Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora

Ron Bekkerman RONB@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts, Amherst, USA

Andrew McCallum MCCALLUM@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts, Amherst, USA

Gary Huang GHUANG@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts, Amherst, USA

UMass CIIR Technical Report IR-418

Abstract

Office workers everywhere are drowning in email—not only spam, but also large quantities of legitimate email to be read and organized for browsing. Although there have been extensive investigations of automatic document categorization, email gives rise to a number of unique challenges, and there has been relatively little study of classifying email into folders.

This paper presents an extensive benchmark study of email foldering using two large corpora of real-world email messages and foldering schemes: one from former Enron employees, another from participants in an SRI research project. We discuss the challenges that arise from differences between email foldering and traditional document classification. We show experimental results from an array of automated classification methods and evaluation methodologies, including a new evaluation method of foldering results based on the email timeline, and including enhancements to the exponential gradient method Winnow, providing top-tier accuracy with a fraction the training time of alternative methods. We also establish that classification accuracy in many cases is relatively low, confirming the challenges of email data, and pointing toward email foldering as an important area for further research.

1. Introduction

In the past decade text categorization has been a highly popular machine learning application. In addition to the standard problem of categorizing documents into semantic topics (Lewis, 1992), a variety of other problem domains have been explored, including categorization by genre (Finn et al., 2002), by authorship (Diederich et al., 2003) and even by author's gender (Koppel et al., 2003).

In the domain of personal email messages, text categorization methods have been widely applied to the problem of spam filtering (see, e.g., Drucker et al., 1999). Other email related problems have also been tackled, such as extracting email threads (Lewis and Knowles, 1997) and automatically creating new folders (Giacoletto and Aberer, 2003). However, there has

been little study of categorizing email into folders—also termed "email foldering". There are only a few notable exceptions.

Payne and Edwards (1997) use a rule induction algorithm CN2 (Clark and Niblett, 1989) and a k nearest neighbor (k-NN) classifier for foldering their own email and argued that CN2 outperforms k-NN on this task. Cohen (1996) considers a number of binary classification problems of one folder vs. all the others; he compares his RIPPER classifier (which also belongs to the rule induction family) with a *tfidf* classifier and demonstrates RIPPER's superior performance. However, Provost (1999) shows that the Naive Bayes classifier can outperform RIPPER on the email classification task. Rennie (2000) uses the Naive Bayes for constructing a real-world email foldering system that suggests three most appropriate folders for each incoming message. With very high precision the desired folder can be found among the three suggested ones, which dramatically simplifies the process of manual foldering. Kiritchenko and Matwin (2001) first use the popular Support Vector Machine (SVM) for the email classification task and show its advantage over the Naive Bayes classifier. We extend the previous research work by comparing email classification results of four classifiers (Maximum Entropy, Naive Bayes, SVM and Winnow), using original evaluation methodology.

Email foldering is a rich and multi-faceted problem, with many difficulties that make it different from traditional topic-based categorization. Email users create new folders, and let other folders fall out of use. Email folders do not necessarily correspond to simple semantic topics—sometimes they correspond to unfinished todo tasks, project groups, certain recipients, or loose agglomerations of topics. It is also interesting to note that email content and foldering habits differ drastically from one email user to another—so while automated methods may perform well for one user, they may fail horribly for another.

Furthermore, email arrives in a stream over time, and this causes other significant difficulties. Some email messages only make sense in the context of previous messages. Occasionally all messages in a thread should go to the same folder, but other times the topic in a thread drifts. The topic associated with a certain email folder can also shift over time. For example, a folder about funding opportunities may at first contain only messages about the National Science Foundation, but later only get new messages about industrial partnerships, each of which may have very different word distributions. Ironically, in all but one related work the temporal aspect of email is missed, only Segal and Kephart (2000) apply the *tfidf* classifier in a time-based incremental setup for foldering email.

A likely reason that the problem of automatic email foldering has not drawn significant attention in the research community is the fact that there has been no standard, publicly-available real-world email dataset on which foldering methods could be evaluated, and on which the work of multiple researchers could be compared.

However, a large corpus of real-world email messages subpoenaed from Enron Corporation was placed in the public record, and recently made available to researchers electronically.[1] The data consists of over 500,000 email messages from the email accounts of 150 people. Furthermore, a smaller but also significant corpus of real-world, foldered email has been created as part of the CALO DARPA/SRI research project.[2] This corpus contains snapshots of the email folders of 196 users, containing approximately 22,000 messages.

1. http://www-2.cs.cmu.edu/∼enron/ and http://iesl.cs.umass.edu/data/enron
2. Cognitive Agent that Learns and Organizes (CALO): http://www.ai.sri.com/project/CALO

In a recent work, Klimt and Yang (2004) present some basic statistics on the Enron dataset and provide useful insights on email classification and thread recognition on this data. They report a classification result that is averaged over all the Enron users and achieved on an unnatural 50/50 training/test split. Klimt and Yang's work cannot be used as a baseline for comparing various email classification methods.

In this paper we present a benchmark case study of email foldering on both the Enron and SRI email datasets. We concentrate on foldering email of particular users and establish a framework for further email foldering experiments. Our resulting graphs and preprocessed portions of the Enron dataset are available online at http://www.cs.umass.edu/∼ronb/. We provide a wide performance comparison across several popular classifiers: Maximum Entropy (MaxEnt), Naive Bayes (NB), Support Vector Machine (SVM), as well as an enhanced variant of Winnow. To our knowledge, this paper is the first work in which MaxEnt and Winnow are applied to email classification. We show that the Winnow classifier is not only computationally attractive and easy to implement, but achieves surprisingly good results on the email foldering task, in many cases achieving statistically similar performance as the well-established SVM classifier. We provide detailed description of the enhancements to Winnow used here. We also propose a novel evaluation method for classification performance particularly appropriate to the email domain. The method involves dividing a dataset into time-based data chunks and incrementally testing classification performance on each chunk while training on all the previous chunks.

The rest of the paper is organized as follows: in Section 2 we state the email foldering problem; in Section 3 we discuss various design choices for our experimental setup, and how they differ from the standard text categorization setting; in Section 4 we briefly overview the classifiers we apply; in Section 5 we describe the datasets used for evaluation; in Section 6 we report and discuss our results; finally, in Section 7 we conclude and outline some open problems.

2. Problem statement

Let us define formally the email foldering problem, as a special case of the general text categorization problem. We are given a training set $\mathcal{D}_{train} = \{(d_1, \ell_1), \ldots, (d_n, \ell_n)\}$ of labeled text documents, where each document d_i belongs to a document set $\mathcal{D}$, and the label ℓ_i of d_i is selected from a predefined set of categories $\mathcal{C} = \{c_1, \ldots, c_m\}$. The goal of text categorization is to induce a learning algorithm that, given the training set $\mathcal{D}_{train}$, generates a classifier $h : \mathcal{D} \rightarrow \mathcal{C}$ that can accurately classify unseen documents.

The design of learning algorithms for text categorization involves solutions to three basic subtasks:

- **Document Representation**: In most cases documents are represented as distributions over *features*, where features can be words (Dumais et al., 1998), sequences of words (Caropreso et al., 2001), part-of-speech tags (Finn et al., 2002), word clusters (Bekkerman et al., 2003), etc. Document representations usually undergo a transformation of dimensionality reduction (see, e.g., Yang and Pedersen, 1997).

- **Classifier Induction**: Various off-the-shelf classifiers have been applied to the text categorization. We distinguish between classifiers that employ generative and dis-

criminative approach to classification. Many times it has been empirically shown that discriminative classifiers outperform generative classifiers on the topic-based text categorization task (see, e.g., Dumais et al., 1998).

- **Model Selection**: Once a document is represented in the form convenient for classification and the appropriate classifier is chosen, various parameters of the system, such as the type of kernel to use for an SVM, need be specified. The categorization results strongly depend on the parameters tuned, and can vary from very low performance if the parameter values do not fit the task to almost perfect performance in certain cases.

We suggest one particular choice of the document representation, then focus on the induction of various classifiers, and briefly discuss essential model selection issues. In addition to applying four classifiers and providing benchmark classification results, we propose a new evaluation method for email foldering.

3. Design choices

There is a large number of design choices in how to set up the email foldering task. In this section we explain our decisions on the major issues. Raw email datasets are often messy and unstructured, and the Enron and SRI datasets are no exception. A large amount of cleaning, preprocessing and organization steps should be taken before training classifiers. The issue of performance evaluation also need to be resolved, because (as discussed in Section 1) standard evaluation methods are not appropriate for the email foldering task.

3.1 Removal of non-topical folders

A folder is considered to be *non-topical* if email messages are stored in this folder regardless of their content. This category includes folders such as Inbox, Sent, Trash, and Drafts. We believe it makes more sense to remove the non-topical folders, because no automatic system is going to assist the user in classifying messages into these folders. Non-topical folders belong to three main categories:

- Automatically created folders of an email application (such as "Sent Items" of MS Outlook or "sent-mail" of Pine).

- Archiving folders that are standard for all the users of a certain organization (such as "all_documents" and "discussion_threads" that can be found in the folder hierarchies of all former Enron employees).

- Archiving folders that are created by a particular user. This can be the case, for example, if the user is currently unable to cope with the email stream and decides to put aside a certain portion of the incoming email for further processing.

We remove all the non-topical folders of the first two types. We do not remove folders of the third type because it requires familiarity with the foldering strategy that differs from one user to another. For more detail, see Section 5.

3.2 Removal of small folders

Folders with a small number of messages are fairly common, and it is expected that a user would especially want automatic categorization to work with these folders. However, the folders must contain enough messages to provide training examples for the classifiers. In our setting, we ignore folders that contain only one or two messages.

3.3 Training/test set splits

In many classification settings, the standard training/test split is done uniformly at random. However, in practice email datasets are constantly growing over time, so random splits may create unnatural dependencies of earlier email on later email. Thus, a more reasonable split would be the one that is based on time: train on earlier email, and test on later email. This approach is employed in (Klimt and Yang, 2004): the classifier is trained on the early half of the email directory and then tested on the later half. A serious problem of this approach can arise when the test set is large: topics discussed in email far in the future may have nothing in common with the email the classifier was trained on. Another problem occurs when some of the folders in the test set did not exist at training time. Still another problem is in designing the training/test splits for assessing statistical significance of the foldering results.

In this paper, we propose an incremental time-based split: after sorting the messages according to their time stamp, we train a classifier on the first N messages and test it on the following N messages, then train on the first $2N$ messages and test on the following N messages etc., until we finally train on the first $(K-1)N$ messages and test on the rest of the messages (where K is the number of dataset splits). This approach provides a practical and intuitively clear split of the data that allows us to monitor the classifiers' performance as the number of messages increases over time. We ignore test messages in folders that do not exist in the training data. We use $N = 100$ for Enron datasets and $N = 50$ for SRI datasets.

To address the problem of achieving statistically significant classification results using a time-based evaluation method, we propose to average the results over all the training/test set splits. Intuitively, such averaging might appear inadequate, because we expect the system to improve its performance as the training set grows, but in practice this improvement is unlikely to occur (see Figures 1 and 3), which justifies the applicability of our averaging approach.

Our evaluation method differs from the one proposed by Segal and Kephart (2000), who apply an incremental classification procedure with $N = 1$. Such an approach, while legitimate, is not computationally feasible for large datasets. This approach is also impractical in real world applications: since a classifier induction is a resource consuming task, the classifier would unlikely be retrained after every single email message is foldered.

Another design possibility could be to employ time-periodic data chunks, e.g., train on email of the first day and test on email of the second day, then train on email of the first two days and test on email of the third day etc. This approach suffers from the problem of (unexpected) bursts of activity (see Kleinberg, 2002): if one day a user receives the amount of email he or she normally receives in an entire week, the time-periodic split may cause undesired anomalies.

3.4 Feature construction

There are many design choices in the feature construction. In this paper, we use the traditional bag-of-words document representation: messages are represented as vectors of word counts. We consider words as sequences of alphabetic, digit and underscore characters appearing anywhere in the email header or body. Words are downcased, 100 most frequent words and words that appear only once in the training set are removed, and the remaining words are counted in each message to compose a vector. We remove MIME attachments and do not apply stemming.

In future work, richer representations could be considered, including the following:

- Different sections of each email can be treated differently. For example, the system could create distinct features for words appearing in the header, body, signature, attachments, etc. Klimt and Yang (2004) consider the *From*, *To* and *CC* fields of the message headers and advocate the importance of using features out of the *From* field.

- Named entities may be highly relevant features. It would be desirable to incorporate a named entity extractor (such as MinorThird[3], see, e.g., Cohen and Sarawagi (2004)) into the foldering system.

3.5 Evaluation measure and confidence ranking

We use traditional classification *accuracy*[4] as our evaluation method. However, in a general case, email messages can with certain probability belong to multiple folders, so it would be beneficial not to directly assign a message into one folder but rather to rank folders according to a confidence measure of the message belonging to each of these folders. This approach is used by Rennie (2000) in his *ifile* foldering system. When such ranking is obtained, the system can receive full credit if the correct folder is found among a number of top-ranked choices. The application of a ranking method depends entirely on the use scenario, e.g., will users read all their email from a common Inbox, performing foldering on demand, or would we like the system to automatically route messages to different folders so the user can read them by category? In either scenario, the importance of the issue of ranking should not be neglected. Our experimental results provide a baseline for the foldering scheme that involves ranking: we plot foldering accuracy as a function of the test set coverage (see Section 6.1).

3.6 Other design choices

Other design choices include:

- **Hierarchical vs. flat foldering.** The hierarchy of email folders can potentially be exploited to provide a flexible evaluation criterion. For example, the penalty for classifying a message into an incorrect folder can be less if the correct folder is "nearby in the hierarchy". We believe however that users may have high expectations of finding messages on a given topic in a single folder—without having to hunt in nearby

3. http://minorthird.sourceforge.net
4. Ratio of correctly classified instances to the total number of instances in the test set.

folders. To meet these expectations, we *flattened* the folder hierarchies of the datasets we use. For example, if a particular user has two top-level folders each containing two subfolders, we generate a flat structure of six folders (two of them corresponding to the top-level folders and the remaining four corresponding to their subfolders).

- **Personal vs. corporative email.** Email foldering can potentially be applied to a dataset that consists of email directories of a number of users, or alternatively to an email directory of one user only. In practice, due to security and privacy reasons, a software system is unlikely to have access to email directories of different users. Furthermore, this is unclear how to conjoin the foldering schema of multiple users. In our experiments we deal with email of each individual user separately.

  We notice that the majority of email directories in both datasets are extremely small, which makes it difficult to do performance comparisons that are of statistical significant value. Therefore, from each of the two datasets we select *seven* users with the largest email directories.

4. Classification procedure

We test three standard classifiers on the email foldering task: Maximum Entropy, Naive Bayes, and Support Vector Machine (SVM). We also present wide-margin Winnow—a variant of the Winnow classifier that demonstrates very promising results on this task. We implement Maximum Entropy, Naive Bayes and Winnow classifiers as a part of the MALLET[5] software system, and we use the SVM*light* package[6] by Thorsten Joachims for our Support Vector Machine classifier.

4.1 Maximum Entropy

The Maximum Entropy approach models the class-conditional distribution with the most uniform one that satisfies constraints given by the training set. Pietra et al. (1997) show that the unique distribution that both satisfies the constraints in the training data and has the maximum entropy belongs to the exponential family, having the form:

$$p(c_j|d_i) = \frac{1}{Z_d} \exp\left(\sum_k \lambda_k f_k(d_i, c_j)\right), \tag{1}$$

where Z_d is the normalization factor that depends on the document, f_k is a feature, and λ_k is the weight or relevance of the feature. When training a maximum entropy classifier, the goal is to find the weights λ_i that give good performance on the training data. The likelihood function is convex, so the λ_i parameters can be estimated by a gradient-climbing algorithm. In our experiments, we use a quasi-Newton method called BFGS (Byrd et al., 1996).

The maximum entropy classifier finds the maximum likelihood parameters λ_i, so it suffers from overfitting when the size of the training data is small (such as when a new folder is created) or when data is sparse (which is in general true for the text domain

5. http://mallet.cs.umass.edu
6. http://svmlight.joachims.org

(Nigam et al., 1999)). To alleviate this problem, we specify a prior distribution for the values of each λ_i and perform choose the λ_i via maximum a posteriori estimation. We use the common Gaussian distribution with a diagonal covariance matrix as the prior. We also apply the independence assumption and set the variance to 1.0 for all the λ_i's:

$$p(\Lambda) = \prod_i \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-\lambda_i^2}{2}\right). \quad (2)$$

4.2 Naive Bayes

The Naive Bayes classifier is commonly used to provide a baseline in text categorization. While applying Bayes rule, a document d_i is assigned to category c_{j_0} if

$$j_0 = \arg \max_{j=1,...,m} P(c_j|d_i, \theta) = \arg \max_{j=1,...,m} P(c_j|\theta)P(d_i|c_j, \theta). \quad (3)$$

The likelihoods $P(d_i|c_j, \theta)$ are computed using the (naive) independence assumption $P(d_i|c_j, \theta) = \prod_{k=1}^{|d_i|} P(w_k|c_j, \theta)$, where $w_1, ..., w_{|d_i|}$ are words of the document d_i. The parameters θ are estimated from the training set, usually using a multinomial or a multiple Bernoulli model. For more details, we refer the reader to McCallum and Nigam (1998) and Rennie (2001). In our experiments we use the multinomial model with Laplace smoothing (adding one to the count of each word in each class), a common practice for text categorization.

4.3 Support Vector Machine

The Support Vector Machine (Boser et al., 1992) is a highly popular vector-space classification method broadly used for text categorization (see, e.g., Joachims, 2002). SVMs are usually considered in a framework of binary classification. In multiclass settings, the classification problem is usually decomposed to multiple binary subproblems by observing one class vs. all the others. The goal of the two-class SVM is to induce a maximum-margin hyperplane that separates training instances from the two classes: document d is assigned a label $y = \{-1, 1\}$ if

$$y = \text{sign}\left(\sum_{i=1}^{l} v_i \langle d, d_i \rangle + b\right), \quad (4)$$

where the slope of the separating hyperplane $\langle \mathbf{w}, d_i \rangle + b$ is represented as a linear combination $\mathbf{w} = \sum_{i=1}^{l} v_i d_i$ of support vectors $\{d_i | i = 1, \ldots, l\}$ that lie on the margin. Parameters v_i are learned on the training set using a quadratic programming algorithm. The dot product $\langle d, d_i \rangle$ can be generalized to basically any function of the input feature space, such as arbitrarily high dimensional polynomials that induce feature conjunctions. The strength of SVMs is its ability to compute such distance measures efficiently, via what is called the *kernel trick*.

We use a simple linear kernel (which has proved its effectiveness on the text classification task) and specify SVM parameters c (trade-off between training error and margin) and j (cost-factor, by which training errors on positive examples outweigh errors on negative examples). We fix $c = 0.01$ (chosen based on general past experience, see Bekkerman et al. (2003)), and choose j among the integers 1-5 on a validation set for each training/test split.

4.4 Wide-Margin Winnow

Winnow (Littlestone, 1988) belongs to the family of on-line learning algorithms, which means that it attempts to minimize the number of incorrect guesses during training as training examples are presented to it one at a time. It has several important theoretical properties, including the ability to easily handle domains with sparse data and high dimensionality, such as text classification (Blum, 1996). Winnow is similar to a perceptron (Rosenblatt, 1958) in that it attempts to construct a linear separator (i.e., a weight vector) for each class. In fact, Winnow is guaranteed to find a linear separator if it exists. It differs from the perceptron by doing multiplicative updates instead of additive updates of the weight vectors during training.

The multi-class implementation of the Winnow algorithm we present here (called Wide-Margin Winnow) is derived from Avrim Blum's implementation at WhizBang! Labs. It contains flavors of the linear-max Winnow in Mesterharm (2003) and the the θ-range two-class version of Winnow in Dagan et al. (1997). Our version differs from those algorithms in its update rules, its feature set used[7], and its set of tunable parameters, which we now describe. Let c be the number of classes, m the size of the feature space, and n the number of training examples. We keep $(m+1)$-dimensional weight vectors $w_1, ..., w_c$, each initialized to contain all 1's. Given a new example (x, y), Winnow guesses its label to be $j = \arg\max_{i=1,...,c}\{w_i \cdot x\}$. It performs t passes through the training set, adjusting the weight vectors when it guesses incorrectly (i.e., $j \neq y$). We adjust the weight vectors by increasing w_y and decreasing w_j at those coordinates where x has non-zero feature values. The amount by which we adjust the weights depends on how many passes we have made through the training data: during the p-th pass, we increase weight $w_{i,f}$ by setting $w_{i,f} \leftarrow w_{i,f}(1 + \epsilon\alpha^p)$ or decrease it by setting $w_{i,f} \leftarrow w_{i,f}(1 - \epsilon\alpha^p)$ (one can think of α as the cooling rate).

In addition, as an attempt to disambiguate all training examples, we use the following heuristic to keep wide margins between classes: instead of only adjusting weight vectors after an incorrect guess during training, we also adjust the weights when it "barely" gets the correct answer (i.e., when $j = y$ but the ratio between the largest and second largest dot products is below δ. One final detail is that when we compute dot products, the vector x is augmented with an additional $(m+1)$-st "feature" whose value is always set to 1.0. This is used to make Winnow classify test examples as the largest class seen during training time in the degenerate case when there are *no* relevant features (i.e., when every feature is present or absent 50% of the time regardless of the true class label). In our experiments, we set $t = 5$, $\epsilon = 0.5$, $\alpha = 0.5$, and $\delta = 0.5$. The pseudo-code of our implementation of Winnow is presented as Algorithm 1.

5. Datasets

We present results on two datasets: Enron and SRI. In this section we describe these two datasets and provide some basic statistics on their content.

7. We treat features as binary: if a feature has a non-zero value, we treat it as having the value 1.

Input: $\{(x_k, y_k)|k = 1, ..., n\}$ is training set
ϵ is weight adjustment rate
α is the cooling rate
δ is the confidence measure
Output: $w_1, ..., w_c$ are $(m + 1)$-dimensional weight vectors for each category
Initialize vectors $w_1, ..., w_c$ to all 1's
for $p \leftarrow 1$ to t **do**
  for $k \leftarrow 1$ to n **do**
    $j \leftarrow argmax_{i=1,...,c}\{w_i \cdot x_k\}$
    $j' \leftarrow argsecondmax_{i=1,...,c}\{w_i \cdot x_k\}$
    if $j \neq y$ **then**
      $w_{y,f} \leftarrow w_{y,f}(1 + \epsilon\alpha^p)$ at those coordinates f where x_k is non-zero
      $w_{j,f} \leftarrow w_{j,f}(1 - \epsilon\alpha^p)$ at those coordinates f where x_k is non-zero
    else if $w_y \cdot x_k < \delta w_{j'} \cdot x_k$ **then**
      $w_{y,f} \leftarrow w_{y,f}(1 + \epsilon\alpha^p)$ at those coordinates f where x_k is non-zero
      $w_{j',f} \leftarrow w_{j,f}(1 - \epsilon\alpha^p)$ at those coordinates f where x_k is non-zero
    end if
  end for
end for
Given an unseen example x, guess its label to be $argmax_{i=1,...,c} w_i \cdot x$

Algorithm 1: The Wide-Margin Winnow algorithm

5.1 Enron Email Dataset

The archived email from many of the senior management of Enron Corporation was subpoenaed, and is now in the public record. The dataset is provided by SRI after major clean-up and removal of attachments. The dataset version we use was released on February 3, 2004.

Although the size of the dataset is large, many users' folders are sparsely populated. We use the email directories of seven former Enron employees that are especially large: *beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, sanders-r* and *williams-w3*. Each of these users has several thousand messages, with *beck-s* having more than one hundred folders.

We remove the non-topical folders "all_documents", "calendar", "contacts", "deleted_items", "discussion_threads", "inbox", "notes_inbox", "sent", "sent_items" and "_sent_mail". We then flatten all the folder hierarchies and remove all the folders that contain fewer than three messages. We also remove the *X-folder* field in the message headers that actually contains the class label. Table 1 shows statistics on the seven resulting datasets.

5.2 SRI Email Dataset

From the February 2, 2004 snapshot of the SRI CALO directories, we select seven users with the largest number of messages: *acheyer, bmark, disrael, mgervasio, mgondek, rperrault,* and *vchaudri*. As in the preprocessing step of the Enron datasets, standard non-topical folders ("Inbox", "Drafts", "Sent" and "Trash") are ignored, the folder hierarchy is flattened, and folders that contain fewer than three messages are removed. Table 2 contains statistics on the resulting datasets.

Table 1: Statistics on Enron datasets—after removing non-topical and small folders.

User	***Number of folders***	***Number of messages***	***Size of smallest folder*** (messages)	***Size of largest folder*** (messages)	***Size of smallest message*** (words)	***Size of largest message*** (words)
beck-s	101	1971	3	166	45	2620
farmer-d	25	3672	5	1192	43	3507
kaminski-v	41	4477	3	547	44	7885
kitchen-l	47	4015	5	715	47	46296
lokay-m	11	2489	6	1159	45	4456
sanders-r	30	1188	4	420	55	19331
williams-w3	18	2769	3	1398	49	2287

Table 2: Statistics on SRI datasets—after removing non-topical and small folders.

User	***Number of folders***	***Number of messages***	***Size of smallest folder*** (messages)	***Size of largest folder*** (messages)	***Size of smallest message*** (words)	***Size of largest message*** (words)
acheyer	38	664	3	72	233	26682
bmark	15	268	4	32	228	26682
disrael	13	244	3	47	260	5947
mgervasio	15	777	6	116	214	17606
mgondek	14	297	3	94	214	26682
rperrault	38	751	3	197	252	26682
vchaudhri	10	558	8	205	292	26454

6. Results and Discussion

We apply Maximum Entropy (MaxEnt), Naive Bayes (NB), Support Vector Machine (SVM), and Winnow classifiers to each of the Enron and SRI datasets. We use the time-based training/test splits described in Section 3.3.

6.1 Experimental Setup

We report on two types of experiments:

- **Timeline.** We calculate the accuracy for each training/test split and plot the accuracy curve over the number of training messages in the splits. If a test set contains messages of a newly created folder, so that no messages of this folder have been seen in the training data, then such test messages are ignored in the accuracy calculation.

- **Coverage.** Accuracy is plotted over the percentage of test set coverage. This result is averaged over all the training/test set splits: for each split, after performing the actual classification, we first sort all the test messages according to a classification score[8] and then threshold the sorted list so that the first 10%, 20%, ..., 100% of messages are chosen. We then calculate the accuracy at each of the ten thresholds and average the

8. To compute the score, we use the probability of a message belonging to the folder for MaxEnt and Naive Bayes. For SVM, we use the (euclidian) distance to the classification hyperplane. For Winnow, we use the dot products between the examples and the learned weight vector of each class.

accuracies at each threshold over *all* training/test set splits. We present the mean accuracy and standard error at each threshold.

6.2 Results

Results on the seven Enron datasets are reported as the accuracy over the timeline in Figure 1 where the X-axis is the training set size. Figure 2 shows the accuracy-coverage curves, averaged over all the training/test set splits where the X-axis is the test set coverage. Error bars represent the standard error of the mean. Figure 3 and Figure 4 are analogous graphs for the SRI dataset. Table 3 summarizes the final results.

To assess statistical significance of the results, for each pair of classifiers we performed the paired-t test on accuracies obtained from all the training/test splits. We present in Table 4 the classifier rankings for each dataset.

Table 3: Final results on SRI and Enron datasets – accuracies averaged over all the training/test splits, with the standard error of the mean. Depending on the level of complexity and homogeneity of each test set, the classification performance can significantly vary, causing relatively large standard errors for the averaged results.

Enron user	***MaxEnt***	***Naive Bayes***	***SVM***	***Wide-margin Winnow***
beck-s	55.8 ± 2.2	32.0 ± 1.9	$\mathbf{56.4 \pm 2.1}$	49.9 ± 1.9
farmer-d	76.6 ± 1.3	64.8 ± 1.7	$\mathbf{77.5 \pm 1.3}$	74.6 ± 1.3
kaminski-v	55.7 ± 1.5	46.1 ± 1.9	$\mathbf{57.4 \pm 1.6}$	51.6 ± 1.5
kitchen-l	58.4 ± 1.6	35.6 ± 2.5	$\mathbf{59.1 \pm 1.5}$	54.6 ± 1.7
lokay-m	$\mathbf{83.6 \pm 0.9}$	75.0 ± 1.2	82.7 ± 1.0	81.8 ± 0.9
sanders-r	71.6 ± 4.5	56.8 ± 6.2	$\mathbf{73.0 \pm 4.3}$	72.1 ± 4.4
williams-w3	94.4 ± 2.3	92.2 ± 2.9	$\mathbf{94.6 \pm 2.1}$	94.5 ± 1.7
SRI user	***MaxEnt***	***Naive Bayes***	***SVM***	***Wide-margin Winnow***
acheyer	37.1 ± 2.2	27.0 ± 2.1	$\mathbf{38.5 \pm 2.5}$	37.0 ± 3.6
bmark	23.3 ± 8.2	13.5 ± 2.8	$\mathbf{24.3 \pm 7.1}$	23.9 ± 4.2
disrael	36.5 ± 5.5	36.5 ± 5.4	40.7 ± 5.9	$\mathbf{41.4 \pm 7.0}$
mgervasio	42.2 ± 5.1	31.3 ± 5.0	43.5 ± 4.5	$\mathbf{50.3 \pm 5.2}$
mgondek	75.2 ± 4.0	55.8 ± 6.2	$\mathbf{75.2 \pm 3.7}$	74.9 ± 5.1
rperrault	66.4 ± 4.1	49.8 ± 3.7	$\mathbf{68.1 \pm 3.5}$	62.7 ± 4.0
vchaudhri	61.1 ± 3.3	58.7 ± 4.7	58.6 ± 3.0	$\mathbf{62.7 \pm 3.5}$

6.3 Discussion

From Table 4, we see that MaxEnt, SVM and Winnow perform similarly on all the SRI datasets. On the Enron datasets, SVM demonstrates the highest accuracies, followed by MaxEnt, then wide-margin Winnow. Despite that, the *kaminski-v* dataset is the only dataset on which the prevalence of SVM over all the other classifiers is statistically significant.[9] In three cases (users *disrael*, *mgervasio* and *vchaudhri* of the SRI collection), wide-margin Winnow outperforms SVM—by notable 7% in the case of *mgervasio*, but the

9. Notably, the *kaminski-v* dataset is the largest one among all the fourteen datasets discussed in this paper.

Table 4: Results of performing paired-t tests on final results of SRI and Enron datasets ranking the accuracies averaged over all the training/test splits for Naive Bayes (NB), MaxEnt (ME), SVM, and wide-margin Winnow (WW). Each row ranks the classifiers from worst to best, with $\sim$, $<$, and $\ll$ denoting $P > 0.05$, $0.01 < P \leq 0.05$, and $P \leq 0.01$. All $\sim$ notations of are transitive except for the *lokay-m* dataset, where MaxEnt outperforms wide-margin Winnow with $0.01 < P \leq 0.05$.

Enron user	***Comparison of classifiers***						
beck-s	NB	$\ll$	WW	$\ll$	ME	$\sim$	SVM
farmer-d	NB	$\ll$	WW	$<$	ME	$<$	SVM
kaminski-v	NB	$\ll$	WW	$\ll$	ME	$\ll$	SVM
kitchen-l	NB	$\ll$	WW	$\ll$	ME	$\sim$	SVM
lokay-m	NB	$\ll$	WW	$\sim$	SVM	$\sim$	ME
sanders-r	NB	$\ll$	ME	$\sim$	WW	$\sim$	SVM
williams-w3	NB	$\ll$	ME	$\sim$	WW	$\sim$	SVM
SRI user	***Comparison of classifiers***						
acheyer	NB	$\ll$	WW	$\sim$	ME	$\sim$	SVM
bmark	NB	$\sim$	ME	$\sim$	WW	$\sim$	SVM
disrael	NB	$\sim$	ME	$\sim$	SVM	$\sim$	WW
mgervasio	NB	$<$	ME	$\sim$	SVM	$\sim$	WW
mgondek	NB	$<$	WW	$\sim$	ME	$\sim$	SVM
rperrault	NB	$\ll$	WW	$\sim$	ME	$\sim$	SVM
vchaudhri	NB	$\sim$	SVM	$\sim$	ME	$\sim$	WW

difference is not statistically significant. On the *lokay-m* dataset, MaxEnt shows the best performance, but again the difference is not statistically significant.

In eleven out of the fourteen cases, the Naive Bayes classifier is significantly inferior to the other three classifiers. It is widely believed that Naive Bayes is not the optimal solution for text categorization (see, e.g., (Dumais et al., 1998)). However, the performance of Naive Bayes could likely be improved by applying feature selection and/or a more sophisticated smoothing method than Laplace.

The experimental results presented support the following observations:

- Accuracies are higher on datasets that have one or two dominant folders. As it can be seen from Tables 1 and 2, in some cases the size of the largest folder is up to one half of the entire dataset size (see, e.g., statistics on *lokay-m* and *vchaudhri*). Obviously, this facilitates the classification process.

- Newly created folders adversely affects the classification accuracy. When a new folder is created, it is represented in the training set by just a few instances. Obviously, they do not capture the essence of the folder, so test instances of this class are likely to be misclassified.

Generally speaking, the timeline curves show unstable, spiky behavior. One might expect the classification performance improve when the training set size increases, but this rarely happens in practice. It can be explained by the observation that email is usually

related to other recently received email, rather than to email received long ago. Thus, old email in the training set probably does not affect the classification procedure.

The classification results are surprisingly low. They show the real complexity of the task of categorizing email, as opposed to the regular topical text classification. They are obtained on the realistic time-based evaluation setup. If we applied random training/test splits (method that is standard for the regular text classification), we would obtain much higher results. For example, on the *acheyer* dataset we can obtain $65.8 \pm 2.3\%$ accuracy using 4-fold cross validation, while on time-base splits we can only achieve $38.5 \pm 2.5\%$.

6.3.1 Dataset phenomena

The *williams-w3* dataset is a special, degenerative case of an email dataset. It basically consists of only two large folders, named "*bill_williams_iii*" and "*schedule_crawler*", which appear to be archives that contain various types of messages. The former folder contains most of early messages, while the latter one contains almost all the later messages. Obviously, classification with one most probable class is an easy task, which explains why the accuracies are so high. At the point of 1300 messages one archive is exchanged with another, causing a serious drop in the performance.

We notice an interesting phenomenon in the *sanders-r* dataset. The performance of the classifiers are significantly worse when the training sets are large (1000 and 1100 instances). There are two different reasons for this drop: at point 1000 a new folder is created that dominates in the test set, while at point 1100 an old folder that has been practically abandoned for two years returns to focus again, probably with new, unrelated topics discussed.

We also notice that worse accuracy is sometimes paradoxically obtained at low coverage rather than at higher coverage (see, e.g. coverage graphs of *sanders-r* and *rperrault*). Such behavior is a result of creating new folders. When a new folder is created, in case of the SVM for example, the classification hyperplane is not yet adjusted for the new class, so the (misclassified) test messages of the new folder accidentally fall far from the hyperplane. Classification of such messages accidentally becomes very confident, which implies that they are taken into account at low coverage. When averaging over all the splits, these rare but steep drops thus affect the overall accuracy and the standard error at low coverage. We can conclude from the above that the considered classifiers generally are not robust to non-stationary data.

One strange artifact of the Enron dataset is that, if a folder at some time point was moved from one place to another in the foldering hierarchy, the dataset preserves both versions of this folder. For instance, the significant drop in the foldering accuracy at point 500 in timeline of the *beck-s* dataset is primarily caused by this phenomenon: two folders *global/japan* and *japan* appear to be the same folder that had been relocated.

6.3.2 Running time

When considering the running time of the three equally accurate classifiers (MaxEnt, SVM and Winnow), we note that Winnow is by far the fastest classifier of the three. It takes no more than 1.5 minutes to train even on largest training sets. SVM training time is more than an order of magnitude slower (which still is not prohibitive), however, our model selection method requires classifier retraining (see Section 4), significantly slowing the process.

Training SVM can take up to half an hour, and it is almost independent of the size of the training set. MaxEnt training is very time consuming, in some cases (such as *kitchen-l* dataset) one MaxEnt classifier training takes two hours, and the training time is significantly dependent on the training set size. The Naive Bayes classifier is the fastest of all the classifiers we used, but, as has already been mentioned, its accuracy is significantly lower. Note that in order to obtain the results presented in Figures 1-4, several classifiers were trained (one classifier per each training/test set split).

6.3.3 Ablation experiments

To compare our version of the Winnow classifier to the one described in (Hurst and Nigam, 2003), we present the accuracies of the two versions in Table 5. As one can see from the table, wide-margin Winnow outperforms regular Winnow by a large and statistically significant margin. We note that most of the improvements come from simply iterating through the data several times to stabilize the learned weight vector.

The only datasets for which wide-margin Winnow does not perform statistically significantly better than regular Winnow are *williams-w3* and *bmark*. The *williams-w3* dataset is a degenerate case, as described in Section 6.3.1, leading to extremely high accuracies on all classifiers we trained. The *bmark* dataset is one of the smallest and the most problematic dataset among the fourteen presented in this paper. All the four classifiers perform poorly on this dataset (less than 25% averaged accuracy with high standard error), leading to statistical insignificance in the difference between the results.

Table 5: Final results on SRI and Enron datasets comparing plain Winnow with wide-margin Winnow. Each entry is the average accuracy $\pm$ the standard error. Results at entries marked by †(‡) are statistically significantly better at the 0.05 (0.01) level from a paired t test.

Enron user	***Winnow***	***Wide-margin Winnow***
beck-s	22.6 ± 2.6	49.9 ± 1.9‡
farmer-d	64.6 ± 2.2	74.6 ± 1.3‡
kaminski-v	30.9 ± 2.1	51.6 ± 1.5‡
kitchen-l	42.8 ± 2.9	54.6 ± 1.7‡
lokay-m	62.3 ± 3.2	81.8 ± 0.9‡
sanders-r	56.1 ± 7.3	72.1 ± 4.4†
williams-w3	94.8 ± 2.5	94.5 ± 1.7
SRI user	***Winnow***	***Wide-margin Winnow***
acheyer	22.2 ± 3.2	37.0 ± 3.6‡
bmark	18.2 ± 7.8	23.9 ± 4.2
disrael	13.6 ± 3.3	41.4 ± 7.0‡
mgervasio	33.5 ± 6.0	50.3 ± 5.2‡
mgondek	55.6 ± 7.2	74.9 ± 5.1†
rperrault	47.8 ± 5.9	62.7 ± 4.0‡
vchaudhri	48.8 ± 4.7	62.7 ± 3.5‡

7. Conclusion

Email foldering is a rich and interesting task. It differs from the standard (topical) text classification in its highly subjective and non-monotonic foldering schema. Folders are constantly being created and abandoned, becoming more active and less active, and even their major common topics are changing over time. In addition, users tend to folder some messages by sender, some by event, some by date and in many cases the logic behind a certain foldering choice can be difficult to discern. Therefore, an email foldering system should be adaptive to the working style of individual users.

The Enron Email dataset for the first time makes available a large real-world collection for shared experimentation with email. In this paper we have proposed an easy and intuitive framework for comparing email foldering results. Preprocessed datasets of the seven former Enron employees are ready for download from http://www.cs.umass.edu/∼ronb.

Standard evaluation techniques based on random training/test set splits are not applicable to the foldering task, because of the time-dependent nature of the data. More appropriate (but more complicated) methods are to be used for the evaluation. We have proposed the step-incremental time-based split that provides a realistic evaluation setup and allows us to examine the statistical significance of the foldering results.

We have applied four classifiers for the email foldering task: Maximum Entropy, Naive Bayes, Support Vector Machine and Winnow. We have presented a new version of the Winnow algorithm and provided detailed information about its implementation. When comparing the four classification techniques on email foldering, we have shown that a fast and simple-to-implement Winnow classifier performs not worse and even sometimes insignificantly better than the more popular and more complex-to-implement SVM and MaxEnt methods.

With respect to the problem difficulty, we note that the obtained foldering accuracies are relatively low: in nine of fourteen cases they are below 70%. There is much room to improve the baseline. Sophisticated methods should be applied to the email foldering task. Interesting avenues for future investigation include relational methods (Getoor et al., 2001), Topic Detection and Tracking (Allan, 2002) and feature induction techniques (Bekkerman et al., 2003; McCallum, 2003).

Acknowledgements

We are grateful to Avrim Blum for providing implementation notes and many helpful details of the Winnow algorithm. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. and in part by AFRL under contract number F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of AFRL or the U.S. Government.

References

J. Allan. *Topic Detection and Tracking, Event-based Information Organization*, chapter 1, pages 1–16. Kluwer Academic Publishers, 2002.

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3:1183–1208, 2003.

A. Blum. *Online Algorithms: The State of the Art*, chapter On-line Algorithms in Machine Learning. Springer, 1996.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learing Theory*, pages 144–152, 1992.

R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. Technical Report NAM-03, Northwestern University, 1996.

M. F. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. 2001.

P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

W. W. Cohen. Learning rules that classify e-mail. In *Proceedings of AAAI Spring Symposium on Machine Learning and Information Retrieval*, 1996.

W. W. Cohen and S. Sarawagi. Combining semi-markov extraction: Exploiting dictionaries in named entity extraction:. In *Proceedings of SIGKDD'04, 10th ACM International Conference on Knowledge Discovery and Data Mining*, 2004.

I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *Proceedings of EMNLP'97, 2nd Conference on Empirical Methods in Natural Language Processing*, pages 55–63, 1997.

J. Diederich, J. L. Kindermann, E. Leopold, and G. Paaß. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1/2):109–123, 2003.

H. Drucker, V. Vapnik, and D. Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM'98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998.

A. Finn, N. Kushmerick, and B. Smyth. Genre classification and domain transfer for information filtering. In F. Crestani, M. Girolami, and C. J. van Rijsbergen, editors, *Proceedings of ECIR-02, 24th European Colloquium on Information Retrieval Research*, pages 353–362, Glasgow, UK, 2002.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proceedings of ICML'01, 18th International Conference on Machine Learning*, 2001.

E. Giacoletto and K. Aberer. Automatic expansion of manual email classifications based on text analysis. In *Proceedings of ODBASE'03, the 2nd International Conference on Ontologies, Databases, and Applications of Semantics*, pages 785–802, 2003.

M. F. Hurst and K. Nigam. Retrieving topical sentiments from online document collections. In *Proceedings of the 11th Conference on Document Recognition and Retrieval*, pages 27–34, 2003.

T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers, Dordrecht, NL, 2002.

S. Kiritchenko and S. Matwin. Email classification with co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, 2001.

J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of SIGKDD'02, 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002.

B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proceedings of ECML'04, 15th European Conference on Machine Learning*, pages 217–226, 2004.

M. Koppel, S. Argamon, and A. R. Shimoni. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4), 2003.

D. D. Lewis. *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

D. D. Lewis and K. A. Knowles. Threading electronic mail: a preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2(4):245–318, 1988.

A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of UAI'03, 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998.

C. Mesterharm. Using linear-threshold algorithms to combine multi-class sub-experts. In *Proceedings of ICML'03, 20th International Conference on Machine Learning*, pages 544–551, 2003.

K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of IJCAI'99, Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

T. R. Payne and P. Edwards. Interface agents that learn: An investigation of learning issues in a mail agent interface. *Applied Artificial Intelligence*, 11(1):1–32, 1997.

S. Della Pietra, V. J. Della Pietra, and J. D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

J. Provost. Naive-bayes vs. rule-learning in classification of email. Technical Report AI-TR-99-284, University of Texas at Austin, Artificial Intelligence Lab, 1999.

J. Rennie. ifile: An application of machine learning to e-mail filtering. In *Proceedings of KDD'2000 Workshop on Text Mining*, 2000.

J. Rennie. Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology, 2001.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

R. Segal and J. Kephart. Incremental learning in swiftfile. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, 2000.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML'97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

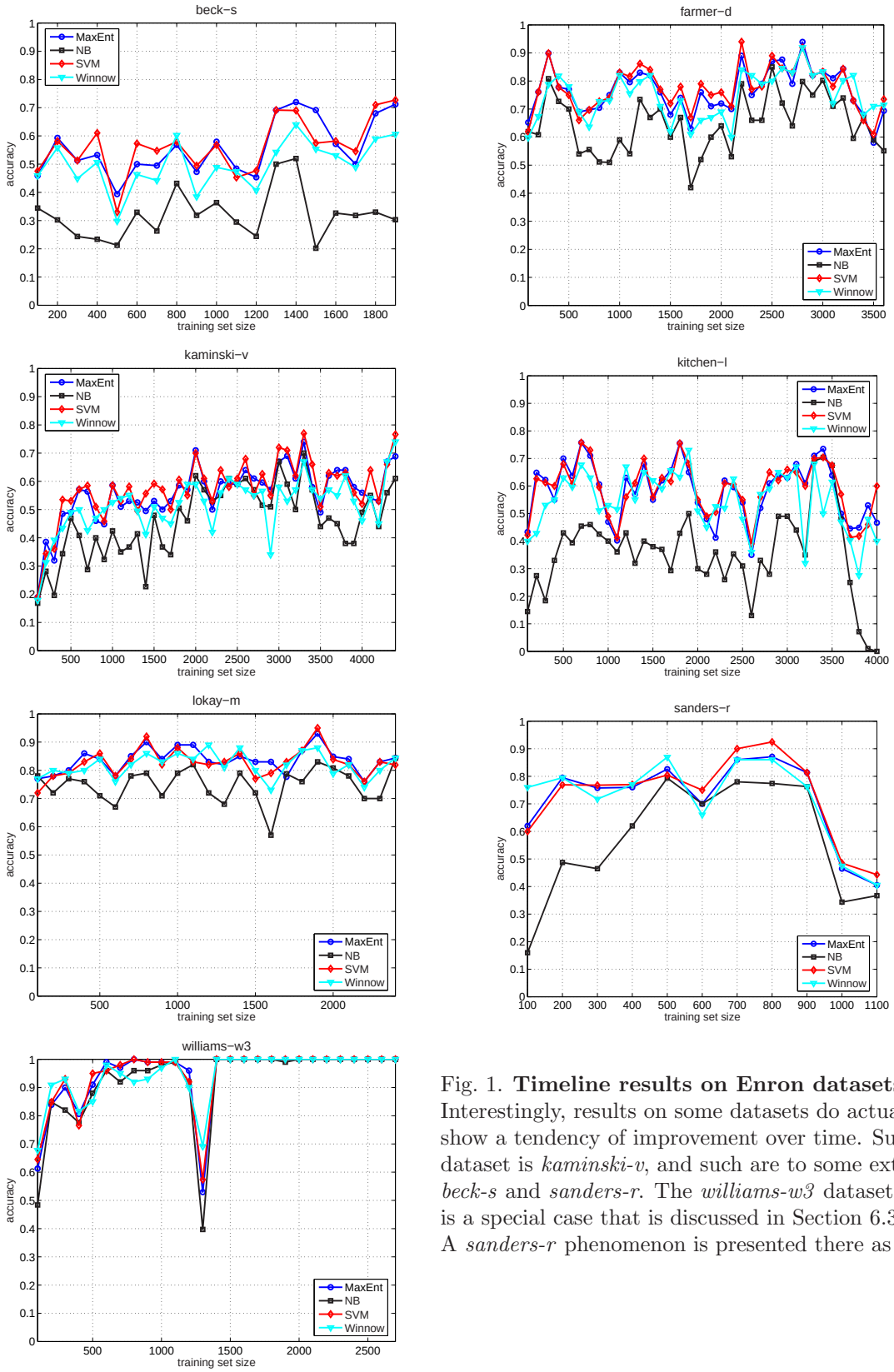

Fig. 1. **Timeline results on Enron datasets.** Interestingly, results on some datasets do actually show a tendency of improvement over time. Such dataset is *kaminski-v*, and such are to some extent *beck-s* and *sanders-r*. The *williams-w3* dataset is a special case that is discussed in Section 6.3.1. A *sanders-r* phenomenon is presented there as well.

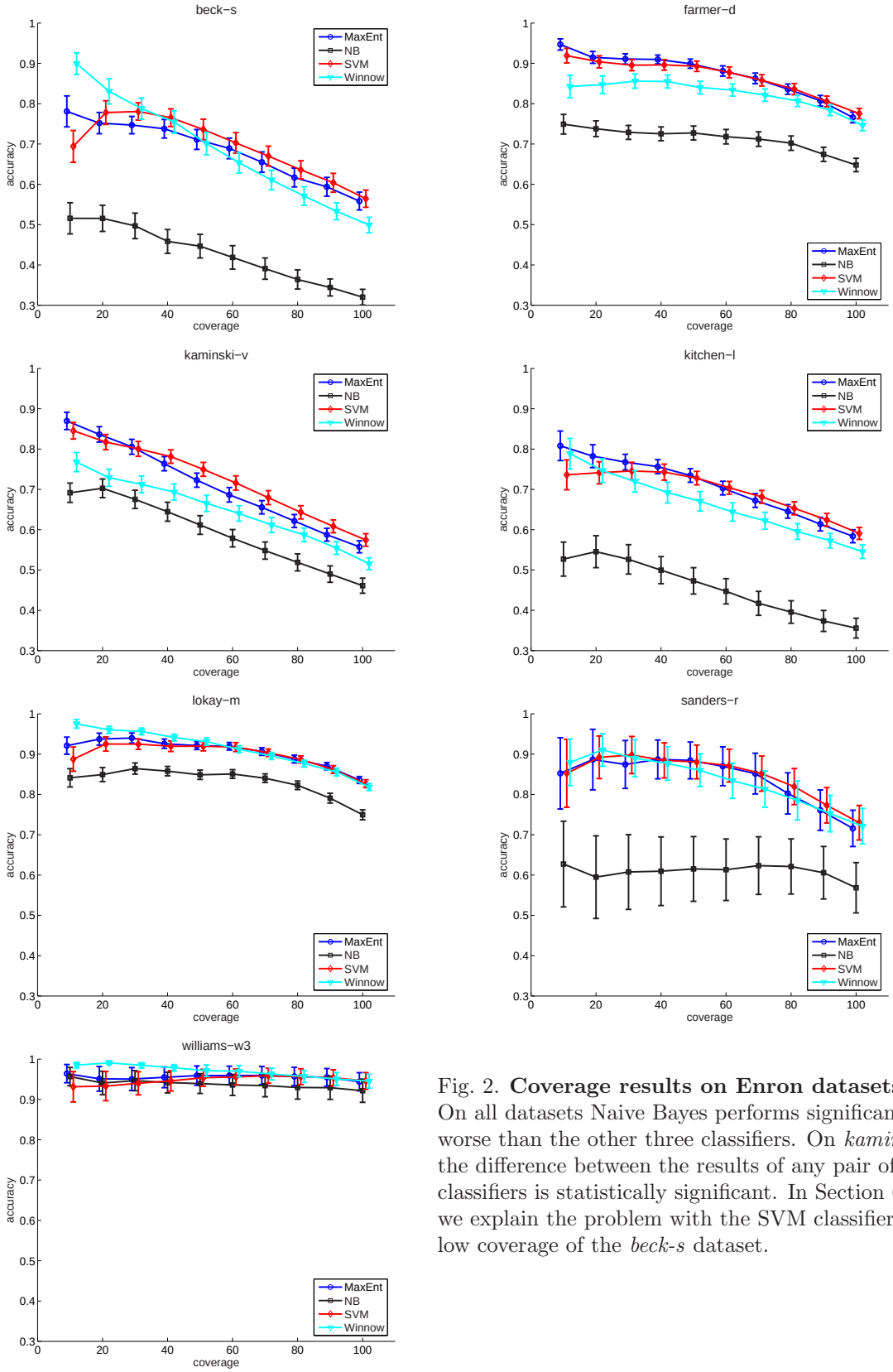

Fig. 2. **Coverage results on Enron datasets.** On all datasets Naive Bayes performs significantly worse than the other three classifiers. On *kaminski-v* the difference between the results of any pair of classifiers is statistically significant. In Section 6.3.1 we explain the problem with the SVM classifier at low coverage of the *beck-s* dataset.

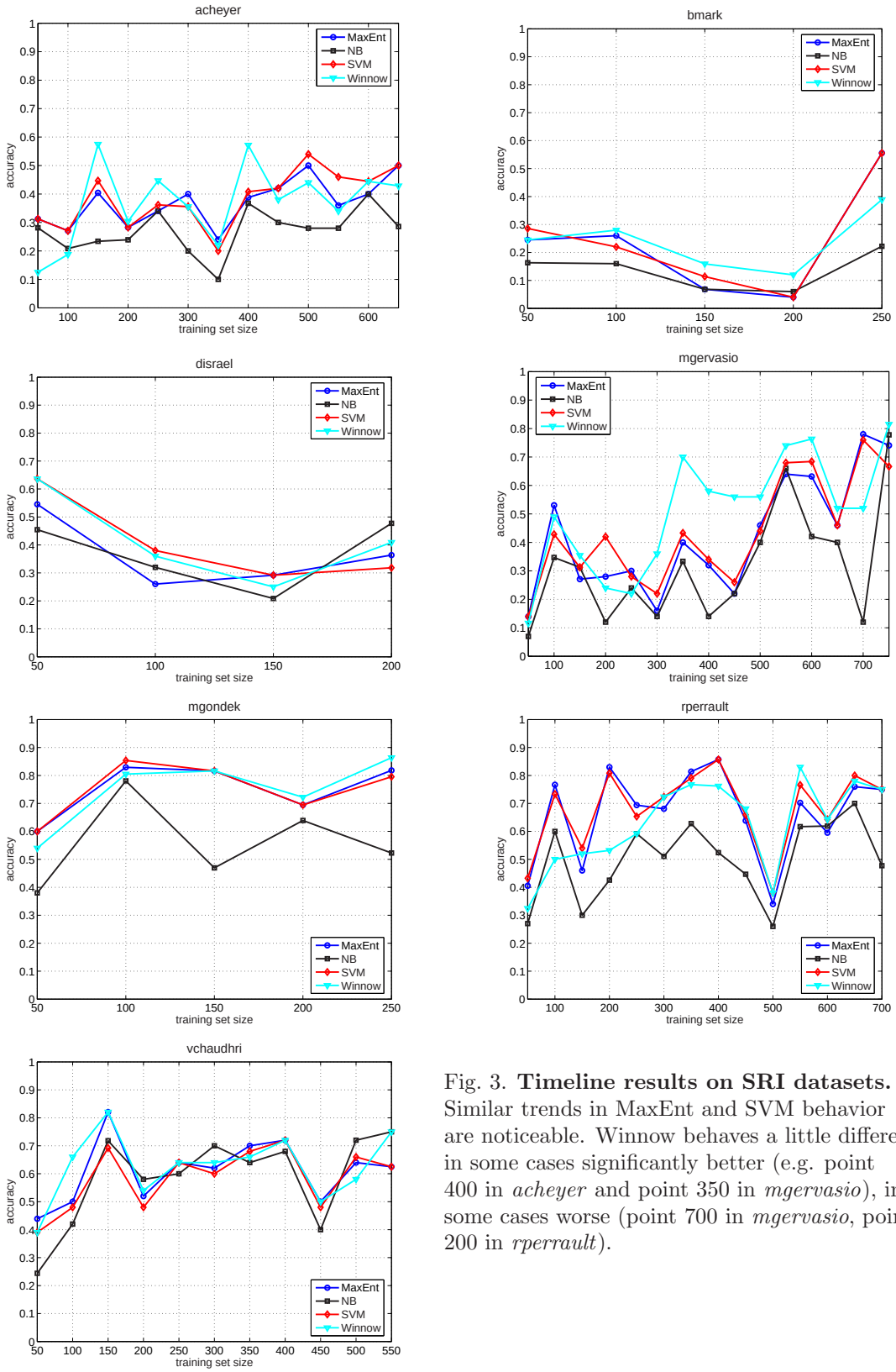

Fig. 3. **Timeline results on SRI datasets.** Similar trends in MaxEnt and SVM behavior are noticeable. Winnow behaves a little differently, in some cases significantly better (e.g. point 400 in *acheyer* and point 350 in *mgervasio*), in some cases worse (point 700 in *mgervasio*, point 200 in *rperrault*).

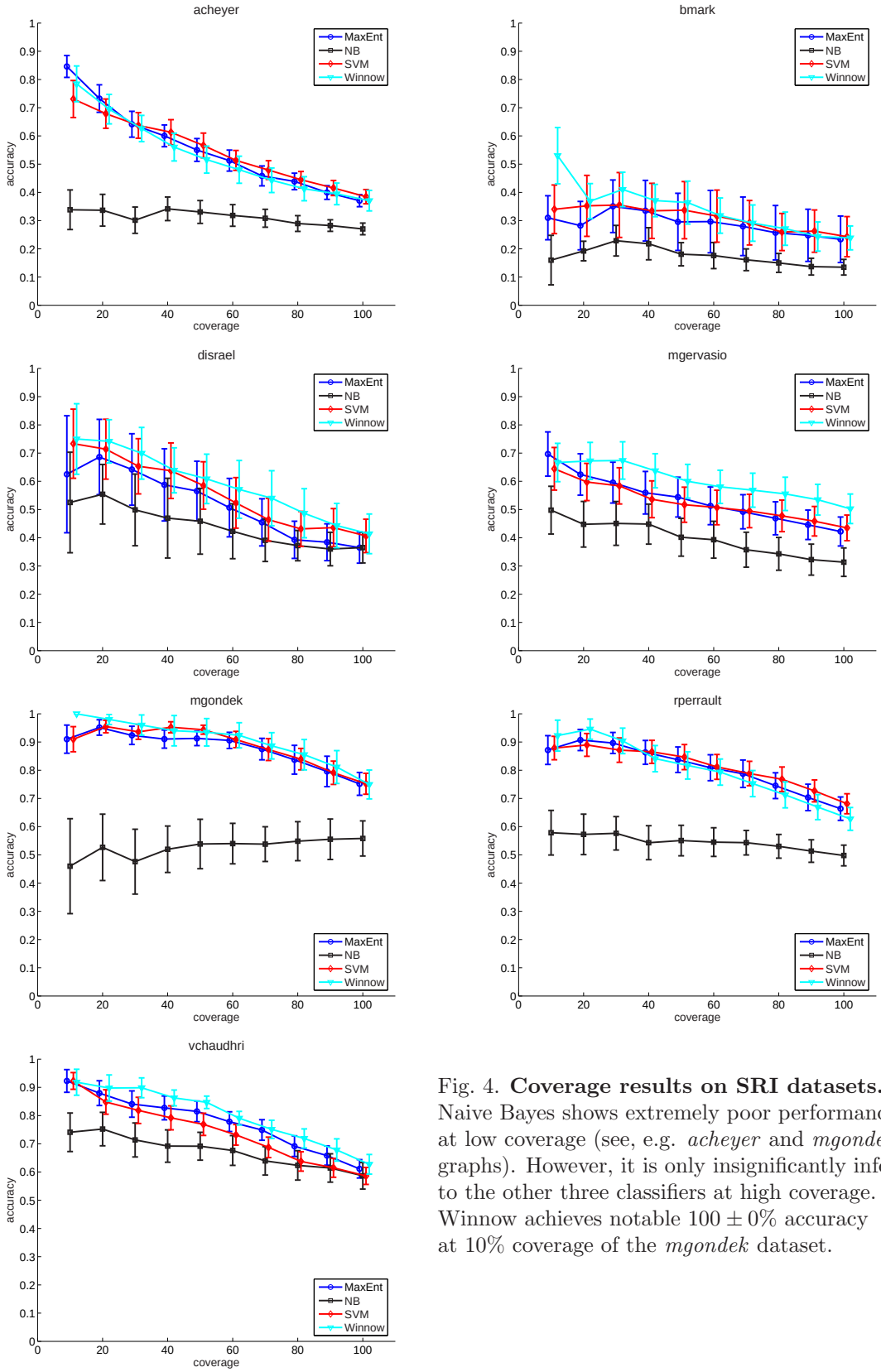

Fig. 4. **Coverage results on SRI datasets.** Naive Bayes shows extremely poor performance at low coverage (see, e.g. *acheyer* and *mgondek* graphs). However, it is only insignificantly inferior to the other three classifiers at high coverage. Winnow achieves notable $100 \pm 0\%$ accuracy at 10% coverage of the *mgondek* dataset.