EmailValet: Where do you want to read your Email?

Sofus A. Macskassy  
Aynur A. Dayanik  
Haym Hirsh  
Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854-8019

Abstract

This paper describes EmailValet, a system that learns users’ email-reading preferences on email-capable wireless platforms – specifically, on two-way pagers with small “qwerty” keyboards and an 8-line 30-character display. The paper more specifically presents results comparing the ability of different learning methods to form models that can predict whether a given message should be forwarded to the user’s wireless device, using the authors’ email-reading preferences for about nine months and more than 15,000 email messages. Our results show that we are able to achieve a break-even point of over 63% for the user with the greatest amount of email, over 11,000 messages. We also find that, in general, all methods are able to achieve better performance than what would be achieved by a baseline of simply forwarding all messages to the wireless device, and that many methods are able to find procedures that, although they forward only a small fraction of the messages that a user would want, achieve 100% precision on those messages that it does actually choose to forward. We also evaluate the extent to which training on email from varying windows of time into the past impacts learning, as well as the effect of user and email contextual features on learning.

1. Problem Description

Imagine that while on your way to a meeting you receive an email message on your pager letting you know that the meeting has been moved to another building. Compare this with visiting another university and receiving an email message on your pager notifying you that a seminar is about to begin in a few minutes at your own university. One of these messages would likely be important to you at that particular point in time, while the other would not be. The goal of the EmailValet is to learn, based on various types of information, to forward messages of the first type to the pager, but not message of the second type. Furthermore, we would like such forwarding to 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, who is sending the message, current user context and what user devices are currently available.

We call such an agent that provides “valet”-like functionality an Information Valet (iValet) (Macskassy, Dayanik, & Hirsh, 2000). The EmailValet is an instantiation of such an iValet, focused on providing valet-like functionality to users reading email (Macskassy, Dayanik, & Hirsh, 1999). In this paper we briefly describe the iValet framework, continuing with a discussion of the EmailValet and how it fits into this framework. The main part of the paper discusses the experimental design for our study and the main results found so far, in which we compare the performance of a range of well established-machine learning methods, including Naive Bayes and TFIDF.

2. Information Valets

Our model of an Information Valet (iValet) is an agent that helps users intelligently access various forms of information via a range of devices (Macskassy et al., 2000). There are a wide ranges of design decisions that are necessary in building an iValet, such as the nature of the information, the form of user feedback, etc. We divide the range of design decisions into four classes, those concerning the user, the user’s access devices, the information being accessed, and
2.1 User

We characterize design decisions concerning the user into three classes:

The user context refers to the information that is known about the user in making information-handling decisions. This includes such variables as the user location – where the user is currently; the user’s state – is the user currently busy, on vacation, in a meeting, etc.; and user availability – is the user currently available to the iValet through any known means (to the iValet), such as whether a given device is turned on. All of these issues impact on the design and ultimate functionality of an iValet.

The user model specifies what information is saved about the user and the user’s overall environment, and the form in which it is maintained. Design decisions concerning the user model include whether the model is based primarily on content-based or collaborative information (Lashkari, Metral, & Maes, 1994; Shardanand & Maes, 1995; Billsus & Pazzani, 1999; Basu, Hirsh, & Cohen, 1998), as well as the structure of the model, such as whether it maintains short-term or long-term interests or a hybrid thereof, as well as what information the user is known to already have viewed (Billsus & Pazzani, 1999).

Ultimately the iValet adapts to a user’s behavior based on user feedback. The amount of available feedback can vary tremendously. At its most coarse level, feedback is either implicit, by “watching over the shoulder” of the user to note, for example, simply which messages a user has read, or explicit, by obtaining additional feedback from the user, such as by allowing the user to critique the reasons for receiving an information item.

2.2 Device

The devices used by an iValet have specific information that tell the iValet what information can be viewed and how it can be presented. This information is split into three types of information: device connectivity, device capabilities, and device context.

Device connectivity refers to the method by which the device receives the user’s information, thereby also defining in part the modality of the user interaction. These features include whether it is always connected or if it is connected intermittently, if it is push-capable – you can send it information directly without waiting for the device to request it, if it has an explicit email-address, what its bandwidth is, and whether it is wired or wireless.

Device capabilities refer to finer-grained details of a device’s available resources. This includes such factos as the size of the largest item it can receive, whether it is text I/O capable, audio I/O capable, if it can present graphics, if it has local storage (and how much), the processor speed, and whether it has any external ports (such as infrared).

Device context refers to the current state and context of the device in its actual use at a point in time. This includes its battery strength, if it is currently connected, how much it costs per byte to send it information, the amount of remaining local storage, and if the device is currently in use by the user.

2.3 Information

Often, the reception modality of a piece of information on a device does not match up with the transmissions modality of the information. In our general iValet framework there may be transformation processes that convert a piece of information from one modality to another. Issues concerning the information being managed by an iValet include the modality of the information, its size, and other medium-specific information, such as the authorship of the information.

2.4 Learning

Ultimately, if the iValet is to learn from the user’s behavior, it must have some underlying learning algorithms that it employs. Although some of the constraints on the decision of a suitable learning method are imposed by the other design decisions (such as by the available user feedback modalities), there are further design decisions based on the desired functionality of the iValet. This includes issues such as whether the learner must be incremental and run in real time and whether its final output is a score on each information item, an unscored ranking of items, or ultimately simply a yes/no decision about the importance of each information item.

3. EmailValet

The paper focuses on one field example of an iValet, called the EmailValet, designed to operate for email read via wireless platforms. It is built on two-way paging devices having unique email addresses available from BellSouth Wireless Data Services. The devices used by all three authors were RIM Inter@ctive Pagers whose functionality through the BellSouth Wireless services includes sending and receiving text messages both through BellSouth’s proprietary interactive paging5M network, as well as to and from arbitrary internet email addresses.

The EmailValet is an agent that resides at the Rutgers mailserver and filters all of the recipient’s email transparently to both the message’s sender and receiver. The headers of all received messages are then passed on to the user’s...
pager, with the headers modified so that any reply from the device goes back to the agent (i.e., to the user’s own email address) rather than directly to the original message sender. Upon receipt of the message headers at the pager, the user can then reply, requesting (among other options) the full text of the message from the EmailValet.

The EmailValet keeps track of a narrow user context. In addition to the user’s wireless platforms, each user also has a workstation at Rutgers and a computer at home. Under the assumption that it is relevant whether the user is online via these wired platforms, the EmailValet tracks whether the user is online; whether the user has been idle, and if so, how long; and the time when the user last read email online. It does not distinguish between long–term and short–term user interests, but has a shallow content-based user model. The only feedback currently used is implicit feedback from the user, when the user requests the full body of an email from which the EmailValet infers that the item was read and should act as a positive example of an email message to be sent to the pager. The EmailValet then tries to learn which messages to forward to the pager.

The only things known about the pager device by the EmailValet is that it is email-capable, it is virtually always on (though the user can turn off the pager if so desired), it has a max item size of 16Kb and it is capable of text I/O only. The EmailValet has an implicit assumption that the device is always available to the user. Other facts about the RIM pagers include a storage capability of 1Mb, it has a small ‘qwerty’ keyboard and a display of 8 lines of 30 characters.

The information items used by the EmailValet consists of email messages only. The items consists of an email, which includes the headers and the body, as well as contextual information about the email, including when the user last received (or sent) email from (to) the sender; as well as the last time the user last received (or sent) email from (to) the sender on that particular subject. The learning method assigns a score to each message to be used to rate the importance of each message.

4. Experimental Methodology

We carried out experiments with the nine months worth of data gathered from the authors totalling over 15,000 messages. From the user logs, two sets of email messages were created for each user, one containing those messages that the user chose to see on the wireless platform, and a second that the user did not choose to see on the wireless platform. The experiments we report here were performed “offline” on this binary data. Each learning method assigned a score to each message, and we evaluate the methods via precision-recall curves (discussed further in Section 4.3).

4.1 Learning Algorithms

We used four different learning methods often used for text categorization: Maximum Entropy (ME) (Nigam, Lafferty, & McCallum, 1999), Naive Bayes (Mitchell, 1997), TFIDF, and Probabilistic TFIDF (Joachims, 1997). We used publicly available versions of all of these systems, available from the Rainbow package (McCallum, 1996).

In early experiments reported elsewhere (Macskassy et al., 1999) we explored the use of two other learning methods. Although Ripper (Cohen, 1995) performed comparably to the other methods, it was much slower and thus was not a realistic choice for this task. In contrast, our results with Winnow (Littlestone, 1988) found it to consistently perform more poorly than all other learning methods. For this reason, neither method is used or reported in this paper.

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 (Joachims, 1997) and Mitchell (Mitchell, 1997) give further details on the use this algorithm for text categorization.

The TFIDF (“Term Frequency times Inverse Document Frequency”) classifier (Joachims, 1997) is based on the relevance feedback algorithm by Rocchio (Rocchio, 1971) using vector space retrieval model. This algorithm represents documents as vectors so that documents with similar content have similar vectors. Each component of such a vector corresponds to a term in the document, typically a word. The weight of each component is computed using the TFIDF weighting scheme, which tries to reward words that occur many times but in few documents. In the learning phase, a prototype vector is formed for each class from the positive and negative examples of that class. To classify a new document, the cosines of the prototype vectors with the corresponding document vector are calculated for each class. d is assigned to the class with which its document vector has the highest cosine.

The Probabilistic TFIDF classifier (Joachims, 1997) is a probabilistic version of the TFIDF classifier, based on estimation of the probability of a category C given document d, Pr(C|d), using the retrieval with probabilistic indexing method proposed in (Fuhr, 1989). To classify a new document d, Pr(Cj|d) is estimated for each class, Cj, (as described in more detail by Joachims (Joachims, 1997)). d is assigned to the class whose probability is the highest.

The Maximum Entropy (ME) classifier for text classification estimates the conditional distribution of the class label given a document, which is viewed a set of word-count features (Nigam et al., 1999). The basic idea of this technique
Table 1. Dataset properties

<table>
<thead>
<tr>
<th>User</th>
<th>Size</th>
<th>% of messages forwarded</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>2.298</td>
<td>27.08%</td>
</tr>
<tr>
<td>HH</td>
<td>11.034</td>
<td>26.86%</td>
</tr>
<tr>
<td>SM</td>
<td>1.675</td>
<td>16.78%</td>
</tr>
</tbody>
</table>

is that uniform distributions should be preferred in the absence of external knowledge. A set of constraints for the model are derived from the labeled training data, which are expected values of the features. These constraints characterize the class-specific expectations for the model distribution and may lead to minimal non-uniform distributions. The solution to the maximum entropy formulation is found by the improved iterative scaling algorithm.

4.2 Data

Table 1 shows the amount of data obtained for each user, and the proportion of email that each user requested forwarded.

As the data is of chronological nature, we do not use cross-validation. Instead, we make a prediction on each test message based solely on those preceding it. Due to the large size of our dataset, for each data set, 500 random data points were chosen. For each method, we then used all preceding instances in the data set as training to score the given test-instance.

We converted each word in every message field into all lower case, with all punctuations, numerics, and special characters removed. Stemming (performing a morphological analysis of each word and only maintaining its root word) and stop-list removal (ignoring words with little information content, such as “the”) were also performed. Email addresses were tokenized to create multiple tokens. Thus, for example, “Sofus Macskassy sofmac@cs.rutgers.edu” became the tokens “Sofus Macskassy sofmac@cs.rutgers.edu sofmac@rutgers.edu sofmac cs.rutgers.edu rutgers.edu”.

Rainbow requires that a message be represented as a bag of words. We split the data representation into two main parts: headers and body. The header features included the date, length and subject of the message, and a tokenization of the from, to, and cc fields. All header tokens were prepended with the name of the header to distinguish it from the same word that might appear in the body. All numeric tokens (such as length and date) were represented as text-valued items by defining bins, with tokens assigned to a value based on the bins in which it fell. For length, there were bins for whether a message was smaller than 0.5, 1, 2, 4, 8 and 16 kilobytes, and a similar set for whether the length was larger than each of these values. Thus, for example, a value of 2380 bytes would be represented with the tokens “MoreHalfKB MoreOneKB MoreTwoKB LessFourKB LessEightKB LessSixteenKB”.

We used a vocabulary size of 1000 words for all methods.

4.3 Evaluation

The aim of EmailValet is to forward the relevant messages to the user’s wireless device without forwarding the irrelevant ones. Therefore, for evaluation purposes, we compute precision/recall curves for each method. The precision of a classification method is the proportion of data that is labeled positive that really is positive. In our case, the precision corresponds to the proportion of messages that were labeled as forward that really should have been forwarded. The recall of a classification method is the proportion of all truly positive data that are labeled as positive, and the recall corresponds to the proportion of messages that should have been forwarded that were labeled forward. Together these two values reflect a trade-off between the coverage of a classifier and its accuracy on the class that is of interest. Since for different people different points on this trade-off may be more or less desirable, we use precision/recall curves to better reflect the range of possibilities that a method provides.

We also consider a second evaluation method, the break-even point of a method. This method attempts to find a point where precision and recall are roughly equal. Although it focuses on only one point on the precision/recall curve, it yields a way to obtain single-value evaluations of a method, simplifying the process of making direct comparisons between methods.

Precision/recall curves are created for the Rainbow methods by sorting instances based on the score assigned by the method for the forward class for that instance. Setting a threshold on this score allows varying the number of examples predicted as forward. This threshold can be varied so as to include examples in the list one-by-one, yielding a precision/recall point for each threshold. Thus, for example, at one extreme all messages are forwarded, yielding a baseline of 100% recall, but typically low precision. The various precision/recall points can then be pruned, where a point is deleted if there is another observed point with better precision and recall. The resulting points can then be plotted as a

---

1 Further details on our binning approach are included in a separate paper (Dayanik, Macskassy, & Hirsh, 2000), which discusses its use on both text and non-text classification problems.

2 We tested the methods on various vocabulary sizes, finding little variance in their performance, leading us to fix on this size.

3 If the scores were equal, we drop down to the next score that is different, since we can obviously not put a threshold to distinguish between instances that have the same score.
5. Results

Table 2 shows the precision/recall break-even points of the four learning methods for each dataset when training takes place on the full set of email that chronologically preceded a given selected test message. There are three immediate observations we can make regarding these break-even points. First, we are able to achieve as high a break-even point as 63% on the HH data-set, which was more accurate for the information available to us than we initially expected. Second, Maximum Entropy seems to consistently outperform the other methods by quite a non-negligible margin. Finally, Naive Bayes also performs plausibly, and may be a desirable method due to its speed and the fact that it lends itself well to providing users with explanations of decisions (by extracting the top \( k \) words that supported the decision (Bill-sus & Pazzani, 1999)).

While the break-even points give one absolute evaluation metric, it does not give a complete picture. To provide a better sense for the overall performance of the methods, we graphed the precision/recall curves for each data-set as shown in Figures 1, 2 and 3.

The precision/recall graphs allow certain conclusions to be made about the use of these four methods. First, a very simple baseline approach is to forward all email to a user. This method yields 100% recall, but a precision that equals the proportion of messages that should be forwarded in the data (see Table 1). This baseline is shown in the three graphs as a vertical line. In all cases, except one (Naive Bayes on the HH data-set), this baseline is surpassed, demonstrating that even the less-successful methods still achieve some modicum of success on this task.

The graphs show that for each user there are learning methods that can enable the EmailValet to forward a small number of messages all of which should have been forwarded, that is, for cases when 100% precision is desired, with small, but non-zero recall. TFIDF was able to do so across all three datasets, with both Maximum Entropy and Pr-TFIDF achieving 100% precision on two of the three datasets. Naive Bayes, on the other hand, was not able to achieve 100% precision on any data-set.

Lastly, it is also clear that while the Maximum Entropy method does achieve the best break-even point, only on the HH data-set is it the clear winner running the whole curve. On the SM data-set it performs suboptimally at the high end.

Table 2. Comparison of break-even points on all methods

<table>
<thead>
<tr>
<th>Data-Set</th>
<th>Maximum Entropy</th>
<th>Naive Bay</th>
<th>TFIDF</th>
<th>Pr-TFIDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>60.725</td>
<td>55.358</td>
<td>54.800</td>
<td>54.265</td>
</tr>
<tr>
<td>HH</td>
<td>63.130</td>
<td>54.340</td>
<td>1</td>
<td>54.620</td>
</tr>
<tr>
<td>SM</td>
<td>48.320</td>
<td>45.230</td>
<td>1</td>
<td>47.963</td>
</tr>
</tbody>
</table>

1. Did not cross the x=y point; last values were precision of 50.00 and recall of 58.68
Table 3. Break-even points on all methods, all window sizes

<table>
<thead>
<tr>
<th>Data-Set</th>
<th>Window Size</th>
<th>Maximum Entropy</th>
<th>Naive Bayes</th>
<th>TFIDF</th>
<th>Pr-TFIDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>100</td>
<td>55.133</td>
<td>55.258</td>
<td>54.988</td>
<td>51.105</td>
</tr>
<tr>
<td>AD</td>
<td>500</td>
<td>58.565</td>
<td>58.263</td>
<td>56.493</td>
<td>49.833</td>
</tr>
<tr>
<td>AD</td>
<td>inf</td>
<td>60.725</td>
<td>55.358</td>
<td>54.800</td>
<td>54.265</td>
</tr>
<tr>
<td>HH</td>
<td>100</td>
<td>56.138</td>
<td>46.958</td>
<td>54.558</td>
<td>48.470</td>
</tr>
<tr>
<td>HH</td>
<td>500</td>
<td>60.890</td>
<td>51.218</td>
<td>56.868</td>
<td>48.803</td>
</tr>
<tr>
<td>HH</td>
<td>inf</td>
<td>63.130</td>
<td>54.340³</td>
<td>46.163</td>
<td></td>
</tr>
<tr>
<td>SM</td>
<td>100</td>
<td>38.570</td>
<td>35.760</td>
<td>36.563</td>
<td>35.553</td>
</tr>
<tr>
<td>SM</td>
<td>500</td>
<td>42.440</td>
<td>40.005²</td>
<td>41.485</td>
<td>36.450</td>
</tr>
<tr>
<td>SM</td>
<td>inf</td>
<td>48.320</td>
<td>45.230</td>
<td>44.743</td>
<td>37.963</td>
</tr>
</tbody>
</table>

1 Did not cross the x=y point; last values were precision of 50.00 and recall of 58.68
2 Did not cross the x=y point; last values were precision of 39.56 and recall of 40.45

of recall, while being superior to the other methods in the middle. This same is seen to a lesser extent in the AD data-set.

We also note one subtle point of these experiments, in that the learning method has access to the full body of all messages, whereas the user is making decisions based only on the headers. We justify this choice by the fact that the email recipient typically has a tremendous amount of background knowledge that can be brought to bear in interpreting the headers of a message, whereas the EmailValet does not. The hope was that the body, since it is accessible at the server, where the EmailValet reside, can make up for some of this absent knowledge.

A second issue we explored experimentally concerns the use of the entire history of email that preceded each test example. By constraining learning to a smaller window of preceding messages we can get a sense for how quickly learning can "get up to speed", as well as whether an "infinite" window-size was always the best. If the EmailValet is meant to be adaptive and learn about a user — it would be helpful if the user could get a somewhat quick return and see that the system actually shows that it is learning the user’s interests within an acceptable timeframe. To explore this question we performed a second set of tests, using three different window sizes: 100, 500 and infinite. We then compared the performance, for each method, on these three window sizes. The break-even points for these runs are shown in Table 3.

The results allow certain observations to be made. First, except for one case, Maximum Entropy still outperforms all other methods and the only time it loses, it is by a very small margin. Second, out of the twelve window-size comparisons, in eleven cases the infinite window outperforms the window size of 100, showing that an infinite window is clearly better than a too-small window. Interestingly, in only seven of the twelve comparisons does the infinite window outperform the window size of 500, leading us to believe that window size does play a role, giving cause for further study. We also note that in eleven cases the 500 window size outperformed the 100 window size. Lastly, the Maximum Entropy method always had a larger window size win over a smaller window size.

Again, to get a better handle on these results, we turned to the precision-recall graphs. For lack of space, only a subset of the twelve graphs are shown in this paper. In Figure 4, we see a clear depiction of how the larger window sizes outperform the window size of a 100 and how the window size of 500 seem to "catch up" with the infinite window size. In Figures 5 and 6, we show some more prototypical cases, where the three window sizes are very close, where more than one of them actually is optimal for a small part of the precision/recall curve, having the curves cross each other and having no best-case or worst-case method.

The last issue we explored concerns the use of date and time information in the learning process. We would ideally like the learning algorithm to know where a user is — at home, work, in a car, etc. However, this is not available to the EmailValet, but we conjectured that date and time information may be a helpful surrogate for such informa-
tion about the message’s context, as opposed to its content. For example, a person is typically at work certain times of day and days of the week. In order to gauge the effect of contextual information, we performed a third set of experiments to compare the performance of the methods without email-contextual features (how long ago the user last received/sent a mail from/to the sender [on that particular subject]) and without user-contextual features (if the user is logged in, how long the user has been idle, and how long ago the user last read email online). If the methods perform worse without those features, then obviously the features are important for the overall performance of the learners. We graphed out the twelve different runs, for each method on each data-set. Unfortunately, the results were very ambiguous. On one data-set, the email context was important on three of the four methods, while for the other sets, there was no discernible difference, having the curves criss-cross many times. Figure 7 shows the runs on the AD data-set, where the removal of the email context clearly hurts the performance, while Figures 8 and 9 show the more prototypical runs where there is no clear distinction between the three types of runs.

6. 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 work is currently exploring many of the issues raised in this paper, such as appropriate role of contextual fields in learning and suitable ways to trade-off window size and accuracy. Of particular interest is the use of more “knowledge-based” features, such as simply counting how many question marks are in a message’s body or whether the words “today” or “tomorrow” appear in the message.

Our ongoing efforts also concern the EmailValet as a prototype of an Information Valet. We are currently looking to explore the use of the 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 for-
ward 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. For example, some devices are asynchronous in that the user has to log in or is only intermittently connected through it, while other devices like pagers are always on. This has an impact, for example, on whether a binary decision must be made as to whether to disturb a user by sending a message, or whether all message headers are presented in some order or with associated scores (as is the case with the current EmailValet) and thresholds that signify whether the message is important enough to forward to the device. Looking even farther ahead, we are considering other modalities for obtaining user feedback. For example, eye-tracking 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. Finally, although the EmailValet concerns email, its architecture was designed to be applicable to multiple devices and multiple information domains, such as the World Wide Web. Ultimately, we target something truly akin to a human valet, able to help us with a wide range of information-access needs.

Acknowledgments
This work was supported in part by a Rutgers Information Sciences Council pilot project as part of the Rutgers University Strategic Opportunities and Analysis initiative. Equipment for this work was provided by BellSouth Wireless Data.

References


