Challenges of the Email Domain for Text Classification

Jake D. Brutlag JAKEB@MICROSOFT.COM
Christopher Meek MEEK@MICROSOFT.COM
Microsoft Research, One Microsoft Way, Redmond, WA, 98052 USA

Abstract

Interactive classification of email into a user-defined hierarchy of folders is a natural domain for application of text classification methods. This domain presents several challenges. First, the user's changing mail-filing habits mandate classification technology adapt in a dynamic environment. Second, the classification technology needs to be able to handle heterogeneity in folder content and folder size. Performance when there are only a small number of messages in a folder is especially important. Third, methods must meet the processing and memory requirements of a software implementation. We study three promising methods and present an analysis of their behavior with respect to these domain-specific challenges.

1. Introduction

Electronic mail is one of the primary applications of the Internet. Users of electronic mail often organize their messages into custom hierarchies of folders. A task routinely performed by users is filing mail from an Inbox into this hierarchy. This filing task may become tedious if the user has a high volume of mail or the user has a large number of filing folders.

In this age of 'smart agent' computing, it is natural to propose an intelligent agent to tackle the email filing task. One role for such an agent is that of surrogate. In this role, an email classification agent processes incoming messages and files them for the user. The action of such an agent is automatic, and need not be restricted to filing. For example, Re:Agent, developed at Georgia Tech, incorporates the actions of delete, forward, and autoreply (Boone, 1998).

A second role for a smart agent is that of an assistant. In this role, the agent interacts with the user, providing assistance as the user processes messages. For the filing task, this interactive agent proposes candidate folders for a message. This leaves the user in control of message processing while easing the cognitive burden of filing (Boone, 1998; Segal & Kephart, 1999).

In this paper we focus on three challenges of applying classification technology to the problem of interactive email filing. The challenges are dynamic nature of the email domain, the heterogeneous nature of folder content and of folder sizes, and practical memory and CPU limitations of an implementation. We evaluate three types of text classification methods with respect to these challenges. In particular, we compare a batch support vector machine (SVM) to two incremental approaches to text classification. We show that the incremental algorithms perform competitively for our classification task. Section 2 describes the email domain, and specific challenges that arise in that domain. Section 3 discusses text classification methods considered, and the motivation for considering those methods. Section 4 presents the results of an evaluation of the selected methods on data extracted from user mail stores. Section 5 concludes with a discussion of the results.

2. Challenges of the Domain

The email domain presents several challenges for an interactive filing agent.

An email system is a dynamic environment. In an email system, "folders and messages are constantly being created destroyed, and reorganized" (Segal & Kephart, 1999). For text classification, this has two ramifications. First, the number of classes may change over time. Second, the content associated with class labels may change over time.

User folder hierarchies contain a large number of sparse folder. Sparcity refers to the number of messages filed into a folder. A 1996 user study of 20 users conducted by Whittaker and Sidner at Lotus found 35% of the average user's folders contain fewer than three messages (Whittaker & Sidner, 1996). On the other hand, it is not uncommon for folder hierarchies to contain one or more dense folders, which contain hundreds of messages. In general, there is a significant amount of variance in the sizes of folders in a hierarchy.

User folder hierarchies are heterogeneous. Creating a set of filing folders is a difficult task (Whittaker & Sidner, 1996), and the resulting hierarchy is unlikely to resemble a well-defined topical hierarchy. Folders in a hierarchy may overlap in content, and a subfolder may not represent a subtopic of the parent folder content.

User folder content is heterogeneous. Heterogeneity refers to the similarity of text content between messages filed into the same folder. In the mail stores collected for our evaluation, folder labels included 'Personal' and 'Fun'. We can assume the contents of these folders are as vague as labels. This can be a problem for both sparse and dense folders. In a dense folder, the "relationships between different messages in the folder become tenuous" as more messages accumulate (Whittaker & Sidner, 1996).

A software implementation must minimize memory and CPU impact. In both client-side and server-side implementations, excessive computation or a large in-memory footprint will adversely impact the user's experience and limit the scalability of the solution.

3. Text Classification Technologies for Email

In this section we briefly describe text classification as applied to the email domain. We then describe three broad classes of classifiers and describe the three specific classification technologies used for our evaluation. The specific classification technologies are a support vector machine (SVM), a TF-IDF classifier, and a simple language model called the unigram model. Previous work has demonstrated that support vector machines (SVMs) are exceptionally well suited to text classification. (Dumais, Platt, Heckerman, & Sahami, 1998) Unfortunately, SVMs have several shortcomings with respect to the challenges of our domain. We describe each of the methods focusing on issues related to the challenges of our domain.

3.1 Text Classification

In the email classification problem we have a set of email documents, the *test documents*, that are to be classified into a set of folders (the *classes*). We are given a set of folders that contain email documents that have previously been filed by a user. These documents are the *training documents.* Each training document has a class label by virtue of the fact that it is a member of some folder.

We use the term *learning* to denote the process by which a set of labeled training examples are used to create a set of classifiers for new documents. *Batch learning* occurs when a batch of labeled documents is accumulated and processed simultaneously. The classifier is fit or updated once for the batch. In contrast, *incremental learning* updates the classifier for each document, processed one-by-one. Using an incremental learning system as opposed to a batch system is one means of addressing the dynamics of the email domain. However, some classification technology is not amenable to incremental learning.

An important choice in designing a text classifier is the choice of how to represent or featurize the text documents. We convert each text document into a vector representation in which each component of the vector represents a unique word. In a word-frequency vector, each component records the number of times the word occurs in the document. In a word-occurrence vector, each component indicates the presence or absence of the word in the document. However, email messages are more than free-form text. While the body of a message is unstructured text, a message also contains a structured header. The header fields include the sender (the *From* field), a list of recipients (the *To* field), and short description (the *Subject* field). We treat the text in the *From*, *To*, *Subject*, and *Body* field separately (Cohen, 1996). For example, if a name 'Thomas' appears in the *From*, *Subject* and *Body* fields, it is recorded in three separate components of the word-frequency or word-occurrence vector.

A technique for managing the computational costs associated with text classification is feature selection. For our application, feature selection amounts to pruning the word list. A reduced feature set decreases the CPU and memory demands of an implementation through reduced featurization costs, decreased training time, and a smaller footprint for the classifiers. There are a variety of computationally inexpensive methods that can be applied globally across a set of classifiers. One such method for the email domain is to ignore all features associated with the body of the message. We evaluate using only header information for classification in section 4.5.

A conventional approach to feature selection is to use a scoring criteria to choose among the available features. In the email domain the global feature set (words) is not static and the relevant features for discrimination may change over time as folder contents evolve. The results of feature selection and projection may be invalidated by a single new example, especially in the case of a sparse folder with one or two messages. Typically, feature selection, projection, and retraining are too CPU-intensive to perform in real time whenever a single new example is added to a folder. Therefore, methods that rely on feature selection to control computational cost are often only used in a batch learning mode. This is a significant drawback in the dynamic email filing domain.

3.2 Discriminant Classifiers

Word-frequency or word-occurrence vectors are points in a high dimensional space. The discriminant approach attempts to partition this space into regions for each class using the labeled examples according to an optimization criteria.

In our experiments using discriminant methods and in particular SVMs, we fit a separate (two-way) classifiers for each folder where each classifier distinguishes between members and non-members for the folder. Assuming each classifier assigns a probability of membership, the classes can be ranked according to probable membership. Some discriminative methods can be used to construct a single N-way classifier which partitions the high dimensional space into N classes associated with one class for each folder. However, this is often a computationally expensive optimization problem, and presupposes the classes are mutually exclusive. In addition to simplifying the optimization problem, using the approach of building separate classifiers, one can retrain individual classifiers as needed making the approach more incremental.

In a preliminary analysis of discriminative methods, we compared several methods, including single-layer (perceptron) neural networks, decision trees, linear discriminant analysis (LDA), and linear support vector machines (SVMs). As identified by other researchers (Dumais et al., 1998; Yang & Liu, 1999), we found linear SVMs are often the best discriminant classifiers. Therefore, we decided to select linear SVMs as a representative discriminant classifier. Our methodology for application of this method closely follows that of (Dumais et al., 1998).

A linear SVM is a hyperplane that separates a set of positive and negative examples with maximum margin, assuming the points are linearly separable (Platt, 1998). The margin is the distance from the hyperplane to the nearest of the positive and negative examples. In the case where points that are not linearly separable, slack variables are introduced which permit, but penalizes, points that violate the margin.

The linear SVM optimization criteria is:

$$\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i \tag{1}$$

where: $\mathbf{w}$ denotes the normal vector of the hyperplane and b is a scalar offset. The word-frequency or word-occurrence vector of example i (of n) is denoted by $\mathbf{x}_i$, the ξ_i are the slack variables and y_i is the target (+1 for examples belonging to the class, -1 for those that do not). Finally, C is a tuning parameter that trades off the number and magnitude of margin failures with the margin width.

An effective algorithm for optimizing this criteria is Sequential Minimal Optimization (SMO) (Platt, 1998). The efficiency of this algorithm implies that a sizeable portion of the CPU cost of applying linear SVMs to the email classification task is feature selection and featurization. This approach requires that the training examples be stored in memory during the optimization. Thus, there is a potentially significant memory cost associated with training and a possible cost associated with refeaturization if the implementation chooses not to store the featurized messages.

The classification function for a linear SVM is $\mathbf{w}\cdot\mathbf{x} - b$, for a incoming feature vector $\mathbf{x}$. This score is translated via a sigmoid function to the posterior probability that $\mathbf{x}$ belongs to the folder represented by the class. In our experiments, C was set according to an empirical scaling rule derived from experiments on independent data sets.

Following the methodology of Dumais et al. (1998) we use mutual information for feature selection and word occurrence vectors. In our preliminary analysis, we experimented with feature sets of size 500, 1000 and 2000. Our results showed a significant improvement in accuracy for 1000 features over 500, but an insignificant difference between 1000 and 2000 features. In this paper, we present results for 1000 features.

3.3 TF-IDF Classifiers

TF-IDF classifiers, also referred to as Information Retrieval (IR) classifiers, represent each class with a single vector in the feature space. This centroid is a vector of weights and each weight is the product of term frequency (TF) and inverse document frequency (IDF). For each folder, a similarity score is computed between the word-frequency vector of the incoming message and the folder centroid. The folders can then be ranked according to the scores.

The term frequency for each word is a measure of the relevancy of that word within the class (folder). The document frequency is a measure of the global importance of the word (across all folders). The TF and

IDF expressions vary with the classifier, but they are designed to be simple to compute from a set summary statistics for features within each folder and across all folders. The computations are simple enough that they can be deferred until classification; therefore learning reduces to collecting and updating the statistics. As these statistics can be updated incrementally, TF-IDF classifiers support incremental learning.

The similarity measure also varies with the classifier, but it is usually some form of normalized dot product between the class centroid and the message word-frequency vector.

In this investigation, we focus on the incremental TF-IDF classifier applied to interactive email classification by Segal and Kephart (Segal & Kephart, 1999).

The following expressions define the term frequency (TF) and inverse document frequency (IDF) components of the IR classifier (Segal & Kephart, 1999):

$$\mathrm{TF}\,(w,f) = {rate\,(w,f)}/{rate(w, corpus)} \tag{2}$$

$$\mathrm{IDF}\,(w) = \left({\#\text{folders}}/{\#\text{folders with } w}\right)^2 \tag{3}$$

where $rate(w,f)$ is the number of times w occurs in f divided by the total number of words in f. In many IR approaches, the TF component is a function of the within class frequency alone, not the within class frequency and the global frequency. Note that because of the computational overhead for feature selection described in Section 3.1 we do no feature selection for the IR method.

For the IR classifier, the TF-IDF weight selects for words with a relatively high within folder rate that occur within very few folders. The square function in equation 3 drastically increases the weight of specialized words; other TF-IDF classifiers substitute a log or identify function in the IDF component.

The similarity score for the IR classifier is SIM4 (Segal & Kephart, 1999):

$$\mathrm{Sim}\,(m,f) = \frac{\sum_w count(w,m)\mathrm{TF}\,(w,f)\,\mathrm{IDF}\,(w)}{\min\left(\sum_w count(w,m), \sum_w \mathrm{TF}\,(w,f)\,\mathrm{IDF}\,(w)\right)} \tag{4}$$

The sums in equation 4 are all over the unique words in the message to be classified. The numerator of the SIM4 score is the dot-product of the message word-frequency vector and the classifier weight vector; SIM4 is a variation of the cosine distance.

The net effect of the IR classifier is to identify specialized keywords, preferably keywords unique to a single class. In a folder hierarchy with a large number of folders, a unique keyword match between the message and the folder can dominate the similarity score.

3.4 Language Models

Developed for speech, language models attempt to predict the probability of the next word in an ordered sequence (Goodman & Chen, 1996). Language models attempt to describe the generation of the string of words in a text document.

Suppose $w_1, \ldots, w_m$ are the words in-sequence of an incoming message. Then a natural language model proposes a likelihood of that sequence given a particular folder f, i.e. $p(w_1, \ldots, w_m|f)$.

$$\begin{aligned} p(w_1, \ldots, w_m|f) &= p(w_1|f)p(w_2|w_1, f) \\ &\ldots\ p(w_m|w_{m-1}, \ldots w_1, f) \end{aligned} \tag{5}$$

Language models use various approximations of the terms in the RHS of Equation 5, e.g., $p(w_i|w_1, \ldots, w_{i-1}, f) \approx p(w_i|w_{i-1}, f)$. To obtain $p(f|w_1, \ldots, w_n)$ one simply applies Bayes Rule. Using a uniform prior over the folders, as we do in our experiments, leads to a posterior proportional to the likelihood, $p(f|w_1, \ldots, w_m) \propto p(w_1, \ldots, w_m|f)$. The folders are ranked according the posterior probabilities to find the folders most likely to have generated the message.

For a variety of language models, simplifying assumptions can be made in estimating the probabilities on the RHS of equation 5 such that the models are amenable to incremental learning. The unigram language model is such a model. In our evaluation, we evaluate a unigram classifier.

The Unigram classifier is a simple language model. The Unigram model assumes that each word occurs in a message independently of all others given the folder:

$$p(w_k|w_{k-1}, \ldots w_1, f) = p(w_k|f) \quad \forall k \tag{6}$$

In this case, $w_1, \ldots w_m$ are a sample from a Multinomial distribution with a parameter vector θ. The maximum likelihood estimate of θ_{w_k}, the component of θ corresponding to the word w_k, is simply the within-folder rate of w_k. A smoothed estimate of θ_{w_k} is obtained using the global rate of occurrence for w_k:

$$\hat{\theta}_{w_k} = p\,(w_k|f) = \frac{count\,(w_k, f) + \kappa\,(globalrate\,(w_k))}{wordcount\,(f) + \kappa} \tag{7}$$

Here κ is a smoothing parameter. This smoothed estimate avoids the pitfall of a zero estimate for words that do not occur in a specific folder. New words in an incoming message (not yet observed in any folder) are removed from the message prior to computing the likelihood. This embodies the assumption that the likelihood of a new words is the same for each folder. Note that because of the computational overhead for feature selection described in Section 3.1 we do no feature selection for the unigram method.

The Unigram classifier supports incremental learning because it only depends on these rates, which are simple to compute from feature within folder statistics.

This method of estimation for the parameters in the Unigram classifier is an approximate empirical Bayes approach. The prior counts of the Dirichlet prior are the global rates of each word scaled by κ. In our experiments we provide results with $\kappa = 1$.

3.5 Other Approaches

Classification approaches are not limited to these three categories and each category admits a wide variety of alternatives. One class of alternative methods utilize hierarchical information. As mail store folders are organized into a hierarchy, this may seem to be a promising approach. In fact, researchers have illustrated the benefits of hierarchical classification when the hierarchy of categories represents a hierarchy of content (Koller & Sahami, 1997). However, a hierarchy of email folders is not guaranteed to be a hierarchy of content categories and our preliminary analysis of one hierarchical method supports this view. Furthermore, survey data suggests that the folder hierarchies of user mail stores are often flat.

4. Evaluation

We conducted an experiment on five user mail stores to address the following goals:

- Compare the accuracy of the three classifiers to establish that incremental learning methods offer competitive accuracy despite heterogeneous folder content and size.
- Assess the performance of classifiers with respect to sparse folders.
- Compare the accuracy of restricting input to message headers versus featurizing the entire message (including headers).

The characteristics of the user mail stores are summarized in Table 1. The first two columns list the number of non-empty folders and total message count in the hierarchy. A folder is considered empty if it has subfolders, but does not directly contain any messages. The third and fourth columns list the percentage of all messages contained in folders with fewer than 10 or 20 messages. The fifth column lists the average message depth for messages in the hierarchy. The message depth for a message is its depth in the hierarchy where a depth of one means the message is in the root folder.

Table 1. Characteristics of User Mail Stores.

Store	# f	# Mess.	% Mess. in Folder < 10	% Mess. in Folder < 20	Avg Mess. Depth
H	670	10262	15%	28%	3.369
M	144	1427	19%	43%	3.66
T	14	766	1%	10%	1.975
P	23	1783	2%	5%	2.28
D	387	7448	10%	28%	2.97

These mail stores were not randomly selected, and are atypical in the context of data available from surveys (Whittaker & Sidner, 1996). Nevertheless, they represent a diverse set of stores, which will provide an accurate picture of the relative behavior of the classifiers considered.

4.1 Experiment Setup

It is difficult to collect data directly from a dynamic email system. Instead, we performed experiments on static data sets. First, for each mail store, we extracted and featurized all mail messages. Then a preprocessing step applied a Zipf filter to each data set. In our case this filter removes common words using a stop word list and extremely rare words (those words that occurred only once in the mail store). Note that this filter does not have a significant effect on the results but reduces the processing time significantly.

We split each mail store data set into training and test sets. For each folder in a mail store, the most recent 20% of messages were selected for the test set. Note for some very sparse folders, no messages were selected for the test set.

Learning occurred for each of the classifiers on the training data. The linear SVMs trained on word-occurrence vectors for each message. Using the two-way classifier approach described in section 3.2, we trained a separate linear SVM for each folder, with the positive examples consisting of all messages belonging to that folder. The negative examples were a sample of messages from all other folders (in the case of User H and User D a 25% sample, a 100% sample for the other users). The IR and Unigram classifiers trained with the word-frequency vectors.

We assessed classification accuracy for each classifier using the test examples. The incremental classifiers, IR and Unigram, were permitted to learn after the presentation of each test message. The test messages were presented folder by folder in temporal order (not in overall time order).

The train/test split does not exactly correspond to classification problem in the dynamic email environment. A more realistic split would be to select the most recent 20% of all messages for the test set, instead of by folder. While more realistic, such a split penalizes the linear SVMs because they cannot adapt (without retraining) to new folders and all messages in such folders would be treated as automatic classification failures. Our split also guarantees that we will have sufficient data to evaluate the performance of the methods on various folder sizes.

4.2 Evaluation Criteria

All of the classifiers yield a ranked list of folders. Given this list, suppose the agent proposes the first M folders to the user. A natural evaluation criterion is the percentage of messages in the test for which one of the top M proposed folders is correct. While Segal and Kephart (1999) use this criterion for evaluating their IR classifier, most comparative studies of text classification methods have used other criterion related to precision and recall or 'best guess' accuracy (Yang & Liu, 1999).

One consequence of the top M criteria is to deemphasize the single 'best guess' accuracy, which is required for an automatic system. We report results on $M = 5$. Although this choice is somewhat arbitrary, our preliminary experiments suggested that differences in best guess accuracy between two classifiers faded for top 3 or top 5 accuracy. For example, there is a sharp difference between the single best guess accuracy on the User D mail store between the linear SVM and IR methods is 13% (linear SVM: 69%, IR: 56%). For top 3 accuracy the gap is 2% (linear SVM: 81%, IR: 79%) and for the choice $M = 5$ accuracy is statistically indistinguishable (linear SVM: 85.7%, IR: 85.1%).

4.3 Overall Accuracy

Figure 1 displays the accuracy results for all five mail stores, using complete messages (headers and body) as input. We draw several conclusions from these results:

- Accuracy varies more between mail stores than between classifiers.
- No classifier is consistently superior to the others.
- Classification accuracy is highest for the two mail stores with the lowest proportion of sparse folders.

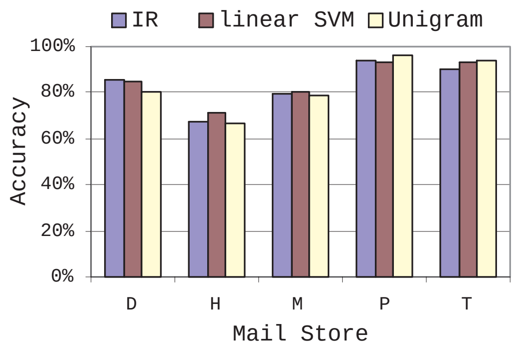

Figure 1. Top 5 Accuracy by Mail Store and Classifier: Entire Message

Table 2. Top 5 Accuracy by Folder Size: Entire Message.

Store	Class	1-5	6-10	11-19	20-49	50-99	100+
H	IR	75.8%	76.0%	78.0%	68.8%	65.1%	57.4%
H	Uni	28.4%	37.2%	51.4%	66.2%	82.1%	94.4%
H	SVM	39.3%	44.8%	65.4%	75.9%	82.7%	89.4%
M	IR	78.3%	74.0%	91.2%	77.1%	N/A	N/A
M	Uni	59.4%	62.0%	88.2%	89.6%	N/A	N/A
M	SVM	56.5%	64.0%	86.8%	94.4%	N/A	N/A
D	IR	88.8%	91.1%	89.8%	85.2%	78.9%	84.0%
D	Uni	45.6%	70.8%	75.3%	76.4%	91.7%	97.3%
D	SVM	56.8%	75.6%	81.8%	87.6%	90.1%	98.0%

4.4 Accuracy by Folder Size

Although overall accuracy for the three classifiers is similar, Table 2 indicates striking difference in the performance on sparse and dense folders. In Table 2, the top 5 accuracy is decomposed by the folder size of the correct classification of each test message. Users P and T are excluded due to an insufficient number of test messages or folders. Folder size is defined as the number of messages in the training set from the folder.

A number of clear patterns emerge in Table 2:

- The IR classifier offers the best performance for test messages from sparse folders. This comes at the cost of a significant reduction of accuracy for test messages from dense folders.
- The Unigram and linear SVM classifiers are very

Table 3. Absolute difference in Top 5 Accuracy (Headers only accuracy - Entire Message accuracy).

Store	IR	linear SVM	Unigram
D	-0.31%	-4.37%	1.11%
H	1.25%	-4.91%	-1.87%
M	5.44%	-4.53%	2.72%
P	2.31%	2.31%	-2.31%
T	0.64%	-4.46%	-2.55%

Table 4. Top 5 Accuracy by Folder Size: Headers Only.

Store	Class	1-5	6-10	11-19	20-49	50-99	100+
H	IR	65.3%	74.0%	78.0%	73.8%	65.4%	64.5%
H	Uni	63.8%	68.0%	70.1%	64.0%	58.5%	65.4%
H	SVM	48.0%	55.2%	60.8%	63.8%	63.7%	84.6%
M	IR	78.3%	82.0%	89.7%	88.2%	N/A	N/A
M	Uni	79.7%	70.0%	86.8%	81.9%	N/A	N/A
M	SVM	62.3%	62.0%	83.8%	83.3%	N/A	N/A
D	IR	78.1%	88.7%	90.6%	83.2%	84.7%	86.0%
D	Uni	76.9%	86.3%	84.0%	76.4%	81.4%	82.5%
D	SVM	64.5%	75.6%	79.3%	74.7%	83.1%	94.8%

accurate for test messages from dense folders, but are significantly worse than the IR method on test messages from sparse folder.

- To some degree, the accuracy of all three classifiers is a monotonic function of folder size. This holds strictly for the Unigram and linear SVM classifiers, and holds approximately for the IR classifier (accuracy of the IR on User M mail store is the exception).

4.5 Accuracy of Headers Only

Table 3 displays the absolute difference between the top 5 classification accuracy using the entire message and headers only. These results suggest that message bodies are largely superfluous for the email classification task. For the Unigram and linear SVM classifiers, disregarding the message body harms accuracy, although only slightly. The accuracy of IR improves slightly. This is likely due to the keyword mechanism of IR.

By restricting attention to only the headers one can significantly reduce the number of distinct features. For example, the number of unique features (post-Zipf filter) for the user with the largest mail store, User H, decreases from approximately 72,000 to 14,000 when the message bodies are discarded. The decrease is not as dramatic for other users but, for instance, the mail store for User P contains approximately 3,000 unique features restricted to headers only (post-Zipf filter).

Table 4 displays top 5 accuracy by folder size for headers only input. Many of the same trends noted in section 4.4 are evident, although mitigated to some extent. For example, the gap between the IR classifier and the other classifiers for sparse folders, while still significant, is less pronounced than in Table 2.

5. Discussion

5.1 Accuracy

The results in Figure 1 and Table 3 illustrate that incremental learning methods offer competitive accuracy as compared with the discriminative SVM method. This is somewhat surprising because previous research has shown SVMs to be exceptionally good at text classification (Dumais et al., 1998). One might expect discriminant methods to outperform the other methods in the email domain given the heterogeneity of folder content. In particular, the TF-IDF models and unigram models are essentially centroid based methods and in our SVM method each of the messages is preserved individually and used in the training. These performance results may be due, in part, to the top 5 evaluation criteria, but we believe this is a better metric for interactive email classification than single 'best guess' accuracy. Of course, it might be possible to improve the performance the SVM or other discriminant classification technology through incremental adaptation, alternative feature selection methods, or alternative featurization methods.

5.2 Sparse Folders

Given the data in Table 2, a natural question is whether any classifier can offer consistent accuracy across all folders sizes. We speculate that the performance of classifiers will vary on test messages from sparse and dense folders. This speculation is supported by experimental evidence of Yang on an alternative performance criterion (Yang & Liu, 1999). Our speculation is based on the fact that the ranking of folders creates a competition between dense and sparse folders. If the mechanics of a classifier upweight the scores of sparse folders, then sparse folders will move towards the top of the list. For example, the IR looks for a few keywords that occur in a small number of folders. The weight of a keyword in the score is the ratio of the within-folder rate to the global rate. In a sparse folder, this ratio is likely to be large, simply because there are fewer words in the folder. A similar argument holds for classifiers that upweight the scores of dense folders. The Unigram classifier, for example, works by identifying the number of matching words. It is more significant that a word occurs in a folder, and less significant that it occurs a large number of times. Each word that does not match adds a heavy penalty to the posterior probability. Dense folders have many more words, and are therefore likely to generate a higher number of matches.

The speculation that it will be difficult to find a classifier that performs consistently across all folder sizes does not preclude significant improvement over the results presented in this paper. We experimented with tuning the C parameter of linear SVMs in an attempt to improve performance on small folders. By tuning the C parameter of the SVMs we were able to improve the performance on smaller folders without significantly degrading the performance on larger folders. We also experimented with variants of the Unigram classifier. Our modifications of the unigram actually perform better in all folder size categories than the Unigram classifier when the input consists of entire messages but not for headers alone . These extensions will be described in a longer version of this paper.

5.3 Computational Costs

Both of the incremental learning methods require maintenance of a feature index. This index translates into a larger in-memory footprint for the smart agent. In addition, the size of the index is not fixed, but grows as new features are introduced. The memory footprint for Linear SVMs is fixed because weights are only stored for a fixed number of features. However, this benefit comes at a cost. Primarily due to feature selection and featurization, initial training for linear SVMs is CPU-intensive. Furthermore, the CPU cost and memory cost must be paid periodically to keep the classifier up to date. These costs make SVMs less amenable to incremental adaptation. Of course the initial training for the IR and Unigram classifiers can be significant; it consists of compiling a feature index, featurizing each mail message and collecting statistics. While this is a considerable cost, the ability to do incremental adaptation means that it is a one-time cost.

The computational cost for performing classification is minimal for all three methods. The score (or probability) for each folder involves a sum (the Unigram operates with log probabilities) that is linear in number of features in the incoming message.

Of course, both the memory and CPU profile are improved by restricting input to headers only. Furthermore, our experiments suggest the loss of accuracy versus featurizing the entire message might be acceptable with respect to the top 5 evaluation criteria. In particular, as noted in Section 3.1, restricting input to messages headers reduces the cost of featurization. Another consequence is significant reduction of the feature index, and hence the memory footprint of incremental learning methods. Using only the headers can also reduce the cost of feature selection and projection for a discriminative method such as our SVM. In fact, because linear SVMs can be fit efficiently when there are several thousand features it may eliminate the need for the feature selection altogether. Other memory issues related to training SVMs might be addressed by sampling the training data. While it may be possible to address the computational costs of applying discriminative methods such as SVMs to the interactive email classification problem, our results demonstrate that one can resort to computationally tractable incremental classifiers without sacrificing accuracy.

References

Boone, G. (1998). Concept features in re:agent, an intelligent email agent. In *Proceedings of the Second International Conference on Autonomous Agents*, pp. 141–148.

Cohen, W. W. (1996). Learning rules that classify e-mail. In *Proceedings of the American Association of Artificial Intelligence Spring Symposium on Machine Learning and Information Access.*

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pp. 148–155.

Goodman, J., & Chen, S. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings Thirty-fourth Annual Meeting of the Association of Computational Linquistics*, pp. 310–318.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 170–178.

Platt, J. C. (1998). Msr-tr-98-14: Sequential minimal optimization: A fast algorithm for training SVMs. Tech. rep., Microsoft Research, Redmond, WA.

Segal, R. B., & Kephart, J. O. (1999). Mailcat: an intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomous Agents*, pp. 276–282.

Whittaker, S., & Sidner, C. (1996). Email overload: exploring personal information management of email. In *Conference Proceedings on Human Factors in Computing Systems*, pp. 276–283.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the Special Interest Group on Information Retrieval*, pp. 42–49.