

EmailValet: Learning Email Preferences for Wireless Platforms

Sofus A. Macskassy, Haym Hirsh, Aynur A. Dayanik

Department of Computer Science
Rutgers University
110 Frelinghuysen Rd
Piscataway, NJ 08854-8019
{sofmac,hirsh,aynur}@cs.rutgers.edu

June 3, 1999

1 Introduction

Imagine that while commuting home you receive an email message on your pager from your spouse asking you to pick up some milk on your way home. Compare this with commuting home and receiving an email message on your pager from a coworker with a monthly reminder to use the recycle bins at work. One of these messages would likely be relevant to you at that particular point in time, while the other would not be. It would be nice if it was possible to automatically decide which messages to send to the pager in such a way that the first message would be forwarded and the second one would not. Furthermore, it would be nice if such forwarding could take place in a transparent fashion for the people sending and receiving the email. For example, it would be desirable to have a single, centralized email address to which email would be sent, and some software agent residing at that location could then forward each message to where the receiver would like to read it, eliminating the need for both receiver or sender to figure out where to read a message from or send a message to. Ideally, this agent would work in a way much akin to a secretary, knowing where the user is and knowing which messages to forward to where. Thus, for example, if the user had a palmtop device with a wireless data service, a pager, a computer at home, and a desktop at work, the agent would need to decide where a message should be delivered, based on a number of factors, such as what the message is about, and when and where the user would want to read this message. Obviously different people would have different preferences even in identical circumstances, so the agent must tailor itself the user's email-reading preferences. We call such an agent an EmailValet, in that it provides "valet"-like functionality to help users manage email.

2 Current Work

This section describes our ongoing work on an EmailValet for two-way pagers on the Bell South Wireless Data Interactive Paging Service that can receive as well as send alpha-numeric messages composed on a "qwerty" keyboard. Using these pagers, and a front-end agent residing at the Rutgers mail server, we have been able to record three months of usage with a total of over 8900 messages. This front-end agent, is a script that forwards all mail sent to the Rutgers account to the pager, rewrapping the email message in such a way that the user of the pager sees it as coming from the original sender, while any reply will go back to Rutgers. The agent initially sends only the headers and the user can decide what action to take. When the user replies to the message, the agent receives it and take the action specified by the user. The actions supported include getting the complete text of the message and replying or forwarding the message. When the user specifies the "reply" command, the agent rewraps the message and forwards it on to the real recipient in such a way that the recipient sees it as coming from Rutgers. In this way, it is totally transparent to the outside world whether the user is replying through the pager or through some other mechanism (e.g., reading and replying from a desktop at work). All interaction goes through the agent, which logs all interactions and messages. By setting up this agent, the user gets the standard mail-reader functionality of getting all headers and choosing which message to read and in which order. This setup also allows us to log all user actions with respect to his or her email behavior, providing data for experiments on suitable methods for learning users' email management preferences. At present we

are focusing on the task of learning which messages the user asked to be sent to the pager, so that the system would be able to take over and only forward messages that the user wants to read on the pager.

2.1 Initial Experiments

We have done some initial experiments with the three months worth of data gathered thus far. From the user logs we are able to divide a user's email data into two sets of messages: those for which the user requested the full text, and those not. The learning task is then to learn to predict to which set a new message belongs. The experiments we report here are performed "off-line", evaluating five different learners on this binary classification task.

2.2 Learners

The experiments made use of five different learners on this task: Naive Bayes [10], TFIDF, Probabilistic TFIDF [7], Ripper [4, 5] and Winnow [8, 2]. The first three were made available through a public system called Rainbow [9], and the version of Ripper is available from www.research.att.com/~wcohen. We used our own implementation of Winnow.

The **Naive Bayes** classifier uses Bayes' rule to estimate the probability of each category for a given document, based on the prior probability of a category occurring, and the conditional probabilities of particular words occurring in a document given that it belongs to a category, assuming that these probabilities are conditionally independent. Joachims [7] and Mitchell [10] give further details on the use this algorithm for text categorization.

The **TFIDF** ("Term Frequency times Inverse Document Frequency") classifier [7] is based on the relevance feedback algorithm by Rocchio [11] using vector space retrieval model [12]. This algorithm represents documents as vectors so that documents with similar content have similar vectors.

The **Probabilistic TFIDF** classifier [7] is a probabilistic version of the TFIDF classifier, based on estimation of the probability of a category C_j given document d , $Pr(C_j|d)$, using the retrieval with probabilistic indexing method proposed in [6].

Ripper [4, 5] is a learning method that forms sets of simple rules for data described by sets of attribute-value pairs. Each rule tests a conjunction of conditions on attribute values. Rules are returned as an ordered list, and the first successful rule provides the prediction for the class label of a new example. Importantly, Ripper allows attributes that take on sets as values, in addition to numeric and nominal features, and a condition can test whether a particular item is part of the value that the attribute takes on for a given example.

Winnow¹ [8, 2] learns linear threshold functions using a multiplicative weight-updating scheme. It works with a set of experts, each of which can either make some prediction about the class of an example or abstain from predicting. Each prediction is given some weight, or vote, and the class that gets the highest overall vote is returned as the final class. The goal of learning is to find a suitable weighting scheme for each expert. Finally, Winnow is an incremental learning method, updating the weights after each example, and given a fixed collection of examples, our implementation makes only one pass through the data, running through the examples in order and updating weights after each incorrect prediction.

2.3 Data

We used the data gathered from the three people involved in the project, giving us three distinct data sets of varying sizes. Table 1 shows the amount of data obtained for each user, and the proportion of email that each user requested forwarded.

Since the data is chronological in nature, it is not appropriate to divide it into random test and training sets. Instead, the data was split into two contiguous segments; the first two-thirds were used as a training set and the last one-third was used for testing. The splits were done *after* removing the first 5% of the data, since this initial data might be more noisy due to the user learning how to use the system.

Each learner had different requirements on the data-format, however in general we were able to split the data into two main parts: headers and the body. The body was just a list of all the words (after some pruning and transformation) that occurred in the body, and the headers included the date of the message (formatted differently for each learner), the length of the message, a tokenization of the fields: from, to, a union of to and cc, and the subject.

¹Strictly speaking, we are making use of the basic "Winnow II" strategy [8]. Since, over time, the number of specialists can increase unboundedly, we are using the *infinite attribute model* by Blum [1, 3].

Table 1: Dataset properties

User	Size	% of messages forwarded in		
		full set	first $\frac{2}{3}$	last $\frac{1}{3}$
AD	1730	13.70%	11.43%	18.22%
HH	5651	17.02%	16.67%	17.72%
SM	1526	10.41%	10.69%	9.86%

2.4 Evaluation

The evaluation used in this paper is the break-even point of a method. This method attempts to find a point where precision and recall are roughly equal. The precision of a classification method is the proportion of data that is labeled positive that really is positive, and the recall of a classification method is the proportion of all truly positive data that is labeled as positive. By varying the parameters to some classifiers, it is possible to get different pairs of these values, giving a curve of trade-offs between the two. The break-even point attempts to find the point on this curve where the two values are roughly equal.

2.5 Results

The results we have obtained show promise. With the exception of one method on one subject, all methods are able to get a break-even point above 30%, over half of them exceed 40% and one method was as high as 53% for one subject. Table 2 shows the break-even points for each method, with the best score for each user being highlighted.

Table 2: Break-even points for all methods

User	Naive Bayes	TFIDF	Pr-TFIDF	Ripper	Winnow
AD	41.71	36.02	39.81	36.5	43.36
HH	46.94	30.02	46.34	53.42	47.15
SM	45.55	39.61	47.53	37.56	27.28

On the other hand, no one method clearly outperformed its competitors, leaving us without a clearly best learning method for this task as yet. However, we were pleased to note that most methods do better than the baseline that labels all email to be forwarded to the user (which would give 100% recall with low precision), getting 100% recall with better precision than this naive baseline. Also, for all subjects, at least one method was able to reach 100% precision, with low recall (in other words, it labels as forward a small number of members all of which the user wanted forwarded).

3 Prospects for the Future

This paper has described the use of machine learning and information retrieval methods to predict whether to forward email to a user’s wireless email device. Our initial results, based on actual use of our EmailValet system, show substantial improvements over the default of sending all messages to a user.

In the short term, we would like to understand whether other features already discernible by the EmailValet system, can further improve performance. In particular, there are many features about the context of an email message, such as when you last read a message from its sender, whether it is part of an ongoing email discussion, or whether you recently sent email to this person, that may be helpful in deciding how to handle a message. We would also like to be able to order pending email messages effectively, so that if the person has not read email for a while, when the messages do appear they are presented in a prioritized fashion.

On a longer time frame, we would like to explore the use of EmailValet across a range of wired and wireless platforms and with a wider range of information sources. For example, forwarding a message should depend on what devices are available to a user at a point in time – if the user is in front of a workstation email probably need not be

sent to a pager. Ideally, we would like an EmailValet to have access to information such as a user's calendar or physical location (as should become possible in the future via GPS systems on wireless devices). This could make it possible for the EmailValet to know to forward a message from the person to whom you are walking across campus for a scheduled meeting if the message might be canceling the meeting. We would also like to expand the range of interaction modalities that a user can have with the EmailValet — eye tracking (which is becoming possible for desktop workstations) would make it possible to identify the portions of a message that a user had read, and voice interaction also provides the opportunity for finer-grained interaction between the user and EmailValet. Similarly, permitting a wider range of interaction between the user and EmailValet — such as via natural-language dialog — could provide a valuable resource to an EmailValet, such as making it possible to ask the user questions about his or her decisions.

Finally, although the EmailValet concerns email, the very same architecture would allow a user to interact with an agent that can access the Web or other sources of information from a range of platforms. In this light, the EmailValet represents a very first step in our overall vision of software “valets” that more generally serve our heterogeneous information needs across a range of interaction devices.

3.1 Acknowledgments

We thank our many colleagues in the Rutgers Machine Learning Research Group and at WINLAB for their feedback on this work. Equipment for this work was provided by BellSouth Wireless Data.

References

- [1] A. Blum. Learning boolean function in an infinite attribute space. *Machine Learning*, 3:373–386, 1992.
- [2] A. Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 1997.
- [3] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 157–166, Santa Cruz, California, 1991. Morgan Kaufman.
- [4] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California, 1995.
- [5] W. W. Cohen. Learning trees and rules with set-valued features. In *AAAI96*, 1996.
- [6] N. Fuhr. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.
- [7] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.
- [8] N. Littlestone. Learning quickly when irrelevant features abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [9] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996.
- [10] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [11] J. Rocchio. Relevance feedback in information retrieval. In Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, 1971.
- [12] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.