37

as the proportion of misclassified legitimate e-mails is much smaller than the proportion of misclassified spams.

We also compare RF with the ensemble approaches used by Sakkis et al. [38] and Carreras and Marquez [8]. Sakkis et al. applied stacking on Lingspam, with NB and k-NN as base classifiers and k-NN as a meta classifier. Carreras and Marquez applied the AdaBoost algorithm to combine decisoon trees on both LingSpam and PU1. To eliminate the effect of possible different pre-processing and ensure fair comparison, we run these algorithms on our versions of LingSpam and PU1 using IG as feature selector, as in their experiments. We also optimised the parameters of the algorithms. Table 9 reports the best results that we achieved, which are actually better than the results reported in [38] and [8]. We used the following parameters: 7 nearest neighbours in stacking, 12 trees for the boosting. Boosted DTs performed as good as RF. AdaBoost, however, was 4 times slower to train than RF (Table 10), and also significantly slower at classification. Stacking performed worse than RF in terms of classification accuracy. It was also the slowest algorithm to run: 8–23 times slower than RF, and the classification time was also very long due to the use of k-NN algorithm. Thus, in the domain of spam filtering RF showed much faster training and classification time than boosted DTs while achieving comparable classification results, which confirms Breiman's results on other datasets [6].

2.6.2.2. Anti-spam filter portability evaluation. Portability of an anti-spam filter is an interesting question and an important issue in real applications. Having a machine learning system that can learn from the e-mail of each user allows us to build a personalised spam filter. Such a filter can be more accurate as it will capture the specific characteristics of the user's legitimate and spam e-mail. Building an accurate personalised filter, however, requires that each user labels a large number of e-mails as legitimate and spam to create a training set, which is time consuming. Effort and time can be saved if a user can obtain a copy of a trained classifier from another user, that will serve as a starting point, and then be re-trained with additional labelled examples. Another possible solution is to use co-training as discussed in Section 3.

We tested the portability across corpora using LingSpam and U5Spam, see Table 11. Note that PU1 cannot be used for cross training as it is encrypted. We trained a filter on LingSpam and then test it on U5Spam (and

Table 9
Performance of stacking and boosted DTs on spam filtering (%) (IG as feature selector)

	$\lambda = 1$				$\lambda = 9$	$\lambda = 999$
	WA	SP	SR	SF1	WA	WA
LingSpam						
Stacking	97.10	90.8	91.90	91.30	98.00	98.13
Boosted DTs	99.59	99.60	97.90	98.70	99.87	99.92
PU1						
Stacking	92.36	92.00	90.40	91.12	93.58	93.85
Boosted DTs	98.18	98.10	97.77	97.90	98.48	98.54

Table 10
Time to build the classifier (s)

	RF	Stacking	Boosted DTs
LingSpam	15.77	366.77	88.20
PU1	6.92	56.45	26.52

Table 11
Portability across corpora using RF (TFV as feature selector, similar results for IG)

	Training set	Testing set	A	SR	SP	SF1
(a)	LingSpam	U5Spam	37.37	75.60	9.40	16.80
(b)	U5Spam	LingSpam	13.45	79.60	13.70	23.40

Table 12
Confusion matrices

# Assigned as	(a)		(b)	
	Spam	Not spam	Spam	Not spam
Spam	62	20	383	98
Not spam	595	305	2406	6

vice versa), using features selected from the training corpus. Typical confusion matrices are given in Table 12. The good news is that in both cases spam e-mails are relatively well recognised (SR = 75.60 and 79.60) which can be explained with the common characteristics of spam e-mails across the two corpora. However, a large proportion of legitimate e-mails were misclassified as spam which is reflected in the low SP. This can be explained with the different nature of the legitimate e-mail of LingSpam (linguistics related) and U5Spam (more diverse). The features selected based on LingSpam are too specific and do not act as a good predictor for the non-spam e-mails of U5Spam. And vice versa, the features selected based on U5Spam are too general to classify correctly the domain specific legitimate e-mails in LingSpam.

Hence, based on our experiments we cannot conclude that the anti-spam filter is portable. Spam e-mails are classified relatively well due to the common characteristics of spam e-mails that are captured by the filter. More extensive experiments with diverse, non-topic specific corpora, are needed to determine the portability of anti-spam filters across different users, and especially the ability to correctly classify legitimate e-mails.

3. Semi-supervised learning for spam e-mail filtering

Supervised machine learning algorithms learn from examples that are labelled with the correct category. To build an accurate classifier, a large number of labelled examples is needed. Obtaining labelled data, however, requires human effort and is a time consuming and tedious process. For example, to build an accurate classifier for spam e-mail filtering, hundreds of labelled training examples (spam and non-spam) are required. A typical user needs to label the incoming e-mail for several days or even weeks before an effective classifier can be built and used to automatically filter e-mail.

To overcome this problem, Blum and Mitchell [5] introduced a new paradigm, called *co-training*, that takes advantage of the more abundant unlaballed data. Co-training learns from a small set of labelled data and a large set of unlabelled data. Blum and Mitchell presented a PAC-analysis of co-training and stated two main dataset requirements for successful co-training. Firstly, the dataset must be described by two disjoint sets of features (called views) that are sufficiently strong. That is, using each view separately, a classifier can be built with reasonably high accuracy. Secondly, the two views should be conditionally independent given the class. They proved that if these conditions are satisfied, a task that is learnable with random noise is learnable with co-training.

In this section, we investigate the applicability of co-training for spam e-mail filtering. We consider the words from the body and the words from the subject as two natural feature sets. We are not only interested in how useful co-training is in the domain of spam e-mail filtering, but also when co-training with a random split of features is likely to be beneficial. The second question is important as in the great majority of practical situations, two natural sets of features do not exist, or the data collected may only belong to one of the possible natural feature sets. To answer this question, we compare co-training with a single natural feature set and co-training with two natural feature sets. As base classifiers we employ RF, SVM, NB and DT and study their performance in a co-training environment. Previous research on co-training uses NB as a base classifier. There is only one study [22] comparing SVM with NB, and showing that the performance of co-training depends on the learning algorithm used. We further investigate this finding.

3.1. Problem statement

E-mail classification can be stated as a semi-supervised problem as follows. Given a training data $D_{\text{train}} = \{D_{\text{lab}} \cup D_{\text{unlab}}\}$ that consists of a small set of labelled e-mail documents $D_{\text{lab}} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$

and a large set of unlabelled e-mail documents $D_{\text{unlab}} = \{d_i, \ldots, d_m\}$, where d_i is an e-mail document from a document set D and c_i is the label chosen from a predefined set of categories C, and $D_{\text{lab}} \cap D_{\text{unlab}} = \emptyset$, the goal is to induce a hypothesis (classifier) $h : D \rightarrow C$ that can correctly classify new, unseen e-mail documents $D_{\text{test}}, D_{\text{test}} \not\subset D_{\text{train}}$.

3.2. The co-training algorithm

In a given application, it may be possible to split up all the features into two sets (views) so that we can build two independent classifiers that can still label the instances correctly. These views are said to be *redundantly sufficient*. As an example, suppose that e-mails can be classified accurately with just the header information (sender, subject, etc.) or just the content in the e-mail body.

The two classifiers are trained with a small set of labelled instances to induce two weak classifiers. They are then employed in a loop to classify the unlabelled examples. Each classifier selects the most confidently predicted examples and adds them to the training set. Both classifiers then re-learn on the enlarged training set. The loop is then repeated for a predefined number of iterations. The co-training algorithm is summarized in Table 13.

The intuition behind this algorithm is as follows. An example may be confidently and correctly predicted by the first classifier using the first set of features, and misclassified by the second classifier using the second set of features. As the first classifier makes a confident prediction, the example will be transferred to the labelled set together with the predicted label. This will allow the second classifier to learn from this example and adjust better in future.

For example, suppose we have two e-mail classifiers using the subject headers and words in the body, respectively. The first one has been trained to categorize any e-mail with the word “assignment” to be placed in the folder “teaching”. If another e-mail comes along with “assignment” in its subject, the first classifier will be very confident that this message should be put in the folder “teaching”, even though the second classifier may be unsure based on the information in the e-mail’s body. By transferring the example in the labelled set, the second classifier will learn that the words in the body indicate class “teaching”.

3.3. Previous work

Blum and Mitchell [5] performed the first experiments on co-training. The task was to identify the home web pages of academic courses from a large collection of web pages collected from several Computer Science

Table 13
Co-training algorithm

```
Given:
   – a small set $\mathbf{L}$ of labelled examples
   – a large set $\mathbf{U}$ of unlabelled examples
   – two feature sets (views) $\mathbf{V}_1$ and $\mathbf{V}_2$ describing the examples

Training:
Create a pool $\mathbf{U}'$ by randomly choosing $\mathbf{u}$ examples from $\mathbf{U}$
Loop for $\boldsymbol{k}$ iterations:
   Learn classifier $\mathbf{C}_1$ from $\mathbf{L}$ based on $\mathbf{V}_1$
   Learn classifier $\mathbf{C}_2$ from $\mathbf{L}$ based on $\mathbf{V}_2$
   $\mathbf{C}_1$ labels examples from $\mathbf{U}'$ based on $\mathbf{V}_1$and chooses the most
      confidently predicted $\mathbf{p}$ positive and $\mathbf{n}$ negative examples $\mathbf{E}_1$
   $\mathbf{C}_2$ labels examples from $\mathbf{U}'$ based on $\mathbf{V}_2$and chooses the most
      confidently predicted $\mathbf{p}$ positive and $\mathbf{n}$ negative examples $\mathbf{E}_2$
   $\mathbf{E}_1$ and $\mathbf{E}_2$ are removed from $\mathbf{U}'$ and added with their labels to
$\mathbf{L}$
  Randomly choose $\mathbf{2p + 2n}$ examples from $\mathbf{U}$ to replenish $\mathbf{U}'$
End

Classification of new examples:
Multiply the probabilities output by C1 and C2
```

departments. They used the following natural feature split: the words present in the web page (page-based classifier) and the words used in another page's link that pointed to the page (hyperlink-based classifier). The results showed that the error rate of the combined classifier was reduced from 11% to 5%.

Kiritchenko and Matwin [22] applied co-training to the domain of e-mail classification into folders. They found that the performance of co-training is sensitive to the learning algorithm used. In particular, co-training with NB worsens performance, while SVM improves it. The authors explained this with the inability of NB to deal with large sparse datasets. This explanation was confirmed by better results after feature selection.

Nigam and Ghani [30] compared the performance of co-training with the Expectation Maximization (EM) algorithm. In their first experiment, co-training was applied to the web pages database from [5]. The results showed that co-training using NB was not better than EM even when there is a natural split of features. Both EM and co-training with NB improved the performance of the initial classifier by 10%. The second experiment investigated the sensitivity of co-training to the independence assumption. A semi-artificial dataset was created so that the two feature sets are truly conditionally independent. In addition, the condition of redundantly sufficient features was met, since the NB trained on each of the data set separately was able to obtain a small error. It was found that co-training with NB well outperformed EM, and even outperformed NB trained with all instances labelled. Their third experiment involved performing co-training on a dataset where a natural split of feature sets is not used. The two feature sets were chosen by randomly assigning all the features of the dataset into two different groups. This was tried for two datasets: one with a clear redundancy of features, and one with an unknown level of redundancy and non-evident natural split in features. The results indicated that the presence of redundancy in the feature sets gave the co-training algorithm a bigger advantage over EM.

The results of these experiments led to the conclusion that co-training is indeed dependant on the assumptions of conditional independence and redundant sufficiency. However, even when either or both of the assumptions are violated, the performance of co-training can still be quite useful in improving a classifier's performance. In particular, in many practical settings, co-training is likely to be beneficial.

We extend previous research by addressing the following questions: (1) how useful is co-training with natural feature sets for spam e-mail filtering, (2) how useful is co-training with a single natural feature set for spam e-mail filtering, and when, in general, it is likely to be beneficial, (3) how sensitive is co-training to the learning algorithms used; we compare RF, SVM, DT and NB.

3.4. Experimental setup

We used the LingSpam corpus, with the standard bag-of-words representation and IG for feature selection, as described in Section 2. Each e-mail was broken up into two sections: the words found in the subject header and the words found in the main body of the message. A summary of the feature sets (views) used in the experiments is given in Table 14. The feature selection was applied individually to each view of the data using IG. Upon inspection of the word lists and their IG values, it was decided that the top 200 words was a suitable cut-off. This means a drastic dimensionality reduction – the percentage of the features retained after feature selection is 0.6% for *All*, 0.9% for *Body*, 1.3% for *Half1* and *Half2,* and 15% for *Subject*. As in the previous experiments, each document was represented using the normalised term frequencies of the selected 200 features for the given view of the data.

An examination of the selected 200 features shows that the feature lists for the *Body* and *All* views are very similar, both in terms of feature rank and IG value. In contrast, only 48% of features in *Subject* also appear in

Table 14
Feature sets (views) used

View	Description
Body	All words that appear in the body of an e-mail
Subject	All words that appear in subject of an e-mail
All	All words that appear in the body and subject of an e-mail
Half1	A random selection of half of the *All* feature set
Half2	The other half of the features not found in *Half1*

Table 15
Top 20 features selected by IG for the various views

	Subject	*Body*	*Half1*	*Half2*	*All*
1	!	language	language	!	!
2	free	!	remove	free	language
3	you	remove	university	linguistic	remove
4	your	university	your	you	free
5	language	free	@	money	university
6	$	linguistic	our	click	linguistic
7	:	you	today	just	you
8	linguistic	your	sell	get	your
9	this	money	english	%	money
10	business	click	market	advertise	click
11	just	@	product	papers	@
12	workshop	our	business	want	our
13	conference	today	linguistics	buy	today
14	get	sell	company	edu	sell
15	of	english	million	ll	english
16	internet	market	internet	every	get
17	million	product	save	purchase	business
18	com	business	income	com	market
19	capitalfm	just	day	best	just
20	?	get	$	over	product

All; their rank is different and their IG value is much lower as they occur in smaller number of e-mails. *Half1* and *Half2* are mutually exclusive views by design, and each of the feature lists contains a subset of *All* with the same IG values and also approximately 100 other terms. Table 15 shows the top 20 words for each view.

The *Body* and *Subject* views will be referred to as the *natural views*, while *Half1* and *Half2* will be referred to as the *random views*.

3.5. Supervised experiment – results and discussion

The objective of this experiment is to determine how redundantly sufficient the different views are. Good classification performance on a given view of the data indicates that it is redundantly sufficient.

Table 16 contains the accuracy results obtained using 10-fold cross validation. Except NB with *Subject,* all other feature sets, with all classifiers, obtained very high accuracy, with RF being the best classifier. The best performing feature sets were *All*, *Body* and *Half1*, closely followed by *Half2*. Thus, the performance of *Body* and the random halves alone is almost the same as using all features.

The accuracy of the *Subject* feature set, with all classifiers except NB, is only 1–8% lower than the accuracy of the other feature sets. Even the 75% accuracy rate of *Subject* with NB is higher than expected. We anticipated worse performance as the number of tokens that appear in the subject is significantly lower than the number of tokens present in the body. Consequently, the IG scores for *Subject* were much lower than for the other feature sets, and it was not clear how discriminative the selected words were. On the other hand, the words from the subject tend to summarise the main topics and are more meaningful than the words in the body. However, this is typically the case for legitimate e-mails, but not necessarily true for spam e-mails

Table 16
Accuracy (%) using various feature sets in the supervised experiment

Classifier	Subject	Body	Half1	Half2	All
RF	96.9	99.6	99.1	99.2	99.9
DT	90.0	97.7	96.9	97.3	97.4
SVM	92.6	95.9	95.8	93.7	99.9
NB	74.9	95.4	95.6	99.8	95.8
Average	88.7±9.6	97.2±1.9	96.9±1.6	97.5±2.7	98.3±2.0

where sometimes the words in the subject are different than the words in the body in order to deceive the user or spam filter.

We noticed that the performance of NB on the *Subject* data can be improved by the use of entropy-based discretization [15], which is more similar to the in-built discretization procedure used in tree classifiers such as RF and DT. These three classifiers cannot directly handle numeric attributes and require discretization as a pre-processing step. The in-built discretization procedure for NB in Weka uses probability density functions and assumes a normal distribution. This assumption is clearly violated on the sparse *Subject* data.

The main conclusion we can draw is that each of the natural and random views are strong enough to learn the classification tasks with the given classifiers. This, however, does not guarantee that co-training will work well. The number of initially labelled examples should allow the construction of a classifier that is accurate enough to label the most confident examples correctly and sustain co-training improvement. Also, it is unrealistic to expect that the view independence assumption holds in the domain of spam e-mail filtering, and this may also affect the performance.

3.6. Co-training experiment – results and discussion

In this experiment we investigate the performance of the co-training algorithm with natural and random views for spam e-mail filtering.

For evaluation of the co-training results we used a procedure that resembles 10-fold-cross validation as suggested in [16]. The standard 10-fold-cross validation procedure uses 90% of the data for training and 10% for testing. The co-training algorithm, on the other hand, uses only a small number of labelled and unlabelled training examples. If 10-fold cross validation is applied in a co-training setting, many examples will not be used neither for training nor for testing. A better utilization of the available data is to increase the size of the test set which will improve the evaluation of the classifier without significantly reducing its quality.

We generated 10 stratified folds. Each time 40% of the data was used for testing and the remaining 60% for training. The required number of initially labelled examples were randomly selected from the first fold of the training data, and the remaining examples from this fold and the other 5 folds were used as unlabelled examples. The experiments were repeated 10 times and the results averaged, each time using different fold to select the labelled examples, and creating different unlabelled and test sets by sliding 1 fold. Thus, each fold is used once to create the labelled set, five times for the unlabelled set and four times for the test set.

We started off with a labelled set of 5 spam and 5 legitimate e-mails. The ratio of spam to legitimate e-mails in LingSpam is 1:5. Following this distribution, 2 newly labelled spam and 10 legitimate e-mails were transferred from the unlabelled set to the labelled set on each co-training iteration. We also closely followed the original algorithm by Blum and Mitchell [5] using a small unlabelled pool $\mathbf{U}'$ of 50 randomly selected examples from $\mathbf{U}$. At each iteration $\mathbf{C}_1$ and $\mathbf{C}_2$ label the examples from $\mathbf{U}'$ and the most confidently predicted 1 spam and 5 legitimate e-mails by each classifier are transferred to the labelled set $\mathbf{L}$. The experiments were run for 20 co-training iterations. The prediction of the co-training classifier on new example is calculated by multiplying the class probabilities output by the two classifiers $\mathbf{C}_1$ and $\mathbf{C}_2$.

Table 17 summarises the results. The column *it0* shows the accuracy of the classifier trained on the initial 10 labelled examples before the co-training (iteration 0). The column *it20* shows the accuracy at the end of co-training, i.e. after iteration 20, where the number of training examples is 250 (10 initially labelled +20 ×

Table 17
Accuracy (%) in the co-training experiment

Classifier	Natural split				Random split			
	It0	It20	Increase (it20–t0)	Gap (goal-it20)	It0	It20	Increase (it20–it0)	Gap (goal-it20)
RF	84.8	94.9	10.1	5.0	89.1	94.7	5.6	5.2
DT	78.0	87.8	9.8	9.6	70.5	83.0	12.5	14.4
SVM	61.0	87.1	26.1	12.8	69.5	85.9	16.4	14
NB	89.1	90.8	1.7	5.8	80.6	88.9	8.3	6.7

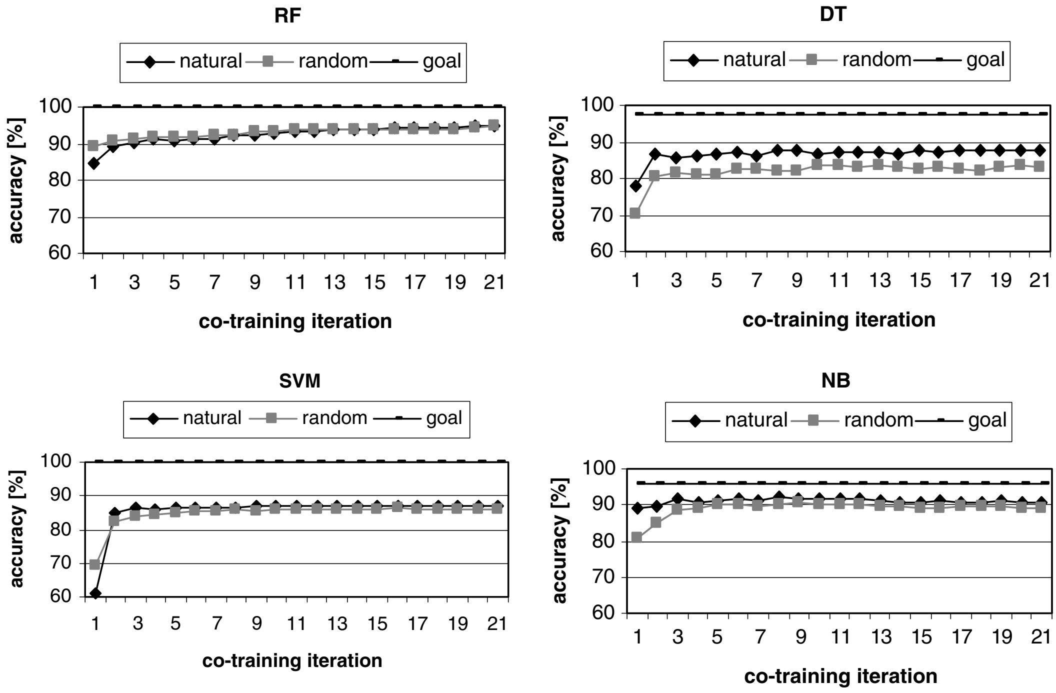

Fig. 2. Co-training learning curves for RF, DT, SVM and NB.

12 self-labelled). The column *increase* presents the difference between iteration 20 and 0; thus, positive numbers indicate improvement over the base classifier trained on the initial labelled examples. The column *gap* indicates the difference between the goal performance and *it20*'s performance. As a goal performance we consider the accuracy of a supervised classifier trained on the labelled version of all training data, using 10-fold cross-validation (i.e. trained on 2604 examples). Thus, the goal accuracies are listed in the last column of Table 16. Fig. 2 show the co-training learning curves for all classifiers, and also the goal performance.

By examining the improvement over the initial classifier (column *increase*), the accuracy value at the end of co-training (column *it20*) and the difference between the goal accuracy and the accuracy at the end of co-training (column *gap*), the first conclusion we can draw is that co-training was successful for all classifiers, both for natural and random views, as all classifiers reduced the error of the initial classifier. The best classifiers was RF achieving the highest absolute accuracy value and the smallest gap, for both natural and random views. NB came second with 4–6% lower accuracy value and 0.8–1.5% bigger gap despite the fact that its *Subject* view was not as strong. SVM started co-training with the weakest classifier (*it0*) and achieved the greatest increase but did not reach the accuracy rate of RF falling behind with 8–9%.

The second observation is that co-training with random views produces results that are comparable with using the natural views. There are two reasons for this. Firstly, the *All* view was found to be redundantly sufficient, as discussed in the previous section. That is, using a random selection of half of the features from all the features resulted in classifiers that perform almost the same as a classifier using all the features available. Secondly, the amount of the initially labelled examples seems to be enough to build an initial weak classifier that is accurate enough to sustain co-training improvement.

As view independence is expected to influence performance, we also measured the conditional independence, given the class, of the two natural and two random views. As an approximation of the statistical independence between two sets, we calculated the sum of the pairwise conditional mutual information between each pair of features, where one of the features belongs to the first view and the second one – to the second view [16]. The results showed that the random views are 9 times more conditionally dependant than the natural views. This does not seem to significantly influence the performance; co-training with a natural split is slightly better than co-training with random split.

It is interesting to note that in [22] SVM was found to be superior to NB and this was explained with the NB's sensitivity to the large number of features because of the violation of the independence assumption. In our experiments we performed drastic feature selection as discussed in Section 3.3. Hence, the SVM, which performs well in high-dimensional feature spaces, did not get to illustrate its advantage over NB in such a setting.

4. Conclusion

In this paper we consider supervised and semi-supervised e-mail classification tasks and investigate the performance of four algorithms: RF, DT, SVM and NB. Our findings can be summarized as follows:

- In both supervised and semi-supervised co-training setting, we have shown that RF is a promising approach for automatic e-mail filing into folders and spam mail filtering. It outperforms in terms of classification performance well-established algorithms such as DT, SVM and NB, with DT and SVM being also more complex than RF. RF is easy to tune, and runs very efficiently on large datasets with high number of features, which makes it very attractive for text categorization.
- We introduced a new feature selector TFV and found that it performs better than the popular and computationally more expensive IG.
- E-mail filing into folders is a complex task, with several different characteristics than the traditional text categorization. The success of an automatic system highly depends on the user classification style: it works well for users categorizing e-mail based on topics and sender, and does not work for users categorizing e-mails based on other criteria, e.g. action performed. E-mail filing into folders is an imbalanced problem, the topics of the bigger folders typically change over time, and some of the abandoned folders contain only a few examples.
- Our preliminary experiments did not show an anti-spam filter to be portable across users. It is able to capture spam characteristics and classifies spam e-mails relatively well, but misclassifies a large number of legitimate e-mails as spam. More extensive experiments with diverse, non-topic specific corpora, are needed.
- We compared the performance of a number of algorithms on the benchmark spam filtering corpora LingSpam and PU1. Some of them, e.g. SVM, NB, boosted trees and stacking have been previously applied but on different versions of the data and using different pre-processing which did not allow fair comparison.
- We have shown empirically that co-training can be successfully applied to boost the performance of a spam e-mail classifier that is given only a very small set of labelled examples. The first view (*Body)* contained the words used in the body, while the second one (*Subject)* contained the words occurring in the subject headers. Although not conditionally independent, these two natural views were shown to be sufficiently redundant, i.e. strong enough to learn the classification task separately. The small number of initially labelled examples allowed to build a classifier that sustains co-training improvement.
- We investigated the performance of co-training with only one natural view for spam filering. This was motivated by the co-training's requirement that the dataset is described by two separate views, which is a limitation as the majority of datasets consist of only a single set of features with no obvious way to divide them. It was shown empirically that co-training using a random split of all the features was as competitive as co-training with the natural feature sets *Body* and *Subject*.
- We conclude that co-training with a random feature split works well if there is high data redundancy as in the domain of spam e-mail filtering. When this condition is met, a random split of the feature set will produce two views, each of which can still be used on its own by a classifier to achieve a sufficiently high classification performance. The conditional independence of the random views was found to be higher that the conditional independence of the natural views but this does not seem to influence the performance significantly.
- Although representative, the corpora we used have some limitations. All of them, except the encrypted PU1 corpus, are created by first excluding sensitive personal e-mails which may bias performance. This probably cannot be avoided as privacy is an important consideration when creating publicly available e-mail corpus. LingSpam and PU1 do not contain information from the *Sende*r field, html tags and attachments which

may lead to over-pessimistic performance. On the other hand, LingSpam is a mixture of mailing list messages with personal e-mails which may lead to over-optimistic results.

Future work will include an adaptation of RF to deal with the problem of concept drift and imbalanced classification in e-mail classification. As shown, these are important problems for both filing e-mails into folders and spam e-mail filtering. Ensemble strategies for changing environments are discussed in [26]. For example, ensemble members trained on the new chunks of data can be added and old or not very useful ensemble members deleted. RF can also be extended to learn imbalanced data [9].

A possible avenue for future work in co-training is to develop an algorithm capable of obtaining the optimal (or a near-optimal) split of the features rather than using a random split. A deeper investigation of the impact of view independence would be very beneficial.

Acknowledgement

We thank Felix Feger for the implementation of the co-training algorithm.

References

[1] LingSpam and PU1 datasets, <http//www.aueb.gr/users/ion/publications.html>.
[2] Weka, http//www.cs.waikato.ac.nz/ml/weka.
[3] I. Androutsopoulos, J. Koutsias, V. Chandrinos, C. Spyropoulos, An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages, in: Proc. ACM SIGIR 2000, pp. 160–167.
[4] I. Androutsopoulos, G. Palioras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, P. Stamatopoulos, Learning to filter spam e-mail: a comparison of a Naïve Bayesian and memory-based approach, in: Proc. 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) 2000, pp. 1–13.
[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proc. Workshop on Computational Learning Theory, 1998.
[6] Breiman, Random forests, Machine Learning 45 (2001) 5–32.
[7] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 49–64.
[8] X. Carreras, L. Marquez, Boosting trees for anti-spam email filtering, in: Proc. 4th International Conference on Recent Advances in Natural Language Processing, 2001.
[9] C. Chen, A. Liaw, L. Breiman, Using Random Forest to Learn Imbalanced Data, University of California, Berkeley, 2004.
[10] J. Clark, I. Koprinska, J. Poon, A neural network based approach to automated e-mail classification, in: Proc. IEEE/WIC International Conference on Web Intelligence (WI), 2003, pp. 702–705.
[11] W. Cohen, Learning rules that classify e-Mail, in: Proc. AAAI Symposium on Machine Learning in Information Access, 1996, pp. 18–25.
[12] E. Crawford, I. Koprinska, J. Patrick, A multi-learner approach to e-mail classification, in: Proc. 7th Australasian Document Computing Symposium (ADCS), 2002.
[13] N. Ducheneaut, L. Watts, In search of coherence: a review of e-mail research, Human-Computer Interaction 20 (2004) 11–48.
[14] S. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive learning algorithms and representations for text categorization, Microsoft Research, 1998.
[15] U. Fayyad, K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proc. 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, France, 1993, pp. 1022–1027.
[16] F. Feger, Learning from labelled and unlabelled data, MIT thesis, University of Sydney, 2005.
[17] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proc. 13th International Conference on Machine Learning, Morgan Kaufmann, 1996, pp. 146–156.
[18] P. Graham, Better Bayesian filtering, 2003, <http://www.paulgraham.com/better.html>.
[19] L. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (1990) 993–1001.
[20] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 832–844.
[21] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proc. 10th European Conference on Machine Learning, 1998.
[22] S. Kiritchenko, S. Matwin, E-mail classification with co-training, in: Proc. CASCON, 2001.
[23] I. Koprinska, F. Trieu, J. Poon, J. Clark, E-mail classification by decision forests, in: Proc. 8th Australasian. Document Computing Symposium (ADCS), 2003.
[24] A. Kotcz, X. Sun, J. Kalita, efficient handling of high-dimensional feature spaces by randomised classifier ensembles, in: Proc. ACM SIGKDD'02, 2002, pp. 307–313.

[25] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: D.T.G. Tesauro, T. Leen (Eds.), Advances in Neural Inf. Processing Systems, Morgan Kaufman, 1995, pp. 650–659.

[26] L. Kuncheva, Classifier ensembles for changing environments, in: Proc. 5th International Workshop on Multiple Classifier systems, Springer-Verlag, 2004, pp. 1–15.

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

[28] MessageLabs, 2005, <http://www.messagelabs.com/Threat_Watch>.

[29] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[30] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proc. 9th International Conference on Information and Knowledge Management, 2000.

[31] P. Pantel, D. Lin, SpamCop: a spam classification and organization program, in: Proc. AAAI Workshop on Learning for Text Categorization 1998.

[32] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schoelkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, MIT Press, 1998.

[33] J. Provost, Naïve-Bayes vs. rule learning in classification of email, University of Texas at Austin, 1999.

[34] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.

[35] J. Rennie, An application of machine learning to e-mail filtering, in: Proc. KDD-2000 Text Mining Workshop, 2000.

[36] G. Rios, H. Zha, Exploring support vector machines and random forests for spam detection, in: Proc. First International Conference on Email and Anti Spam (CEAS), 2004.

[37] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A Bayesian approach to filtering junk e-mail, in: Proc. AAAI Workshop on Learning for Text Categorization, 1998.

[38] G. Sakkis, I. Androutsopoulos, G. Palioras, V. Karkaletsis, C. Spyropoulos, P. Stamatopoulos, Stacking classifiers for anti-spam filtering of e-mail, in: Proc. 6th Conference on Empirical Methods in Natural Language Processing, 2001, pp. 44–50.

[39] F. Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys 34 (2002) 1–47.

[40] R. Segal, M. Kephart, MailCat: An intelligent assistant for organizing e-mail, in: Proc. Third International Conference on Autonomous Agents, 1999.

[41] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison Wesley, 2005.

[42] V. Vapnik, Statistical Learning Theory, Wiley, 1998.

[43] S. Whittaker, V. Bellotti, P. Moody, Introduction to this special issue on revisiting and reinventing e-mail, Human-Computer Interaction 20 (2005) 1–9.

[44] Y. Yang, J. Pedersen, A comparative study on feature selection in text categorization, in: Proc. 4th International Conference on Machine Learning, 1997.